Notes for the CABANA Workshop: "Genomic analysis of crop diversity using R" – June 25 to 28, 2019 – Langebio Cinvestav Irapuato, México.

OCTAVIO MARTÍNEZ DE LA VEGA
COMPUTATIONAL BIOLOGY LAB., LANGEBIO, CINVESTAV IRAPUATO, MÉXICO
E-mail: octavio.martinez@cinvestav.mx.

Contents

Acknowledgments

This workshop is sponsored by CABANA; see Genomic Analysis of Crop Biodiversity Using R. Within CABANA we are grateful to Dr. Piv Gopalasingam (Scientific Training Officer, EMBL-EBI) for coordination of the workshop. We are also very grateful with MC José Manuel Villalobos-Escobedo (RNA computational genomics lab, PI Dr. Cei Abreu-Goodger) for his help in the teaching of the R part of this workshop. We also acknowledge Dr. June Simpson and Dr. Neftalí Ochoa-Alejo for allowing the use of the maize and chili data, respectively.

Date: June 12, 2019.

General notes

> *The statistician cannot excuse himself from the duty of getting his head clear on the principles of scientific inference, but equally no other thinking man can avoid a like obligation.*
>
> **Sir Ronald Fisher.**

See more R. Fisher quotations here.

If you find this document on the web and want to get the necessary functions and datasets for the workshop, please send me an e-mail to octavio.martinez@cinvestav.mx with subject: **Analysis of Diversity using R**.

In this workshop we will give a hands-on introduction to the analysis of genomic diversity with emphasis in crop species. We will use R to perform calculations, which will be very difficult to perform without a computer, but we will privilege the **understanding** of the processes over the ability to just carry on such data analyses. In other words, we must understand exactly what we are doing with R, to appreciate advantages as well as limitations of both, the data at hand as well as the numerical and statistical approaches employed. The interpretation of the the analyses and the balance of biological conclusions that can be reached depends heavily in the understanding of the methods employed –there in not an R package that could do that!

In this notes words in blue are links to web sites; for topic introduction we will use, among others, Wikipedia entrances, but keep in mind that the quality of such pages is variable. R text will be given within monospaced boxes, resembling what is input / output in the R terminal:

```
# This is an example of a box for R input/output
> seq(from=0, to=1, by=0.2)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

It must be obvious to you which parts are the input (always beginning with R prompt '>') and which parts are the output (as seen in the R terminal). Lines with comments begin always with '#'. I do not use Rstudio (can't teach an old dog new tricks!) but of course you are welcome to use it if you are familiar with it. You can copy the input lines from a box and paste it on your R terminal (of course omit the R prompt '>') but always try to understand what is going on. Please ASK any questions that you have; if you have a doubt, it is very likely that others will have the same query.

When faced with an R function that you do not known or not remember, please use R help facility, '?' followed by the function name. Also, remember that you have the '??' help operator to look for a general topic, e.g., '`??  Fisher`' will show some R objects that include the word 'Fisher' within their documentations.

We recommend that you set a directory to perform all R work of the project. By default we will assume that you created a directory named "`CabanaR`" (in fact, a compressed version of that directory will be available to you with all files needed). Also, it is very important is that you document all your work in log files (I use plain text files to do so). When you quit an R session with function "`q()`", save the content of the environment (that is the default behavior), thus in the following session you can proceed having the objects previously used.

1. Introduction

The term 'molecular marker' currently means a DNA-based marker; a polymorphism in the sequence of the genome of the specie studied that allows to distinguish individuals. At its finer level, a molecular marker is a difference in a single nucleotide at a given position on the genome, which is known as 'Single Nucleotide Polymorphism' or SNP (Gupta et al., 2001; Mah and Chia, 2007). It is important to note that, except for insertions or deletions (indels) (Väli et al., 2008) and whole chromosome or ploidy variations

(Suda et al., 2006), all differences between two or more individual genomes can be seen as collections of SNPs. Here we will briefly discuss different molecular marker methods to detect and measure genome variation within a single specie.

The use of molecular markers to study diversity has recently experienced a paradigm shift, going from '*a few markers in many individuals*' to '*many markers in a few individuals*'. When we have a few markers in many individuals (as was the case for example with isoenzymes), we could use methods of population genetics, i.e., calculate frequencies of alleles for subsets of individuals (populations) and study the diversity of allele frequencies between and within populations. With the advent of high-throughput or 'Next Generation DNA Sequencing' (NGS) methods, we now have 'many' –some time millions, of markers in a relatively modest number of sampled genomes. Thus, the methods to study genome diversity change from calculating frequencies of alleles to study the many genomic changes (polymorphisms) in a 'few' genomes. The ideal will to have '**many markers in many individual genomes**', and it will be reviewed in this workshop by the part given by Dr. M. Humberto Reyes-Valdés with the analysis of wheat data.

1.1. **Genomic Molecular Markers.** This section was adapted from (Martínez, 2018). We can divide molecular marker techniques first in two main groups, say, systems that need information from a reference genome and those that do not need such resource. In general, protocols that do not need a reference genome depend on restriction enzymes and polymerase chain reaction (PCR), and can give information about non-located anonymous genome places, or alternatively about a particular locus or set of loci.

Table 1 shows sets of molecular marker protocols classified by their type and suitability for detection and measurement of variation at the genomic level.

Given that we are concerned here with differences at genomic level, we are not very interested in methods that study a small number of very specific loci (see boxes **D** and **E** in Table 1). However, techniques that measure variation at a small set of loci that are randomly scattered in the genome by measuring the number of tandem repeated sequences flanked by conserved regions (mini and microsatellites, SSR, STMS, VNTR, ISSR, RAMP; see Table 1) can be useful mainly when genome position of markers are known. Such methods are very powerful to measure the diversity of populations, for example in maize (Hayano-Kanashiro et al., 2017), a case that here we will review in detail (section 'Maize data'); however, the fact that they target a single locus at the time, needing the knowledge of the conserved flanking sequences and thus particular PCR primers for each locus, made them relatively inefficient for detection of mutations distributed in a completely random fashion along the whole genome. Other method that targets specific genes is the TRAP technique (see Table 1).

Transposable elements (TE) (Deragon et al., 2008) are an important source of genome variations, and activation of quiescent transposable elements and retrotransposons has been reported to occur, for example, during tissue culture (Hirochika, 1997; Kaeppler et al., 2000). There are various marker systems associated with TE, as IRAP, MITES, MSTD, RBIP and REMAP (see box **D** in Table 1). In general, TE cause genome modifications that can be considered as insertions or deletions, when a transposition event creates a new copy of the transposon, while the original copy remains intact at the donor site, or when the TE changes its location in the genome (Agarwal et al., 2008). These changes can also be detected by techniques different to the TE related markers mentioned above, and thus, unless there is a strong reason to focus on the search for TE genome variation, it is more efficient to use general protocols that will detect, apart of TE related variations, other mutations. Also, it had been reported that TE mutations were not found in sampled regenerants of Arabidopsis, where high levels of SNPs were found (Jiang et al., 2011), thus the relative importance of variation produced by TEs appears to be small when compared with SNPs.

Having discarded in principle molecular marker protocols that detect variations in particular loci or depend on TEs (boxes **C** and **D** in Table 1, respectively), we need to decide between methods that need a reference genome (box **B** in Table 1) or those that do not need such resource (box **A** in Table 1).

TABLE 1. Molecular marker methods classified by suitability for detection of genomic variation. Adapted from (Martínez, 2018).

A) Genome Independent (GI) methods.		
Acronym	Description	Reference
ISSR	Inter-Simple Sequence Repeat (inter-microsatellite regions). Useful also when genome is available.	Nybom (2004)
AFLP	Amplified fragment length polymorphism	Vos et al. (1995)
AP-PCR	Arbitrarily primed-PCR	Williams et al. (1990)
DAF	DNA amplification fingerprinting	Williams et al. (1990)
DArT	Diversity Arrays Technology	Wenzl et al. (2004)
RAD-seq	Restriction-site-associated DNA sequencing	Miller et al. (2007)
SRAP	Sequence-related amplified polymorphism	Li and Quiros (2001)
B) Genome dependent (GD) methods.		
Acronym	Description	Reference
GBS	Genotyping-by-sequencing (produce SNPs)	He et al. (2014)
HTG	High-throughput genotyping by whole-genome resequencing	Huang et al. (2009)
RAD	Restriction-site Associated DNA tags	Baird et al. (2008)
SNP	Single nucleotide polymorphism	Mah and Chia (2007)
SNP array	Large SNP arrays for plant genotyping	Ganal et al. (2012)
TILLING	Targeting Induced Local Lesions in Genomes	McCallum et al. (2000)
C) Not very useful for genomic level detection of variation; better alternatives exist.		
Acronym	Description	Reference
RAMP	Randomly amplified microsatellite polymorphisms	Wu et al. (1994)
RAPD	Random amplified polymorphic DNA	Williams et al. (1990)
SCAR	Sequence characterized amplified region (derived from RAPD)	Paran and Michelmore (1993)
SSCP	Single strand conformation polymorphism	Orita et al. (1989)
SSR	Simple sequence repeats or "microsatellites"	Beckmann and Soller (1990)
STMS	Sequence Tagged Microsatellite Sites or "microsatellites"	Beckmann and Soller (1990)
RFLP	Restriction fragment length polymorphism	Botstein et al. (1980)
TRAP	Target region amplification polymorphism	Wang et al. (2003)
VNTR	Variable number of tandem repeats	Nakamura et al. (1987)
D) Transposable elements (TE) related; not optimal to general genome variation.		
Acronym	Description	Reference
IMP	Inter-MITE polymorphism	Chang et al. (2001)
IRAP	Inter-retrotransposon amplified polymorphism	Kalendar et al. (1999)
MITES	Miniature inverted repeat transposable elements	Casa et al. (2000)
MSTD	Methyl-sensitive transposon display	Azman et al. (2014)
RBIP	Retrotransposon-based insertion polymorphism	Flavell et al. (1998)
REMAP	REtransposon-microsatellite amplified polymorphism	Kalendar et al. (1999)
S-SAP	Sequence-specific amplification polymorphism	Waugh et al. (1997)
TD	Transposon display	Va et al. (1998)
E) Other methods with specific proposes.		
Acronym	Description	Reference
CAPS	Cleaved amplified polymorphic sequence (SNPs to PCR markers)	Komori and Nitta (2005)
MSAP	Methylation-Sensitive Amplification Polymorphism	Peredo et al. (2009)
RAP-PCR	RNA fingerprinting by arbitrarily primed PCR	McClelland and Welsh (1994)
RNA-Seq-SNPs	SNPs obtained from RNA-Seq	Novaes et al. (2008)

As a consequence of next generation DNA sequencing technologies (NGS), we have the possibility to re-sequence entire genomes or sample entire transcriptomes more efficiently and economically and in greater depth than ever before (Varshney et al., 2009). In particular, for detection and measurement of variation in species with a reference genome, it is now possible to employ methods of 'Genotyping By Sequencing' or 'GBS' (He et al., 2014) which have multiple protocol variations to reduce genome complexity; see for example (Elshire et al., 2011; Davey et al., 2011; Poland and Rife, 2012; Narum et al., 2013). Examples of other genome-dependent methods to obtain SNPs are HTG, RAD, SNP arrays and TILLING (see box **B** in Table 1).

When transcriptome data are accessible for the specie of interest, and even if a reference genome is not available, it is possible to develop and use SNPs derived from the transcripts (Barbazuk et al., 2007; Novaes et al., 2008; Haseneyer et al., 2011; Hiremath et al., 2011) to measure variation. Also, differences in gene expression using RNA-Seq expression are a valuable source to explore phenotypic variations that are the result of whole genome variations. We will review that in section 'Molecular phenotypes with RNA-Seq'.

Finally, when a reference genome is not available, we can employ one of the genome sampling methods which depend on restriction enzymes, PCR reactions and measurement of the molecular size of DNA fragments, as AFLP, AP-PCR, DAF, DArT, RAD-seq or SRAP (see box **A** in Table 1 for a short description and references). The selection of one of these particular protocols depends on considerations of equipment and reactants availability as well as cost / benefit.

It is important to keep in mind that both, GD and GI, protocols are methods to *sample* the genomes, and as such the results will detect genetic variants (polymorphisms) only in a proportion, say **P**, of the genomes in the individuals studied. **P** determines the precision (statistical variability) and accuracy (statistical bias) of the estimates. If the sampling is fully random, i.e., if each base has the same probability to be included in the samples, the method will be unbiased or completely accurate, and under such condition small values of **P** will produce precise estimates with narrow confidence intervals. Unfortunately in general it is difficult to guarantee that the sampling method will be 'fully random', because for sampling in both, GD and GI protocols, we reduce the full genome to a sample by selecting particular DNA motifs –in general enzyme restriction sites. Thus the randomness of the sampling depends on the distribution of such sites in the genome; for GI there is no way to estimate such distribution *a priory*, while for GD methods such distribution can be analyzed, to assure that it is relatively uniform along all the reference genome. In all cases methods that give polymorphisms clustered at particular loci or genome segments –as the ones presented in boxes **C** and **D** of Table 1, must be avoided to ensure the unbiased and precise estimation of variation.

Other factor to take into account for method selection is if the detected variation will be employed in future plant breeding programs (Karp, 1995), or if its evaluation is intended simply to measure the genetic homogeneity (Sahijram et al., 2003). Because GD will always give polymorphisms of known physical genome location, they are advantageous in the former case (polymorphism will be used in breeding), while if GI methods are employed, a set of crosses will be needed to obtain their approximate location in a recombination map. The high costs in resources and time to perform crosses for mapping polymorphisms will almost surly exceed the differential cost of GI (cheap) compared with GD (relatively expensive); thus, if a reference genome is available and the markers will be used in breeding, the wiser decision appears to be to select a GD protocol. For breeding proposes there are relatively simple ways to transform SNPs into markers that can be easily evaluated in populations (Hayashi et al., 2004; Komori and Nitta, 2005; Shahinnia and Sayed-Tabatabaei, 2009), for example for marker assisted selection (Gupta et al., 2001).

1.1.1. *Excercise: Your dreamed project.* In a short paragraph (less than about 200 words) present the summary of a project related with genomic diversity. Include title, a phrase of introduction, aims (goals), methods and expected results. Assume that you do not have financial or time limitations.

1.2. **R packages and web resources for biodiversity.** There are many R packages and web resources related with the study of biodiversity. Table 2 presents only some of them. Unfortunately, there are still relatively few resources related to genomic diversity for plant species. Naturally, there are many resources to study human genetic variation, but these are not of direct interest to us. However, the panorama is rapidly changing with specialized databases for diversity of important crop species, as for example PANZEA for maize, the SolGenomics for solanaceae species, RiceVarMap for rice, etc.

TABLE 2. R packages and web resources related with Biodiversity.

Name	Description	Reference
BiodiversityR	Graphical User Interface (via the R-Commander) and utility functions (often based on the vegan package) for statistical analysis of biodiversity and ecological communities, including species accumulation curves, diversity indices, Renyi profiles, GLMs for analysis of species abundance and presence-absence, distance matrices, Mantel tests, and cluster, constrained and unconstrained ordination analysis.	Kindt and Coe (2005)
BioFTF	An R Package for Biodiversity Assessment with the Functional Data Analysis Approach	Di Battista et al. (2017)
BIO-R	Biodiversity analysis with R (BIO-R) is a set of R programs that do biodiversity analysis of molecular data, in order to calculate heterozygosity, diversity among and within groups, shannon index, number of effective allele, percent of polymorphic loci, Rogers distance, Nei distance, cluster analysis and multidimensional scaling 2D plot and 3D plot	Reyes-Valdés et al. (2018)
Darwin Core	An Evolving Community-Developed Biodiversity Data Standard. It includes a glossary of terms intended to facilitate the sharing of information about biological diversity by providing identifiers, labels, and definitions.	Wieczorek et al. (2012)
bdvis	bdvis: visualizing biodiversity data in R	Barve and Otegui (2016)
GBIF	Global Biodiversity Information Facility. Integrated publishing toolkit: facilitating the efficient publishing of biodiversity data on the internet (lots of data!)	Robertson et al. (2014)
mobsim	An R package for the simulation and measurement of biodiversity across spatial scales	May et al. (2018)
BAT	Biodiversity Assessment Tools, an R package for the measurement and estimation of alpha and beta taxon, phylogenetic and functional diversity	Cardoso et al. (2015)
EstimateS	Statistical estimation of species richness.	Chazdon et al. (1998)
BIEN	Provides Tools for Accessing the Botanical Information and Ecology Network Database. The BIEN database contains cleaned and standardized botanical data including occurrence, trait, plot and taxonomic data	Maitner et al. (2018)
BIEN	Botanical Information and Ecology Network	Maitner et al. (2018)
Rphylip	Rphylip provides an R interface for the PHYLIP package. PHYLIP is a free package of programs for inferring phylogenies (Felsenstein, 1993; Retief, 2000). See PHYLIP for more information about installing PHYLIP	Chamberlain (2013)
Pvclust	An R package for assessing the uncertainty in hierarchical clustering	Suzuki and Shimodaira (2006)

1.3. **Overview of R.** R is the more used platform by statisticians and researchers for arithmetical calculations and statistical analyses. And there are multiple reasons for that, I will say that the most

important is the solid and comprehensive set of functions present in the minimal installation and also the plethora of 'packages' to perform specialized analysis. Possibly the less attractive aspect of R for people that begin to use it is that R is 'command line' oriented, i.e., you must type a command instead of simply click in an icon. This produces a very steep learning curve; you know what manipulation do you need to carry on, but do not know the 'magic words' to perform them (i.e., operations or functions names). Even when this is not an introductory R course –I assume that you have seen the recomended page r-introduction, we encourage you to ASK about any doubt that you could have. An introduction or remainder of basic R stuff that I particularly like is here NetSci X Tutorial. One great advantage is that the R community is huge, and always eager to help novices; if you have question (out of this classroom) you can always 'Google' your question, and in most cases you will find the answer.

Let's do our quick introduction (remainder) of R with the content of `Box 1`. I assume that you have created a directory, say "`CabanaR`", where you are going to be working. I also recommend that you open a text file to document your calculations...

```
1  ------------------------------------> Box  1 <---------------------------------------
2  # Open R by making click in the corresponding icon...
3  # Some stuff appears in the terminal, before it settle down:
4  R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
5  Copyright (C) 2018 The R Foundation for Statistical Computing
6  Platform: x86_64-apple-darwin15.6.0 (64-bit)
7
8  R is free software and comes with ABSOLUTELY NO WARRANTY.
9  You are welcome to redistribute it under certain conditions.
10 Type 'license()' or 'licence()' for distribution details.
11
12   Natural language support but running in an English locale
13 # ... (more stuff)
14
15 # Now, I am going to navigate to find the place where I have the CabanaR directory
16 # (The results that you will obtain will be different, depending on your instalation)
17 # that is why I remarked the following lines
18
19 # > dir()
20 # [1] "Applications" "BLASTDB"      "Desktop"      "Documents"    "Downloads"
21 # ... (more files)
22
23 # > dir("Documents/Cursos/2019/6_JuneCABANA/CABANA/CabanaR/")
24 # [1] "DummyExample.RData" # ... more files are displayed
25
26 # I am going to use
27 # > setwd("Documents/Cursos/2019/6_JuneCABANA/CABANA/CabanaR/")
28 # Thus now I will be working in THAT directory
29 # If you cannot do that, please ask one of the instructors.
30
31 # OK Now, let's begin.
32 > help.start() # Must open an HTML document in your web navigator...
33 starting httpd help server ... done
34 If the browser launched by '/usr/bin/open' is already running, it is *not*
35     restarted, and you must switch to its window.
36 Otherwise, be patient ...
37
38 > demo() # Which "demos" are there?
```

```
39
40 # Let's see a sample of R plots:
41 > demo("graphics")
42
43 Type  <Return>         to start :
44 ### Various nice plots appear...
45
46 ## R as numerical calculator
47 > 2^5; log2(64); sin(0); pi # Note ";" can be used as carriage return
48 [1] 32
49 [1] 6
50 [1] 0
51 [1] 3.141593
52
53 ## R has memory (keep objects) and also distinguish between small and capital letters
54 > x # Is there an object call "x" in the "environment"?
55 Error: object 'x' not found
56 > x # Is there an object call "x" in the "environment"?
57 Error: object 'x' not found
58 > x <- 55 # Note the symbols "<-" are the assignment operator
59 > x # Now we have the value of 5 "stored" in an object called "x"
60 [1] 55
61
62 > ? class # Ask about the function "class"
63 > class(x) # which kind of object is x?
64 [1] "numeric"
65 > X <- 20
66 > x == X # R knows about logical comparisons
67 [1] FALSE
68 > x <= X # Is x less or equal than X
69 [1] FALSE
70 > x > X # Is x larger than X?
71 [1] TRUE
72
73 # R can also deal with vectors (one dimensional ordered arrays)
74 > x <- c(1,2,5,7,15,20) # A vector with 6 numbers
75 > x # Note that we ERASED the previous value of "x"; let's see the new
76 [1]  1  2  5  7 15 20
77 # Of course operations can be performed with vectors...
78 > y <- 2*x
79 > y
80 [1]  2  4 10 14 30 40
81 > y^x # Will give some large numbers
82 [1] 2.000000e+00 1.600000e+01 1.000000e+05 1.054135e+08 1.434891e+22 1.099512e+32
83
84 ## Assume that you want to make a Tukey test...
85 > Tukey # Just try your look...
86 Error: object 'Tukey' not found
87 > ?? Tukey # Asking to see if the word "Tukey" appears somewhere
88
89 # A new window is open, and eventually you see that the Tukey test is named as "TukeyHSD"...
90 # (there are also other Tukey's stuff!)
```

```
91
92  # R knows also about matrices...
93  > ? matrix
94  > ? matrix
95  > z <- matrix(c(1:4), nrow=2, ncol=2) # A 2 x 2 matrix...
96  > z
97       [,1] [,2]
98  [1,]    1    3
99  [2,]    2    4
100
101 # What the heck means "c(1:4)"?
102 > ? c
103 > c(1:4)
104 [1] 1 2 3 4
105 > c(1.1:2.2)
106 [1] 1.1 2.1
107 > c(1.1:5.2)
108 [1] 1.1 2.1 3.1 4.1 5.1
109
110 # A more powerful way to create numeric sequences
111 > ? seq
112 > seq(from=0, to=10, by=2)
113
114 [1]  0  2  4  6  8 10
115 > seq(0, 10, 2) # You do not need to write the parameters names (respect the order)
116 [1]  0  2  4  6  8 10
117 > seq(0,1,.05)
118  [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80
119 [18] 0.85 0.90 0.95 1.00
120
121 # There is also a function to "repeat" (replicate) elements
122 > ? rep
123 > rep(0, 5)
124 [1] 0 0 0 0 0
125 > rep(c(1:2), 3)
126 [1] 1 2 1 2 1 2
127 > rep(c(1:2), each=3)
128 [1] 1 1 1 2 2 2
129
130 # Characters can also been used
131 > rep(c("a", "b", "c"), each=2)
132 [1] "a" "a" "b" "b" "c" "c"
133 > rep(c("a", "b", "c"), 2)
134 [1] "a" "b" "c" "a" "b" "c"
135
136 > letters # A predefined set
137  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u"
138 [22] "v" "w" "x" "y" "z"
139 > LETTERS # Another
140  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U"
141 [22] "V" "W" "X" "Y" "Z"
142
```

```
# There are ways to access the elements within a structure; the "[ ]" operator
> temp <- seq(0, 10, 2) # Or seq(from=0, to=10, by=2)
> temp
[1]  0  2  4  6  8 10
 length(temp) # The length (number of elements) of temp
[1] 6
> temp[3] # The third element
[1] 4
> temp[3:5] # Elements 3 to 5
[1] 4 6 8
> temp[7] # An element that does NOT exist
[1] NA

# The special value "NA" means "Not Available" or "missing" (important)

# Let's see the operator "[ ]" for matrices
> temp2 <- matrix(c(1:9), nrow=3, ncol=3, byrow=TRUE)
> temp2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> temp2[2,3] # The value in the second row, third column
[1] 6
> temp2[,3] # Values in the third column
[1] 3 6 9
> temp2[1,] # Values in the first row
[1] 1 2 3

# R has many pre-loaded data sets; let's see
> ? data # Loads specified data sets, or list the available data sets.
> data() # List the ones available
> data(CO2) # Loads the "CO2" dataset
> ? CO2
> CO2 # See the whole dataset (84 rows; not shown)
> head(CO2, n=5) # Shows the first 5 rows
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
> class(CO2) # Which class(es)?
[1] "nfnGroupedData" "nfGroupedData"  "groupedData"    "data.frame"
> nrow(CO2) # Number of rows
[1] 84
> ncol(CO2) # Number of columns
[1] 5
> summary(CO2) # Let's see
     Plant             Type         Treatment       conc          uptake
 Qn1    : 7   Quebec     :42   nonchilled:42   Min.   :  95   Min.   : 7.70
 Qn2    : 7   Mississippi:42   chilled   :42   1st Qu.: 175   1st Qu.:17.90
```

```
 Qn3    : 7                                     Median : 350   Median :28.30
 Qc1    : 7                                     Mean   : 435   Mean   :27.21
 Qc3    : 7                                     3rd Qu.: 675   3rd Qu.:37.12
 Qc2    : 7                                     Max.   :1000   Max.   :45.50
 (Other):42

> CO2$Plant # All values of column Plant in the dataframe (not shown)
> CO2$Plant[1:5] # Values 1 to 5 of column Plant
[1] Qn1 Qn1 Qn1 Qn1 Qn1
Levels: Qn1 < Qn2 < Qn3 < Qc1 < Qc3 < Qc2 < Mn3 < Mn2 < Mn1 < Mc2 < Mc3 < Mc1
> class(CO2$Plant) # What kind of variable
[1] "ordered" "factor"
> class(CO2$Type) # What kind of variable
[1] "factor"
> CO2$Type[1:5]
[1] Quebec Quebec Quebec Quebec Quebec
Levels: Quebec Mississippi
> CO2[2:4,3:5] # Elements in rows 2 to 4 and columns 3 to 5
   Treatment conc uptake
2 nonchilled  175   30.4
3 nonchilled  250   34.8
4 nonchilled  350   37.2

# Data Frames are a class of objects that can have columns with numbers or factors.
# Let's create a data.frame
> ? data.frame
> temp.df <- data.frame(Treatment=rep(c("A", "B"), each=10),
   Result=c(rnorm(10, 10), rnorm(10, 11)))
# What I did? Let's find out:
> ? rnorm # rnorm(n, mean = 0, sd = 1)
> summary(temp.df) # Note that your results for "Result" will be different!
 Treatment     Result
 A:10      Min.   : 8.857
 B:10      1st Qu.: 9.950
           Median :10.606
           Mean   :10.631
           3rd Qu.:11.319
           Max.   :12.570
> tapply(temp.df$Result, temp.df$Treatment, summary)
$A
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  8.857   9.532   9.917  10.000  10.344  11.385

$B
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.14   10.75   11.27   11.26   11.70   12.57

> summary(aov(Result ~ Treatment, data=temp.df)) # Summary of ANOVA
            Df Sum Sq Mean Sq F value  Pr(>F)
Treatment    1  7.967   7.967   15.17 0.00106 **
Residuals   18  9.454   0.525
---
```

```
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1
> boxplot(Result ~ Treatment, data=temp.df) # Result not shown

# A powerful kind of structure within R are lists; let's see:
> temp.l <- list("This list contains my temp objects", temp, temp2,
   temp.df, summary(aov(Result ~ Treatment, data=temp.df)))
> temp.l # Show all elements of the list (not shown here)
> temp.l[[1]] # The first element of the list
[1] "This list contains my temp objects"
> class(temp.l[[1]])
[1] "character"
> temp.l[[3]] # The third element of the list
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> temp.l[[3]][,2] # The second column of the third element of the list
[1] 2 5 8

# in R you can "program"; i.e., create functions
> temp.f <- function(x=5){paste("my input was =", x)} # A silly function
> class(temp.f)
[1] "function"
> temp.f # Shows the function
function(x=5){paste("my input was =", x)}
> temp.f() # Execute the function with the default
[1] "my input was = 5"
> temp.f(3.141592654) # Execute the function with other value
[1] "my input was = 3.141592654"
> temp.f("Hello World!") # Execute the function with other value
[1] "my input was = Hello World!"

# LOOPS
# As any programing language R can perform loops; see for example:
> for(i in 1:5){print(i)}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
> for(i in 1:5){cat(i, "! = ", factorial(i), "\n", sep="")}
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120

# Be creative and EXPLORE R possibilities!!!
> q() # TO quit the R session; you can save (or not) these results.
--------------------------------------------------------------------------------
```

1.3.1. *Excercises.*

(1) Investigate if R has functions to deal with sets (as 'union' for example), look to those functions (by typing their names and giving 'return') How much can you understand of such functions?.

(2) Using your web navigator find (if there exist) at least one r package to work with DNA sequences.

(3) Install the R package `pvclust`.

2. Measuring diversity

In this workshop we are concerned with the evaluation of genetic diversity at genomic level. We will not study here the problem of reconstructing a molecular phylogeny –which is a separate problem. For those interested in that topic, I strongly recommend the phylogeny inference package PHYLIP (Felsenstein, 1993), for which there is an R interface (see Table 2).

2.1. **A simple example.** We will begin by studying nucleotide diversity using a very simple ('dummy') example, which is presented in `Box 2`.

