Molecular data in R

Phylogeny, evolution & R

Vojtěch Zeisek

Department of Botany, Faculty of Science, Charles University, Prague
Institute of Botany, Czech Academy of Sciences, Průhonice
https://trapa.cz/, zeisek@natur.cuni.cz

January 31 to February 3, 2022
Outline I

1. Introduction

2. R
   - Installation
   - Let’s start with R
   - Basic operations in R
   - Tasks
   - Packages for our work

3. Data
   - Overview of data and data types
   - Microsatellites
   - AFLP
   - Notes about data
   - DNA sequences, SNP
   - VCF
Outline II

Export
Tasks

4 Alignment
   Overview and MAFFT
   MAFFT, Clustal, MUSCLE and T-Coffee
   Multiple genes
   Display and cleaning
Tasks

5 Basic analysis
   First look at the data
Statistics
MSN
Genetic distances
AMOVA
Hierarchical clustering
NJ (and UPGMA) tree
PCoA
Tasks

SNP
PCA and NJ

DAPC
Bayesian clustering
Discriminant analysis and visualization
Tasks

Spatial analysis
Moran’s I
Outline IV

sPCA
Mantel test
Monmonier
Geneland
Plotting maps
Tasks

Trees
Manipulations
MP
Seeing trees in the forest
Comparisons
Notes about plotting the trees
Tasks
Outline V

10 Evolution
   - PIC
   - Autocorrelation
   - pPCA
   - Decomposition
   - PGLS
   - GEE
   - Phylosignal
   - Ancestral state
   - Phenogram
   - Tasks

11 The end
   - Graphics
Outline VI

GitHub
Scripts
MetaCentrum
Functions
Loops
If-else branching
Solving problems
Resources
Summary
The end
The course information

• The course page:
  • Česky: https://trapa.cz/cs/kurz-molekularni-data-r-2022
• Subject in SIS: https://is.cuni.cz/studium/eng/predmety/index.php?do=predmet&kod=MB120C16
  • Česky: https://is.cuni.cz/studium/predmety/index.php?do=predmet&kod=MB120C16
  • For students having subscribed the subject, requirements are on next slide
• Working version is available at
  https://github.com/V-Z/course-r-mol-data — feel free to contribute, request new parts or report bugs
Requirements to exam ("zápočet")

1. Be present whole course.
2. Be active — ask and answer questions.
3. Process some data. This will be very variable and individual. Everyone should be able to take some data (according to her/his interest) and do several simple analysis (according to her/his interest). Students can of course use manual, internet, discuss with anyone. The aim is to repeat part the most interesting/important for the student and edit introduced commands to fit her/his needs. Students can thus bring their data (if they are not too large), download any data from the internet or I can give them some toy data.
4. Write at least one page (can be split into multiple articles) on Wikipedia about any method or related topic discussed during the course. Again, this is very open, students can write about any topic they like. I prefer native language of the student (typically to make larger non-English Wikipedia).
Materials to help you...

• Download the presentation from https://soubory.trapa.cz/rcourse/r_mol_data_phylogen.pdf

• Download the script from https://soubory.trapa.cz/rcourse/course_commands.r, use it and write your comments and notes to it during the course
  • Note: Open the R script in some good text editor (next slide) — showing syntax highlight, line numbers, etc. (NO Windows Notepad); the file is in UTF-8 encoding and with UNIX end of lines (so that too silly programs like Windows Notepad won’t be able to open it correctly)
  • The best is to open the script (or copy-paste the text) in e.g. RStudio or any other R GUI (slide 16) and directly work with it
  • Downloaded file must have extension *.r, not *.txt
  • Never ever open R script in software like MS Word — it destroys quotation marks and other things making script unusable
Importance of good text editor

Can your text editor...?

- Show syntax highlight
- Show line numbers
- Show space between brackets
- Open any encoding and EOL
- Fold source code
- Show line breaks
- Mark lines
- Kate
- KWrite
- Vim
- GNU Emacs
- Geany
- Bluefish
- Open multiple files
- Advanced search and replace
- Use regular expressions
- Make projects, add notes
- Use command line
- Check spelling
- Debug source code
- Gedit
- Notepad++
- Sublime
- Atom
- Nano
- And more...

The best option is to use text editor of selected R GUI (slide 16)...
Think before you type (and hit Enter)...

- Commands from file `course_commands.r` can be mostly directly launched without editing them, but before you do so...
  1. **Read** the command and all comments around, **do not blindly launch** it
  2. Ensure you understand, what is **aim** of the command (what it is supposed to do)
  3. Ensure you understand all **limitations** of the method (when you can use it and when not)
  4. Ensure you understand **syntax** of the command (its grammatical structure and what and how it technically does)

- Some **commands** from `course_commands.r` **do require** to be edited according to particular user’s computer — it is described in the comments around the command — **read also comments** around the command

- Learning R is effective only if you learn R syntax (language grammar), otherwise you only memorize commands without understanding them or blindly repeat someone’s code — in such case, you wouldn’t be able to solve any issue with your workflow
What we will and what we will not do...

We will go through...

• Basic introduction into R
• Analyzing phylogeny and evolution, population genetics and basic theory
  • DNA sequences, SNP, SSRs, AFLP, VCF, ...
  • Alignments
  • NJ, UPGMA, PCoA, DAPC, Bayesian clustering, ML, maximum parsimony, ...
  • Character evolution, ancestral state reconstructions, ...
  • Manipulations and analysis with trees
• Plotting

• Maps, spatial analysis, ...
• Basic creation of scripts
• And more...

We will not dig deep into...

• Detailed theory behind used methods
• Programming in R
• Other software related to the methods used (with exceptions of applications called from R)
• Other areas of R usage (ecology, biomedicine, proteomics, ...)

Vojtěch Zeisek (https://trapa.cz/)
The R

Basic introduction to work with R, installation of all required software

2 R

Installation
Let’s start with R
Basic operations in R
Tasks
Packages for our work
About R

• Project for Statistical Computing
• Open-source — freely available with source code — anyone can use and modify it and contribute its development
• Development is organized by non-governmental non-profit organization from Vienna
• Thousands of packages extending its functionality are available — all fields of computations in any scientific discipline
• Provides only command line interface — full control over the analysis, easy to rerun and/or modify analysis in the future, easy creation of scripts for batch analysis etc.
• Several projects provide convenient graphical user interfaces (GUI, slide 16)
• More details: https://www.r-project.org/
Graphical user interfaces (GUI) I

- Most users use some GUI — it is more convenient than plain command line
- Provide more comfortable interface for work with scripts (source code highlight, ...), overview of loaded packages and variables, easier work with figures, ...
- RStudio https://rstudio.com/products/rstudio/ — probably the most common, multi-platform, very powerful
- RKWard https://rkward.kde.org/ — feature very rich, developed mainly for Linux, available also for another operating systems
  - RKWard must be compiled for the same version of R as you use
  - If downloading for Windows or macOS, check your version of R and download respective version of RKWard
  - On Linux, do not mix package repositories, ensure RKWard is compiled for your R version (typically install both from same resource)
- R commander (Rcmdr) https://www.rcommander.com/ — multi-platform, not so rich as previous
Graphical user interfaces (GUI) II

• Java GUI for R https://rforge.net/JGR/ — Java (multi-platform, but with all Java issues like memory consumption)
• Emacs speaks statistics, R Tools for MS Visual Studio, Rattle, Radiant, ...
• And more...
• Pick one you like (from above list or any else) and install it...
When using RKward, consider change of settings of text editor for more comfortable work
Installation

Let's start with R

Basic operations in R

Data

Basic analysis

SNP

DAPC

Spatial analysis

Trees

Evolution

Packages for our work

Introduction

R

Data

Alignment

Basic analysis

SNP

DAPC

Spatial analysis

Trees

Evolution

The end

RStudio

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

20/421
When using RStudio, turn on soft line wrap, select Czech mirror to download packages and consider change of appearance for more comfortable work.
MS Windows & Apple macOS

- Got to [https://CRAN.R-project.org/](https://CRAN.R-project.org/)
- Download appropriate version and install as usual
- Download and install selected GUI (not required, but highly recommended)
- Most of packages are available as pre-compiled and can be immediately installed from R — it is convenient, but usually not tuned for particular computer architecture (type of CPU)
- Usually there are some problems every time new version of OS is released — it takes time to modify and recompile packages for new version of OS
- You have to check for new version of R manually
- RStudio is available from its download page
- RKWard is also available for Windows and macOS, but it requires some work to install it
Linux — general

- R, and usually also GUI, is available in repositories — use standard package management according to distribution
- Linux repositories provide automatic updates
- Packages are also partially available in repositories and can be installed and updated as usual application or from R
- Packages commonly have to be compiled — R will do it automatically, but install basic Linux packages for building of C, C++, FORTRAN, ...
- Compilation takes longer time and there are sometimes issues with missing dependencies (tools required by particular packages), but it can then provide higher performance...
• Install package **build-essential** (general tools to compile software, including R packages)

• Debian (and derivatives): follow instructions at [https://CRAN.R-project.org/bin/linux/debian/](https://CRAN.R-project.org/bin/linux/debian/)

• Ubuntu (and derivatives): follow instructions at [https://CRAN.R-project.org/bin/linux/ubuntu/](https://CRAN.R-project.org/bin/linux/ubuntu/)

• Install packages **R-base** (the R), **R-base-dev** (required to compile additional R packages — only some are available in repositories) and optionally **rkward** and/or **rstudio**

• RStudio is also available from its download page
Linux — openSUSE and SUSE Linux Enterprise

- See instructions at [https://CRAN.R-project.org/bin/linux/suse/](https://CRAN.R-project.org/bin/linux/suse/)
- Add repository/ies for appropriate version of your distribution
  - [https://download.opensuse.org/repositories/devel:/languages:/R:/patched/](https://download.opensuse.org/repositories/devel:/languages:/R:/patched/) (daily updated) or/and
  - [https://download.opensuse.org/repositories/devel:/languages:/R:/released/](https://download.opensuse.org/repositories/devel:/languages:/R:/released/) (updated with new R release)
- Install packages `R-base` (the R), `R-base-devel` (required to compile additional R packages — only some are available in repositories) and optionally `rstudio` and/or `rkward`
- Install packages `patterns-openSUSE-devel_basis` and `gcc-fortran` for compilation of R packages when installing them from R (only some R packages are available in openSUSE repositories)
- RStudio is also available from its download page
Linux — RedHat, Fedora and derivatives like CENTOS, Scientific Linux, etc.

- See instructions at [https://CRAN.R-project.org/bin/linux/redhat/](https://CRAN.R-project.org/bin/linux/redhat/)
- Install packages **R-core** (the R), **R-core-devel** (required to compile additional R packages — only some are available in repositories) and optionally **rkward**
- RStudio is available from its [download page](https://trapa.cz/)
Important note about names of directories

- **There must not be any spaces or accented characters** in the path to R working directory or local R library, otherwise some R functions can fail (and there is no other solution than creating a new directory/user).

- If the user has e.g. on Windows (where the problem is the biggest) path like `C:/Documents and Settings/Šíleně úpějící kůň/kurzíček`, change it to something like `C:/Users/username/rcourse`, otherwise user can experience a lot of problems...

- It might be required to make a new user on the computer...

- Similarly on macOS and Linux, avoid directory names with spaces and accented characters...
Sources of R packages

- **R CRAN** [https://CRAN.R-project.org/](https://CRAN.R-project.org/) — main and largest source of R packages (over 18,000 packages + many orphaned and archived — abandoned by developers, might be working)

- **Bioconductor** [https://bioconductor.org/](https://bioconductor.org/) — mainly bioinformatics packages, genomic data (over 2,000 packages)

- **R-Forge** [https://r-forge.r-project.org/](https://r-forge.r-project.org/) (over 2,100 packages)

- **RForge** [https://www.rforge.net/](https://www.rforge.net/) (much smaller)

- And more ([GitHub](https)), custom webs, ...

- Some packages are available from more resources

- Same name for function can be used in different packages (there is no central index) — to distinguish them call functions like this: `muscle::read.fasta()` vs. `seqinr::read.fasta()` — call function `read.fasta()` from package `muscle` or `seqinr` (and their parameters can be different...) — see further
CRAN keeps growing...

https://journal.r-project.org/archive/2021-1/cran.pdf
First steps in R

Recommended is usage of GUI (RKWard or RStudio)

- Linux (UNIX): open any terminal, type `R` and hit Enter
- Windows and Mac: find it as normal application in menu
- Type commands to work...
- Ever wished to be Harry Potter? Secret spells make magic operations :-) 
- Use arrows up and down to navigate in history
- `Ctrl+R` works as reverse search — searches text in history
How it works

• General look of R commands:

```r
function(argument1="SomeName", argument2=SomeVariable, argument3=8)
ModifiedObject <- SomeFunction(argument1=MyData, argument2=TRUE)
```

• New/modified object (with data, ...) is on the left: “<-” says to insert result of the function `SomeFunction` on the right into the object `ModifiedObject` on the left

• Functions have various parameters/arguments (in brackets, separated by commas):

```r
argument=ItsValue
```

• Arguments are named — if you keep order, no need to name them:

```r
SomeFunction(MyData, TRUE, 123, "SomeName")
```

• When only some of the arguments are in use, use the names (order doesn’t matter any more)

```r
SomeFunction(argument2=TRUE, argument3=123, argument1=MyData)
```

• Some arguments are required, some optional
Get help in R

# "#" marks comments - notes within code which are not executed
help(function) # Help for particular function (package must be loaded)
?funsion # Help for particular function (package must be loaded)
??SearchedTerm # Search for the term within all installed packages
help.search("searched phrase") # Search for the phrase within all
  # installed packages - return list of hits sorted according to
  # type and package (i.e. package::function)
require(sos) # More comprehensive search from packages
findFn("function") # Search for function name

? shows help for questioned function (in console type q to close it):
  ● Name of the package (top left)
  ● Function name (headline)
  ● Description
  ● Usage
  ● Comments on arguments
  ● Details
  ● Output value(s)
  ● About author(s)
  ● References to cite
  ● Example code

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Importance of working directory

- Very important point to get familiar with in R
- Default place to load/save, import/export data/results
  - It changes paths — one of the most common mistakes — something (input file, ...) is not found because of wrong path
  - Private folder for particular R project (task) prevents unwanted inferences with another tools/projects
- Without saving and loading the R data next time, it is not possible to do any longer work or to check the work in the future
- Get used that R **always** work in some directory and by default saves/loads files there
  - When starting RStudio/RK Ward, R usually starts in user’s home directory or so
- RStudio and RK Ward also save session information (list of opened files, ...) — very convenient
- Regularly save your work to prevent looses in case of crash or any other accident
- There **must not** be any space or accented characters in the path/directory name!
Where we are?

- In Linux/UNIX, R starts in current directory (use `cd` to change it before launching R)

- Set and check **working directory** in R:

```r
setwd("/some/path/") # Or "~/...". In Windows "C:/...
getwd() # Verifies where we are
dir() # Lists files and folders on the disk
ls() # Lists currently available R objects
```

- In Windows plain R (**File** | **working directory**), in RStudio (**Session** | **Set working directory**) or in RKWard (**Workspace** | **Set Working Directory**) set it in menu or by the above command

- R saves history of commands into file `.Rhistory` file within working directory (by default hidden in Linux/macOS)

- When closing R by `q()` you can save all R data in `.RData` (and command history in `.Rhistory`) file(s) and it/they can be loaded next time (files can be renamed)

- RStudio and RKWard help with this very much

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Types of objects

- As any programming language, R has plenty of types of objects (variables) with different features, usage and aims
- **Vectors** — numbers, characters, boolean
- **Matrices** — columns are of same type (numeric, character, etc.) and the same length
- **Arrays** — like matrices, but with possibly more dimensions
- **Data frames** — more general — columns can be of different type ("sheet of Excel")
- **Lists** — ordered collections of objects (vectors, matrices, ...) — not necessarily of the same type
- **Factors** — a vector of levels, e.g. populations, colors, etc.
- More “advanced” objects to store plots, genetic data, ...
  - Commonly called “**S3**” and “**S4**” objects in R terminology
  - Technically commonly just lists putting together various information
  - We will meet many of them...
- Functions require particular object types — take care about it
Popular object classes (we are going to use) I

- **AAbin** — stores amino acid sequences (aligned or not)
- **alignment** — aligned sequences (package seqinr)
- **dapc** — results of DAPC
- **dist** — distance matrices
- **DNAbin** — stores DNA sequences (aligned or not)
- **genind** — stores various genetic information for individuals
- **genlight** — variant of genind to store large multiple genomes
- **genpop** — like genind, but on population level
- **haplonet** — networks without reticulation
- **haplotype** — unique sequences from DNAbin
Popular object classes (we are going to use) II

- **hclust** — output of hierarchical clustering, can be converted to phylo
- **loci** — extension of data frame (DF), stores information about loci
- **matching** — binary phylogenetic trees
- **matrix** — general matrix (numeric or not)
- **pco; dudi** — results of PCA, PCoA, ...
- **phyDat** — “preparation” of data for some phylogenetic analysis (usually sequences and characters)
- **phylo** — phylogenetic information, typically trees
- **phylo4** — derived from phylo (more data), S4 instead of S3
- **SNPbin** — stores large SNP data for single genome
Popular object classes (we are going to use) III

- `spca` — results of sPCA
- `treeshape` — derived from hclust
- `vcfR` — imported (and possibly edited) VCF
- and more... common task is converting among formats...
- ...not all formats are (easily) convertible among each other...
- To get information about content of each data type see `getClassDef("data.frame")` (Or any other class name of loaded package) — there are information about slots within that classes you can access
- Common task is conversion among various formats — functions commonly require different input format
# Conversions among data types I

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>phylo</td>
<td>phylo4</td>
<td><code>as(x, &quot;phylo4&quot;)</code></td>
<td>phylobase</td>
</tr>
<tr>
<td>phylo</td>
<td>matching</td>
<td><code>as.matching(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>treeshape</td>
<td><code>as.treeshape(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>hclust</td>
<td><code>as.hclust(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>prop.part</td>
<td><code>prop.part(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>splits</td>
<td><code>as.splits(x)</code></td>
<td>phangorn</td>
</tr>
<tr>
<td>phylo</td>
<td>evonet</td>
<td><code>evonet(x, from, to)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>network</td>
<td><code>as.network(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo</td>
<td>igraph</td>
<td><code>as.igraph(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>phylo4</td>
<td>phylo</td>
<td><code>as(x, &quot;phylo&quot;)</code></td>
<td>phylobase</td>
</tr>
<tr>
<td>matching</td>
<td>phylo</td>
<td><code>as.phylo(x)</code></td>
<td>ape</td>
</tr>
</tbody>
</table>
### Conversions among data types II

