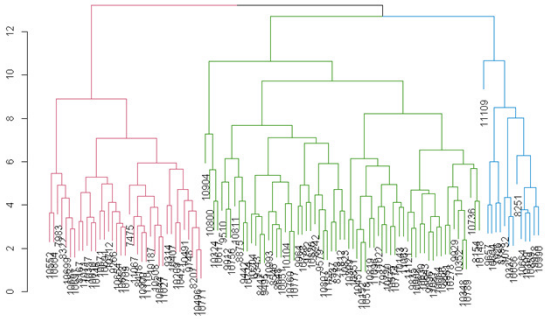
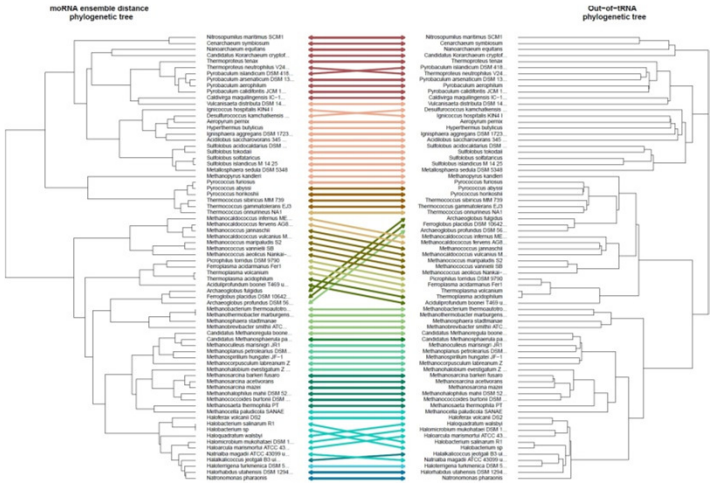
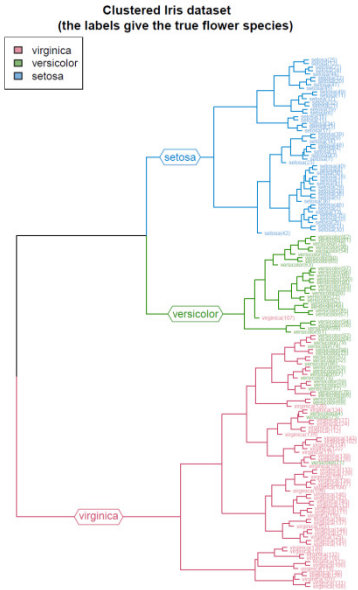
useR!2014

dendextend: an R package for easier manipulation and visualization of dendrograms

Tal.Galili@gmail.com







Talk outline:

- hclust => dendrogram
- Using ***dendextend*** for
  - Manipulation
  - Visualization
  - Comparison
- Speed (***dendextend****Rcpp*)

Toy data: the *precip* dataset

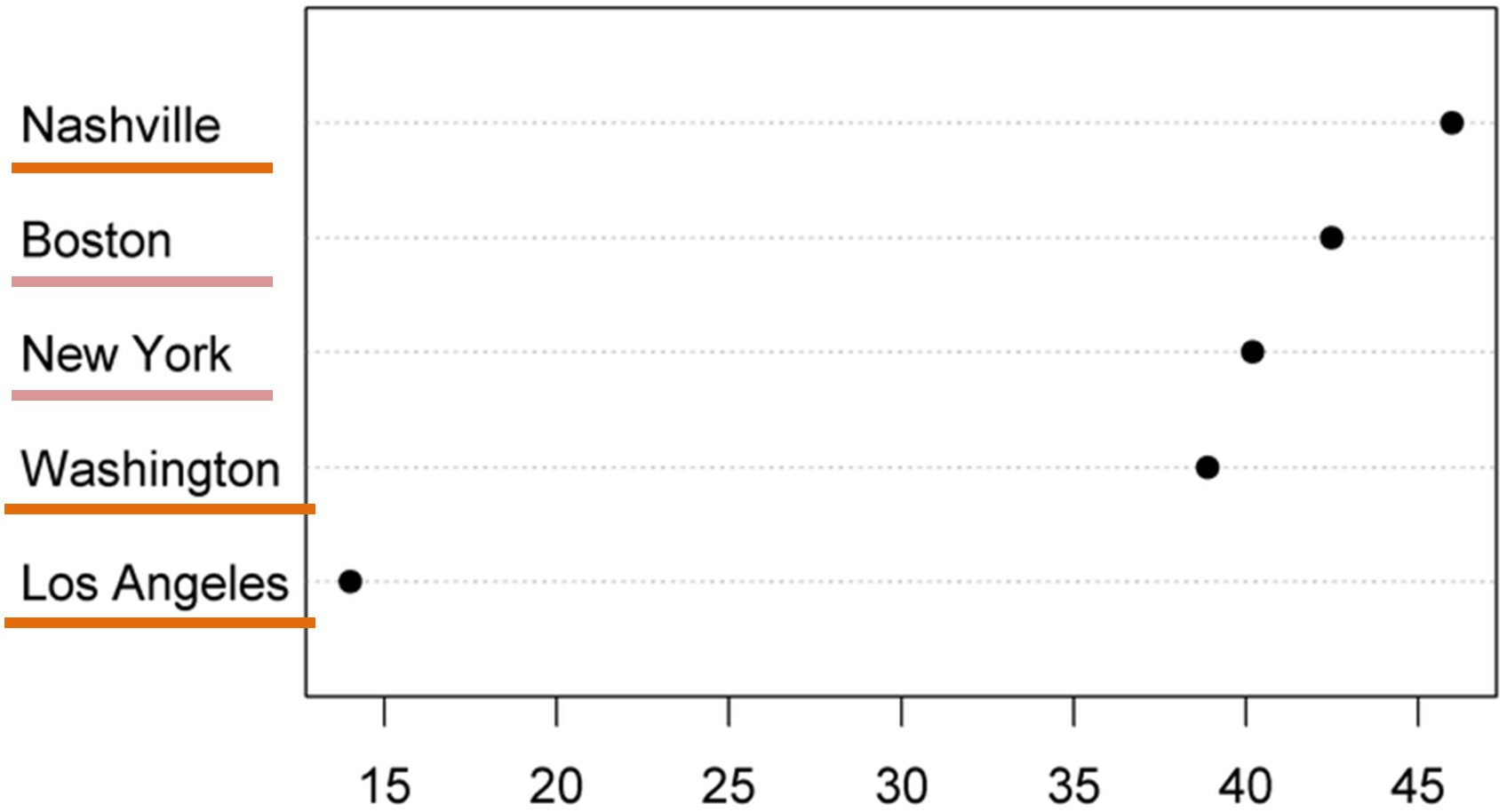
The average amount of rainfall (**precip**itation) in inches, for many US cities (in 1975).

```
data(precip)
precip_data <- precip[c("Washington", "Nashville", "Los Angeles", "Boston", "New York")]
precip_data
```

```
##  Washington   Nashville Los Angeles      Boston    New York
##        38.9        46.0        14.0        42.5        40.2
```