```
1  ------------------------------------> Box  2  <---------------------------------------
2  # Run R and go to your 'CabanaR' directory
3  # Note that IN YOUR computer the next line may point to a different place!
4  # > setwd("Documents/Cursos/2019/6_JuneCABANA/CABANA/CabanaR/")
5  > load("DummyExample.RData") # Load some stuff
6
7  # You can list the objects currently in your environment with
8  > ls()
9  [1] "dummy"       "dummy.stuff" "info.sites"  "mis.seq"     "mutateDNA"   "my.dummy"
10 [7] "randomDNA"   "SNP.dist"
11 # (note you could have MORE objects; but those here MUST appear (where loaded))
12
13 # The object that we will be abalysing:
14 > my.dummy
15 [1] "AACCCATTCGCAGAGTAACTAGTGTGACGTCCGCAGGGCATCCAAAGCCA"
16 [2] "AAGCCATTCGCAGAATAACTAGTGTGACGTTCGCAGGGCATCAAAAGCCA"
17 [3] "AACACATTGGCAGAAGAACTAGTGCGACGTTCGCAGAGCAACCAAATCCG"
18 [4] "CACACTTTCGCAGAAGAACTAGTGCGACGTTCGCAGGGCAACCAAATCCG"
19 > class(my.dummy)
20 [1] "character"
21 > length(my.dummy)
22 [1] 4
23 > my.dummy[1]
24 [1] "AACCCATTCGCAGAGTAACTAGTGTGACGTCCGCAGGGCATCCAAAGCCA"
25 > my.dummy[1] == my.dummy[2] # Are those IDENTICAL?
26 [1] FALSE
27 # Confirm that all possible pairs of sequences are DIFFERENT.
28 ------------------------------------------------------------------------------------
```

The object `my.dummy` contains the representation of 4 small DNA sequences. Such sequences are assumed to be ridiculously small 'genomes'. They are small because we want to be able to note their differences and similitudes 'by eye'. Note also that all 4 DNA's are of the same size, they are aligned and, furthermore they are 'alike' –many of the bases in the same position are the same. Take a moment to observe them and try to make a mental 'summary'. Clearly, to analyze those sequences we must convert them to vectors of letters (bases), and then compare them base to base. We do that in `Box 3`.

```
----------------------------------> Box  3  <--------------------------------------
> ? strsplit # A function to segregate characters

# Let's split each one of the 4 sequences in my.dummy into characters:
> temp <- strsplit(x=my.dummy, split="") # Let's try
> class(temp) # Which class of object is "temp"
[1] "list"
> temp # Let's see it in full [ OUTPUT not shown here! ]

> length(temp) # Number of sequences in the list
[1] 4
> length(temp[[1]]) # Number of "bases" in the first sequence
[1] 50

# Now, having each one of the sequences as vectors in temp[[1]], temp[[2]], ..., temp[[4]]
# we could compare them "base to base", for example:
> temp[[1]] == temp[[2]] # Compare sequences 1 and 2 "base to base"
 [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[15] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[29]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[43] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

# Note how R convert logical values TRUE to 1 and FALSE to 0 when needed:
> 1 == 1 # That is a question
[1] TRUE
> 1*(1 == 1) # That is a numerical value
[1] 1
> "A" == "A" # That is a question
[1] TRUE
> 1*("A" == "A") # That is a numerical value
[1] 1
> "A" == "T" # That is a question
[1] FALSE
> 1*("A" == "T") # That is a numerical value
[1] 0

# We can use that fact to detect how many bases are equal (or different) between two sequences
> sum(temp[[1]] == temp[[2]]) # Because we "sum" logical are converted to numbers
[1] 46
> sum(temp[[1]] != temp[[2]]) # "!=" is short for different
[1] 4
> 46+4 # The full length of the sequences
[1] 50
-------------------------------------------------------------------------------------
```

We want to evaluate how alike are the sequences that we have at hand. A reasonable measure could be given by the number of bases differences between a pair of sequences, probably divided by the total length of the sequences. First we can notice that not all sequence positions are 'informative', because some of the bases are identical in all 4 sequences (see `Box 2`). Note that any measure of likeness –or distance, between sequences will depend ONLY on informative sites. Let's see a function that do that and run it for our sequences in `Box 4`

```
----------------------------------> Box  4  <--------------------------------------
> info.sites # My function to isolate informative sites:
function(x=my.dummy, out.as.matrix = TRUE){
        # info.sites
        # Takes as input a set of sequences of the same length
        # and detects the sites that are different between one pair
        # or more of the sequences.

        # Output a matrix with the results
        # (if out.as.matrix = TRUE)
        # ELSE return pseudo molecules and vector of original informative positions

        n.s <- length(x) # Number of sequences.

        # Splits the sequences into their components
        # (the result is a list, where each component is one of the sequences)
        x.l <- strsplit(x=x, split="", fixed=TRUE)
        n.b <- length(x.l[[1]])
        # Number of bases of first sequence (will ASSUME all sequences are of the same length!)
        x.data <-
        data.frame(matrix("", nrow=n.s, ncol=n.b, dimnames=list(c(1:n.s),
        paste("b", c(1:n.b), sep=""))), stringsAsFactors=FALSE)
        # Fils x.data
        for(i in 1:n.s){
                x.data[i,] <- x.l[[i]]
        }

        base.index <- c(1:n.b) # Index for the bases
        informative <- c() # an empty vector (will contain the bases that are informative!)

        # Obtain the number of the bases that are informative
        for(i in 1:n.b){
                if(length(unique(x.data[,i])) > 1) informative <- c(informative, i)
        }
        x.data <- x.data[,informative] # Only informative bases!
        if(out.as.matrix){
                # Return the matrix of informative sites
                return(x.data)
        } else {
                # will convert the matrix to sequences
                x.seq <- c()
                for(i in 1:n.s){
                        x.seq <- c(x.seq, paste(x.data[i,], collapse=""))

                }
                res <- list(x.seq, informative)
                names(res) <- c("pseudo.seq", "base.pos")
                return(res)

        }

}
```

```
<bytecode: 0x7f9f2cafd118>

> info.sites(x=my.dummy) # Run with our 4 sequences
  b1 b3 b4 b6 b9 b15 b16 b25 b31 b37 b41 b43 b47 b50
1  A  C  C  A  C   G   T   T   C   G   T   C   G   A
2  A  G  C  A  C   A   T   T   T   G   T   A   G   A
3  A  C  A  A  G   A   G   C   T   A   A   C   T   G
4  C  C  A  T  C   A   G   C   T   G   A   C   T   G
> my.dumy.info <- info.sites(x=my.dummy) # Keep the result in an object

> class(my.dumy.info)
[1] "data.frame"
> ncol(my.dumy.info) # Number of informative bases
[1] 14
> nrow(my.dumy.info) # Number of sequences
[1] 4
> info.sites(x=my.dummy, out.as.matrix=FALSE) # Alternative output
$pseudo.seq
[1] "ACCACGTTCGTCGA" "AGCACATTTGTAGA" "ACAAGAGCTAACTG" "CCATCAGCTGACTG"

$base.pos
 [1]  1  3  4  6  9 15 16 25 31 37 41 43 47 50
-------------------------------------------------------------------------------
```

Note that the function `info.sites` input a set of (aligned) sequences (of the same size) and output a `data.frame` which has as rows the sequences, as columns ONLY the informative positions, i.e., the cases for which at least one of the sequences has a different base. This function summarizes all differences among the sequences presenting the results as informative sites. Those sites represent SNP's between pair of sequences.

2.1.1. *Excercises.* In your R environment you have a 'mysterious sequence' which name is "`mis.seq`". You are going to work with such sequence here.

(1) Is "`mis.seq`" a random DNA? or does it corresponds to some 'true' sequence from an organism? To answer you may use the NCBI BLAST site. If you do not how to format a sequence in FASTA format ask one of the instructors.

(2) How many nucleotides has "`mis.seq`"? What is the GC content of this tiny 'genome'?

(3) Convert "`mis.seq`" to a vector of bases and call it "`mis.0`".

(4) Generate mutated versions of "`mis.seq`" (using function "`mutateDNA()`") with approximately 1, 5 and 10% of the original number of bases changed (mutated). Call such sequences "`mis.1`", "`mis.5`" and "`mis.10`", respectively. Convert back the vectors into sequences and name them "`mis.1.s`", "`mis.5.s`" and "`mis.10.s`", respectively. Put all 4 sequences, "`mis.seq`", "`mis.1.s`", "`mis.5.s`" and "`mis.10.s`" into a single vector, say "`mis.v`".

(5) Obtain the informative sites that exist in "`mis.v`" into the object "`mis.info`".

(6) Save all objects created in this exercise into a file called "`mi_stuff.RData`".

Now we can calculate some kind of 'distance' between the sequences. The first putative distance that we are going to consider is given simply as the number of differences divided by the number of informative sites. A function to calculate such measure and its result with the data at hand are presented in `Box 5`.

```
----------------------------------> Box  5  <--------------------------------------
# The function that I programmed
> SNP.dist
function(x=info.sites()){
        # SNP.dist
        # Obtain a distance matrix from a matrix of informative sites
        # as the one given by the function "info.sites"
        n.s <- nrow(x) # Number of sequences.
        n.b <- ncol(x) # Number of bases in the sequences

        res <- matrix(0, nrow=n.s, ncol=n.s, dimnames=list(c(1:n.s), c(1:n.s)))
        # Makes all pair comparisons
        # filling res elements
        for(i in 1:(n.s-1)){
                for(j in (i+1):n.s){
                        # Give the number of differences
                        res[i,j] <- sum(x[i,] != x[j,])
                }
        }
        as.dist(t(res/n.b))
}
<bytecode: 0x7f9f2cb172a0>

## Look what the function "as.dist" does
> ? as.dist

# Run the function...
> SNP.dist(my.dumy.info) # Running with our matrix of informative sites
          1         2         3
2 0.2857143
3 0.7142857 0.7142857
4 0.7142857 0.7142857 0.2857143
> my.dummy.dis <- SNP.dist(my.dumy.info) # Keep that result in an object
-------------------------------------------------------------------------------------
```

2.1.2. *Excercises.* In your object "`mis.info`" you have a **data frame** with the informative sites to compare the sequences "`mis.seq`", "`mis.1.s`", "`mis.5.s`" and "`mis.10.s`"; i.e., each one of the rows are the corresponding sequences while each column is one informative site (base) within the sequences.

(1) Is your data frame "`mis.seq`" **identical** to the ones of your partners? (Yes or no and why).

(2) Put the names of the sequences ("`mis.seq`", "`mis.1.s`", "`mis.5.s`" and "`mis.10.s`") as the "`row.names`" attribute of "`mis.info`". Hint: try "`attributes(mis.info)$row.names`"

(3) Can you predict **without using** `SNP.dist` the distance between "`mis.seq`" and "`mis.1.s`"? (hint: you could use "`ncol(mis.info)`" and remember how you created "`mis.1.s`").

(4) Define the object "`mis.dis`" as the result of applying the function "`SNP.dist`" to the data "`mis.info`". If you want save all objects with pattern "`ls(patt="mis")`" into a file called "`mi_stuff.RData`".

It appears reasonable to measure the 'distance' between two sequences as the number of SNPs (differences in a single base), divided by the number of informative sites. Such measure could vary between 0 -when

two sequences are identical, to 1 when the two sequences are 'completely different'. Note that this measure is dependent on the initial set of sequences. How can the matrix of distances

```
          1         2         3
2 0.2857143
3 0.7142857 0.7142857
4 0.7142857 0.7142857 0.2857143
```

be interpreted? First note that the representation of the matrix is as 'triangular inferior', i.e., the elements of the diagonal are not presented (because, of course, differences between the same sequence is always 0), also distance is calculated only between different pairs of sequences and the result is one of the cells of this matrix, and given that `distance(i, j) = distance(i, j)` only one pair is presented. If we multiply the distance by the number of informative sites (14), we obtain the matrix of differences (SNP's) between the sequences,

```
> ncol(my.dumy.info)
[1] 14
> 14*my.dummy.dis
   1  2  3
2  4
3 10 10
4 10 10  4
```

Because the matrix `my.dummy.dis` is very small, it contains the $4 \times (4-1)/2 = 6$ distances between all pairs of the 4 sequences, it can be easily interpreted, we could say: "*sequences 1 and 2 as well as 3 and 4 are 'close' (4 SNPs between each one of those two pairs); however, distances between sequences 1 and 3, 1 and 4, 2 and 3 and 2 and 4 are larger.*".

2.2. **Dendrograms.** In any case, if we have n sequences (individual genomes) we will have a matrix with $e = n \times (n-1)/2$ elements; e grows almost quadratically as function of n, for example for $n = 10$ we have $r = 54$, for $n = 50$, $r = 1225$, etc. Thus, it is impossible to directly interpret that information, we must construct some kind of summary. Hierarchical clustering algorithms give a reasonable way to obtain a summary of a distance matrix, by grouping the individuals (in our case sequences) into a tree-like structure called 'dendrogram', or less formally 'tree', which can be more easily interpreted than a numeric matrix. Bifurcating dendrograms are plots which in one of the axis (X) have the set of individuals and in the other (Y) a measure of 'height' or distance at which individuals or clusters are united (see Figure 1). The (vertical) lines are called 'branches' (of the tree) and the points (in the X axis) where two groups are united are called 'nodes'. Before discussing the different algorithms for hierarchical clustering, let's see an example of how to obtain and plot such structure in R (`Box 6`).

```
1  ------------------------------------> Box  6  <--------------------------------------
2  > my.dummy.dis # To remember the content of that object.
3            1         2         3
4  2 0.2857143
5  3 0.7142857 0.7142857
6  4 0.7142857 0.7142857 0.2857143
7  > ? hclust # See the function to obtain the clustering from a distance matrix
8  > my.dummy.clu.1 <- hclust(my.dummy.dis, method = "complete")
9  > plot(my.dummy.clu.1) # Plot shown as Figure 1.
10 ------------------------------------------------------------------------------------
```

Figure 1 shows the result of applying one method ('complete') to our matrix. We obtain a bifurcating tree which hierarchically group all 4 sequences. The 'height' in the Y axis shows the distance between groups estimated by the algorithm. As could we expected sequences 1 and 2 as well as 3 and 4 are grouped first into two groups or 'clusters', say (1,2) and (3,4), in both cases at a distance ≈ 0.28. Then

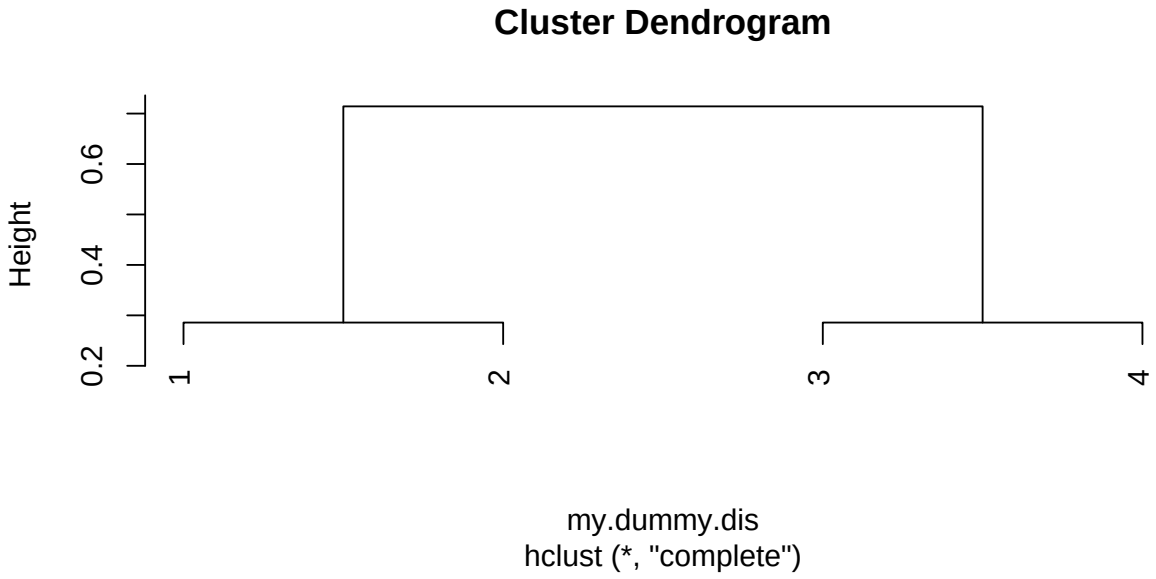


FIGURE 1. R command: '`plot(my.dummy.clu.1)`', See: '`Box 6`'.

the two clusters are grouped together as ((1,2), (3,4)). This notation using parenthesis show the topology or 'form' of the tree. Note that there are other possible tree forms in the case of 4 sequences, for example (((1,2),3),4) or ((1,4),3),2) Can you draw such trees? The 'parenthesis notation' disregards the height at which clustering between groups take place, but otherwise it fully preserve the tree form or topology. The number of different bifurcating trees, n_B, that can be obtained with n objects is given by

$$n_B = \frac{(2n-3)!}{2^{n-2}(n-2)!};\ n > 2 \tag{1}$$

(Felsenstein, 1978) and of course this number grows rapidly as function of n; for example while for $n = 4$ the number of different trees is $n_B = 15$, for $n = 10$ it is $n_B = 34,459,425$.

2.2.1. *Excercises.*

(1) Imagine how a dendrogram obtained by clustering the sequences that you created, and which distance matrix is in "`mis.dis`", will look like. To base that imaginative exercise, remember how you created the sequences involved.

(2) Use the function "`hclust`" to plot a dendrogram of the data in "`mis.dis`". Does it look as you imagined such tree? Can you put the correct labels (instead of numbers) in the dendrogram?

(3) Program a function to calculate n_B. Hint: 5! can be obtained in R using "`factorial(5)`".

It is very important to understand that hierarchical clustering algorithms are **deterministic numerical methods** that transform a matrix distance into a set of bifurcating groups that can be plot as a dendrogram; there is non biological or genetical consideration build into these algorithms. The general and simplified form of the hierarchical clustering algorithm is

(1) Input the set of distances between all different pairs of n individuals (those will be considered as 'clusters' of a single individual). There will be $r = n(n-1)/2$ of such pairs, and the r distances are denoted by $\{d_{ij}\};\ i \neq j, i = 1, 2, \cdots, n-1,\ j = i, i+1, i+2, \cdots, n$.

(2) If the number of elements in $\{d_{ij}\}$ is larger than 1 then go to (3) else go to (4).

(3) Look for the minimum value of $\{d_{ij}\}$, group the clusters corresponding to the $\min(\{d_{ij}\})$ into a cluster, **recalculate** the distances between the clusters and go to (2).

(4) Output the structure of clusters.

This simplified form of the algorithm is sufficient to understand a basic fact: We look for the smallest distance and then consider the clusters with that minimal distance as a new group. The differences between the algorithm methods arise in how to **recalculate** the distance between a newly formed cluster and the remaining ones. To see the problem assume that at some point in the algorithm we have two clusters, say $c_1 = \{a, b, c\}$ and $c_2 = \{d, e\}$ and we need now to calculate the distance $d(c_1, c_2)$. How to do so? R function `hclust` presents various alternatives which can be selected by the "`method`" parameter. Such alternative methods (In R see `> ? hclust`) are presented in the next list

Main methods for hierarchical clustering

`ward :` With variants `ward.D` and `ward.D2`. Minimizes the total within-cluster variance. Aims at finding compact, spherical clusters. See Ward's method.

`single :` Selects the *minimum* distance between the elements of the two clusters as representative. In our example

$$d(c_1, c_2) = d(\{a, b, c\}, \{d, e\}) = \min(d(a, d), d(a, e), d(b, d), d(b, e), d(c, d), d(c, e))$$

See Single-linkage clustering.

`complete :` (the default in `hclust`) Selects the *maximum* distance between the elements of the two clusters as representative. In our example

$$d(c_1, c_2) = d(\{a, b, c\}, \{d, e\}) = \max(d(a, d), d(a, e), d(b, d), d(b, e), d(c, d), d(c, e))$$

In some sense this is the oposite of the `single` method, because it selects as representative distance the one between those two elements (one in each cluster) that are farthest away from each other. See Complete-linkage clustering.

`average :` Also known as 'Unweighted Pair Group Method with Arithmetic mean' or UPGMA for short. This method use as representative distance between two clusters the arithmetic mean of the distances between all pairs; in our example

$$d(c_1, c_2) = d(\{a, b, c\}, \{d, e\}) = \frac{1}{6}(d(a, d) + d(a, e) + d(b, d) + d(b, e) + d(c, d) + d(c, e))$$

See UPGMA.

`mcquitty :` Also known as 'Weighted Pair Group Method with Arithmetic mean' or WPGMA. At each step, the nearest two clusters, say i and j, are combined into a higher-level cluster $i \cup j$. Then, its distance to another cluster k is simply the arithmetic mean of the distances between k and members of $i \cup j$,

$$d_{(i \cup j), k} = \frac{d_{i,j} + d_{j,k}}{2}$$

See the example in WPGMA; to follow the algorithm in our example will demand various steps of the distance matrix.

`median :` In contrast with the `average` (UPGMA) method, the `median` method uses the median of the distances –instead of the mean, to update the distance between clusters; in our example

$$d(c_1, c_2) = d(\{a, b, c\}, \{d, e\}) = median(d(a, d), d(a, e), d(b, d), d(b, e), d(c, d), d(c, e))$$

`centroid :` Centroid linkage clustering, or 'Unweighted Pair Group Method with Centroids' (UPGMC). The 'centroid' or geometric center of a plane figure is the arithmetic mean position of all the points in the figure. This method finds the centroid of each original unit (sequence), and then update the distances by finding for each new cluster its corresponding centroid.

The methods presented in the previous list are not exhaustive –there are others, but we have restricted ourselves to the alternatives that exit in the function `hclust`. In any case, the question is which method is 'the best' for a given set of data, or, in general for any 'genetic distance' between genomes that we could find or define. Unfortunately there is not a single and 'best' answer to such question. In my experience, when the data are 'highly informative', i.e., when all clusters are well defined, the topology of the trees will be the same –or almost the same, independently of the method employed. A second point to select the 'best' method is to include within the genomes studied at least one that could be reasonably considered as an 'out group', i.e., a closely related specie or very different variety. In that case we could select the 'best' method guided by the fact that such 'out group' will be clearly differentiated in the dendrogram; the 'out group' must be in the root of our tree.

2.2.2. *Excercise.* Try all methods available in the function `hclust`, using the distance in "`my.dummy.dis`", to construct and plot dendrograms using all possible methods available (change the `method` in that function). Note and discuss the differences given by the methods; in particular, note if the topology between the dendrograms change from the one estimated in `Box 6` and shown in Figure 1, say: ((1, 2), (3, 4)).

2.3. **Where the data of the *dummy* example came from?** An approach to understand or test data analysis tools is to simulate a small dataset, with known values of the parameters of interest and then try different methods of analysis. This is useful at least in two senses, first, to demonstrate methods that when applied to real (and complex) datasets are difficult to understand or follow and, second, to test novel approaches. Because we know *a priory* the nature of the data, we can test how our analysis recovers data information. R has many tools to generate pseudorandom numbers from different distributions and parameter sets.

If you wondered about the origin of the data in our dummy example, `Box 7` will give you the answer, and can evaluate which method of clustering gives better results in recovering the information present in the data.

```
1  ------------------------------------> Box  7  <---------------------------------------
2  # List the functions that you have in your current environment:
3  > lsf.str() # See the help for that function!
4  info.sites : function (x = my.dummy, out.as.matrix = TRUE)
5  mutateDNA : function (x = randomDNA(50), how.many = 5)
6  randomDNA : function (n)
7  SNP.dist : function (x = info.sites())
8
9  # You also have a vector with the names of objects important for the example:
10 > dummy.stuff
11 [1] "dummy.stuff" "info.sites"  "mutateDNA"   "randomDNA"   "SNP.dist"    "dummy"
12 [7] "my.dummy"    "mis.seq"
13
14 > dummy # What do we have here?
15                                                    Genome
16 reference AACGCATGCGCAGAATAACTAGTGTGACGTGCGCATGGCCACCAAATCCA
17 ancestor  AACACATTCGCAGAATAACTAGTGTGACGTTCGCAGGGCAACCAAATCCA
18 lin1      AACCCATTCGCAGAATAACTAGTGTGACGTTCGCAGGGCATCCAAAGCCA
19 lin1.1    AACCCATTCGCAGAGTAACTAGTGTGACGTCCGCAGGGCATCCAAAGCCA
20 lin1.2    AAGCCATTCGCAGAATAACTAGTGTGACGTTCGCAGGGCATCAAAAGCCA
21 lin2      AACACATTCGCAGAAGAACTAGTGCGACGTTCGCAGGGCAACCAAATCCG
22 lin2.1    AACACATTGGCAGAAGAACTAGTGCGACGTTCGCAGAGCAACCAAATCCG
23 lin2.2    CACACTTTCGCAGAAGAACTAGTGCGACGTTCGCAGGGCAACCAAATCCG
24
```

```
# Which were the sequences that we previously analyzed? (in my.dummy; 4 sequences)
> my.dummy[1]
[1] "AACCCATTCGCAGAGTAACTAGTGTGACGTCCGCAGGGCATCCAAAGCCA"
> my.dummy[1]==dummy$Genome[4] # This is lin1.1
[1] TRUE
> my.dummy[2]==dummy$Genome[5] # This is lin1.2
[1] TRUE
> my.dummy[3]==dummy$Genome[7] # This is lin2.1
[1] TRUE
> my.dummy[4]==dummy$Genome[8] # This is lin2.2
[1] TRUE

# Let's see the functions "randomDNA" and "mutateDNA"
> randomDNA
function(n){
        # randomDNA
        # Produces a random DNA molecule with equiprobable bases
        sample(x=c("A", "T", "G", "C"), size=n, replace = TRUE, prob = NULL)
}
<bytecode: 0x7f9f2cb16070>
> mutateDNA
function(x=randomDNA(50), how.many=5){
        # mutateDNA
        # Input:
        #        x - A DNA molecule
        #        how.many - How many "true" mutations (not mute mutations)
        # Output: The mutated DNA.

        n <- length(x) # Length of the DNA
        pos.2.change <- sample(x=c(1:n), size=how.many, replace = FALSE)
        for(i in 1:how.many){
                x[pos.2.change[i]] <- sample(x=setdiff(c("A", "T", "G", "C"),
                x[pos.2.change[i]]), size=1)
        }
        x
}
<bytecode: 0x7f9f2cb12ee8>

# Let's see two examples (NOTE your output will be different from the one shown, why?)
> temp <- randomDNA(15)
> temp
 [1] "C" "G" "T" "C" "G" "C" "A" "G" "T" "G" "A" "T" "G" "G" "T"
> temp2 <- mutateDNA(temp, how.many=5)
> temp2
 [1] "A" "G" "T" "G" "G" "G" "T" "G" "T" "C" "A" "T" "G" "G" "T"
> temp == temp2
 [1] FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[15]  TRUE
> sum(temp == temp2)
[1] 10
```

```
# Here is the history to create the dataset 'dummy' (output not shown)
# > set.seed(1959) # To obtain exactly the same results than in my run
# reference <- randomDNA(50)
# ancestor <- mutateDNA(reference, how.many=5)
# lin1 <- mutateDNA(ancestor, how.many=3)
# lin1.1 <- mutateDNA(lin1, how.many=2)
# lin1.2 <- mutateDNA(lin1, how.many=2)
# lin2 <- mutateDNA(ancestor, how.many=3)
# lin2.1 <- mutateDNA(lin2, how.many=2)
# lin2.2 <- mutateDNA(lin2, how.many=2)
# dummy <- data.frame(matrix(c(paste(reference, collapse=""), paste(ancestor, collapse=""),
    paste(lin1,  collapse=""), paste(lin1.1,  collapse=""), paste(lin1.2,  collapse=""),
    paste(lin2,  collapse=""), paste(lin2.1,  collapse=""), paste(lin2.2,  collapse="")),
    nrow=8, ncol=1, dimnames=list(c("reference", "ancestor", "lin1", "lin1.1", "lin1.2",
    "lin2", "lin2.1", "lin2.2"), "Genome")), stringsAsFactors=FALSE)
-------------------------------------------------------------------------------
```

Function `randomDNA` is a very simple way to 'simulate' a completely random DNA molecule, Do you understand each one of the steps? Can you see ways to do it more 'realistic'? On the other hand, `mutateDNA` change a *fixed* number of the bases located 'at random' in the molecule for different bases; again, Do you understand each one of the steps? Can you see ways to do it more 'realistic'?

2.3.1. *Excercises.* A point mutation, which will cause an SNP, can be assumed to happen at a relatively fixed rate in time, generations or number of DNA replications. The number of point mutations that happen in a DNA molecule in a fixed time (or a fixed number of replication or generations) is likely to follow a Poisson distribution. This distribution depends on a single parameter, called λ (lambda). In R we can generate random numbers with Poisson distribution with the function "`rpois(n, lambda)`", where "n" is the number of random data to be generated and and "lambda" is the aforementioned rate of expected changes (mutations).

(1) To familiarize yourself with the Poisson distribution simulate 'large' sets ($n = 1000$?) of random numbers with different values of λ, for example $\lambda = 1, 10$ and 100. For each simulated dataset obtain: a) summary statistics (use "`summary()`"), variance (use "`var()`") and construct an histogram (use "`hist()`") to answer the following questions:

   (a) Is it true that the mean and the variance in the Poisson distribution are approximately equal between them and also equal to λ?

   (b) Is it true that, as λ increases the Poisson distribution tends to be symmetric around λ?

(2) Modify the function "`mutateDNA(x, how.many)`" to make it more 'realistic' in the sense that instead of a fixed number of mutations given by the parameter "`how.many`" it will produce a random number of mutations which follow the Poisson distribution with a given value of "`lambda`". You can first copy the function into an object; for example "`my.mutateDNA <- mutateDNA`" and then make the changes using a text editor; for example in R you could use "`my.mutateDNA <- edit(my.mutateDNA)`".

(3) From your sequence "`mis.seq`" (or, more easily from the equivalent representation "`mis.0`") obtain 5 sequences with a random number of mutations arising from the Poisson distribution with $\lambda = 100$; names such sequences as "`m.1`", "`m.2`", $\cdots$, "`m.5`" and construct a dendrogram for those sequences using the method "`average`".

(4) Briefly discuss your results.

From the 'history' presented at the bottom of `Box 7` we can see which was my intention when simulating this data. In the context of a ridiculously small genome –only 50 bp long, I first simulated a sequence

to be taken as 'reference'. Then, a sequence named 'ancestor' was obtained by mutating 5 bases from 'reference'. Furthermore, two independent 'linages' ('lin1' and 'lin2') were obtained from 'ancestor' by mutating 3 bases in each case. In a next step we obtained two 'descendents' of each one of the two linages, lin1.1 and lin1.2 from lin1 and lin2.1 and lin2.2 from lin2. In each case, lin1.1 and lin1.2 are separated from each other by two mutations from lin1, and the same happens for lin2.1 and lin2.2 which are separated from each other by two mutations arising independently from lin1. The true topology of the simulated tree is given by

(reference, (ancestor, ((lin1, (lin1.1, lin1.2)), (lin2, (lin2.1, lin2.2)))))

Do you agree? If we assume that the process could follow a molecular clock, i.e., if we assume that the number of mutations is roughly linear with time, the total length of the branches separating 'reference' from 'ancestor' will be $5X$, where X is the average time (in generations or in years?) that takes a single mutation to happen. In a similar way we will expect that 'reference' and 'lin1.1' will be separated by $(5+3+2)X = 10X$ (note that I am neglecting the possibility of two mutations in different generations happening at the same site). Let's see how much of the information in the simulated process can be recovered from our tools.

