Using scripts

The scripts are best used in conjunction with Rstudio. Experience users can use the script directly from the command line.

For each diagram type, there exists multiple scripts. To use a script, first load it into Rstudio, then set the working directory and finally create diagrams.

Loading a script: Load and execute the r script in Rstudio: `Code > Source File...`

Loading the script does not create a diagram. It defines a function, that has the same name as the script. This function creates the diagrams. The function should visible in the *Environment* tab of Rstudio.

Setting the working directory: Rstudio has a working directory. All relative paths, to input files or source files, are given relative to this working directory. Set the working directory in Rstudio: `Session > Set Working Directory > Choose Directory...`

Setting the working directory to the folder that contains the input csv files is advantageous. First, only the file names, like `mid_3.csv`, and not longer path, `path\to\mid_23.csv` have to passed to the function. Second, all files created by the functions, output diagrams and parameter files, are contained inside the working directory.

When switching to another data set, the working directory should also be switched.

Creating a diagram: To create a diagram, just call the function using the *Console* tab in Rstudio.

The parameter file

Every function uses a parameter file, that has the same name as the function. It stores all necessary parameters to recreate the diagram. The file is stored in the specified `saveFolder` or same folder as the first file passed to the function, if `saveFolder` was not specified. If the parameter file does not already exists, a new one is created.

Before calling a function, make sure that its parameter file is not opened by any other program. An already opened parameter file stops the function from storing the parameters of any newly created diagrams.

Adapting scripts

To adapt or change a script we can either open the script in an editor, or, if the script was already loaded in Rstudio, edit it using Rstudio.

In the first case, after saving the changes to the script, reload it as described above. Editing the script in Rstudio, double click on the corresponding function in the *Environment* tab. This opens up a new window in Rstudio, displaying the contents of the script. After edition the script, save the changes using the *Save current document* button. Then rerun the script using the *run* button.

Note that both methods overwrite the existing script. So make sure to have a backup or to save the changes to a new script.

Adapting to a new csv format

All functions depend on a set format of the input csv files. When this format changes we can easily adapt the scripts.

All functions define a sub-function `readData(file)` that takes the path to an input file and returns a data frame with two columns. Each row corresponds to one clone. `id` contains a string that is unique for the clone over all input files. `Count` contains the absolute number of reads of the clone.

Input file:

Count	CDR3 nucleotide sequence	CDR3 amino acid sequence	V segments	J segments	Percentage
10	[sequence]	[amino sequence]	[v-segment]	[j-segment]	0.1

Output data frame:

Count	id
10	[sequence][v-segment][j-segment]

Code:

```
readData = function(file) {
    data = read.table(file, header = TRUE, sep = "\t", colClasses = c("integer",
"character", "character", "character", "character", "character", "numeric"))
    data = cbind(data, data.frame(id =
paste(data$CDR3.nucleotide.sequence,data$V.segments, data$J.segments, sep = "")))
    subset(data, select = -c(CDR3.nucleotide.sequence, CDR3.amino.acid.sequence,
V.segments, J.segments, Percentage))
}
```

The first line reads the csv table and creates a data frame. The second line merges specific columns to create the `id` column. It can also rename a column to the `Count` column. The third line removes all columns from the data frame that are not `id` or `Count`.

Circos plots

Circos plots use the the `circlize` package. Install the package in Rstudio using the install button in the *Package* tab. There are circos functions for 1, 2, 3, 4, and 6 files.

```
circos1(file1,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    countColors = c("#FFFFFFFF", "#0000FFFF"))

circos2(file1, file2,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    sort = FALSE,
    countColors = c("#FFFFFFFF", "#0000FFFF"),
    linkColors = c("FF000000"))

circos3(file1, file2, file3,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    sort = FALSE,
    countColors = c("#FFFFFFFF", "#0000FFFF"),
    linkColors = c("FF000000", "FF000000", "FF000000"),
    showLinks = c(TRUE, TRUE, TRUE))

circos4(file1, file2, file3, file4,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    sort = FALSE,
    countColors = c("#FFFFFFFF", "#0000FFFF"),
    linkColors = c("FF000000", "FF000000", "FF000000", "FF000000", "FF000000",
"FF000000"),
    showLinks = c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE))

circos6(file1, file2, file3, file4, file5, file6,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    sort = FALSE,
    countColors = c("#FFFFFFFF", "#0000FFFF"),
    linkColors = c("FF000000"))
```

Here we show the options for the 3 file variant.

In `circos3` there are three different types of links:

1. Between `file1` and `file2`
2. Between `file1` and `file3`
3. Between `file2` and `file3`

And there are seven different share statuses:

1. In `file1`
2. In `file2`
3. In `file3`
4. In `file1` and `file2`
5. In `file1` and `file3`
6. In `file2` and `file3`
7. In `file1`, `file2` and `file3`

`circos3` has the following parameters:

- `file1`, `file2`, `file3`: Specifies the path to the three input csv files.
- `fileAliases`: A three element vector specifying alternative names for the input files that are used as lables in the diagram.
- `saveFolder`: Specifies the path to the folder where the diagram and parameter file are saved.
- `cutoff`: Is a value between 0 and 1.0. For each file, all reads not in the top portion described by the cutoff are dropped, before share status and links are computed.
- `sort`: Flag that indicates if the clones in each file are sorted by their share status.
- `countColors`: A two element vector specifying the extreme points of a linear color ramp. The ramp ranges from 0 to the maximal relative count of all clones in all files.
- `linkColors`: A three or six element vector. In the three element case, it defines for each type of link the color that is used. In the six element case `linkColors[1]`, `linkColors[3]` and `linkColors[5]` specify the lower extreme points of three linear color ramps. `linkColors[2]`, `linkColors[4]` and `linkColors[6]` specify the upper extreme points. All three ramps range from 0 to the maximal sum of the relative count of both end points over all links.
- `showLinks`: A three element vector indicating which type of link should be displayed.

