Natalie Andersson 230623 v1

How to build MP and ML trees from a segment file

1. Install R and R studio

Preferred version: R 4.2.2

https://cran.r-project.org/

https://posit.co/download/rstudio-desktop/

2. Install the dependencies

```
install.packages(c("readxl","xlsx","stringr","ape","phangorn","ggplot2",
                   "ggtree","ggimage","dplyr","RColorBrewer","ggridges","cowplot","dbscan"))
```

Or install them one by one:

```
install.packages("packagename")
```

3. Load the dependencies

```
library("readxl") #Needed to load the data from the xlsx file.
library("xlsx") #Needed to save matrices into xlsx-files.
library("stringr") #Needed for using the function "word".
library("ape")
library("phangorn") #Needed to transform the EM into phyDat and make trees.
library("ggplot2") #Needed to visualize the trees.
library("ggtree")
library("ggimage") #Needed to insert the pies in the tree.
library("dplyr") #Needed for the distinct function in pie.it.
library("RColorBrewer") #Needed to add the colored pie charts.
library("ggridges") #Used to plot the distribution.
library("cowplot")
library("dbscan") #Clustering
```

3. Install and load DEVOLUTION

```
install.packages("devtools") #Needed to install packages from Github.

library(devtools)
devtools::install_github('NatalieKAndersson/DEVOLUTION')
library("DEVOLUTION") #Loading it.
```

4. Prepare your input file

Make sure that your input file has each of these columns. The first row in the excel sheet should have these names.

```
> head(datasegment)
  Tumor ID Samples Chr    Start       End Med LogR VAF (TRS) Type    Method Cytoband/ Gene Clone size (%)
1   Tumor1     ALL   1        0 247249719       NA        NA GAIN SNP Array          WHOLE            100
2   Tumor1     ALL   3    63411 197852564     0.35        NA GAIN SNP array          WHOLE            100
3   Tumor1     ALL   4 40421567  41888869    -0.65        NA LOSS SNP array        4p14p13            100
4   Tumor1     ALL   6   156974 170919481      0.1        NA GAIN SNP array          WHOLE            100
5   Tumor1      B1   2    21494  45575110       NA        NA GAIN SNP array        2p25p22            100
6   Tumor1      B1   4    69404  43444897       NA        NA LOSS SNP array        4p16p15             80
```

Now we are ready to start the analyses!

5. Set your working directory (the path to where your files are)

setwd("~/yourpath")

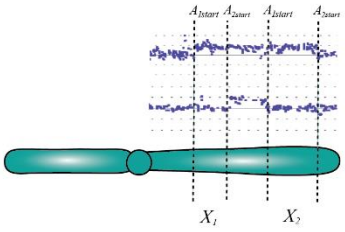
6. Load your segment file

data <- load_matrix(filename="Segment_example.xlsx",sheetname ="Example_tumors")

7. Set the input parameters

datatypes <- "All"
event_co <- 10*10^6
root <- "Normal"
x <- “Tumor1” #The name of the tumor you want to analyze.
datasegment <- splitdata(data,name=x) #Extracting the beginning and end position of each sample in the segment file. #Declare which tumor you want to analyze. Specified by the first column in your data set.

- **datatypes:** These are your data types such as SNP-array, TDS, WGS, WES etc that you want to include in the analysis. If you write "All", all events will be included in the input file. If you exclusively want to analyze particular alterations, you can set an input vector indicating which events should be kept for analysis. The command "#c(unique(test[,9]))" gives you the unique methods in your dataset.
- **event_co:** The cutoff for the start and end positions of the events in your segment file. It is used to determine if two copy number alterations with a little different start and end positions, actually are the same event and differ a bit in their size due to noise.



$$X_1 = \left\| A_{1_{start}} - A_{2_{start}} \right\| \leq co_{ev}$$

$$X_2 = \left\| A_{1_{end}} - A_{2_{end}} \right\| \leq co_{ev}$$

- **root:** In what cell you want to root your tree. Choose between "Normal" or "Stem" (this is just a cell having the genetic alterations present in the stem of the tree).

