# Canopy vignette

Yuchao Jiang
yuchaoj@upenn.edu

September 17, 2024

This is a demo for using the `Canopy` package in R. `Canopy` is a statistical framework and computational procedure for identifying subpopulations within a tumor, determining the mutation profiles of each subpopulation, and inferring the tumor's phylogenetic history. The input to `Canopy` are variant allele frequencies of somatic single nucleotide alterations (SNAs) along with allele-specific coverage ratios between the tumor and matched normal sample for somatic copy number alterations (CNAs). These quantities can be directly taken from the output of existing software. `Canopy` provides a general mathematical framework for pooling data across samples and sites to infer the underlying parameters. For SNAs that fall within CNA regions, `Canopy` infers their temporal ordering and resolves their phase. When there are multiple evolutionary configurations consistent with the data, `Canopy` outputs all configurations along with their confidence.

Below is an example on reconstructing tumor phylogeny of a transplantable metastasis model system derived from a heterogeneous human breast cancer cell line MDA-MB-231. Cancer cells from the parental line MDA-MB-231 were engrafted into mouse hosts leading to organ-specific metastasis. Mixed cell populations (MCPs) were in vivo selected from either bone or lung metastasis and grew into phenotypically stable and metastatically competent cancer cell lines. The parental line as well as the MCP sublines were whole-exome sequenced with somatic SNAs and CNAs profiled. `Canopy` is used to infer metastatic phylogeny. Code for analysis of this dataset is broken down below with explanations and is further available here.

`Canopy`'s **webpage** is here. **Demo codes** for `Canopy` under various settings/modes can be found here. Script for dataset from the MDA231 study is attached below with step-by-step decomposition and explanation. Online **Q&A forum** for `Canopy` is available here. If you've any questions regarding the software, you can also email us at canopy_phylogeny@googlegroups.com.

## 1. Installation

R package `Canopy` is availble from `CRAN` (https://CRAN.R-project.org/package=Canopy):

```
> install.packages('Canopy')  # updated every 3-4 months
```

A devel version can be installed from GitHub ((https://github.com/yuchaojiang/Canopy):

```
> install.packages("devtools")
> library(devtools)
> install_github("yuchaojiang/Canopy/package")  # STRONGLY recommended
```

# 2. Canopy workflow

## 2.1 CNA and SNA input

The input to `Canopy` are variant allele frequencies of somatic SNAs along with allele-specific coverage ratios between the tumor and matched normal sample for somatic CNAs. For SNAs, let the matrices $R$ and $X$ be, respectively, the number of reads containing the mutant allele and the total number of reads for each locus across all samples. The ratio $R/X$ is the proportion of reads supporting the mutant allele, known as the variant allele frequency. For CNAs, `Canopy` directly takes output from allele-specific copy number estimation softwares, such as FALCON-X or Sequenza. These outputs are in the form of estimated major and minor copy number ratios, respectively denoted by $W^M$ and $W^m$, with their corresponding standard errors $\epsilon^M$ and $\epsilon^m$. Matrix $Y$ specifies whether SNAs are affected by CNAs; matrix $C$ specifies whether CNA regions harbor specific CNAs (this input is only needed if overlapping CNA events are observed).

How to generate CNA and SNA data input is futher discussed here. How to select *informative* SNA and CNA input is discussed here.

