phyloscannerR: reconstructing transmission networks

Oliver Ratmann and Matthew Hall

2019-09-02

Introduction

In this vignette, we describe how transmission networks can be reconstructed from *phyloscanner* output with the R package `phyloscannerR`.

This will involve the following steps:

1. Find pairs of individuals between whom phylogenetic linkage cannot be excluded based on the distance and topological relationship of viral reads from both individuals. This will use output from a previous *phyloscanner* analysis.
2. Inspect the *phyloscanner* statistics for each deep-sequence tree for these pairs.
3. Inspect the *phyloscanner* relationship counts across all deep-sequence tree for these pairs.
4. Reconstruct distinct transmission networks between these pairs, and reconstruct the most likely transmission chain for each network.
5. Plot the transmission networks and the transmission chains.

We will introduce the following functions from the `phyloscannerR` package:

- `find.pairs.in.networks`
- `produce.pairwise.graphs2`
- `find.networks`
- `plot.network`
- `plot.chain`

Data

The data that we will be using in this vignette is from the MRC UVRI cohort in south-eastern Uganda, and was generated by the PANGEA consortium.

Getting started

We assume that output from a previous *phyloscanner* analysis is available, containing files that end in `*workspace.rda`. We will need only these files to reconstruct the transmission networks. Let us get started.

```r
# load packages
require(tidyverse)
require(RBGL)
require(igraph)
require(network)
require(ggnet)
require(phyloscannerR)

# directory with phyloscanner output containing the *workspace.rda file
home <- "/Users/Oliver/Box Sync/OR_Work/2019/2019_PANGEA_BBosa"
indir <- file.path(home, "MRCPopSample_phsc_stage2_output_newali_300_HKC_phsc")

# output files
```

```r
outdir <- file.path(home, "MRCPopSample_phsc_stage2_output_newali_300_HKC_analysis")
file.pairs <- file.path(outdir, "MRCUVRI_phscallpairs_190827.rda")
file.nets <- file.path(outdir, "MRCUVRI_phscnetworks_190827.rda")
```

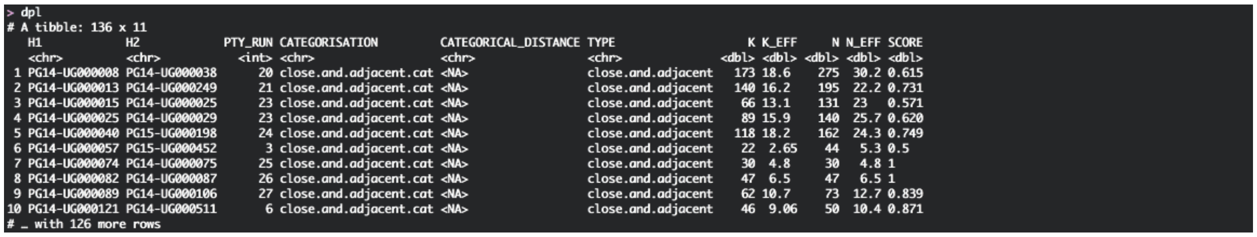
Pairs of individuals between whom linkage is not excluded

We will now processes the *phyloscanner* output in `indir`. We will use the *phyloscanner* specification as used for the Rakai analysis. The next lines of code specify a list of `control` options, and then use function `find.pairs.in.networks`.

```r
#   control options
#
#   phyloscanner produces several different phylogenetic classifications
#   schemes; for example by phylogenetic distance only; or using phylogenetic
#   distance and topologic adjacency (`close.and.adjacent.cat`); or phylogenetic
#   distance and topologic contiguity. We select one of them.
#
control <- list(    linked.group='close.and.adjacent.cat',
#   ( use classification based on phylogenetic distance and topological adjacency )
                    linked.no='not.close.or.nonadjacent',
#   ( pairs are interpreted to be unlinked
#     if classified as 'not.close.or.nonadjacent' )
                    linked.yes='close.and.adjacent',
#   ( pairs are interpreted to be linked
#     if classified as 'close.or.nonadjacent' )
                    conf.cut=0.6,
#   ( threshold on the proportion of deep-sequence phylogenies
#     pairs unlinked in more than this are discarded )
                    neff.cut=3
#   ( threshold on the effective number of deep-sequence phylogenies
#     pairs with fewer data are discarded )
                    )

tmp <- find.pairs.in.networks(  indir,
                                batch.regex='^ptyr([0-9]+)_.*',
#   ( batch.regex identifies the batch number from the
#     file names of *phyloscanner* output. Typically, this batch number corresponds
#     to the analysis of a particular transmission network. )
                                control=control,
                                verbose=TRUE)

dpl <- copy(tmp$network.pairs)
#   the pairs
dc <- copy(tmp$relationship.counts)
#   their relationship counts summed over phylogenies
dw <- copy(tmp$windows)
#   their relationship stats for all phylogenies

#save(dpl, dc, dw, file=file.pairs)
```