For the `circos6` we do not specify individual color ramps for all link types, but only one ramp shared by all links. There is also no way to hide individual links.

Examples

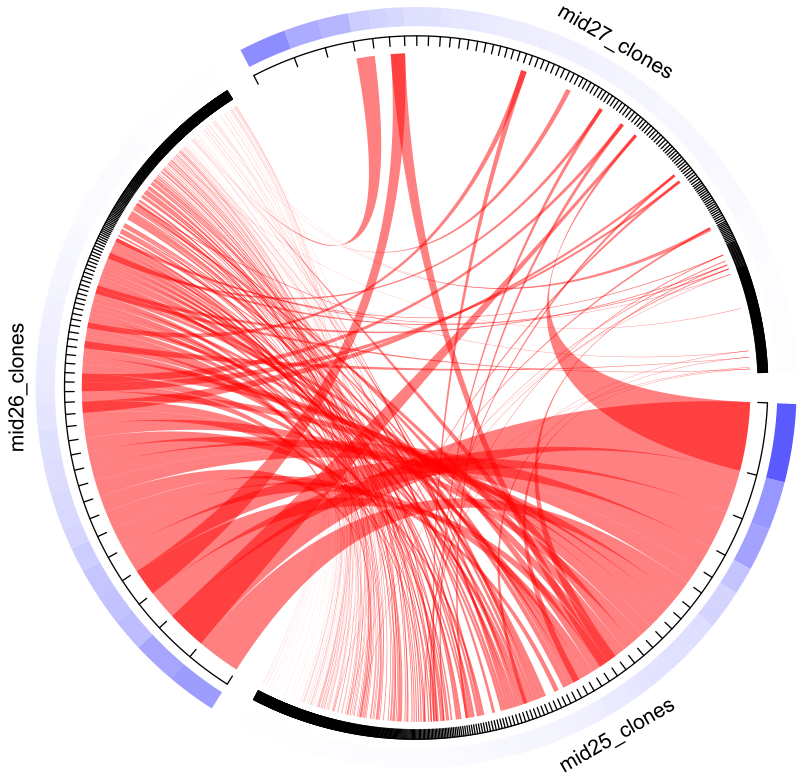

Figure 1.1: `circos3` *plot with default values*

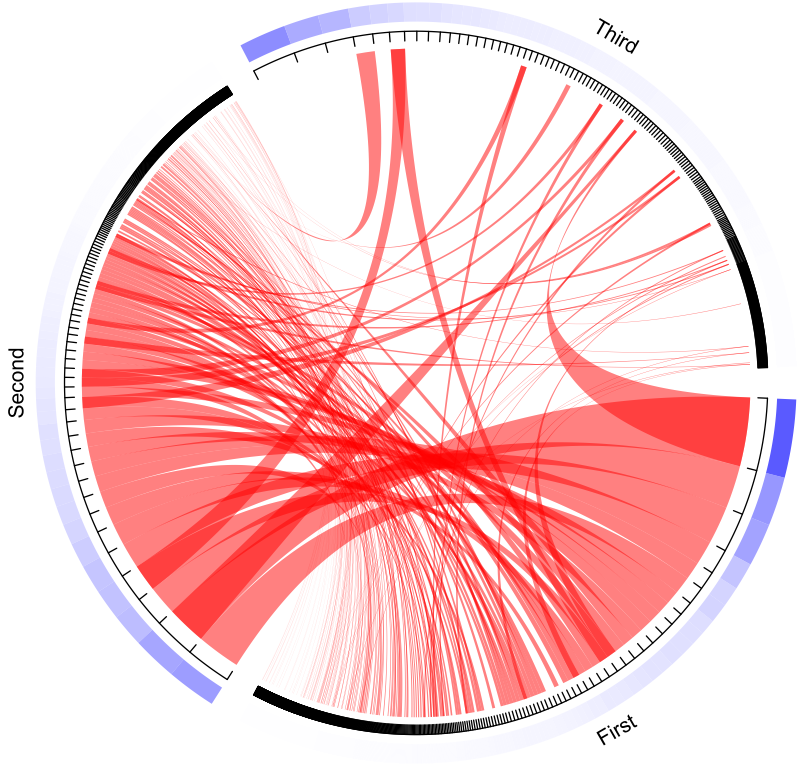