Below is demo data input from project MDA231 (first case study in our paper).

```
> library(Canopy)
> data("MDA231")
> projectname = MDA231$projectname ## name of project
> R = MDA231$R; R ## mutant allele read depth (for SNAs)

      MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
BRAF           155           59          136                  77           49
KRAS            44           21           54                  19           17
ALPK2           37           17           28                  10            7
RYR1            44            0           26                   0            0

> X = MDA231$X; X ## total depth (for SNAs)

      MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
BRAF           157          111          177                 146           71
KRAS            44           30           64                  42           27
ALPK2           63           17           65                  24            7
RYR1           107           56          165                  55           43

> WM = MDA231$WM; WM ## observed major copy number (for CNA regions)

      MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
chr7         2.998        2.002        2.603               2.000        2.001
chr12        1.998        1.998        1.603               1.001        1.999
chr18        1.000        2.992        1.000               1.002        2.996
chr19        2.000        2.000        2.000               2.000        2.000

> Wm = MDA231$Wm; Wm ## observed minor copy number (for CNA regions)

      MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
chr7         0.002        0.998        0.397               1.000        0.999
chr12        0.002        0.998        0.397               1.000        0.999
chr18        1.000        0.004        1.000               0.999        0.002
chr19        1.000        1.000        1.000               1.000        1.000
```

```
> epsilonM = MDA231$epsilonM ## standard deviation of WM, pre-fixed here
> epsilonm = MDA231$epsilonm ## standard deviation of Wm, pre-fixed here
> ## Matrix C specifices whether CNA regions harbor specific CNAs
> ## only needed if overlapping CNAs are observed, specifying which CNAs overlap
> C = MDA231$C; C

      chr7_1 chr7_2 chr12_1 chr12_2 chr18 chr19
chr7       1      1       0       0     0     0
chr12      0      0       1       1     0     0
chr18      0      0       0       0     1     0
chr19      0      0       0       0     0     1

> Y = MDA231$Y; Y ## whether SNAs are affected by CNAs

      non-cna_region chr7 chr12 chr18 chr19
BRAF               0    1     0     0     0
KRAS               0    0     1     0     0
ALPK2              0    0     0     1     0
RYR1               0    0     0     0     1
```

## 2.2 Binomial clustering of SNAs

A multivariate binomial mixture clustering step can be applied to the SNAs before MCMC sampling. We show in our paper via simulations that this pre-clustering method helps the Markov chain converge faster with smaller estimation error (especially when mutations show clear cluster patterns by visualization). This clustering step can also remove likely false positives before feeding the mutations to the MCMC algorithm.

Below is a toy example, where three bulk tumor samples were in silico simulated from a tree of 4 clones/leaves. The 5 tree segments (excluding the leftmost branch, which corresponds to the normal clone) separate 200 mutations into 5 mutation clusters. More detailed demo codes for clustering can be found here. Detailed methods can be found in the supplements of our paper under section Binomial mixture clustering. BIC is used for model selection. 2D (two longitudinal/spatial samples) or 3D (three samples) plots are generated for visualization.

```
> library(Canopy)
> data(toy3)
> R=toy3$R; X=toy3$X # 200 mutations across 3 samples
> num_cluster=2:9 # Range of number of clusters to run
> num_run=10 # How many EM runs per clustering step for each mutation cluster wave
> canopy.cluster=canopy.cluster(R = R,
+                               X = X,
+                               num_cluster = num_cluster,
+                               num_run = num_run)

Running EM with 2 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 3 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 4 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 5 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 6 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 7 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 8 clusters...        1   2   3   4   5   6   7   8   9   10
Running EM with 9 clusters...        1   2   3   4   5   6   7   8   9   10
```

```
> bic_output=canopy.cluster$bic_output # BIC for model selection (# of clusters)
> Mu=canopy.cluster$Mu # VAF centroid for each cluster
> Tau=canopy.cluster$Tau  # Prior for mutation cluster, with a K+1 component
> sna_cluster=canopy.cluster$sna_cluster # cluster identity for each mutation
```