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>treeshape</td>
<td>phylo</td>
<td>as.phylo(x)</td>
<td>apTreeshape</td>
</tr>
<tr>
<td>splits</td>
<td>phylo</td>
<td>as.phylo(x)</td>
<td>phangorn</td>
</tr>
<tr>
<td>splits</td>
<td>networx</td>
<td>as.networx(x)</td>
<td>phangorn</td>
</tr>
<tr>
<td>evonet</td>
<td>phylo</td>
<td>as.phylo(x)</td>
<td>ape</td>
</tr>
<tr>
<td>evonet</td>
<td>networx</td>
<td>as.networx(x)</td>
<td>ape</td>
</tr>
<tr>
<td>evonet</td>
<td>network</td>
<td>as.network(x)</td>
<td>ape</td>
</tr>
<tr>
<td>evonet</td>
<td>igraph</td>
<td>as.igraph(x)</td>
<td>ape</td>
</tr>
<tr>
<td>haploNet</td>
<td>network</td>
<td>as.network(x)</td>
<td>pegas</td>
</tr>
<tr>
<td>haploNet</td>
<td>igraph</td>
<td>as.igraph(x)</td>
<td>pegas</td>
</tr>
<tr>
<td>hclust</td>
<td>phylo</td>
<td>as.phylo(x)</td>
<td>ape</td>
</tr>
<tr>
<td>hclust</td>
<td>dendrogram</td>
<td>as.dendrogram(x)</td>
<td>stats</td>
</tr>
</tbody>
</table>
## Conversions among data types III

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNAbin</td>
<td>character</td>
<td><code>as.character(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>DNAbin</td>
<td>alignment</td>
<td><code>as.alignment(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>DNAbin</td>
<td>phyDat</td>
<td><code>as.phyDat(x)</code></td>
<td>phangorn</td>
</tr>
<tr>
<td>DNAbin</td>
<td>genind</td>
<td><code>DNAbin2genind(x)</code></td>
<td>adegenet</td>
</tr>
<tr>
<td>character</td>
<td>DNAbin</td>
<td><code>as.DNAbin(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>character</td>
<td>loci</td>
<td><code>as.loci(x)</code></td>
<td>pegas</td>
</tr>
<tr>
<td>alignment</td>
<td>DNAbin</td>
<td><code>as.DNAbin(x)</code></td>
<td>ape</td>
</tr>
<tr>
<td>alignment</td>
<td>phyDat</td>
<td><code>as.phyDat(x)</code></td>
<td>phangorn</td>
</tr>
<tr>
<td>alignment</td>
<td>character</td>
<td><code>as.matrix(x)</code></td>
<td>seqinr</td>
</tr>
<tr>
<td>alignment</td>
<td>genind</td>
<td><code>alignment2genind(x)</code></td>
<td>adegenet</td>
</tr>
</tbody>
</table>

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
### Conversions among data types IV

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>phyDat</td>
<td>DNAbin</td>
<td>as.DNAbin(x)</td>
<td>phangorn</td>
</tr>
<tr>
<td>phyDat</td>
<td>character</td>
<td>as.character(x)</td>
<td>phangorn</td>
</tr>
<tr>
<td>loci</td>
<td>genind</td>
<td>loci2genind(x)</td>
<td>pegas</td>
</tr>
<tr>
<td>loci</td>
<td>data frame</td>
<td>class(x) &lt;- &quot;data.frame&quot;</td>
<td>—</td>
</tr>
<tr>
<td>genind</td>
<td>loci</td>
<td>genind2loci(x)</td>
<td>pegas</td>
</tr>
<tr>
<td>genind</td>
<td>genpop</td>
<td>genind2genpop(x)</td>
<td>adegenet</td>
</tr>
<tr>
<td>genind</td>
<td>df</td>
<td>genind2df(x)</td>
<td>adegenet</td>
</tr>
<tr>
<td>data frame</td>
<td>phyDat</td>
<td>as.phyDat(x)</td>
<td>phangorn</td>
</tr>
<tr>
<td>data frame</td>
<td>loci</td>
<td>as.loci(x)</td>
<td>pegas</td>
</tr>
<tr>
<td>data frame</td>
<td>genind</td>
<td>df2genind(x)</td>
<td>adegenet</td>
</tr>
<tr>
<td>matrix</td>
<td>phyDat</td>
<td>as.phyDat(x)</td>
<td>phangorn</td>
</tr>
</tbody>
</table>

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Conversions among data types V

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Command</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcfR</td>
<td>chromR</td>
<td>vcfR2chromR(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>genind</td>
<td>vcfR2genind(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>migrate</td>
<td>vcfR2migrate(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>loci</td>
<td>vcfR2loci(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>tidy</td>
<td>vcfR2tidy(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>DNAbin</td>
<td>vcfR2DNAbin(x)</td>
<td>vcfR</td>
</tr>
<tr>
<td>vcfR</td>
<td>genlight</td>
<td>vcfR2genlight(x)</td>
<td>vcfR</td>
</tr>
</tbody>
</table>

Vojtěch Zeisek (https://trapa.cz/)
Basic operations with data I

R doesn’t ask neither notifies when overwriting objects! Be careful!

```r
x <- c(5, 6, 7, 8, 9)  # Creates vector (see also ?rep)
x # Print "x" content

# Is generic function to concatenate objects into new one

length(x)  # Length of the object - for matrices and DF use dim()

str(x) # Information about structure of the object

mode(x) # Gets type of storage mode of the object

class(x) # Shows class of the object

x[2] # Shows second element of the object

x <- x[-5] # Removes fifth element

y <- matrix(data=5:20, nrow=4, ncol=4)  # Creates a matrix

is.matrix(y) # Is it matrix? Try is.<TAB><TAB>

# TAB key shows available functions and objects starting by typed text

y # Prints the matrix

dim(y) # Dimensions of "y"

y[,2] # Prints second column
```
Basic operations with data II

1. `y[3,]` # Prints third row
2. `y[4,3]` # Prints element from fourth row and third column
3. `c(x, y[4,])` # Concatenate "x" and 4th row of "y"
4. `x2 <- c(x, 3, 2, 1)` # Concatenate "x" and values "3", "2" and "1"
5. `x <- y[2,]` # Replaces "x" by second row of "y" (no warning)
6. `rm(x)` # Deletes x (no warning)
7. `y[,1:3]` # Prints first through third column of the matrix
8. `y[3,] <- rep(x=20, each=4)` # Replaces third line by value of 20
9. `y[y==20] <- 10` # If value of y's element is 20, replace it by 10
10. `summary(y)` # Basic statistics - according to columns
11. `colnames(y) <- c("A", "B", "C", "D")` # Set column names
12. # Objects and functions are without quotation marks; files and text with
13. `colnames(y)` # Prints column names, use rownames() in very same way
14. `y` # See modified object
15. `y,"C"]` # Prints column C (R is case sensitive!)
16. `y[,c("C", "B")]]` # Extract columns "C" and "B"
1. `t(y)` # Transposes the matrix
2. `diag(y)` # Get diagonal of the matrix
3. # Replace diagonal by repetition of values 50 and 100
4. `diag(y) <- rep(x=c(50, 100), times=2)`
5. `y` # See modified object
6. `y <- as.data.frame(y)` # Turns into DF (see other functions as.*)
7. `class(y)` # Is it data frame now?
8. `y[y==17] <- "NA"` # Removes values of 17 (NA = not available = missing)
9. `y$B` # Gets variable B of data frame y ($ works similarly in S3 objects)
10. # When loading saved project, you have to load again libraries and
11. # scripts (see further), data objects are restored
12. # This can be conveniently done in RStudio/RKward
13. `save(list=ls(), file="test.RData")` # Saves all objects during the work
14. `load("test.RData")` # Loads saved R environment with all objects
15. `fix(y)` # Use to edit matrices, data frames, functions, ...
16. `rm(y)` # Removing...

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022
Some basic statistics

```
summary(iris) # Basic summary statistics
# Boxplot comparing sepal lengths of the three species
boxplot(formula = iris$Sepal.Length ~ iris$Species)
setosa <- iris[which(iris$Species == "setosa"),] # Extract only setosa
# Testing correlation between sepal length and width of setosa
cor.test(x = setosa$Sepal.Length, y = setosa$Sepal.Width)
# Plot correlation between sepal length and width of setosa
plot(x = setosa$Sepal.Length, y = setosa$Sepal.Width, main = "Sepals",
     xlab = "Length", ylab = "Width", pch = 16, cex = 1.5, col = "red")
# Add linear model line
abline(reg = lm(setosa$Sepal.Width ~ setosa$Sepal.Length), col = "brown", lwd = 3)
# Add sepal width mean and standard deviation
abline(h = mean(setosa$Sepal.Width), col = "green", lwd = 2, lty = 2)
abline(h = mean(setosa$Sepal.Width) + sd(setosa$Sepal.Width), col = "green",
       lwd = 2, lty = 3)
abline(h = mean(setosa$Sepal.Width) - sd(setosa$Sepal.Width), col = "green",
       lwd = 2, lty = 3)
```
Figures from previous basic statistics
Analysis of Variance (ANOVA)

• **ANOVA** tells us if there is significant difference among means of samples

```r
aov(formula = iris[['Sepal.Length']] ~ iris[['Species']]) # Output:
Call:
aov(formula = iris[['Sepal.Length']] ~ iris[['Species']])
Terms:
  iris[['Species']] Residuals
Sum of Squares       63.21213  38.95620
Deg. of Freedom      2            147
Residual standard error: 0.5147894
Estimated effects may be unbalanced
summary(aov(formula = iris[['Sepal.Length']] ~ iris[['Species']])) # Output:
  Df  Sum Sq Mean Sq   F value  Pr(>F)
iris[['Species']]     2   63.21   31.606   119.3  <2e-16 ***
Residuals            147  38.96    0.265
---
Signif. codes:  0 ‘***’  0.001 ‘**’  0.01 ‘*’  0.05 ‘.’  0.1 ‘ ’  1
```
Practice basic operations I

Tasks I

1. Load R training dataset `iris` (data(iris)) and read about it (?iris).
2. Explore it — print it, display only beginning of data (head()). How many rows and columns does it have? Which variables? Which species (levels())?
3. Extract from `iris` dataset only sepal lengths and save it as new vector.
4. Extract from `iris` dataset petal length and species and save it as new data frame.
6. Read help of `rep` function and
   1. Create repetition of five times value of “A”.
   2. Create repetition of five times vector of “1, 3”.

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022
Practice basic operations II

Tasks II

1. Read help of `seq` function and create sequence from 5 to 20 with steps of 0.5 and save it as new vector.
2. Extract from that vector 2\textsuperscript{nd}, 4\textsuperscript{th} and 8\textsuperscript{th} position and save it as new vector.
3. Remove from that vector 2\textsuperscript{nd} element.
4. Concatenate previous two vectors into new vector.
6. Try similar tasks as on slide 46 with another species and characters. Try to improve the plots.
Packages, repositories and their management

- Standalone plain R doesn’t have enough tools for most of scientific disciplines — only basic methods and tools for programmers, including for package management
- Users/developers contribute by making extra packages extending computational possibilities — one of biggest R advantages — it then has unlimited possibilities
- R has infrastructure for maintaining (for developers) and installing (for users) packages — the CRAN repository
- For various reasons, some people build their own infrastructures to maintain and install R packages — compatible with R, but separated
- User has basically two options
  1. Set all repositories in R and use basic commands to install packages (slide 54)
  2. Specify non-CRAN repository every time installing from it (e.g. slide 60) or use special tools (e.g. for Bioconductor — slide 59)
Repositories

- Repositories (internet directories full of R packages — slide 28) can be set via `options(repos=c(…))` or as `repos` parameter for each `install.packages(…)` command (slide 54 and onward)

- Repositories don’t have to be set as global options, e.g. Bioconductor (slide 59) has its own way to manage packages

- Similar concepts as app stores of Android, iOS, etc.

Installation of packages in GUI

- **RStudio**: set repositories by command from slide 54 and in bottom right pane select Packages and click on Install Packages...

- **RKWard**: go to menu Settings | Configure ’RKWard’ and select R-Packages. Add URLs of repositories from slide 54. OK. Go to menu Settings | Manage R packages and plugins..., click to Install..., select and install desired packages...
Set repositories

# Basic package installation
install.packages("PackageName") # Case sensitive!
?install.packages # Shows all available parameters (options)
getOption("repos") # Shows actual repositories
options() # Generic function to modify various settings
?options # Gives details

- Keep newest version of R and and newest versions of packages!
- Installation of multiple packages may sometimes fail — install then packages in smaller groups or one by one — check output and examine why installation failed — commonly due to missing external dependency (read installation output and look for notes about missing libraries, etc.)
- Avoid mixing of several R versions
- After upgrade of R (e.g. from 3.6 to 4.0), user **must** reinstall all packages.

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Install packages

- If repositories from slide 54 are not set, it is possible to install in several steps packages from main repository (CRAN) and from another sources (following slides)
- This is the basic and default the most common usage
- After upgrade of R (e.g. from 3.6 to 4.0), all packages must be reinstalled

```r
# Simplest usage
install.packages("PackageName")  # Case sensitive!

?install.packages  # See for more options
install.packages(pkgs=c("pkg1", "pkg2", "pkg3", ...),
                 dependencies="Imports")

# Installed packages are "inactive" - the must by loaded to use them:
library(PackageName)  # Loads installed package (we will do it on the fly)

# Updates installed packages (by default from CRAN)
update.packages(ask=FALSE)
```
# Install packages needed for the course

```
# Install packages. Installation of multiple packages may sometimes fail -
# install then packages in smaller groups or one by one
install.packages(pkgs=c("ade4", "adegenet", "adegraphics", "adephylo",
                         "akima", "ape", "BiocManager", "caper", "corrplot", "devtools",
                         "adespatial", "gee", "geiger", "ggplot2", "gplots", "hierfstat", "ips",
                         "kdetrees", "lattice", "mapdata", "mapplots", "mapproj", "maps",
                         "maptools", "nlme", "PBSmapping", "pegas", "permute", "phangorn",
                         "philentropy", "phylobase", "rentrez", "phytools", "picante", "plotrix",
                         "popprr", "raster", "rgdal", "RgoogleMaps", "Rmpi", "rworldmap",
                         "rworldxtra", "seqinr", "shapefiles", "snow", "sos", "sp", "spdep",
                         "splancs", "StAMPP", "TeachingDemos", "tripack", "vcfR", "vegan"),
                         repos="https://mirrors.nic.cz/R/", dependencies="Imports")
update.packages(ask=FALSE)  # Regularly update installed packages
# Upgrade all packages e.g. from R 3.6 to 4.0
install.packages(pkgs=installed.packages())
```
Install Geneland package

- Since version 4, not in CRAN anymore, check its manual, GitHub and homepage https://i-pri.org/special/Biostatistics/Software/Geneland/
- On Windows install Rtools first to be able to compile source package on Windows

```r
# Other packages used when using Geneland
# Needed is PBSmapping or mapproj for conversion of coordinates
# GUI uses for parallelisation snow and Rmpi
# RgoogleMaps (requires rgdal) can be used to plot output on top of Google map, maptools, shapefiles (requires foreign) and tripack on GIS layer

# Install Geneland from GitHub
# Devtools package is required to install from GitHub
if(! "devtools" %in% installed.packages()) {install.packages("devtools")}
# Install Geneland itself
devtools::install_github("gilles-guillot/Geneland")
```
Install phyloch package

Example of installation of package not available in any repository

- Check [http://www.christophheibl.de/Rpackages.html](http://www.christophheibl.de/Rpackages.html)
- Package phyloch is similar to ips from same author (but some functions behave differently) — both are great for usage of external applications within R, ips seems to develop more and phyloch will probably be deprecated...

```r
# If not done already, install required packages first
install.packages(pkgs=c("ape", "colorsace", "XML"),
               dependencies="Imports")
# It is possible to specify direct path
# Local or web URL - be careful about correct path) to package source
install.packages(pkgs="http://www.christophheibl.de/phyloch_1.5-3.tar.gz",
                repos=NULL, type="source")
```
Bioconductor

- Tools for analysis of genomic data, see https://bioconductor.org/
- To install it use Bioconductor’s special helper package
- Explore available packages: https://bioconductor.org/packages/release/BiocViews.html

```r
# Install CRAN package BiocManager used to manage Bioconductor packages
if (!requireNamespace("BiocManager")) install.packages("BiocManager")

?BiocManager::install # See options
# Install Bioconductor packages
BiocManager::install("muscle") # Simplest usage
BiocManager::install(pkgs=c("Biostrings", "muscle"))
# Update installed packages
BiocManager::install()
# Search for Bioconductor packages
BiocManager::available() # List everything

?BiocManager::available # See options
```
Bioconductor and others — differences from another repositories

Bioconductor has its own installation method, little comparison

# Standard installation
install.packages(c("adegenet", "poppr", "phytools"))
update.packages() # Update packages

# Installation from custom repository (ParallelStructure is not used here)
install.packages(pkgs="ParallelStructure",
repos="https://r-forge.r-project.org/"
)
?install.packages # See help for details

# Install CRAN package BiocManager used to manage Bioconductor packages
if (!requireNamespace("BiocManager")) install.packages("BiocManager")

# Install Bioconductor packages
BiocManager::install("muscle") # Simplest usage
BiocManager::install(pkgs=c("Biostrings", "muscle"))
?BiocManager::install # See more options
BiocManager::install() # Update installed packages
Non-R software I

- We use several software packages outside R
  - R functions can use this software
  - External software can be used (depending on R package) to create/modify R object, or just as different method for (batch) usage of the software (similarly to BASH, Python, etc.)
  - User must install this software manually
- Clustal (W/X; Omega is not used in the course) [http://clustal.org/](http://clustal.org/)
  - Aligner of sequences (from slide 130)
- Gdal
  - Recommended for conversion of coordinates for Geneland (from slide 256), loading of SHP files, etc.
  - Optional for Windows users, Linux users should use `gdal` and respective development (`-dev(el)`) packages
- GIMP [https://www.gimp.org/](https://www.gimp.org/)
Non-R software II

- Image manipulation, free open-source alternative to products of Adobe or Corel
- Optional, one of possibilities to view and edit output graphics from R

- **Inkscape** [https://inkscape.org/](https://inkscape.org/)
  - Vector drawing, free open-source alternative to products of Adobe or Corel
  - Optional, one of possibilities to view and edit output graphics from R

- **MAFFT** [https://mafft.cbrc.jp/alignment/software/](https://mafft.cbrc.jp/alignment/software/)
  - Aligner of sequences (from slide 130)

- **MPI** [http://fisher.stats.uwo.ca/faculty/yu/Rmpi/](http://fisher.stats.uwo.ca/faculty/yu/Rmpi/)
  - Library for parallelisation used by **Rmpi** package (optional, recommended especially on Windows; macOS and Linux have more options for parallelisation)
  - If it is not available, respective function can sometimes use different parallelisation backend or user can turn off parallelisation for respective function

- **MUSCLE** [https://www.drive5.com/muscle/](https://www.drive5.com/muscle/)
  - Aligner of sequences (from slide 130)

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Non-R software III

- **proj** [https://proj.org/download.html](https://proj.org/download.html)
  - Required for conversion of coordinates for Geneland (from slide 256)
  - Optional for Windows users, Linux users should use **proj** and respective development (-dev(e1)) packages

- **Rtools** [https://cran.r-project.org/bin/windows/Rtools/](https://cran.r-project.org/bin/windows/Rtools/)
  - On Windows only — to be able to compile source packages on Windows
Our data

Import and export of data we will use through the course, data types

3 Data

Overview of data and data types
Microsatellites
AFLP
Notes about data
DNA sequences, SNP
VCF
Export
Tasks
Brief overview of molecular data types I

Sorted with respect to usage in R

- **Isozymes** — forms of proteins differing in electrophoresis by their weight and/or charge
  - Typically coded as presence/absence (1/0) data
  - Old fashioned, but same mathematical tools can be used to analyze any presence/absence (1/0) data matrices

