Plotting Haplotype Networks with pegas

Emmanuel Paradis

April 8, 2021

Contents

1 Introduction

Haplotype networks are powerful tools to explore the relationships among individuals characterised with genotypic or sequence data [3, 5]. `pegas` has had tools to infer and plot haplotype networks since its first version (0.1, released in May 2009). These tools have improved over the years and are appreciated in the community working on population genetics and genomics (see John Bhorne's blog[1]).

This document covers some aspects of drawing haplotype networks with `pegas` with an emphasis on recent improvements. Not all details and options are covered here: see the respective help pages (`?plot.haploNet` and `?mutations`) for full details. The function `plotNetMDS`, which offers an alternative approach to plotting networks, is not considered in this document.

2 The Function `plot.haploNet`

The current version of `pegas` includes five methods to reconstruct haplotype networks as listed in the table below.

Method	Acronym	Input data	Function	Ref.
Parsimony network	TCS	distances	`haploNet`	[6]
Minimum spanning tree	MST	"	`mst`	[4]
Minimum spanning network	MSN	"	`msn`	[1]
Randomized minimum spanning tree	RMST	"	`rmst`	[5]
Median-joining network	MJN	sequences	`mjn`	[1]

[1] https://johnbhorne.wordpress.com/2016/09/15/still-making-haplotype-networks-the-old-way-how-to-do-it-in-r/

All these functions output an object of class `"haploNet"` so that they are plotted with the same `plot` method (`plot.haploNet`).

2.1 Node Layout

The coordinates of the nodes (or vertices) representing the haplotypes are computed in two steps: first, an equal-angle algorithm borrowed from Felsenstein [2] is used; second, the spacing between nodes is optimised. The second step is ignored if the option `fast = TRUE` is used when calling `plot`. These two steps are detailed a bit in the next paragraphs.

In the first step, the haplotype with the largest number of links is placed at the centre of the plot (i.e., its coordinates are $x = y = 0$), then the haplotypes connected to this first haplotype are arranged around it and given equal angles. This is then applied recursively until all haplotypes are plotted. To perform this layout, an initial 'backbone' network based on an MST is used, so there is no reticulation and the equal-angle algorithm makes sure that there is no segment-crossing. In practice, it is likely that this backbone MST is arbitrary with respect to the rest of the network. The other segments are then drawn on this MST.

In the second step, a 'global energy' is calculated based on the spaces between the nodes of the network (closer nodes imply higher energies). The nodes are then moved repeatedly, while keeping the initial structure of the backbone MST, until the global energy is not improved (decreased).

We illustrate the procedure with the `woodmouse` data, a set of sequences of cytochrome *b* from 15 woodmice (*Apodemus sylvaticus*):

```
> library(pegas) # loads also ape
> data(woodmouse)
```

In order, to simulate some population genetic data, we sample, with replacement, 80 sequences, and create two hierarchical groupings: `region` with two levels each containing 40 haplotypes, and `pop` with four levels each containing 20 haplotypes:

```
> set.seed(10)
> x <- woodmouse[sample.int(nrow(woodmouse), 80, TRUE), ]
> region <- rep(c("regA", "regB"), each = 40)
> pop <- rep(paste0("pop", 1:4), each = 20)
> table(region, pop)
      pop
region pop1 pop2 pop3 pop4
  regA   20   20    0    0
  regB    0    0   20   20
```

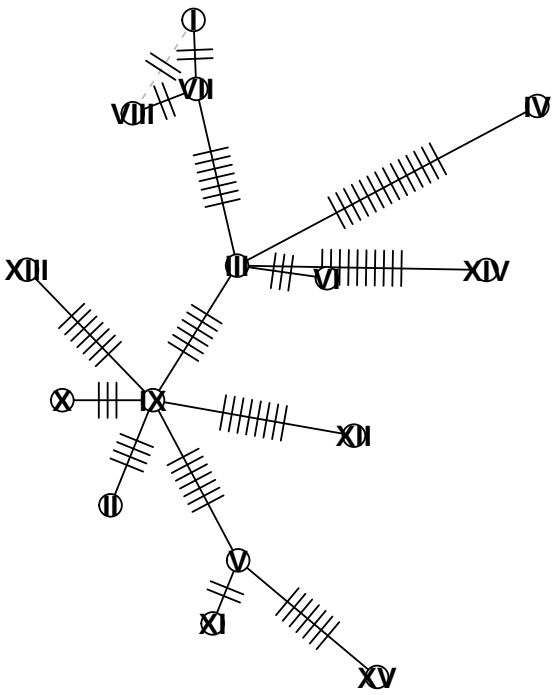
We extract the haplotypes which are used to reconstruct the RMST after computing the pairwise Hamming distances:

```
> h <- haplotype(x)
> h
Haplotypes extracted from: x

    Number of haplotypes: 15
         Sequence length: 965

Haplotype labels and frequencies:

   I   II  III   IV    V   VI  VII VIII   IX    X   XI  XII XIII  XIV   XV
   5    6    9    3    7    7    4    6    3    7    7    6    3    5    2
```

```
> d <- dist.dna(h, "N")
> nt <- rmst(d, quiet = TRUE)
> nt
Haplotype network with:
  15 haplotypes
  22 links
  link lengths between 2 and 14 steps

Use print.default() to display all elements.
```