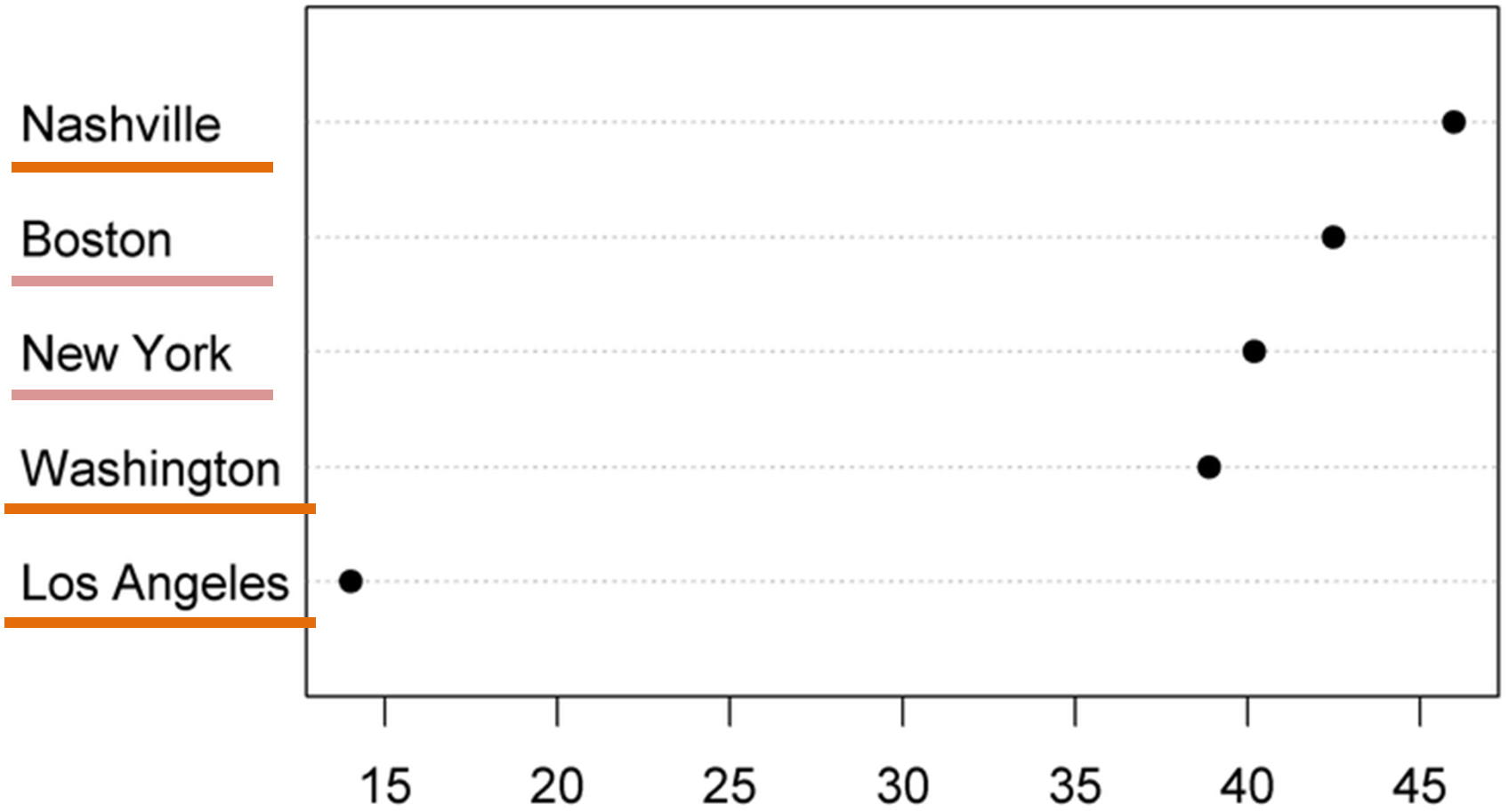
Toy data: the *precip* dataset

Let's quickly visualize it:



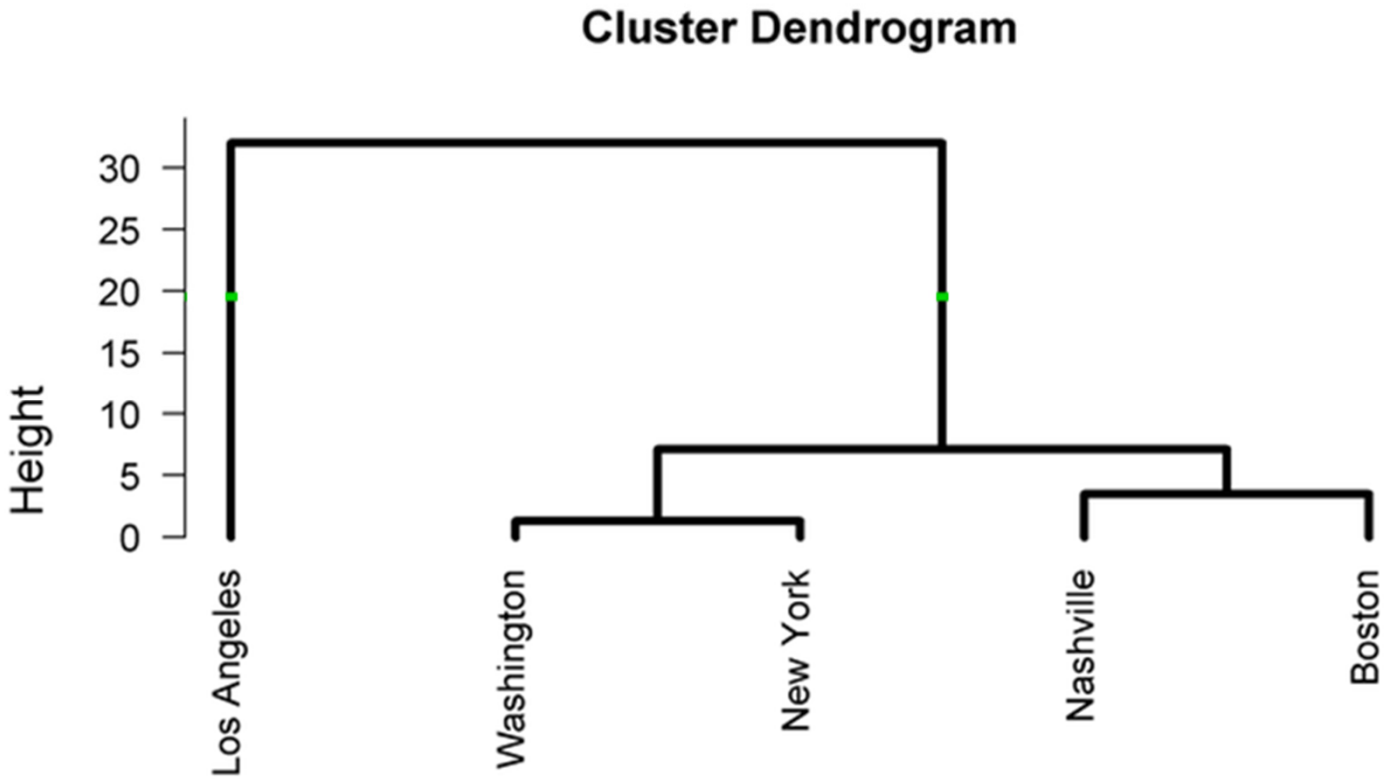
Toy data: the *precip* dataset

Let's quickly visualize it.



Toy data: the *precip* dataset

Let's quickly visualize it. And again, but with hierarchical clustering



Toy data: the *precip* dataset

Let's quickly visualize it. And again, but with hierarchical clustering

The code for it:

```
x_dist <- dist(precip_data, diag=TRUE)
```

```
hc1 <- hclust(x_dist)
```

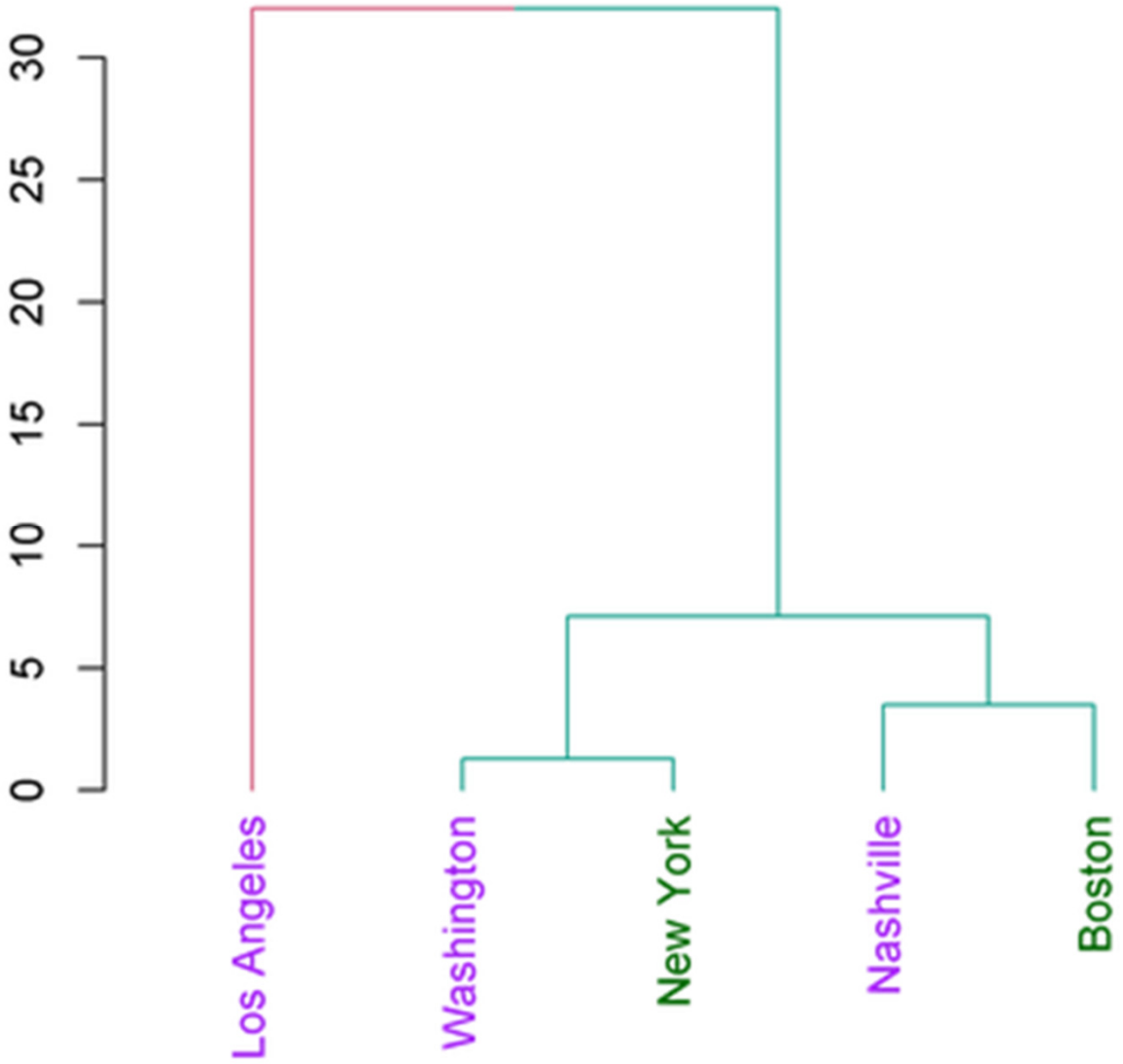
```
plot(hc1, hang = -1)
```

(+...)