```
1  ------------------------------------> Box  8  <---------------------------------------
2  # First, isolate the set of informative sites in the 8 sequences
3  > dummy.info <- info.sites(dummy$Genome) # There are 8 sequences there
4  > dummy.info
5    b1 b3 b4 b6 b8 b9 b15 b16 b25 b31 b36 b37 b40 b41 b43 b47 b50
6  1  A  C  G  A  G  C   A   T   T   G   T   G   C   A   C   T   A
7  2  A  C  A  A  T  C   A   T   T   T   G   G   A   A   C   T   A
8  3  A  C  C  A  T  C   A   T   T   T   G   G   A   T   C   G   A
9  4  A  C  C  A  T  C   G   T   T   C   G   G   A   T   C   G   A
10 5  A  G  C  A  T  C   A   T   T   T   G   G   A   T   A   G   A
11 6  A  C  A  A  T  C   A   G   C   T   G   G   A   A   C   T   G
12 7  A  C  A  A  T  G   A   G   C   T   G   A   A   A   C   T   G
13 8  C  C  A  T  T  C   A   G   C   T   G   G   A   A   C   T   G
14 > attributes(dummy)$row.names # The names of the sequences
15 [1] "reference" "ancestor"  "lin1"      "lin1.1"    "lin1.2"    "lin2"      "lin2.1"
16 [8] "lin2.2"
17 > attributes(dummy.info)$row.names <- attributes(dummy)$row.names
18 > dummy.info
19           b1 b3 b4 b6 b8 b9 b15 b16 b25 b31 b36 b37 b40 b41 b43 b47 b50
20 reference  A  C  G  A  G  C   A   T   T   G   T   G   C   A   C   T   A
21 ancestor   A  C  A  A  T  C   A   T   T   T   G   G   A   A   C   T   A
22 lin1       A  C  C  A  T  C   A   T   T   T   G   G   A   T   C   G   A
23 lin1.1     A  C  C  A  T  C   G   T   T   C   G   G   A   T   C   G   A
24 lin1.2     A  G  C  A  T  C   A   T   T   T   G   G   A   T   A   G   A
25 lin2       A  C  A  A  T  C   A   G   C   T   G   G   A   A   C   T   G
26 lin2.1     A  C  A  A  T  G   A   G   C   T   G   A   A   A   C   T   G
27 lin2.2     C  C  A  T  T  C   A   G   C   T   G   G   A   A   C   T   G
28
29 > dummy.dis <- SNP.dist(dummy.info)
30 > dummy.dis
31           1         2         3         4         5         6         7
32 2 0.2941176
33 3 0.4117647 0.1764706
34 4 0.4705882 0.2941176 0.1176471
35 5 0.5294118 0.2941176 0.1176471 0.2352941
```

```
6 0.4705882 0.1764706 0.3529412 0.4705882 0.4705882
7 0.5882353 0.2941176 0.4705882 0.5882353 0.5882353 0.1176471
8 0.5882353 0.2941176 0.4705882 0.5882353 0.5882353 0.1176471 0.2352941
> attributes(dummy.dis) # See the attributes of that object
$Labels
[1] "1" "2" "3" "4" "5" "6" "7" "8"
# ... Other attributes
> attributes(dummy.dis)$Labels <- attributes(dummy.info)$row.names
> dummy.dis
         reference  ancestor      lin1    lin1.1    lin1.2      lin2    lin2.1
ancestor 0.2941176
lin1     0.4117647 0.1764706
lin1.1   0.4705882 0.2941176 0.1176471
lin1.2   0.5294118 0.2941176 0.1176471 0.2352941
lin2     0.4705882 0.1764706 0.3529412 0.4705882 0.4705882
lin2.1   0.5882353 0.2941176 0.4705882 0.5882353 0.5882353 0.1176471
lin2.2   0.5882353 0.2941176 0.4705882 0.5882353 0.5882353 0.1176471 0.2352941
> ncol(dummy.info) # Number of informative sites
[1] 17
> 17*dummy.dis # Row number of SNPs (mutations between ndividuals)
         reference ancestor lin1 lin1.1 lin1.2 lin2 lin2.1
ancestor         5
lin1             7        3
lin1.1           8        5    2
lin1.2           9        5    2      4
lin2             8        3    6      8      8
lin2.1          10        5    8     10     10    2
lin2.2          10        5    8     10     10    2      4

> 17*dummy.dis # Row number of SNPs (mutations between ndividuals)
         reference ancestor lin1 lin1.1 lin1.2 lin2 lin2.1
ancestor         5
lin1             7        3
lin1.1           8        5    2
lin1.2           9        5    2      4
lin2             8        3    6      8      8
lin2.1          10        5    8     10     10    2
lin2.2          10        5    8     10     10    2      4

> ? hclust # To remember the methods:
# "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA),
#  "median" (= WPGMC) or "centroid"
# I will multiply by 17 the distance to measure height directly in "SNPs".
> plot(hclust(17*dummy.dis, "ward.D")) # Figure 3
> plot(hclust(17*dummy.dis, "ward.D2")) # Not shown
> plot(hclust(17*dummy.dis, "single")) # Figure 2
> plot(hclust(17*dummy.dis, "complete")) # Not shown
> plot(hclust(17*dummy.dis, "average")) # Figure 4
> plot(hclust(17*dummy.dis, "mcquitty")) # Not shown
> plot(hclust(17*dummy.dis, "median")) # Figure 5
> plot(hclust(17*dummy.dis, "centroid")) # Not shown
```

```
## Notes of the topologies for each method:
"ward.D" => ((lin2.2, (lin2, lin2.1)), ((lin1.2, (lin1,lin1.1)), (reference, ancestor)))
"ward.D2" => ((lin2.2, (lin2, lin2.1)), ((lin1.2, (lin1,lin1.1)), (reference, ancestor)))
"single" => (reference, ((lin2.2, lin2, lin2.1), ancestor, (lin1.2, lin1, lin1.1)))
"complete" => ((lin2.2, (lin2, lin2.1)), ((lin1.2, (lin1,lin1.1)), (reference, ancestor)))
"average" => ((lin2.2, (lin2, lin2.1)), (reference, (ancestor, (lin1.2, (lin1,lin1.1)))))
"mcquitty" => ((lin2.2, (lin2, lin2.1)), (reference, (ancestor, (lin1.2, (lin1,lin1.1)))))
"median" => (reference, ((lin2.2, (lin2, lin2.1)), (ancestor, (lin1.2, (lin1,lin1.1)))))
"centroid" => (reference, ((lin2.2, (lin2, lin2.1)), (ancestor, (lin1.2, (lin1,lin1.1)))))

# Grouping dendrograms by topology
"single" - Stands alone (almost not bifurcating because ties?)
"ward.D" = "ward.D2" = "complete" # (reference, ancestor)
"average" = "mcquitty"
"median" = "centroid"
--------------------------------------------------------------------------------
```

Figures 2 to 5 present the four different topologies that we obtain by employing the 8 methods available in function `hclust`. Figure 2, obtained with the 'single' method, i.e., employing the minimum distance (mutations) between sequences to measure the distance between clusters is the one that gives a topology closer to the expected,

(reference, (ancestor, ((lin1, (lin1.1, lin1.2)), (lin2, (lin2.1, lin2.2)))))

However, it is important to notice that this topology arises because there are ties in the (minimum) distances between clusters; in fact the dendrogram in Figure 2 **is** bifurcating, but with the ties some of the distances are too small to be appreciated. The next topology more alike to the one expected is given in Figure 5, and this topology is equal for methods 'median' and 'centroid', presenting the 'reference' sequence as root of the tree as we know is the true case. An important fact to take into account in

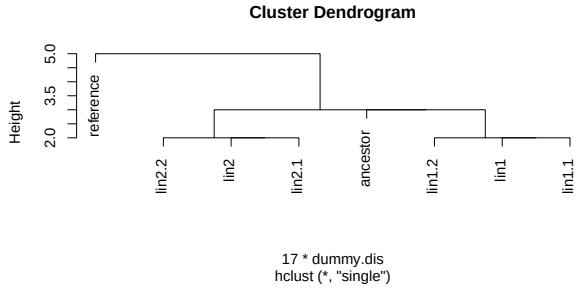


FIGURE 2. Single 'almost' not bifurcating (unique topology), See: '`Box 8`'.

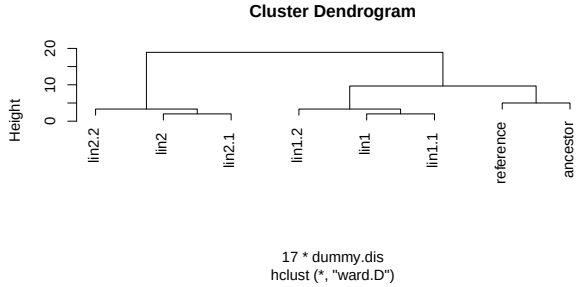


FIGURE 3. Topology: `"ward.D" = "ward.D2" = "complete"`, See: '`Box 8`'.

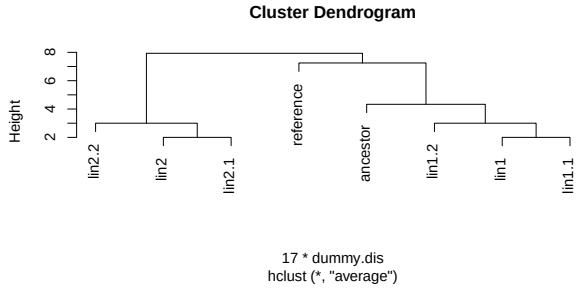


FIGURE 4. Topology: `"average" = "mcquitty"`, See: '`Box 8`'.

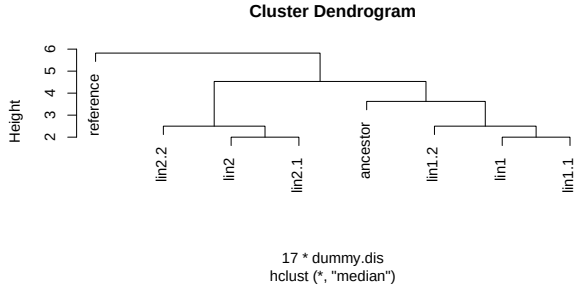


FIGURE 5. Topology: `"median" = "centroid"`, See: '`Box 8`'.

figures 3, 4 and 5 is that the 'parent' sequences 'lin1' and 'lin2' appear 'closer' to one of the sequences that are obtained from them, lin2.1 is 'closer' to lin2 than the distance shown in the dendrogram between lin2.1 and lin2.2. The methods are not really misbehaving, in fact, as can be seen in the distance matrix (`Box 8`) the number of mutations between lin1.1 and lin1.2 is 4, while the number of mutations between lin1 and lin1.1 as well as between lin1 and lin1.2 are 2 –and the same is true for the lin2 'family'; given the independent 'evolution' of lin1.1 and lin1.2 from lin1, these two sequences have diverged more between them that with respect to their 'parent' sequence, lin1. Ties in the process of clustering are resolved by grouping first the first of two or more equal distances that is found (this of course is arbitrary, but there is no better solution). Fortunately, as the number of bases considered in the sequences (the 'genome size') increases, this ties are less and less likely; in real cases the probability of a tie in the process can be ignored. It is a very extended misconception to think that a dendrogram recapitulate the true evolutionary tree; a dendrogram just help to visualize as a tree the information given by the distance matrix, and such matrix does NOT contains the full story of the evolution of the sequences.

In `Box 9` we analyze the numeric structure resulting from function `hclust`.

```
1  ------------------------------------> Box  9  <---------------------------------------
2  # Structure of our distance matrix
3  > 17*dummy.dis
4           reference ancestor lin1 lin1.1 lin1.2 lin2 lin2.1
5  ancestor         5
6  lin1             7        3
7  lin1.1           8        5    2
8  lin1.2           9        5    2      4
9  lin2             8        3    6      8      8
10 lin2.1          10        5    8     10     10    2
11 lin2.2          10        5    8     10     10    2      4
12
13 > dummy.clu.med <- hclust(17*dummy.dis, method="median") # A cluster object
```

```
> dummy.clu.med # If we ask to see (print) the object...

Call:
hclust(d = 17 * dummy.dis, method = "median")

Cluster method   : median
Number of objects: 8

> class(dummy.clu.med) # Which kind of class is it?
[1] "hclust"

# However, the "primitive" form of this object is a "list", see
> length(dummy.clu.med) # How many components?
[1] 7
> names(dummy.clu.med) # Which names?
[1] "merge"       "height"      "order"       "labels"      "method"      "call"
[7] "dist.method"

## See the description of the output with
> ? hclust # See section "Value"
# and analyze each component

> dummy.clu.med$merge # Or also dummy.clu.med[[1]]
     [,1] [,2]
[1,]   -3   -4
[2,]   -6   -7
[3,]   -5    1
[4,]   -8    2
[5,]   -2    3
[6,]    4    5
[7,]   -1    6
# merge - an n-1 by 2 matrix.
# Row i of merge describes the merging of clusters at step i of the clustering.
# If an element j in the row is negative, then observation -j was merged at this stage.
# If j is positive then the merge was with the cluster formed at the (earlier)
        stage j of the algorithm.
# Thus negative entries in merge indicate agglomerations of singletons,
        and positive entries indicate agglomerations of non-singletons.

> dummy.clu.med$height # The second component, dummy.clu.med[[2]]
[1] 2.000000 2.000000 2.500000 2.500000 3.625000 4.531250 5.820312
# height - a set of n-1 real values (non-decreasing for ultrametric trees).
# The clustering height: that is, the value of the criterion associated
#        with the clustering method for the particular agglomeration.

> dummy.clu.med$order # Also dummy.clu.med[[3]]
[1] 1 8 6 7 2 5 3 4
# order - a vector giving the permutation of the original observations suitable for plotting,
#        in the sense that a cluster plot using this ordering and matrix merge
#        will not have crossings of the branches.

> dummy.clu.med$labels # Also dummy.clu.med[[4]]
```

```
[1] "reference" "ancestor"  "lin1"      "lin1.1"    "lin1.2"    "lin2"      "lin2.1"
[8] "lin2.2"
# labels - labels for each of the objects being clustered.

> dummy.clu.med$call # Also dummy.clu.med[[5]]
hclust(d = 17 * dummy.dis, method = "median")
# call - the call which produced the result.
-------------------------------------------------------------------------------------
```

Knowing the content of an `hclust` we gain a better understanding of how the results of the procedure are kept and give us the possibility of modify the plot of the results. For example, we could want to plot the units in a different order, or create a custom tree, etc.

How R know how to plot (with the "`plot()`" function) a dendrogram? The trick is that "`plot()`" has different methods for different objects; it see the class of an object, and if there is an specific method for that class it uses it. Let's see

```
------------------------------------> Box  10  <--------------------------------------
# Let's play with some of the parameters:
> plot(dummy.clu.med, hang=0) # Result not shown as figure
> plot(dummy.clu.med, hang=0.1) # Result not shown as figure

# Let's see the methods that exist for function "plot"
> methods(plot)
 [1] plot.acf*           plot.data.frame*    plot.decomposed.ts* plot.default
 [5] plot.dendrogram*    plot.density*       plot.ecdf           plot.factor*
 [9] plot.formula*       plot.function       plot.hclust*        plot.histogram*
[13] plot.HoltWinters*   plot.isoreg*        plot.lm*            plot.medpolish*
[17] plot.mlm*           plot.ppr*           plot.prcomp*        plot.princomp*
[21] plot.profile.nls*   plot.raster*        plot.spec*          plot.stepfun
[25] plot.stl*           plot.table*         plot.ts             plot.tskernel*
[29] plot.TukeyHSD*
see '?methods' for accessing help and source code
> ?methods

# Ask help SPECIFICALLY for the plot.dendrogram method:
> ? plot.dendrogram
# Let's convert our object "dummy.clu.med" to a dendrogram:
> dummy.med.den <- as.dendrogram(dummy.clu.med)
# Now in "plot(dummy.med.den)" we can use different options present in "? plot.dendrogram".
# Different options:
> plot(dummy.med.den, type="triangle")
> plot(dummy.med.den, center=T)
> plot(dummy.med.den, edge.root=T)
> plot(dummy.med.den, horiz=T)

# Modifying the edges (lines) of our dendrogram
> my.edgePar <- vector("list", 3) # Define a list with three components
> names(my.edgePar) <- c("col", "lty", "lwd") # Color, line type and line width
> my.edgePar$col <- "red"
> my.edgePar$lty <- 2
> my.edgePar$lwd <- 2
> plot(dummy.med.den, edgePar=my.edgePar)
```

```
> my.edgePar$col <- c("red", "brown")
> my.edgePar$lty <- 1
> my.edgePar$lwd <- 2
> plot(dummy.med.den, edgePar=my.edgePar)
> plot(dummy.med.den, edgePar=my.edgePar, type="triangle")

# Modifying the nodes
> my.nodePar <- vector("list", 3)
> names(my.nodePar) <- c("pch", "cex", "col")
> my.nodePar$pch <- c(1,2)
> my.nodePar$cex <- c(1,2)
> my.nodePar$col <- c("red", "brown")
> plot(dummy.med.den, nodePar=my.nodePar)
> plot(dummy.med.den, edgePar=my.edgePar, nodePar=my.nodePar)
-------------------------------------------------------------------------------
```

Other two useful function to work with results of `hclust` are

`rect.hclust :` Draws rectangles around the branches of a dendrogram highlighting the corresponding clusters. First the dendrogram is cut at a certain level, then a rectangle is drawn around selected branches.

`cutree :` Cuts a tree, e.g., as resulting from hclust, into several groups either by specifying the desired number(s) of groups or the cut height(s).

(the list above was taken from the help of the functions; see the full help for those functions). Let's see examples of those functions

```
-----------------------------------> Box  11  <-------------------------------------
# Using dummy.clu.med
> rect.hclust(dummy.clu.med, h=3, which=c(2,4))
> rect.hclust(dummy.clu.med, h=3, which=c(2,4), border="darkviolet")
> rect.hclust(dummy.clu.med, h=3, which=c(2,4), border=c("red","blue"))

> cutree(dummy.clu.med, k=2)
reference  ancestor      lin1    lin1.1    lin1.2      lin2    lin2.1    lin2.2
        1         2         2         2         2         2         2         2

> cutree(dummy.clu.med, k=c(2:5))
          2 3 4 5
reference 1 1 1 1
ancestor  2 2 2 2
lin1      2 2 3 3
lin1.1    2 2 3 3
lin1.2    2 2 3 3
lin2      2 3 4 4
lin2.1    2 3 4 4
lin2.2    2 3 4 5
-------------------------------------------------------------------------------
```

We have used a very particular way to represent the informative sites of a set of sequences trough the function `info.sites(x)`, and then used the function `SNP.dist()` to obtain the distances between sequences, as the number of SNPs (mutations) divided by the number of informative sites. Now we are going to change the representation obtained with `info.sites(x)` into a more standard form, where each

one of the informative sites is converted into 4 different columns, each one of this with the presence (denoted as 1) or absence (denoted as 0) of the four possible DNA bases. See `Box 12`

```
1  ------------------------------------> Box  12  <---------------------------------------
2  # Function to convert a matrix of informative sites into a matrix
3  # that could be used by the "dist" function.
4  # You can input the function using
5  > source("info.sites2num.txt") # That file exist in your directory
6
7  # Or you can copy the function and paste it in your R window:
8  # Begin to copy:
9  info.sites2num <- function(x = dummy.info){
10         # info.sites2num
11         # Converts a matrix of informative sites to a matrix
12         # that can be used by the "dist" function.
13         ncx <- ncol(x)
14         nrx <- nrow(x)
15         r.nam <- attributes(x)$row.names
16         c.nam <- names(x)
17         bases <- c("A","T","G","C")
18         res <- matrix(0, nrow=nrx, ncol=4*ncx, dimnames=
19         list(r.nam, paste(rep(c.nam, each=4), rep(bases, ncx), sep="")))
20         l <- 0 # Index for column of res
21         for(i in 1:ncx){
22                 # For each column of the original matrix
23                 k <- 0 # Index for the base
24                 for(j in 1:4){
25                         k <- k+1
26                         l <- l + 1
27                         res[,l] <- 1*(x[,i] == bases[j])
28                 }
29         }
30         res
31 }
32 # End copy
33 > dummy.dat <- info.sites2num(x = dummy.info) # Obtains the new form of the data
34
35 # Checking
36 > dummy.dat[, 1:4] # All rows and the first 4 columns of the new matrix
37           b1A b1T b1G b1C
38 reference   1   0   0   0
39 ancestor    1   0   0   0
40 lin1        1   0   0   0
41 lin1.1      1   0   0   0
42 lin1.2      1   0   0   0
43 lin2        1   0   0   0
44 lin2.1      1   0   0   0
45 lin2.2      0   0   0   1
46 > dummy.info[,1] # This values were converted into the data above.
47 [1] "A" "A" "A" "A" "A" "A" "A" "C"
48
49 # Also for example:
```

```
> dummy.info[,4:5]
          b6 b8
reference  A  G
ancestor   A  T
lin1       A  T
lin1.1     A  T
lin1.2     A  T
lin2       A  T
lin2.1     A  T
lin2.2     T  T
> dummy.dat[,13:20]
          b6A b6T b6G b6C b8A b8T b8G b8C
reference   1   0   0   0   0   0   1   0
ancestor    1   0   0   0   0   1   0   0
lin1        1   0   0   0   0   1   0   0
lin1.1      1   0   0   0   0   1   0   0
lin1.2      1   0   0   0   0   1   0   0
lin2        1   0   0   0   0   1   0   0
lin2.1      1   0   0   0   0   1   0   0
lin2.2      0   1   0   0   0   1   0   0
# Etc.

> ? dist # To see the options of the "dist" function.
# method can be "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".

## Now, see that:
# Our distance given in mutations
# (obtained with my.dummy.dis <- SNP.dist(my.dumy.info))
> 17*dummy.dis
         reference ancestor lin1 lin1.1 lin1.2 lin2 lin2.1
ancestor         5
lin1             7        3
lin1.1           8        5    2
lin1.2           9        5    2      4
lin2             8        3    6      8      8
lin2.1          10        5    8     10     10    2
lin2.2          10        5    8     10     10    2      4

# Is equal than
> (dist(dummy.dat, method="euclidean")^2) / 2
         reference ancestor lin1 lin1.1 lin1.2 lin2 lin2.1
ancestor         5
lin1             7        3
lin1.1           8        5    2
lin1.2           9        5    2      4
lin2             8        3    6      8      8
lin2.1          10        5    8     10     10    2
lin2.2          10        5    8     10     10    2      4

# Now you can try some alternative distances and clustering methods!
# Compare this with Figure 2
> plot(hclust(dist(dummy.dat, method="euclidean"), "single"))
```

```
# Other possibilities...
> plot(hclust(dist(dummy.dat, method="euclidean")^2, "median"))
> plot(hclust(dist(dummy.dat, method="euclidean")^2, "average"))
> plot(hclust(dist(dummy.dat, method="euclidean"), "average"))
# Changing distances
> plot(hclust(dist(dummy.dat, method="manhattan"), "average"))
> plot(hclust(dist(dummy.dat, method="binary"), "average"))
> plot(hclust(dist(dummy.dat, method="binary"), "complete"))
## You have 6 distances and 8 clustering methods: 48 possible combinations!
-------------------------------------------------------------------------------
```

2.3.2. *Excercises.* The function "`hclust`" produces an structure with all necessary parts to plot an histogram. In this exercise you are going to learn exactly what is inside such structures by modifying them. We begin by creating a 'template', i.e., a, dummy dendrogram for 5 objects.

```
-------------------------------------------------------------------------------
# First create a random matrix of 5 rows and 10 columns
set.seed(1959) # For you and me to have the same results
temp.r <- matrix(rnorm(50), nrow=5, ncol=10, dimnames=list(letters[1:5], c(1:10)))

# Now, create a matrix of (Euclidean) distances between the rows
temp.d <- dist(temp.r)
# Now we create an object:
temp.hc <- hclust(temp.d, method="single")

> temp.hc

Call:
hclust(d = temp.d, method = "single")

Cluster method   : single
Distance         : euclidean
Number of objects: 5

> class(temp.hc)
[1] "hclust"

# However, this object is ALSO a list; see
> names(temp.hc)
[1] "merge"       "height"      "order"       "labels"      "method"      "call"
[7] "dist.method"

# Let's see some components
> temp.hc$merge
     [,1] [,2]
[1,]   -2   -5
[2,]   -4    1
[3,]   -1    2
[4,]   -3    3
# NOTE: Negative values are original objects with labels
> temp.hc$labels
[1] "a" "b" "c" "d" "e"
```

```
# And the "height"
> temp.hc$height
[1] 4.018810 4.162407 4.653369 5.020059

## See the plot of the dendrogram
plot(temp.hc)

# Now, you can use this template to modify
temp2.hc <- temp.hc # Make a copy
# Modify one of the labels:
temp2.hc$labels[1] <- "A"
plot(temp2.hc) # And see the result
temp2.hc$height <- c(1,2,3,4) # Modify the heigths
plot(temp2.hc) # See the result
temp2.hc$order[4:5] <- c(5,2) # Modify part of the order
plot(temp2.hc) # See the result
## ... You can play more with this
-------------------------------------------------------------------------------
```

Using the above knowledge you will be able to directly 'construct' dendrograms.

(1) Make and plot dendrogram with topology (((A,B),C), (D,E)) [Use any suitable heights].

(2) Make and plot dendrogram with topology (A,(B,(C,(D,E)))) [Use any suitable heights].

2.4. **Assessing the uncertainty in hierarchical clustering.** In any scientific endeavor, we must measure how reliable are hour results, because it will tell how robust are the biological conclusions that could be infer from the analysis. Here we are going to estimate how robust is a given dendrogram using the R package "`pvclust`" (Suzuki and Shimodaira, 2006) in our 'dummy' example. We have some number of SNPs, and the idea is to use a re-sampling method as bootstrap which briefly consist in obtaining samples with replacement from the data and calculating the statistic of interest. For dendrograms the process consist in taking a sample of the data (SNPs) and calculating a dendrogram, evaluating if each node of the original dendrogram re-appears in the one obtained with the random sample. By repeating this process many times we can see if a node is 'robust', i.e., it appears in a high percentage of the samples, or weak, appearing only in a small proportion of the samples. In `Box 13` we exemplify the use of the package "`pvclust`" in our example.