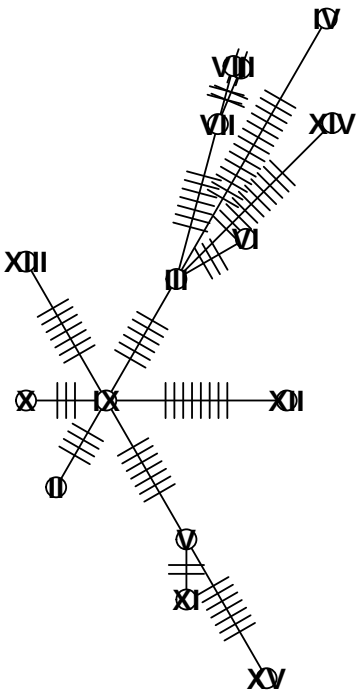
We now plot the network with the default arguments:

```
> plot(nt)
```



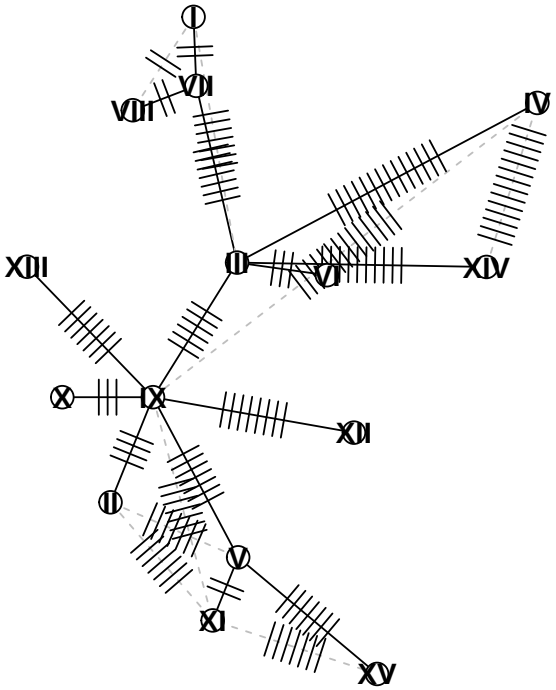
We compare the layout obtained with `fast = TRUE`:

```
> plot(nt, fast = TRUE)
```



By default, not all links are drawn. This is controlled with the option `threshold` which takes two values in order to set the lower and upper bounds of the number of mutations for a link to be drawn:

```
> plot(nt, threshold = c(1, 14))
```



The visual aspect of the links is arbitrary: the links of the backbone MST are shown with continuous segments, while "alternative" links are shown with dashed segments.

2.2 Options

`plot.haploNet` has a relatively large number of options:

```
> args(pegas:::plot.haploNet)
function (x, size = 1, col, bg, col.link, lwd, lty, shape = "circles",
    pie = NULL, labels, font, cex, scale.ratio, asp = 1, legend = FALSE,
    fast = FALSE, show.mutation, threshold = c(1, 2), xy = NULL,
    ...)
NULL
```

Like for most `plot` methods, the first argument (`x`) is the object to be plotted. Until `pegas` 0.14, all other arguments were defined with default values. In recent versions, as shown above, only `size` and `shape` are defined with default values; the other options, if not modified in the call to `plot`, are taken from a set of parameters which can be modified as explained in Section 3.4.

The motivation for this new function definition is that in most cases users need to modify

`size` and `shape` with their own data, such as haplotype frequencies or else, and these might be changed repeatedly (e.g., with different data sets or subsets). On the other hand, the other options are more likely to be used to modify the visual aspect of the graph, so it could be more useful to change them once during a session as explained later in this document.

The size of the haplotype symbols can be used to display haplotype frequencies. The function `summary` can extract these frequencies from the `"haplotype"` object:

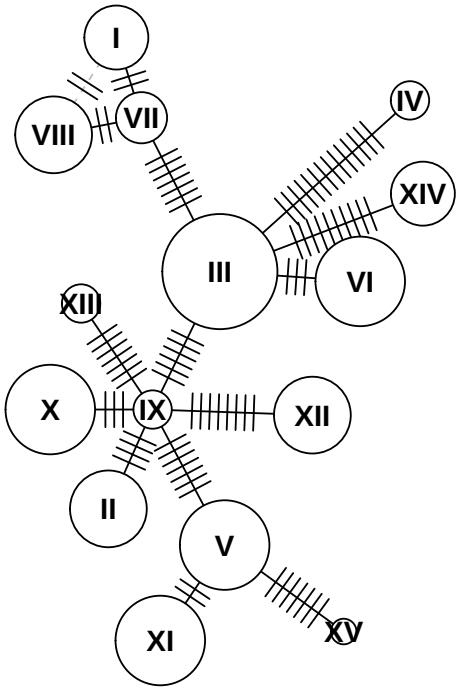
```
> (sz <- summary(h))
  I  II III  IV   V  VI VII VIII  IX   X  XI XII XIII XIV  XV
  5   6   9   3   7   7   4   6   3   7   7   6   3   5   2
```