- **Fragmentation data** — length polymorphism
  - Codominant data — e.g. **microsatellites** (SSRs — Simple Sequence Repeats)
    - Variability in number of short (usually 1–3 bp) oligonucleotide repeats (ATAT vs. ATATAT, typically ca. 25–250 repeats) bordered by unique primer sequences
    - Very variable, fast evolving, species-specific primers needed
    - Mainly for population genetics, relationships among closely related species
    - Similarly ISSRs (Inter Simple Sequence Repeats)
  - Presence/absence (1/0) dominant data
    - The allele is or is not present — it is impossible to distinguish heterozygots from dominant homozygots
Brief overview of molecular data types II

Sorted with respect to usage in R

- **AFLP** (Amplified Fragment Length Polymorphism) — very variable, technically complicated, nowadays bit expensive and outdated
- Simpler methods **RAPD** (Random Amplified Polymorphic DNA) and **PCR-RFLP** (Polymerase Chain Reaction-Restriction Fragment Length Polymorphism) are not used anymore at all

- **Protein sequences** — not used in the course
  - Apart similar usage as with DNA/RNA (sequence analysis) it is possible to work with the structure and conformation of the proteins
  - R (especially Bioconductor) has plenty of packages for specialized protein analysis and more

- **Nucleic acid sequences** (in nearly any form) — DNA or RNA
  - From “classical” Sanger sequencing — long individual reads (of single/few genes)
  - From “modern” HTS (NGS) — 454 pyrosequencing, Illumina, ... methods
  - Probably most used are **RADseq** scanning whole genome, **HybSeq** and other target enrichment methods using specific probes to sequence only single/low-copy nuclear markers, **Genome Skimming** getting the most abundant part of the genome (plastid and mitochondrial sequences and ITS1-5.8S rRNA-ITS2 region), **Genotyping by sequencing (GBS)**; and their variants
Brief overview of molecular data types III

Sorted with respect to usage in R

- There are special tools to process raw data from the machines — not part of the course
- Modern methods are quickly developing and able to produce $\sim 10^{12}$ bp per run and multiplex many individuals
  - Whole sequences (probes/loci or longer assembled regions) or **SNPs** (Single Nucleotide Polymorphism — only polymorphic sites are retained)
- Most of methods are mathematically well defined for haploids and/or diploids, higher ploidies or mixing of ploidies is not always possible
- Most of methods shown in the course work with more data types — not every variant is shown
  - Explore more options yourselves
- For details about the molecular markers check specialized course like **Use of molecular markers in plant systematics and population biology** (česky)
R training data

- R packages commonly contain training data to illustrate its abilities
- We will use some of them during the course (we already used `iris`)
- We will also use data provided by the teacher and/or his colleagues

```r
data() # List data available in currently loaded packages
# List data available in all installed packages (can be very long)
data(package=rownames(installed.packages()))
# Load selected data set; for example phylogeny and species traits
# of shorebirds from package caper (we will use it much later)
data(shorebird, package="caper")
# Optional method (load respective library and then data)
library(caper) # Library containing (among others) desired data
data(shorebird) # Loading the data
?shorebird # See content of the dataset # or ?caper::shorebird
?adegenet::microbov
?adegenet::rupica
?ape::carnivora
```
Our data... I
We will use...

- Modified subset of diploid microsatellite data of *Taraxacum haussknechtii* (Asteraceae) from Macedonia by Zeisek et al 2015
  - Population genetics — genetic structure, characteristics and comparisons of sampled populations
  - Spatial genetics — genetic relationships in spatial context — relationships among populations regarding their spatial position

  - Population genetics
  - Species delimitation (genotyping, identification and assignment of species)

- Small subset of population AFLP data of *Cardamine amara* (Brassicaceae) from Europe by Karol Marhold and his team
  - Population genetics
Our data... II
We will use...

- Small subset of non-synonymous SNPs from ASY3 gene (required for meiosis) from diploids and tetraploids of *Arabidopsis arenosa* (Brassicaceae) from central and northern Europe by Magdalena Bohutínská and her colleagues
  - Population genomics
  - Associations of various traits (physiological, ...) with particular genetic loci/alleles
- Small subset of trees and taxa from phylogeny of *Oxalis* spp. (Oxalidaceae) from South Africa (the Cape region) by Schmickl et al. 2016
  - Phylogenomics, target enrichment (HybSeq) sequencing of multiple genes — construction of individual gene trees, their evaluation (identification of trees with significantly different topology — genes with different evolutionary pathway) and construction of species trees and networks
- Internal transcribed spacer sequences of *Gunnera* spp. (Gunneraceae) from Wanntorp et al. 2014 downloaded from GenBank
Our data... III
We will use...

- Phylogenetic relationships among species with “Gondwana” distribution (south of South America, South Africa, Australia, New Zealand, ...)

- Maturase K (matK) plastid sequences of *Nothofagus* (Nothofagaceae) downloaded from EMBL-EBI and/or NCBI (various authors; mainly from phylogenetic studies)

- Training data from packages
  - Sequence of influenza from USA sampled in several years (package adegenet, part of its vignette)
  - SSRs genotypes of cattle breeds, `?adegenet::microbov` — population genetics (genetic structure)
  - *Rupicapra rupicapra* (Bovidae) SSRs genotypes from French Bauges mountains, `?adegenet::rupica` — spatial genetics (spatial distribution of genotypes and their relationships)
Our data... IV
We will use...

- Morphological traits and phylogeny of 17 *Acer* species, `?adephylo::maples` — phylogeny and correlated evolution of characters
- Morphological traits of Carnivora, `?ape::carnivora` — life history patterns: allometric, phylogenetic and ecological associations
- Measurements of *Iris setosa*, *I. versicolor* and *I. virginica*, `?iris` — measurements of morphological characters, their correlations and differences among species
- Phylogeny and morphological traits of shorebirds, `?caper::shorebird` — comparative evolution of characters

Work with microsatellites is in most cases (except some methods taking advance from microsatellite mutational nature) same as with presence-absence data and methods can handle both data types in nearly same fashion
- Examples are shown mainly with microsatellites, but another (sequencing, presence/absence, ...) data are used in same way — try it
**Our data... V**

We will use...

- Distance-based methods are same regardless input data on the beginning (microsatellites, AFLP, DNA sequences, ...)
- Extraction of SNP from DNA is useful in case of huge data sets — for smaller data sets it is not necessary
- Many methods can process (nearly) any input data type

**Always save your work!**

We will use data objects during whole course — all the time save your workspace! Use possibilities of your GUI or save / load functions.
Input/output data formats I

Representative selection

- **BAM** (Binary Alignment Map)
  - Sequences (usually short from Illumina) mapped to reference
  - Each file contains data for single sample
  - Contains also information about mapping quality etc., complex structure
  - Compressed version of SAM (see below)
  - Special applications are needed to work with BAM files
  - Extension *.bam

- **CSV** (Comma Separated Values)
  - “One sheet of Excel”
  - Common format to store data (traits, coordinates, ...), similar to TSV (see below)
  - Columns (cells) are separated by commas, cells are commonly bordered by quotation marks — it is important to check structure before import into R (and verify it after import)
  - Extension usually *.csv
  - Can have many formatting forms — carefully set parameter of import function
Input/output data formats II

Representative selection

- **FASTA**
  - Simple and popular text format to store genetic sequences (DNA, RNA, proteins)
  - Each file contains one or more sequences
  - **Line 1** of every records starts with `>` and contain name/description of the sequence (e.g. `> Seq 1`), **line 2+** contain(s) the sequence (`ATCG...`) — until end of file or next line starting with `>`
  - Each sequence can be on single line, or on multiple lines
  - Can store also alignments (practically sequences of same length, with marked gaps and missing data)
  - Larger sequences are sometimes compressed (mostly by gzip — `*.gz`)
  - Extension usually `*.fasta` (generic, also `*.fas`, `*.fa`, `*.seq`, `*.fsa`, `*.fna` (nucleic acid), `*.ffn` (nucleotides, coding regions), `*.faa` (amino acids), `*.frn` (non-coding RNA), ...
Input/output data formats III

Representative selection

- **FASTQ**
  - Text based format to store sequences (mainly nucleotides)
  - Every record consists of **4 lines**: (1) sequence ID (with possible description) starting with @,
    (2) the sequence (ATCG...), (3) + optionally followed by the same ID as line 1, and (4) quality values for nucleotides from line 2
  - Probably the most common format for output of modern high throughput sequencing machines (e.g. Illumina), each file contains huge number of sequences
  - Commonly compressed by gzip (*.gz), sometimes by other compression methods
  - Extension usually *.fastq, *.fq, *.fastq.gz, *.fq.gz, ...

- **NEWICK**
  - Every line contains one tree represented by brackets, optionally with numbers (separated by : ) labeling nodes and/or branches (bootstrap supports, likelihoods, branch lengths, ages, ...)
  - File can contain one or more trees
Representative selection

- E.g. \((A, B, (C, D)E)F; \) or \((A:0.1, B:0.2, (C:0.3, D:0.4):0.5); \)
- Simple logic, but very hard to read by human
- Extension usually \(^{\text{.newick, .nwk, .tre, ...}}\)

**NEXUS**

- Popular plain text format used by software like Mesquite, MrBayes, PAUP*, SplitsTree, ...
- Structure can be complex, is divided into blocks containing e.g. sequences, trees (in NEWICK format), distance matrix, fragmentation data, networks (e.g. for SplitsTree), MrBayes commands, traits, ...
- Several variants, sometimes problems with interoperability
- Extension usually \(^{\text{.nexus, .nex, .nxs}}\)

**SAM (Sequence Alignment Map)**

- Text-based format for storing biological sequences aligned to a reference sequence
- Each file contains data for one sample
Input/output data formats V

Representative selection

- Structure is relatively complex
- Used by applications like bamtools or SAMtools
- Extension usually *.sam

**TSV** (TAB separated values)
- "One sheet of Excel"
- Common format to store data (traits, coordinates, ...), plain text, similar to CSV
- Columns (cells) are separated by tabs (\t) — it is important to check structure before import into R (and verify it after import)
- Extension usually *.tsv, *.tab

**TXT**
- Plain text file can contain content of all listed file formats (except binary BAM) — extension (*.txt) is not really reliable...
- Technically, all listed plain text formats belong also to this category
Input/output data formats VI

Representative selection

- **VCF** (Variant Call Format)
  - Do not confuse with vCard (*.vcf, *.vcard) storing virtual business cards and address books
  - Bioinformatics plain text format storing gene variants, annotations, quality data and more information
  - Used by software like Bcftools, GATK, Picard, VCFtools, ...
  - Complex structure, sequences are not stored as in FASTA, but as SNP variants on respective positions — useful to store processed NGS/HTS data (e.g. from Illumina machines)
  - Commonly compressed by gzip (*.gz), sometimes by other compression methods
  - Several versions and variants (including binary BCF, *.bcf), sometimes there are problems with interoperability
  - Extension usually *.vcf or *.vcf.gz
Examples of data and formats I — AFLP (presence/absence) gel and microsatellites from sequencing machine
Examples of data and formats II — Aligned DNA sequences displayed in Geneious
Examples of data and formats III — FASTA and FASTQ sequences in text view

> CY013200
atgaagactatcattgctttgagctacattttatgtctggtttttcgctcaaaacttccccctagtgaaaaca
ggaatgacaacacacagcacaacgctgtgcttgacaccatgcagtgcagccaaacgcaggtacagatgat
> CY013781
atgaagactatcattgctttgagctacattttatgtctggtttttcgctcaaaacttccccaaattgaagtg

... # FASTA continues...

NCGATTCTTTTAGCAATTAGACGTGAAAGGTCTCTTGATGAAAGACACTAACGAACTCTTTCCTTGGACACC+
#<<BBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
TCGATTCACTGAACTGAATGTCCGACAAACTTTAGTTTGTCGTTTCTACCTCACAAAGTTCGGAGCTTCGA+

... # FASTQ continues...
Examples of data and formats IV — BAM displayed in IGV
It contains Illumina short reads mapped to reference
Examples of data and formats V — VCF with multiple samples displayed in IGV

Variants of alleles and depth of coverage for each sample, mapped to reference

Vojtěch Zeisek (https://trapa.cz/)
Examples of data and formats VI — Parts of VCF in text view

```plaintext
... # Header
##ALT=<ID=NON_REF,Description="Represents any possible alternative al...
##FILTER=<ID=DP_4,Description="DP < 4">
... # Information about data stored
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in geno...
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for ...
... # Information about chromosomes etc.
##contig=<ID=scaffold_1,length=33132539>
##contig=<ID=scaffold_2,length=19320864>
... # The data (variants of nucleotides)
#CHROM  POS   ID     REF   ALT  QUAL  FILTER INFO FORMA...
scaffold_4 22846289 .     C     A    91.52 PASS ...
scaffold_4 22846291 .     C     T    325.58 PASS ...
...AC=1;AF=1.057e-03;AN=946;BaseQRankSum=0.825;ClippingRankSum=0.118;...
...DP=4046;ExcessHet=3.0302;FS=0.000;InbreedingCoeff=-0.0096;MQ=79.21...
... # More data...
```
Notes about paths to import the data

- Generally, R can accept nearly any local or web location
- If unsure where you are, open any file manager, go to the R working directory (verify with `getwd()` and `dir()`) and verify where everything is
- Local paths (within one computer) can be absolute or relative
  - Absolute paths start from the top of files hierarchy: on UNIX (Linux, macOS, ...) it use to look like `/home/USER/...`, on Windows like `C:/...` (e.g. `FileParameter="/path/to/some/file.txt"`) 
  - Relative paths start in current directory (so no with `/` or `C:`)
Notes about paths to import the data II

• In the easiest case the input file is in same directory as is R’s working directory — verify by 
getwd() and dir() — you then need to specify only the filename (e.g.

FileParameter="SomeFile.txt"")

• For subdirectory start with its name (no with / or C: ), e.g.

FileParameter="subdirectory/another/directory/file.txt"

• When going directory up, use one .. for each level, e.g.

FileParameter="../upper/directory/file.txt"

• On UNIX (macOS, Linux, ...) tilde ~ means user’s home directory (e.g. /home/USER/), so

FileParameter="~/some/file.txt" is same as

FileParameter="/home/USER/some/file.txt"

• If loading data from computer, carefully check the paths or use function

file.choose() to interactively pick up the file anywhere in the computer — it can replace nearly any filename parameter (e.g. FileParameter=file.choose() )
Notes about paths to import the data III

- Some R functions have problems with spaces and special (non-alphanumeric and accented) characters — Avoid them!

- One of the most common source of errors — **when the command fails, double check paths** (and Internet connection, if applicable) — for another common problems see slide 385

---

**Working in dedicated directory**

R always work in some directory (see by `getwd()` ) and by default load input files from there (see them by `dir()` ) and save output there — relative paths starts there. This is common source of confusion and errors for beginners.
Population genetics and phylogenetics in R

Microsatellites, AFLP, SNP & sequences

- Now we will use mainly packages `adegenet` and `poppr`
- Other important genetic packages: `ape`, `ade4` and `pegas`
- Dominant/co-dominant marker data of any ploidy level including SSRs, SNP, and AFLP are analyzed in same way
- Most of methods are available for polyploids (although not all)
- Some methods are unavailable for dominant (presence/absence) data
- Mixing of ploidy levels is tricky (but possible) — it doesn’t matter when data are encoded as PA, otherwise it is mathematically problematic
- Import, basic checking, manipulation, conversion, and export of basic common data types
- Analyses are in further chapters...

```r
# Load needed libraries
library(ape)
library(ade4)
library(adegenet)
library(pegas)
library(poppr)
#
# Now let's start to work...
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Microsatellites

- Microsatellites are short (1–3 bp) tandem repeats (in plants typically AT) very commonly occurring in Eucaryotic genomes
- Defined by number of repeats (usually ca. 25–250×), usually recorded as length of whole region
- On the beginning and end of microsatellite region there is unique primer sequence (ca. 20 bp) — unique primers must be designed for each species
- Generally considered as evolutionary neutral, but sometimes associated with gene expression regulations or diseases
- Commonly used in population genetics due to their very high mutation rate (high diversity) and possibility to distinguish heterozygots from dominant homozygots
- Suitable for fine-scale population genetics, relationships among closely related species, not for phylogeny
- Cheap, easy to sequence
Load microsatellite data

# Source data

pop msta93 msta101 msta102 msta103 msta105 msta131 ...
H01 He 269/269 198/198 221/223 419/419 197/197 196/196 ...
H02 He 275/283 198/198 221/223 419/419 193/193 168/190 ...
...

# Loading the data

# Load training data (Taraxacum haussknechtii from Macedonia)

hauss.loci <- read.loci(file = "https://soubory.trapa.cz/rcourse/haussknechtii_ssrs.txt", header=TRUE, loci.sep="\t", allele.sep="/",
col.pop=2, col.loci=3:14, row.names=1) # \t means TAB key

hauss.loci # Data control

print(hauss.loci, details=TRUE)

First line starts with empty cell (if header is presented), there can be any extra column, take care about

col.loci. row.names are individual names (first column). Take care about loci.sep (here TAB \t) and allele.sep (here /) – according to data formatting.
Prepare genind object for analysis and load coordinates

```r
# Conversion of loci to genind - used for many analysis
hauss.genind <- loci2genind(hauss.loci)

# See population names
pop(hauss.genind)

# "$" separates extra slots within object
hauss.genind$pop

# Source data
Ind  lon  lat
H01  21.3333 41.1
...

# Read coordinates
hauss.coord <- read.csv("https://soubory.trapa.cz/rcourse/haussknechtii_coordinates.csv", header=TRUE, sep="\t", quote="", dec=".", row.names=1)

• Coordinates can be in any projection or scale — according to aim
• Take care about parameters of `read.csv()`! See `?read.csv`
• `pegas::geod` calculates geodesic distances (on Earth surface)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
92/421
Add coordinates to genind and create genpop object

```r
# Add coordinates - note identification of slots within object
hauss.genind$other$xy <- hauss.coord
hauss.genind$other$xy # See result - the coordinates
hauss.genind # See result - whole object

# Conversion to genpop - for population-level analysis
hauss.genpop <- genind2genpop(hauss.genind, process.other = TRUE)
hauss.genpop # See result

# Removes missing data - see ?missingno for types of dealing them
# Use with caution! It modifies original data!
hauss.genind.cor <- missingno(pop = hauss.genind, type = "mean", cutoff = 0.1, quiet = FALSE)

# See other options of handling missing data

# Convert corrected genind to loci
hauss.loci.cor <- genind2loci(hauss.genind.cor)

# Writes loci file to the disk
write.loci(hauss.loci.cor, file = "hauss.loci.cor.txt", loci.sep = "\t", allele.sep = "/")
```
Import existing data set from popular software

1. `read.genalex` # poppr - reads *.csv file
2. `read.fstat` # adegenet - reads *.dat files, only haploid/diploid data
3. `read.genetix` # adegenet - reads *.gtx files, only haploid/diploid data
4. `read.genepop` # adegenet - reads *.gen files, only haploid/diploid data
5. `read.structure` # adegenet - reads *.str files, only haploid/diploid data
6. `import2genind` # adegenet - more automated version of above functions

One function rules them all...