Figure 1.2: `circos3` *plot with* `fileAliases = c("First", "Second", "Third")`

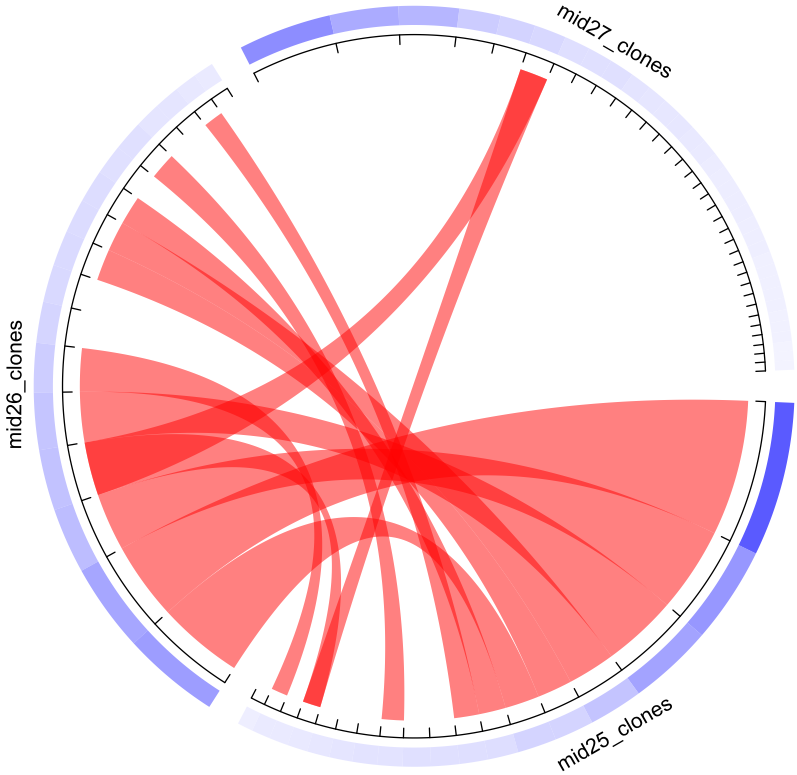

Figure 1.3: `circos3` *plot with* `cutoff = 0.5`

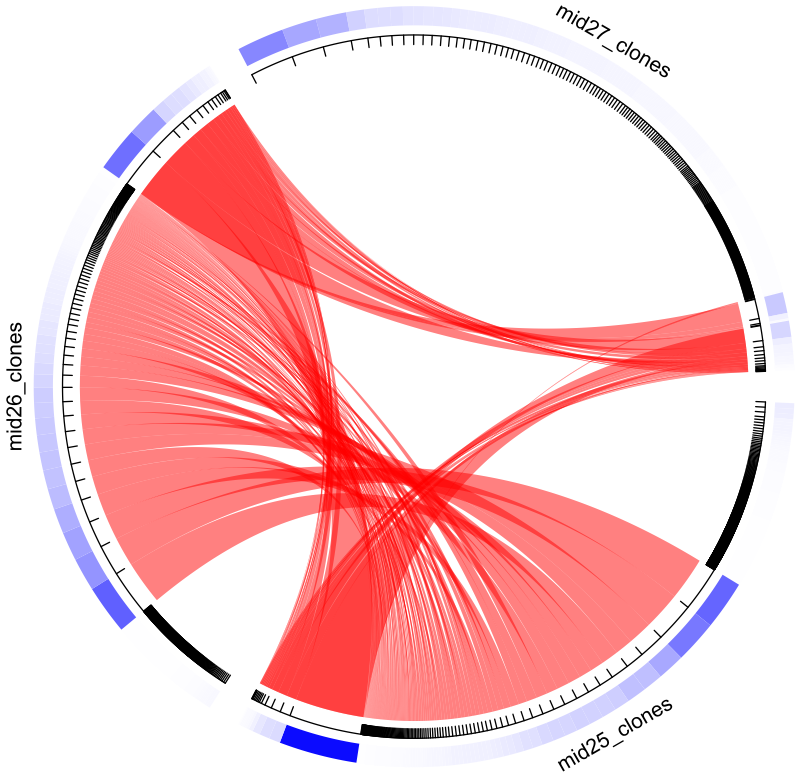

Figure 1.4: `circos3` *plot with* `sort = TRUE`

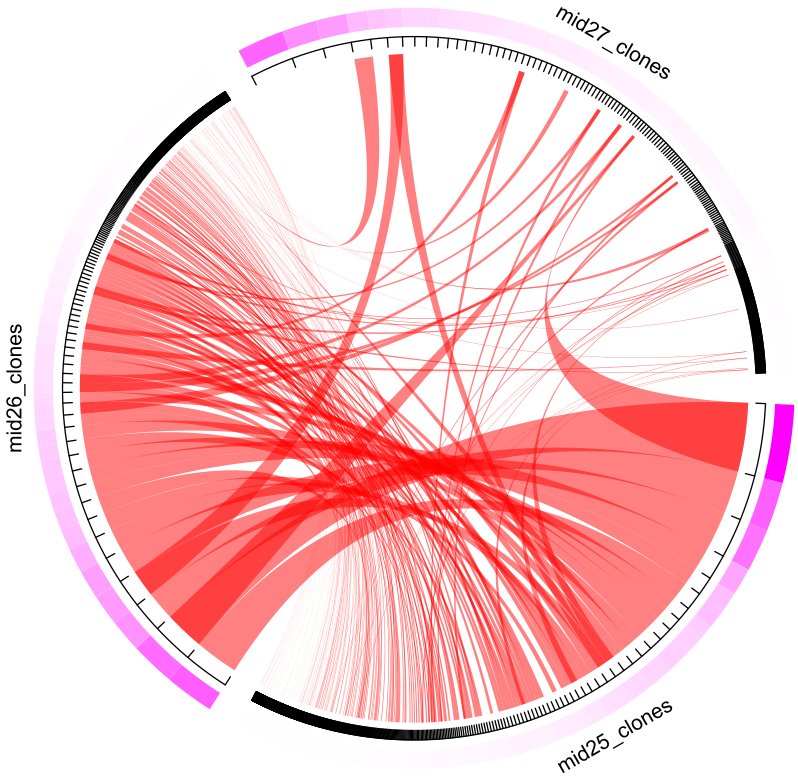

Figure 1.5: `circos3` *plot with* `countColors = c("#FFFFFFFF", "#FF00FFFF")`