Lessons (so far):

- hclust - Good for creating hierarchical clustering, but limited for plotting

Goal 1: more colors



Goal 1: more colors

1. From ***hclust*** to ***dendrogram***

Goal 1: more colors

We first coerce hclust into a dendrogram

```
dend1 <- as.dendrogram(hc1)
hc1
```

```
##
## Call:
## hclust(d = x_dist)
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 5
```

```
dend1
```

```
## 'dendrogram' with 2 branches and 5 members total, at height 32
```

Goal 1: more colors

Notice the structure of hclust

```
str(hc1)
```

```
## List of 7
##  $ merge      : int [1:4, 1:2] -1 -2 1 -3 -5 -4 2 3
##  $ height     : num [1:4] 1.3 3.5 7.1 32
##  $ order      : int [1:5] 3 1 5 2 4
##  $ labels     : chr [1:5] "Washington" "Nashville" "Los
##  $ method     : chr "complete"
##  $ call       : language hclust(d = x_dist)
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Goal 1: more colors

Notice the structure of a dendrogram object

```
str(dend1)
```

```
## --[dendrogram w/ 2 branches and 5 members at h = 32]
##   |--leaf "Los Angeles"
##   `--[dendrogram w/ 2 branches and 4 members at h = 7.1]
##      |--[dendrogram w/ 2 branches and 2 members at h = 1.3]
##      |   |--leaf "Washington"
##      |   `--leaf "New York"
##      `--[dendrogram w/ 2 branches and 2 members at h = 3.5]
##         |--leaf "Nashville"
##         `--leaf "Boston"
```

```
## 'dendrogram' with 2 branches and 5 members total, at height 32
```

Goal 1: more colors

Notice the structure of a dendrogram object

```
str(dend1)
```

```
## --[dendrogram w/
##    |--leaf "Los An
##    `--[dendrogram
##       |--[dendrogr
##       |   |--leaf "
##       |   `--leaf "
##       `--[dendrogr
##           |--leaf "
##           `--leaf "
## 'dendrogram' with 2
```

```
str(unclass(dend1))
```

```
## List of 2
##  $ : atomic [1:1] 3
##   ..- attr(*, "members")= int 1
##   ..- attr(*, "height")= num 0
##   ..- attr(*, "label")= chr "Los Angeles"
##   ..- attr(*, "leaf")= logi TRUE
##  $ :List of 2
##   ..$ :List of 2
##   .. ..$ : atomic [1:1] 1
##   .. .. ..- attr(*, "label")= chr "Washington"
##   .. .. ..- attr(*, "members")= int 1
##   .. .. ..- attr(*, "height")= num 0
##   .. .. ..- attr(*, "leaf")= logi TRUE
```

Lessons:

- hclust - Good for creating hierarchical clustering, but limited for plotting
- dendrogram object are
  - a nested list of lists
  - with attributes!

Back to Goal 1: more colors

Let's modify the ***dendrogram*** object we got to have colors!

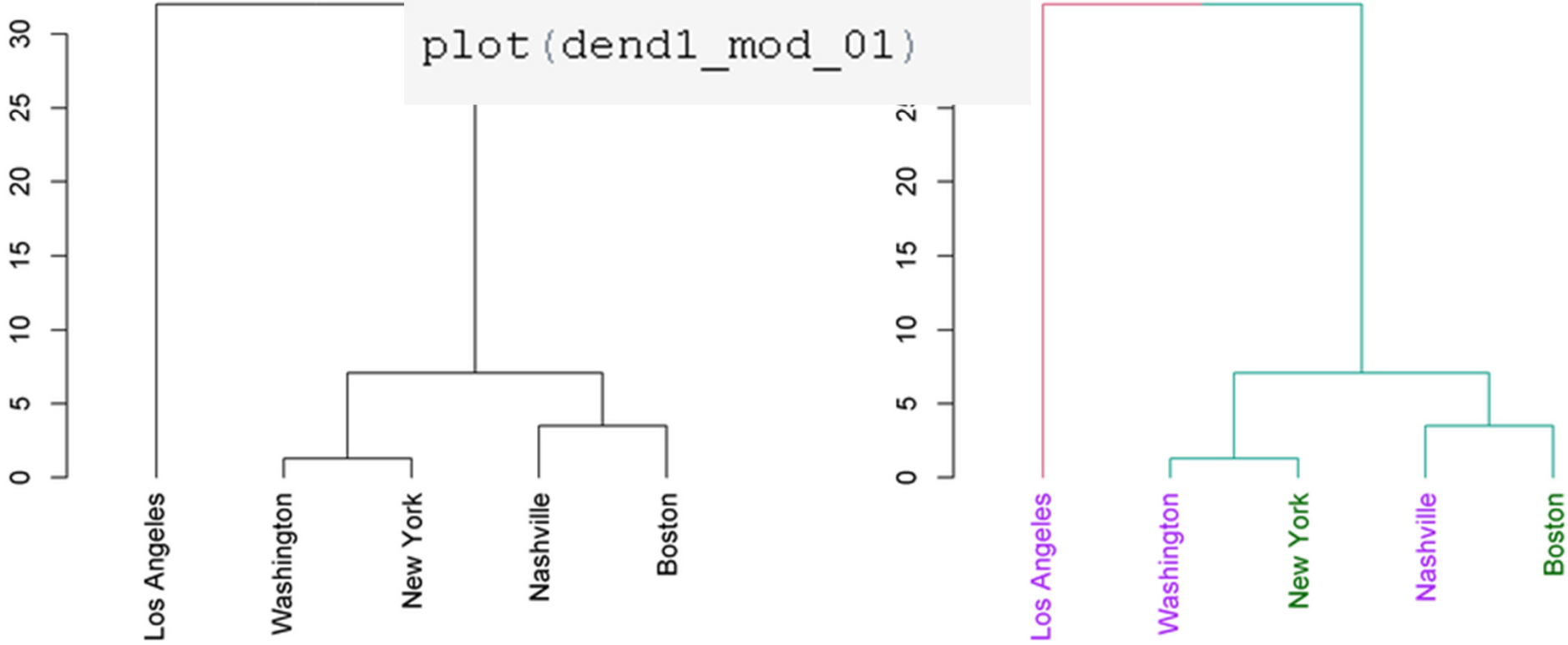
```
require(dendextend)


dend1_mod_01 <- dend1
dend1_mod_01 <- color_branches(dend1_mod_01, k =2 )
col_for_labels <- c("purple", "purple",
                    "darkgreen",  "purple",
                    "darkgreen")
dend1_mod_01 <- color_labels(dend1_mod_01,
                             col =col_for_labels)
```

Goal 1: more colors

And here is how it looks.

```
par(mfrow = c(1,2))
plot(dend1)
plot(dend1_mod_01)
```



Notice the change in the object's attributes:

Before

```
str(dend1)
```

```
## --[dendrogram w/ 2 branches and 5 members at h = 32]
##    |--leaf "Los Angeles"
```

After