All these functions (including e.g. `read.loci()` and `read.csv()`) are only modifications of `read.table()` . You can use it directly to import any data. Look at `?read.table` and play with it. Take care about parameters. Does the table use quotes to mark cell (e.g. `quote=""`) ? How are columns separated (e.g. `sep="\t"`) ? Is there a header with names of populations/loci/whatever ( `header=T/F`) ? What is decimal separator (e.g. `dec="."`) ? Are there row names (used typically as names of individuals; e.g. `row.names=1`) ?

Always check data after import!
Import of polyploid microsatellites

- adegenet, poppr and related packages can for most of functions handle any ploidy level (including mixing of ploidy levels, but not for all analysis)
- polysat package can handle mixed ploidy levels for microsatellites, but range of methods is limited
- As for AFLP, we need two files: the data matrix and individual’s populations (it can be combined in one file — next slide)

Triploid microsatellite data:

```
1  msat58  msat31  msat78  msat61 ...
2  ala1 124/124/124 237/237/237 164/164/172 136/136/138 ...
3  ala2 124/124/124 237/237/237 164/164/172 136/136/138 ...
4  ala4 124/124/124 237/237/237 164/164/172 136/136/138 ...
5  ...   ...   ...   ...   ...
```

Triploid species of Taraxacum sect. Taraxacum

Vojtěch Zeisek (https://trapa.cz/)
How to import polyploid microsatellites

```r
# Import of table is as usual. Last column contains populations

```tarax3n.table <- read.table("https://soubory.trapa.cz/rcourse/tarax3n.txt", header=TRUE, sep="\t", quote="", row.names=1)
# Check the data
tarax3n.table
class(tarax3n.table)
dim(tarax3n.table)
# See parameter "X" - we don't import whole tarax3n.table as last column contains populations - this column we use for "pop" parameter (note different style of calling the column - just to show the possibility).
# Check "ploidy" and "ncode" (how many digits code one allele - must be same everywhere). See ?df2genind for more details.
tarax3n.genind <- df2genind(X=tarax3n.table[,1:6], sep="/", ncode=3,
  pop=tarax3n.table[["pop"]], ploidy=3, type="codom")
# See resulting genind object
tarax3n.genind
summary(tarax3n.genind)
```
Amplified Fragment Length Polymorphism

1. Whole genomic DNA is split by few restriction enzymes into huge number of fragments of various length

2. Adaptors are ligated to each fragment

3. Pre-amplification — only fragments with “selection” nucleotide, 1/16 of all fragments are amplified

4. Selective amplification — like previous step, 2 more “selection” nucleotides, 1/256 of all fragments are amplified
   - Resulting fragments are visualized and recorded as presence/absence (1/0) matrix of fragment of particular length
   - Highly variable, suitable for fine-scale population genetics, relationships among closely related species, not for phylogeny
   - Technically relatively demanding, nowadays people use to prefer RAD-Seq or similar
   - Same methods are used also for another presence/absence data

Vojtěch Zeisek (https://trapa.cz/)
### Import of presence/absence (e.g. AFLP) data — background

Two files — AFLP data with individual names, and populations

<table>
<thead>
<tr>
<th>AFLP or any other presence/absence data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

AFLP data of *Cardamine amara* group

**Individual’s populations:**

<table>
<thead>
<tr>
<th>POP</th>
<th>pop1</th>
<th>popZ</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP</td>
<td>pop1</td>
<td>popZ</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Use any names, just keep one word (no spaces) and don’t use special characters
- Keep names of loci as simple as possible, there are some issues when they contain dots
- As soon as one line of data = one individual, ploidies and their mixing doesn’t matter
- Not all methods introduced later are available/meaningful for PA
Import of AFLP data — the code

```r
amara.aflp <- read.table(file = "https://soubory.trapa.cz/rcourse/amara_aflp.txt", header = TRUE, sep = "\t", quote = "")
amara.aflp

dim(amara.aflp)
class(amara.aflp) # Must be matrix or data frame

# Populations - just one column with population names for all inds
amara.pop <- read.table(file = "https://soubory.trapa.cz/rcourse/amara_pop.txt", header = TRUE, sep = "\t", quote = "")
amara.pop

# You can use just one file, where populations are in last column and
# in df2genind() use for example X=aflp[,1:XXX] and pop=aflp[,YYY]
# Create genind object - ind.names and loc.names are taken from X
aflp.genind <- df2genind(X = amara.aflp, sep = "", ind.names = NULL,
loc.names = NULL, pop = amara.pop[, 1], type = "PA")

indNames(aflp.genind) <- amara.aflp[, 1] # Add individual names

aflp.genind

# You can add any other variables into genind$other$XXX

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
99/421
Another data manipulation

- `genind2df` # adegenet - export into data frame
- `genind2genalex` # poppr - export for genalex
- `splitcombine` # poppr - edits population hierarchy
- `popsub` # poppr - extracts only selected population(s)
- `clonecorrect` # poppr - corrects for clones
- `informloci` # poppr - removes uninformative loci
- `seppop` # adegenet - separates populations from genind or genlight
- `seploc` # adegenet - splits genind, genpop or genlight by markers
- `alleles2loci` # pegas - transforms a matrix of alleles into "loci"
  # seppop and seploc return lists of genind objects - for further
  # analysis using special functions to work on lists (see further)
  # read manuals (?...) of the functions before usage

- SNPs can be into genind imported in same way as AFLP (PA)
- `alleles2loci()` is very useful when each allele is in separated columns (not like in our case where one column contains one loci with all alleles) — saves time needed to change input file formatting
Notes about getting data into R

• When importing fragmentation, character, etc. data, we somehow use function `read.table()` — it is important to understand it.

• I recommend to use TAB (TSV — tab separated values; encoded as \t in R) to separate columns (no quotation marks, no commas).

• When importing microsatellites, all alleles must have same number of digits. Separate alleles by “/”, “|” or something similar and correctly specify it in `read.loci()` or `df2genind()` (or read the data with `read.table()`, convert into matrix and use `alleles2loci()`).

• Do not use underscores (“_”) or minuses (“-”) to name objects in R — only numbers, Latin letters or dots.

• `read.loci()` sometimes doesn’t work correctly on AFLP or polyploid microsatellites — try `read.table()` instead.

• Genind object is able to store mixed ploidy data, but not all analysis are able to handle it.
Nucleotide sequences

- DNA/RNA sequences from “traditional” Sanger sequencing
- Probably most common genetic data
- Genes of various lengths and mutation rate — from variable introns suitable for population genetics to conservative genes suitable for phylogenies
- Examples are shown for DNA, but same apply for RNA and mostly also for protein sequences (use class `AAbin` instead of `DNAbin`)
- Same methods are used for single nucleotide polymorphism extracted from larger-scale sequencing, including from modern sequencing methods
- Data use to be stored in FASTA or NEXUS format
- Outputs of modern high-throughput sequencing use to be stored in FASTQ and later BAM and VCF (see further)
# Reading FASTA (read.dna() reads also another formats, see ?read.dna)
# Sequences of flu viruses from various years from USA (Adegenet toy data)
usflu.dna <- read.dna(file = "https://adegenet.r-forge.r-project.org/files/usflu.fasta", format = "fasta")
class(usflu.dna) # Check the object

usflu.dna # Check the object

# Another possibility (only for FASTA alignments, same result):
usflu.dna2 <- fasta2DNAbin(file = "https://adegenet.r-forge.r-project.org/files/usflu.fasta") # Normally keeps only SNP - see ?fasta2DNAbin
class(usflu.dna2) # Check the object

usflu.dna2 # Check the object

as.character(usflu.dna2)[1:5, 1:10] # Check the object
dim(usflu.dna2) # Does it have correct size?

# Read annotations
usflu.annot <- read.csv("https://adegenet.r-forge.r-project.org/files/usflu.annot.csv", header = TRUE, row.names = 1)
head(usflu.annot) # See result

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022 103/421
Import of DNA sequence data II

1. Convert DNAbin to genind - only polymorphic loci (SNPs) are retained
2. When converting DNAbin to genind, the sequences must be aligned!
   ```
   usflu.genind <- DNAbin2genind(x = usflu.dna, pop = usflu.annot[["year"]])
   ```
3. Check it
4. Read sequence data in NEXUS (similar to reading FASTA)
   ```
   ?read.nexus.data
   ```

- RNA or protein sequences can be handed in same way — see `?read.dna` and `?read.FASTA`
- For nucleic acid sequences there is R class `DNAbin`, for protein sequences `AAbin` — they are handled in same way
- Do not confuse `read.nexus.data` (reads sequences) and `read.nexus` (reads trees)
Import sequences from GenBank


```r
# Importing sequences according to sequence ID
gunnera.dna <- read.GenBank(c("AF447749.1", "AF447748.1", "AF447747.1", "AF447746.1", "AF447745.1", "AF447744.1", "AF447743.1", "AF447742.1", "AF447741.1", "AF447740.1", "AF447739.1", "AF447738.1", "AF447737.1", "AF447736.1", "AF447735.1", "AF447734.1", "AF447733.1", "AF447732.1", "AF447731.1", "AF447730.1", "AF447729.1", "AF447728.1"))
gunnera.dna
class(gunnera.dna)
```

• To be converted into `genind` (useful for many population genetic analysis), sequences in `DNAbin` must be aligned (from slide 130)

• To query on-line database as through web we use `seqinr` (next slide) or `rentrez` (slide after)
library(seqinr)
choosebank()  # List genetic banks available for seqinr
choosebank("embl", timeout=20)  # Choose some bank
?query  # See how to construct the query
# Query selected database - there are a lot of possibilities
nothofagus <- query(listname="nothofagus",
    query="SP=Nothofagus AND K=rbcl", verbose=TRUE)
nothofagus$req  # See the sequences information
# Get the sequences as a list
nothofagus.sequences <- getSequence(nothofagus$req)
nothofagus.sequences  # See sequences
nothofagus.annot <- getAnnot(nothofagus[["req"]])  # Get annotations
nothofagus.annot
closebank()  # Close the bank when work is over
# Convert sequences from a list to DNAbin (functions as.DNAbin*)
nothofagus.dna <- as.DNAbin.list(nothofagus.sequences)
nothofagus.dna  # See it
Query NCBI databases

```r
library(rentrez)
entrez_dbs() # Genetic banks available for rentrez
entrez_db_summary("nucleotide") # Brief description of what the database is
# Set of search terms that can used with this database
entrez_db_searchable("nucleotide")
# Search the database and get IDs of matched records
nothofagus.search <- entrez_search(db="nucleotide",
  term="Nothofagus[ORGN] AND rbcL[GENE]"
)
nothofagus.search
# Fetch desired records according to their IDs
nothofagus.fasta <- entrez_fetch(db="nucleotide",
  id=nothofagus.search[['ids']], rettype="fasta")
nothofagus.fasta
# Conversion into DNAbin requires saving to disk as FASTA and then loading
write(x=nothofagus.fasta, file="nothofagus.fasta")
nothofagus.dna <- read.dna(file="nothofagus.fasta", format="fasta")
nothofagus.dna
```
Importing SNP

- Import from PLINK requires saving of data with option “--recodeA”

```R
read.PLINK # How to read PLINK files
```

- Extracting SNP from alignments reads FASTA alignments and keep only SNPs. The method is relatively efficient even for large data sets with several genomes:

```R
usflu.genlight <- fasta2genlight
  (file="https://adegenet.r-forge.r-project.org/files/usflu.fasta",
   quiet=FALSE, saveNbAlleles=TRUE)
usflu.genlight # See genlight
```

```R
?fasta2genlight # Function has several options to speed up reading
# If it crashes (on Windows), try to add parameter "parallel=FALSE"
```

- For small data sets, keep data as genind as it is more information-rich — genlight is more efficient for large data (> ~100,000 SNPs)

- Adegenet has custom format to store SNP as plain text file and function `read.snp` to import it into `genlight` object — check Adegenet tutorial `genomics`, `?read.snp`
Checking SNPs

```r
# Position of polymorphism within alignment - snpposi.plot requi-
# res input data in form of matrix
snpposi.plot(x = as.matrix(usflu.dna), codon = FALSE)

# Position of polymorphism within alignment-differentiating codons
snpposi.plot(as.matrix(usflu.dna))

# When converting DNAbin to genind
# only polymorphic loci are kept -
# threshold for polymorphism can
# be arbitrary (polyThres=...)
usflu.genind2 <- DNAbin2genind(x = usflu.dna, polyThres = 0.01)
usflu.genind2 # See it
```
Checking sequences

```r
# Test is distribution of SNPs is random (1000 permutations)
snpposi.test(as.matrix(usflu.dna))

pegas::nuc.div(x=usflu.dna) # Nucleotide diversity
ape::base.freq(x=usflu.dna) # Base frequencies
ape::GC.content(x=usflu.dna) # GC content

# Number of times any dimer/trimer/etc oligomers occur in a sequence
# Note: count() requires single sequence as DNAbin/character
seqinr::count(seq=as.character.DNAbin(gunnera.dna[['AF447749.1']]),
   wordsize=3)

# View sequences - all must be of the same length
# Function "image.DNAbine" requires as input matrix
# So that sequences must be of same length (aligned)
image.DNAbin(x=usflu.dna)

# Sequences must be of same length - as.matrix.DNAbin() can help
image.DNAbin(x=as.matrix.DNAbin(usflu.dna))
```
U.S. flu sequences

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

111/421
Notes about using genlight (vs. genind)

- Genlight is “just” version of more common genind object to store large data sets with (nearly) complete multiple genomes
- “Large” is tricky — there is no easy criterion (roughly, genind is inefficient since dozens or hundreds thousands of SNPs) — try genind and when work fails because of not enough computer resources, go on with genlight
- Use is basically same as when working with genind — but not all functions are able to deal with it (on the other hand, others are optimized to work well on large data sets)
- SNPbin is version of genind/genlight to store one large genome — serves basically as storage, no need to deal with it
- Genlight as well as genind allow varying ploidy level
- Functions working with genlight use to use parallelisation to speed up operations — this commonly doesn’t work properly on MS Windows
Variant Call Format

- Raw sequencing data from modern HTS (Illumina, Pacific Biosciences, ...) are stored in FASTQ format and pre-processed via BAM into VCF
- VCF is most common format for storing pre-processed data for various downstream analysis from HTS
- Can effectively store large WGS data
- Keeps only SNPs — differences of particular sample to reference
- Contains plenty of “meta information” — depth of coverage (ow many times was each site sequenced), mapping quality, ... used for quality filtering of SNPs
- Can handle any ploidy level, including mixing of ploidy levels, but not every application can work with that
Reading VCF

- Download into working directory input file
  
  https://soubory.trapa.cz/rcourse/arabidopsis.vcf.gz and reference sequence
  
  https://soubory.trapa.cz/rcourse/alygenomes.fasta

- Non-synonymous SNPs from ASY3 gene (required for meiosis) from diploids and tetraploids of Arabidopsis arenosa from central and northern Europe

- Package vcfR has functions to manipulate and explore VCF; other option is usage of VariantAnnotation

```r
library(vcfR) # Required library
# Pick up downloaded file 'arabidopsis.vcf.gz' from the disk
arabidopsis.vcf <- read.vcfR(file=file.choose())
# File choose dialog can open in background - search for it :-)
# Or directly load remote file
arabidopsis.vcf <- read.vcfR(file="https://soubory.trapa.cz/rcourse/arabidopsis.vcf.gz")
```
Checking VCF

# It returns object of class vcfR-class
?read.vcfR  # See more import options
?pegas::read.vcf  # This one returns list of objects loci and data.frame

arabidopsis.vcf
head(arabidopsis.vcf)
arabidopsis.vcf@fix[1:10, 1:5]
strwrap(arabidopsis.vcf@meta[1:21])

queryMETA(x=arabidopsis.vcf)
queryMETA(x=arabidopsis.vcf, element="DP")
queryMETA(x=arabidopsis.vcf, element="FORMAT.+DP")
queryMETA(x=arabidopsis.vcf, element="FORMAT=<ID=DP")
head(x=getFIX(x=arabidopsis.vcf))
head(x=is.polymorphic(x=arabidopsis.vcf, na.omit=TRUE))
head(x=is.biallelic(x=arabidopsis.vcf))
arabidopsis.vcf@gt[1:10, 1:4]
# See description of depth of coverage (DP) slot
strwrap(x=grep(pattern="ID=DP,"; x=arabidopsis.vcf@meta, value=TRUE))
Checking depth of coverage (DP)

```r
arabidopsis.vcf.dp <- extract.gt(x = arabidopsis.vcf, element = "DP", as.numeric = TRUE) # GT:GQ:DP:HQ
dim(arabidopsis.vcf.dp) # See it
head(arabidopsis.vcf.dp)

# Boxplot of DP
boxplot(x = arabidopsis.vcf.dp, col = "#808080", ylab = "Depth of coverage",
        las = 3)
title("DP per specimen")
abline(h = seq(from = 0, to = 90, by = 10), col = "#b3b3b3")

# Bar plot of mean DP
barplot(apply(X = arabidopsis.vcf.dp, MARGIN = 2, FUN = mean, na.rm = TRUE), las = 3)
title("Mean DP per specimen")
abline(h = seq(from = 0, to = 60, by = 10), col = "#b3b3b3")

# Heatmap of DP (subset)
heatmap.bp(x = arabidopsis.vcf.dp[1:100, 1:100], col.ramp = rainbow(n = 100, start = 0.1)) # Subset - only first 100 loci and individuals
title("DP per specimens and loci")
```
DP per specimen

[Graph showing depth of coverage per specimen]
Bar plot of mean DP

Mean DP per specimen

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

118/421
Heat map of DP

DP per specimens and loci

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

119/421
Extract the genotype quality (GQ)

```r
# Extract the GQ
arabidopsis.vcf.gq <- extract.gt(x=arabidopsis.vcf, element="GQ",
   as.numeric=TRUE)
dim(arabidopsis.vcf.gq)
arabidopsis.vcf.gq[1:10,1:10]
# Heatmap of GQ (subset)
heatmap.bp(x=arabidopsis.vcf.gq[1:100,1:100])
# Bar plot of mean GQ
barplot(apply(X=arabidopsis.vcf.gq, MARGIN=2, FUN=mean, na.rm=TRUE),
   las=3)
abline(h=seq(from=0, to=90, by=5), col="grey")
# Boxplot of GQ
boxplot(arabidopsis.vcf.gq, las=2, main="Genotype Quality (GQ)"
abline(h=seq(from=0, to=100, by=5), col="grey")
# Basically same as work with DP...
```
# Extract information about missing data
arabidopsis.vcf.miss <- apply(X=arabidopsis.vcf.dp, MARGIN=2, 
  FUN=function(x){sum(is.na(x))})
arabidopsis.vcf.miss <- arabidopsis.vcf.miss/nrow(arabidopsis.vcf)

# Bar plot of missing data
barplot(height=arabidopsis.vcf.miss, ylab="Percentage of missing data", 
  las=2)
abline(h=seq(from=0, to=1, by=0.05), col="grey")

# Histogram of frequencies of missing data
arabidopsis.vcf.missg <- apply(X=arabidopsis.vcf.dp, MARGIN=1, 
  FUN=function(x){sum(is.na(x))})
arabidopsis.vcf.missg <-
  arabidopsis.vcf.miss/ncol(arabidopsis.vcf@gt[,,-1])
hist(x=arabidopsis.vcf.missg, xlab="Missingness (%)")
abline(h=seq(from=0, to=350, by=25), col="grey")