Output of `find.pairs.in.networks`

Let us have a look at the selected pairs:

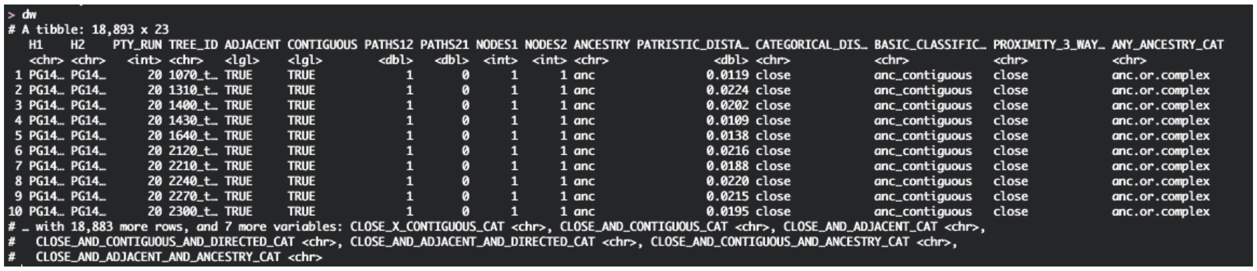


```
> dpl
# A tibble: 136 x 11
   H1           H2           PTY_RUN CATEGORISATION         CATEGORICAL_DISTANCE TYPE                 K K_EFF     N N_EFF SCORE
   <chr>        <chr>          <int> <chr>                  <chr>                <chr>            <dbl> <dbl> <dbl> <dbl> <dbl>
 1 PG14-UG000008 PG14-UG000038      20 close.and.adjacent.cat <NA>                 close.and.adjacent   173 18.6    275  30.2 0.615
 2 PG14-UG000013 PG14-UG000249      21 close.and.adjacent.cat <NA>                 close.and.adjacent   140 16.2    195  22.2 0.731
 3 PG14-UG000015 PG14-UG000025      23 close.and.adjacent.cat <NA>                 close.and.adjacent    66 13.1    131  23   0.571
 4 PG14-UG000025 PG14-UG000029      23 close.and.adjacent.cat <NA>                 close.and.adjacent    89 15.9    140  25.7 0.620
 5 PG14-UG000040 PG15-UG000198      24 close.and.adjacent.cat <NA>                 close.and.adjacent   118 18.2    162  24.3 0.749
 6 PG14-UG000057 PG15-UG000452       3 close.and.adjacent.cat <NA>                 close.and.adjacent    22  2.65    44   5.3 0.5
 7 PG14-UG000074 PG14-UG000075      25 close.and.adjacent.cat <NA>                 close.and.adjacent    30  4.8     30   4.8 1
 8 PG14-UG000082 PG14-UG000087      26 close.and.adjacent.cat <NA>                 close.and.adjacent    47  6.5     47   6.5 1
 9 PG14-UG000089 PG14-UG000106      27 close.and.adjacent.cat <NA>                 close.and.adjacent    62 10.7     73  12.7 0.839
10 PG14-UG000121 PG14-UG000511       6 close.and.adjacent.cat <NA>                 close.and.adjacent    46  9.06    50  10.4 0.871
# … with 126 more rows
```

Here,

1. Columns `H1` and `H2` list the individual ID of the individuals between whom linkage is not rejected.
2. Column `PTY_RUN` lists the batch number of the phyloscanner analysis specified through `batch.regex`.
3. Column `NEFF` gives the total number of deep-sequence phylogenies for the pair, and `KEFF` gives the total number of phylogenies in which the two individuals have virus that is `close.and.adjacent`.