```
1  ------------------------------------> Box  13  <--------------------------------------
2  # You need to have installed the package "pvclust" into your computer!
3  > library(pvclust) # Loads the package for its use.
4  # See the help for the package
5  > ? pvclust
6  # Or even better, visit web site
7  # http://stat.sys.i.kyoto-u.ac.jp/prog/pvclust/
8
9  # Perform the procedure
10 system.time(
11 dummy.pv.1 <- pvclust(data=t(dummy.dat), method.hclust="average",
12         method.dist="euclidean", nboot=1000, iseed=1959)
13 )
14 Bootstrap (r = 0.5)... Done.
15 Bootstrap (r = 0.59)... Done.
16 Bootstrap (r = 0.69)... Done.
17 Bootstrap (r = 0.79)... Done.
```

```
Bootstrap (r = 0.9)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.09)... Done.
Bootstrap (r = 1.19)... Done.
Bootstrap (r = 1.29)... Done.
Bootstrap (r = 1.4)... Done.
   user  system elapsed
  6.680   0.089   6.737
> dummy.pv.1

Cluster method: average
Distance      : euclidean

Estimates on edges:

     au    bp se.au se.bp      v     c  pchi
1 0.522 0.473 0.030 0.005  0.006 0.062 0.433
2 0.561 0.478 0.030 0.005 -0.049 0.104 0.457
3 0.776 0.663 0.023 0.005 -0.589 0.169 0.255
4 0.777 0.674 0.023 0.005 -0.606 0.155 0.251
5 0.662 0.389 0.029 0.005 -0.067 0.349 0.246
6 0.613 0.346 0.030 0.005  0.055 0.342 0.056
7 1.000 1.000 0.000 0.000  0.000 0.000 0.000

> plot(dummy.pv.1) # Result presented as Figure 6.
```

Interpretation of these results will be briefly review in the class.

2.5. **Conclusions of this section.** To understand the basics of SNP variations between sequences we saw a simple example of how to generate an analyze a ridiculously small genome (only 50 bp). However, these results are of general application for sequences that differ only in SNPs, i.e., sequences of the same size that are aligned among them in its whole length. In real cases genomes evolve not only by point mutations (SNPs), but also by including more complex mechanisms, as for example, insertion or deletion of bases (called 'indels'), translocations or other chromosomal rearrangements, polyploidy, etc. Modeling and evaluating all these complex processes is difficult, because the plethora of possibilities impede to setup a reasonable probability framework. In contrast, for SNPs it is possible to estimate the rates of mutation and thus employ more sophisticated ways to evaluate genetic diversity. Unfortunately we will not have time here to present and discuss such methods, as for example maximum parsimony or maximum likelihood, etc.

Molecular phylogenetics and the tools of computational phylogenetics are rapidly advancing from the analysis at gene level to the examination of whole genomes. Synteny, i.e., co-localization of genes among genomes, helps to study and understand evolution of species, and the almost complete synteny of genomes within the a single specie allows the use of SNP methods to study intra-specific diversity. Also, many molecular markers (see Table 1) will present polymorphisms (different forms) which are based, at the bottom level, on SNPs; for example, SNPs can change the recognition sites of enzymes that cut DNA and thus produce different 'alleles' of the markers.

3. Maize data

The data that we are going to analyze here are microsatellites also called 'short tandem repeats' (STR) by forensic geneticists or 'simple sequence repeats' (SSR) by plant geneticists. SSR loci have conserved

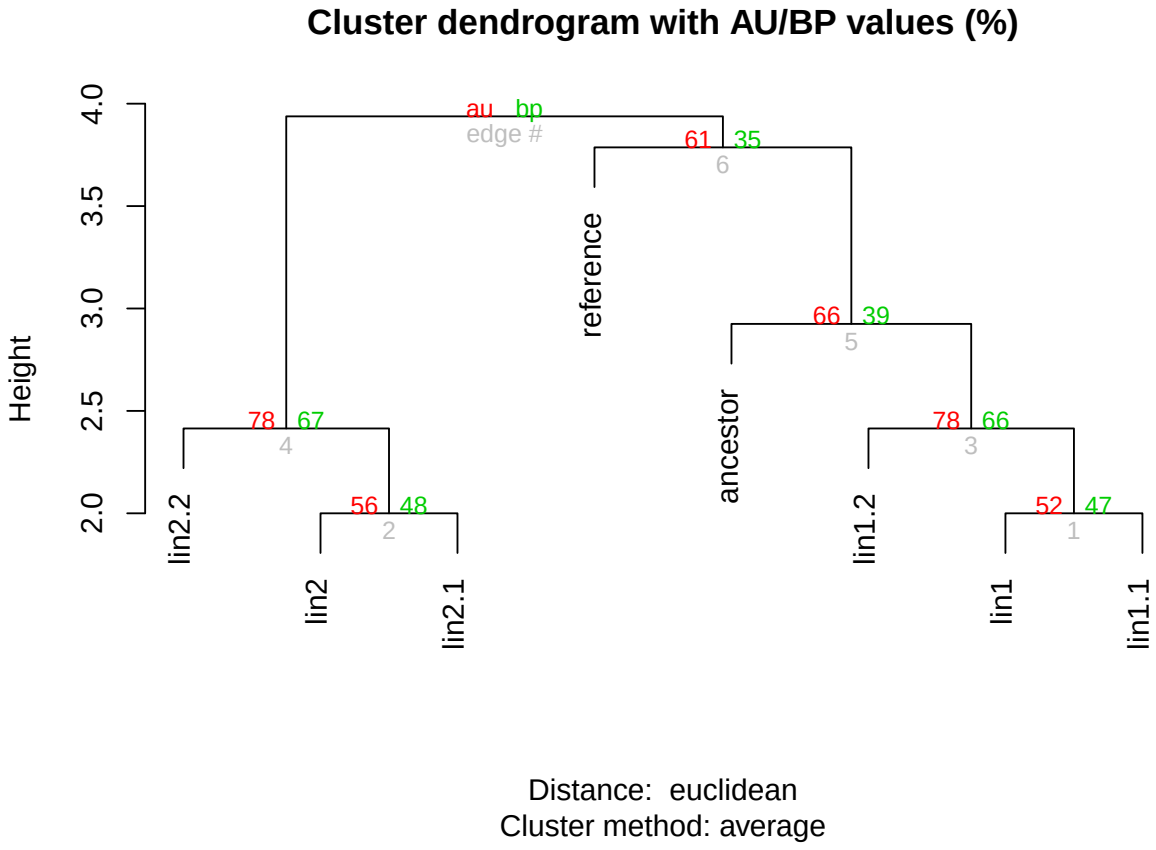


FIGURE 6. R command: '`plot(dummy.pv.1)`', See: '`Box 13`'.

flanking regions, and between those there is a motif (ranging in length from one to six or more base pairs) which are repeated a variable number of times. The number of times that the motif is repeated determines the 'allele' of an SSR. Using primers for the flanking regions, an SSR locus can be amplified by PCR and sequenced, determining the specific allele that is present. SSRs appear in many places on the genomes, thus we can consider that these markers reflect an important component of genomic diversity. Also these markers have a high mutation rate, making them an ideal way to study diversity. Other important aspect is that in contrasts with SNPs, which can present only 4 variants, the number of alleles in SSRs is usually much higher, and they are inherited in a 'codominant' way, i.e., if two different alleles are present in an individual, then we can directly detect the heterozygous.

The data that we are going to analyze are from a project lead by June Simpson of Cinvestav Irapuato. The idea of the study was to analyze the diversity of an ample set of maize land races or 'accessions' from many Mexican localities and with a large phenotypic diversity. Having a limit in the number of sequencing reactions that could be performed, we studied different strategies for the design of the research. The option was to perform analyses of individual plants or, alternatively, to group plants in 'bulks' analyzing an equimolar DNA mixture from those plants into a a single reaction, determining in that mixture the set of alleles present in the mixture. An analysis of cost / benefit, leaded by Dr. M. Humberto Reyes-Valdés resulted in the publication of the paper "*Analysis and optimization of bulk DNA sampling with binary scoring for germplasm characterization*" (Reyes-Valdés et al., 2013). As a conclusion the group determined that the highest information (given a fixed number of PCR reactions) can be gained by sequencing 3 bulks of ten plants from each accession. The first set of results were published in the paper "*An SSR-based approach incorporating a novel algorithm for identification of rare maize genotypes facilitates criteria for landrace conservation in Mexico*" (Hayano-Kanashiro et al., 2017).

Here we are going to analyze a larger dataset than the one that was published, which includes almost all the information obtained in the project.

3.1. **A first look at the maize data.** Let's have a first look at the data in R.

```
1  ------------------------------------> Box  14  <---------------------------------------
2  # Make sure that you are in your CabanaR directory !
3  # IN MY CASE:
4  # > setwd("Documents/Cursos/2019/6_JuneCABANA/CABANA/CabanaR/")
5
6  # Note: If you want to keep the objects previously obtained and clean your
7  # R environment for the analyses that we are going to do, you can
8  # > save.image("DummyExampleStuff.RData") # Save the content
9  # > rm(list=ls()) # Removes all previously defined objects.
10
11 > load("incidence.RData") # This has the "incidence" matrix
12 > class(incidence)
13 [1] "matrix"
14 > nrow(incidence)
15 [1] 1338
16 > ncol(incidence)
17 [1] 333
18 > incidence[1:5, 1:5] # See some rows and columns of the matrix
19       M1.A1 M1.A2 M1.A3 M1.A4 M1.A5
20 CH001     0     0     0     0     0
21 CH002     0     0     0     0     0
22 CH003     0     0     0     0     0
23 CH004     0     0     0     0     2
24 CH005     0     0     0     0     0
25
26 # We have a matrix with results for 1338 accessions (rows) and
27 # 333 combinations Marker / Allele (columns).
28 # Let's keep the names of rows and columns in two vectors
29 > acc.names <- attributes(incidence)$dimnames[[1]] # Names of accessions (columns)
30 > mar.ale.names <- attributes(incidence)$dimnames[[2]] # Names of marker/alleles (rows)
31 > head(acc.names)
32 [1] "CH001" "CH002" "CH003" "CH004" "CH005" "CH006"
33 > head(mar.ale.names)
34 [1] "M1.A1" "M1.A2" "M1.A3" "M1.A4" "M1.A5" "M1.A6"
35
36 # We have 1338*333 = 445,554 data points in this large matrix
37 # A first summary can be given by tabulating the data
38 > inc.table <- table(as.vector(incidence))
39 > inc.table
40
41      0      1      2      3
42 338451  36502  29042  41559
43 > sum(inc.table)
44 [1] 445554
45 > round(100*inc.table/sum(inc.table), 2) # In percentages
46
47     0     1     2     3
```

```
75.96  8.19  6.52  9.33
-------------------------------------------------------------------------------
```

The process to produce the `incidence` matrix needs to be summarized to be able to understand the data, briefly,

(1) For each accession, three bulks of ten plants were germinated and DNA was extracted of each one of these 30 plants.

(2) For each bulk (ten plants) an equimolar mixture of DNA was obtained. This sample was subjected to PCR using, in turn, primers specific to each one of the SSRs; there were 14 different markers employed.

(3) In each bulk and for each marker the PCR products were sequenced, obtaining signals for the presence of different 'alleles', which are distinguished by the number of times that the particular SSR motif is repeated.

As a final result, each cell of the `incidence` matrix has the number of bulks that presented a given marker / allele combination (column) in a particular accession (row); that is why the numbers in the matrix can be 0 –when the marker allele combination did not appeared in the accession or 1, 2 or 3, informing how many of the three bulks presented such marker / allele combination.

It is clear that the distribution of the values 0 to 3 in `incidence` is highly asymmetric; more than 3/4 of the values are zeroes (75.96%), thus we have an sparse matrix. On the other hand, of the cases where the presence of a given marker / allele combination is present, the percentages for 1, 2 or 3 bulks given a combination are 8.19, 6.52 and 9.33%, respectively. This suggest that cases where the allele is 'rare' represented by the cells with '1' (8.19%) are more frequent that cases when we have 2 bulks with the same allele (6.52%), but the higher frequency is for cases in which the three bulk have the allele (9.33%). Unfortunately, as we demonstrated in the supplementary material of Hayano-Kanashiro et al. (2017), information about the **frequencies** of the combinations are confounded, and the maximum likelihood estimation approach fails. This is basically due to the fact that a value of '1' in a cell of the matrix simply means that "at least one" of the $30 \times 2 = 60$ haploid genomes in the sample has the combination observed (but of course, the number of haplotypes could be 2, 3, $\cdots$, 20).

On the other hand, the names of the accessions were conformed by two letters –indicating the origin, followed by three digits; as an example `CH001` means accession 001 from 'CH' (the Chihuahua Mexican state). Table 3 presents the number and percentages of accessions by their origin ('state').

The first analysis is to obtain a table with the number of alleles for each one of the markers. That is performed in `Box 15`.

```
-----------------------------------> Box  15  <-------------------------------------
# The information that we need is already in "mar.ale.names".
# Example of segregation of the names using "strsplit" (only the first two components)
> strsplit(head(mar.ale.names, 2), split=".", fixed=T)
[[1]]
[1] "M1" "A1"

[[2]]
[1] "M1" "A2"

# Make it for the whole vector
> temp <- strsplit(mar.ale.names, split=".", fixed=T)
> length(temp) # Same than the number of columns of incidence
[1] 333

```

TABLE 3. **Number and percentage of accessions per 'state'.**

Id.	State	n	%
GR	Guerrero	227	16.97
MC	Michoacan	226	16.89
PL	Puebla	185	13.83
CH	Chihuahua	116	8.67
CL	Coahuila	86	6.43
GT	Guanajuato	83	6.20
DG	Durango	78	5.83
NT	Nayarit	75	5.61
TL	Tlaxcala	50	3.74
NL	Nuevo Leon	43	3.21
SL	Sinaloa	40	2.99
TS	Tamaulipas	40	2.99
PA†	Palomero	32	2.39
TE‡	Teosinte	23	1.72
TB	Tabasco	20	1.49
SR	Sonora	14	1.05
Total:		1,338	100.00

† An ancient maize landrace.
‡ A sub-specie of maize.

```
# Make a data.frame to order the data
> mar.allele <- data.frame(marker=rep("", 333), allele=rep("", 333), stringsAsFactors=F)
# And fill that data.frame
for(i in 1:333){
mar.allele[i, ] <- c(temp[[i]][1], temp[[i]][2])
}

> head(mar.allele)
  marker allele
1     M1     A1
2     M1     A2
3     M1     A3
4     M1     A4
5     M1     A5
6     M1     A6
> tail(mar.allele)
    marker allele
328    M14    A15
329    M14    A16
330    M14    A17
331    M14    A18
332    M14    A19
333    M14    A20

# Now we can obtain a table with the information that we want
> length(unique(mar.allele$marker)) # How many different markers?
[1] 14
> temp <- unique(mar.allele$marker) # Put them in a vector
> temp
```

```
 [1] "M1"  "M2"  "M3"  "M4"  "M5"  "M6"  "M7"  "M8"  "M9"  "M10" "M11" "M12" "M13" "M14"

> aleles.per.marker <- data.frame(Marker=temp, N.alleles=rep(NA, 14), stringsAsFactors=F)
for(i in 1:14){
aleles.per.marker$N.alleles[i] <- nrow(mar.allele[mar.allele$marker ==
     aleles.per.marker$Marker[i], ])
}

> aleles.per.marker # See our table
   Marker N.alleles
1      M1        24
2      M2        24
3      M3        23
4      M4        26
5      M5        11
6      M6        30
7      M7        41
8      M8        27
9      M9        20
10    M10        22
11    M11        20
12    M12        21
13    M13        24
14    M14        20

> aleles.per.marker[order(aleles.per.marker$N.alleles),] # Orderd by N.alleles
   Marker N.alleles
5      M5        11
9      M9        20
11    M11        20
14    M14        20
12    M12        21
10    M10        22
3      M3        23
1      M1        24
2      M2        24
13    M13        24
4      M4        26
8      M8        27
6      M6        30
7      M7        41
> table(aleles.per.marker$N.alleles) # Number of cases
11 20 21 22 23 24 26 27 30 41
 1  3  1  1  1  3  1  1  1  1
 > summary(aleles.per.marker$N.alleles)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.00   20.25   23.50   23.79   25.50   41.00
> sd(aleles.per.marker$N.alleles)
[1] 6.612093

## As an EXERCISE try to reproduce Table 3 with your data.
--------------------------------------------------------------------------------
```

From the above results we see that the 'informativeness' of the markers is highly variable; the minimum number of alleles is 11, while the maximum is of 41 with a median of 23.5 alleles per marker.

On the other hand we have 1,338 accessions divided into 16 groups, 14 of them are Mexican states, while the other two (PA, and TE) are an ancient maize landrace, 'Palomero' and a accessions of maize sub-species, 'Teosintes', both of the genus *Zea*.

The problem that this dataset presents is the fact that we have a large number of accessions (1,338), and making a dendrogram of such large number of accessions is not useful (try "`plot(hclust(dist(incidence), "average"), lebels=NULL)`" in R; it can take some time to compute). However we have the classification of the accessions in 'states' (see Table 3) and it will be interesting to find if that classification has statistical support. For this we can investigate genetic diversity between and within the *a priori* groups that we have.

3.1.1. *Excercises.*

(1) Try to reproduce Table 3 in your R session (only Id. n and %).

(2) Given that we have many (1,338) accessions, it will be useful to obtain a dendrogram for only one accession from each state (id). Obtain from the incidence matrix a single representative of each state and construct a dendrogram by using Euclidean distance and UPGMA ("average") method. Plot that dendrogram, and keep the result into a PDF file named "`DendOneAccPerState.pdf`". You can use the R function "`pdf()`". Immediately after plotting use function "`dev.off()`" to 'close' the PDF devise.

(3) From the matrix obtained in the previous exercise, containing a single accession per state, obtain a sub-matrix containing information ONLY for the most informative marker, marker M7. Again obtain and plot a dendrogram by using Euclidean distance and UPGMA ("average") method. Plot that dendrogram, and keep the result into a PDF file named "`DendOneAccPerStateM7.pdf`"

(4) Are the dendrograms obtained in exercises (2) and (3) equal or very different in form (topology)? Discuss.

(5) Propose a method to obtain a dendrogram PER STATE which uses the information from all accessions and discuss with your partners.

3.2. **Segregating the distance matrix into sets of distances.** Assume that the original data are presented as a matrix, $\mathbf{O}_{\{n\times v\}}$ where the n rows are 'entities' which have realized measures for v variables (columns). From these data we can obtain a distance matrix, say $\mathbf{D}$, where $\mathbf{D}$ of order $n\times n$ is a symmetric matrix ($d_{ij} = d_{ji}; i \neq j, i$ and j from 1 to n) and zeros in the main diagonal ($d_{ii} = 0; i = 1, 2, \cdots, n$). Taking into account only non-redundant elements, i.e., the triangular superior or inferior of $\mathbf{D}$ we have $n(n-1)/2$ distinct elements in $\mathbf{D}$ which represent all distances between pairs of entities (rows of the original data).

Now, assume that we have k putative clusters, $k < n$, each one of them having $k_i > 0$ individuals, $\sum_i k_i = n$. Then D can be divided into $k+k(k-1)/2$ matrices, the first k of them (first term of the previous addition) will have distances within entities in one of the k clusters (and will be symmetric triangular matrices), while the remaining $k(k-1)/2$ matrices (which are not restricted to be symmetric or triangular) will have distances between two of the putative clusters. Table 4 presents a simple example.

Table 4 presents the minimum representation of a distance matrix, $\mathbf{D}$, for $n = 6$ individuals as a triangular inferior matrix. Rows of the matrix are named $r1, r2, \cdots, r6$ and columns $c1, c2, \cdots, c6$. It is assumed the existing of three putative clusters, $\mathbf{k_1}, \mathbf{k_2}$ and $\mathbf{k_3}$, each one with 3, 2 and 1 of the original 6 entities. Elements of the triangular matrices containing distances for entities within clusters are colored in blue, while elements of the rectangular matrices containing distances for entities between clusters are colored

TABLE 4. Partitioning a matrix **D** of order 6×6 in 6 matrices, 3 of them representing distances within clusters $\mathbf{k_1}$ (with 3 elements), $\mathbf{k_2}$ (with two elements) and $\mathbf{k_3}$ (with one element). Elements belonging to within cluster distances are colored in blue. The other 3 matrices represent distances between clusters $\mathbf{k_1}$&$\mathbf{k_2}$, $\mathbf{k_1}$&$\mathbf{k_3}$ and $\mathbf{k_2}$&$\mathbf{k_3}$ (colored in red, except the last one which unique distance is zero).

		$\mathbf{k_1}$			$\mathbf{k_2}$		$\mathbf{k_3}$
		$c1$	$c2$	$c3$	$c4$	$c5$	$c6$
	$r1$	0					
$\mathbf{k_1}$	$r2$	d_{21}	0				
	$r3$	d_{31}	d_{32}	0			
	$r4$	d_{41}	d_{42}	d_{43}	0		
$\mathbf{k_2}$	$r5$	d_{51}	d_{52}	d_{53}	d_{54}	0	
$\mathbf{k_3}$	$r6$	d_{61}	d_{62}	d_{63}	d_{64}	d_{65}	0

in red. The decomposition shown can be generalized for any number of clusters and is sumarized in Table 5 denoting each array as a sub-matrix $\mathbf{D_{ij}}$.

TABLE 5. Generalization of the segregation of a distance matrix **D** into sub-matrices by a set of r clusters.

	$\mathbf{k_1}$	$\mathbf{k_2}$	$\mathbf{k_3}$	$\cdots$	$\mathbf{k_r}$
$\mathbf{k_1}$	$\mathbf{D_{11}}$	$\mathbf{D_{12}}$	$\mathbf{D_{13}}$	$\cdots$	$\mathbf{D_{1r}}$
$\mathbf{k_2}$	$\mathbf{D_{21}}$	$\mathbf{D_{22}}$	$\mathbf{D_{23}}$	$\cdots$	$\mathbf{D_{2r}}$
$\mathbf{k_3}$	$\mathbf{D_{31}}$	$\mathbf{D_{32}}$	$\mathbf{D_{33}}$	$\cdots$	$\mathbf{D_{3r}}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{k_r}$	$\mathbf{D_{r1}}$	$\mathbf{D_{r2}}$	$\mathbf{D_{r3}}$	$\cdots$	$\mathbf{D_{rr}}$

In Table 5 we assume that each $\mathbf{k_i}$ cluster has k_i entities (rows of the original data), and that $\sum_i k_i = n$ where n is the number of rows and columns of **D**. In that case each one of the $\mathbf{D_{ii}}; i = 1, 2, \cdots r$ matrices (of order $k_i \times k_i$ and colored in blue) will be a triangular inferior matrix with $k_i(k_i - 1)/2$ distinct elements, containing distances within entities in the cluster $\mathbf{k_i}$, while each one of the different $r(r-1)/2$ matrices, $\mathbf{D_{ij}}; i > j, i = 2, \cdots r$ (of order $k_i \times k_j$ colored in red) will have $k_i k_j$ distances between clusters $\mathbf{k_i}$ and $\mathbf{k_j}$. Note that $\mathbf{D_{ij}}; i > j$ is equal to the transpose of $\mathbf{D_{ji}}$, say $\mathbf{D_{ij}} = t(\mathbf{D_{ji}})$, thus we only take into account the set of non redundant distances between entities in different clusters (sub-matrices colored in red).

A simpler alternative to the matrix notation is to represent the distance matrix, **D**, as a data-frame with three columns, say the row, column and distance value as shown in Table 6.

Table 6 shows that this representation avoids the inclusion of the zeroes in the main diagonal of **D** – corresponding to the elements $d_{ii} = 0$, as well as the redundancy given by the fact that $d_{ij} = d_{ji}$, and thus only the $n(n-1)/2$ numeric values existent in the triangular inferior matrix **D** are present as rows of **d**. The convention used to transform **D** into its data frame representation, **d**, is to increase first the row (from the second to the last) and then the column (from the first to the penultimate), and thus avoiding all cases where row and column are the same.

We can convert our incidence matrix to a dataset with the R function '`dist2data()`'[1] Let's begin that analysis as shown in `Box 16`.

```
1 ------------------------------------> Box  16  <--------------------------------------
2 # Obtain a vector to distinguish the accession group, remember
3 > head(acc.names)
```

[1]Function '`dist2data()`' was modified from a version found at 'stackoverflow'.

TABLE 6. Non-redundant data-frame, **d**, which is a different form (representation) of the distance matrix **D** of order $n \times n$. This data frame has $n(n-1)/2$ rows.

Row	Column	Value
r_2	c_1	d_{21}
r_3	c_1	d_{31}
$\cdots$	c_1	$\cdots$
r_n	c_1	d_{n1}
r_2	c_2	d_{21}
r_3	c_2	d_{31}
$\cdots$	c_2	$\cdots$
r_n	c_2	d_{n2}
$\cdots$	$\cdots$	$\cdots$
r_n	c_{n-1}	$d_{n,n-1}$

```
[1] "CH001" "CH002" "CH003" "CH004" "CH005" "CH006"
# Using "substring" we can get the accession group (acc.group)
> substring(head(acc.names), 1, 2)
[1] "CH" "CH" "CH" "CH" "CH" "CH"
> acc.group <- substring(acc.names, 1, 2)
> table(acc.group) # Compare with Table 3.
acc.group
 CH  CL  DG  GR  GT  MC  NL  NT  PA  PL  SL  SR  TB  TE  TL  TS
116  86  78 227  83 226  43  75  32 185  40  14  20  23  50  40

# We are going to need some extra function which are in file "ClustStructFun.txt"
> source("ClustStructFun.txt") # Load such functions
> lsf.str() # List functions in our current environment:
AMA : function (mat)
dist2data : function (inDist)
myColorDendrogram : function (hc, y, main = "", branchlength = 0.7,
    labels = NULL, xlab = NULL, sub = NULL,
    ylab = "", cex.main = NULL, lwd = par("lwd"), cex = 0.2)
permute.seg : function (y = segregate.dis(sim2classes()),
    cont = "all.within", versus = "all.other",
    B = 1000, conf.level = 0.95, alternative = "less",
    print.results = TRUE, onlyPs = FALSE,
    perform.permutations = TRUE)
segregate.dis : function (x = sim2classes(), classes = rep(c("c1", "c2"),
    each = 10), method = "euclidean", sepa = ".")
segregate.plot : function (y = segregate.dis())
segregate.summary : function (y = segregate.dis())
segregate.test : function (y = segregate.dis(), cont = "a.a", versus = "all.other")
segregated2matrix : function (y = segregate.dis(), stat = "mean")
sim2classes : function (c1 = 10, c2 = 10, v = 10, mc1 = 0, sc1 = 1, mc2 = 0, sc2 = 1)
two.states.den : function (s1 = "DG", s2 = "NL", cex = 0.5, main = "")

# Obtain the Euclidean distance from the incidence matrix
> incidence.dist <- dist(incidence, method="euclidean")
> length(incidence.dist)
[1] 894453
> summary(incidence.dist)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.28   16.55   17.92   17.80   19.16   24.82
> incidence.dist.df <- dist2data(incidence.dist)
> class(incidence.dist.df)
[1] "data.frame"
> nrow(incidence.dist.df)
[1] 894453
> head(incidence.dist.df)
    row   col    value
1 CH002 CH001 12.36932
2 CH003 CH001 17.20465
3 CH004 CH001 13.63818
4 CH005 CH001 13.26650
5 CH006 CH001 15.03330
6 CH007 CH001 15.09967
> tail(incidence.dist.df)
         row   col    value
894448 TS074 TS073 13.41641
894449 TS075 TS073 15.32971
894450 TS089 TS073 13.74773
894451 TS075 TS074 13.74773
894452 TS089 TS074 12.92285
894453 TS089 TS075 14.07125

# Now we can add variables for the state, simply by taking the first two
# characters of the names
> incidence.dist.df <- data.frame(incidence.dist.df,
        row.st=substring(incidence.dist.df$row,1,2),
        col.st=substring(incidence.dist.df$col,1,2),
        stringsAsFactors=F)
> head(incidence.dist.df)
    row   col    value row.st col.st
1 CH002 CH001 12.36932     CH     CH
2 CH003 CH001 17.20465     CH     CH
3 CH004 CH001 13.63818     CH     CH
4 CH005 CH001 13.26650     CH     CH
5 CH006 CH001 15.03330     CH     CH
6 CH007 CH001 15.09967     CH     CH

# Now is the distance "between" or "within" groups?
> incidence.dist.df <- data.frame(incidence.dist.df,
        within=incidence.dist.df$row.st == incidence.dist.df$col.st)
> head(incidence.dist.df)
    row   col    value row.st col.st within
1 CH002 CH001 12.36932     CH     CH   TRUE
2 CH003 CH001 17.20465     CH     CH   TRUE
3 CH004 CH001 13.63818     CH     CH   TRUE
4 CH005 CH001 13.26650     CH     CH   TRUE
5 CH006 CH001 15.03330     CH     CH   TRUE
6 CH007 CH001 15.09967     CH     CH   TRUE

> head(incidence.dist.df[!incidence.dist.df$within,])
```

```
      row   col    value row.st col.st within
116 CL007 CH001 18.84144     CL     CH  FALSE
117 CL008 CH001 17.32051     CL     CH  FALSE
118 CL009 CH001 18.33030     CL     CH  FALSE
119 CL010 CH001 18.11077     CL     CH  FALSE
120 CL011 CH001 18.24829     CL     CH  FALSE
121 CL012 CH001 17.54993     CL     CH  FALSE

# Now some statistics for the value by the "within" variable
> tapply(incidence.dist.df$value, incidence.dist.df$within, length)
 FALSE   TRUE
802133  92320
> tapply(incidence.dist.df$value, incidence.dist.df$within, summary)
$`FALSE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.63   16.97   18.14   18.13   19.31   24.82

$`TRUE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.28   13.96   15.03   14.98   16.03   22.61

> tapply(incidence.dist.df$value, incidence.dist.df$within, sd)
   FALSE     TRUE
1.734292 1.569213

> t.test(value ~ within, data=incidence.dist.df)

        Welch Two Sample t-test

data:  value by within
t = 570.95, df = 119830, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.138372 3.159993
sample estimates:
mean in group FALSE  mean in group TRUE
           18.12530            14.97612
> 18.12530/14.97612
[1] 1.21028

> ? wilcox.test # A non-parametric test
> wilcox.test(value ~ within, data=incidence.dist.df)

        Wilcoxon rank sum test with continuity correction

data:  value by within
W = 6.7516e+10, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0

# Boxplot (Figure 7)
> boxplot(value ~ within, data=incidence.dist.df,
        xlab="Is distance within groups?",
```

```
        ylab="Euclidean distance",
        main="Distances in maize groups.",
        sub="(see Box 16).")
----------------------------------------------------------------------------------------
```

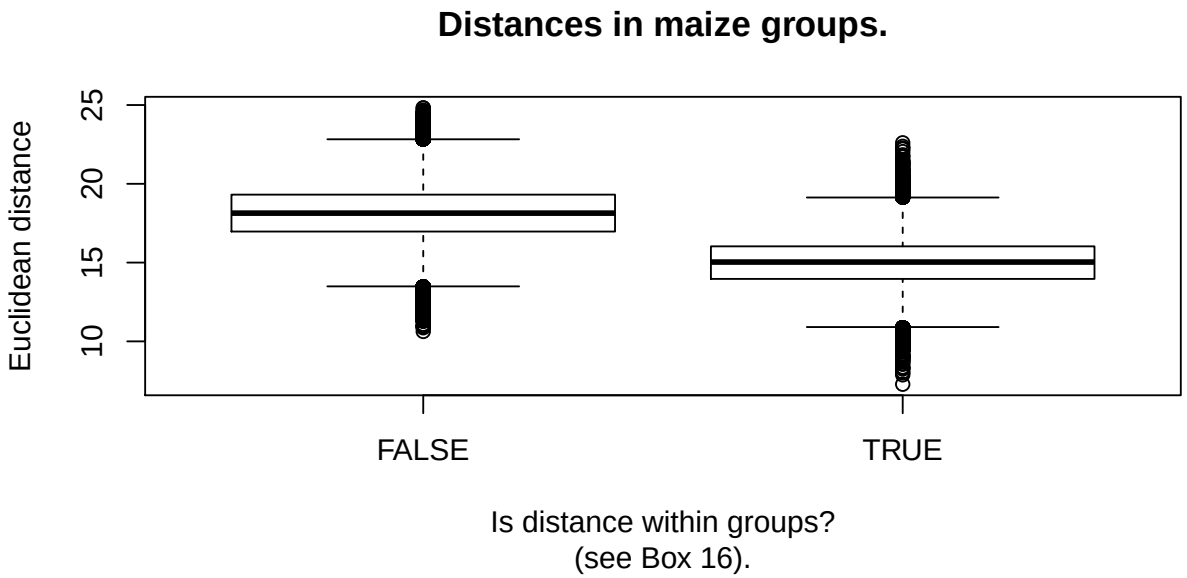


FIGURE 7. R command: '`boxplot(value ~ within, data=incidence.dist.df)`'.

From the analysis above we can conclude that there is a highly significant ($P \approx 0$) difference in the genetic distances between groups (18.12530), and the distance within groups (14.97612). Apart from being statistically significant, the difference appears to be biologically relevant, given that 18.12530/14.97612 = 1.21028, thus we estimate to have around 21% more genetic distance between than within groups.

Accepting at this point that the groups of accessions are 'valid' we can try to plot a dendrogram showing the genetical relations between such groups. For this it is necessary to segregate all information in the `incidence` matrix, classifying it in the assumed groups. My function '`segregate.dis`' will do this.