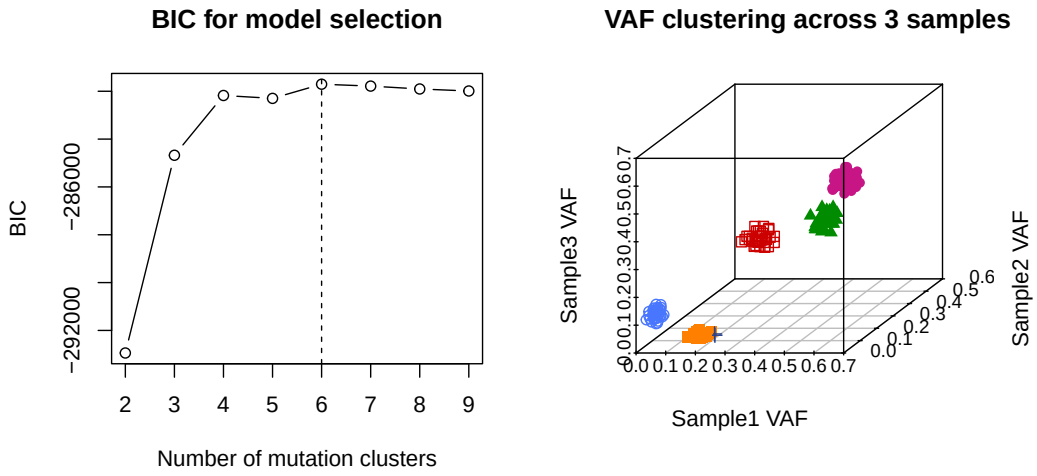


Figure 1: BIC to select the optimal number of clusters and Binomial mixture clustering results.

Below is real dataset from Ding et al. (Nature 2012), where a leukemia patient was sequenced at two timepoints – primary tumor (sample 1) and relapse genome (sample 2). The real dataset is noisier and can potentially contain false positives for somatic mutations. We thus include in the mixture a multivariate uniform component on the unit interval, which corresponds to mutations that have high standard errors during sequencing or that are likely to be false positives. The code and SNA input for this dataset can be found here.

```
> library(Canopy)
> data(AML43)
> R=AML43$R; X=AML43$X
> num_cluster=4 # Range of number of clusters to run
> num_run=6 # How many EM runs per clustering step for each mutation cluster wave
> Tau_Kplus1=0.05 # Pre-specified proportion of noise component
> Mu.init=cbind(c(0.01,0.15,0.25,0.45),
+               c(0.2,0.2,0.01,0.2)) # Initial value for centroid
> canopy.cluster=canopy.cluster(R = R,
+                               X = X,
+                               num_cluster = num_cluster,
+                               num_run = num_run,
+                               Mu.init = Mu.init,
+                               Tau_Kplus1=Tau_Kplus1)

Running EM with 4 clusters...        1   2   3   4   5   6

> Mu=canopy.cluster$Mu # VAF centroid for each cluster
> Tau=canopy.cluster$Tau  # Prior for mutation cluster, with a K+1 component
```

```
> sna_cluster=canopy.cluster$sna_cluster # cluster identity for each mutation
> R.qc=R[sna_cluster<=4,] # exclude mutations in the noise cluster
> X.qc=X[sna_cluster<=4,]
> sna_cluster.qc=sna_cluster[sna_cluster<=4]
> R.cluster=round(Mu*100)  # Generate pseudo-SNAs correponding to each cluster.
> X.cluster=pmax(R.cluster,100)   # Total depth is set at 100 but can be obtained as median instead
> rownames(R.cluster)=rownames(X.cluster)=paste('SNA.cluster',1:4,sep='')
```

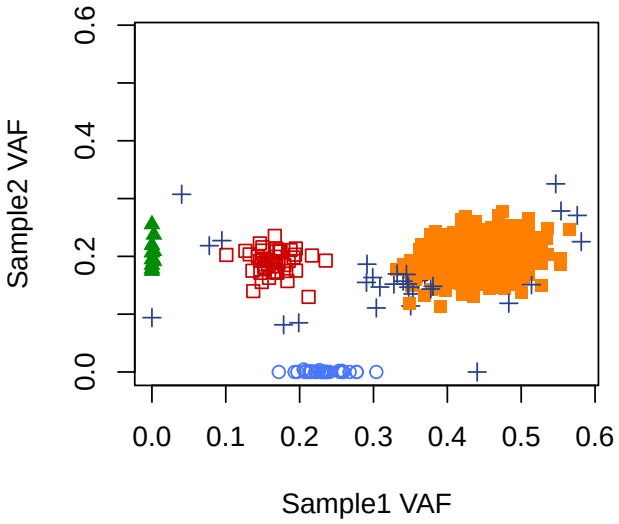


Figure 2: Binomial mixture clustering on real dataset of primary tumor and relapse genome.