8. Run DEVOLUTION

EM <- DEVOLUTION(**datasegment**,event_co,datatypes=c("All"), eps = 0.5, names="numbers",preclustered=FALSE) #Creating an event matrix based on the segment file chosen.
EM_dev <- subclones(EM,file_samples_subclones,root = "Normal",possible_mothers,cutoff=30,simplified=FALSE,names="numbers")

What does the input names mean?

Function DEVOLUTION - Inputs

- **Datasegment: The input data file.**
- event_co: The cutoff you have chosen for the start and end position for an event to be considered the same.
- Datatypes: The input data file may contain data from both e.g SNP-array and WGS etc. Writing "All" includes all alterations present in the input datasegment file. You can although choose here to only include SNP-array data for example, in which case you write "SNP-array". Make sure the name corresponds to the name in column 10 in the input data file i.e. the "Method" column. You can include both SNP-array and WGS by writing c("SNP-array","WGS").
- eps: The epsilon for the dbscan clustering. Choose according to the kNNdistplot that will appear when running the code. Should be at where the elbow of the curve is. The default is 0.5. Usually works fine.
- names: The subclone names. Declaring "subclone" gives names such as "Subclone_ A", "Subclone_ B" etc. Declaring "letters" will give subclonenames such as "A", "B" etc. which might give a more clear appearance of the phylogeny. Declaring "numbers" will give subclone names such as "1", "2" etc. which might be good if you have a complex dataset with a lot of subclones.

Function subclones - Inputs:

- EM: De output from DEVOLUTION.
- file_samples_subclones: More or less the input data file together with the clustering. No need to change this. It is part of the global environment after running DEVOLUTION and has this name.
- root: Choose which cell the tree should be rooted in.
- possible_mothers: It is part of the global environment after running DEVOLUTION. Matrix containing each cluster and in which clusters it can be nested.
- cutoff: If you want to access alternative solutions for the phylogeny. Putting 30 means that all clusters > 30 % in size across all samples will be reshuffled to produce a new phylogeny, if it is possible.
- names: The subclone names. Letters gives names such as "Subclone_ A", "Subclone_ B" etc. Putting "numbers" will give subclone names such as "1", "2" etc. which might be good if you have a complex dataset with a lot of subclones.

9. Build the trees

```
#Visualizing the trees without pies and saving them
type <- "nocol"
EM_phy <- phydatevent(EM_dev[[1]]) #Transforming the EM to phyDat format.

#Maximum parsimony.
EM_mptree <- mp_tree(EM_phy,root) #Constructing the maximum parsimony tree.
limitmp <- xlim(c(0, 20)) #Here you can determine the limits for the graph for mp. 20

#Maximum likelihood
dm_h <- dist.hamming(EM_phy)
starting_tree <- NJ(dm_h)
starting_tree <- root(starting_tree, outgroup = root, resolve.root = TRUE)
```

```
Lf <- pml(starting_tree, EM_phy)
EM_mltree <- optim.pml(Lf, model = "SYM", optEdge = TRUE)
limitml <- xlim(c(0, 2)) #Here you can determine the limits for the graph for ml. 1.5

Treemp <- MP_treeplot(EM_mptree,limitmp,col = type) #Illustrating the maximum parsimony tree.
Treeml <- ML_treeplot(EM_mltree,limitml,col = type) #Illustrating the maximum likelihood tree.
#ggsave(Treemp,filename="My_tree_mp.png",width=10,height=10)


#Add pie charts to the tree.
s <- 10 #Size of the saved image.
coltype <- "col" #Choose how you want your pies. nocol = Just red pie charts with a biopsy name above. col = colored pies. custom = create your own color scheme.
samples <- as.vector(unique(datasegment[datasegment[,2]!="ALL",2])) #Or just write it.
pieData <- make_pie(EM_dev[[2]],root,samples,type=coltype,custom_col = FALSE) #Creates the pie charts.

pietree_mp <- pie_it(Treemp,pieData,offset=1,size=0.15,col=coltype) #Adds pie charts to the tree. 0.21. Used 0.17 lately.
pietree_ml <- pie_it(Treeml,pieData,offset=1,size=0.15,col=coltype) #Adds pie charts to the tree. 0.21. Used 0.17 lately.

ggsave(pietree_mp,filename=paste(x,"_mp",".png",sep=""),width = s,height = s) #Saving the image.
ggsave(pietree_mp,filename=paste(x,"_mp",".pdf",sep=""),width = s,height = s) #Saving the image.

ggsave(pietree_ml,filename=paste(x,"_mp",".png",sep=""),width = s,height = s) #Saving the image.
ggsave(pietree_ml,filename=paste(x,"_mp",".pdf",sep=""),width = s,height = s) #Saving the image.
```

Now you should have a png and pdf version of the phylogeny saved in your working directory.