```
----------------------------------> Box  17  <--------------------------------------
# > incidence.seg # See the function (not help available)

# Obtain the segregation of the data (remember what we have in "acc.group")
> incidence.seg <- segregate.dis(incidence, classes=acc.group)
> class(incidence.seg)
[1] "segregate.dis"
> names(incidence.seg)
[1] "main"          "classes.table" "sepa"
> length(incidence.seg$main)
[1] 136
> class(incidence.seg$main)
[1] "list"
> head(names(incidence.seg$main))
[1] "CH.CH" "CL.CH" "DG.CH" "GR.CH" "GT.CH" "MC.CH"
> class(incidence.seg$classes.table)
[1] "integer"
> length(incidence.seg$classes.table)
```

```
[1] 16
> incidence.seg$classes.table # We have 16 groups
 CH  CL  DG  GR  GT  MC  NL  NT  PA  PL  SL  SR  TB  TE  TL  TS
116  86  78 227  83 226  43  75  32 185  40  14  20  23  50  40

# Thus, there will be 16*15/2 = 120 comparisons "between" groups
# plus 16 within groups; a total of 120+16=136 comparisons.

> incidence.seg.sum <- segregate.summary(incidence.seg)
> nrow(incidence.seg.sum)
[1] 137
> head(incidence.seg.sum)
          n     Min.     Qu.1   Median     Mean     Qu.3     Max.        S
CH.CH  6670 10.00000 14.59452 15.68439 15.68333 16.76305 22.60531 1.678018
CL.CH  9976 13.00000 16.88194 17.74824 17.72651 18.60108 22.04541 1.267946
DG.CH  9048 11.83216 15.49193 16.30951 16.37267 17.23369 22.33831 1.321412
GR.CH 26332 15.45962 18.92089 19.72308 19.72078 20.51828 24.16609 1.190451
GT.CH  9628 14.17745 17.11724 18.00000 17.99146 18.86796 23.10844 1.283969
MC.CH 26216 13.63818 19.72308 20.51828 20.52037 21.35416 24.81935 1.198334
> tail(incidence.seg.sum)
           n      Min.     Qu.1   Median     Mean     Qu.3     Max.        S
TL.TE   1150 14.628739 17.80449 18.68154 18.65926 19.46792 22.64950 1.185084
TS.TE    920 17.204651 19.54482 20.19901 20.20459 20.90454 23.55844 0.990566
TL.TL   1225  9.486833 13.41641 14.66288 14.56799 15.71623 19.46792 1.760078
TS.TL   2000 16.000000 18.35756 19.13113 19.14300 19.92486 22.58318 1.086420
TS.TS    780 11.489125 14.66288 15.62050 15.62830 16.58312 20.24846 1.405454
All   894453  7.280110 16.55295 17.91647 17.80026 19.15724 24.81935 1.967088
# Note: Last row gives a summary of ALL 894453 distances between accessions.

# Now we are going to use another function to convert "incidence.seg" into
# a matrix. See function "segregated2matrix" (no help available)

> maiz.group.dis <- segregated2matrix(incidence.seg)
> round(maiz.group.dis)
   CH CL DG GR GT MC NL NT PA PL SL SR TB TE TL TS
CH 16 18 16 20 18 21 16 16 20 19 17 17 19 21 19 18
CL 18 15 17 18 17 19 16 17 19 18 17 17 17 19 18 17
DG 16 17 15 19 17 20 15 16 19 18 16 16 18 20 18 16
GR 20 18 19 15 17 18 18 19 17 17 19 19 17 19 17 19
GT 18 17 17 17 14 18 16 17 17 16 18 17 16 18 16 18
MC 21 19 20 18 18 15 19 20 18 18 20 20 19 20 19 21
NL 16 16 15 18 16 19 14 15 19 17 16 16 17 19 17 16
NT 16 17 16 19 17 20 15 15 19 18 16 16 17 20 18 17
PA 20 19 19 17 17 18 19 19 14 17 20 19 18 18 17 20
PL 19 18 18 17 16 18 17 18 17 15 19 18 17 19 16 19
SL 17 17 16 19 18 20 16 16 20 19 16 17 18 20 19 17
SR 17 17 16 19 17 20 16 16 19 18 17 15 18 20 18 17
TB 19 17 18 17 16 19 17 17 18 17 18 18 14 19 17 18
TE 21 19 20 19 18 20 19 20 18 19 20 20 19 16 19 20
TL 19 18 18 17 16 19 17 18 17 16 19 18 17 19 15 19
TS 18 17 16 19 18 21 16 17 20 19 17 17 18 20 19 16
```

```
# Note: Elements in the main diagonal are distances within group
# while other elements are distances between groups
# (of course, this matrix is symmetric)

# Finally we can construct a dendrogram to show the relation between groups.
> maiz.group.t.dis <- as.dist(maiz.group.dis)
> hclu.maiz.group <- hclust(maiz.group.t.dis, method="average")
# The plot presented as  Figure 8
> plot(hclu.maiz.group, main="Cluster of groups of maize accessions\n
        (using mean distance between groups)")
```

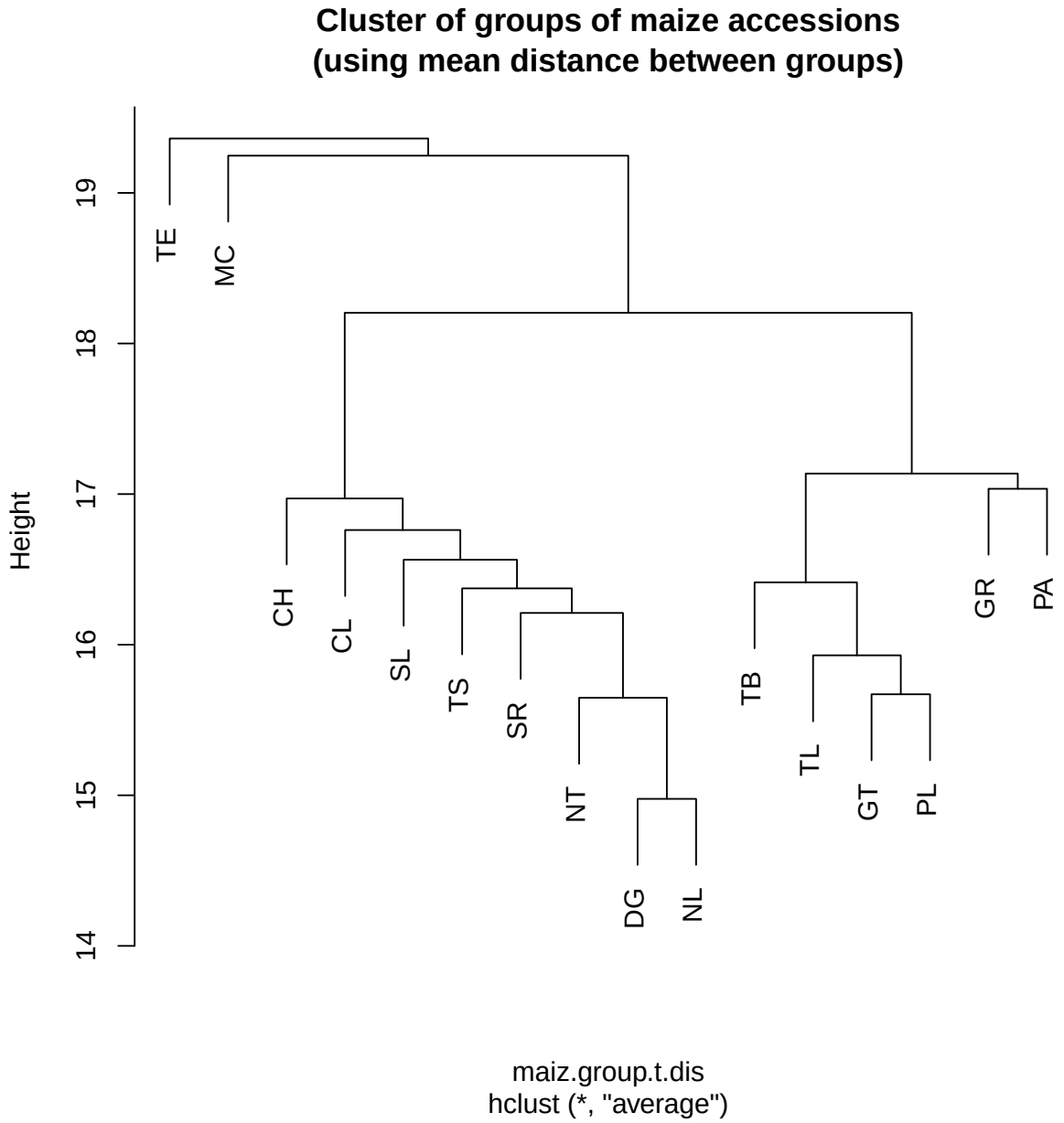


FIGURE 8. R command: '`plot(hclu.maiz.group)`', See '`Box 17`'.

In Figure 8 we observe the dendrogram produced by the `average` method for the **groups** of accessions. As expected from the origin of the accessions, the teosintes (TE group) are in the root of the tree. This suggest that our analysis is appropriate (correct?) for this data.

3.2.1. *Excercises.*

(1) By examining the diagonal of the matrix distance "`maiz.group.dis`" order the 'states' from the less to the more 'compact', taking has criterion the man average distance within the group. You could find useful the R expression "`diag(maiz.group.dis)`".

(2) You must find that the less compact, i.e., the more 'diverse' of the groups is TE (teosintes). Discuss the biological implications of this fact.

It is important to take into account that we are *assuming* that the groups (mainly by states) have a genetic base, i.e., we are supposing that the groups have less genetic diversity within the accessions than between them. While this is in general true (we have seen that genetic distances between groups are larger than within groups), we cannot be sure that that is the case for all pairs of groups. For example, the two groups that are clustered in Figure 8 at a lower distance are DG and NL. The question is if those two groups are in fact segregated or not. `Box 18` shows how to construct a dendrogram for the accessions in those two groups.

```
1  ----------------------------------> Box  18  <--------------------------------------
2  # Let's obtain a matrix with only the "DG" or "NL" accessions
3
4  # How to select which rows of incidence correspond to "DG" or "NL"?
5  # Example
6  > head(substring(attributes(incidence)$dimnames[[1]],1,2))
7  [1] "CH" "CH" "CH" "CH" "CH" "CH"
8  > head(incidence[substring(attributes(incidence)$dimnames[[1]],1,2)=="DG", 1:5])
9        M1.A1 M1.A2 M1.A3 M1.A4 M1.A5
10 DG001     0     0     0     0     1
11 DG002     0     0     0     0     0
12 DG003     0     0     0     0     0
13 DG004     1     0     1     0     0
14 DG005     0     0     0     0     0
15 DG006     0     0     0     0     0
16
17 > inc.DG.NL <- incidence[(substring(attributes(incidence)$dimnames[[1]],1,2)=="DG")|
18         (substring(attributes(incidence)$dimnames[[1]],1,2)=="NL"),]
19 > nrow(inc.DG.NL)
20 [1] 121
21 > head(inc.DG.NL[,1:5])
22       M1.A1 M1.A2 M1.A3 M1.A4 M1.A5
23 DG001     0     0     0     0     1
24 DG002     0     0     0     0     0
25 DG003     0     0     0     0     0
26 DG004     1     0     1     0     0
27 DG005     0     0     0     0     0
28 DG006     0     0     0     0     0
29 > tail(inc.DG.NL[,1:5])
30       M1.A1 M1.A2 M1.A3 M1.A4 M1.A5
31 NL071     0     0     0     0     0
32 NL077     0     0     0     0     1
33 NL085     0     0     0     0     0
```

```
NL087     0     0     0     0     0
NL088     0     0     0     0     0
NL089     0     0     0     0     0

# Let's obtain labels for each accession
> groupDG.NL <- substring(attributes(inc.DG.NL)$dimnames[[1]],1,2)
> DG.NL.dis <- dist(inc.DG.NL) # Euclidean distance (default)
> col.DG.NL <- groupDG.NL
> col.DG.NL[col.DG.NL=="DG"] <- "red" # Assign red to DG
> col.DG.NL[col.DG.NL=="NL"] <- "blue" # Assign blue to NL
# Now use "myColorDendrogram" (not help available)
# Presented as Figure 9
> myColorDendrogram(hclust(DG.NL.dis, "average"), y=col.DG.NL, branchlength=1.6)
> title(main="Dendrogram of accessions from DG and NL")
> legend("topright", bty="n", legend=c("DG", "NL"), lty=1, col=c("red", "blue"))

# Statistics for distances within and between DG and NL
> round(incidence.seg.sum[attributes(incidence.seg.sum)$dimnames[[1]]=="DG.DG",])
     n   Min.   Qu.1 Median   Mean   Qu.3   Max.      S
  3003     10     14     15     15     16     20      1
> round(incidence.seg.sum[attributes(incidence.seg.sum)$dimnames[[1]]=="NL.NL",])
     n   Min.   Qu.1 Median   Mean   Qu.3   Max.      S
   903     10     13     14     14     15     18      1
> round(incidence.seg.sum[attributes(incidence.seg.sum)$dimnames[[1]]=="NL.DG",])
     n   Min.   Qu.1 Median   Mean   Qu.3   Max.      S
  3354     11     14     15     15     16     20      1
-------------------------------------------------------------------------------
```

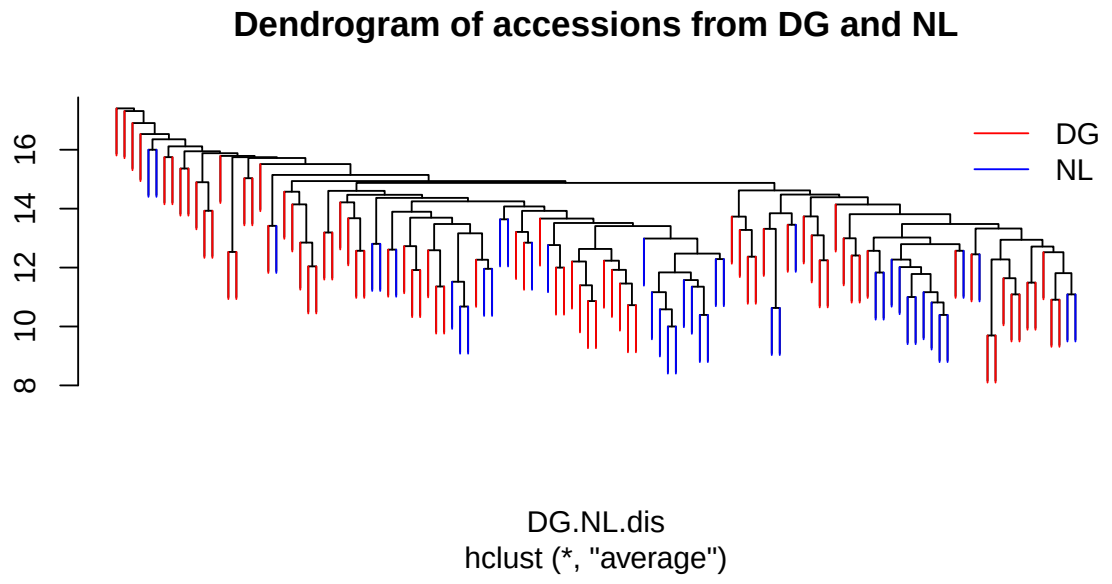


FIGURE 9. R command:
'`myColorDendrogram(hclust(DG.NL.dis, "average"), y=col.DG.NL)`'.

In the dendrogram of Figure 9 we can see that there is NO evidence at all that the accessions from NL and DG form two segregated groups. Also, by looking at the statistics within and between these to groups

(las rows of `Box 18`) we see no evidence that the distances between DG and NL could be larger than the distances within individuals from these two groups. How can we evaluate is a given *a priory* structure has statistical bases?

3.3. **Assessing the structure of a dendrogram.** The problem that we have is to evaluate if a given structure (groups formed *a priory*) is supported by the data at hand. To solve this problem, I program a set of functions that you have available in your R session (they are in file "ClustStructFun.txt", and were loaded with the `source` function in `Box 16`). The idea is to perform a permutation test for the mean distances of two 'contrasts'. It is important to use a non-parametric test because a) data in a distance matrix are correlated and b) we do not the true distribution of such distances. Let's apply the procedure to the case of DG and NL (`Box 19` below)

```
1  ------------------------------------> Box  19  <--------------------------------------
2  # TESTING DG vs NL
3  > seg.DG.NL <- segregate.dis(inc.DG.NL, classes=groupDG.NL)
4  > segregate.summary(seg.DG.NL)
5           n     Min.     Qu.1   Median     Mean     Qu.3     Max.        S
6  DG.DG 3003  9.69536 14.24781 15.23155 15.23270 16.18641 20.24846 1.438714
7  NL.DG 3354 10.90871 14.03567 14.89966 14.97618 15.90597 19.72308 1.337495
8  NL.NL  903 10.00000 13.11488 14.10674 14.06138 15.06652 18.43909 1.426965
9  All   7260  9.69536 14.00000 14.93318 14.96850 15.96872 20.24846 1.437571
10 > segregate.plot(seg.DG.NL)
11
12 # Trying to apply "segregate.test" blindly...
13 > segregate.test(seg.DG.NL)
14 [1] "Available contrasts for that object are:"
15 [1] "DG.DG" "NL.DG" "NL.NL"
16
17 # The contrast of interest:
18 > segregate.test(seg.DG.NL, cont="NL.DG")
19 [1] "[NL.DG] versus [DG.DG, NL.NL]"
20
21         Welch Two Sample t-test
22
23 data:  g1 and g2
24 t = 0.4257, df = 7253.3, p-value = 0.6703
25 alternative hypothesis: true difference in means is not equal to 0
26 95 percent confidence interval:
27  -0.05144004  0.07997894
28 sample estimates:
29 mean of x mean of y
30  14.97618  14.96191
31
32
33         Wilcoxon rank sum test with continuity correction
34
35 data:  g1 and g2
36 W = 6527100, p-value = 0.7939
37 alternative hypothesis: true location shift is not equal to 0
38 ------------------------------------------------------------------------------------
```

First note that the null hypothesis that we are interested in testing is if the mean distances between groups (DG and NL) are equal to the means of distances within the groups (within DG and within

NL). If we cannot reject that hypothesis, i.e., if the estimated value of P is 'large' then we will conclude that there is no real segregation between the two groups. From the results in `Box 19` we see that the hypothesis of equality of means of the between groups (`[NL.DG]`) *versus* the within groups (`[DG.DG, NL.NL]`) cannot be reasonably rejected; the P values are $P = 0.6703$ for the Student's t-tes –which is of dubious application here, and $P = 0.7939$ for the Wilcoxon test.

Let's demonstrate the method with a different case; for example, assume that the researcher is interested in knowing if the teosintes (group TE) is well differentiated from the ancient group of the Palomero maize accessions (group PA). `Box 20` presents the calculations.

```
1  ------------------------------------> Box  20  <--------------------------------------
2  # Analyzing TE (teosintes) versus PA (palomero)
3  > inc.TE.PA <- incidence[(substring(attributes(incidence)$dimnames[[1]],1,2)=="TE")|
4         (substring(attributes(incidence)$dimnames[[1]],1,2)=="PA"),]
5  > nrow(inc.TE.PA)
6  [1] 55
7  > groupTE.PA <- substring(attributes(inc.TE.PA)$dimnames[[1]],1,2)
8  > table(groupTE.PA)
9  groupTE.PA
10 PA TE
11 32 23
12 > TE.PA.dis <- dist(inc.TE.PA) # Euclidean distance (default)
13 > col.TE.PA <- groupTE.PA
14 > col.TE.PA[col.TE.PA=="TE"] <- "red" # Assign red to TE
15 > col.TE.PA[col.TE.PA=="PA"] <- "blue" # Assign red to PA
16 > myColorDendrogram(hclust(TE.PA.dis, "average"), y=col.TE.PA, branchlength=1.6)
17 > title(main="Dendrogram of accessions from TE and PL")
18 > legend("topright", bty="n", legend=c("TE", "PL"), lty=1, col=c("red", "blue"))
19
20 > seg.TE.PA <- segregate.dis(inc.TE.PA, classes=groupTE.PA)
21 > segregate.summary(seg.TE.PA)
22          n     Min.     Qu.1   Median     Mean     Qu.3     Max.        S
23 PA.PA  496 11.00000 13.34166 14.42221 14.41496 15.43534 18.43909 1.431672
24 TE.PA  736 15.19868 17.29162 18.08314 18.05146 18.89444 21.86321 1.113587
25 TE.TE  253 10.95445 15.45962 16.61325 16.44214 17.66352 20.17424 1.639000
26 All   1485 10.95445 15.06652 16.91153 16.56267 18.19341 21.86321 2.097024
27
28 > segregate.test(seg.TE.PA, cont="TE.PA")
29 [1] "[TE.PA] versus [PA.PA, TE.TE]"
30
31         Welch Two Sample t-test
32
33 data:  g1 and g2
34 t = 38.321, df = 1257.7, p-value < 2.2e-16
35 alternative hypothesis: true difference in means is not equal to 0
36 95 percent confidence interval:
37  2.800639 3.102868
38 sample estimates:
39 mean of x mean of y
40  18.05146  15.09971
41
42
43         Wilcoxon rank sum test with continuity correction
```

```
44
45 data:  g1 and g2
46 W = 503810, p-value < 2.2e-16
47 alternative hypothesis: true location shift is not equal to 0
48
49 # For Figure 10
50 col.TE.PA <- groupTE.PA
51 col.TE.PA[col.TE.PA=="TE"] <- "red" # Assign red to TE
52 col.TE.PA[col.TE.PA=="PA"] <- "blue" # Assign blue to PA
53
54 myColorDendrogram(hclust(TE.PA.dis, "average"), y=col.TE.PA, branchlength=1.6)
55 title(main="Dendrogram of accessions from TE and PA")
56 legend("topright", bty="n", legend=c("TE", "PA"), lty=1, col=c("red", "blue"))
57 ------------------------------------------------------------------------------
```

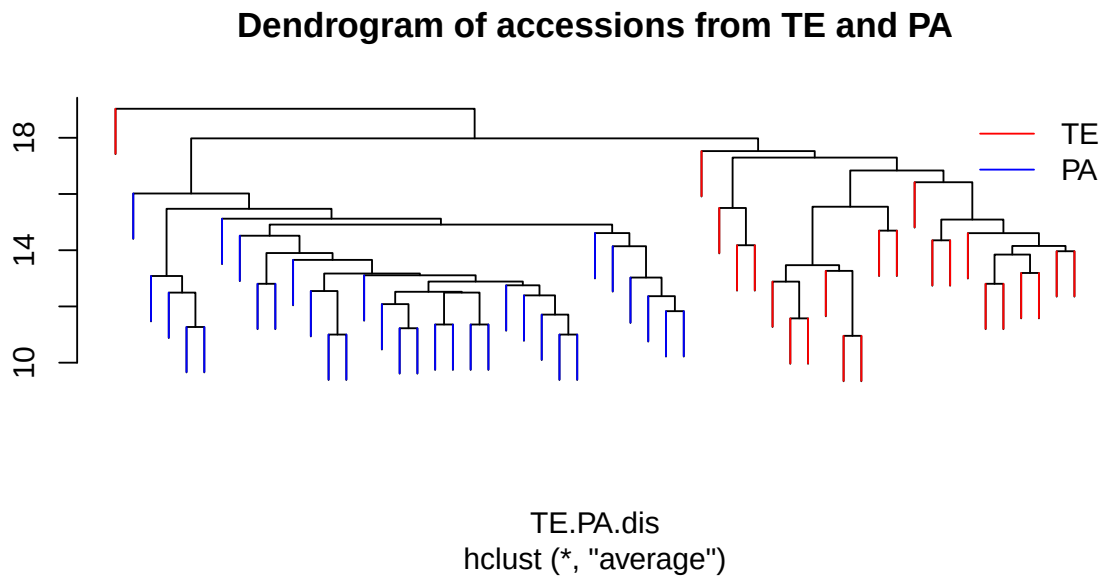


FIGURE 10. R command:
'`myColorDendrogram(hclust(TE.PA.dis, "average"), y=col.TE.PA)`'.

3.3.1. *Excercise.* Program a function that takes as input the incidence matrix and the names of two of the sets of accessions (for example `PA` and `TE`) and that automatically performs the analysis to see if such groups are statistically different. After you try, you could see the text file "`adhoc.txt`" in your directory. Using your function or the one in the file try testing groups that you think are interesting.

The dendrogram in Figure 10 shows that the PA and TE accessions form two well segregated groups, except by one TE accession that appears in the root of the two groups. Also, the results shown in `Box 20` show that the means between accessions in the groups PA and TE are significantly ($P \approx 0$) larger than the means within these groups; the actual mean values are ≈ 18 and ≈ 15, respectively, confirming the fact observed in the dendrogram that the accession from those two groups are significantly segregated.

It is clear that when we have a bifurcating tree (dendrogram) for n individuals –groups of accessions in Figure 8, we will have $n-1$ 'nodes' (clusters; without taking into account the one that groups all individuals). Also, by having n groups we can perform $(n \times (n-1))/2$ pair comparison between groups. In our case we have $n = 16$, thus we have $(16 \times (16-1))/2 = 120$ pair comparisons between groups (for example 'TE' *versus* 'PA' in `Box 20`, etc.).

3.3.2. *Excercises.*

(1) An important biological fact in maize research is that teosintes are subspecies of the *Zea* genus, and thus will be genetically segregated from maize accessions. In `Box 20` we showed how to evaluate the segregation between the group of teosintes (TE) from the ancient Palomero maize accessions (PA). Select other group of accessions different to TE or PA from Table 3 and repeat the analyses in `Box 20` for the sets TE *versus* your selected set and PA *versus* your selected set. Discuss your conclusions with the group.

Before finishing the analysis, let's see the results of comparing groups PL and GT in `Box 21` (only results presented).

```
1  ------------------------------------> Box  21  <--------------------------------------
2  # (only results but not calculations are presented)
3
4  Analysis of groups PL and GT
5  Total of accessions = 268
6
7  Number of accessions per:
8  group
9   GT  PL
10  83 185
11
12 Summary of distances:
13           n      Min.     Qu.1   Median     Mean     Qu.3     Max.        S
14 GT.GT  3403  7.280110 13.22876 14.38749 14.30373 15.39480 20.27313 1.606096
15 PL.GT 15355 11.090537 14.79865 15.68439 15.67076 16.52271 20.17424 1.270214
16 PL.PL 17020  7.874008 13.67479 14.69694 14.58038 15.62050 19.23538 1.501575
17 All   35778  7.280110 14.10674 15.13275 15.02203 16.06238 20.27313 1.527339
18 [1] "[PL.GT] versus [GT.GT, PL.PL]"
19
20         Welch Two Sample t-test
21
22 data:  g1 and g2
23 t = 76.858, df = 35394, p-value < 2.2e-16
24 alternative hypothesis: true difference in means is not equal to 0
25 95 percent confidence interval:
26  1.107488 1.165452
27 sample estimates:
28 mean of x mean of y
29  15.67076  14.53429
30
31
32         Wilcoxon rank sum test with continuity correction
33
34 data:  g1 and g2
35 W = 222900000, p-value < 2.2e-16
36 alternative hypothesis: true location shift is not equal to 0
37 ------------------------------------------------------------------------------------
```

The statistical analysis comparing groups PL and GT shown in `Box 21` is 'highly significant' ($P < 2.2 \times 10^{-16} \approx 0$ in both tests); mean distances between and within groups of accessions are 15.67 and 14.53, respectively. Clearly, the dendrogram of the individual accessions from these two origins, presented in Figure 11 **do not present** two main groups; in fact, two GT accessions (top right corner) are 'well

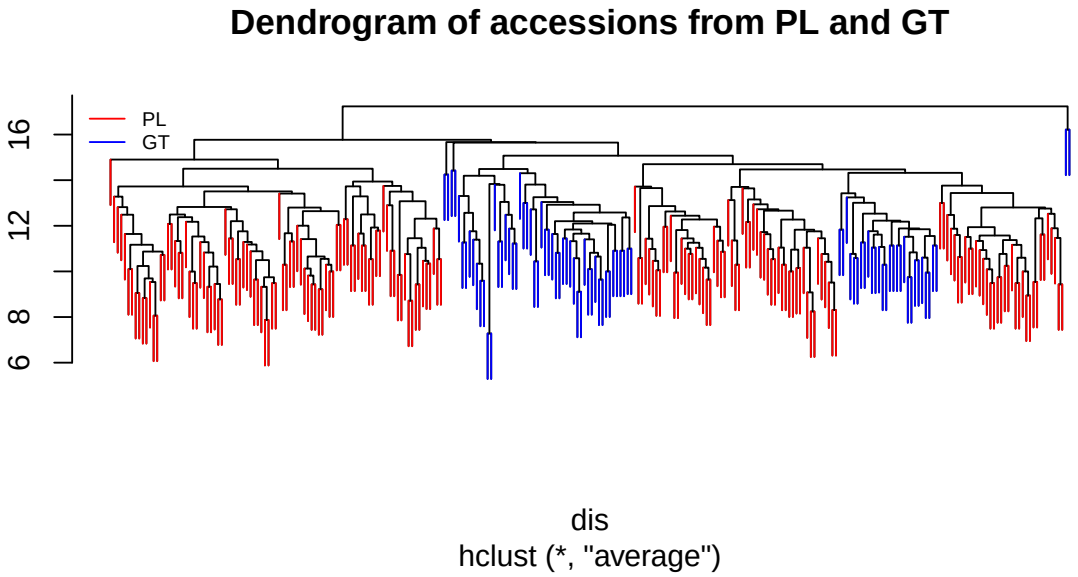


FIGURE 11. (calculations not detailed)

segregated' from all other accessions, which group in around 6 groups at height ≈ 15. However, accessions from PL and GT do not tend to be scattered in many small groups, as was the case in the accessions from DG and NL (Figure 9), but present an important tendency to group together, i.e., PL accessions are group 'near' other PL and the same happens with GT. Even when we cannot rule out the existence of other compact genetic groups in this collection (PL $\cup$ GT) it is statistically valid to say that they form segregate collections.

The relation between groups of accessions presented in Figure 8 can be tested by performing different 'contrasts' between putative sets. For example, we have seen that there is no reason to consider accessions in DG and NL as different groups (they can be merged into a single group). The final result of the analysis by groups of accessions is presented in Figure 12.

Comparing Figure 12 with Figure 8, we see that the only 'new' group is 'NL-DG', which results of the fusion of NL with DG, all other groups are significantly segregated, and thus after many additional tests (data not published yet) the final relation between groups of accessions is the one shown in Figure 12.

An important topic that I do not present in this notes is the inference of association between particular markers and phenotypic characters, as for example association between particular combinations of marker alleles with the height above sea level at which accessions were sampled, that we present and discuss in (Hayano-Kanashiro et al., 2017).

Other important problem that I will not discuss here is the selection of a set of accessions that could represent the whole diversity (all marker / allele combinations) present in the whole collection of accessions. The algorithm that we presented in (Hayano-Kanashiro et al., 2017) is program in the `AMA` function, that you have available in your R session. `Box 22` briefly presents the results of the application of the AMA algorithm to the full collection of data that we have.