Now let us have a look at the relationship stats for all phylogenies:

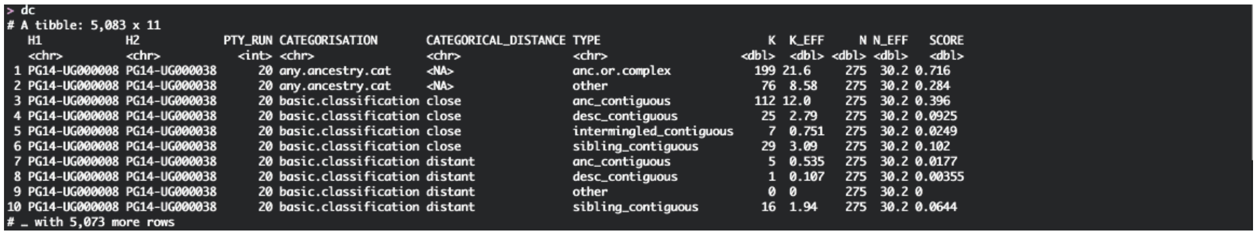


```
> dw
# A tibble: 18,893 x 23
   H1    H2    PTY_RUN TREE_ID ADJACENT CONTIGUOUS PATHS12 PATHS21 NODES1 NODES2 ANCESTRY PATRISTIC_DISTA… CATEGORICAL_DIS… BASIC_CLASSIFIC… PROXIMITY_3_WAY… ANY_ANCESTRY_CAT
   <chr> <chr>   <int> <chr>   <lgl>    <lgl>        <dbl>   <dbl>  <int>  <int> <chr>               <dbl> <chr>            <chr>            <chr>            <chr>
 1 PG14… PG14…      20 1070_t… TRUE     TRUE             1       0      1      1 anc                0.0119 close            anc_contiguous   close            anc.or.complex
 2 PG14… PG14…      20 1310_t… TRUE     TRUE             1       0      1      1 anc                0.0224 close            anc_contiguous   close            anc.or.complex
 3 PG14… PG14…      20 1400_t… TRUE     TRUE             1       0      1      1 anc                0.0202 close            anc_contiguous   close            anc.or.complex
 4 PG14… PG14…      20 1430_t… TRUE     TRUE             1       0      1      1 anc                0.0109 close            anc_contiguous   close            anc.or.complex
 5 PG14… PG14…      20 1640_t… TRUE     TRUE             1       0      1      1 anc                0.0138 close            anc_contiguous   close            anc.or.complex
 6 PG14… PG14…      20 2120_t… TRUE     TRUE             1       0      1      1 anc                0.0216 close            anc_contiguous   close            anc.or.complex
 7 PG14… PG14…      20 2210_t… TRUE     TRUE             1       0      1      1 anc                0.0188 close            anc_contiguous   close            anc.or.complex
 8 PG14… PG14…      20 2240_t… TRUE     TRUE             1       0      1      1 anc                0.0220 close            anc_contiguous   close            anc.or.complex
 9 PG14… PG14…      20 2270_t… TRUE     TRUE             1       0      1      1 anc                0.0215 close            anc_contiguous   close            anc.or.complex
10 PG14… PG14…      20 2300_t… TRUE     TRUE             1       0      1      1 anc                0.0195 close            anc_contiguous   close            anc.or.complex
# … with 18,883 more rows, and 7 more variables: CLOSE_X_CONTIGUOUS_CAT <chr>, CLOSE_AND_CONTIGUOUS_CAT <chr>, CLOSE_AND_ADJACENT_CAT <chr>,
#   CLOSE_AND_CONTIGUOUS_AND_DIRECTED_CAT <chr>, CLOSE_AND_ADJACENT_AND_DIRECTED_CAT <chr>, CLOSE_AND_CONTIGUOUS_AND_ANCESTRY_CAT <chr>,
#   CLOSE_AND_ADJACENT_AND_ANCESTRY_CAT <chr>
```

Here,

1. Columns `H1`, `H2` and `PTY_RUN` are as before.
2. Column `TREE_ID` gives the identifier of the phylogeny for which the *phyloscanner* statistics are evaluated.
3. Columns `ADJACENT`, `CONTIGUOUS`, `PATHS12`, `PATHS21`, `ANCESTRY`, `PATRISTIC_DISTANCE` give the basic *phyloscanner* statistics that describe the relationship of virus from two individuals in this phylogeny.
4. The remaining columns list all the different classification schemes that *phyloscanner* produces by default (e.g. `PROXIMITY-3_WAY_CAT`), and the classifications in rows (e.g. `close`).