```
str(dend1_mod_01)
```

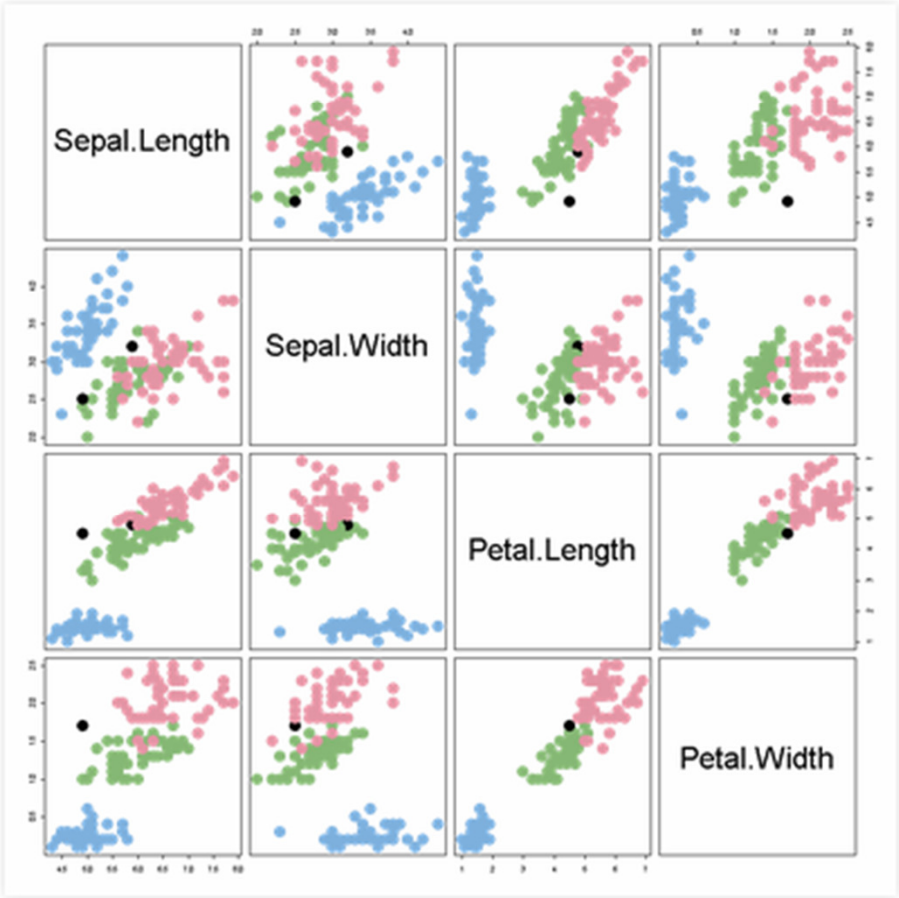
```
## --[dendrogram w/ 2 branches and 5 members at h = 32]
##    |--leaf "Los Angeles" ( edgePar = #CC476B, nodePar = c("purple", "NA") )
##    `--[dendrogram w/ 2 branches and 4 members at h = 7.1]
##       |--[dendrogram w/ 2 branches and 2 members at h = 1.3]
##       |  |--leaf "Washington" ( edgePar = #009681, nodePar = c("purple", "NA")
##       |  `--leaf "New York" ( edgePar = #009681, nodePar = c("darkgreen", "NA")
##       `--[dendrogram w/ 2 branches and 2 members at h = 3.5]
##          |--leaf "Nashville" ( edgePar = #009681, nodePar = c("purple", "NA") )
##          `--leaf "Boston" ( edgePar = #009681, nodePar = c("darkgreen", "NA") )
```

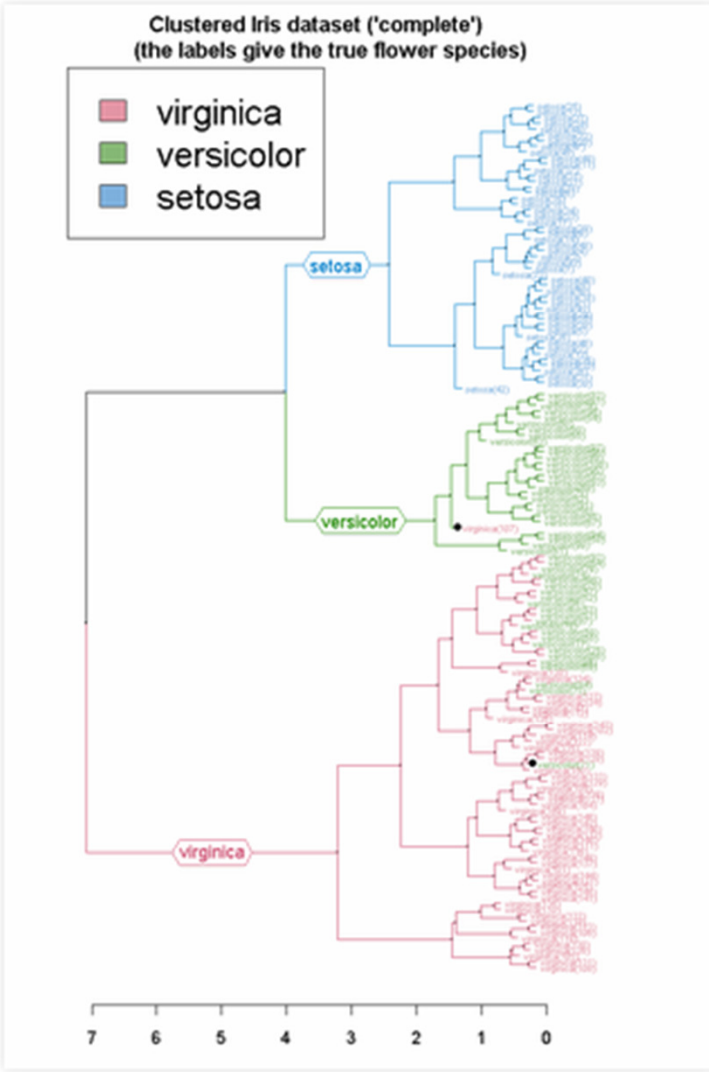
An example for nice looking dendrograms

Visually diagnosing clusters with given labels

Iris dataset- a quick example

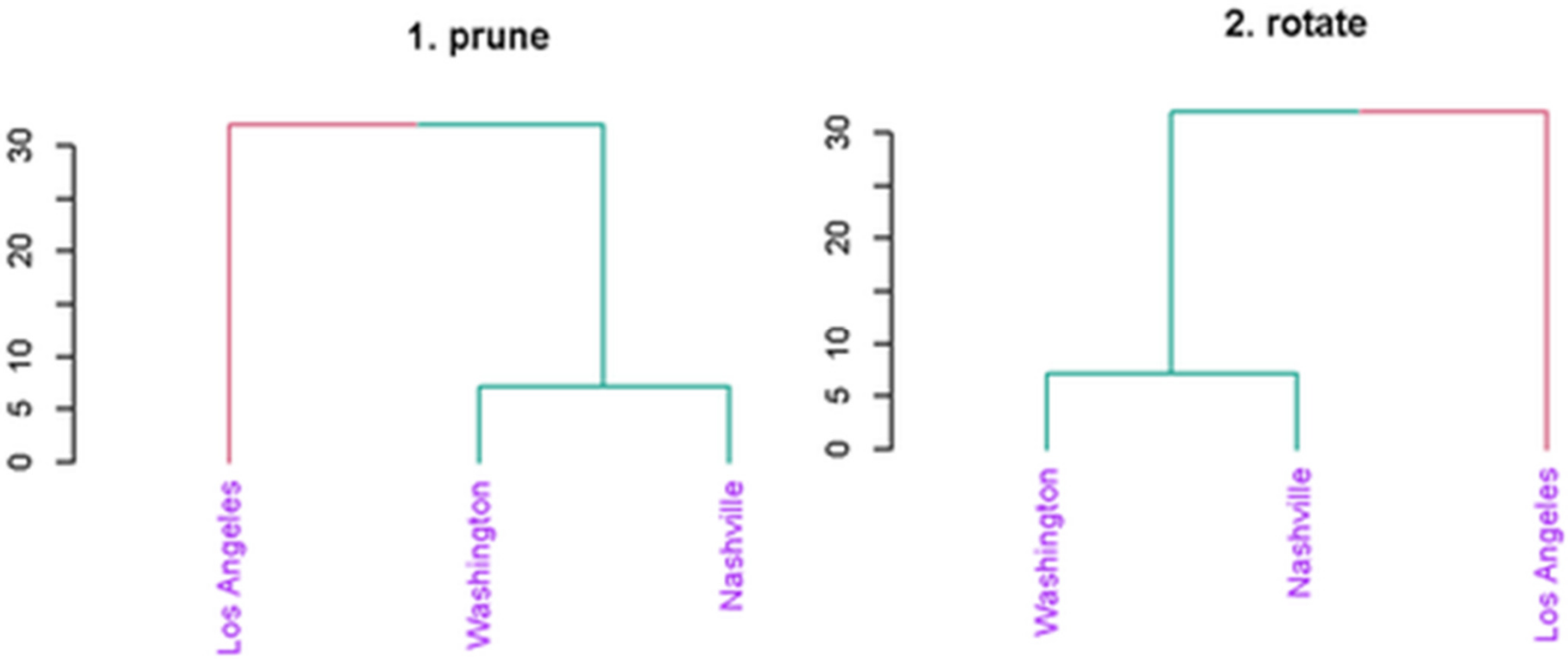
Iris dataset: - 150 items, 50 from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.



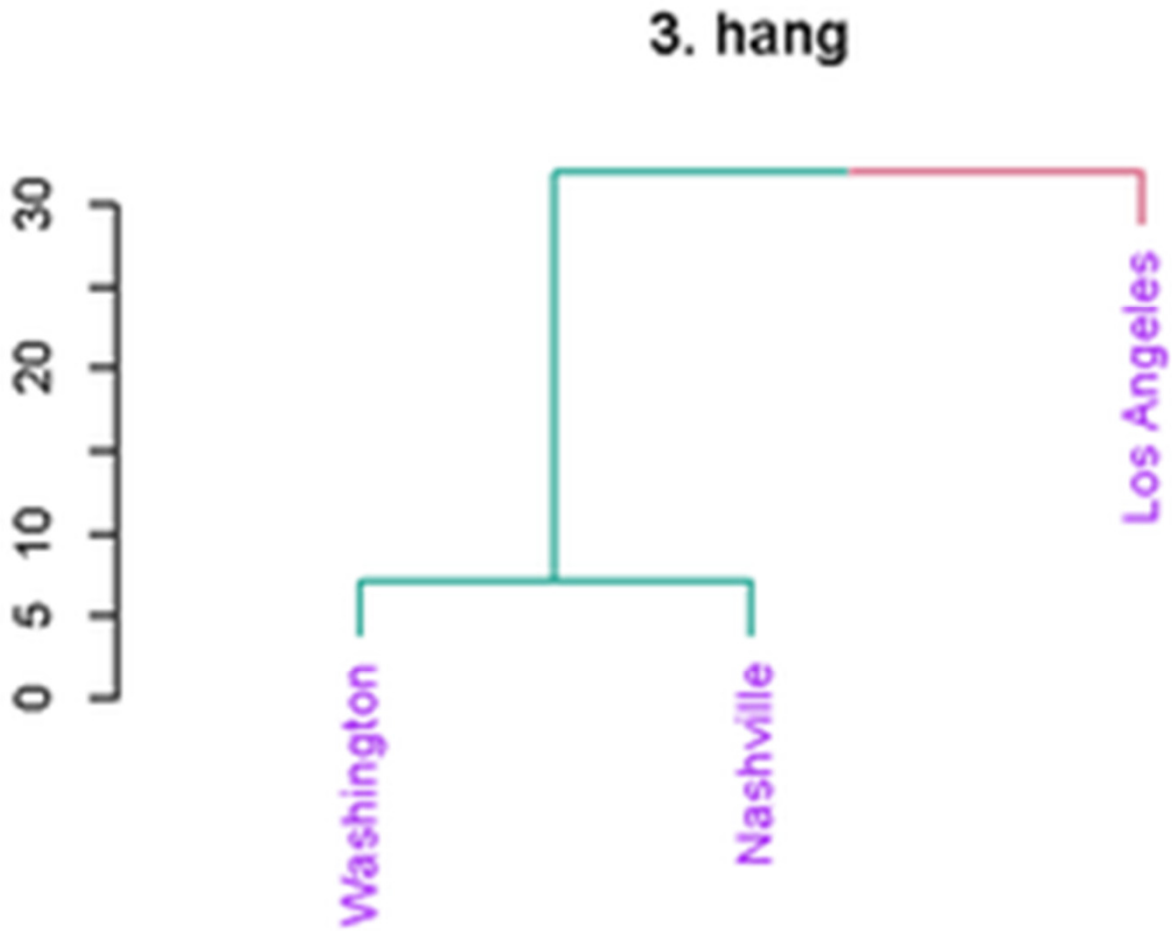


Examples of more things we could do (code comes a bit later)

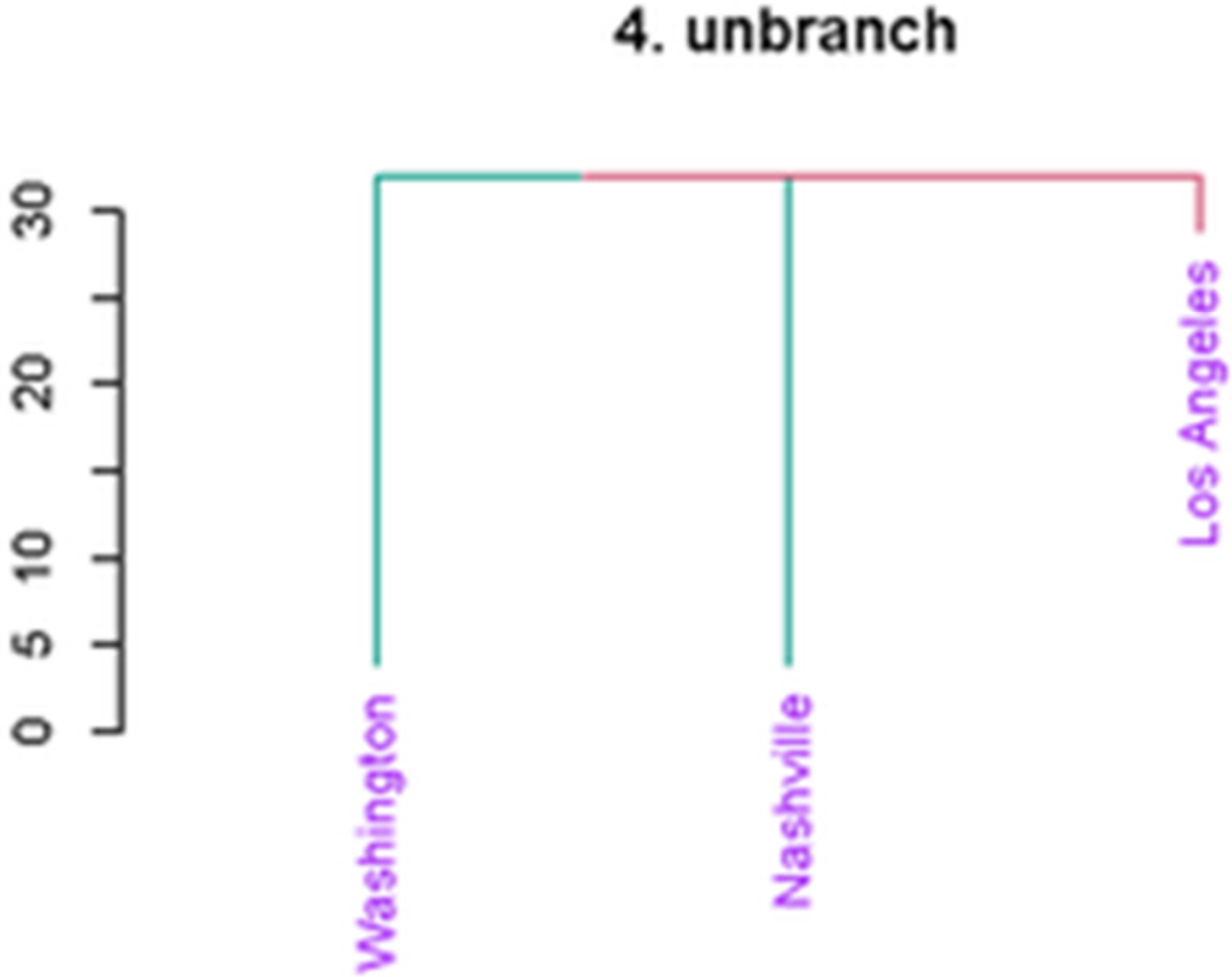
Goal 2: tweaking the tree



Goal 2: tweaking the tree



Goal 2: tweaking the tree



Goal 2: tweaking the tree

Goal 2: tweaking the tree