## 2.3 Phylogenetic tree (unknown)

Each sampled tree is modeled as a list by `Canopy`. Below are the tree elements of the most likely tree from the project MDA231 (first case study in the paper). The most likely tree is obtained from the posterior distribution in the tree space from the MCMC sampling (detailed in section 2.3). How to visualize/plot the sampled trees is discussed later.

```
> data('MDA231_tree')
> MDA231_tree$Z # Z matrix specifies the position of the SNAs along the tree branch
```

| | clone1 | clone2 | clone3 | clone4 |
|---|---|---|---|---|
| BRAF | 0 | 1 | 1 | 1 |
| KRAS | 0 | 1 | 1 | 1 |
| ALPK2 | 0 | 1 | 1 | 1 |
| RYR1 | 0 | 0 | 1 | 0 |

```
> MDA231_tree$cna.copy # major and minor copy number (interger values) for each CNA
```

| | chr7_LOH | chr7_dup | chr12_dup | chr12_LOH | chr18_LOH | chr19_dup |
|---|---|---|---|---|---|---|
| major_copy | 3 | 2 | 2 | 2 | 3 | 2 |
| minor_copy | 0 | 1 | 1 | 0 | 0 | 1 |

```
> MDA231_tree$CM # Major copy per clone for each CNA

      clone1 clone2 clone3 clone4
chr7       1      2      3      2
chr12      1      1      2      2
chr18      1      1      1      3
chr19      1      2      2      2

> MDA231_tree$Cm # Minor copy per clone for each CNA

      clone1 clone2 clone3 clone4
chr7       1      1      0      1
chr12      1      1      0      1
chr18      1      1      1      0
chr19      1      1      1      1

> MDA231_tree$Q # whether an SNA precedes a CNA

      chr7_LOH chr7_dup chr12_dup chr12_LOH chr18_LOH chr19_dup
BRAF         1        1         0         0         0         0
KRAS         0        0         1         1         0         0
ALPK2        0        0         0         0         1         0
RYR1         0        0         0         0         0         0

> MDA231_tree$H # if an SNA precedes a CNA, whether it resides in the major copy

      chr7_LOH chr7_dup chr12_dup chr12_LOH chr18_LOH chr19_dup
BRAF         1        1         0         0         0         0
KRAS         0        0         1         1         0         0
ALPK2        0        0         0         0         1         0
RYR1         0        0         0         0         0         0

> MDA231_tree$P # clonal compostion for each sample

       MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
clone1        0.000        0.000        0.000                0.000        0.000
clone2        0.006        0.000        0.399                0.992        0.003
clone3        0.991        0.001        0.600                0.001        0.001
clone4        0.003        0.999        0.001                0.007        0.996

> MDA231_tree$VAF # VAF based on current tree structure

      MCP1833_bone MCP1834_lung MCP2287_bone MDA-MB-231_parental MCP3481_lung
BRAF         0.997        0.667        0.867                0.667        0.667
KRAS         0.996        0.667        0.800                0.502        0.667
ALPK2        0.502        1.000        0.501                0.505        0.999
RYR1         0.330        0.000        0.200                0.000        0.000
```

## 2.4 MCMC sampling

`Canopy` samples in subtree space with varying number of subclones (denoted as $K$) by a Markov chain Monte Carlo (MCMC) method. A plot of posterior likelihood (pdf format) will be generated for each subtree space and we recommend users to refer to the plot as a sanity check for sampling convergence and to choose the number of burn-ins and

thinning accordingly. Note that this step can be time-consuming, especially with larger number of chains (`numchain` specifies the number of chains with random initiations, a larger value of which is in favor of not getting stuck in local optima) and longer chains (`simrun` specifies number of iterations per chain). MCMC sampling is the most computationally heavy step in `Canopy`. It is recommended that jobs are run in parallel on high-performance cluster.