Now let us have a look at the relationship counts summed over all phylogenies:



```
> dc
# A tibble: 5,083 x 11
   H1           H2           PTY_RUN CATEGORISATION       CATEGORICAL_DISTANCE TYPE                        K  K_EFF     N N_EFF   SCORE
   <chr>        <chr>          <int> <chr>                <chr>                <chr>                   <dbl>  <dbl> <dbl> <dbl>   <dbl>
 1 PG14-UG000008 PG14-UG000038      20 any.ancestry.cat     <NA>                 anc.or.complex            199 21.6     275  30.2 0.716
 2 PG14-UG000008 PG14-UG000038      20 any.ancestry.cat     <NA>                 other                      76  8.58    275  30.2 0.284
 3 PG14-UG000008 PG14-UG000038      20 basic.classification close                anc_contiguous            112 12.0     275  30.2 0.396
 4 PG14-UG000008 PG14-UG000038      20 basic.classification close                desc_contiguous            25  2.79    275  30.2 0.0925
 5 PG14-UG000008 PG14-UG000038      20 basic.classification close                intermingled_contiguous     7  0.751   275  30.2 0.0249
 6 PG14-UG000008 PG14-UG000038      20 basic.classification close                sibling_contiguous         29  3.09    275  30.2 0.102
 7 PG14-UG000008 PG14-UG000038      20 basic.classification distant              anc_contiguous              5  0.535   275  30.2 0.0177
 8 PG14-UG000008 PG14-UG000038      20 basic.classification distant              desc_contiguous             1  0.107   275  30.2 0.00355
 9 PG14-UG000008 PG14-UG000038      20 basic.classification distant              other                       0  0       275  30.2 0
10 PG14-UG000008 PG14-UG000038      20 basic.classification distant              sibling_contiguous         16  1.94    275  30.2 0.0644
# … with 5,073 more rows
```

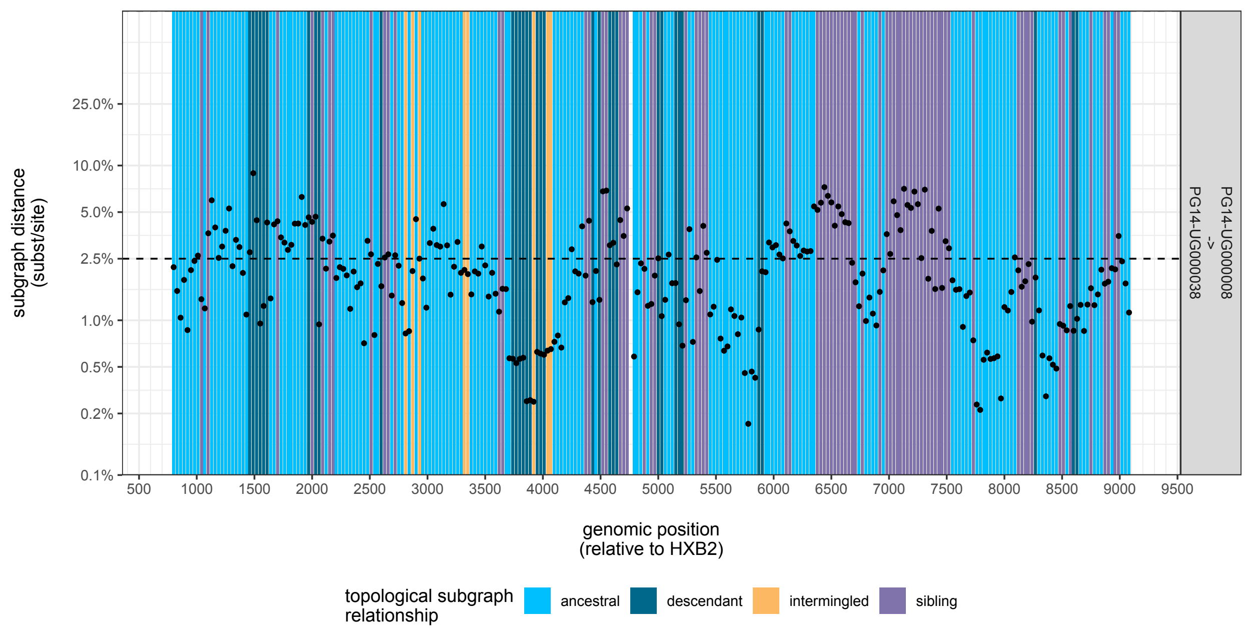
Here,

1. Columns `H1`, `H2` and `PTY_RUN` are as before.
2. Column `CATEGORISATION` then lists one of the classification schemes and column `TYPE` all of the classifications in this scheme.
3. For each classification, `NEFF` gives the total number of deep-sequence phylogenies for the pair, and `KEFF` gives the total number of phylogenies in which the two individuals have that classification.