It is likely that these values are not ordered in the same way than haplotypes are ordered in the network:

```
> (nt.labs <- attr(nt, "labels"))
 [1] "VII"  "XV"   "I"    "XIII" "III"  "IX"   "VI"   "X"    "VIII" "XIV"
[11] "II"   "V"    "XII"  "IV"   "XI"
```

It is simple to reorder the frequencies before using them into `plot`:

```
> sz <- sz[nt.labs]
> plot(nt, size = sz)
```



A similar mechanism can be used to show variables such as `region` or `pop`. The function `haploFreq` is useful here because it computes the frequencies of haplotypes for each region or population:

```
> (R <- haploFreq(x, fac = region, haplo = h))
     regA regB
I       2    3
II      2    4
III     5    4
IV      2    1
V       6    1
VI      4    3
VII     3    1
VIII    4    2
IX      1    2
X       3    4
XI      4    3
XII     1    5
XIII    2    1
XIV     1    4
XV      0    2
```

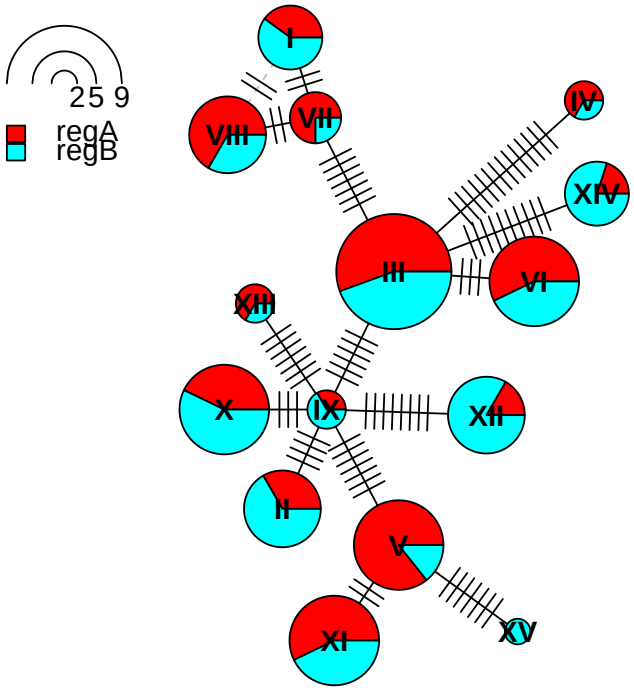
```
> (P <- haploFreq(x, fac = pop, haplo = h))
     pop1 pop2 pop3 pop4
I       1    1    3    0
II      1    1    1    3
III     3    2    3    1
IV      1    1    1    0
V       3    3    0    1
VI      3    1    1    2
VII     1    2    1    0
VIII    4    0    1    1
IX      1    0    1    1
X       1    2    3    1
XI      1    3    1    2
XII     0    1    2    3
XIII    0    2    0    1
XIV     0    1    1    3
XV      0    0    1    1
```

Like with `size`, we have to reorder these matrices so that their rows are in the same order than in the network:

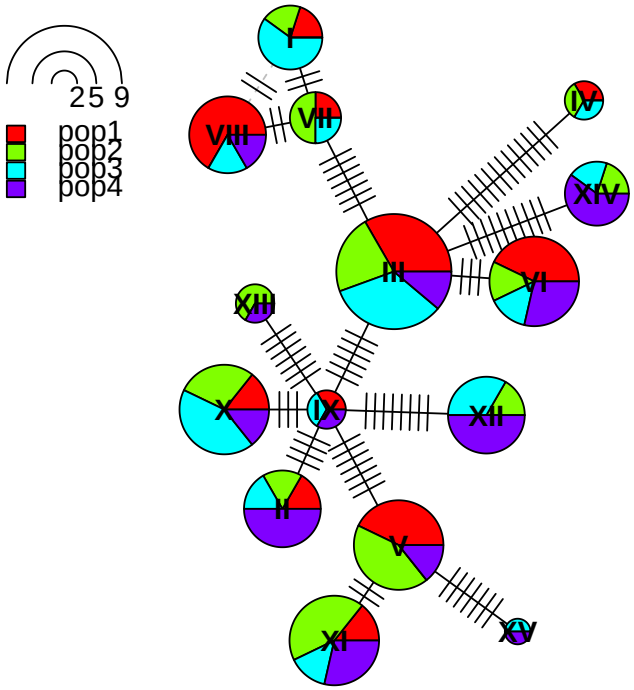
```
> R <- R[nt.labs, ]
> P <- P[nt.labs, ]
```

We may now plot the network with either information on haplotype frequencies by just changing the argument `pie`:

```
> plot(nt, size = sz, pie = R, legend = c(-25, 30))
```




```
> plot(nt, size = sz, pie = P, legend = c(-25, 30))
```



The option `legend` can be:

- `FALSE` (the default): no legend is shown;
- `TRUE`: the user is asked to click where the legend should be printed;
- a vector of two values with the coordinates where the print the legend (for non-interactive use like in this vignette).

3 New Features in pegas 1.0

This section details some of the improvements made to haplotype network drawing after pegas 0.14.