There are four modes of MCMC sampling embedded in Canopy: (1) `canopy.sample` which takes both SNA and CNA as input by default; (2) `canopy.sample.nocna` for cases where there is no CNA input; (3) `canopy.sample.cluster` for cases where SNAs are pre-clustered by the Binomial mixture EM algorithm; (4) `canopy.sample.cluster.nocna` for cases where there is no CNA input and SNAs are pre-clustered by the Binomial mixture EM algorithm. More details can be found here.

Below is sampling code for the MDA231 dataset where both SNA and CNA are used as input.

```
> K = 3:6 # number of subclones
> numchain = 20 # number of chains with random initiations
> sampchain = canopy.sample(R = R, X = X, WM = WM, Wm = Wm, epsilonM = epsilonM,
+             epsilonm = epsilonm, C = C, Y = Y, K = K, numchain = numchain,
+             max.simrun = 50000, min.simrun = 10000,
+             writeskip = 200, projectname = projectname, cell.line = TRUE,
+             plot.likelihood = TRUE)
> save.image(file = paste(projectname, '_postmcmc_image.rda',sep=''),
+            compress = 'xz')

> length(sampchain) ## number of subtree spaces (K=3:6)

[1] 4

> length(sampchain[[which(K==4)]]) ## number of chains for subtree space with 4 subclones

[1] 20

> length(sampchain[[which(K==4)]][[1]]) ## number of posterior trees in each chain

[1] 250
```

## 2.5 BIC for model selection

`Canopy` uses BIC as a model selection criterion to determine to optimal number of subclones.

```
> burnin = 100
> thin = 5 # If there is error in the bic and canopy.post step below, make sure
>          # burnin and thinning parameters are wisely selected so that there are
>          # posterior trees left.
> bic = canopy.BIC(sampchain = sampchain, projectname = projectname, K = K,
+                numchain = numchain, burnin = burnin, thin = thin, pdf = FALSE)

k = 3 : mean likelihood -17243.57 ; BIC -34650.33 .
k = 4 : mean likelihood -548.2876 ; BIC -1305.67 .
k = 5 : mean likelihood -611.4774 ; BIC -1477.948 .
k = 6 : mean likelihood -586.8295 ; BIC -1474.551 .

> optK = K[which.max(bic)]
```

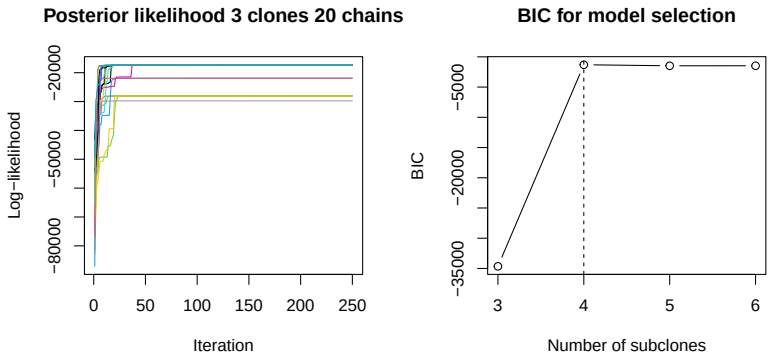


Figure 3: Posterior likelihood of MCMC (chains are colored differently) and BIC as a model selection method.