# Set abline parameters according to your data
Remove non-biallelic loci and indels

- This is commonly done with SNPs as many downstream analysis are well defined only for biallelic loci (and do not work well with other loci)
- Other than biallelic loci are often suspicious of being laboratory/computational artefact

```r
# Remove indels
arabidopsis.vcf <- extract.indels(x=arabidopsis.vcf)

# Remove non-biallelic loci
arabidopsis.vcf <- arabidopsis.vcf[is.biallelic(x=arabidopsis.vcf),]

# See result
arabidopsis.vcf
```

- vcfR has relatively limited possibilities to filter VCF when comparing to specialized software like GATK
- More options are in VariantAnnotation, but it uses different R class, so it’s not handy for future work in this workflow
ChromR — filtration of VCF I

```r
# Loading reference sequence - download
# https://soubory.trapa.cz/rcourse/alygenomes.fasta into working directory
arabidopsis.dna <- read.dna(file="alygenomes.fasta", format="fasta")
arabidopsis.dna

# Conversion into chromosome object (ChromR)
arabidopsis.chrom <- create.chromR(vcf=arabidopsis.vcf,
    name="Arabidopsis arenosa", seq=arabidopsis.dna)
arabidopsis.chrom
plot(arabidopsis.chrom)

# Masking sites with too low/high DP and/or MQ
arabidopsis.chrom.mask <- masker(x=arabidopsis.chrom, min_QUAL=1,
    min_DP=3, max_DP=5000, min_MQ=40, max_MQ=200)
arabidopsis.chrom.mask
plot(arabidopsis.chrom.mask)
variant.table(arabidopsis.chrom.mask)

# Saving mask into new object
arabidopsis.chrom.fin <- proc.chromR(x=arabidopsis.chrom.mask)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
ChromR — filtration of VCF II

# See results from the previous slide
arabidopsis.chrom.fin
chromoqc(chrom, arabidopsis.chrom.fin)
# The plot is bit empty as we have only single gene
Convert VCF into various objects for later processing

```r
# Genind - convert chromR or vcfR objects
arabidopsis.genind <- vcfR2genind(x=arabidopsis.chrom.fin@vcf, ploidy=4)
arabidopsis.genind # Check it
nInd(arabidopsis.genind)
indNames(arabidopsis.genind)
nLoc(arabidopsis.genind)
locNames(arabidopsis.genind)

# Genlight (suitable for huge data, not required now)
# Note that it introduces a lot of missing data due to variable ploidies
arabidopsis.genlight <- vcfR2genlight(x=arabidopsis.chrom.fin@vcf, n.cores=1) # On Linux/macOS and with large data use higher n.cores
warnings() # See errors - due to missing data when handling 4N vs. 2N
arabidopsis.genlight # Check it
arabidopsis.loci <- vcfR2loci(x=arabidopsis.chrom.fin) # Loci
arabidopsis.loci # Check it
print(x=arabidopsis.loci, details=TRUE)
```
Convert vcfR into DNAbin

```r
# There are various options how
# to process variants in VCF

?vcfR2DNAbin

arabidopsis.dnabin <- vcfR2DNAbin(
x=arabidopsis.chrom.fin,
consensus=FALSE, extract.haps=
TRUE, unphased_as_NA=FALSE,
asterisk_as_del=FALSE)

arabidopsis.dnabin # Check it
dim(arabidopsis.dnabin)
as.character.DNAbin
  (arabidopsis.dnabin[1:15, 1:12])
image.DNAbin(arabidopsis.dnabin)
snpposi.plot.DNAbin
  (arabidopsis.dnabin)
snpposi.test.DNAbin
  (arabidopsis.dnabin)
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Export data

```r
# Convert genind into DF using genind2df()
hauss.df <- genind2df(x=hauss.genind, pop=NULL, sep="/",
    usepop=TRUE, oneColPerAll=FALSE)

# Save microsatellites to disk - check settings of write.table
write.table(x=hauss.df, file="haussdata.txt", quote=FALSE,
    sep="\t", na="NA", dec=".", row.names=TRUE, col.names=TRUE)

# Export of DNA sequences into FASTA format
write.dna(x=usflu.dna, file="usflu.fasta", format="fasta",
    append=FALSE, nbcol=6)

seqinr::write.fasta(sequences=as.character.DNAbin(gunnera.dna),
    names=names(gunnera.dna), file.out="gunnera.fasta", open="w")

# Export DNA sequences as NEXUS
write.nexus.data(x=gunnera.dna, file="gunnera.nexus", format="dna")

write.vcf(x=arabidopsis.vcf, file="arabidopsis.vcf.gz") # Export VCF

# Export tree(s) (objects of class phylo) - will be introduced later
write.tree(phy=hauss.nj.bruvo, file="haussknechtii.nwk") # In NEWICK
write.nexus(hauss.nj.bruvo, file="haussknechtii.nexus") # In NEXUS
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

127/421
Introduction

Overview

R

Data

Alignment

Basic analysis

SNP

DAPC

Spatial analysis

Trees

Evolution

The end

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

128/421

Import your own data

Tasks

1. Prior to import into R, **ensure your data are correct** — same decimal separator everywhere, consistent structure of CSV/TSV, no syntactic problems in FASTA/NEXUS/NEWICK/...

2. Import some of your data into R
   - Be inspired by previous slides — edit commands to fit your needs and process your data
   - R is extremely flexible, but not everything is figured out within one minute...
   - Import preferably your data — you’ll later use them to perform selected analysis

3. Check your data after import to ensure they were correctly read

   - Upcoming chapters can serve like inspiration (not exhaustive) how to process your data in R, what is possible to do with them...
   - Previous examples are not covering all possibilities...
Alignment

Overview and MAFFT
MAFFT, Clustal, MUSCLE and T-Coffee
Multiple genes
Display and cleaning
Tasks

Importance of alignment

- All sequences must be aligned prior to any analysis!
- Be sure to either import already aligned sequences of same length — or align them
- Aligned sequences commonly require post-processing — trimming, ... as especially e.g. distance-based analysis are sensitive to missing data
Multiple sequence alignment

- Good alignment is basic condition for any analysis of DNA sequences
- DNA/RNA and protein sequences must be aligned prior any subsequent analysis (tree building, ...)
- R doesn’t have any possibility for visual editing (use rather software like Unipro UGENE, Geneious or CLC Genomics Workbench
- R can automatically (in batch) run multiple sequence alignments of multiple genes (there are several possibilities)
  - Simple scripts for this task can be written in any scripting language like BASH, Perl or Python — only matters what user likes, knows and wish to do with the results...
  - See e.g. https://github.com/V-Z/hybseq-scripts/blob/master/bin/hybseq_4_alignment_3_run.r from HybSeq scripts
- R packages use common alignment software: MAFFT, MUSCLE, Clustal, ...
  - User must install this software manually — R is just using external applications (in the examples shown)
Multiple sequence alignment with MAFFT

- MUSCLE is available in packages `muscle` and `ape` — first one reads “*StringSet” classes R objects and writes “*MultipleAlignment” R classes objects; the latter reads and writes object of class “DNAbin”

- `ape` also contains functions to use Clustal and T-Coffee — both read and write `DNAbin`

- MAFFT is available from (same author) in packages `ips` and `phyloch` — both read and write `DNAbin`

```r
library(ape)
library(ips)
# Requires path to MAFFT binary - set it according to your installation
# read ?mafft and mafft's documentation
# Change "exec" to fit your path to mafft (on Windows point to mafft.bat)!
gunnera.mafft <- mafft(x=gunnera.dna, method="localpair", maxiterate=100, options="--adjustdirection", exec="/usr/bin/mafft")
gunnera.mafft # See results, compare with 'gunnera'
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Clustal, MUSCLE and T-Coffee from ape

class(gunnera.mafft)
image.DNAbin(gunnera.mafft)
# read ?clustal and documentation of Clustal, Muscle and T-Coffee
# when using them to set correct parameters
gunnera.clustal <- ape::clustal(x=gunnera.dna, pw.gapopen=10, pw.gapext=0.1, gapopen=10, gapext=0.2, exec="/usr/bin/clustalw2", quiet=FALSE, original.ordering=TRUE) # Change "exec" to fit your path to clustal!
gunnera.clustal
class(gunnera.clustal)
image.DNAbin(gunnera.clustal)
gunnera.muscle <- muscle(x=gunnera.dna, exec="muscle", quiet=FALSE, original.ordering=TRUE) # Change "exec" to fit your path to muscle!
gunnera.muscle
class(gunnera.muscle)
image.DNAbin(gunnera.muscle)
?muscle::muscle # See options in muscle package
Multiple sequence alignment with MUSCLE

# Remove gaps from alignment - destroy it

gunnera.nogaps <- del.gaps(gunnera.muscle)

del.gaps # See for details
Align multiple genes

- NGS/HTS introduced work with hundreds and thousands of genes, it makes sense to process them in batch and not manually one-by-one

```r
# Create a list of DNAbin objects to process
multialign <- list(gunnera.dna, usflu.dna, usflu.dna2)

# See it
multialign
class(multialign)
lapply(X=multialign, FUN=class)

# Do the alignment
# Change "exec" to fit your path to mafft (on Windows point to mafft.bat)!
multialign.aln <- lapply(X=multialign, FUN=ips::mafft, method="localpair",
                        maxiterate=100, exec="/usr/bin/mafft")

# See result
multialign.aln
multialign.aln[[1]]
lapply(X=multialign.aln, FUN=class)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Align multiple genes in parallel

- There are plenty of implementations of parallelisation and using of computer clusters, see https://CRAN.R-project.org/web/views/HighPerformanceComputing.html

```r
library(parallel)
# Do the same in parallel (mclapply do the tasks in parallel, not
# one-by-one like lapply)
multialign.aln2 <- mclapply(X=multialign, FUN=ape::muscle,
  exec="muscle", quiet=FALSE, original.ordering=TRUE)
# Change "path" to fit your path to muscle!
# mclapply() relies on forking and hence is not available on Windows
# unless "mc.cores=1"
# See result
multialign.aln2
lapply(X=multialign.aln2, FUN=class)
?mclapply # See more options
?clusterApply # See more options (parLapply should work on Windows)
```
Checking the alignment

```r
# Plotting alignment
image.DNAbin(x = gunnera.mafft)

# Check the alignment
checkAlignment(x = usflu.dna, check.gaps = TRUE, plot = TRUE, what = 1:4)
checkAlignment(x = as.matrix.DNAbin(x = gunnera.clustal), check.gaps = TRUE,
plot = TRUE, what = 1:4)
?checkAlignment # See details

# DNAbin can be technically list or matrix - some functions require
# list, some matrix, some can handle both - check manual and if needed,
# use:
as.matrix.DNAbin()
as.list.DNAbin()

# Matrix makes sense only for alignments, list for any import
# (sequences do no have to have same lengths)
```
Checking the alignment

![Image of alignment and sequence analysis](attachment:image.png)
Cleaning the alignment

```r
# Delete all columns/rows containing only gaps or missing data (N, ?, -)
gunnera.mafft <- deleteEmptyCells(DNAbin=gunnera.mafft)

?ips::deleteEmptyCells # See help page for details

?phyloch::delete.empty.cells # See help page for details

# Delete all columns containing at least 25% of gaps
gunnera.mafft.ng <- deleteGaps(x=gunnera.mafft, gap.max=nrow(gunnera.mafft)/4)
gunnera.mafft.ng

# Do not confuse with function delete.gaps() from phyloch package

?deleteGaps # "nmax=0" deletes all columns with any gap

multialign.aln.ng <- lapply(X=multialign.aln, FUN=deleteGaps, gap.max=5)
multialign.aln.ng

# Delete every line (sample) containing at least 20% of missing data
gunnera.mafft.ng <- del.rowgapsonly(x=gunnera.mafft.ng, threshold=0.2, freq.only=FALSE)
gunnera.mafft.ng

?ape::del.rowgapsonly # See help page for details
```
Cleaning the alignment

```r
# Delete every alignment position having at least 20% of missing data
gunnera.mafft.ng <- del.colgapsonly(x=gunnera.mafft.ng, threshold=0.2, freq.only=FALSE)
gunnera.mafft.ng
# See help page for details
# Display the result
image.DNAbin(x=gunnera.mafft.ng)
lapply(X=multialign.aln.ng, FUN=image.DNAbin)
```

- “Strictness” of alignment cleaning depends on following steps — NJ (and another distance-based methods) doesn’t like more than ~10–15% of missing data, but some tree builders are able to work with gaps — check their documentation...

- Automated cleanup is useful especially if batch processing plenty of genes
Introduction

R

Data

Alignment

Basic analysis

SNP

DAPC

Spatial analysis

Trees

Evolution

The end

Overview and MAFFT

MAFFT, Clustal, MUSCLE and T-Coffee

Multiple genes

Display and cleaning

Tasks

Practice alignment

Tasks

1. Download (import from on-line database or your file from disk) sequences of ITS (or other variable gene if you prefer) of your favourite organism.

2. Check the imported data.

3. Align sequences with your preferred aligner.

4. Check the resulting alignment.

5. Trim the alignment — delete columns/rows with too much missing data. Think about various thresholds and their implications.

6. Compare outputs of several aligners and/or different parameters (gap penalty etc.).
Basic analysis

5 Basic analysis
First look at the data
Statistics
MSN
Genetic distances
AMOVA
Hierarchical clustering
NJ (and UPGMA) tree
PCoA
Tasks
Introductory overview of statistics and methods I

• Selected method depends on data type, question to answer, ... — see further slides and chapters
  • Check assumptions and requirements of the methods before usage
  • Think if the method answers your question
  • Always be opened for new possibilities coming with new methods and packages developed...

• Population-genetic indices — from slide 150
  • Huge number...
  • Characterize differences among individuals/groups or genetic variability on various levels (within/among individuals/populations, ...)
  • One number tries to describe whole situation — always very rough
  • Description of heterozygosity, allelic richness, distribution of multi locus genotypes among populations, level of inbreeding, ...

• Distance-based methods — from slide 171
  • It is crucial to select appropriate distance method for given data type
  • Usually require the distance matrix to be Euclidean
Introductory overview of statistics and methods II

- Distance matrix has one single number (index) for each pair of comparisons (individuals, populations) — rough
- Generally, the matrices describe pairwise similarities among the individuals/populations
- Distance-based methods are phenetic
  - Based on similarity (described by the matrix), not on any (evolutionary) model
  - The matrix based on genetic data is supposed to well reflect the genetic similarity, thus real relationships among individuals/populations
- **Hierarchical clustering** — from slide 187
  - Several methods clustering individuals according to their (dis)similarity from top or down into clusters
  - (Un)weighted per-group mean average (U/WPGMA) and others
  - Used more in ecology, for genetic data not so much anymore (following methods use to produce better results)
- **Neighbor-Joining (NJ)** — from slide 191
  - A tree starting from the two most similar individuals and connecting in the next steps next and next the most similar individual
Introductory overview of statistics and methods III

- In some cases artificially chains individuals
- Several methods try to improve it — slide 206
  - **Principal Coordinates Analysis (PCoA)** — from slide 207
    - The most common method of multivariate statistics for genetic data
    - Shows individuals in 2D scatter plot to retain maximum variability (by finding correlations among loci)

- **Minimum Spanning Network (MSN)** — slide 170
  - Simple network connecting the most similar genotypes/haplotypes
  - Useful for clones, cpDNA, mtDNA, ...

- **Multivariate statistics**
  - Two variables are easily displayable in 2D xy-scatter plot (we can calculate correlation, whatever)
  - In molecular data, each locus is more or less independent variable — 1000 bp alignment has 1000 variables: How to display plot with 1000 axes to be able to really see something?
• Methods like Principal Component Analysis (PCA), Non-Metric Multidimensional Scaling (NMDS) or PCoA look for correlations between pairs of variables to reduce them into new variables — after many steps new uncorrelated variables retaining maximum of original variability are constructed
• New variables are sorted according amount of variability they show (the decrease is very steep — first 1–4 axes are usually enough) — it is possible to display xy-scatter plot showing most of variability of the data
• Good for data display and creation of hypotheses — not to verify them (there is no statistical test)
• Data are commonly scaled — all variables are in same scale

• **Maximum Parsimony (MP)** — from slide 296
• Generally, the methods are looking for the most simple solution under given model, e.g. to construct phylogenetic tree requiring the lowest number of evolutionary changes (DNA mutations)
Introduction

First look at the data

Statistics

MSN

Genetic distances

AMOVA

Hierarchical clustering

NJ (and UPGMA) tree

PCoA

Tasks

Introductory overview of statistics and methods V

• It is easy to score how good the solution is (comparing to another solution), but computationally demanding to find the best one

• **Maximum Likelihood (ML)**
  • Methods look for the most likely (probable) solution of the data under given model, e.g. the most likely tree under given mutational model
  • It is easy to score how good the solution is (comparing to another solution), but computationally demanding to find the best one

• **Bayesian statistics**
  • Based on **Bayesian theorem** — probability of model under given data
  • Methods are looking for the best (e.g. evolutionary) **model** (e.g. phylogenetic tree) **explaining the data** (e.g. DNA sequences)
  • Algorithm exploring possible models, scoring them and approaching the best runs in steps (iterative generations)
    • After some time it converges to find optimal solution (usually described by logarithms of likelihood of given model)
Introductory overview of statistics and methods VI

- Usually, ∼millions (or even more) of generations (iterative steps) are required
- Beginning use to be very unstable — it is discarded as burn-in (“heating” of Markov Chain Monte Carlo (MCMC) doing the exploration and optimization of models), usually ∼10–25% of steps

- MP, ML and Bayesian statistics contain (evolutionary) models — they are not based on similarity (as matrix-based methods), so that they are supposed to reveal real structure in the data, on the other hand they are computationally demanding

- Permutations, bootstraps and another tests
  - It is necessary to test statistical significance of the obtained results
  - Most common methods somehow shuffle the data (drop one column, ...) and repeat the calculation to see how stable is the result (it might be driven by one or few loci, ...)
  - Whole process is repeated ∼100–1000 times and output is shown as histogram of simulations vs. the observed value, in how many percents the same result was obtained (e.g. bootstrap) or as p-value (what is probability that the pattern was created by random process)
  - p = 0.05 means 95% probability that the data are non-random
Questions and data

- Methods in this section answer questions about genetic characteristics of individuals/populations
  - Overall genetic similarity of individuals/populations (without spatial or another context) — various population-genetic indices, PCoA, ...
  - Distribution of genotypes within/among populations
  - Description of genetic characteristics of populations — heterozygosity, Hardy-Weinberg equilibrium, F-statistics, ...
  - Hierarchical relationships among individuals/populations (UPGMA, NJ, ...)
- Any data can be used
  - Population genetic studies use to use as variable genetic markers as possible (depending on scale, e.g. SSRs, AFLP, RAD-Seq, highly variable introns like ITS, ...)
- Nearly all data types are processed in similar/same way
  - It’s important to select correct genetic index, distance method, etc. for particular data types and/or question
Load needed libraries

1. library(ape) # Analysis of phylogenetics and evolution
2. library(ade4) # Analysis of ecological data, multivariate methods
3. library(adegenet) # Exploratory analysis of genetic and genomic data
4. library(pegas) # Population and evolutionary genetics
   # Population genetic analysis, including populations with mixed
   # reproduction
5. library(poppr)
6. library(hierfstat) # Hierarchical F-statistics
7. library(corrplot) # Visualization of correlation matrix
8. library(StAMPP) # Statistical analysis of mixed ploidy populations
9. library(philentropy) # Various genetic distances

• Of course, there are plenty of another options...
  • See e.g. CRAN Views for genetics or multivariate statistics
  • Representative, but not exhaustive examples are shown
Descriptive statistics I

- We will now work mainly with diploid SSRs of *Taraxacum haussknechtii*, you can **try with other data** examples by **yourselves**