```
tmp1 <- prune(dend1_mod_01,
              c("New York" , "Boston"))
tmp2 <- rotate(tmp1,
               c(2,3,1))
tmp3 <- hang.dendrogram(tmp2)
tmp4 <- unbranch(tmp3)

tmp5 <- color_branches(tmp4, h = 31)
tmp5 <- color_labels(tmp5, h = 31)

tmp6 <- tmp5
labels(tmp6) <-  abbreviate(labels(tmp5),5)
```

Goal 2: tweaking the tree


```
par(mfrow = c(2,3))
plot(tmp1, main = "1. prune")
plot(tmp2, main = "2. rotate")
plot(tmp3, main = "3. hang")
plot(tmp4, main = "4. unbranch")
plot(tmp5, main = "5. Color branches/labels \n(for non-binary tree)")
plot(tmp6, main = "6. Change labels")
```

Lessons:

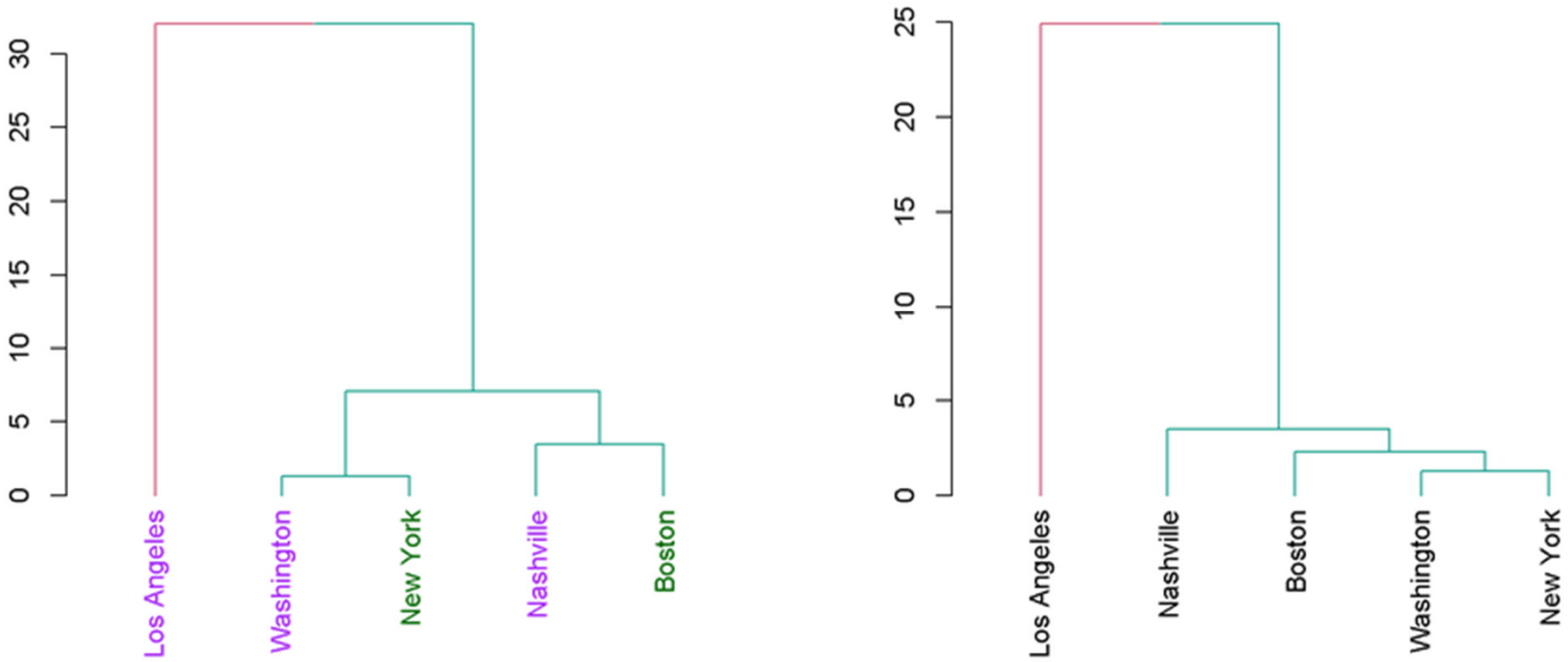
- hclust - Good for creating hierarchical clustering, but limited for plotting
- dendrogram object
  - a nested list of lists
  - with attributes!
  - should be modified step by step before plotting

Let’s create another tree this time with a different method.

And then – let’s try to compare our two tree to one another….

Goal 3: comparing trees

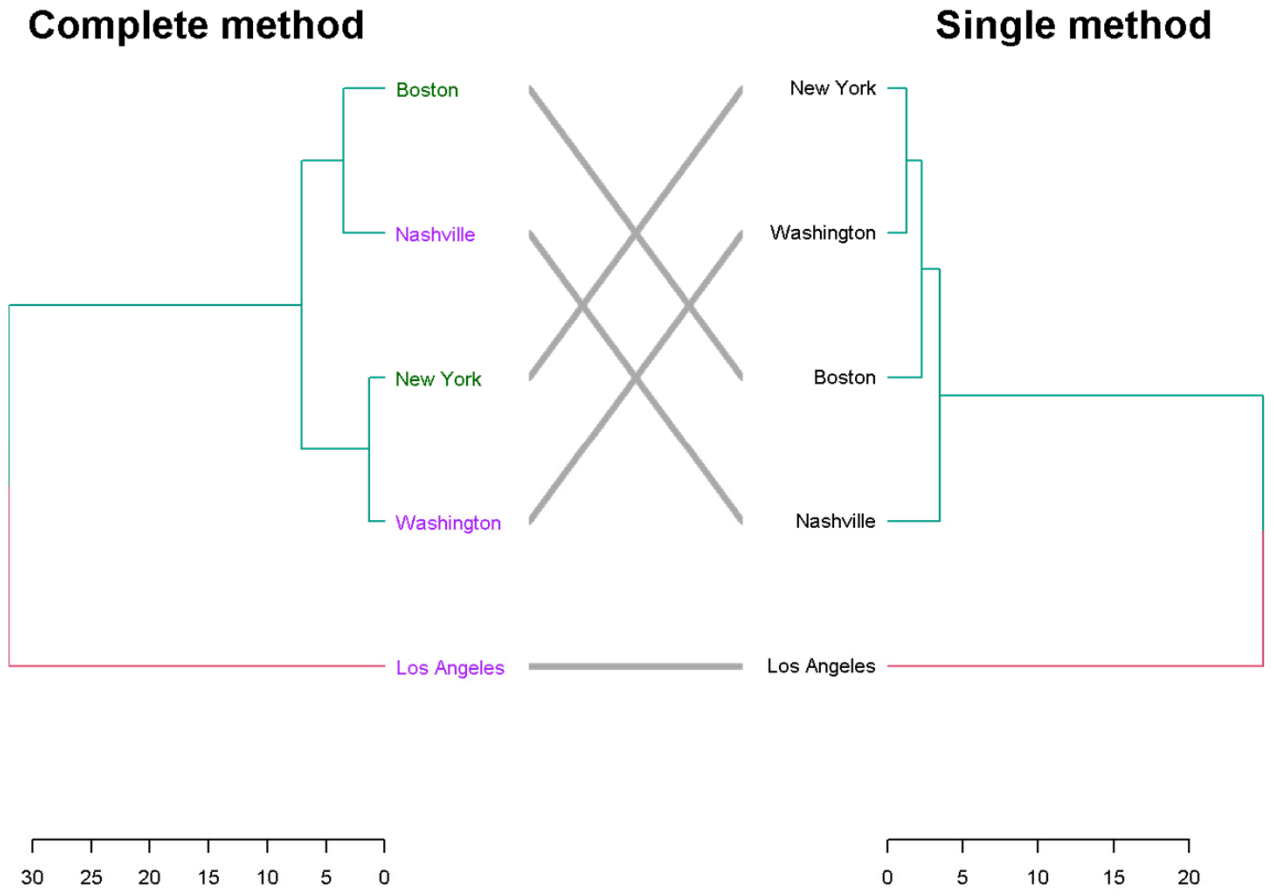
```
hc2 <- hclust(x_dist, method = "single")
dend2 <- as.dendrogram(hc2)
dend2_mod_01 <- color_branches(dend2, k = 2)
par(mfrow = c(1,2))
plot(dend1_mod_01)
plot(dend2_mod_01)
```



But we can do better using tanglegrams!

Goal 3: comparing trees