## 2.6 Posterior evaluation of sampled trees

`Canopy` then runs a posterior evaluation of all sampled trees by MCMC. If modes of posterior probabilities (second column of `config.summary`) aren't obvious, check if the algorithm has converged (and run sampling longer if not).

```
> post = canopy.post(sampchain = sampchain, projectname = projectname, K = K,
+                    numchain = numchain, burnin = burnin, thin = thin, optK = optK,
+                    C = C, post.config.cutoff = 0.05)
> samptreethin = post[[1]]   # list of all post-burnin and thinning trees
> samptreethin.lik = post[[2]]   # likelihoods of trees in samptree
> config = post[[3]] # configuration for each posterior tree
> config.summary = post[[4]] # configuration summary
> print(config.summary)
```

```
     Configuration Post_prob Mean_post_lik
[1,]             1     0.562       -547.55
[2,]             2     0.066       -549.68
[3,]             3     0.073       -548.62
[4,]             4     0.146       -547.68
[5,]             5     0.153       -547.80
```

```
> # first column: tree configuration
> # second column: posterior configuration probability in the entire tree space
> # third column: posterior configuration likelihood in the subtree space
```

## 2.7 Tree output and plotting

One can then use `Canopy` to output and plot the most likely tree (i.e., tree with the highest posterior likelihood). Mutations, clonal frequencies, and tree topology, etc., of the tree are obtained from the posterior distributions of subtree space with trees having the same configuration. In our MDA231 example, the most likely tree is the tree with configuration 3.

**Note**: A separate txt file can be generated (with txt=TRUE and txt.name='*.txt') if the figure legend of mutational profiles (texts below the phylogenetic tree) in the plot is too long to be fitted entirely.

```
> config.i = config.summary[which.max(config.summary[,3]),1]
> cat('Configuration', config.i, 'has the highest posterior likelihood!\n')

Configuration 1 has the highest posterior likelihood!

> # plot the most likely tree in the posterior tree space
> output.tree = canopy.output(post, config.i, C)
> canopy.plottree(output.tree)
> # plot the tree with configuration 1 in the posterior tree space
> output.tree = canopy.output(post, 1, C)
> canopy.plottree(output.tree, pdf=TRUE, pdf.name =
+                     paste(projectname,'_first_config.pdf',sep=''))
```

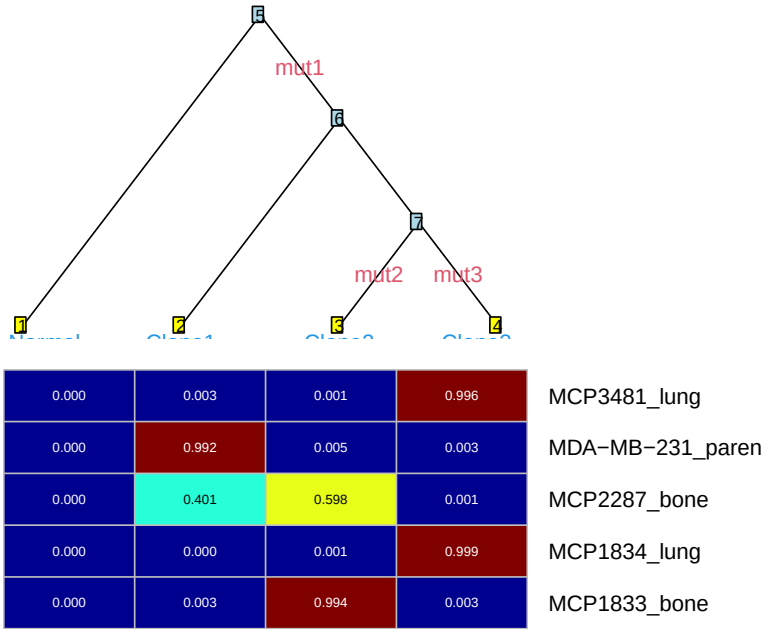


Figure 4: Most likely tree by Canopy for project MDA231.