3.1 Improved 'Replotting'

The graphical display of networks is a notoriously difficult problem, especially when there is an undefined number of links (or edges). The occurrence of reticulations makes line crossings almost inevitable. The packages igraph and network have algorithms to optimise the layouts of nodes and edges when plotting such networks.

The function `replot` (introduced in `pegas` 0.7, March 2015) lets the user modify the layout of nodes interactively by clicking on the graphical window where a network has been plotted beforehand. `replot`—which cannot be used in this non-interactive vignette—has been improved substantially:

- The explanations printed when the function is called are more detailed and the node to be moved is visually identified after clicking.
- The final coordinates, for instance saved with `xy <- replot()`, can be used directly into `plot(nt, xy = xy)`. This also makes possible to input coordinates calculated with another software.
- In previous versions, the limits of the plot tended to drift when increasing the number of node moves. This has been fixed, and the network is correctly displayed whatever the number of moves done.

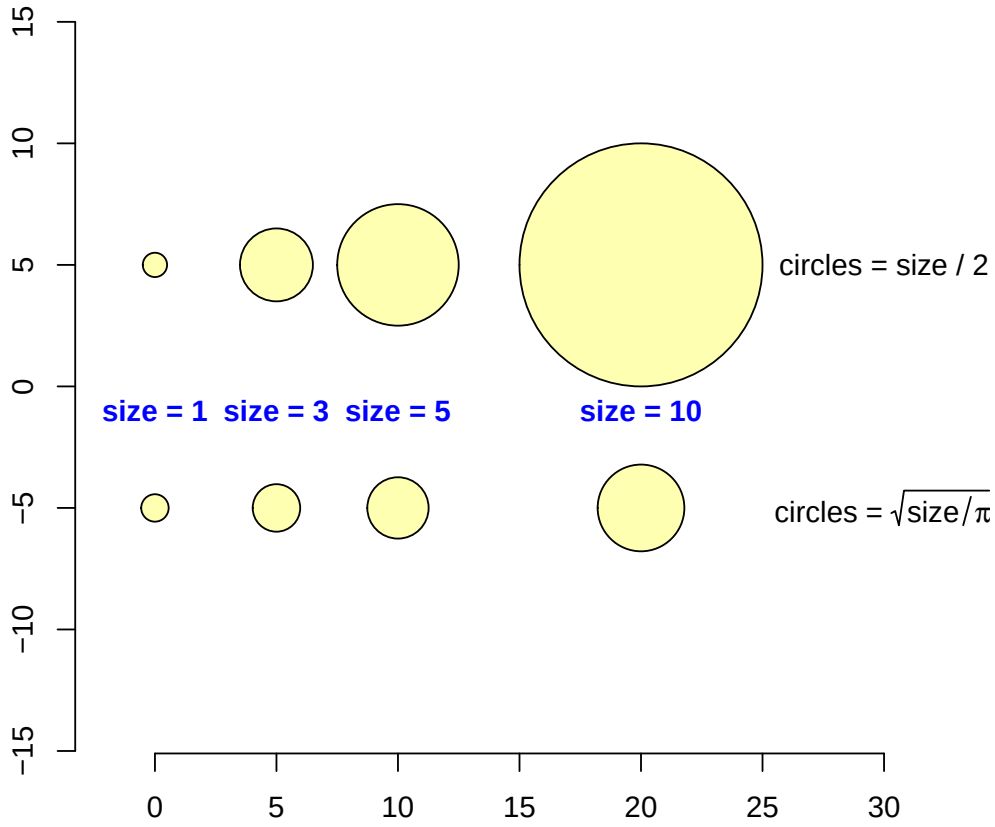
3.2 Haplotype Symbol Shapes

Haplotypes can be represented with three different shapes: circles, squares, or diamonds. The argument `shape` of `plot.haploNet` is used in the same way than `size` as explained above (including the evental need to reorder the values). Some details are given below about how these symbols are scaled.

There are two ways to display a quantitative variable using the size of a circle: either with its radius (r) or with the area of the disc defined by the circle. This area is πr^2, so if we want the area of the symbols to be proportional to `size`, we should square-root these last values. However, in practice this masks variation if most values in `size` are not very different (see below). In `pegas`, the diameters of the circles ($2r$) are equal to the values given by `size`. If these are very heterogeneous, they could be transformed with `size = sqrt(....` keeping in mind that the legend will be relative to this new scale.