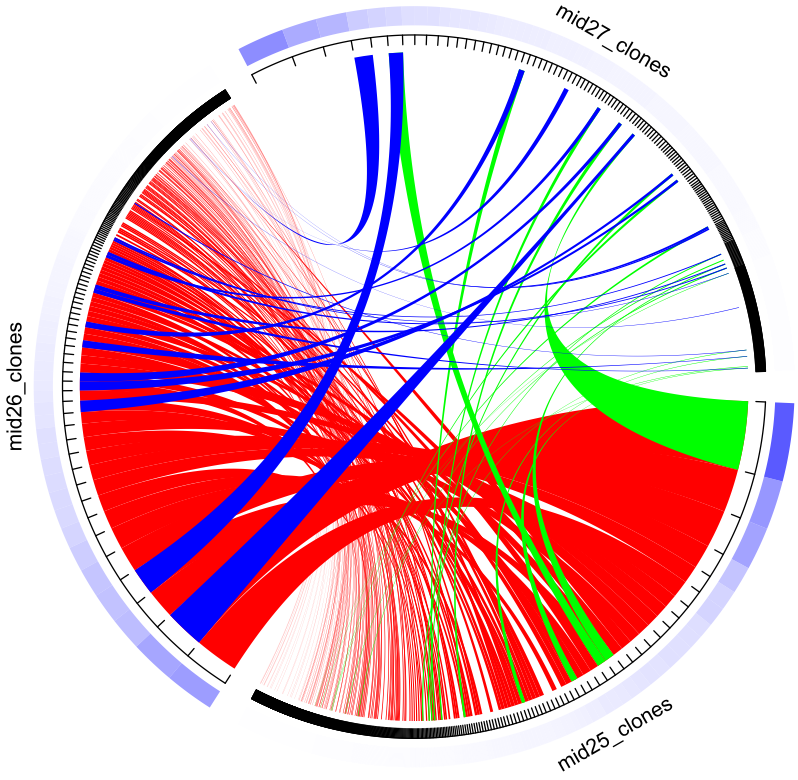

Figure 1.6: `circos3` *plot with* `linkColors = c("#FF0000FF", "#00FF00FF", "#0000FFFF")`

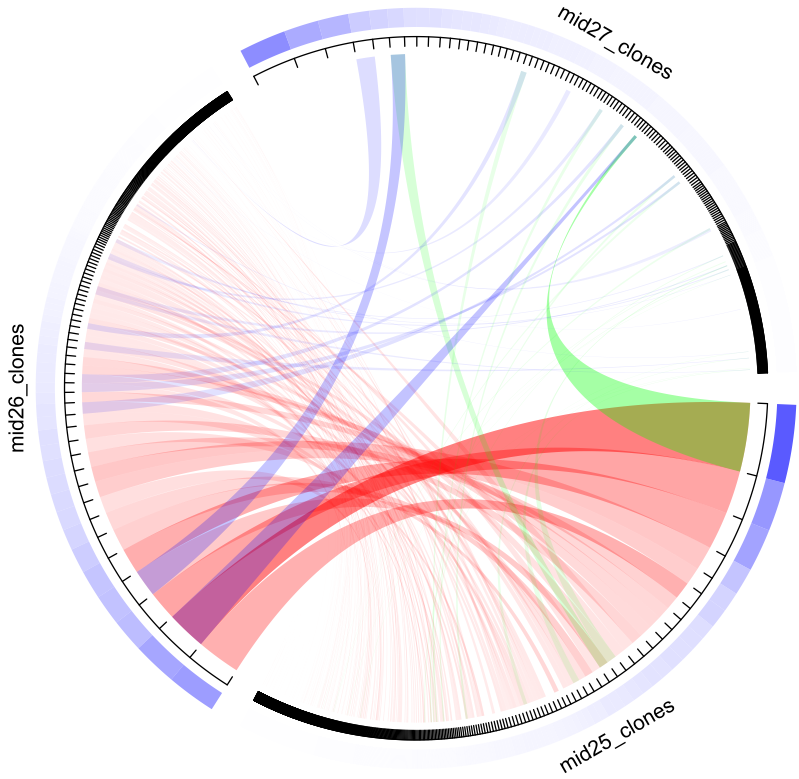

Figure 1.7: `circos3` *plot with* `linkColors = c("#FF000010", "#FF000080", "#00FF0010", "#00FF0080", "#0000FF10", "#0000FF80")`