```
1  ------------------------------------> Box  22  <--------------------------------------
2  # The AMA algorithm
3  > inc.AMA <- AMA(incidence)
4
5  A set of 56 accessions contains all 333 marker/alleles from the collection of 1338 accessions.
```

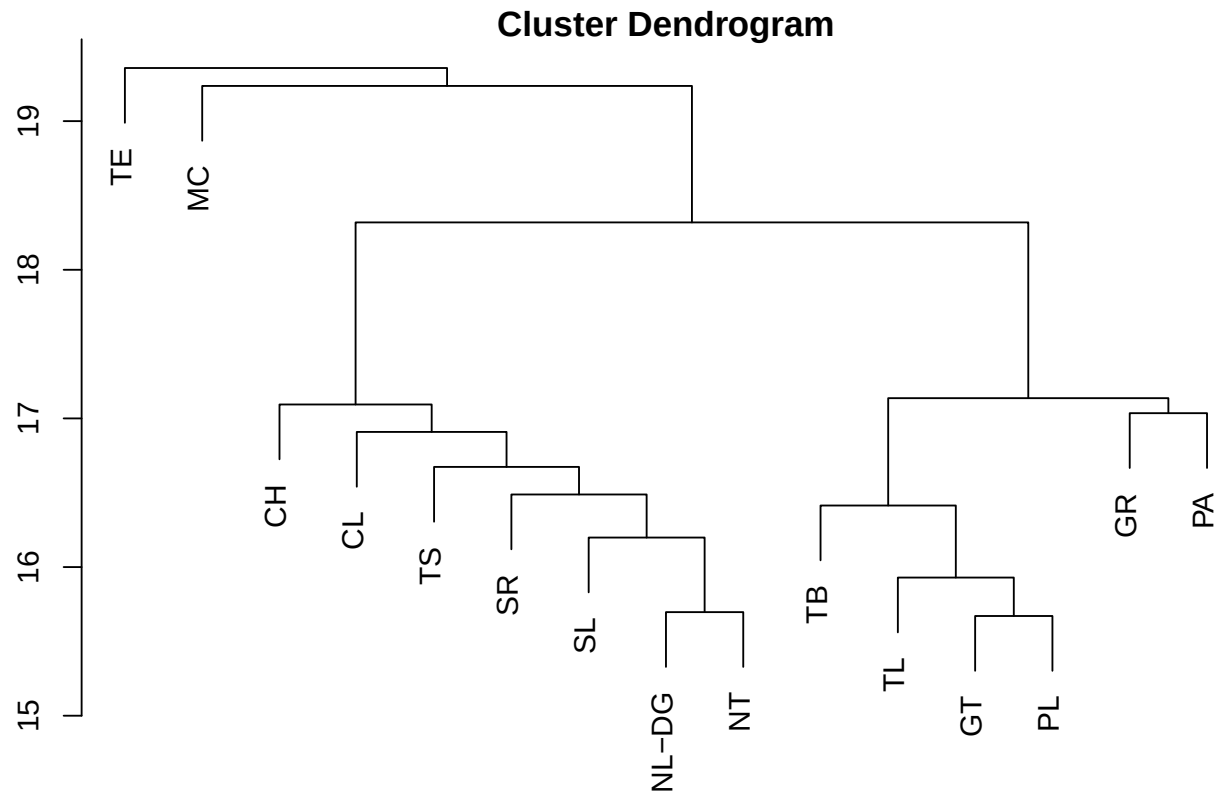


FIGURE 12. Final result of the analysis of relation between accession groups.

```
The set of selected accessions represents the 4.19% of the original.

> class(inc.AMA)
[1] "list"
> names(inc.AMA)
 [1] "Set.names"  "Set.nums"   "Richness"   "Ties"       "MinMat"     "rho"
 [7] "nam.unique" "acc.unique" "on.acc"     "comment"    "call"

> inc.AMA$Set.names
 [1] "MC140" "MC113" "MC169" "GR521" "TL019" "PL107" "MC118" "TE063" "GR170" "PA007"
[11] "TL042" "TE017" "PA035" "PL102" "MC083" "MC028" "MC310" "TL044" "MC224" "GR050"
[21] "CH069" "TS022" "PL131" "GR160" "PL015" "TE004" "TE005" "MC385" "PA027" "PA031"
[31] "PL075" "CH124" "GR166" "MC080" "TE039" "CL088" "TS035" "GR518" "PA022" "PL099"
[41] "GT070" "MC121" "MC059" "CH015" "MC112" "CH010" "MC070" "MC103" "CH007" "GR298"
[51] "GR191" "PA047" "PL135" "MC378" "TL032" "PL047"
> inc.AMA$Richness
 [1] 245 259 266 272 276 280 283 286 289 292 295 298 300 302 304 306 308 310 312 314 316
[22] 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333

> table(substring(inc.AMA$Set.names,1,2))

CH CL GR GT MC PA PL TE TL TS
 5  1  8  1 16  6  8  5  4  2

# If we have time we will discuss the results.
```

```
31
32 ### NOTE: It will be good idea at this point to save all the stuff that you have
33 # in your R environment into a file. You could use, for example
34 # save.image("MaizeStuff.RData")
35 ------------------------------------------------------------------------------
```

3.4. **Conclusions of the methods and analysis of the maize dataset.** From the analyses we can conclude that the set of SSR markers are highly informative and allow the segregation of all maize accessions, presenting a group structure concordant with their pre-defined origin. Even when we did not had time to do a comprehensive analysis, it is clear that the use of bulks of plants to sample crop diversity is a cost effective way to study crop genetic diversity.

The study of genetic distances between and within pre-defined groups gives a statistical tool to judge the significance and biological relevance of crop diversity.

4. Molecular phenotypes with RNA-Seq

A phenotype is any 'observable' characteristic of a living organism, and the complex way in which the interaction of the genome information with the environment determines the phenotype is, in my view, the main problem in Biology. While genomics give us the opportunity to study the primal source of genetic diversity, transcriptomics technologies give us the chance to observe the expression of all the genes in a genome. Gene expression is a 'molecular phenotype' and methods as RNA-Seq allow the estimation of these character in a genomic wide perspective.

4.1. **The chili dataset.** We are currently advancing in a project to obtain gene expression and metabolomic profiles during the development of chili pepper fruits in 8 accessions (you can see the proposed methods here: "chili project methods", while (Martínez-López et al., 2014) presents some antecedents of this research). We used 8 accessions (see Table 7) and from each one of these we obtained RNA-Seq data at 7 stages in fruit development, at anthesis, i.e., mature and open flower; time 0 Days After Anthesis (DAA), and then at 10, 20, 30, $\cdots$, 60 DAA, when the fruit is fully mature and beginning senescence. Thus, the experiment has a factorial nature: accessions (8) $\times$ developing time (7 points) and we have measures from many genes.

Table 7. Accessions in the chili project.

Key	Class	Name
CM	D	Criollo de Morelos (CM334)
CW	D	California Wonder
ST	D	Serrano Tampiqueño
ZU	D	Zunla 1
CO	W	Piquín Coahuila (Humberto)
QU	W	Piquín Querétaro
CQ	C	F_1 Criollo de Morelos (CM) ♀ $\times$ Piquín Querétaro (QU) ♂
QC	C	F_1 Piquín Querétaro (QU) ♀ $\times$ Criollo de Morelos (CM) ♂
Type: D - Domesticated, W - Wild, C - Cross		

The aim of the analysis that we are going to perform here is to understand the genome wide phenotypic relation between the accessions and the way in which time of development affects such relations. For this we will use the normalized mean of 'Fragments Per Kilobase of transcript per Million of mapped reads'. For each gene, the mean is the result of averaging two biological replicates. In `Box 22` we will have a first look at the data.

```
----------------------------------> Box  22  <-------------------------------------
# We begin with an empty environment and change the working directory
# to "CabanaR".
# In my case:
# > getwd()
# [1] "/Users/OMV/Documents/Cursos/2019/6_JuneCABANA/CABANA/CabanaR"

# Load the file "ChiliStuff.RData":
> load("ChiliStuff.RData")
> chili.stuff # A vector with the names of the objects that we loaded
[1] "chili.stuff" "ac.data"     "ac.type"     "chili"

# Let's see what we have
> ac.data
  ac                                                                  name         type
1 CM                                          Criollo de Morelos (CM334) Domesticated
2 CO                                          Piquin Coahuila (Humberto)         Wild
3 CQ F1 Criollo de Morelos (CM) hembra x  Piquin Queretaro (QU) macho        Cross
4 CW                                                   California Wonder Domesticated
5 QC  F1 Piquin Queretaro (QU) hembra x Criollo de Morelos (CM) macho        Cross
6 QU                                                    Piquin Queretaro         Wild
7 ST                                                  Serrano Tampiqueno Domesticated
8 ZU                                                           Zunla 1 Domesticated
> ac.type
  ac type  key
1 CM    D CM.D
2 CO    W CO.W
3 CQ    C CQ.C
4 CW    D CW.D
5 QC    C QC.C
6 QU    W QU.W
7 ST    D ST.D
8 ZU    D ZU.D

# Now, the "chili" object is large:
> class(chili)
[1] "data.frame"
> nrow(chili)
[1] 231030
> head(chili)
  ac id        zT0       zT10       zT20       zT30       zT40       zT50       zT60
1 CM  3 -0.2681764 -0.4507998 -0.4507012 -0.4203951 -0.3464629 -0.3259264  2.2624617
2 CM 12 -0.7945581  0.7927462 -0.7945581 -0.7945581  1.1704922 -0.7945581  1.2149941
3 CM 15 -0.2101871 -0.5714159 -0.4298572 -0.5714159 -0.5714159  0.1721211  2.1821710
4 CM 17 -0.5188853 -0.5188853 -0.5188853 -0.5188853 -0.3738449  0.2803612  2.1690250
5 CM 19 -0.7889519  2.0926774  0.4182137 -0.4911143 -0.3900891 -0.6160367 -0.2246991
6 CM 22  2.2292760 -0.5380721 -0.3590978 -0.1331966 -0.1227653 -0.5380721 -0.5380721

# id - Numerical identifier of the gene
# How many different genes do we have?
> length(unique(chili$id))
[1] 32095
```

```
> length(unique(chili$ac)) # Number of accessions
[1] 8
> unique(chili$ac) # Accessions
[1] "CM" "ST" "CO" "CW" "QU" "CQ" "QC" "ZU"

# Important: Not all genes are expressed in all accessions!, see
> tapply(chili$id, chili$ac, length)
   CM    CO    CQ    CW    QC    QU    ST    ZU
28778 28787 29191 28638 29246 29027 28506 28857

> chili[chili$id==3,] # All data for the gene with id==3
       ac id        zT0        zT10       zT20         zT30        zT40         zT50
1      CM  3 -0.2681764 -0.45079979 -0.4507012 -0.420395085 -0.34646287 -0.325926385
28779  ST  3 -0.2768107 -0.84994285 -1.2311372  0.210730306  0.87316960 -0.379595853
57285  CO  3 -0.2250544 -0.48291729 -0.4791442 -0.594440315 -0.17966364 -0.279892398
86073  CW  3  0.9014902 -1.17121167  1.1443854  0.052527644  0.67971280 -1.415874130
114710 QU  3  1.8725442 -0.01177807  0.3072857 -0.587270217 -1.12443432 -0.813686073
143737 CQ  3  1.5150472 -1.02133629  0.7853821  0.685726405 -0.67940774 -1.021336293
172928 QC  3 -0.8040784 -0.80407838  0.5961468  0.008001352 -0.11338702  1.921473966
202174 ZU  3  1.9620788  0.22112700 -0.2608614 -0.863379534  0.07045355  0.003234409
              zT60
1       2.2624617
28779   1.6535867
57285   2.2411123
86073  -0.1910302
114710  0.3573387
143737 -0.2640754
172928 -0.8040784
202174 -1.1326528

# It will be more convenient to add the type (D, W or C) to the "ac"
for(i in 1:8){
chili$ac[chili$ac==ac.type$ac[i]] <- ac.type$key[i]
}

# Make a little experiment
# (obtain dendrograms based in expression of only one gene)
> temp <- chili[chili$id==3,]
> temp$ac
[1] "CM.D" "ST.D" "CO.W" "CW.D" "QU.W" "CQ.C" "QC.C" "ZU.D"
> temp2 <- temp$ac
> temp <- as.matrix(temp[,3:9])
> attributes(temp)$dimnames[[1]] <- temp2
> round(dist(temp),2)
     CM.D ST.D CO.W CW.D QU.W CQ.C QC.C
ST.D 1.74
CO.W 0.25 1.68
CW.D 3.59 3.42 3.57
QU.W 3.14 3.77 3.16 2.70
CQ.C 3.64 3.75 3.69 1.71 1.90
QC.C 4.02 4.00 4.01 3.94 4.25 3.90
ZU.D 4.18 4.10 4.11 3.02 2.19 2.76 3.74
```

```
105
106 # Plot the dendrograms with different methods
107 > plot(hclust(dist(temp), method="complete")) # Figure 13 A
108 > plot(hclust(dist(temp), method="average")) # Figure 13 B
109 > plot(hclust(dist(temp), method="single")) # Figure 13 C
110 > plot(hclust(dist(temp), method="ward.D")) # Figure 13 D
111 ------------------------------------------------------------------------------
```

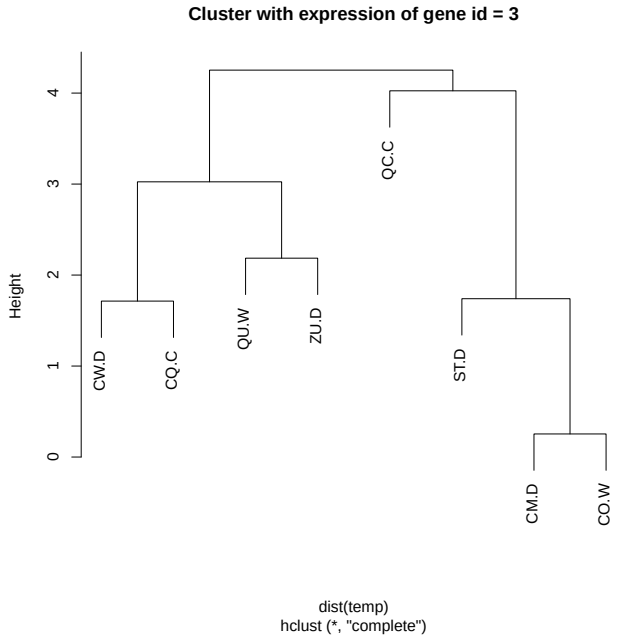
For a better understanding of the data, let's make a plot of the gene expression of gene with id=3.

```
1  ------------------------------------> Box  23  <-------------------------------------
2  # Commands to plot gene expression for gene with id=3
3  # (data are in the "temp" object)
4  plot(seq(0,60,10), temp[1,], col=rainbow(8)[1], type="l",
5  ylim=c(min(as.vector(temp)), max(as.vector(temp))),
6  lwd=2, xlab="Time (DAA)",
7  ylab="Normalized gene expression",
8  main="Gene expression of gene with id=3")
9  for(i in 2:8){
10 points(seq(0,60,10), temp[i,], type="l", col=rainbow(8)[i], lwd=2, lty=i)
11 }
12 legend(25,2.5, bty="n", legend=attributes(temp)$dimnames[[1]], lwd=2,
13         lty=c(1:8), col=rainbow(8))
14 # Presented as Figure 14
15 ------------------------------------------------------------------------------
```

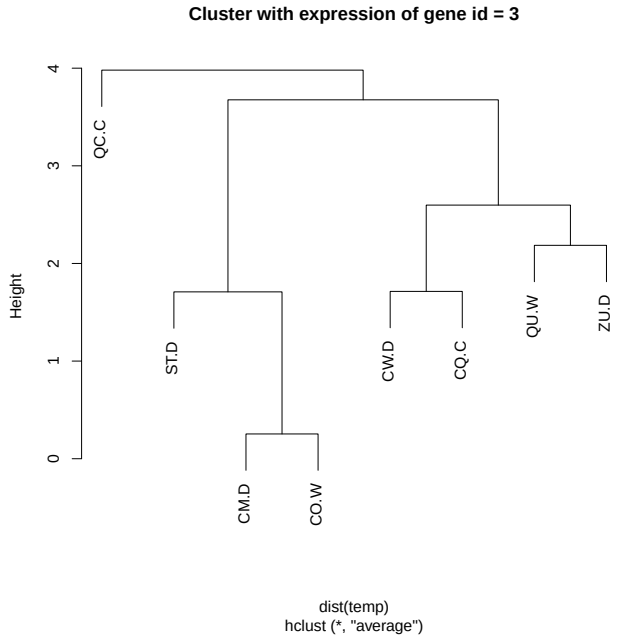
Can you see from figures 13 and 14 if there is a clustering method which better reflects the diversity in the expression pattern of the gene for different accessions? –that is a difficult problem, because it implies subjective criteria. In my personal opinion the average method (UPGMA) appears to give a reasonable result.

Note that we have gene expression data for three aspects: accession (8), time (7) and gene (many; more than 25,000). The data are normalized with reference to time, thus for a specific gene (id) and accession (ac), the vector of gene expression trough time has a mean of 0 and standard deviation of 1; i.e., it gives information only about the change in gene expression trough time. Evidently, there is no way that we could interpret dendrograms for every one of the different genes; we need in some sense to 'summarize' the information trying to loose few relevant information. Before proceeding we must solve the fact that not all genes were expressed in all the 8 accessions. `Box 28` shows how to do that.

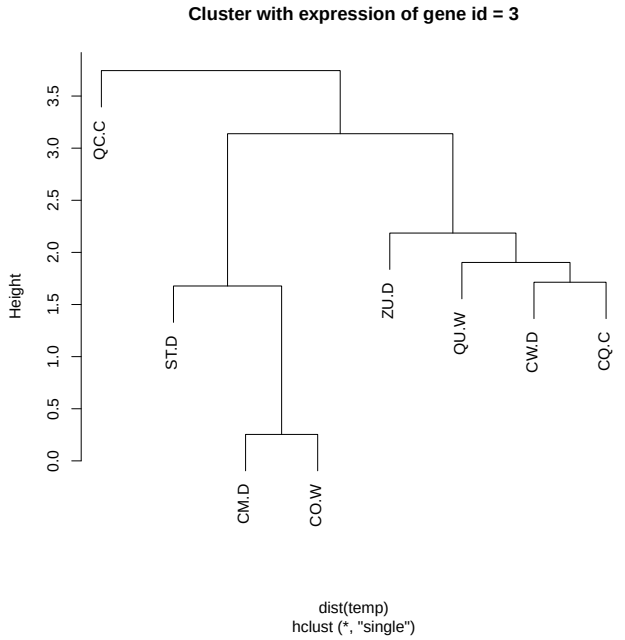
```
1  ------------------------------------> Box  24  <-------------------------------------
2  # Finding the set of genes expressed in all accessions
3  > ac.type$key
4  [1] "CM.D" "CO.W" "CQ.C" "CW.D" "QC.C" "QU.W" "ST.D" "ZU.D"
5  # Let's defines the sets of genes expressed in each accession
6  # using assign()
7  for(i in 1:8){
8  assign(paste("in", ac.type$key[i], sep="."), chili$id[chili$ac==ac.type$key[i]])
9  }
10 > ls(patt="in") # The objects that were created by assign
11 [1] "in.CM.D" "in.CO.W" "in.CQ.C" "in.CW.D" "in.QC.C" "in.QU.W" "in.ST.D" "in.ZU.D"
12 > length(in.CM.D) # Number of ids (expressed genes) in CM.D
13 [1] 28778
14
15 # We need the intersection of ALL 8 sets; let's do it by parts
```



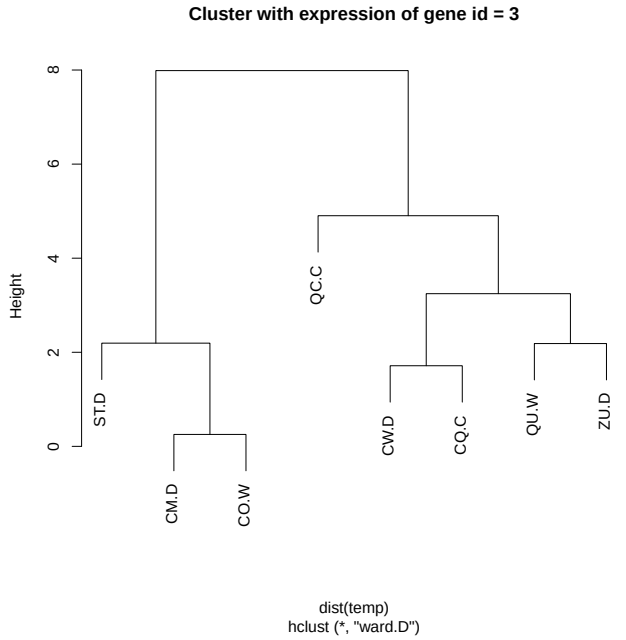
(A) method complete



(B) method average



(C) method single



(D) method ward.D

FIGURE 13. Dendrograms obtained from the distances of a single gene (gene id=3) employing different methods.

```
> in.all <- intersect(in.CM.D, in.CO.W) # The first two
> in.all <- intersect(in.all, in.CQ.C) # The third... (and so on)
> in.all <- intersect(in.all, in.CW.D)
> in.all <- intersect(in.all, in.QC.C)
> in.all <- intersect(in.all, in.QU.W)
> in.all <- intersect(in.all, in.ST.D)
```

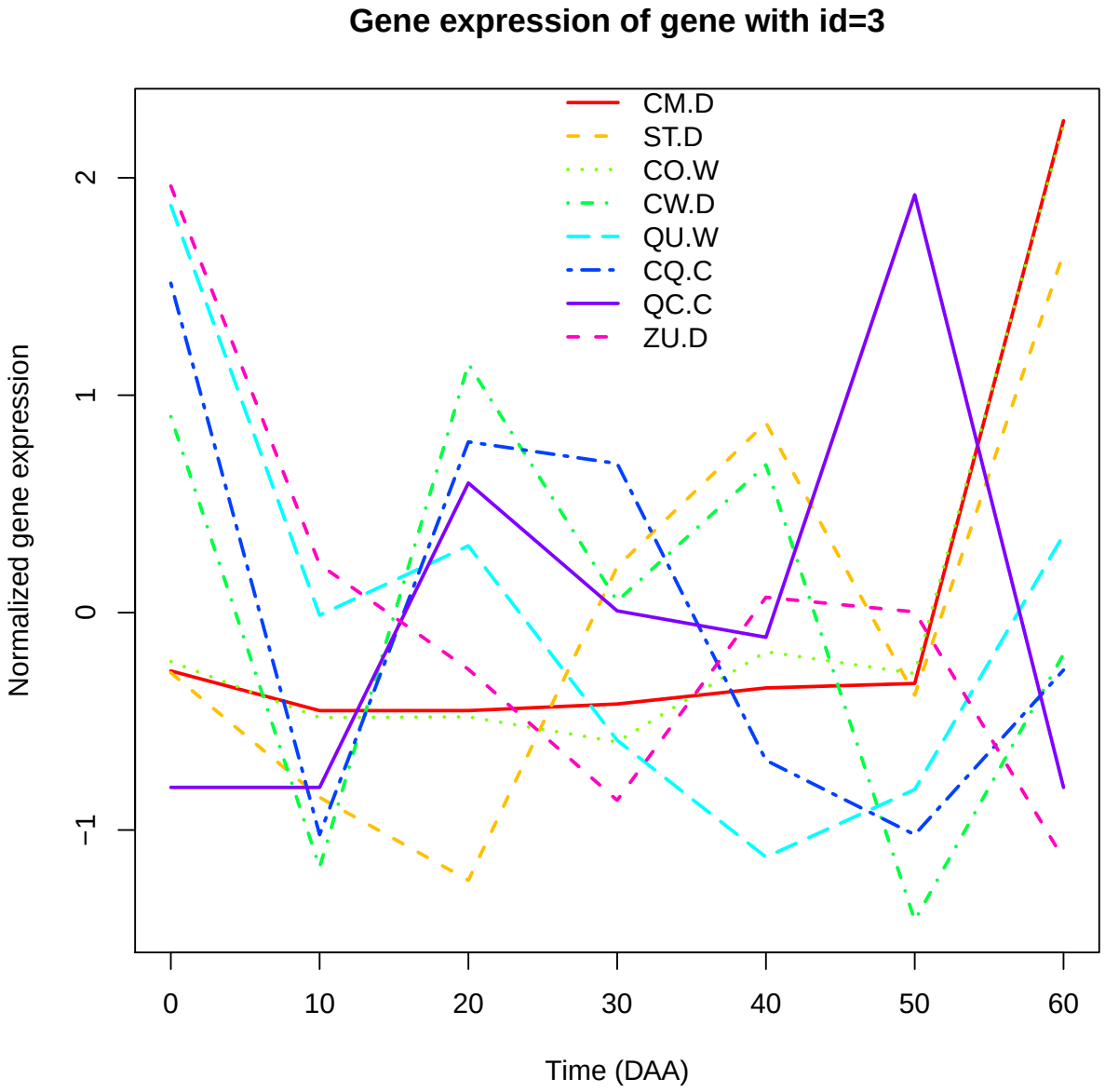


FIGURE 14. Gene expression per accession (an example).

```
> in.all <- intersect(in.all, in.ZU.D)
> length(in.all) # Number of genes expressed in all 8 accessions
[1] 25645
> head(in.all)
[1]  3 12 15 19 22 26
> tail(in.all)
[1] 35878 35879 35881 35882 35883 35884
# Now we can define the set of genes that are not expressed in all accessions
> not.in.all <- setdiff(unique(chili$id), in.all)
> length(not.in.all) # Number of genes not expressed in some accessions
[1] 6450
> head(not.in.all)
```

```
[1]  17  23  38 127 130 131

# Now, define the logical variable "in.all" in "chili"
> chili <- data.frame(chili, in.all=rep(TRUE, nrow(chili)), stringsAsFactors=F)
# And update to FALSE for all cases in "not.in.all"
for(i in 1:length(not.in.all)){
chili$in.all[chili$id == not.in.all[i]] <- FALSE
}
> table(chili$in.all) # Number of cases

 FALSE   TRUE
 25870 205160
> table(chili$in.all, chili$ac) # Number of cases

         CM.D  CO.W  CQ.C  CW.D  QC.C  QU.W  ST.D  ZU.D
  FALSE  3133  3142  3546  2993  3601  3382  2861  3212
  TRUE  25645 25645 25645 25645 25645 25645 25645 25645
```

4.1.1. *Excercises.* In `Box 22` we saw how to construct a dendrogram for the 8 accessions using the information of a single gene. Here I want you to obtain dendrograms from data of two genes randomly chosen from the set of genes expressed in all accessions. To select such genes you could use

```
my.genes <- sample(unique(chili$id[chili$in.all]), size=2)
> my.genes # Note: YOUR results will almost surely be different to these!
[1] 34088 23077
```

(1) Using the two selected genes, construct and plot the corresponding dendrograms using Euclidean distance and the UPGMA (`"average"`) method.

(2) How many groups (nodes or clusters) these two dendrograms share?

(3) Which is your conclusion of this exercise?

From the calculations above we now know that there is a total of 25,645 genes which are expressed in all the 8 accessions. To summarize the information of all those genes we could use a measure of 'central tendency' as the arithmetic mean (the average) or the median. Let's obtain the dendrograms using measures of central tendency for the set of all expressed genes (`Box 25`).