```
tanglegram(dend1_mod_01,dend2_mod_01,
           margin_inner = 6, columns_width = c(5,2,5),
           main_left = "Complete method", main_right = "Single method",
           cex_main = 2)
```



We notice that the lines are not aligned. We can try to rotate the trees to better align them…

Goal 3: comparing trees

```
dend12_untang <- untangle_step_rotate_2side(dend1_mod_01,dend2_mod_01)
```

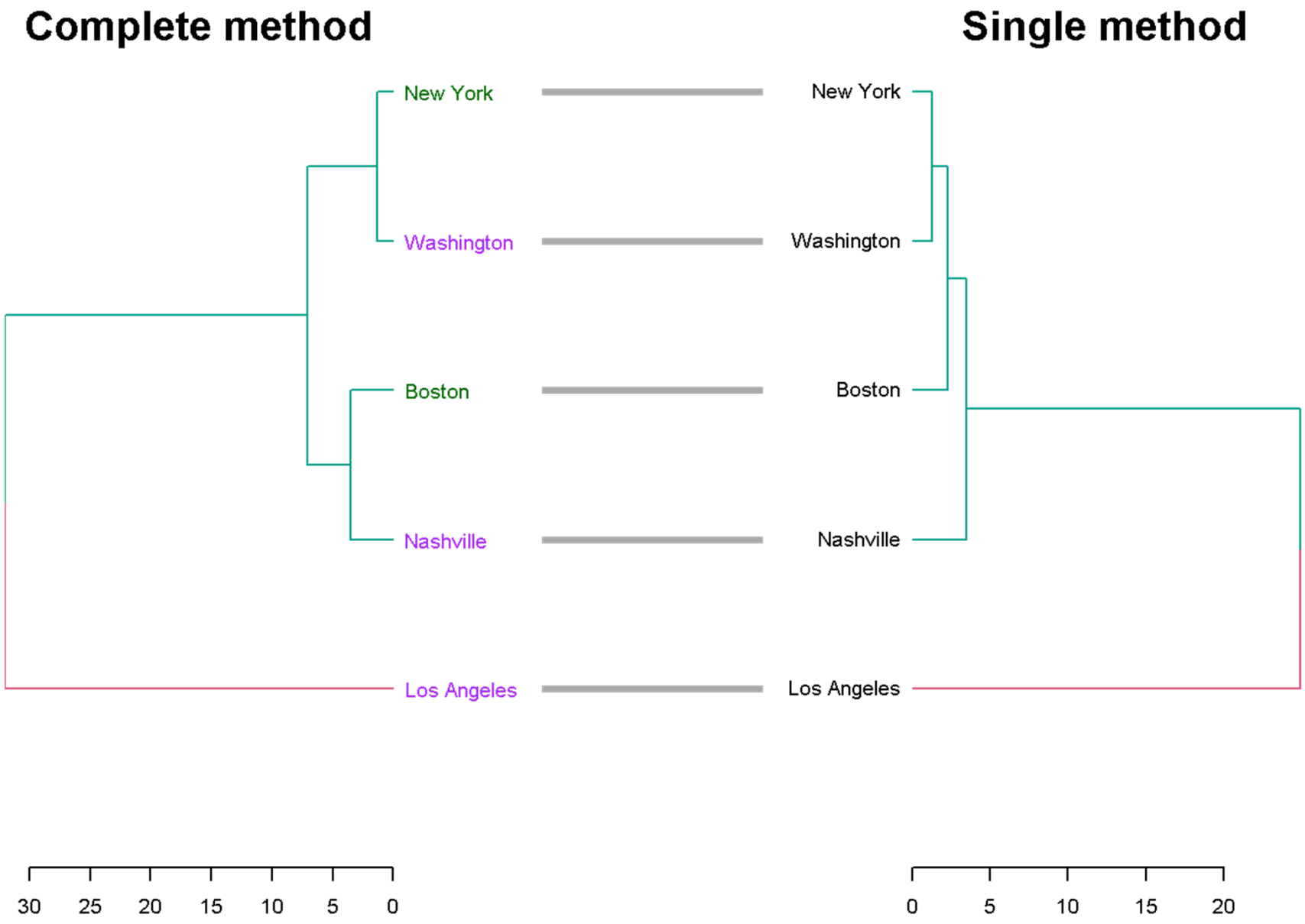
```
## We ran untangle  1  times
```

```
tanglegram(dend12_untang[[1]],dend12_untang[[2]],
           margin_inner = 6, columns_width = c(5,2,5),
           main_left = "Complete method", main_right = "Single method"
)
```

Available functions :

- untangle_random_search
- untangle_step_rotate_1side
- untangle_step_rotate_2side

Goal 3: comparing trees



We can also calculate a statistic that will measure the level of “correlation” between the two trees (from -1 to 1)

Goal 3: comparing trees (statistically)

Complete method **Single method**