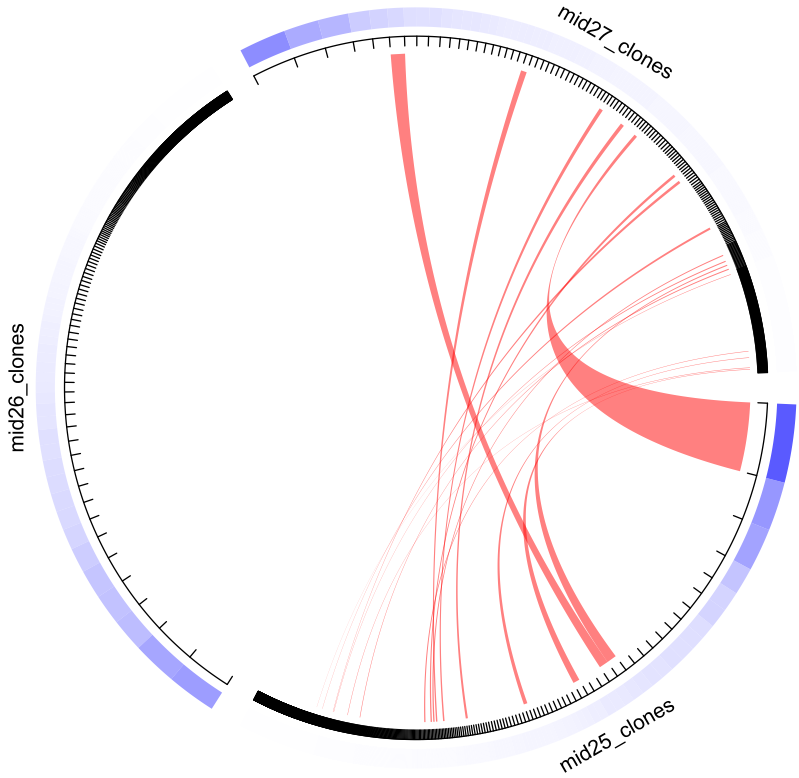

Figure 1.8: `circos3` *plot with* `showLinks = c(FALSE, TRUE, FALSE)`

Treemaps

Tree maps use the the `ggplot2` and `treemapify` package. Install the package in Rstudio using the install button in the *Package* tab. There are tree map functions for 1, 2, 3 and 4 files.

```
tree1(file1,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0)

tree2(file1, file2,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    colors = c("#FF0000"))

tree3(file1, file2, file3,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    colors = c("#FF0000", "#00FF00", "#0000FF", "#5A5A5A"))

tree4(file1, file2, file3, file4,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    colors = c("#FF0000", "#00FF00", "#0000FF", "#0000FF", "#00FF00", "#FF0000",
"#5A5A5A", "#5A5A5A", "#5A5A5A", "#5A5A5A", "#5A5A5A"))
```

Here we show the options for the 3 file variant.

In `tree3` there are seven different share statuses:

1. In `file1`
2. In `file2`
3. In `file3`
4. In `file1` and `file2`
5. In `file1` and `file3`
6. In `file2` and `file3`
7. In `file1`, `file2` and `file3`

`tree3` has the following parameters:

- `file1`, `file2`, `file3`: Specifies the path to the three input csv files.
- `fileAliases`: A three element vector specifying alternative names for the input files that are used as lables in the diagram.
- `saveFolder`: Specifies the path to the folder where the diagram and parameter file are saved.
- `cutoff`: Is a value between 0 and 1.0. For each file, all reads not in the top portion described by the cutoff are dropped, before share status are computed.

- `colors`: A four element vector specifying the colors used to denote the last four share status inside the tree map. The first three status are denoted by the color `#FFFFFF`.