```r
# Get summary - names and sizes of populations, # heterozygosity, some info about loci
hauss.summ <- summary(hauss.genind)

# Plot expected vs. observed heterozygosity it looks like big difference
plot(x = hauss.summ$Hexp, y = hauss.summ$Hobs,
     main = "Observed vs expected heterozygosity",
     xlab = "Expected heterozygosity", ylab = "Observed heterozygosity")
abline(0, 1, col = "red")

# Bartlett's K-squared test of difference # between observed and expected heterozygosity - not significant
bartlett.test(list(hauss.summ$Hexp, hauss.summ$Hobs))

Bartlett test of homogeneity of variances
data:  list(hauss.summ$Hexp, hauss.summ$Hobs)
Bartlett's K-squared = 0.069894, df = 1, p-value = 0.7915
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Descriptive statistics II

- `t.test` and `bartlett.test` require data to have normal distribution — if the condition is not met, it is necessary to use some weaker non-parametric test (`kruskal.test`, `wilcox.test`, ...)
- See respective manual pages for details
- `shapiro.test()` tests the normality of given vector

```r
# T-test of difference between observed and expected heterozygosity
t.test(x = hauss.summ$Hexp, y = hauss.summ$Hobs, paired = TRUE, var.equal = T)

  data:  hauss.summ$Hexp and hauss.summ$Hobs  t = 3.5622, df = 11, p-value = 0.004456  # strongly significant
  alternative hypothesis: true difference in means is not equal to 0
  95 percent confidence interval:
       0.06114303 0.25887357
  sample estimates:
     mean of the differences
       0.1600083
```
# Create pane with some information
par(mfrow=c(2, 2))  # Divide graphical devices into 4 smaller spaces
# Plot alleles number vs. population sizes
plot(x=hauss.summ$n.by.pop, y=hauss.summ$pop.nall, xlab="Populations sample size", ylab="Number of alleles", main="Alleles numbers and sample sizes", col="red", pch=20)
# Add text description to the point
text(x=hauss.summ$n.by.pop, y=hauss.summ$pop.nall, lab=names(hauss.summ$n.by.pop), cex=1.5)
# Barplots of various data
barplot(height=hauss.summ$loc.n.all, ylab="Number of alleles", main="Number of alleles per locus", las=3)
barplot(height=hauss.summ$Hexp-hauss.summ$Hobs, main="Heterozygosity: expected-observed", ylab="Hexp - Hobs", las=3)
barplot(height=hauss.summ[["n.by.pop"]], main="Sample sizes per population", ylab="Number of genotypes", las=3)
dev.off()  # Closes graphical device to reset graphical settings
Graphs from previous slides
Population statistics by `poppr()`

- `poppr()` is the central function of the `poppr` package calculating plenty of population genetic indices.

```r
# See details
poppr(dat=hauss.genind, total=TRUE, sample=1000, method=4,
      missing="geno", cutoff=0.15, quiet=FALSE, clonecorrect=FALSE,
      plot=TRUE, index="rbarD", minsamp=1, legend=TRUE)

# Output table with indices:

<table>
<thead>
<tr>
<th>Pop</th>
<th>N</th>
<th>MLG</th>
<th>eMLG</th>
<th>SE</th>
<th>H</th>
<th>G</th>
<th>lambda</th>
<th>E.5</th>
<th>Hexp</th>
<th>Ia</th>
<th># More...</th>
</tr>
</thead>
<tbody>
<tr>
<td>He</td>
<td>13</td>
<td>1.97</td>
<td>1.58e-01</td>
<td>2.352</td>
<td>9.94</td>
<td>0.899</td>
<td>0.941</td>
<td>0.503</td>
<td>1.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ne</td>
<td>6</td>
<td>1.93</td>
<td>2.49e-01</td>
<td>1.561</td>
<td>4.50</td>
<td>0.778</td>
<td>0.930</td>
<td>0.604</td>
<td>3.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>47</td>
<td>43</td>
<td>6.07e-02</td>
<td>3.732</td>
<td>40.16</td>
<td>0.975</td>
<td>0.961</td>
<td>0.742</td>
<td>1.88</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- If `plot=TRUE`, a histogram of simulations (sample must be > 1) is plotted for each population for `rbarD` or `Ia` (according to selected index — see following slides for details).
Histograms of simulations of rbarD for each population

The populations are significantly far from being clonal

Data: hauss.genind
Permutations: 1000

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Too much to choose from?

Generally, there are plenty of different population indices (and distances and another statistics) with different assumptions and usage in many packages — it can be complicated to pick the best one... The course shows many examples, but the list is far from being exhaustive...

- **Pop** — Population analyzed
  - If `total=TRUE`, there are also statistics for whole dataset
- **N** — Number of individuals/isolates in the specified population
- **MLG** — Number of multilocus genotypes found in the specified population (see ?mlg)
- **eMLG** — The expected number of MLG at the lowest common sample size (set by `minsamp`)

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Population statistics returned by poppr() II

- **SE** — The standard error for the rarefaction analysis (assets species richness — how it grows with growing sample size)
  - Big difference between MLG and eMLG indicate some process lowering/increasing genetic diversity

- **H** — Shannon-Wiener Diversity index — evaluates number of genotypes and their distribution, takes entropy into account, grows with higher richness and diversity, sensitive to uneven sample size (is population sizes are very different, indices are not comparable)

- **G** — Stoddard and Taylor’s Index — roughly, similar approach as the previous one, highly enhanced

- **lambda** — Simpson’s index $\lambda = 1$ minus the sum of squared genotype frequencies — estimation of the probability that two randomly selected genotypes are different and scales from 0 (no genotypes are different) to 1 (all genotypes are different)
Population statistics returned by poppr() III

- **E.5** — Evenness — measure of the distribution of genotype abundances, wherein a population with equally abundant genotypes yields a value equal to 1 and a population dominated by a single genotype is closer to 0
- **Hexp** — Nei’s gene diversity (expected heterozygosity) — unbiased gene diversity (from 0 = no diversity to 1 = highest diversity)
- **Ia** — Index of Association (\(?ia\)) — widely used to detect clonal reproduction within populations
  - Populations whose members are undergoing sexual reproduction will produce gametes via meiosis, and thus have a chance to shuffle alleles in the next generation
  - Populations whose members are undergoing clonal reproduction generally do so via mitosis — most likely mechanism for a change in genotype is via mutation — the rate of mutation varies from species to species, but it is rarely sufficiently high to approximate a random shuffling of alleles
First look at the data

Statistics

• The index of association is a calculation based on the ratio of the variance of the raw number of differences between individuals and the sum of those variances over each locus
• It as the observed variance over the expected variance — if they are the same, then the index is zero (=prevailing clonal reproduction) after subtracting one — it rises with with increasing differences

• \( p.Ia \) — P-value for \( Ia \) from the number of reshuffling indicated in \( \text{sample} \)
• \( r\text{barD} \) — Standardized Index of Association for each population (see \( ?ia \)) — corrected for higher number of loci not to rise so steeply
• \( p.rD \) — P-value for \( \text{rbarD} \) from the number of reshuffles indicated in \( \text{sample} \)
• See \textit{poppr}'s manual} and \texttt{vignette("algo", package="poppr")} for details
**Departure from Hardy-Weinberg equilibrium**

- **In theory**, in large panmictic population without evolutionary influence everyone can mate with everyone (it is in equilibrium) and allele frequencies remain stable — in reality, environment, behavior, mutations, genetic drift, etc. are structuring the population.

```r
# According to loci
hauss.hwe.test <- hw.test(x=hauss.loci, B=1000)
hauss.hwe.test # See results per locus

<table>
<thead>
<tr>
<th>Locus</th>
<th>chi^2</th>
<th>df</th>
<th>Pr(chi^2 &gt;)</th>
<th>Pr.exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>msta93</td>
<td>383.55</td>
<td>190</td>
<td>3.552714e-15</td>
<td>0.000</td>
</tr>
<tr>
<td>msta101</td>
<td>0.69</td>
<td>1</td>
<td>4.052393e-01</td>
<td>0.657</td>
</tr>
<tr>
<td>msta102</td>
<td>83.07</td>
<td>10</td>
<td>1.25011e-13</td>
<td>0.000</td>
</tr>
<tr>
<td>msta103</td>
<td>77.18</td>
<td>45</td>
<td>1.998865e-03</td>
<td>0.000</td>
</tr>
</tbody>
</table>
```

- **Pr.exact** shows significance of the departure (i.e. non-equilibrium distribution of alleles within population — calculated per loci)
- **$\chi^2$ test** (without or with the permutations) test the departure — if it is significant or not — not how much it is departing.

[Visit Vojtěch Zeisek](https://trapa.cz/)

---

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

160/421
Departure from HWE

- Calculation is always done per-locus — see differences, possibly do statistics like `summary(hauss.hwe.test)` or so

```r
# According to populations
# Separate genind object into list of genind objects for individual populations
hauss.pops <- seppop(hauss.genind)
hauss.pops

# Convert genind back to loci (list of loci objects according to populations)
hauss.pops.loci <- lapply(X = hauss.pops, FUN = genind2loci)

# Calculate the results per populations
lapply(X = hauss.pops.loci, FUN = hw.test, B = 1000)
```

- If there is significant departure from HWE, think about biological process (with respect to life traits of species studied) which could cause such structuring
F-statistics I

- Functions return tables of F-statistics values for populations/loci (roughly 0 — no structure, 1 — fully structured)
- The different F-statistics look at different levels of population structure:
  - $F_{IT}$ is the inbreeding coefficient of an individual relative to the total population (all samples)
  - $F_{IS}$ is the inbreeding coefficient of an individual relative to the subpopulation (“population” in common terminology) and averaging them
  - $F_{ST}$ is the effect of subpopulations (“population”) compared to the total population (all samples)
- For `Fst`, `theta.msat` and another similar functions the data object `must` contain population column (see manual of respective function)

```r
# Fit, Fst and Fis for each locus
Fst(x=hauss.loci, pop=1)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>msta93</td>
<td>0.31835291</td>
<td>0.17867087</td>
</tr>
<tr>
<td>msta101</td>
<td>-0.09968472</td>
<td>0.04064928</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

162/421
# F-statistics II

1. The end

## Pairwise Fst comparing populations

1. Convert Adagent's genind to format of hierfstat package

   ```r
   genind2hierfstat
   ```

2. Nei's pairwise Fst between all pairs of populations.

   ```r
   pairwise.neifst(dat = genind2hierfstat(dat = hauss.genind))
   ```

<table>
<thead>
<tr>
<th></th>
<th>He</th>
<th>Ne</th>
<th>Oh</th>
<th>Pr</th>
<th>Sk</th>
</tr>
</thead>
<tbody>
<tr>
<td>He</td>
<td>NA</td>
<td>0.3204</td>
<td>0.1824</td>
<td>0.1942</td>
<td>0.1599</td>
</tr>
<tr>
<td>Ne</td>
<td>0.3204</td>
<td>NA</td>
<td>0.1210</td>
<td>0.2548</td>
<td>0.1229</td>
</tr>
<tr>
<td>Oh</td>
<td>0.1824</td>
<td>0.1210</td>
<td>NA</td>
<td>0.1038</td>
<td>0.0833</td>
</tr>
<tr>
<td>Pr</td>
<td>0.1942</td>
<td>0.2548</td>
<td>0.1038</td>
<td>NA</td>
<td>0.1407</td>
</tr>
<tr>
<td>Sk</td>
<td>0.1599</td>
<td>0.1229</td>
<td>0.0833</td>
<td>0.1407</td>
<td>NA</td>
</tr>
</tbody>
</table>

- Check also another functions of `hierfstat` package, there are more options
- Hierfstat package implements $F_{ST}$ only for haploid and diploid populations, **StAMPP** (next slide) also for another ploidies and mixed ploidy data

Vojtěch Zeisek (https://trapa.cz/)
F-statistics for mixed ploidy data I

- Methods from StAMPP package (the same is the case for any method working somehow with distances) are sensitive to missing data...
  - Carefully filter the VCF before doing any analysis
- Populations must be already defined in the genlight object

```r
# stamppFst requires population factor in genlight (here, population code consists of first three letters of individual's name)
indNames(arabidopsis.genlight)
# Population code consists of first three letters of individual's name - # extract the population name part
substr(x=indNames(arabidopsis.genlight), start=1, stop=3)
pop(arabidopsis.genlight) <- substr(x=indNames(arabidopsis.genlight),
  start=1, stop=3)
pop(arabidopsis.genlight) # Check it
popNames(arabidopsis.genlight)
?StAMPP::stamppFst # See method details
```

Vojtěch Zeisek (https://trapa.cz/)
# Calculating the Fst

```r
arabidopsis.fst <- StAMPP::stamppFst(geno = arabidopsis.genlight, 
nboots = 100, percent = 95, nclusters = 1)

# For large data use higher nclusters to parallelize calculations
arabidopsis.fst[["Fsts"]]
# Matrix of Fst among populations
arabidopsis.fst[["Pvalues"]]
# Matrix of P values

# Save results - open in spreadsheet (e.g. LibreOffice Calc)
write.table(x = arabidopsis.fst[["Fsts"]], file = "arabidopsis_fst.tsv", 
quote = FALSE, sep = "\t")

# Correlation plot of pairwise Fst

corrplot(corr = arabidopsis.fst[["Fsts"]], method = "circle", type = "lower", 
col = funky(15), title = "Correlation matrix of Fst among populations", 
is.cor = FALSE, diag = FALSE, outline = TRUE, order = "alphabet", 
tl.pos = "lt", 
tl.col = "black")

?corrplot # See for more options
```

# Display in similar way also another Fst tables
Multi locus genotypes and inbreeding coefficient

• Especially when working with clonal species or species with low genetic structure it is beneficial to know how many unique genotypes there are and how they are distributed across populations

• Inbreeding coefficient estimates level of mating among individuals with (nearly) identical genotypes — important e.g. for conservation studies or work with agricultural species

• Such studies require sufficiently variable marker (e.g. SSRs) so that results show real genetic structuring and not “just” low variability of selected marker

• Reliable estimation of such parameters also require high number of individuals from each population
  • 10 is usually considered as minimum, but more is recommended
Multi locus genotypes

Functions from poppr package — the best for microsatellites, although available also for another data types

# Total number of MLGs
# (simple value)
mlg(gid=hauss.genind, quiet=FALSE)

# MLGs shared among populations
mlg.crosspop(gid=hauss.genind, df=TRUE, quiet=FALSE)

# Detailed view on distribution
# of MLGs into populations
# (table and/or plot)
mlg.table(gid=hauss.genind, bar=TRUE, total=TRUE, quiet=FALSE)
mlg.vector(hauss.genind)
mlg.id(hauss.genind)
Inbreeding

Average inbreeding in Salers cattle

# Load training data (cattle)
data(microbov)

# Separate populations of Salers
microbov.pops <- seppop(microbov)
  [["Salers"]]

# Calculate the inbreeding
microbov.inbr <- inbreeding(x=microbov.pops, N=100)
inbreeding # Check for settings

# population means for plotting
microbov.bar <- sapply(X=microbov.inbr, FUN=mean)

# Plot it
hist(x=microbov.bar, col="firebrick", main="Average inbreeding in Salers cattle")
Minimum Spanning Network (MSN)

- Package `poppr`, for SSRs based on Bruvo’s distance, can handle any data type (requires `genind` objects)

- Shows relationships among haplotypes (unique genotypes), can be labeled by population, haplotype, ...
  - Size of pie is proportional to number of individuals assigned
  - Lines connect haplotypes according to their similarity

- Suitable for less variable datasets, e.g. some mitochondrial or plastide genes (or SSRs for less variable species) — otherwise the figure is messy
Minimum Spanning Network

```r
Bruvo.msn
# Get the MSN
# Note SSRs repeats 'rep(2, 12)' -
# change according to your data
bruvo.msn(gid=hauss.genind,
replen=rep(2, 12), loss=TRUE,
palette=rainbow, vertex.label="inds",
gscale=TRUE, wscale=TRUE, showplot=TRUE)
# For another data types
# (not only microsatellites)
imsn.poppr
# Interactive creation of MSN
# Opens browser window
imsn
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022

170/421
Distances

- Distance-based methods are among the most popular in biology
- Huge number of applications
- A lot of different distances — it is crucial to select correct distance matrix for particular task
  - Which input data?
  - Which purpose?
  - Selecting wrong distance method can lead in misleading results in PCoA, NJ, ...
- All method have lots of assumptions and limits — check them prior usage
  - Genetic drift, infinite alleles, ...
- If resulting distance matrix is not Euclidean (see also further), following analysis can be misleading
# Simple dissimilarity distance matrix

```r
?dist.gene # Details about methods of this distance constructions

hauss.dist <- dist.gene(x = hauss.genind@tab, method = "pairwise")

# Nei's distance (not Euclidean) for populations (other methods are
# available, see ?dist.genpop)

hauss.dist.pop <- dist.genpop(x = hauss.genpop, method = 1, diag = T, upper = T)

# Test if it is Euclidean

is.euclid(hauss.dist.pop, plot = TRUE, print = TRUE, tol = 1e-10) # FALSE = No

# Turn to be Euclidean

hauss.dist.pop <- cailliez(distmat = hauss.dist.pop, print = FALSE, cor.zero = TRUE)

is.euclid(hauss.dist.pop, plot = TRUE, print = TRUE, tol = 1e-10) # TRUE = OK
```

Most of analysis based on distances more or less require **Euclidean distances** (non-negative, Pythagorean theorem is valid, etc.). If the distance matrix contains non-Euclidean distances, the result can be weird...
## Distances reflecting microsatellite repeats

```r
# Bruvo's distances weighting SSRs repeats - take care about replen
# parameter - requires repetition length for every SSRs locus
# E.g. if having 5 SSRs with repeat lengths 2, 2, 3, 3 and 2 bp use:
# bruvo.dist(pop=... replen=c(2, 2, 3, 3, 2)...)  
hauss.dist.bruvo <- bruvo.dist(pop=hauss.genind, replen=rep(2, 12),
    loss=TRUE)