```
-----------------------------------> Box  25  <-------------------------------------
# Considering all expressed genes per accession
# Note that there is interesting variation on times...
> round(apply(chili[chili$in.all, 3:9], 2, mean), 2) # For all accessions
  zT0  zT10  zT20  zT30  zT40  zT50  zT60
 0.15  0.19  0.14  0.11 -0.10 -0.34 -0.14
> round(apply(chili[(chili$in.all)&(chili$ac=="CM.D"), 3:9], 2, mean),2) # Only one
  zT0  zT10  zT20  zT30  zT40  zT50  zT60
-0.01  0.29  0.17  0.01 -0.01 -0.38 -0.07
> ac.type$key[1]
[1] "CM.D"

# Let's put the means per accession in a matrix for each time
> chili.all.mean <- matrix(NA, nrow=8, ncol=7, dimnames=list(ac.type$key, names(chili)[3:9]))
# Now fill that matrix
for(i in 1:8){
```

```
chili.all.mean[i, ] <- apply(chili[(chili$in.all)&(chili$ac==ac.type$key[i]), 3:9], 2, mean)
}

> round(chili.all.mean,2) # To see our results
       zT0 zT10  zT20 zT30  zT40  zT50  zT60
CM.D -0.01 0.29  0.17 0.01 -0.01 -0.38 -0.07
CO.W  0.11 0.19  0.15 0.12 -0.23 -0.33 -0.01
CQ.C  0.28 0.27  0.17 0.18 -0.38 -0.54  0.01
CW.D  0.28 0.05  0.12 0.04  0.03 -0.30 -0.22
QC.C  0.21 0.19 -0.01 0.05 -0.05 -0.38 -0.01
QU.W -0.22 0.08  0.16 0.10  0.20 -0.19 -0.14
ST.D  0.18 0.17  0.16 0.30 -0.11 -0.29 -0.40
ZU.D  0.34 0.25  0.23 0.09 -0.29 -0.29 -0.32

# Now we will do the same for the median
> chili.all.median <- matrix(NA, nrow=8, ncol=7, dimnames=list(ac.type$key, names(chili)[3:9]))
# Now fill that matrix
for(i in 1:8){
chili.all.median[i, ] <- apply(chili[(chili$in.all)&(chili$ac==ac.type$key[i]), 3:9], 2, median)
}

> round(chili.all.median,2) # To see our results
       zT0  zT10  zT20  zT30  zT40  zT50  zT60
CM.D -0.38  0.02  0.01 -0.15 -0.19 -0.53 -0.43
CO.W -0.21 -0.06 -0.01 -0.08 -0.38 -0.48 -0.38
CQ.C  0.04  0.07  0.09  0.05 -0.48 -0.67 -0.38
CW.D  0.01 -0.29 -0.04 -0.13 -0.18 -0.54 -0.48
QC.C -0.09 -0.07 -0.22 -0.20 -0.25 -0.52 -0.38
QU.W -0.64 -0.09  0.09 -0.04  0.10 -0.38 -0.44
ST.D -0.16  0.06  0.02  0.15 -0.32 -0.51 -0.62
ZU.D  0.09  0.09  0.20 -0.03 -0.43 -0.48 -0.64

# Let's plot the means (Figure 15)
plot(seq(0,60,10), chili.all.mean[1,], col=rainbow(8)[1], type="l",
ylim=c(min(as.vector(chili.all.mean)), max(as.vector(chili.all.mean))),
lwd=2, xlab="Time (DAA)",
ylab="Mean of normalized gene expression",
main="Average of gene expression of 25,645 genes\nper time and accession")
for(i in 2:8){
points(seq(0,60,10), chili.all.mean[i,], type="l", col=rainbow(8)[i], lwd=2, lty=i)
}
legend("bottomleft", bty="n", legend=attributes(chili.all.mean)$dimnames[[1]],
        lwd=2, lty=c(1:8), col=rainbow(8))

# And obtain the dendrogram (Figure 16).
> library(pvclust) # To evaluate "robustness"
> ? pvclust # Remember the use
# Without robustness measures
> plot(hclust(dist(chili.all.mean), "average"))

system.time(
# Because ic can take some time
```

```
chili.all.mean.pv <- pvclust(t(chili.all.mean), method.hclust="average",
        method.dist="euclidean", nboot=1000)
)
Bootstrap (r = 0.43)... Done.
Bootstrap (r = 0.57)... Done.
Bootstrap (r = 0.57)... Done.
Bootstrap (r = 0.71)... Done.
Bootstrap (r = 0.86)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.14)... Done.
Bootstrap (r = 1.29)... Done.
Bootstrap (r = 1.29)... Done.
   user  system elapsed
  7.222   0.101   7.317
Warning message:
In a$p[] <- c(1, bp[r == 1]) :
  number of items to replace is not a multiple of replacement length

> chili.all.mean.pv

Cluster method: average
Distance      : euclidean

Estimates on edges:

     au    bp se.au se.bp      v     c  pchi
1 0.812 0.302 0.020 0.005 -0.182 0.702 0.002
2 0.948 0.334 0.008 0.005 -0.596 1.026 0.521
3 0.863 0.395 0.016 0.005 -0.413 0.679 0.354
4 0.995 0.058 0.003 0.003 -0.492 2.061 0.173
5 0.834 0.129 0.024 0.004  0.080 1.049 0.000
6 0.954 0.533 0.007 0.006 -0.882 0.800 0.078
7 1.000 1.000 0.000 0.000  0.000 0.000 0.000
# Figure 16
> plot(chili.all.mean.pv)

# With medians
chili.all.median.pv <- pvclust(t(chili.all.median), method.hclust="average",
        method.dist="euclidean", nboot=1000)
Bootstrap (r = 0.43)... Done.
Bootstrap (r = 0.57)... Done.
Bootstrap (r = 0.57)... Done.
Bootstrap (r = 0.71)... Done.
Bootstrap (r = 0.86)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.14)... Done.
Bootstrap (r = 1.29)... Done.
Bootstrap (r = 1.29)... Done.
Warning message:
In a$p[] <- c(1, bp[r == 1]) :
```

```
  number of items to replace is not a multiple of replacement length
# Figure 17
plot(seq(0,60,10), chili.all.median[1,], col=rainbow(8)[1], type="l",
ylim=c(min(as.vector(chili.all.median)), max(as.vector(chili.all.median))),
lwd=2, xlab="Time (DAA)",
ylab="Median of normalized gene expression",
main="Median of gene expression of 25,645 genes\nper time and accession")
for(i in 2:8){
points(seq(0,60,10), chili.all.median[i,], type="l", col=rainbow(8)[i], lwd=2, lty=i)
}
legend("topright", bty="n", legend=attributes(chili.all.median)$dimnames[[1]], lwd=2,
        lty=c(1:8), col=rainbow(8), cex=0.75)
# Figure 18
plot(chili.all.median.pv)
-------------------------------------------------------------------------------
```

The interpretation of the results above must take into account the fact that in summarizing the (already normalized) gene expression of 25,645 genes we are loosing valuable information. For example, some genes can have a highly similar expression pattern in all accessions, while the expression pattern of others could be 'accession specific'. Also, it is important to consider that the mean (arithmetic average) is sensitive to extreme values, while the median is more robust because it is not alter by extreme values (outliers). The interpretation of these results will be discussed during the exposition.

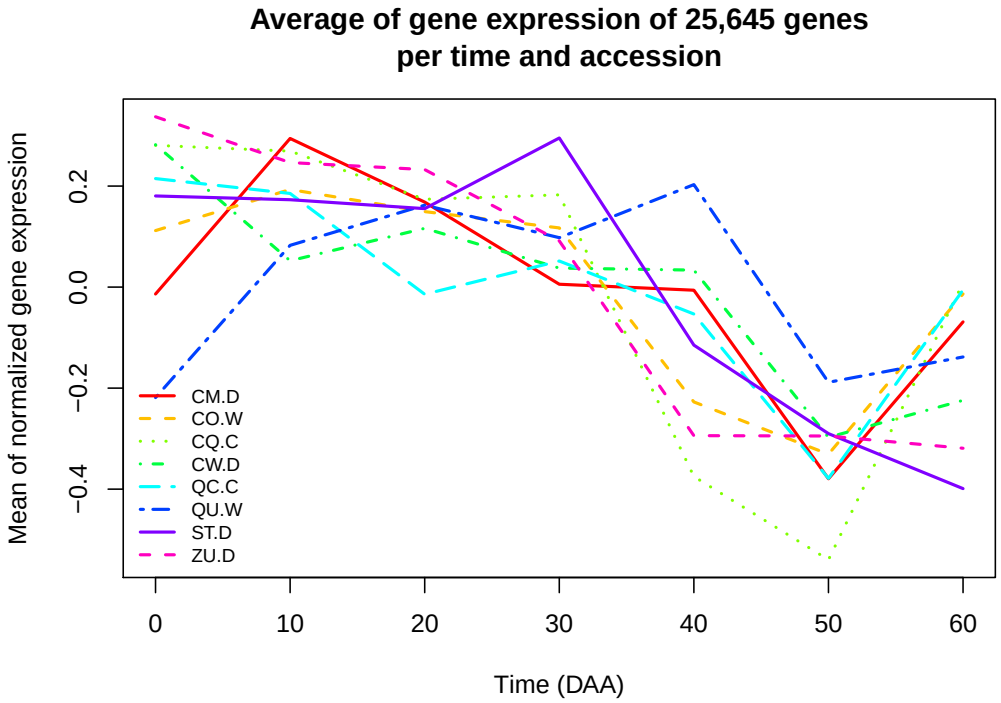


FIGURE 15

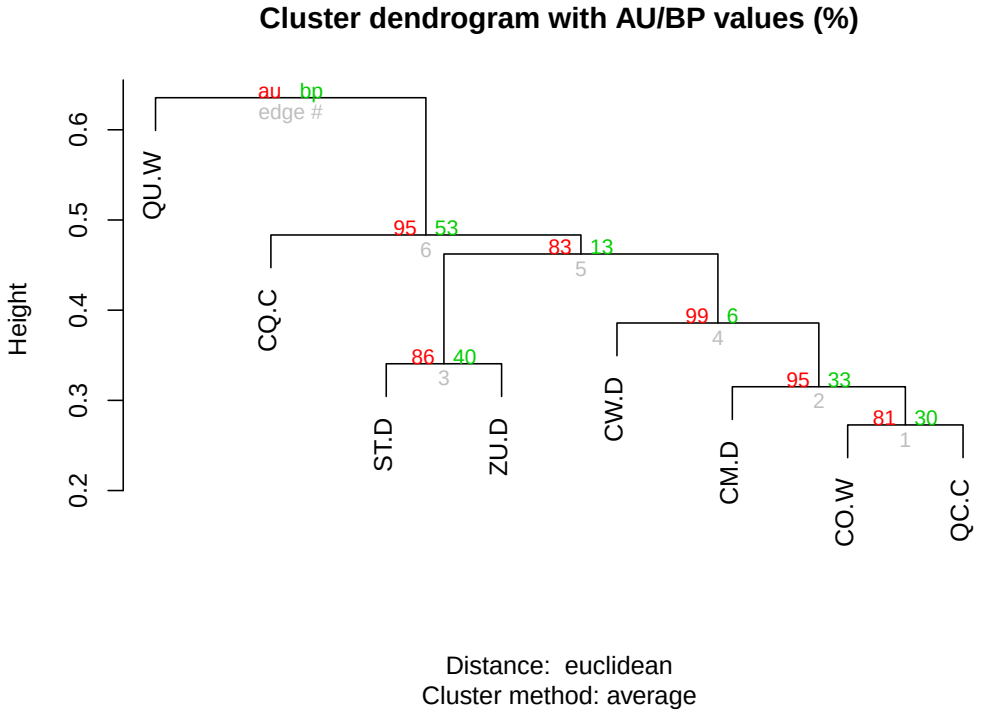


FIGURE 16. Dendrogram using the mean of gene expression.

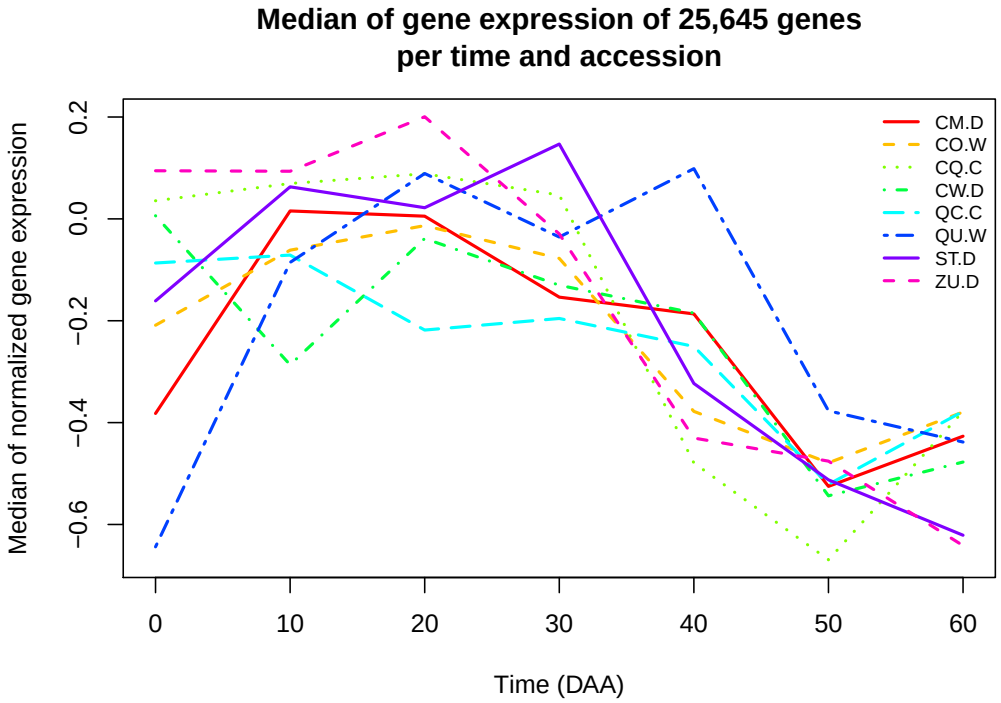


FIGURE 17

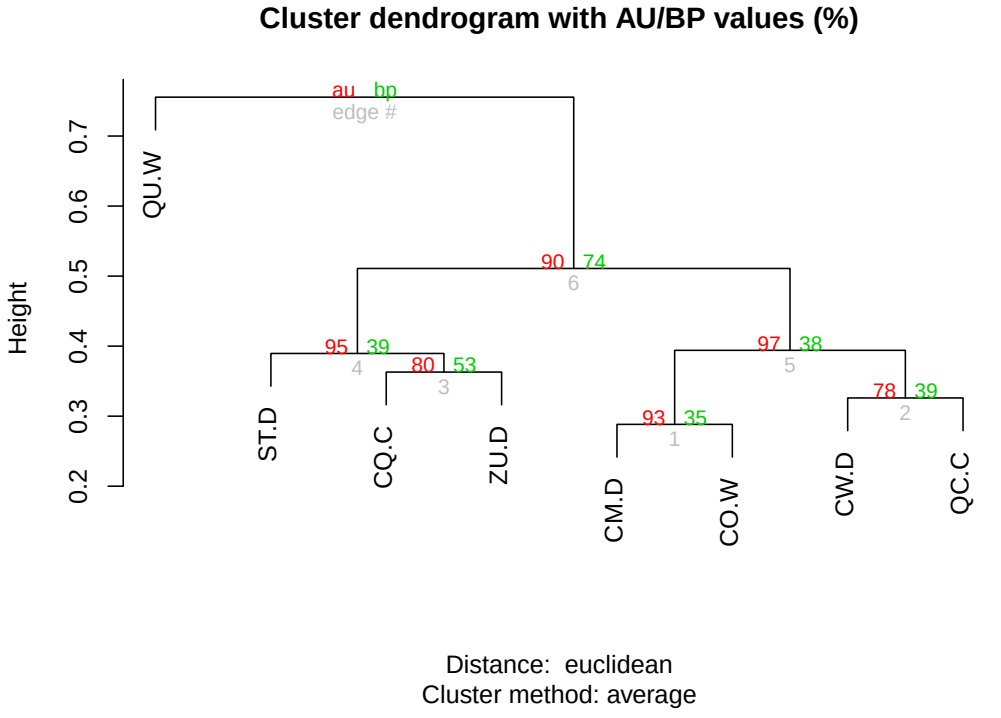


FIGURE 18. Dendrogram using the median of gene expression.

4.1.2. *Excercises.* In `Box 25` we constructed dendrograms using the mean and median of all expressed genes (figures 16 and 18, respectively). In this exercise you must construct dendrograms using the

minimum and maximum of the gene expression (instead of the mean and median) and compare the results.

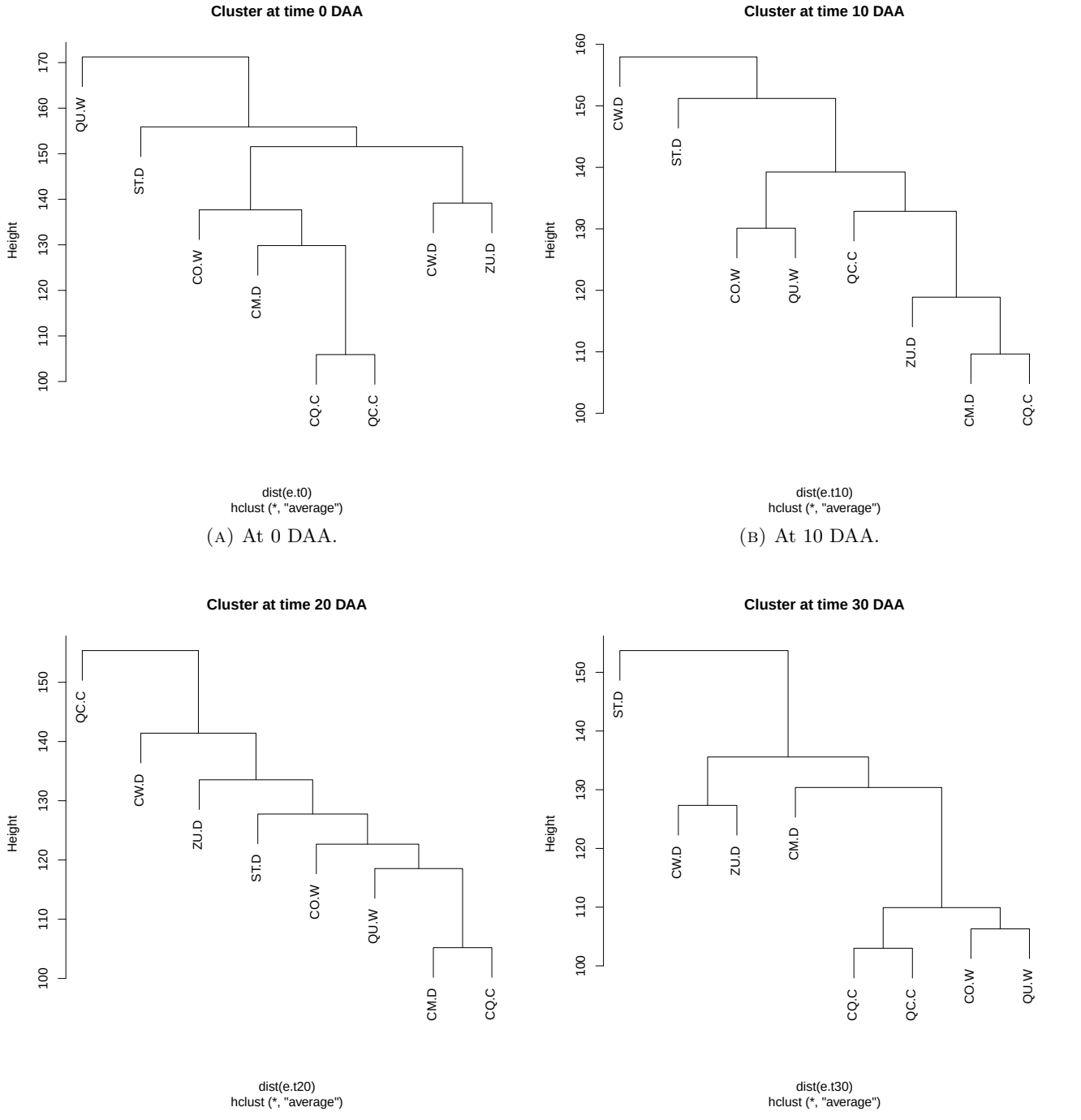
(1) Compare between them the dendrograms obtained with the minimum and maximum of the gene expression. How many groups (nodes, clusters) are shared between them?

(2) Compare the dendrograms obtained with the minimum and maximum of the gene expression with the ones in figures 16 and 18. Which of all the four dendrograms (if any) do you think that better reflect the phenotypic relations between accessions and why?

A different approach to the study of the diversity in gene expression that we have in the dataset is to construct dendrograms taking the expression of all genes at each time. With this we will can investigate how relation in the phenotypes per accession vary during fruit development. For each one of the times studied, we need to construct matrices in which one dimension (for example columns) is given by all 25,645 expressed genes and the other (rows) are the different accessions. Using such matrices we can construct dendrograms for the accessions at each one of the times. Calculations are presented in Box 26.

```
1  ------------------------------------> Box  26  <--------------------------------------
2  # Constructing matrices per expression time
3  > ac.type$key
4  [1] "CM.D" "CO.W" "CQ.C" "CW.D" "QC.C" "QU.W" "ST.D" "ZU.D"
5  # We can isolate the data only for expressed genes
6  > names(chili) # Remember the column names
7   [1] "ac"     "id"     "zT0"    "zT10"   "zT20"   "zT30"   "zT40"   "zT50"   "zT60"
8  [10] "in.all"
9
10 > temp <- chili[chili$in.all, 1:9]
11 > nrow(temp)
12 [1] 205160
13 > nrow(temp)/8 # Number of genes expressed in all accessions
14 [1] 25645
15 # We need to be sure that the order of the identifiers
16 # (id) is the same for all 8 accessions
17
18 # All values must be 1 (8*7/2=28 comparisons)
19 for(i in 1:7){
20 for(j in (i+1):8){
21 print(prod(temp$id[temp$ac==ac.type$key[i]] ==  temp$id[temp$ac==ac.type$key[j]]))
22 }}
23 [1] 1
24 [1] 1
25 [1] 1
26 ... # OK!
27
28 # Now we can form the matrices, one for each time
29 > e.t0 <- matrix(NA, nrow=8, ncol=25645, dimnames=list(ac.type$key,
30         temp$id[temp$ac==ac.type$key[1]]))
31 > e.t0[1:3,1:5]
32       3 12 15 19 22
33 CM.D NA NA NA NA NA
34 CO.W NA NA NA NA NA
35 CQ.C NA NA NA NA NA
36 # The other empty matrices
```

```
> e.t10<-e.t0;e.t20<-e.t0;e.t30<-e.t0;e.t40<-e.t0;e.t50<-e.t0;e.t60<-e.t0

# Fill the matrices
for(i in 1:8){
e.t0[i,] <- as.vector(temp$zT0[temp$ac==ac.type$key[i]])
e.t10[i,] <- as.vector(temp$zT10[temp$ac==ac.type$key[i]])
e.t20[i,] <- as.vector(temp$zT20[temp$ac==ac.type$key[i]])
e.t30[i,] <- as.vector(temp$zT30[temp$ac==ac.type$key[i]])
e.t40[i,] <- as.vector(temp$zT40[temp$ac==ac.type$key[i]])
e.t50[i,] <- as.vector(temp$zT50[temp$ac==ac.type$key[i]])
e.t60[i,] <- as.vector(temp$zT60[temp$ac==ac.type$key[i]])
}

# Now we can construct the dendrograms
# Note that the distances will be very large
# (they are distance in an space of 25645 dimensions!)
# Example
> dist(e.t0)
         CM.D     CO.W     CQ.C     CW.D     QC.C     QU.W     ST.D
CO.W 142.3664
CQ.C 141.4306 146.7854
CW.D 161.0883 154.8904 150.5572
QC.C 118.2417 123.8758 105.9111 145.0851
QU.W 154.0925 142.6508 185.6415 197.9718 158.2962
ST.D 155.5271 150.9860 162.2688 165.7146 145.8805 163.7364
ZU.D 168.8677 146.2548 144.8632 139.1648 140.8023 196.3171 154.9773

# Constructing the dendrograms
d.t0 <- hclust(dist(e.t0), "average")
d.t10 <- hclust(dist(e.t10), "average")
d.t20 <- hclust(dist(e.t20), "average")
d.t30 <- hclust(dist(e.t30), "average")
d.t40 <- hclust(dist(e.t40), "average")
d.t50 <- hclust(dist(e.t50), "average")
d.t60 <- hclust(dist(e.t60), "average")

# Plotting the dendrogram
# (figures 19 and 20)

# Figure 19
plot(d.t0, main="Cluster at time 0 DAA")
plot(d.t10, main="Cluster at time 10 DAA")
plot(d.t20, main="Cluster at time 20 DAA")
plot(d.t30, main="Cluster at time 30 DAA")

# Figure 20
plot(d.t40, main="Cluster at time 40 DAA")
plot(d.t50, main="Cluster at time 50 DAA")
plot(d.t60, main="Cluster at time 60 DAA")
--------------------------------------------------------------------------------
```



(A) At 0 DAA.

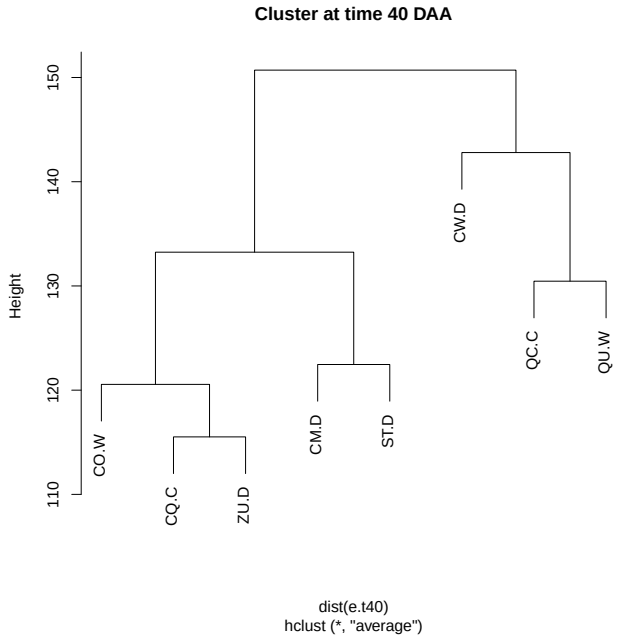
(B) At 10 DAA.

(C) At 20 DAA.

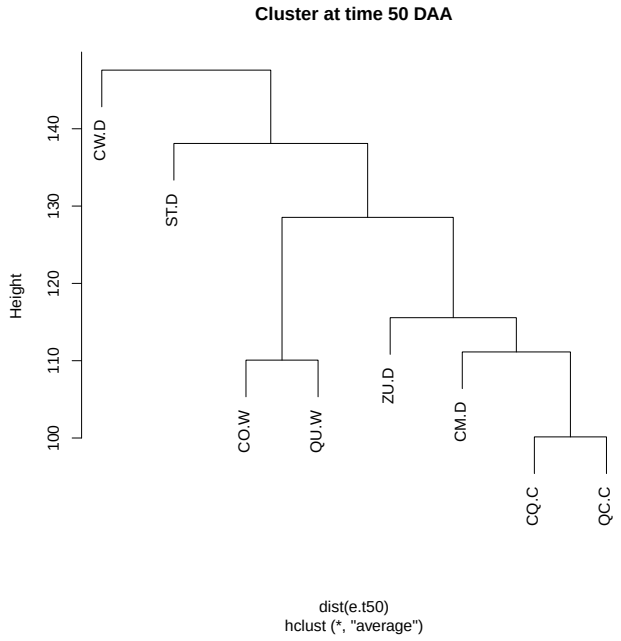
(D) At 30 DAA.

FIGURE 19. Dendrograms obtained for each specific time.

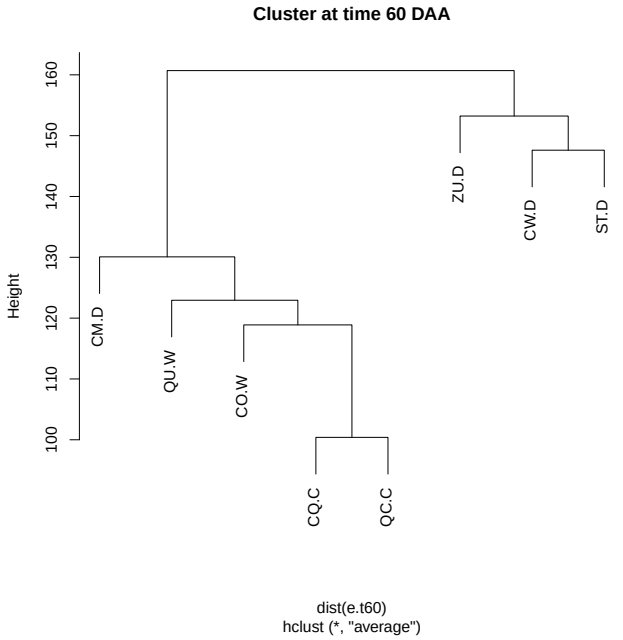
In Figures 19 and 20 we can notice that there are some changes in the clustering between accessions at different time points. Biologically this means that the molecular phenotype of the accessions is dynamic trough time, and thus the 'likeness' between accessions depends on the time point when we are measuring the phenotype. There are two aspects in the determination of a dendrogram, first its 'topology', that is the groups which are formed and, second, the heights at which the group is formed. In the next section we will see how to make a quantitative assessment of the differences between two dendrograms.



(A) At 40 DAA.



(B) At 50 DAA.



(C) At 60 DAA.

FIGURE 20. Dendrograms obtained for each specific time.

4.2. **Likeness between dendrograms.** A (bifurcating) dendrogram obtained from n units (accessions in our case) will have exactly $n-1$ different clusters or 'nodes'. If the same units are clustered by different means (for example, by using different clustering methods or different measures) we are interested in obtaining a coefficient of likeness between them. An obvious measure of the likeness between two dendrograms of the same units is given by the number of shared 'nodes' or clusters between such structures. Note that if two dendrograms, say, D_1 and D_2 are the result of clustering the same n units, and thus

they have $n-1$ clusters each, the largest cluster will include all n units and thus will be always equal for D_1 and D_2. However, the remaining $n-2$ clusters can be compared between D_1 and D_2.

We have seen that a cluster object, obtained with `hclust()` is an object with various components. To compare clusters we need first to convert them into sets, where each one of the nodes is a set. I programed the function "`hclust2sets()`" to do that. Then a second function, "`compare.hclust()`" can be used to compare the clusters, identifying how many nodes are shared between them. `Box 27` presents examples and show the comparisons between dendrograms in figures 19 and 20.