Examples

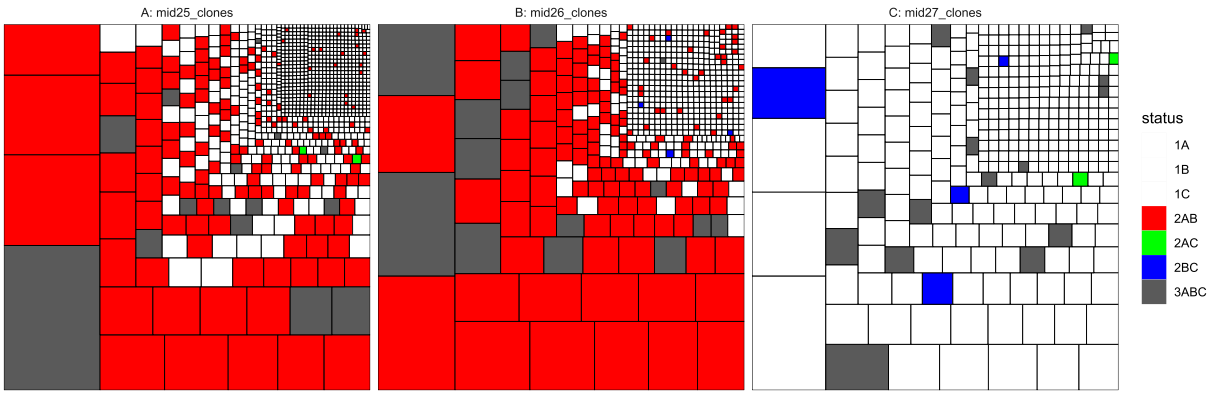

Figure 2.1: `tree3` *map with default values*

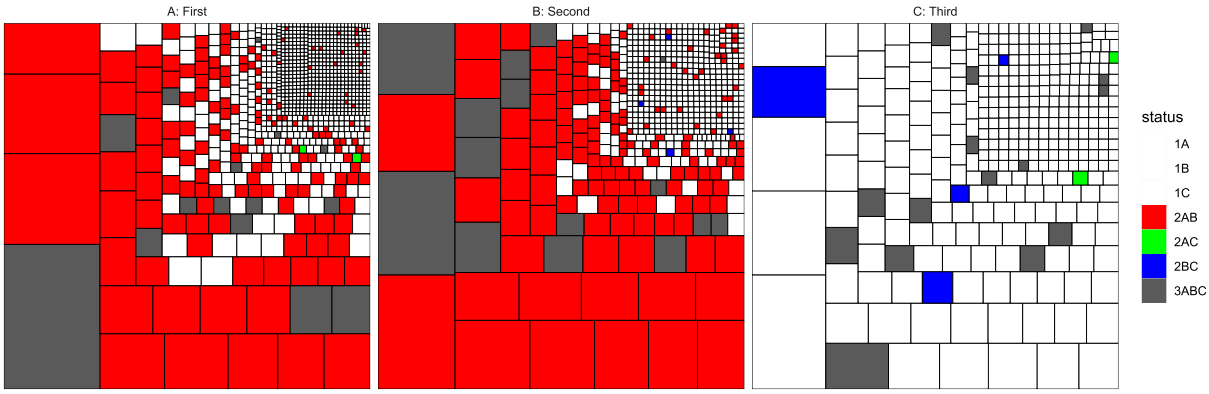

Figure 2.2: `tree3` *map with* `fileAliases = c("First", "Second", "Third")`

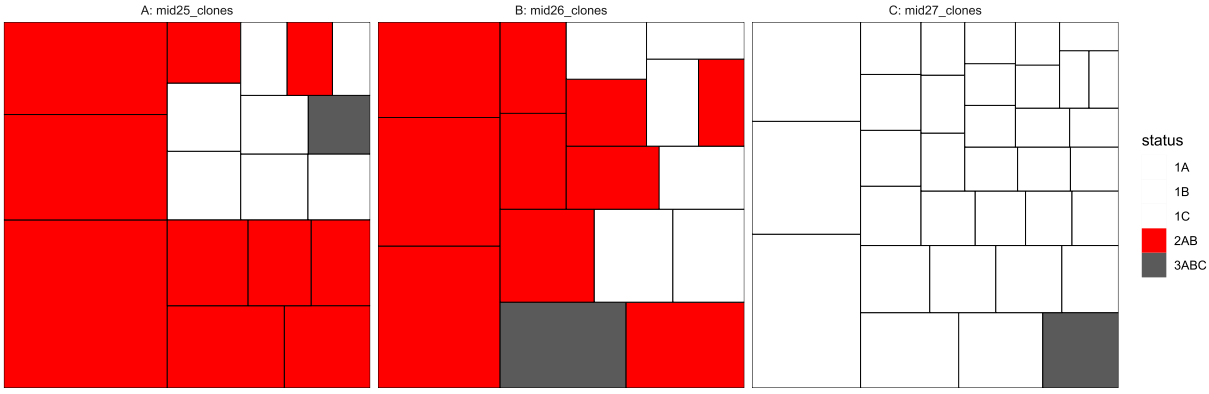

Figure 2.3: `tree3` *map with* `cutoff = 0.5`

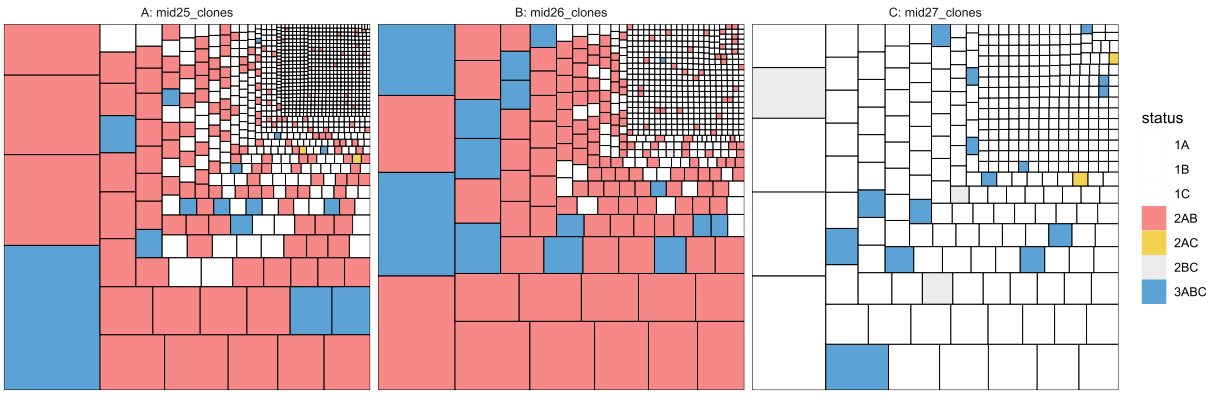

Figure 2.4: `tree3` *map with* `colors = c("#F78888", "#F3D250", "#ECECEC", "#5DA2D5")`

Tracking plots

Tracking plots use the the `ggplot2` and `treemapify` package. Install the package in Rstudio using the install button in the *Package* tab. There are tracking plot functions for 1, 2, 3 and 4 files.

```
track1(file1,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    numToTrack = 5,
    baseFileId = 1,
    baseFile = NULL,
    colors = rainbow(5))

track2(file1, file2,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    numToTrack = 5,
    baseFileId = 1,
    baseFile = NULL,
    colors = rainbow(5))

track3(file1, file2, file3,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    numToTrack = 5,
    baseFileId = 1,
    baseFile = NULL,
    colors = rainbow(5))

track4(file1, file2, file3, file4,
    fileAliases = NULL,
    saveFolder = NULL,
    cutoff = 1.0,
    numToTrack = 5,
    baseFileId = 1,
    baseFile = NULL,
    colors = rainbow(5))
```

Here we show the options for the 3 file variant.

`track3` operates in four different modes:

1. `baseFileId = 1`
2. `baseFileId = 2`
3. `baseFileId = 3`
4. `baseFileId = 0`

`track3` has the following parameters:

- `file1`, `file2`, `file3`: Specifies the path to the three input csv files.

- `fileAliases`: A three element vector specifying alternative names for the input files that are used as lables in the diagram.
- `saveFolder`: Specifies the path to the folder where the diagram and parameter file are saved.
- `cutoff`: Is a value between 0 and 1.0. For each file, all reads not in the top portion described by the cutoff are dropped, before tracking.
- `numToTrack`: In the first three modes, the value specifies the number of clones that are tracked. In the fourth mode it is ignored.
- `baseFileId`: Specifies the mode. In the first mode we chose the clones to track from `file1`, in the second `file2`, in the third `file3` and in the fourth mode we use a base file.
- `baseFile`: In the first three modes the value is ignored. In the fourth it specifies a path to a csv file, serving as the base file.
- `colors`: A vector of colors. In the first three modes the value should be as long as `numToTrack`. In the fourth as long as the base file. If it is not long enough we use equally spaced colors from the `rainbow` color pallet.