Plotting the phylogenetic relationships between pairs of individuals

At this point, we can easily visualise the phylogenetic relationships between two individuals across the genome. We call these plots phyloscans:

```
# plot phyloscans of all likely pairs
hosts <- dw %>% select(H1, H2) %>% gather("HOST_TYPE", "H") %>% select(-HOST_TYPE) %>%
    distinct() %>% arrange(H) %>% pull(H)
# plot phyloscans of one pair
hosts <- dw[1, ] %>% select(H1, H2) %>% gather("HOST_TYPE", "H") %>% select(-HOST_TYPE) %>%
    pull(H)
tmp <- copy(dw)
tmp <- produce.pairwise.graphs2(NULL, hosts = hosts, dwin = tmp, inclusion = "both")
tmp$graph
```



Reconstructing transmission networks

We can now reconstruct transmission networks, which are defined as sets of individuals between whom phylogenetic linkage is not excluded. Between any two individuals in a network, there are three weighted edges that describe the phylogenetic support for transmission in the one direction, in the other direction, and support for phylogenetic linkage without evidence for directionality.

The same function also reconstructs the most likely transmission chain for each network. A transmission chain is defined as the directed graph with nodes of indegree 1 and arbitrary outdegree that connects all individuals and has the largest product of edge weights. Full details are given in the the Rakai paper.

Finally, from the transmission chains it is straightforward to extract highly supported pairs.

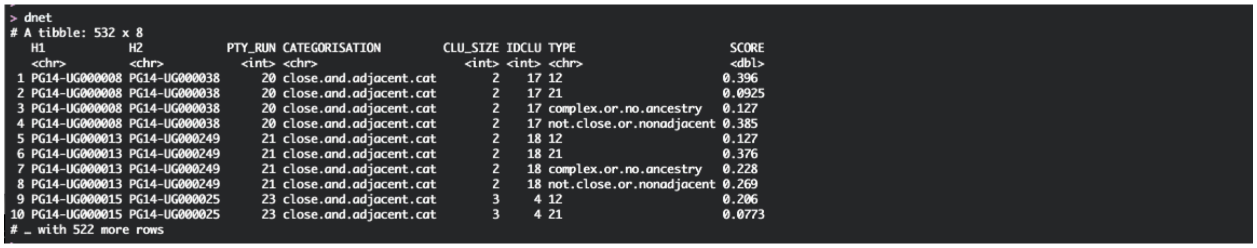
Here is the code:

```
#   construct networks between pairs using the same
#   control options as before
tmp <- find.networks(dc, control=control, verbose=TRUE)
#   extract networks and the transmission chains within them
```

```r
dnet <- copy(tmp$transmission.networks)
dchain <- copy(tmp$most.likely.transmission.chains)

#   construct highly supported pairs
conf.cut <- 0.6
dconfpairs <- dchain %>%
        filter( SCORE_LINKED>conf.cut &
                pmax(SCORE_DIR_12, SCORE_DIR_21)>conf.cut) %>%
        select(-c(SCORE_NW_12, SCORE_NW_12, SCORE_NW_AMB))

#   save
#save(dpl, dc, dw, dnet, dchain, dconfpairs, file=file.nets)
```

Output from reconstructing transmission networks

Let us have a look at the transmission networks:



```
> dnet
# A tibble: 532 x 8
   H1           H2           PTY_RUN CATEGORISATION         CLU_SIZE IDCLU TYPE                      SCORE
   <chr>        <chr>          <int> <chr>                     <int> <int> <chr>                     <dbl>
 1 PG14-UG000008 PG14-UG000038      20 close.and.adjacent.cat        2    17 12                        0.396
 2 PG14-UG000008 PG14-UG000038      20 close.and.adjacent.cat        2    17 21                        0.0925
 3 PG14-UG000008 PG14-UG000038      20 close.and.adjacent.cat        2    17 complex.or.no.ancestry    0.127
 4 PG14-UG000008 PG14-UG000038      20 close.and.adjacent.cat        2    17 not.close.or.nonadjacent  0.385
 5 PG14-UG000013 PG14-UG000249      21 close.and.adjacent.cat        2    18 12                        0.127
 6 PG14-UG000013 PG14-UG000249      21 close.and.adjacent.cat        2    18 21                        0.376
 7 PG14-UG000013 PG14-UG000249      21 close.and.adjacent.cat        2    18 complex.or.no.ancestry    0.228
 8 PG14-UG000013 PG14-UG000249      21 close.and.adjacent.cat        2    18 not.close.or.nonadjacent  0.269
 9 PG14-UG000015 PG14-UG000025      23 close.and.adjacent.cat        3     4 12                        0.206
10 PG14-UG000015 PG14-UG000025      23 close.and.adjacent.cat        3     4 21                        0.0773
# … with 522 more rows
```