# Test if it is Euclidean
is.euclid(hauss.dist.bruvo, plot=TRUE, print=TRUE, tol=1e-10)

hauss.dist.bruvo <- cailliez(distmat=hauss.dist.bruvo, print=FALSE,
    tol=1e-07, cor.zero=TRUE)  # Turn to be Euclidean and verify below
is.euclid(hauss.dist.bruvo, plot=TRUE, print=TRUE, tol=1e-10)

hauss.dist.bruvo  # Show it
```

- See poppr’s manual and manual pages of the functions for details and different possibilities of settings
- Be careful when changing non-Euclidean distances to Euclidean — the transformation more or less changes meaning of the distances!
Turning distance matrix into Euclidean is controversial...

How to deal with zero distances in original matrix? There is no really good solution...

Histograms of Bruvo distance before and after transformation:
More distances...

# Nei's distance (not Euclidean) for individuals
# (other methods are available, see ?nei.dist from poppr package)
hauss.dist.nei <- nei.dist(x=hauss.genind, warning=TRUE)
is.euclid(distmat=hauss.dist.nei, plot=TRUE, print=TRUE, tol=1e-10)

# Dissimilarity matrix returns a distance reflecting the number of
# allelic differences between two individuals
hauss.dist.diss <- diss.dist(x=hauss.genind, percent=FALSE, mat=TRUE)
is.euclid(as.dist(hauss.dist.diss), plot=TRUE, print=TRUE, tol=1e-10)

Import own distance matrix from another software:

Fe 0.0000   132.019  109.159 ...
He 132.019  0.00000  9.89111 ...
Oh 109.159  9.89111  0.00000 ...
Pr 139.5669 8.55312  4.40562 ...
Ne 156.7619 9.96143 16.69274 ...

MyDistance <- read.csv("distances.txt", header=TRUE, sep="\t",
dec=".", row.names=1) # Or so...
MyDistance <- as.dist(MyDistance)
class(MyDistance)
dim(MyDistance)
MyDistance
### Different distances have different use case and outputs...

Different distances available in package `poppr`

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
<th>Assumption</th>
<th>Euclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevosti 1975</td>
<td>prevosti.dist,</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>diss.dist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nei 1972, 1978</td>
<td>nei.dist</td>
<td>Infinite Alleles, Genetic Drift</td>
<td>No</td>
</tr>
<tr>
<td>Edwards 1971</td>
<td>edwards.dist</td>
<td>Genetic Drift</td>
<td>Yes</td>
</tr>
<tr>
<td>Reynolds 1983</td>
<td>reynolds.dist</td>
<td>Genetic Drift</td>
<td>Yes</td>
</tr>
<tr>
<td>Rogers 1972¹</td>
<td>rogers.dist</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>Bruvo 2004</td>
<td>bruvo.dist</td>
<td>Step-wise Mutation</td>
<td>No</td>
</tr>
</tbody>
</table>


---

# See details of distance methods in package poppr

```r
vignette("algo", package="poppr")
```
Comparison of different matrices

```r
# Compare different distance matrices
# List of functions to be parsed to respective dist.* function
distances <- c("Nei", "Rogers", "Edwards", "Reynolds", "Prevosti")
# Calculate the distance matrices
dists <- lapply(distances, function(x) {
  DISTFUN <- match.fun(paste(tolower(x), "dist", sep="."))
  DISTFUN(hauss.genind.cor)
})
# Add names for the distance names
names(dists) <- distances
dists[["Bruvo"]]<- hauss.dist.bruvo # Add Bruvo distance
dists # Check list of distances
par(mfrow=c(2, 3)) # Split graphical device into 2 lines, 3 panes each
# Calculate NJ and plot all trees
x <- lapply(names(dists), function(x) {
  plot(njs(dists[[x]]), main=x, type="unrooted"
  add.scale.bar(lcol="red", length=0.1)
})
dev.off() # Close graphical device to reset settings
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Molecular data in R
January 31 to February 3, 2022
Neighbor-Joining of same dataset under different matrices

The results are very different...
Distances among DNA sequences

- The sequences must be aligned before calculating distances among them!
- Selection of mutational model has significant impact to results...

```r
# There are various models available
dist.dna

# Create the distance matrix
usflu.dist <- dist.dna(x = usflu.dna, model = "TN93")

# Check the resulting distance matrix
usflu.dist
class(usflu.dist)
dim(as.matrix(usflu.dist))

# Create another distance matrix
gunnera.dist <- dist.dna(x = gunnera.mafft.ng, model = "F81")

# Check it
gunnera.dist
class(gunnera.dist)
dim(as.matrix(gunnera.dist))
```
Distances and genlight object

Pairwise genetic distances for each data block (genlight objects with whole genome data) — sensitive to missing data (not useful in every case):

```r
usflu.dists.l <- seploc(usflu.genlight, n.block=10, parallel=FALSE)
class(usflu.dists.l)
usflu.dists <- lapply(X=usflu.dists.l, FUN=function(D) dist(as.matrix(D)))
class(usflu.dists)
names(usflu.dists)
class(usflu.dists[[1]])
usflu.distr <- Reduce(f="+", x=usflu.dists)
class(usflu.distr)
usflu.distr
# It is possible to use just basic dist function on whole genlight object
# (might require a lot of RAM)
usflu.distg <- dist.gene(as.matrix(usflu.genlight))
```

Rationale of this approach is to save resources when dividing whole data set into smaller blocks — useful for huge data, not for all of the cases.
# stamppNeisD requires population factor in genlight Nei's 1972 distance
# between individuals (use pop=TRUE to calculate among populations)
arabidopsis.dist <- stamppNeisD(geno=arabidopsis.genlight, pop=FALSE)
# Check it
head(arabidopsis.dist)
dim(arabidopsis.dist)
class(arabidopsis.dist)
# The same on population level
arabidopsis.dist.pop <- stamppNeisD(geno=arabidopsis.genlight, pop=TRUE)
# Check it
head(arabidopsis.dist.pop)
dim(arabidopsis.dist.pop)
class(arabidopsis.dist.pop)
Distances in mixed-ploidy data sets II

```r
# Export the distance matrix as Phylip format for usage in external
# software (e.g. SplitsTree)
stamppPhylip(distance.mat = arabidopsis.dist, file = "arabidopsis_dist.txt")

# Genomic relationship matrix
?stamppGmatrix # Method details
arabidopsis.genomat <- stamppGmatrix(geno = arabidopsis.genlight)

# Check it
head(arabidopsis.genomat)
dim(arabidopsis.genomat)
class(arabidopsis.genomat)
```

- If there are plenty of missing data and/or the distance is far from being Euclidean, it will not work very well... — sanitize missing data prior calculating distance (e.g. using `poppr::missingno`)
- Always check created distances

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Over 40 distances from philentropy package

- There is enormous number of various distance measures...
- For example Jaccard index is used to compare binary (presence/absence) data like AFLP

```r
getDistMethods() # See available distances
?distance # See details of distances
# Calculate e.g. Jaccard index for AFLP data
# amara.aflp has 30 columns, see dim(amara.aflp)
# column 1 contains names, see head(amara.aflp)
amara.jac <- distance(x=amara.aflp[,2:30], method="jaccard")
# See result
class(amara.jac)
amara.jac
# Make it distance matrix
amara.jac <- as.dist(m=amara.jac, diag=TRUE, upper=TRUE)
amara.jac
```
Visualize pairwise genetic similarities

```r
# table.paint() requires data
# frame, dist can't be directly
# converted to DF

table.paint(df = as.data.frame(as.matrix(usflu.dist)), cleg = 0, clabel.row = 0.5, clabel.col = 0.5)

# Same visualization, colored
# heatmap() reorders values
# because by default it plots
# also dendrograms on the edges

heatmap(x = as.matrix(usflu.dist), Rowv = NA, Colv = NA, symm = TRUE)
```

- Colored according to value
- Another possibility is to use `corrplot::corrplot()` for correlation plots

---

Molecular data in R

Vojtěch Zeisek (https://trapa.cz/)

January 31 to February 3, 2022

184/421
AMOVA I

- Analysis of molecular variance tests if there are significant differences among populations (and/or another levels)
- Some implementations can partition variance into various levels
- `pegas::amova` returns a table of sums of square deviations (SSD), mean square deviations (MSD), and the number of degrees of freedom (df), and a vector of variance components (sigma2)
- See `sigma2` column for how much of the variance is on which level — percentage can be calculated as percentage of each level from total
- For more complicated hierarchy see `?poppr::poppr.amova`
- For mixed-ploidy data sets see `?StAMPP::stamppAmova`
## AMOVA II

```r
hauss.pop <- pop(hauss.genind)
hauss.amova <- pegas::amova(hauss.dist~hauss.pop, data=NULL,
  nperm=1000, is.squared=FALSE)
# See results
hauss.amova
...

<table>
<thead>
<tr>
<th></th>
<th>SSD</th>
<th>MSD</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>hauss.pop</td>
<td>30.71923</td>
<td>7.679809</td>
<td>4</td>
</tr>
<tr>
<td>Error</td>
<td>119.58100</td>
<td>2.847167</td>
<td>42</td>
</tr>
<tr>
<td>Total</td>
<td>150.30023</td>
<td>3.267396</td>
<td>46</td>
</tr>
</tbody>
</table>

Variance components:

<table>
<thead>
<tr>
<th></th>
<th>sigma2</th>
<th>P.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hauss.pop</td>
<td>0.55738</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>2.84717</td>
<td></td>
</tr>
</tbody>
</table>

# From here we can calculate the percentage
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

186/421
Heat maps

```r
# Based on various distances
heatmap(as.matrix(hauss.dist), symm=TRUE, labRow=rownames(
  as.matrix(hauss.dist.bruvo)), labCol=colnames(as.matrix(
  hauss.dist.bruvo))
# hauss.dist doesn't contain
# names of individuals - add here
heatmap(as.matrix(hauss.dist.pop), symm=TRUE)
heatmap(as.matrix(hauss.dist.bruvo), symm=TRUE)
heatmap(as.matrix(hauss.dist.diss), symm=TRUE)
# See settings like colors,
# dendrogram, etc.
?heatmap
```
Hierarchical clustering — UPGMA and others

- This is very basic function to make dendrogram
- There are better possibilities (NJ etc — see slide 191 and onward)

```
# According to distance used
# How to use hierarchical clustering
?hclust
plot(hclust(d=hauss.dist, method="complete"))
plot(hclust(d=hauss.dist.pop, method="complete"))
plot(hclust(d=hauss.dist.bruvo, method="complete"))
```
UPGMA and its test

```r
# Calculate it
# Saving as phylo object (and not hclust) gives more
# possibilities for further plotting and manipulations
usflu.upgma <- as.phylo(hclust(d = usflu.dist, method = "average"))
plot.phylo(x = usflu.upgma, cex = 0.75)
title("UPGMA tree")

# Test quality - tests correlation of original distance in the matrix
# and reconstructed distance from hclust object
plot(x = as.vector(usflu.dist), y = as.vector(as.dist(
cophenetic(usflu.upgma))), xlab = "Original pairwise distances",
ylab = "Pairwise distances on the tree", main = "Is UPGMA
appropriate?", pch = 20, col = transp(col = "black",
alpha = 0.1), cex = 2)
# Add correlation line
abline(lm(as.vector(as.dist(cophenetic(usflu.upgma))) ~
as.vector(usflu.dist)), col = "red")
```
UPGMA is not the best choice here...

All points in the right graph should be clustered along the red line...
Neighbor-Joining tree

- One of the oldest methods to reconstruct tree-like relationships among samples/populations, still commonly used
- Quality relies on good distance matrix (must be Euclidean etc.) — choose it well according to your data
- Relatively sensitive to amount of missing data — keep under ca. 10–15% missingness in input data
- Usually much better than hierarchical clustering (UPGMA and similar), can be constructed from any data
- Super fast to compute, easy to do bootstrap test, but usually not so accurate as MP, ML or Bayesian methods
- Suffers of some issues like chaining of individuals in case of low signal in data
- Several method try to reimplement and improve it (see slide 206)
Calculate and test NJ tree

hauss.nj <- nj(hauss.dist)  # Calculates the tree (try various distances)
# Test tree quality - plot original vs. reconstructed distance
plot(as.vector(hauss.dist), as.vector(as.dist(cophenetic(hauss.nj))),
    xlab="Original distance", ylab="Reconstructed distance")
abline(lm(as.vector(hauss.dist) ~
    as.vector(as.dist(cophenetic(hauss.nj)))))
# Testing the correlation
# Linear model for above graph
summary(lm(as.vector(hauss.dist) ~
    as.vector(as.dist(cophenetic(hauss.nj)))))  # Prints summary text
# Plot a basic tree - see ?plot.phylo for details
plot.phylo(x=hauss.nj, type="phylogram")
plot.phylo(x=hauss.nj, type="cladogram", edge.width=2)
plot.phylo(x=hauss.nj, type="fan", edge.width=2, edge.lty=2)
plot.phylo(x=hauss.nj, type="radial", edge.color="red", edge.width=2,
    edge.lty=3, cex=2)  # There are enormous graphical possibilities...
Choose your tree...
# boot.phylo() re-samples all columns - remove population column first
hauss.loci.nopop <- hauss.loci
hauss.loci.nopop[["population"]]<- NULL

# Calculate the bootstrap
hauss.boot <- boot.phylo(phy = hauss.nj, x = hauss.loci.nopop, FUN = function(XXX) nj(dist.gene(loci2genind(XXX)@tab, method="pairwise")), B = 1000)

# boot.phylo returns NUMBER of replicates - NO PERCENTAGE

# Plot the tree
plot.phylo(x = hauss.nj, type = "unrooted", main = "Neighbor-Joining tree")

# Labels for nodes - bootstrap - see ?nodelabels for graphical settings
nodelabels(text = round(hauss.boot/10))

?boot.phylo # See details

# Another possibility
hauss.aboot <- aboot(x = hauss.genind, tree = "nj", distance = nei.dist, sample = 100)  # Bootstrap values are in slot node.label

?aboot # Package poppr
# Plot the tree, explicitly display node labels
plot.phylo(x=hauss.aboot, show.node.label=TRUE)

plot.phylo(x=hauss.aboot, show.node.label=TRUE)

## Plot a nice tree with colored tips
plot.phylo(x=hauss.nj, type="unr", show.tip=F, edge.width=3, main="NJ")

# Labels for nodes - bootstrap - see ?nodelabels for graphical settings
nodelabels(text=round(hauss.boot/10))

# Colored labels - creates vector of colors according to populations
nj.rainbow <- colorRampPalette(rainbow(length(popNames(hauss.genind))))
tiplabels(text=indNames(hauss.genind), bg=fac2col(x=pop(hauss.genind),
  col.pal=nj.rainbow)) # Colored tips

## Plot BW tree with tip symbols and legend
plot.phylo(x=hauss.nj, type="clad", show.tip=F, edge.width=3, main="NJ")
axisPhylo() # Add axis with distances

# From node labels let's remove unneeded frame
nodelabels(text=round(hauss.boot/10), frame="none", bg="white")
Nicer trees

```r
# As tip label we use only symbols - see ?points for graphical details
tiplabels(frame="none", pch=rep(0:4, times=c(13, 17, 2, 6, 9)), lwd=2, cex=2)

# Plot a legend explaining symbols
legend(x="topleft", legend=c("He", "Oh", "Pr", "Ne", "Sk"), 
border="black", pch=0:4, pt.lwd=2, pt.cex=2, bty="o", bg="lightgrey", 
box.lwd=2, cex=1.2, title="Populations")

# See more options...
?plot.phylo
?nodelabels
?legend
?axisPhylo
```

- Functions in `ape` (plot.phylo and others), `adegenet` and another packages provide plenty of options to manipulate and display trees, for users of `ggplot2`, `ggtree` is an interesting alternative
Choose your tree...
Trees based on Bruvo’s distance

Package poppr (bootstrap is incorporated within the function)

```r
# NJ
hauss.nj.bruvo <- bruvo.boot(pop=hauss.genind, replen=rep(2, 12),
  sample=1000, tree="nj", showtree=TRUE, cutoff=1, quiet=FALSE)
plot.phylo(x=hauss.nj.bruvo, type="unrooted", show.tip=FALSE,
  edge.width=3, main="Neighbor-Joining tree")

# Call node labels as phylo$node.labels or phylo[["node.labels"]]
nodelabels(hauss.nj.bruvo[["node.labels"]]) tiplabels(hauss.nj.bruvo
  [["tip.label"]], bg=fac2col(x=hauss.genind$pop, col.pal=nj.rainbow))

# UPGMA
hauss.upgma <- bruvo.boot(pop=hauss.genind, replen=rep(2, 12),
  sample=1000, tree="upgma", showtree=TRUE, cutoff=1, quiet=FALSE)
plot.phylo(hauss.upgma, type="unrooted", show.tip=FALSE, edge.width=3,
  main="UPGMA tree")
nodelabels(hauss.upgma[["node.labels"]])
tiplabels(hauss.upgma[["tip.label"]], bg=fac2col(x=hauss.genind$pop,
  col.pal=nj.rainbow))
```

Vojtěch Zeisek (https://trapa.cz/)
Choose your tree...
**NJ tree of populations**

```r
# aboot() can use distances implemented in poppr:
# Calculations
hauss.nj.pop <- aboot(x=hauss.genpop, tree="nj", distance="nei.dist", sample=1000, showtree=FALSE)
print.phylo(hauss.nj.pop)  # Information about result
# Plot a tree
plot.phylo(x=hauss.nj.pop, type="radial", show.node.label=TRUE, cex=1.2, edge.width=3, main="Neighbor-Joining tree of populations")
```
# Calculate the tree

```r
usflu.tree <- nj(X=usflu.dist)
```

# Plot it

```r
plot.phylo(x=usflu.tree, type="unrooted", show.tip=FALSE)
title("Unrooted NJ tree")
```

# Colored tips

```r
usflu.pal <- colorRampPalette(topo.colors(length(levels(as.factor(usflu.annot[['year']])))))
```

# Tip labels

```r
tiplabels(text=usflu.annot$year, bg=num2col(usflu.annot$year, col.pal=usflu.pal), cex=0.75)
```

# Legend - describing years - pretty() automatically shows best values from given range, num2col() selects colors from color scale

```r
legend(x="bottomright", fill=num2col(x=pretty(x=1993:2008, n=8), col.pal=usflu.pal), leg=pretty(x=1993:2008, n=8), ncol=1)
```
Root the tree

```
# Root the tree - "outgroup" is name of accession (in quotation
# marks) or number (position within phy object)
usflu.tree.rooted <- root.phylo(phy=usflu.tree, outgroup=1)

# Plot it
plot.phylo(x=usflu.tree.rooted, show.tip=FALSE, edge.width=2)
title("Rooted NJ tree")

# Labeling of tips
tiplabels(text=usflu.annot$year, bg=transp(num2col(x=usflu.annot$year,
     col.pal=usflu.pal), alpha=0.7), cex=0.75, fg="transparent")

# Add axis with phylogenetic distance
axisPhylo()

# Legend - describing years - pretty() automatically shows best
# values from given range, num2col() selects colors from color scale
legend(x="topright", fill=num2col(x=pretty(x=1993:2008, n=8),
     col.pal=usflu.pal), leg=pretty(x=1993:2008, n=8), ncol=1)
```
Bootstrap rooted tree

```r
# Calculate it
usflu.boot <- boot.phylo(phy = usflu.tree.rooted, x = usflu.dna, FUN =
  function(EEE) root.phylo(nj(dist.dna(EEE, model="TN93")), outgroup=1),
  B=1000)

# Plot the tree
plot.phylo(x = usflu.tree.rooted, show.tip=FALSE, edge.width=2)
title("NJ tree + bootstrap values")
tiplabels(frame="none", pch=20, col=transp(num2col(x=usflu.annot[["year"]],
  col.pal=usflu.pal), alpha=0.7), cex=3.5, fg="transparent")

# Legend - describing years - pretty() automatically shows best
# values from given range, num2col() selects colors from color scale
legend(x="topright", fill=num2col(x=pretty(x=1993:2008, n=8),
  col.pal=usflu.pal), leg=pretty(x=1993:2008, n=8), ncol=1)