Examples

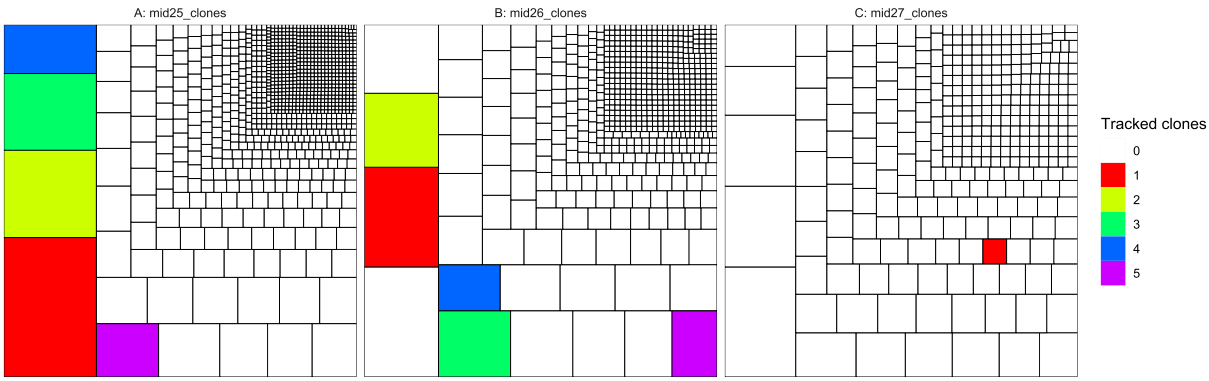

Figure 3.1: `track3` *plot with default values*

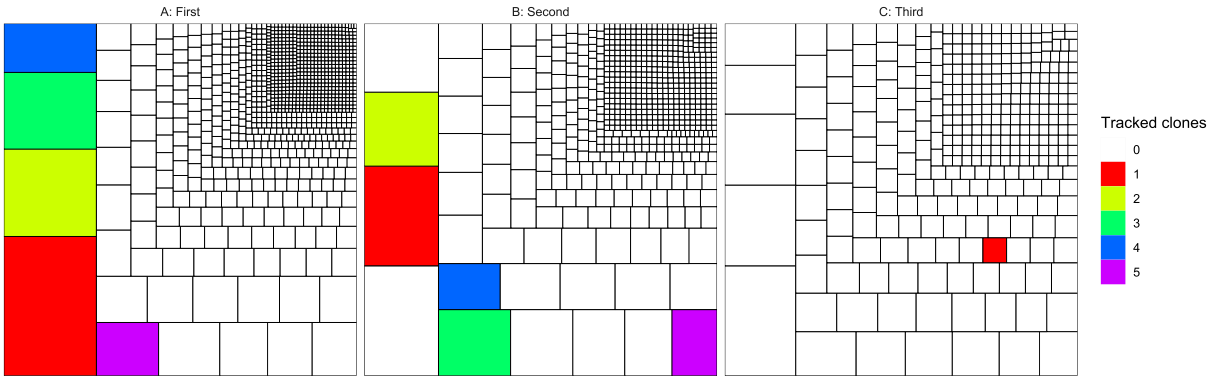

Figure 3.2: `track3` *plot with* `fileAliases = c("First", "Second", "Third")`

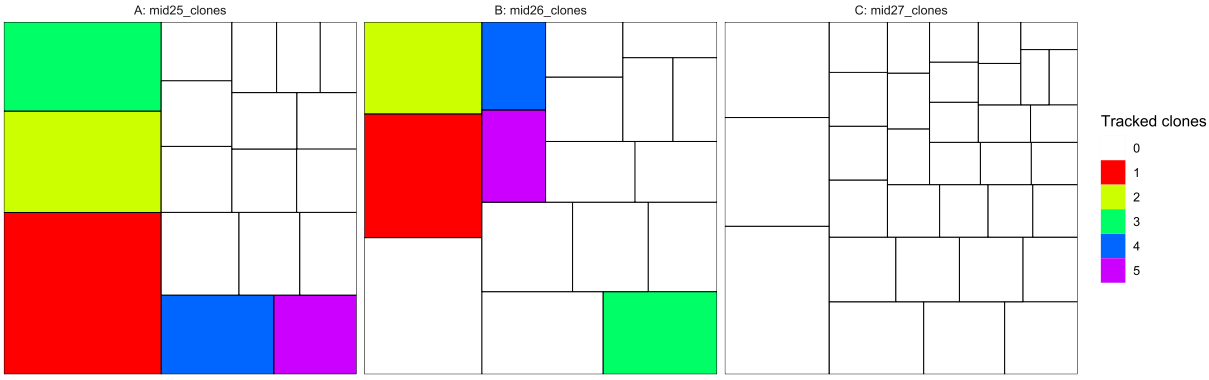

Figure 3.3: `track3` *plot with* `cutoff = 0.5`

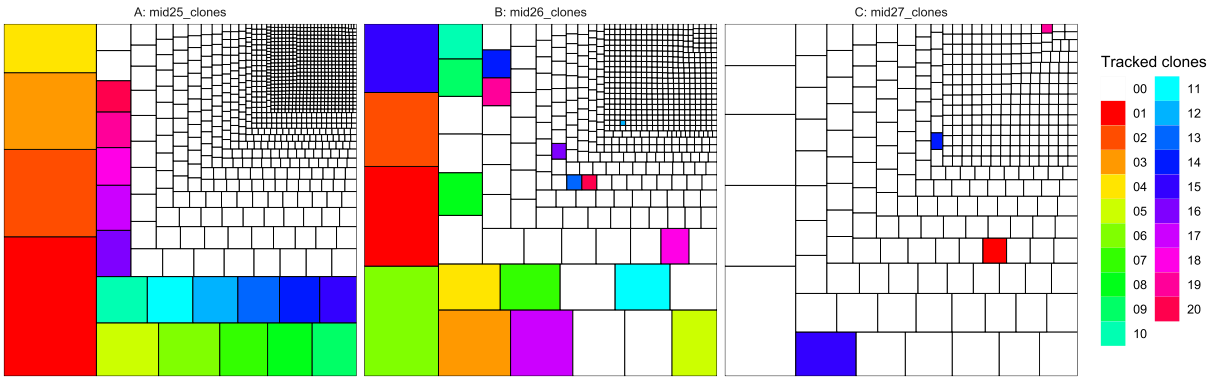

Figure 3.4: `track3` *plot with* `numToTrack = 20`