They can both be opened in adobe illustrator. The PDF-image is already in vector format and the pies etc can be moved around. You can make the png-version into a vector object by pressing “Fönster” and then “Bildkalkering” and run that. Then you go to “Objekt”, then “Bildkalkering” and then “Expandera”. In this way you can get the components of the tree if you want to change colors or something.

10. Save the DEVOLUTION output excel document

```
write.xlsx(as.data.frame(t(EM_dev[[1]])),"DEVOLUTION.xlsx",sheetName="Event matrix")
write.xlsx(Clustering,append = TRUE, "DEVOLUTION.xlsx",sheetName = "Clustering")
write.xlsx(as.data.frame(t(EM)),append = TRUE,"DEVOLUTION.xlsx",sheetName="Event matrix samples")
write.xlsx(as.data.frame(EM_dev[[3]]),append = TRUE,"DEVOLUTION.xlsx",sheetName="Overview")
```

11. For even more details, please see the github page:

Natalie Andersson 230623 v1

https://github.com/NatalieKAndersson/DEVOLUTION

How to build a MMP tree

The input to the MMP-function is the output excel file from DEVOLUTION.

1. Install MMP

```
library(devtools)
devtools::install_github('NatalieKAndersson/MMP')
library("MMP")
```

2. Run the MMP function

```
setwd("~/yourpath") #Should be where the DEVOLUTION-file is.

MMP_tree <- MMP(file="DEVOLUTION.xlsx",tumorname=x) #The output is the tree object.
```

- **File:** The filename of the output excel file from DEVOLUTION. This contains the event matrix, clustering and pie sizes. The algorithm uses the pie sizes etc. to minimize contradictions where “a larger pie comes after a smaller one” in the tree.
  - **EM:** The event matrix. All columns in the figure below except the last column.
    - **Type:** Whether it is a point mutation or a copy number alteration. Last column in the matrix below.

	Stem	A	B	C	Normal	Type
3+	1	1	1	1	0	W
5+	0	1	1	0	0	W
7+	0	0	0	1	0	W
8+	0	0	1	1	0	W

- **Pies:** The proportion of cells in each sample that have a particular genotype.

Stem	P1	P2
100	100	100
A	P1	P2
80	80	0
B	P2	0
70	70	0
C	P1	P2
60	60	60

- **Overview:** The mutated clone fraction for each alteration across samples.

	P1	P2
3+	100	100
5+	80	0
7+	0	70
8+	60	60

- **Tumorname:** The name of the tumor you are analyzing. In the function this is only used when the output file is saved in the end of the algorithm.
- The output of the MMP-function is the phylogenetic tree in the shape of a so called “phylo object” which can be plotted with ggplot.

2. Plot the tree – Same functions as in DEVOLUTION

```
#Plot the MMP-tree using ggplot
limitmmp <- xlim(c(0,15))
Treemmp <- ggplot(MMP_tree) + geom_tree() + geom_tiplab(size=4, color = "black") #vjust -0.7.+ geom_treescale(width = 1)
Treemmp <- Treemmp +  theme_tree() + limitmmp+theme(plot.title = element_text(hjust = 0.5, size = (14), color = "black"))
print(Treemmp)


#Add the pies – The same functions as in DEVOLUTION
pietree_mmp <- pie_it(Treemmp,pieData,offset=1,size=0.15,col=coltype) #Adds pie charts to the tree. 0.21. Used 0.17 lately.
#Save the trees in png- and pdf-format
ggsave(pietree_mp,filename=paste(x,"_mmp",".png",sep=""),width = s,height = s) #Saving the image.
ggsave(pietree_mp,filename=paste(x,"_mmp",".pdf",sep=""),width = s,height = s) #Saving the image.
```