```
cor_cophenetic(dend1_mod_01,dend2_mod_01)
```

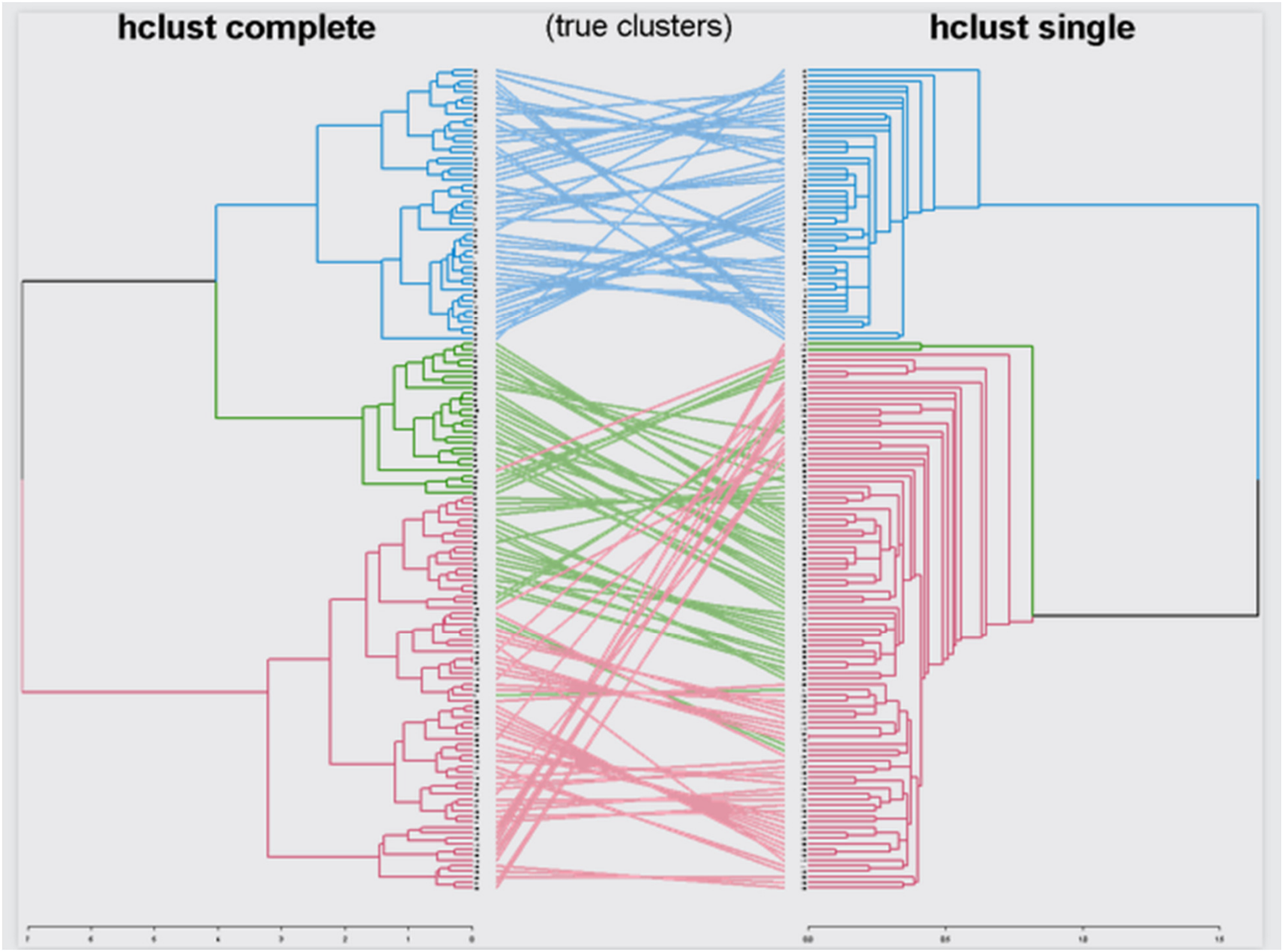
```
## [1] 0.9928
```

```
cor_bakers_gamma(dend1_mod_01,dend2_mod_01)
```

```
## [1] 0.8815
```

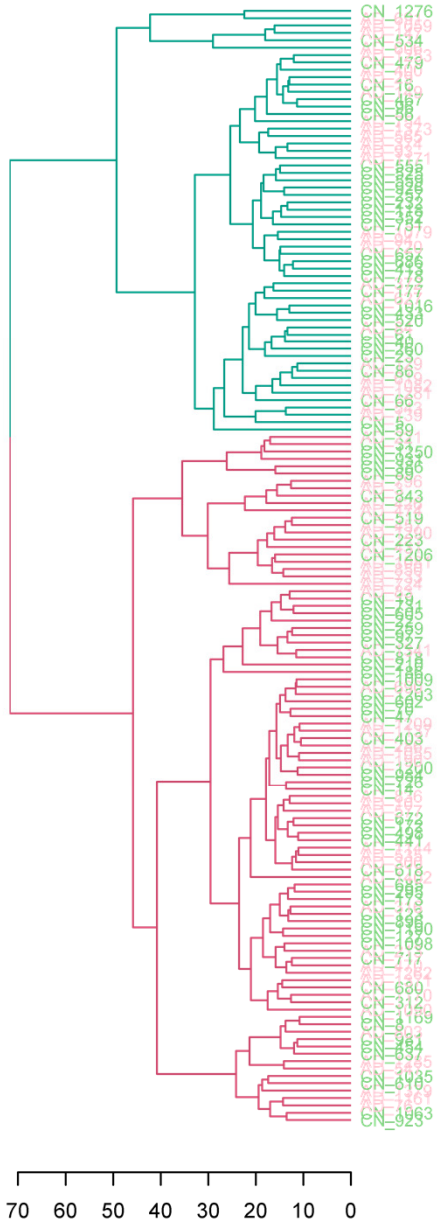
Some examples of comparing two trees

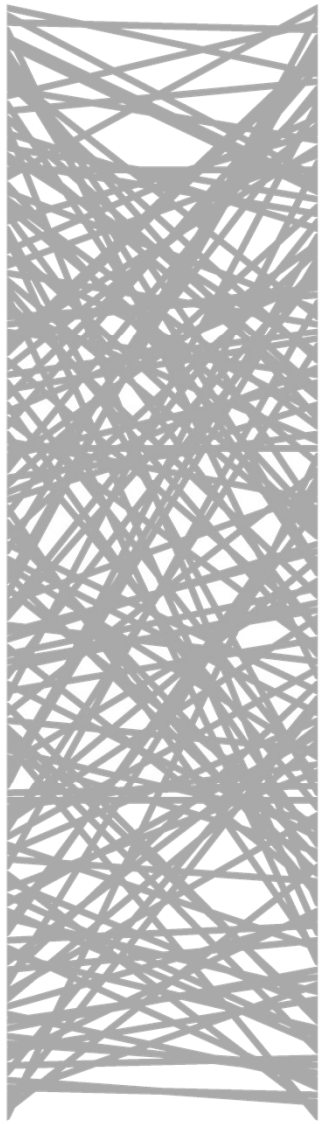
Visually comparing two clustering methods



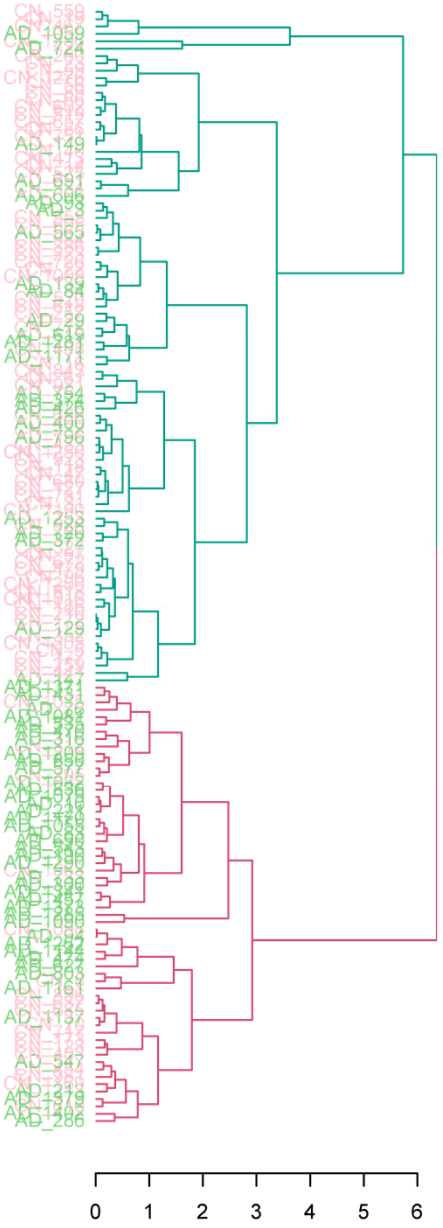
A plot from a recent HBP meeting in Lausanne

Dendrogram Using all variables



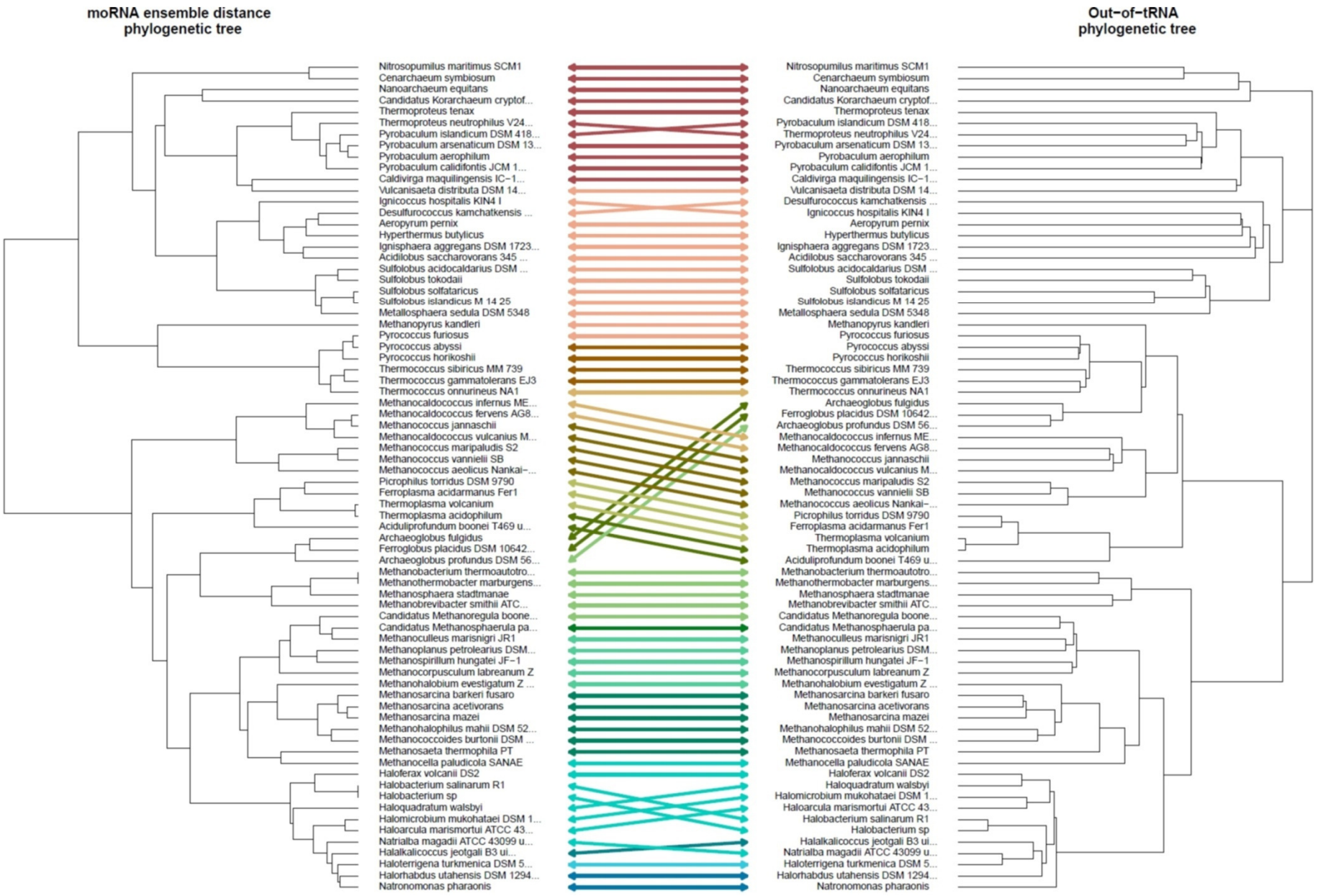


Tanglegram of the two models



Dendrogram Using 2 variables

Visually comparing two phylogenic trees



Lessons:

- hclust - Good for creating hierarchical clustering, but limited for plotting
- dendrogram object
  - a nested list of lists
  - with attributes!
  - should be modified step by step before plotting
- Dendrograms can be compared

By using C++, we can gain
a lot of speed gains on
in some bottleneck
functions
Here is how…

Goal 4: speed!

1. Use ***dendextendRcpp***

   i.e: install.packages("dendextendRcpp")

Examples for how much faster we can get…

Goal 4: speed!

labels