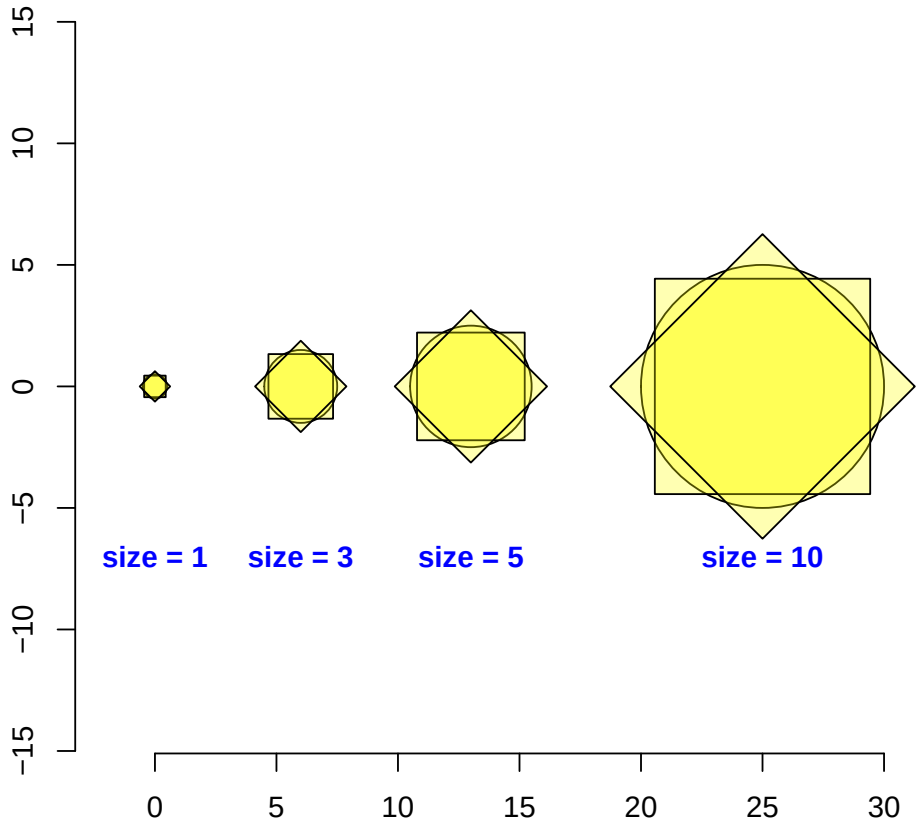
The next figure shows both ways of scaling the size of the circles: the top one is the scaling used in `pegas`.

```
> par(xpd = TRUE)
> size <- c(1, 3, 5, 10)
> x <- c(0, 5, 10, 20)
> plot(0, 0, type="n", xlim=c(-2, 30), asp=1, bty="n", ann=FALSE)
> other.args <- list(y = -5, inches = FALSE, add = TRUE,
+                    bg = rgb(1, 1, 0, .3))
> o <- mapply(symbols, x = x, circles = sqrt(size / pi),
+             MoreArgs = other.args)
> other.args$y <- 5
> o <- mapply(symbols, x = x, circles = size / 2,
+             MoreArgs = other.args)
> text(x, -1, paste("size =", size), font = 2, col = "blue")
> text(30, -5, expression("circles = "*sqrt(size / pi)))
> text(30, 5, "circles = size / 2")
```



For squares and diamonds (`shape = "s"` and `shape = "d"`, respectively), they are scaled so that their areas are equal to the disc areas for the same values given to `size`. The figure below shows these three symbol shapes superposed for several values of this parameter. Note that a diamond is a square rotated 45° around its center.

```
> x <- c(0, 6, 13, 25)
> plot(0, 0, type="n", xlim=c(-2, 30), asp=1, bty="n", ann=FALSE)
> other.args$y <- 0
> o <- mapply(symbols, x = x, circles = size/2, MoreArgs = other.args)
> other.args$col <- "black"
> other.args$add <- other.args$inches <- NULL
> o <- mapply(pegas:::square, x = x, size = size, MoreArgs = other.args)
> o <- mapply(pegas:::diamond, x = x, size = size, MoreArgs = other.args)
> text(x, -7, paste("size =", size), font = 2, col = "blue")
```



3.3 The Function `mutations`

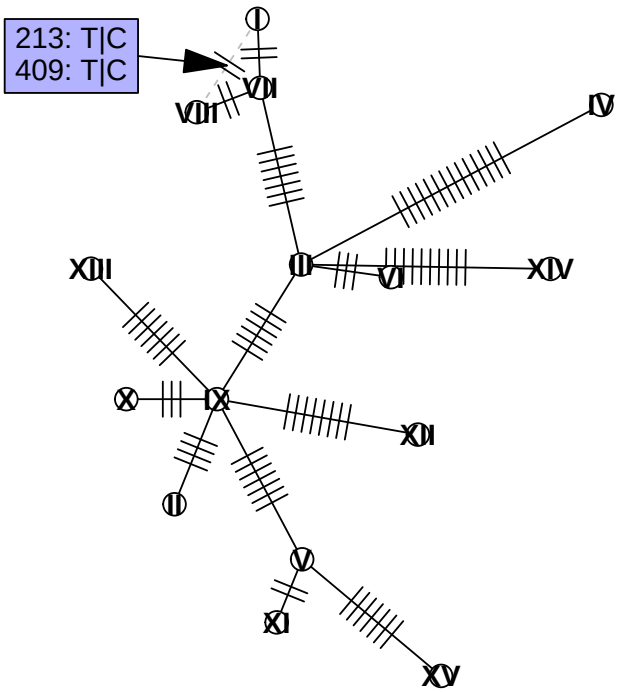
`mutations()` is a low-level plotting function which displays information about the mutations related to a particular link of the network. This function can be used interactively. For instance, the following is copied from an interactive R session:

```
> mutations(nt)
Link is missing: select one below
1: VII-I
2: VII-VIII
3: V-XI
4: III-VI
5: IX-X
6: IX-II
7: III-IX
8: VII-III
9: XV-V
10: XIII-IX
11: IX-V
12: IX-XII
```

```
13: III-XIV
14: III-IV
15: IX-XI
16: XV-XI
17: IX-IV
18: I-VIII
19: I-III
20: XIV-IV
21: II-XI
22: II-V

Enter a link number: 18
Coordinates are missing: click where you want to place the annotations:
The coordinates x = -8.880335, y = 16.313 are used
```