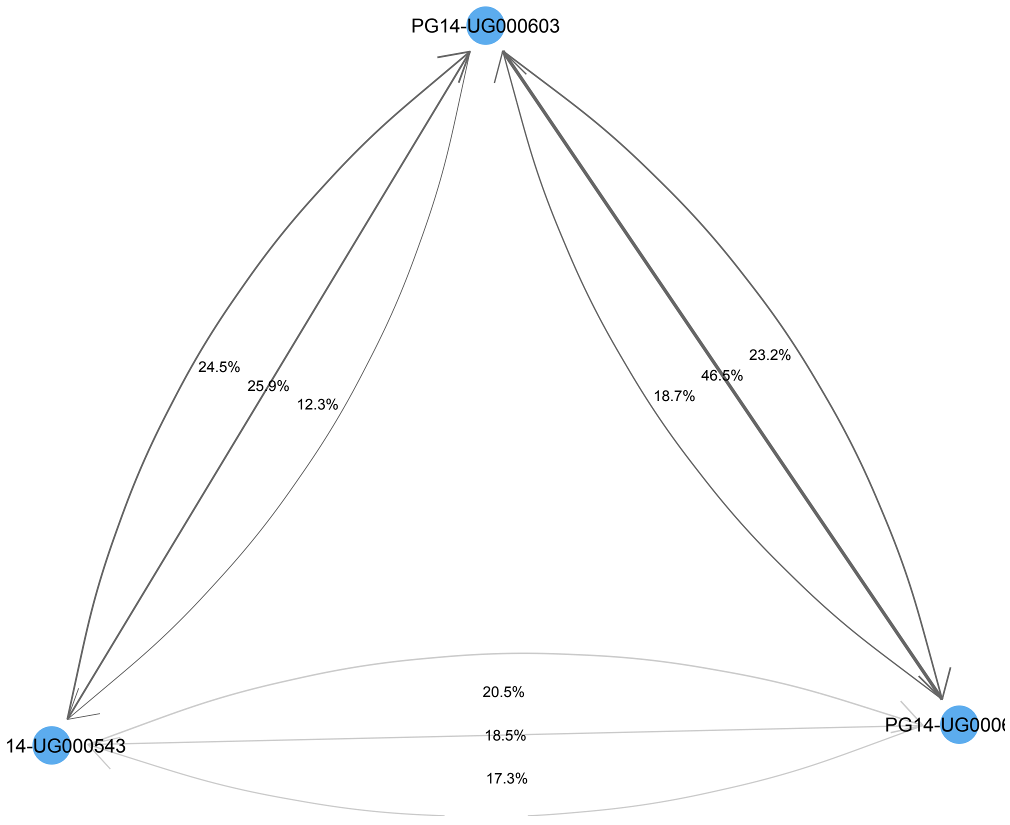
Here,

1. Columns `H1` and `H2` list as before the individual ID of the individuals between whom linkage is not rejected.
2. Column `PTY_RUN` lists as before the batch number of the phyloscanner analysis specified through `batch.regex`.
3. Column `IDCLU` gives an identifier for each reconstructed transmission network.
4. Column `TYPE` gives the possible classifications of the phylogenetic relationship types between the individuals. Using the default options in `control`, this is evidence for transmission in either direction (`12` and `21`), support for phylogenetic linkage without evidence for directionality (`complex.or.no.ancestry`), and evidence for no phylogenetic linkage (`not.close.or.nonadjacent`).
5. Column `SCORE` gives the *phyloscanner* score for each classification, which is defined by `KEFF` divided by `NEFF`.