## 3. Try it yourself

Now try Canopy yourself using the simulated toy dataset below! Note that no overlapping CNAs are used as input and thus matrix $C$ doesn't need to be specified.

```
> library(Canopy)
> data(toy)
> projectname = 'toy'
> R = toy$R; X = toy$X; WM = toy$WM; Wm = toy$Wm
> epsilonM = toy$epsilonM; epsilonm = toy$epsilonm; Y = toy$Y
> K = 3:6; numchain = 10
> sampchain = canopy.sample(R = R, X = X, WM = WM, Wm = Wm, epsilonM = epsilonM,
+                           epsilonm = epsilonm, C = NULL, Y = Y, K = K,
+                           numchain = numchain, simrun = 50000, writeskip = 200,
+                           projectname = projectname, cell.line = FALSE,
+                           plot.likelihood = TRUE)
```

The most likely tree is shown below. There should be only one tree configuration from the posterior tree space. The code for this toy dataset analysis can be found here.

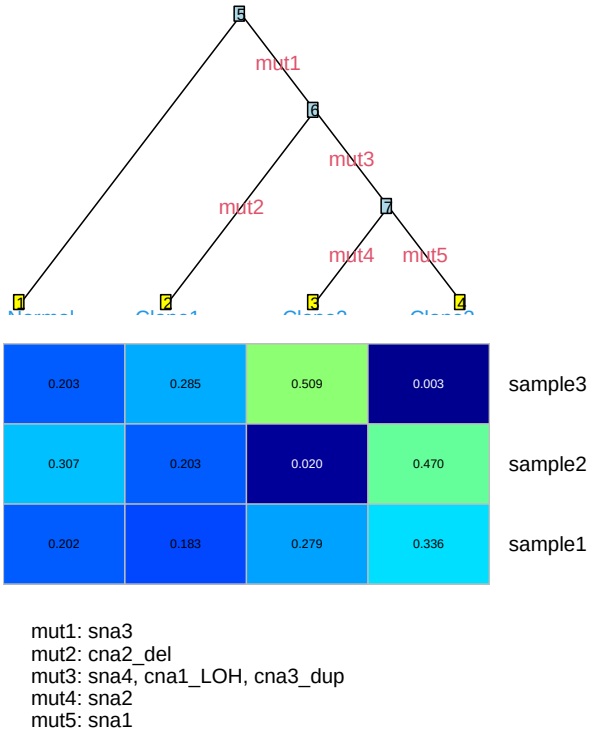


Figure 5: Most likely tree by Canopy for simulated toy dataset.

The second toy example has a different tree topology. Feel free to try Canopy on this dataset too! There should be also just one tree configuration as is shown below from the posterior tree space. The code for this toy dataset analysis can be found here.

```
> library(Canopy)
> data(toy2)
> projectname = 'toy2'
> R = toy2$R; X = toy2$X; WM = toy2$WM; Wm = toy2$Wm
> epsilonM = toy2$epsilonM; epsilonm = toy2$epsilonm; Y = toy2$Y
> true.tree = toy2$true.tree  # true underlying tree
> K = 3:6; numchain = 15
> sampchain = canopy.sample(R = R, X = X, WM = WM, Wm = Wm, epsilonM = epsilonM,
+                           epsilonm = epsilonm, C = NULL, Y = Y, K = K,
+                           numchain = numchain, max.simrun = 100000,
+                           min.simrun = 10000, writeskip = 200,
+                           projectname = projectname, cell.line = FALSE,
+                           plot.likelihood = TRUE)
```