The values entered interactively can be written in a script to reproduce the figure:

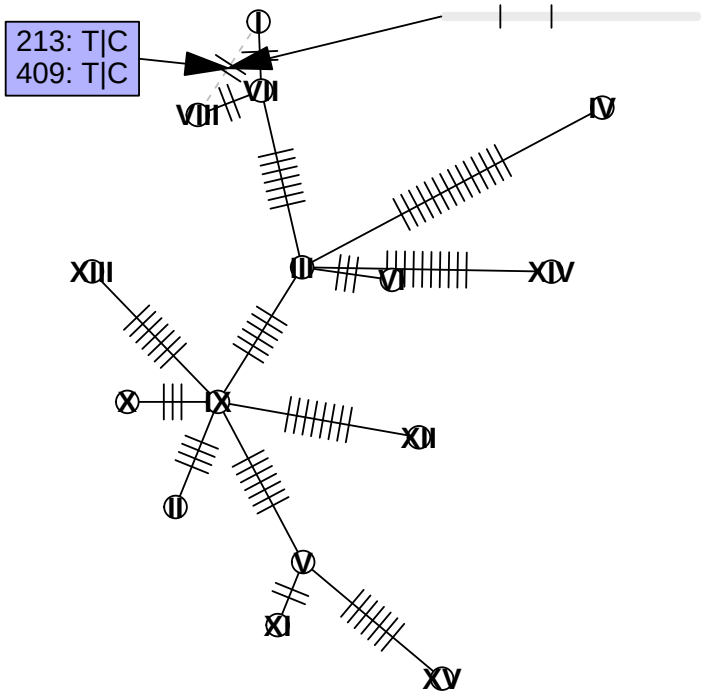
```
> plot(nt)
> mutations(nt, 18, x = -8.9, y = 16.3, data = h)
```



Like any low-level plotting function, `mutations()` can be called as many times as needed

to display similar information on other links. The option `style` takes the value `"table"` (the default) or `"sequence"`. In the second, the positions of the mutations are drawn on a horizontal segment representing the sequence:

```
> plot(nt)
> mutations(nt, 18, x = -8.9, y = 16.3, data = h)
> mutations(nt, 18, x = 10, y = 17, data = h, style = "s")
```



The visual aspect of these annotations is controlled by parameters as explained in the next section.

3.4 Getting and Setting Options

The new version of `pegas` has two ways to change some of the parameters of the plot: either by changing the appropriate option(s) in one of the above functions, or by setting these values with the function `setHaploNetOptions`, in which case all subsequent plots will be affected.[2] The list of the option values currently in use can be printed with `getHaploNetOptions`. There is a relatively large number of options that affect either `plot.haploNet()` or `mutations()`. Their names are quite explicit so that the user

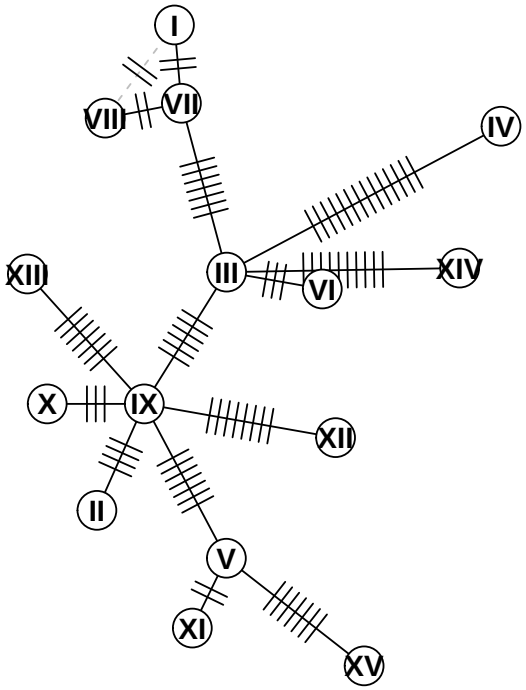
[2]See `?par` for a similar mechanism with basic R graphical functions.

should find which one(s) to modify easily:

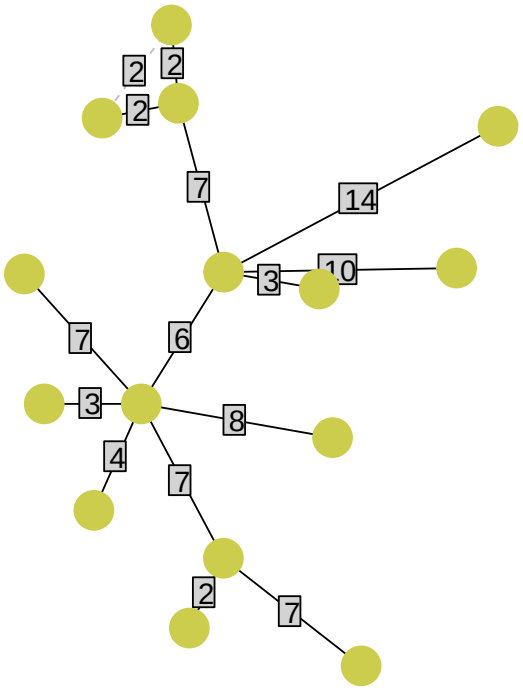
```
> names(getHaploNetOptions())
 [1] "labels"                      "labels.cex"
 [3] "labels.font"                 "link.color"
 [5] "link.type"                   "link.type.alt"
 [7] "link.width"                  "link.width.alt"
 [9] "haplotype.inner.color"       "haplotype.outer.color"
[11] "mutations.cex"               "mutations.font"
[13] "mutations.frame.background" "mutations.frame.border"
[15] "mutations.text.color"        "mutations.arrow.color"
[17] "mutations.arrow.type"        "mutations.sequence.color"
[19] "mutations.sequence.end"      "mutations.sequence.length"
[21] "mutations.sequence.width"    "pie.outer.color"
[23] "pie.inner.segments.color"    "pie.colors.function"
[25] "scale.ratio"                 "show.mutation"
```

We see here several examples with the command `plot(nt, size = 2)` which is repeated after calling `setHaploNetOptions`:

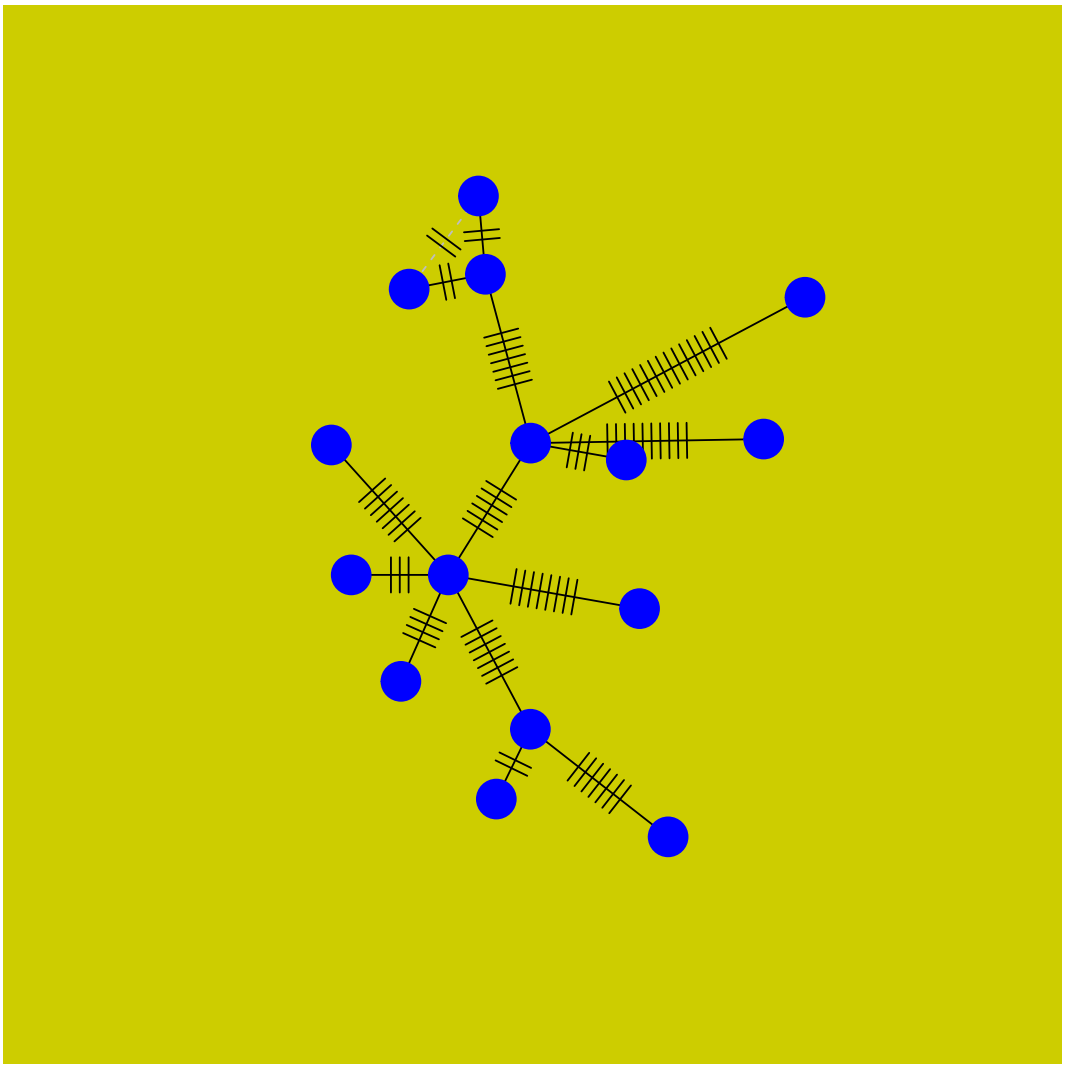
```
> plot(nt, size = 2)
```



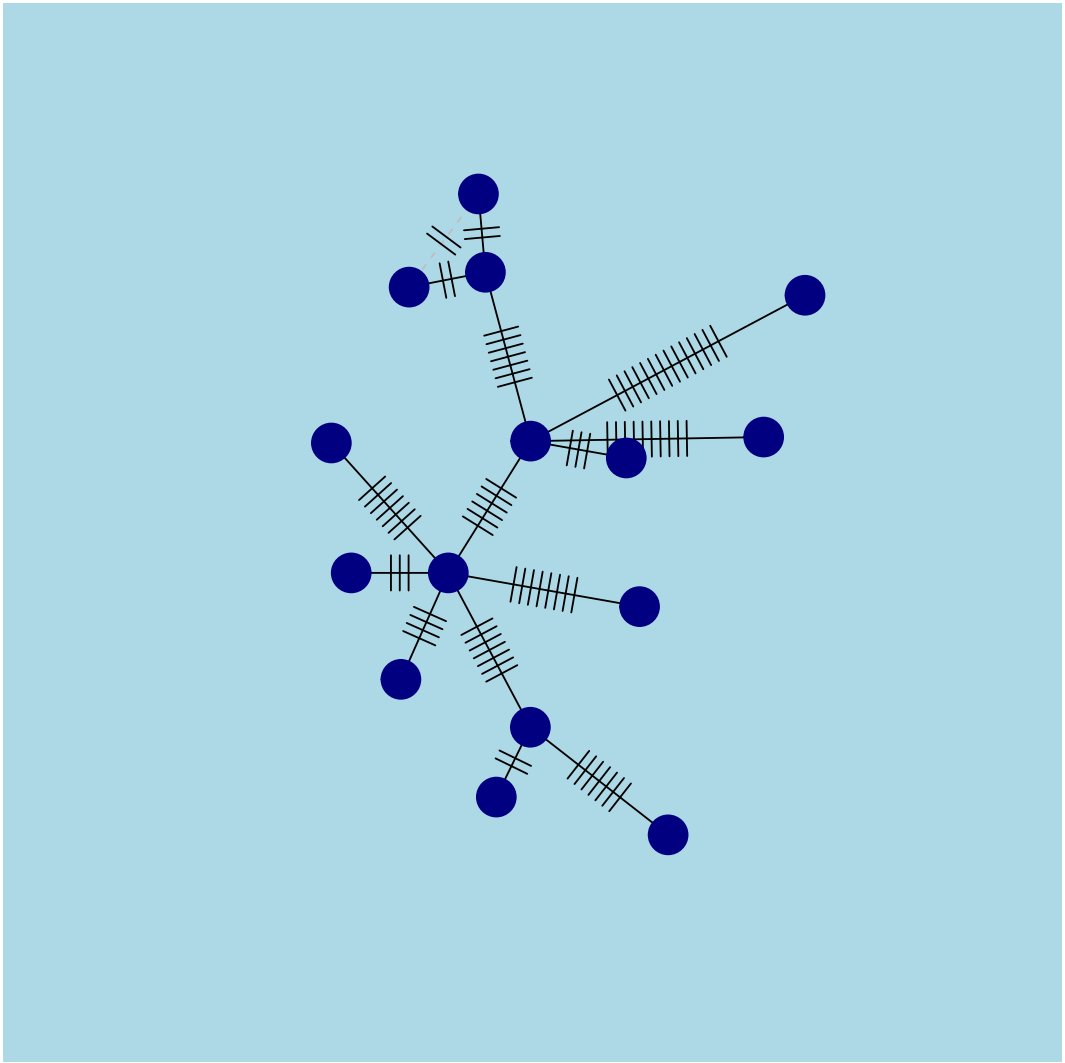

```
> setHaploNetOptions(haplotype.inner.color = "#CCCC4D",
+                    haplotype.outer.color = "#CCCC4D",
+                    show.mutation = 3, labels = FALSE)
> plot(nt, size = 2)
```




```
> setHaploNetOptions(haplotype.inner.color = "blue",
+                    haplotype.outer.color = "blue",
+                    show.mutation = 1)
> par(bg = "yellow3")
> plot(nt, size = 2)
```



```
> setHaploNetOptions(haplotype.inner.color = "navy",
+                    haplotype.outer.color = "navy")
> par(bg = "lightblue")
> plot(nt, size = 2)
```



References

[1] H. J. Bandelt, P. Forster, and A. Röhl. Median-joining networks for inferring intraspecific phylogenies. *Molecular Biology and Evolution*, 16(1):37–48, 1999.

[2] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, 2004.

[3] D. H. Huson and D. Bryant. Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution*, 23(2):254–267, 2006.

[4] J. B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[5] E. Paradis. Analysis of haplotype networks: the randomized minimum spanning tree method. *Methods in Ecology and Evolution*, 9(5):1308–1317, 2018.

[6] A. R. Templeton, K. A. Crandall, and C. F. Sing. A cladistic analysis of phenotypic association with haplotypes inferred from restriction endonuclease mapping and DNA sequence data. III. Cladogram estimation. *Genetics*, 132:619–635, 1992.