Let us plot one of these networks. The code to do this is as follows:

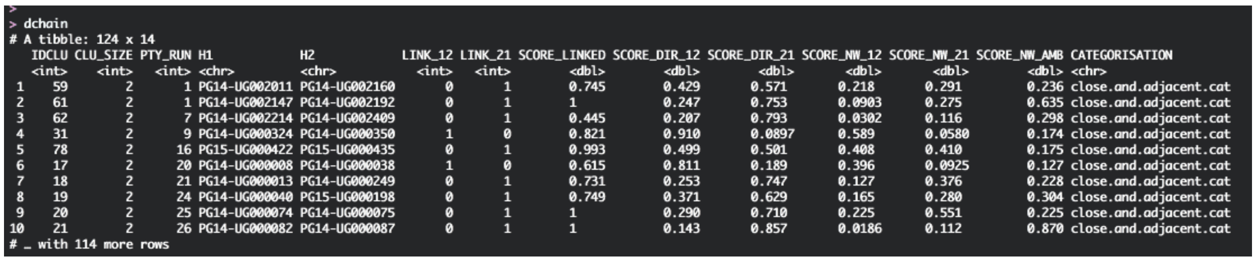
```r
# plot all networks
idclus <- sort(unique(dnet$IDCLU))
# find IDs of all networks
control<- list()
control$point.size = 10
control$edge.gap = 0.04
control$edge.size = 2
control$curvature = -0.2
control$arrow = arrow(length = unit(0.04, "npc"), type = "open")
control$curv.shift = 0.06
control$label.size = 3
# the above options may need to be changed, depending on the
# size of the networks and the size of your pdf output
```

```
control$node.label = "H"
# specify the column in 'di' below that should be used as nodel label
control$node.fill = NA_character_
control$node.fill.values = c(`NA` = "steelblue2")
# specify the background colour for each node
control$node.shape = NA_character_
control$node.shape.values = c(`NA` = 16)
# specify the shape for each node
control$threshold.linked = 0.6
# edges will be highlighted in darkgrey if 'SCORE_LINKED' is above this threshold
pns     <- lapply(seq_along(idclus), function(i)
        {
            idclu <- idclus[i]
            df <- dnet %>%
                    filter(IDCLU == idclu)
            di <- df %>%
                    select(H1, H2) %>%
                    gather('HOST_TYPE','H') %>%
                    select(-HOST_TYPE) %>%
                    distinct()
            p <- plot.network(df, di, control)
            p
        })
pns[[10]]
```



Output from reconstructing transmission chains

Now, let us have a look at the corresponding transmission chains:



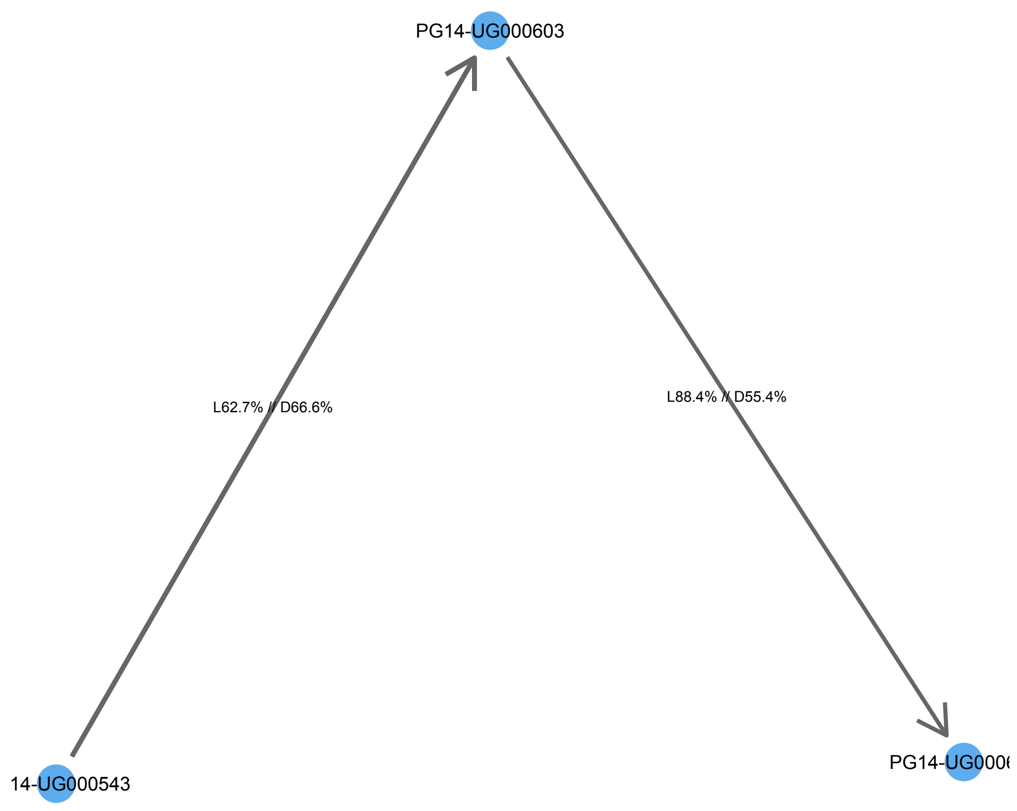
```
>
> dchain
# A tibble: 124 x 14
   IDCLU CLU_SIZE PTY_RUN H1           H2           LINK_12 LINK_21 SCORE_LINKED SCORE_DIR_12 SCORE_DIR_21 SCORE_NW_12 SCORE_NW_21 SCORE_NW_AMB CATEGORISATION
   <int>    <int>   <int> <chr>        <chr>          <int>   <int>        <dbl>        <dbl>        <dbl>       <dbl>       <dbl>        <dbl> <chr>
 1    59        2       1 PG14-UG002011 PG14-UG002160       0       1        0.745        0.429        0.571      0.218       0.291         0.236 close.and.adjacent.cat
 2    61        2       1 PG14-UG002147 PG14-UG002192       0       1        1            0.247        0.753      0.0903      0.275         0.635 close.and.adjacent.cat
 3    62        2       7 PG14-UG002214 PG14-UG002409       0       1        0.445        0.207        0.793      0.0302      0.116         0.298 close.and.adjacent.cat
 4    31        2       9 PG14-UG000324 PG14-UG000350       1       0        0.821        0.910        0.0897     0.589       0.0580        0.174 close.and.adjacent.cat
 5    78        2      16 PG15-UG000422 PG15-UG000435       0       1        0.993        0.499        0.501      0.408       0.410         0.175 close.and.adjacent.cat
 6    17        2      20 PG14-UG000008 PG14-UG000038       1       0        0.615        0.811        0.189      0.396       0.0925        0.127 close.and.adjacent.cat
 7    18        2      21 PG14-UG000013 PG14-UG000249       0       1        0.731        0.253        0.747      0.127       0.376         0.228 close.and.adjacent.cat
 8    19        2      24 PG14-UG000040 PG15-UG000198       0       1        0.749        0.371        0.629      0.165       0.280         0.304 close.and.adjacent.cat
 9    20        2      25 PG14-UG000074 PG14-UG000075       0       1        1            0.290        0.710      0.225       0.551         0.225 close.and.adjacent.cat
10    21        2      26 PG14-UG000082 PG14-UG000087       0       1        1            0.143        0.857      0.0186      0.112         0.870 close.and.adjacent.cat
# … with 114 more rows
```

Here,

1. Columns `H1`, `H2`, `PTY_RUN` and `IDCLU` are as before.
2. Column `LINK_12` states if there is a directed edge from `H1` to `H2` in the most likely transmission chain, and `LINK_21` states if there is a directed edge in the other direction.
3. Column `SCORE_LINKED` gives the *phyloscanner* score for phylogenetic linkage. Using the default options, this is the sum of `KEFF` for `12`, `21` and `complex.or.no.ancestry`, divided by `NEFF`.
4. Column `SCORE_DIR_12` gives the *phyloscanner* score for transmission direction `H1` to `H2` among phylogenies supporting phylogenetic linkage. Column `SCORE_DIR_21` gives the *phyloscanner* score for transmission in the opposite direction.

We can also plot the transmission chains. The code to do this is as follows:

```
# plot corresponding most likely chains
pcs     <- lapply(seq_along(idclus), function(i)
        {
            idclu <- idclus[i]
            control$layout <- pns[[i]][['layout']]
            df <- dchain %>%
                    filter(IDCLU == idclu)
            di <- df %>%
                    select(H1, H2) %>%
                    gather('HOST_TYPE','H') %>%
                    select(-HOST_TYPE) %>%
                    distinct()
            p <- plot.chain(df, di, control)
            p
        })
pcs[[10]]
```



Final notes

1. It is also easy to add any further individual-level meta-data to the analysis output. Specify the `dmeta` input variable `find.pairs.in.networks`.