```
check_speed <- function(x) {
    require(dendextendRcpp)
    require(microbenchmark)
    hc <- hclust(dist(x))
    dend <- as.dendrogram(hc)
    print(microbenchmark(
        stats:::labels.dendrogram(dend),
        dendextendRcpp:::labels.dendrogram(dend),
        labels(dend),
        labels(hc),
                    times = 10),
        unit = "s")
}
```

Goal 4: speed!

labels

```
x <- 1:1000
names(x) <- x
```

```
check_speed(x[1:10])
```

```
Unit: seconds
                                                 expr         median
                stats:::labels.dendrogram(dend)     0.00182396
 dendextendRcpp:::labels.dendrogram(dend)     0.00014614
                                 labels(dend)     0.00010051
                                   labels(hc)     0.00006103
```

Goal 4: speed!

labels

```
x <- 1:1000
names(x) <- x
```

```
check_speed(x)
```

```
Unit: seconds
                                                 expr      median
                  stats:::labels.dendrogram(dend)   0.1897733
 dendextendRcpp:::labels.dendrogram(dend)   0.0044677
                                       labels(dend)   0.0045105
                                         labels(hc)   0.0001053
```

Goal 4: speed!

cutree

```
check_speed <- function(x) {
    require(microbenchmark)
    hc <- hclust(dist(x))
    dend <- as.dendrogram(hc)

    print(microbenchmark(
        cutree(hc, k = 3),
                   times = 10),
        unit = "s")

    assign_dendextend_options() # restore old functions

    print(microbenchmark(
        cutree(dend, k = 3, try_cutree_hclust = FALSE),
                   times = 10),
        unit = "s")
```

Goal 4: speed!

cutree

```
check_speed <- function(x) {
    require(microbenchmark)
    hc <- hclust(dist(x))
    dend <- as.dendrogram(hc)

    print(microbenchmark(
        cutree(hc, k = 3),
            require(dendextendRcpp)
            assign_dendextendRcpp_to_dendextend() # just to be on th

    ass
            print(microbenchmark(
                    cutree(dend, k = 3, try_cutree_hclust = FALSE),
    pri                                times = 10),
                    unit = "s")


}
```

Goal 4: speed!

cutree

```
x <- 1:1000
names(x) <- x
```

```
check_speed(x[1:10])
```

```
## Unit: seconds
##                                                                  expr
##  cutree(hc, k = 3)
##  cutree(dend, k = 3, try_cutree_hclust = FALSE)
##  cutree(dend, k = 3, try_cutree_hclust = FALSE)
```

median
0.0005303
0.01201
0.001501

Rcpp version

Goal 4: speed!

cutree

```
x <- 1:1000
names(x) <- x
```

```
check_speed(x[1:100])
```

```
## Unit: seconds
##                                                          expr
##  cutree(hc, k = 3)                                                  0.0006901
##  cutree(dend, k = 3, try_cutree_hclust = FALSE)                     0.1129
##  cutree(dend, k = 3, try_cutree_hclust = FALSE)                     0.003587
```

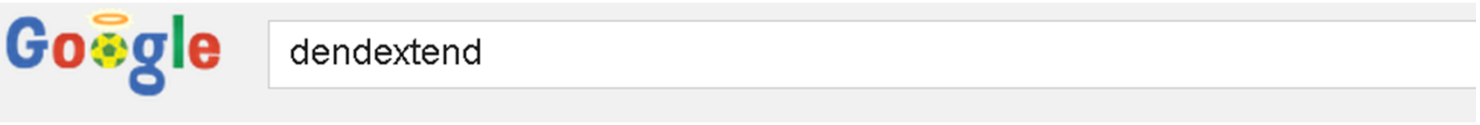
Rcpp version

Lessons:

- hclust - Good for creating hierarchical clustering, but limited for plotting
- dendrogram object
  - a nested list of lists
  - with attributes!
  - should be modified step by step before plotting
- Dendrograms can be compared
- Use dendextendRcpp for (“free”) speed

You can easily get dendextend:

```
install.packages("dendextend")
library(dendextend)
```



About 3,660 results (0.49 seconds)

CRAN - Package dendextend
cran.r-project.org/package=dendextend
Mar 15, 2014 - **dendextend**: Extending R's dendrogram functionality. Functions and methods for extending dendrogram objects in R.

[PDF] Package 'dendextend'
cran.r-project.org/web/packages/dendextend/dendextend.pdf
Package '**dendextend**'. March 15, 2014. Type Package. Title Extending R's dendrogram functionality. Version 0.14.2. Date 2014-03-15. Description Functions ...
tanglegram - match_order_by_labels - entanglement - flip_leaves

talgalili/dendextend · GitHub
https://github.com/talgalili/dendextend
R package. Contribute to **dendextend** development by creating an account on GitHub.

```
install.packages("dendextend")
library(dendextend)
```

Usage

Please see:

- Vignette: https://github.com/talgalili/dendextend/blob/master/vignettes/dendextend-tutorial.pdf
- Presentations:
  - http://htmlpreview.github.com/?https://raw.github.com/talgalili/dendextend/master/inst/doc/2013-09-05_Boston-useR/2013-09-05_Boston-useR_01_intro.html
  - http://htmlpreview.github.com/?https://raw.github.com/talgalili/dendextend/master/inst/doc/2013-09-05_Boston-useR/2013-09-05_Boston-useR_02_dendextend.html
  - http://htmlpreview.github.com/?https://raw.github.com/talgalili/dendextend/master/inst/doc/2013-09-05_Boston-useR/2013-09-05_Boston-useR_03_untangle_iris.html
  - http://htmlpreview.github.com/?https://raw.github.com/talgalili/dendextend/master/inst/doc/2013-09-05_Boston-useR/2013-09-05_Boston-useR_04_tree_inference.html

Credits!

`dendextend: Extending R's dendrogram functionality`

Functions and methods for extending dendrogram objects in R.

Author: Tal Galili [aut, cre, cph] (http://www.r-statistics.com), Gavin Simpson [ctb], jefferis [ctb] (imported code from his dendroextras package), Marco Gallotta [ctb] (a.k.a: marcog), plannapus [ctb], Gregory [ctb], R core team [ctb] (Other than the Infastructure, some code came from their examples), Kurt Hornik [ctb] Uwe Ligges [ctb], Yoav Benjamini [ths]

`dendextendRcpp: Faster dendrogram manipulation using Rcpp`

This package offers faster manipulation of dendrogram objects in R.

Author: Tal Galili [aut, cre, cph] (http://www.r-statistics.com), Romain Francois [ctb], Dirk Eddelbuettel [ctb], Kevin Ushey [ctb], Yoav Benjamini [ths]

useR!2014

dendextend: an R package for easier manipulation and visualization of dendrograms

Thank you!

Tal.Galili@gmail.com

R-statistics.com

R-bloggers.com