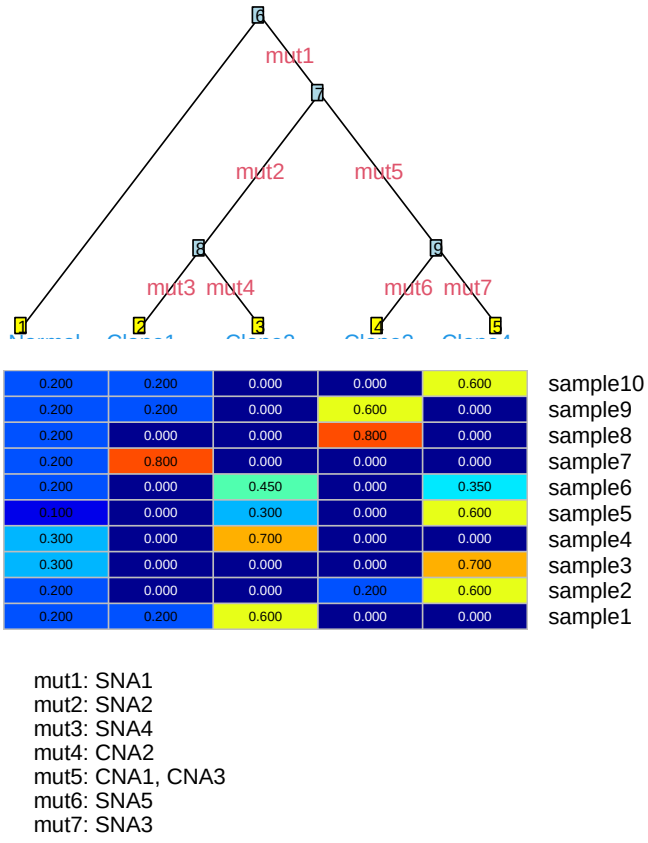


Figure 6: Most likely tree by Canopy for simulated toy dataset 2.

The third toy example consists of three bulk tumor samples, in silico simulated from a tree of 4 clones/leaves. The 5 tree segments (excluding the leftmost branch, which corresponds to the normal clone) separate 200 mutations into 5 mutation clusters. The SNA clustering details are outlined in section "Binomial clustering of SNAs". The code for this toy dataset analysis is briefly attached below. Code in more details with visualization and posterior analysis can be found here.

```
> library(Canopy)
> data(toy3)
> R=toy3$R; X=toy3$X
> num_cluster=2:9 # Range of number of clusters to run
> num_run=10 # How many EM runs per clustering step for each mutation cluster wave
> canopy.cluster=canopy.cluster(R = R,
+                               X = X,
+                               num_cluster = num_cluster,
+                               num_run = num_run)
> bic_output=canopy.cluster$bic_output # BIC for model selection (# of clusters)
> Mu=canopy.cluster$Mu # VAF centroid for each cluster
> Tau=canopy.cluster$Tau  # Prior for mutation cluster, with a K+1 component
> sna_cluster=canopy.cluster$sna_cluster # cluster identity for each mutation
> projectname='toy3'
> K = 3:5 # number of subclones
> numchain = 15 # number of chains with random initiations
> sampchain = canopy.sample.cluster.nocna(R = R, X = X, sna_cluster = sna_cluster,
+                                         K = K, numchain = numchain,
+                                         max.simrun = 100000, min.simrun = 20000,
+                                         writeskip = 200, projectname = projectname,
+                                         cell.line = FALSE, plot.likelihood = TRUE)
> save.image(file = paste(projectname, '_postmcmc_image.rda',sep=''),
+            compress = 'xz')
```

# 4. Citation

Assessing intra-tumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing, Yuchao Jiang, Yu Qiu, Andy J Minn, Nancy R zhang, Proceedings of the National Academy of Sciences, 2016. (html, pdf)

# 5. Session information:

Output of sessionInfo on the system on which this document was compiled:

- R version 4.4.1 (2024-06-14), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 24.04.1 LTS`

- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so` ; LAPACK version3.12.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Canopy 1.3.0, ape 5.8, fields 16.2, pheatmap 1.0.12, scatterplot3d 0.3-44, spam 2.10-0, viridisLite 0.4.2
- Loaded via a namespace (and not attached): R6 2.5.1, RColorBrewer 1.1-3, Rcpp 1.0.13, buildtools 1.0.0, cli 3.6.3, colorspace 2.1-1, compiler 4.4.1, digest 0.6.37, dotCall64 1.1-1, glue 1.7.0, grid 4.4.1, gtable 0.3.5, knitr 1.48, lattice 0.22-6, lifecycle 1.0.4, maketools 1.3.0, maps 3.4.2, munsell 0.5.1, nlme 3.1-166, parallel 4.4.1, rlang 1.1.4, scales 1.3.0, sys 3.4.2, tools 4.4.1, xfun 0.47