```
1  ----------------------------------> Box  27  <--------------------------------------
2  # Being in your CabanaR/ directory load the functions to compare dendrograms
3  > source("LikenessFun.txt") # Load the text file that contains the functions.
4  > likeness # A vector with the names of the functions
5  [1] "likeness"       "hclust2sets"    "compare.hclust"
6
7  # Before proceeding, remember the names of the objects that contain the
8  # dendrograms of figures 19 and 20:
9
10 # Let's see and briefly analyze the function "hclust2sets"
11 > ls(patt="d.t") # Output not shown
12
13 # How an hclust object is composed?
14 > class(d.t0)
15 [1] "hclust"
16 > names(d.t0)
17 [1] "merge"       "height"      "order"       "labels"      "method"      "call"
18 [7] "dist.method"
19 > d.t0$merge # The matrix of "merges"
20      [,1] [,2]
21 [1,]   -3   -5
22 [2,]   -1    1
23 [3,]   -2    2
24 [4,]   -4   -8
25 [5,]    3    4
26 [6,]   -7    5
27 [7,]   -6    6
28 > d.t0$labels # The vector of labels (units in the dendrogram)
29 [1] "CM.D" "CO.W" "CQ.C" "CW.D" "QC.C" "QU.W" "ST.D" "ZU.D"
30
31 # Let's now see the function "hclust2sets"
32 > hclust2sets
33 function(x){
34         # hclust2sets
35         # Converts an hclust object to sets of the original elements
36         if(class(x) != "hclust") stop("Argument x must be an hclust object!")
37         # names of x:
38         # 1 - "merge", 2 - "height", 3 - "order", 4 - "labels",
39         # 5 - "method", 6 - "call", 7 - "dist.method".
40         m <- x[[1]] # The merge component
41         la <- x[[4]] # Labels (names of OTUs)
42         n.c <- nrow(m) # Number of clusters (n-1)
43
44         # print(m)
```

```
        # print(la)
        # NOTE that NEGATIVE numbers within m are the original elements
        clu <- vector("list", n.c)
        names(clu) <- paste("clu", c(1:n.c), sep=".")
        for(i in 1:n.c){
                # for each row of m (cluster)
                for(j in 1:2){
                        # For first and second column
                        if(m[i,j]<0){
                                # If the element is negative
                                clu[[i]] <- unique(c(clu[[i]], la[-m[i,j]])) # The label
                                m[i,j] <- 0
                        }
                }
        }

        # Now clusters have the original values and m contains
        # only positive (or 0) values
        for(i in 1:n.c){
                for(j in 1:2){
                        if(m[i,j]>0){
                                # We need to include the members of the corresponding clusters
                                clu[[i]] <- union(clu[[i]], clu[[m[i,j]]])
                                }
                }
        }
        clu
        }

# Let's try that function on two of our dendrograms
> hclust2sets(d.t0) # Dendrogram of Figure 19 A seen as sets:
$clu.1
[1] "CQ.C" "QC.C"
$clu.2
[1] "CM.D" "CQ.C" "QC.C"
$clu.3
[1] "CO.W" "CM.D" "CQ.C" "QC.C"
$clu.4
[1] "CW.D" "ZU.D"
$clu.5
[1] "CO.W" "CM.D" "CQ.C" "QC.C" "CW.D" "ZU.D"
$clu.6
[1] "ST.D" "CO.W" "CM.D" "CQ.C" "QC.C" "CW.D" "ZU.D"
$clu.7
[1] "QU.W" "ST.D" "CO.W" "CM.D" "CQ.C" "QC.C" "CW.D" "ZU.D"
# Check this result visually with Figure 19 A.

> hclust2sets(d.t10) # Dendrogram of Figure 19 b seen as sets:
$clu.1
[1] "CM.D" "CQ.C"
$clu.2
[1] "ZU.D" "CM.D" "CQ.C"
```

```
$clu.3
[1] "CO.W" "QU.W"
$clu.4
[1] "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.5
[1] "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.6
[1] "ST.D" "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.7
[1] "CW.D" "ST.D" "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"
# Check this result visually with Figure 19 B.

# Can you say which sets (clusters) are shared?

# Now, let's see function "compare.hclust"
> compare.hclust
function(x=d.t0, y=d.t10, give.matrix = FALSE, do.cat = FALSE){
        # compare.hclust
        # Compare two products of hclust to give the number of shared clusters
        # depends on "hclust2sets"
        x.sets <- hclust2sets(x)
        y.sets <- hclust2sets(y)
        n.c <- length(x.sets)
        ind <- c(1:n.c)
        if(n.c != length(y.sets)) stop("x and y are not comparable.")
        res <- matrix(FALSE, nrow=n.c, ncol=n.c, dimnames=list(ind, ind))
        for(i in 1:n.c){
                for(j in 1:n.c){
                        res[i, j] <- setequal(x.sets[[i]], y.sets[[j]])
                }
        }
        if(give.matrix){
                return(res)
        } else {
                n.e.d <- sum(1*diag(res)) # Number of equalities in main diagonal
                e.o.d <- sum(1*res[upper.tri(res)]) # Number of equalities out main diagonal
                eq <- n.e.d+e.o.d-1
                if(do.cat){
                        cat("\nValid Equalities / Possible Equalities ", eq, "/", n.c-1, " = " ,
                        eq / (n.c-1), "\n", sep='')
                }
                return(eq / (n.c-1))
        }
}

# And let's try that function (with alternative parameters)
> compare.hclust(x=d.t0, y=d.t10, give.matrix = FALSE, do.cat = FALSE)
[1] 0
> compare.hclust(x=d.t0, y=d.t10, give.matrix = FALSE, do.cat = TRUE)

Valid Equalities / Possible Equalities 0/6 = 0
[1] 0
```

```
> compare.hclust(x=d.t0, y=d.t10, give.matrix = TRUE, do.cat = TRUE)
      1     2     3     4     5     6     7
1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
6 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
7 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
--------------------------------------------------------------------------------
```

From the above results we see that the dendrograms obtained from gene expression at 0 and 10 DAA do not share any (internal) cluster! Thus the 'likeness' between them is 0. This is not very surprising if we take into account that with $n = 8$, the number of bifurcating trees is $n_B = 135,135$, thus, *a priory* the probability of obtaining exactly the SAME dendrogram in this case is very small, $1/135135 \approx 7.4 \times 10^{-6}$ An interesting –but difficult question, is to calculate the distribution of the 'likeness' coefficient. Now, we can ask which is the likeness between all possible pairs of the 7 dendrograms presented in figures 19 and 20. Note that there are $(7 \times (7-1))/2 = 21$ of such pairs. `Box 28` presents the calculations.

```
------------------------------------> Box  28  <-------------------------------------
# Dendrograms in figures 19 and 20 are 7 we can make all 7*6/2 = 21 comparisons
# Let's make a data frame to keep the results
> comp.dend <- data.frame(d1=rep("",21), d2=rep("",21), lik=rep(NA,21), stringsAsFactors=F)

> temp.list <- list(d.t0, d.t10, d.t20, d.t30, d.t40, d.t50, d.t60)
> names(temp.list) <- c("d.t0", "d.t10", "d.t20", "d.t30", "d.t40", "d.t50", "d.t60")

k <- 0
for(i in 1:6){
for(j in (i+1):7){
k <- k+1
comp.dend$d1[k] <- names(temp.list)[i]
comp.dend$d2[k] <- names(temp.list)[j]
comp.dend$lik[k] <- compare.hclust(temp.list[[i]], temp.list[[j]])
}
}

> head(comp.dend)
    d1    d2       lik
1 d.t0 d.t10 0.0000000
2 d.t0 d.t20 0.0000000
3 d.t0 d.t30 0.3333333
4 d.t0 d.t40 0.0000000
5 d.t0 d.t50 0.3333333
6 d.t0 d.t60 0.1666667
> summary(comp.dend$lik)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.1111  0.1667  0.5000
> comp.dend[comp.dend$lik>0,]
      d1    d2       lik
3   d.t0 d.t30 0.3333333
5   d.t0 d.t50 0.3333333
6   d.t0 d.t60 0.1666667
```

```
7  d.t10 d.t20 0.1666667
10 d.t10 d.t50 0.5000000
17 d.t30 d.t50 0.3333333
18 d.t30 d.t60 0.3333333
21 d.t50 d.t60 0.1666667

# The "more alike" dendrograms share 50% of their clusters
> 1*compare.hclust(d.t10, d.t50, give.matrix=T)
  1 2 3 4 5 6 7
1 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0
3 0 1 0 0 0 0 0
4 0 0 0 1 0 0 0
5 0 0 0 0 1 0 0
6 0 0 0 0 0 1 0
7 0 0 0 0 0 0 1
# (here we convert FALSE to 0 and TRUE to 1)

> hclust2sets(d.t10)
$clu.1
[1] "CM.D" "CQ.C"
$clu.2
[1] "ZU.D" "CM.D" "CQ.C"
$clu.3
[1] "CO.W" "QU.W"
$clu.4
[1] "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.5
[1] "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.6
[1] "ST.D" "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"
$clu.7
[1] "CW.D" "ST.D" "CO.W" "QU.W" "QC.C" "ZU.D" "CM.D" "CQ.C"

> hclust2sets(d.t50)
$clu.1
[1] "CQ.C" "QC.C"
$clu.2
[1] "CO.W" "QU.W"
$clu.3
[1] "CM.D" "CQ.C" "QC.C"
$clu.4
[1] "ZU.D" "CM.D" "CQ.C" "QC.C"
$clu.5
[1] "CO.W" "QU.W" "ZU.D" "CM.D" "CQ.C" "QC.C"
$clu.6
[1] "ST.D" "CO.W" "QU.W" "ZU.D" "CM.D" "CQ.C" "QC.C"
$clu.7
[1] "CW.D" "ST.D" "CO.W" "QU.W" "ZU.D" "CM.D" "CQ.C" "QC.C"
-------------------------------------------------------------------------------
```

4.2.1. *Excercise.* How can we calculate the distribution of the likeness coefficient between two dendrograms given by the function "`compare.hclust`"? An interesting approach will be to use a resampling method, as for example bootstrapping. Here I present the proposed method as `Box 29` and the exercise consist in understanding the function and using this for different data, answering the questions below.

```
1  ------------------------------------> Box  29  <--------------------------------------
2  likeness.boot <-
3          function(de=d.t0, da=e.t0, B=100){
4          # likeness.boot
5          # Obtains a vector with likeness between an original dendrogram: de
6          # (de must be a dendrogram produced by "hclust" from matrix "da")
7          # and B bootstrap replicates of such dendrogram.
8
9          # Recover the method and distance method.
10         me <- de$method
11         dis.me <- de$dist.method
12         res <- rep(NA, B) # An empty vector to contain the results.
13
14         n.c <- ncol(da)
15
16         # Loop to obtain each value of likeness using function
17         # compare.hclust(x = d.t0, y = d.t10, give.matrix = FALSE, do.cat = FALSE)
18         for(i in 1:B){
19                 sam <- sample(c(1:n.c), size=n.c, replace=TRUE)
20                 db <- hclust(dist(da[,sam], method=dis.me), method=me)
21                 res[i] <- compare.hclust(x = de, y = db, give.matrix = FALSE, do.cat = FALSE)
22         }
23         res
24 }
25 ### NOTE: The function is also in file "likeness_boot.txt"
26 # You can use source("likeness_boot.txt") to load it in your environment.
27 ------------------------------------------------------------------------------------
```

(1) Run the function with default parameters, keeping the results. Obtain an interpret a summary of the results.

(2) You have a *known* dendrogram, `d.t0`. What will happen if you use less 'information' to construct the bootstrap replicates? To use less information you could limit the number of columns of the matrix `e.t0` which has 25,645 columns, corresponding to genes expressed in all accession. What will happen with the likeness coefficient if you use, for example, 10, 100, 1000 and 10000 and all (25,645) of those genes? Discuss your results.

5. Design of biodiversity studies

To consult the statistician after an experiment is finished is often merely to ask him to conduct a post mortem examination. He can perhaps say what the experiment died of.

Sir Ronald Fisher.

"**Crop production needs to increase to secure future food supplies, while reducing its impact on ecosystems. Detailed characterization of plant genomes and genetic diversity is crucial for meeting these challenges. Advances in genome sequencing and assembly are being used to access the large and complex genomes of crops and their wild relatives. These have helped to identify a wide spectrum of genetic variation and permitted the association of genetic diversity with diverse agronomic phenotypes. In combination with improved and automated phenotyping assays and functional genomic studies, genomics is providing new foundations for crop-breeding systems.**"

Genomic innovation for crop improvement (Nature).
(Bevan et al., 2017)

As next generation sequencing becomes cheaper and more available, plant genomics studies will gain a central role in the improvement of crops, including discovery of genetic variation and this will enhance performance and increase the efficiency of plant breeding. Here we have review just the tip of the iceberg of methods to study genomic diversity, an area so large that it has been difficult for me to make a rational selection of the topics to present. My focus in the design of the themes and datasets employed has been to try to gain a good understanding –trough practice, of some of the elemental methods to study genomic diversity. I avoided the approach of just answering "*How do I run this R package with this data?*" It is my conviction that it is far more important to know what you are doing, that just gaining the ability to do it. I hope that you will have more confidence in the data analysis of diversity after this workshop.

When designing a genomic study of diversity it is paramount to have the general aim of the study as clear as possible and have a list of particular objectives. The following list present relevant questions that could help in determining the research plan.

Planning a genomic study of crop diversity

- General aim of the study.
  - Is a single specie to be studied or closely related subspecies?
  - Are the results to be used in plant breeding programs?
  - In conservation of diversity efforts?
  - Are farmers, agronomists, plant breeders, $\cdots$, involved in sample selection?
- Available knowledge.
  - Is the genome of the specie available?
  - What is known in this specie from previous diversity studies?
- Available samples.
  - Is the sampling part of the project or it will rely on a collection of samples?
  - In the first case:
    - Which is the sampling unit? (plant, group of plants, accession, landrace, $\cdots$)
    - Are the geographic limits of the study well determined?
    - Is the sampling scheme well determined? (random, stratified, $\cdots$)

  - In the second case (available collection):
    * Which is the sampling unit? (plant, group of plants, accession, landrace, $\cdots$)
    * Is the collection in the field (*in situ*) or is it in a seed bank (*ex situ*)?
    * Number of sampling units and seeds (or vegetative tissues) available: How large is the collection?
  - Are phenotypic and geographic characters available for each sample unit?
- Selection of marker method.
  - Which molecular marker technology is available (genome sequencing, PCR, etc.)?
  - Perform a cost / benefit analysis. Could include simulation using data from previous studies.
  - Is the molecular marker study to be commissioned to a company or performed in the own lab?
  - Make a detailed description of the molecular methods to be employed.
- Statistical analysis of data.
  - Describe the pipeline of the analysis and clearly show how the results will fulfill the main objective.

5.0.1. *Excercise: Your dreamed project revisited.* How will you modify the project that you previously proposed? Again, you have around 200 words to present it.

I hope that your abilities and confidence for the study of genomic diversity of crops had improved with this workshop.

References

Agarwal M, Shrivastava N, and Padh H (2008) Advances in molecular marker techniques and their applications in plant sciences. *Plant cell reports*, 27, 617–631.

Azman A, Mhiri C, Grandbastien M, and TAM S (2014) Transposable elements and the detection of somaclonal variation in plant tissue culture: a review. *Malaysian Applied Biology*, 43, 1–12.

Baird NA, Etter PD, Atwood TS, Currey MC, Shiver AL, Lewis ZA, Selker EU, Cresko WA, and Johnson EA (2008) Rapid snp discovery and genetic mapping using sequenced rad markers. *PloS one*, 3, e3376.

Barbazuk WB, Emrich SJ, Chen HD, Li L, and Schnable PS (2007) Snp discovery via 454 transcriptome sequencing. *The plant journal*, 51, 910–918.

Barve V and Otegui J (2016) bdvis: visualizing biodiversity data in r. *Bioinformatics*, 32, 3049–3050.

Beckmann J and Soller M (1990) Toward a unified approach to genetic mapping of eukaryotes based on sequence tagged microsatellite sites. *Nature Biotechnology*, 8, 930–932.

Bevan MW, Uauy C, Wulff BB, Zhou J, Krasileva K, and Clark MD (2017) Genomic innovation for crop improvement. *Nature*, 543, 346.

Botstein D, White RL, Skolnick M, and Davis RW (1980) Construction of a genetic linkage map in man using restriction fragment length polymorphisms. *American journal of human genetics*, 32, 314.

Cardoso P, Rigal F, and Carvalho JC (2015) Bat–biodiversity assessment tools, an r package for the measurement and estimation of alpha and beta taxon, phylogenetic and functional diversity. *Methods in Ecology and Evolution*, 6, 232–236.

Casa AM, Brouwer C, Nagel A, Wang L, Zhang Q, Kresovich S, and Wessler SR (2000) The mite family heartbreaker (hbr): molecular markers in maize. *Proceedings of the National Academy of Sciences*, 97, 10083–10089.

Chamberlain LJRSA (2013) *Rphylip: An R interface for PHYLIP.* URL `http://www.phytools.org/Rphylip`.

Chang RY, O?donoughue L, and Bureau T (2001) Inter-mite polymorphisms (imp): a high throughput transposon-based genome mapping and fingerprinting approach. *TAG Theoretical and Applied Genetics*, 102, 773–781.

Chazdon RL, Colwell RK, Denslow JS, and Guariguata MR (1998) Statistical methods for estimating species richness of woody regeneration in primary and secondary rain forests of northeastern costa rica. Technical report, Costa Rica.

Davey JW, Hohenlohe PA, Etter PD, Boone JQ, Catchen JM, and Blaxter ML (2011) Genome-wide genetic marker discovery and genotyping using next-generation sequencing. *Nature Reviews Genetics*, 12, 499–510.

Deragon J, Casacuberta J, and Panaud O (2008) Plant transposable elements. In *Plant Genomes*, volume 4, pp. 69–82. Karger Publishers.

Di Battista T, Fortuna F, and Maturo F (2017) Bioftf: an r package for biodiversity assessment with the functional data analysis approach. *Ecological indicators*, 73, 726–732.

Elshire RJ, Glaubitz JC, Sun Q, Poland JA, Kawamoto K, Buckler ES, and Mitchell SE (2011) A robust, simple genotyping-by-sequencing (gbs) approach for high diversity species. *PloS one*, 6, e19379.

Felsenstein J (1978) The number of evolutionary trees. *Systematic zoology*, 27, 27–33.

Felsenstein J (1993) *PHYLIP (phylogeny inference package), version 3.5 c.* Joseph Felsenstein.

Flavell AJ, Knox MR, Pearce SR, and Ellis T (1998) Retrotransposon-based insertion polymorphisms (rbip) for high throughput marker analysis. *The Plant Journal*, 16, 643–650.

Ganal MW, Polley A, Graner EM, Plieske J, Wieseke R, Luerssen H, and Durstewitz G (2012) Large snp arrays for genotyping in crop plants. *Journal of biosciences*, 37, 821–828.

Gupta P, Roy J, and Prasad M (2001) Single nucleotide polymorphisms: a new paradigm for molecular marker technology and dna polymorphism detection with emphasis on their use in plants. *Curr Sci*, 80, 524–535.

Haseneyer G, Schmutzer T, Seidel M, Zhou R, Mascher M, Schön CC, Taudien S, Scholz U, Stein N, Mayer KF, et al. (2011) From rna-seq to large-scale genotyping-genomics resources for rye (secale cereale l.). *BMC Plant Biology*, 11, 131.

Hayano-Kanashiro C, Martínez de la Vega O, Reyes-Valdés MH, Pons-Hernández JL, Hernández-Godinez F, Alfaro-Laguna E, Herrera-Ayala JL, Vega-Sánchez MC, Carrera-Valtierra JA, and Simpson J (2017) An ssr-based approach incorporating a novel algorithm for identification of rare maize genotypes facilitates criteria for landrace conservation in mexico. *Ecology and evolution*, 7, 1680–1690.

Hayashi K, Hashimoto N, Daigen M, and Ashikawa I (2004) Development of pcr-based snp markers for rice blast resistance genes at the piz locus. *Theoretical and Applied Genetics*, 108, 1212–1220.

He J, Zhao X, Laroche A, Lu ZX, Liu H, and Li Z (2014) Genotyping-by-sequencing (gbs), an ultimate marker-assisted selection (mas) tool to accelerate plant breeding. *Frontiers in plant science*, 5.

Hiremath PJ, Farmer A, Cannon SB, Woodward J, Kudapa H, Tuteja R, Kumar A, BhanuPrakash A, Mulaosmanovic B, Gujaria N, et al. (2011) Large-scale transcriptome analysis in chickpea (cicer arietinum l.), an orphan legume crop of the semi-arid tropics of asia and africa. *Plant biotechnology journal*, 9, 922–931.

Hirochika H (1997) Retrotransposons of rice: their regulation and use for genome analysis. *Plant molecular biology*, 35, 231–240.

Huang X, Feng Q, Qian Q, Zhao Q, Wang L, Wang A, Guan J, Fan D, Weng Q, Huang T, et al. (2009) High-throughput genotyping by whole-genome resequencing. *Genome research*, 19, 1068–1076.

Jiang C, Mithani A, Gan X, Belfield EJ, Klingler JP, Zhu JK, Ragoussis J, Mott R, and Harberd NP (2011) Regenerant arabidopsis lineages display a distinct genome-wide spectrum of mutations conferring variant phenotypes. *Current Biology*, 21, 1385–1390.

Kaeppler SM, Kaeppler HF, and Rhee Y (2000) Epigenetic aspects of somaclonal variation in plants. *Plant molecular biology*, 43, 179–188.

Kalendar R, Grob T, Regina M, Suoniemi A, and Schulman A (1999) Irap and remap: two new retrotransposon-based dna fingerprinting techniques. *TAG Theoretical and Applied Genetics*, 98, 704–711.

Karp A (1995) Somaclonal variation as a tool for crop improvement. *Euphytica*, 85, 295–302.

Kindt R and Coe R (2005) *Tree diversity analysis. A manual and software for common statistical methods for ecological and biodiversity studies*. World Agroforestry Centre (ICRAF), Nairobi (Kenya). URL `http://www.worldagroforestry.org/output/tree-diversity-analysis`. ISBN 92-9059-179-X.

Komori T and Nitta N (2005) Utilization of the caps/dcaps method to convert rice snps into pcr-based markers. *Breeding Science*, 55, 93–98.

Li G and Quiros CF (2001) Sequence-related amplified polymorphism (srap), a new marker system based on a simple pcr reaction: its application to mapping and gene tagging in brassica. *TAG Theoretical and Applied Genetics*, 103, 455–461.

Mah JT and Chia K (2007) A gentle introduction to snp analysis: resources and tools. *Journal of bioinformatics and computational biology*, 5, 1123–1138.

Maitner BS, Boyle B, Casler N, Condit R, Donoghue J, Durán SM, Guaderrama D, Hinchliff CE, Jørgensen PM, Kraft NJ, et al. (2018) The bien R package: A tool to access the botanical information and ecology network (bien) database. *Methods in Ecology and Evolution*, 9, 373–379.

Martínez O (2018) Selection of molecular markers for the estimation of somaclonal variation. In *Plant Cell Culture Protocols*, pp. 103–129. Springer.

Martínez-López LA, Ochoa-Alejo N, and Martínez O (2014) Dynamics of the chili pepper transcriptome during fruit development. *BMC genomics*, 15, 143.

May F, Gerstner K, McGlinn DJ, Xiao X, and Chase JM (2018) mobsim: An r package for the simulation and measurement of biodiversity across spatial scales. *Methods in Ecology and Evolution*, 9, 1401–1408.

McCallum CM, Comai L, Greene EA, and Henikoff S (2000) Targeted screening for induced mutations. *Nature biotechnology*, 18, 455.

McClelland M and Welsh J (1994) Rna fingerprinting by arbitrarily primed pcr. *Genome Research*, 4, S66–S81.

Miller MR, Dunham JP, Amores A, Cresko WA, and Johnson EA (2007) Rapid and cost-effective polymorphism identification and genotyping using restriction site associated dna (rad) markers. *Genome research*, 17, 240–248.

Nakamura Y, Leppert M, O'Connell P, Wolff R, Holm T, Culver M, Martin C, Fujimoto E, Hoff M, Kumlin E, et al. (1987) Variable number of tandem repeat (vntr) markers for human gene mapping. *Science*, 235, 1616–1623.

Narum SR, Buerkle CA, Davey JW, Miller MR, and Hohenlohe PA (2013) Genotyping-by-sequencing in ecological and conservation genomics. *Molecular ecology*, 22, 2841–2847.

Novaes E, Drost DR, Farmerie WG, Pappas GJ, Grattapaglia D, Sederoff RR, and Kirst M (2008) High-throughput gene and snp discovery in eucalyptus grandis, an uncharacterized genome. *BMC genomics*, 9, 312.

Nybom H (2004) Comparison of different nuclear dna markers for estimating intraspecific genetic diversity in plants. *Molecular ecology*, 13, 1143–1155.

Orita M, Iwahana H, Kanazawa H, Hayashi K, and Sekiya T (1989) Detection of polymorphisms of human dna by gel electrophoresis as single-strand conformation polymorphisms. *Proceedings of the National Academy of Sciences*, 86, 2766–2770.

Paran I and Michelmore R (1993) Development of reliable pcr-based markers linked to downy mildew resistance genes in lettuce. *Tag Theoretical and Applied Genetics*, 85, 985–993.

Peredo EL, Arroyo-Garcia R, and Revilla MÁ (2009) Epigenetic changes detected in micropropagated hop plants. *Journal of plant physiology*, 166, 1101–1111.

Poland JA and Rife TW (2012) Genotyping-by-sequencing for plant breeding and genetics. *The Plant Genome*, 5, 92–102.

Retief JD (2000) Phylogenetic analysis using phylip. In *Bioinformatics methods and protocols*, pp. 243–258. Springer.

Reyes-Valdés MH, Burgueño J, Singh S, Martínez O, and Sansaloni CP (2018) An informational view of accession rarity and allele specificity in germplasm banks for management and conservation. *PloS one*, 13, e0193346.

Reyes-Valdés MH, Santacruz-Varela A, Martínez O, Simpson J, Hayano-Kanashiro C, and Cortés-Romero C (2013) Analysis and optimization of bulk dna sampling with binary scoring for germplasm characterization. *PloS one*, 8, e79936.

Robertson T, Döring M, Guralnick R, Bloom D, Wieczorek J, Braak K, Otegui J, Russell L, and Desmet P (2014) The gbif integrated publishing toolkit: facilitating the efficient publishing of biodiversity data on the internet. *PloS one*, 9, e102623.

Sahijram L, Soneji JR, and Bollamma K (2003) Invited review: Analyzing somaclonal variation in micropropagated bananas (musa spp.). *In Vitro Cellular and Developmental Biology-Plant*, 39, 551–556.

Shahinnia F and Sayed-Tabatabaei BE (2009) Conversion of barley snps into pcr-based markers using dcaps method. *Genetics and molecular biology*, 32, 564–567.

Suda J, Krahulcová A, Trávníek P, and Krahulec F (2006) Ploidy level versus dna ploidy level: an appeal for consistent terminology. *Taxon*, 55, 447–450.

Suzuki R and Shimodaira H (2006) Pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, 22, 1540–1542.

Va D, De N, Broeck N, Maes T, Sauer M, Zethof J, Keukeleire E, D?hauw M, Va M, Montagu N, et al. (1998) Transposon display identifies individual transposable elements in high copy number lines. *The Plant Journal*, 13, 121–129.

Väli Ü, Brandström M, Johansson M, and Ellegren H (2008) Insertion-deletion polymorphisms (indels) as genetic markers in natural populations. *BMC genetics*, 9, 8.

Varshney RK, Nayak SN, May GD, and Jackson SA (2009) Next-generation sequencing technologies and their implications for crop genetics and breeding. *Trends in biotechnology*, 27, 522–530.

Vos P, Hogers R, Bleeker M, Reijans M, Lee Tvd, Hornes M, Friters A, Pot J, Paleman J, Kuiper M, et al. (1995) Aflp: a new technique for dna fingerprinting. *Nucleic acids research*, 23, 4407–4414.

Wang D, Gong Z, et al. (2003) Target region amplification polymorphism (trap): a novel marker technique for plant genotyping. *Fen zi zhi wu yu zhong*, 2, 740–750.

Waugh R, McLean K, Flavell A, Pearce S, Kumar A, Thomas B, and Powell W (1997) Genetic distribution of bare–1-like retrotransposable elements in the barley genome revealed by sequence-specific amplification polymorphisms (s-sap). *Molecular and General Genetics MGG*, 253, 687–694.

Wenzl P, Carling J, Kudrna D, Jaccoud D, Huttner E, Kleinhofs A, and Kilian A (2004) Diversity arrays technology (dart) for whole-genome profiling of barley. *Proceedings of the National Academy of Sciences of the United States of America*, 101, 9915–9920.

Wieczorek J, Bloom D, Guralnick R, Blum S, Döring M, Giovanni R, Robertson T, and Vieglais D (2012) Darwin core: an evolving community-developed biodiversity data standard. *PloS one*, 7, e29715.

Williams JG, Kubelik AR, Livak KJ, Rafalski JA, and Tingey SV (1990) Dna polymorphisms amplified by arbitrary primers are useful as genetic markers. *Nucleic acids research*, 18, 6531–6535.

Wu Ks, Jones R, Danneberger L, and Scolnik PA (1994) Detection of microsatellite polymorphisms without cloning. *Nucleic Acids Research*, 22, 3257–3258.