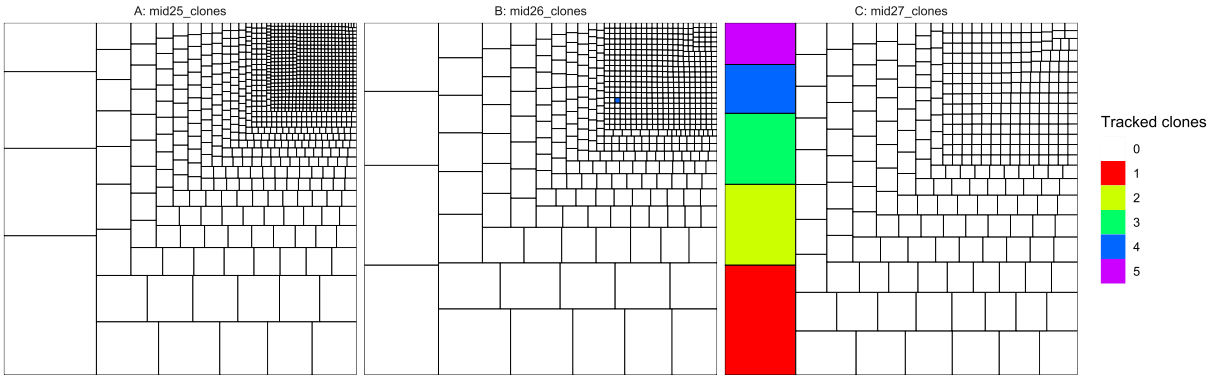

Figure 3.5: `track3` *plot with* `baseFileId = 3`

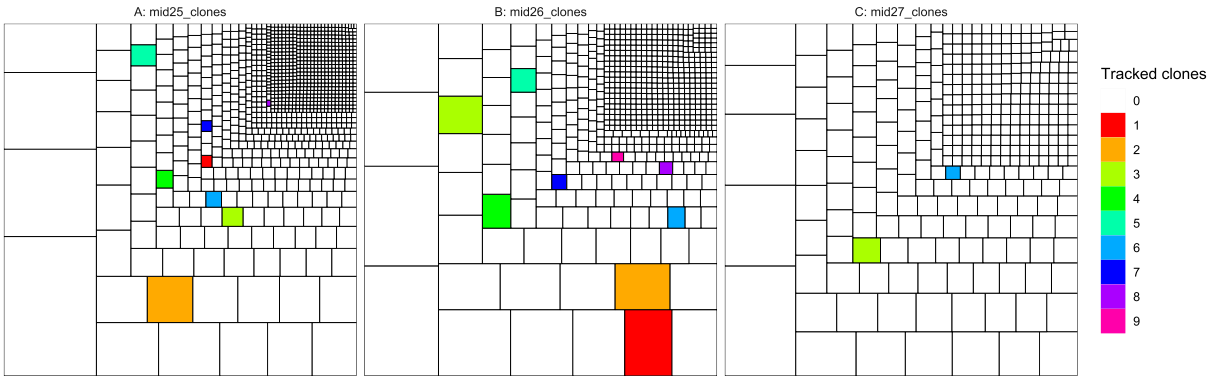

Figure 3.6: `track3` *plot with* `baseFileId = 0` *and a base file containing random clones from* `file2`

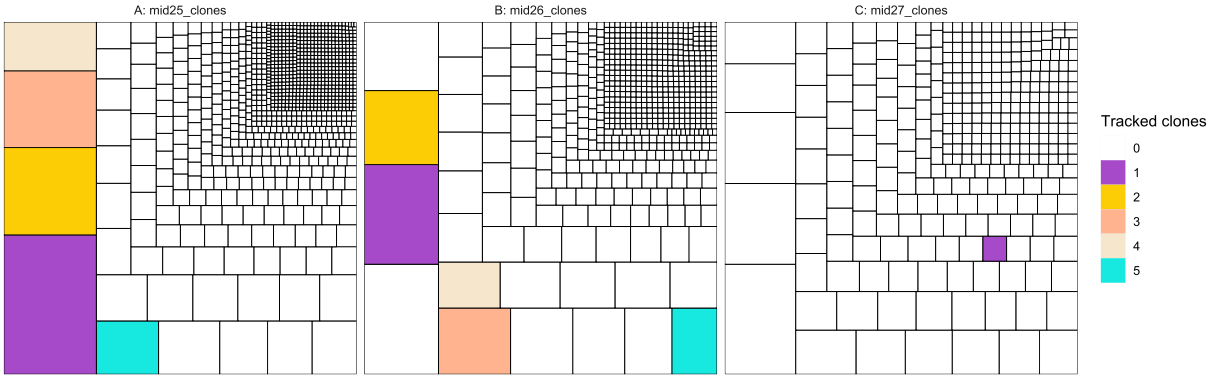

Figure 3.7: `track3` *plot with* `colors = c("#A64AC9", "#FCCD04", "#FFB48F", "#F5E6CC", "#17E9E0")`