# Plots bootstrap support - note usflu.boot contains raw numbers
# transform it into percent
nodelabels(text=round(usflu.boot/10), cex=0.75)
```

Vojtěch Zeisek (https://trapa.cz/)

Collapse branches with low bootstrap support

```r
usflu.tree.usflu.na.density <- usflu.tree.rooted

# Determine branches with low support - note BS values are in raw numbers - use desired percentage with respect to number of bootstraps
usflu.tocollapse <- match(x=which(usflu.boot < 700)+length(usflu.tree.rooted$tip.label), table=usflu.tree.usflu.na.density$edge[,2])

# Set length of bad branches to zero
usflu.tree.usflu.na.density$edge.length[usflu.tocollapse] <- 0

# Create new tree
usflu.tree.collapsed <- di2multi(usflu.tree.usflu.na.density, tol=0.00001)

# Plot the consensus tree
plot.phylo(x=usflu.tree.collapsed, show.tip=FALSE, edge.width=2)
title("NJ tree after collapsing weak nodes")
tiplabels(text=usflu.annot$year, bg=transp(num2col(x=usflu.annot[["year"]], col.pal=usflu.pal), alpha=0.7), cex=0.5, fg="transparent")
axisPhylo()
legend(x="topright", fill=num2col(x=pretty(x=1993:2008, n=8), col.pal=usflu.pal), leg=pretty(x=1993:2008, n=8), ncol=1)
```

Vojtěch Zeisek (https://trapa.cz/)
The trees
NJ is death. Long live NJ!

- "Basic" NJ has many limitations (problems with missing data, chaining of individuals, ...) — there are several tries to overcome them

- Package `phangorn` has functions `NJ()` and unweighted version `UNJ()`

- Package `ape` has functions `njs()` and `bionjs()` which are designed to perform well on distances with (more) missing values

- Function `bionj()` from `ape` implements BIONJ algorithm

- FastME functions (package `ape`) perform the minimum evolution algorithm and aim to be replacement of NJ — read `?fastme` before use

- All those functions read distance matrix and their usage is same as with "classical" `nj()` (read manual pages before using them) — it is also from package `ape`
Principal Coordinate Analysis (PCoA), Principal Component Analysis (PCA) and relatives

- Create 2-D scatter plot showing relationships among samples and their grouping
- PCoA is variant of PCA using distance matrix as input
  - It’s crucial to select correct Euclidean distance matrix
- PCA is using as input data frame with any data type, but results are unstable for large number of variables
  - In genetic data, each locus (position in alignment, SSRs locus, ...) is separate variable from technical point of view
  - PCA is unstable if there are more variables than samples
- These methods are exploratory
  - For creating hypothesis, not testing them
  - There are no tests involved, no bootstraps, nothing like that
- There are variants of PCA for spatial data, for analysis of traits, etc.
- More such multivariate methods are used in another fields, like NMDS in ecology, ...
PCoA I

```r
hauss.pcoa <- dudi.pco(dist.gene(x=scaleGen(x=hauss.genind, center=TRUE, scale=FALSE, truenames=TRUE), method="pairwise"), scannf=FALSE, nf=3)
s.label(dfxy=hauss.pcoa$li, clabel=0.75)  # # Basic display
# To plot different axes use for example dfxy=hauss.pcoa$li[c(2, 3)]
s.kde2d(dfxy=hauss.pcoa$li, cpoint=0, add.plot=TRUE)  # # Add kernel density
# Add histogram of Eigenvalues
add.scatter.eig(w=hauss.pcoa$eig, nf=3, xax=1, yax=2, posi="topleft",
                sub="Eigenvalues")
# Percentage of variance explained by each PC axis
100*hauss.pcoa$eig/sum(hauss.pcoa$eig)
# Colored display according to populations
# Creates vector of colors according to populations
hauss.pcoa.col <- rainbow(length(popNames(hauss.genind)))
s.class(dfxy=hauss.pcoa$li, fac=pop(hauss.genind), col=hauss.pcoa.col)
add.scatter.eig(w=hauss.pcoa$eig, nf=3, xax=1, yax=2, posi="bottomleft",
                sub="Eigenvalues")
title("Principal Coordinates Analysis")  # # Adds title to the graph
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
208/421
# Based on Bruvo's distance
hauss.pcoa.bruvo <- dudi.pco(d = bruvo.dist(pop = hauss.genind, replen = rep(2, 12)),
   scannf = FALSE, nf = 3)
s.class(dfxy = hauss.pcoa.bruvo$li, fac = pop(hauss.genind),
   col = hauss.pcoa.col)
add.scatter.eig(hauss.pcoa.bruvo$eig, posi = "bottomright", 3, 1, 2)
# Another possibility for colored
# plot (see ?colorplot for details)
colorplot(xy = hauss.pcoa$li[c(1, 2)],
   X = hauss.pcoa$li, transp = TRUE,
   cex = 3, xlab = "PC 1", ylab = "PC 2")
title(main = "PCoA, axes 1 and 2")
abline(v = 0, h = 0, col = "gray", lty = 2)
PCoA — Bruvo and colorplot
Process more data
Not all combinations and possibilities were shown...

Tasks

1. Most of examples of basic analysis were shown with the *Taraxacum haussknechtii* microsatellite dataset — try to do some analysis with another imported data
2. Try some of the introduced analysis with your own custom imported data
3. Try at least 2–3 analysis according to your interests

Note...

- Of course, following chapters will show more possible analysis...
- Previous examples are not covering all possibilities...
- It is crucial to be able to edit the introduced commands to be able to handle your data
- Check help pages of the functions for more options what to do with your data
Single Nucleotide Polymorphism
Special methods for large next-generation sequencing data

6 SNP

PCA and NJ

- Large whole-genome sequencing data (usually from Illumina sequencing) use to be imported as VCF (and converted to genlight) or read from FASTA directly into genlight — it’s designed to efficiently store and process large datasets

- Several special functions are available to directly work with genlight, taking advantage of its design, and using multiple CPU cores (on Linux and macOS)
Special functions to work with huge SNP data sets

```r
# Plot of missing data (white) and number of 2nd alleles
glPlot(x = usflu.genlight, legend = TRUE, posi = "topleft")

# Sum of the number of second allele in each SNP
usflu.freq <- glSum(usflu.genlight)

# Plot distribution of (second) allele frequencies
hist(x = usflu.freq, proba = TRUE, col = "gold", xlab = "Allele frequencies",
     main = "Distribution of (second) allele frequencies")
lines(x = density(usflu.freq)$x, y = density(usflu.freq)$y * 1.5, col = "red",
      lwd = 3)

# Mean number of second allele in each SNP
usflu.mean <- glMean(usflu.genlight)
usflu.mean <- c(usflu.mean, 1 - usflu.mean)

# Plot distribution of allele frequencies
hist(x = usflu.mean, proba = TRUE, col = "darkseagreen3", xlab = "Allele frequencies",
     main = "Distribution of allele frequencies", nclass = 20)
lines(x = density(usflu.mean, bw = 0.05)$x, y = density(usflu.mean, bw = 0.05)$y * 2, lwd = 3)
```
Number of missing values in each locus

```r
# Play with bw parameter to get optimal image
usflu.na.density <- density(glNA(usflu.genlight), bw=10)

# Set range of xlim parameter from 0 to the length of original alignment
plot(x=usflu.na.density, type="n", xlab="Position in the alignment",
     main="Location of the missing values (NA)", xlim=c(0, 1701))
polygon(c(usflu.na.density$x, rev(usflu.na.density$x)),
       c(usflu.na.density$y, rep(0, length(usflu.na.density$x)));
       col=transp("blue", alpha=0.3))
points(glNA(usflu.genlight), rep(0, nLoc(usflu.genlight)), pch="|", cex=2,
       col="blue")
```

- Those tools are designed mainly for situation when having multiple (nearly) complete genomes — not needed for smaller data sets
- Lets keep hoping in fast development of computers...
- **Windows users:** To speed up the processing, `gl*` functions use parallelisation library unavailable on Windows — add parameter `parallel=FALSE` to be able to use them.
Basic information about SNP: distribution of $2^{nd}$ allele frequencies, missing data and number of $2^{nd}$ allele, distribution of allele frequencies, and number of missing values in each locus.
PCA, NJ and genlight objects

```r
usflu.pca <- glPca(x = usflu.genlight, center = TRUE, scale = FALSE, loadings = TRUE) # Select number of retained PC axes, about 10 here
scatter.glPca(x = usflu.pca, posi = "bottomright") # Plot PCA
title("PCA of the US influenza data")
# Loading plot - contribution of variables to the pattern observed
loadingplot.glPca(x = usflu.pca)
colorplot(usflu.pca$scores, usflu.pca$scores, transp = TRUE, cex = 4) # Cols
title("PCA of the US influenza data")
abline(h = 0, v = 0, col = "gray")
add.scatter.eig(usflu.pca[["eig"]][1:40], 2, 1, 2, posi = "topright",
    inset = 0.05, ratio = 0.3)
usflu.tree.genlight <- nj(dist.gene(as.matrix(usflu.genlight))) # Get tree
# Plot colored phylogenetic tree
plot.phylo(x = usflu.tree.genlight, type = "fan", show.tip = FALSE)
tiplabels(pch = 20, col = num2col(usflu.annot[["year"]], col.pal = usflu.pal), cex = 4)
title("NJ tree of the US influenza data")
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
216/421
PCA, NJ and genlight objects

```r
legend(x="topright", fill=num2col(x=pretty(x=1993:2008, n=8), col.pal=usflu.pal), leg=pretty(x=1993:2008, n=8), ncol=1)
```
Discriminant Analysis of Principal components

7 DAPC
   Bayesian clustering
   Discriminant analysis and visualization
   Tasks

1 Bayesian clustering on data pre-processed by PCA
2 Discriminant analysis using this above clustering and original data
### DAPC

- Discriminant Analysis of Principal components ([Jombart et al. 2010](https://doi.org/10.1093/molbev/msq122))
- Runs K-means Bayesian clustering on data transformed with PCA (reduces number of variables, speeds up process)
- User selects best $K$ — number of clusters; according to scores shown
- Finally it runs discriminant analysis (DA) to maximize differences among groups
- Various modes of displaying of results — “Structure-like”, “PCA-like” and more
- More information at [https://adegenet.r-forge.r-project.org/](https://adegenet.r-forge.r-project.org/)
- If following commands would seem too complicated, try web interface

```r
library(adegenet)
adegenetServer("DAPC")  # Recommended to open in Google Chrome/Chromium
adegenetTutorial("dapc")  # Tutorial, more information about DAPC
```
Principal difference between PCA and DA

PCA searches for the direction showing the largest total variance. PCA fails to discriminate the groups.

DA maximizes the separation between groups while minimizing variation within group. DA displays group differences.

Principal difference between PCA and DA:

- PCA searches for the direction showing the largest total variance.
- PCA fails to discriminate the groups.
- DA maximizes the separation between groups while minimizing variation within group.
- DA displays group differences.
K-find — Bayesian K-means clustering

```r
# Retain all informative PC (here about 35)
# According to second graph select best K (here 2 or 3)
# Now we select K=2 and later rerun the analysis for K=3 (lines 14-18)
hauss.kfind <- find.clusters(x=hauss.genind, stat="BIC",
    choose.n.clust=TRUE, max.n.clust=10, n.iter=100000, n.start=100,
    scale=FALSE, truenames=TRUE)
table(pop(hauss.genind), hauss.kfind$grp) # See results as text

# Graph showing table of original and inferred populations and
# assignment of individuals

hauss.kfind

table.value(df=table(pop(hauss.genind), hauss.kfind$grp), col.lab=
    paste("Inferred\ncluster", 1:length(hauss.kfind$size)), grid=TRUE)

# For K=3 - note parameters n.pca and n.clust - we just rerun the
# analysis and when results are stable, no problem here

hauss.kfind3 <- find.clusters(x=hauss.genind, n.pca=35, n.clust=3,
    stat="BIC", choose.n.clust=FALSE, n.iter=100000, n.start=100,
    scale=FALSE, truenames=TRUE)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
K-find outputs

- Cumulative variance of axis
- BIC helps to select the best K
- Original and inferred groups

```r
# See results as text
table(pop(hauss.genind), hauss.kfind3$grp)

# Graph showing table of original and inferred populations and assignment of individuals
table.value(
  df = table(pop(hauss.genind), hauss.kfind3$grp),
  col.lab = paste("Inferred\ncluster", 1:length(hauss.kfind3$size)),
  grid = TRUE)

# If needed, use custom text for parameter col.lab=c("...", "...")
# as many labels as inferred groups
```


DAPC code I

```r
## K=2
# Create DAPC
# Number of informative PC (Here 15, adegenet recommends < N/3). Select
# number of informative DA (here only one is available - no PCA graph)
hauss.dapc <- dapc(x=hauss.genind, pop=hauss.kfind$grp, center=TRUE,
  scale=FALSE, var.contrib=TRUE, pca.info=TRUE, truenames=TRUE)

# Information
hauss.dapc

# Density function - only for first axis here!
scatter(x=hauss.dapc, xax=1, yax=1, main="DAPC", bg="white", solid=0.5,
  leg=TRUE, txt.leg=c("Group 1", "Group 2"), posi.leg="topright")

# Assignment of individuals to clusters
assignplot(x=hauss.dapc)

# Structure-like plot
compoplot(x=hauss.dapc, xlab="Individuals", leg=FALSE)

# Loadingplot - alleles the most adding to separation of individuals
loadingplot(x=hauss.dapc$var.contr)
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R
DAPC for K=2
DAPC code II

```r
# alfa-score - according to number of PC axis
optim.a.score(x=hauss.dapc)
## K=3
# Create DAPC
# Number of informative PC (Here 15, adegenet recommends < N/3)
# Select number of informative DA (here 2 - usually keep all of them)
hauss.dapc3 <- dapc(x=hauss.genind, pop=hauss.kfind3$grp, center=TRUE,
                    scale=FALSE, var.contrib=TRUE, pca.info=TRUE, truenames=TRUE)
hauss.dapc # Information
# A la PCA graph
scatter(x=hauss.dapc3, main="DAPC, Taraxacum haussknechtii",
        bg="white", cex=3, clab=0, col=rainbow(3), posi.da="bottomleft",
        scree.pca=TRUE, posi.pca="bottomright", leg=TRUE,
        txt.leg=c("Group 1", "Group 2", "Group 3"), posi.leg="topleft")
```

- Especially graphical parameters have huge possibilities...
- See `?scatter` and play with it...
DAPC for K=3
DAPC code III

# Same in BW
scatter(x=hauss.dapc3, main="DAPC, Taraxacum haussknechtii",
bg="white", pch=c(15:17), cell=0, cstar=0, solid=1, cex=2.5, clab=0,
col=gray.colors(3, start=0, end=0.8, gamma=2, alpha=0), posi.da="bottomleft",
scree.pca=TRUE, posi.pca="bottomright", leg=TRUE,
txt.leg=c("Group 1", "Group 2", "Group 3"), posi.leg="topleft")

# Density function - only for first axis here!
scatter(x=hauss.dapc3, xax=1, yax=1, main="DAPC", bg="white", solid=0.5,
leg=T, txt.leg=c("Group 1", "Group 2", "Group 3"), posi.leg="topleft")

# Assignment of individuals to clusters
assignplot(hauss.dapc3)

# Structure-like plot
compoplot(hauss.dapc3, xlab="Individuals", leg=FALSE)

# Loadingplot - alleles the most adding to separation of individuals
loadingplot(hauss.dapc3$var.contr)

# alfa-score - according to number of PC axis
optim.a.score(hauss.dapc3)
DAPC for K=3, extra information
Another DAPC example
Try DAPC

Tasks

1. Try DAPC with `microbov` dataset (`data(microbov)`, see `?microbov`), the U.S. flu dataset (the `usflu.genind` object), or some other data according to your choice.
   - Try K-means clustering as well as DAPC itself.
   - Try various Ks (if appropriate).
   - Try various displays.

2. Interpret the results.

3. When is DAPC good tool and when should you avoid it?

4. Which data can you cluster using K-means clustering?
Spatial analysis and genetic data
Correlation of genes and space, spatial structure of genotypes

Genes in spatial context...

- Spatial analysis
  - Moran’s I
  - sPCA
  - Mantel test
  - Monmonier
  - Geneland
  - Plotting maps
  - Tasks

Molecular data in R

Vojtěch Zeisek (https://trapa.cz/)

January 31 to February 3, 2022

231/421
Short overview of spatial genetics (in R)

Basic approaches

- **Moran’s I** — several implementations, generally it is autocorrelation coefficient with broader use
  - As “classical” correlation index, spatial PCA, Monmonier’s algorithm searching for genetic boundaries, ...

- **Mantel test** — several implementations, popular, although recently criticized as biologically irrelevant, generally correlation of two matrices (here genetic and geographical)

- Bayesian clustering using geographical information as a proxy and showing results in geographical context (here as implemented in Geneland)

- Plenty of options with plotting maps, including interactive maps (e.g. with R Leaflet)

- There are unlimited possibilities with connections with GIS software — check specialized courses and literature...
Moran’s I

- “Only” autocorrelation index — no genetic/evolutionary model involved — sometimes criticized as biologically irrelevant mechanism
- Used in many variants for plenty of applications
- This (or similar) approach can be used to test correlation between another characteristics (typically used in ecology or evolutionary studies)
- Calculations are done according to matrix of geographic distances, or connectivity network connecting individuals/populations (created by chooseCN) — carefully check its options and try several parameters
- Pay attention which hypothesis is tested (i.e. if lower, greater or two-sided) — similar to T-test
library(adegenet) # Load required libraries
library(spdep)

# Creates connection network
hauss.connectivity <- chooseCN(xy=hauss.genind$other$xy, type=5, d1=0,
    d2=1, plot.nb=TRUE, result.type="listw", edit.nb=FALSE)

# Test of Moran's I for 1st PCoA axis
# Results can be checked against permuted values of moran.mc()
moran.test(x=hauss.pcoa[["li"]][,1], listw=hauss.connectivity,
    alternative="greater", randomisation=TRUE)

Moran's I test under randomisation
data: hauss.pcoa$li[, 1]
Moran I statistic standard deviate = -18.514, p-value = 1
alternative hypothesis: greater
sample estimates:
Moran I statistic  Expectation  Variance
   -0.5232003724   -0.0217391304   0.0007336276

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Calculation of Moran's I / II

```r
# Test of Moran's I for 1st PCoA axis
hauss.pcoa1.mctest <- moran.mc(x=hauss.pcoa$li[,1],
                              listw=hauss.connectivity, alternative="greater", nsim=1000)
hauss.pcoa1.mctest
# Output:
# Monte-Carlo simulation of Moran I
data:  hauss.pcoa$li[, 1] weights:  hauss.connectivity
number of simulations + 1: 1001
statistic = -0.5163, observed rank = 1, p-value = 0.999
alternative hypothesis: greater
# Plot the results
plot(hauss.pcoa1.mctest) # Plot of density of permutations
moran.plot(x=hauss.pcoa$li[,1], listw=hauss.connectivity) # PC plot
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Moran’s $I$ for our 1\textsuperscript{st} axis isn’t significant

- Tested hypothesis “greater” — no significant positive autocorrelation
- If testing for hypothesis “less” — significant negative autocorrelation — individuals are significantly different
Calculation of Moran’s I \( (2^{nd} \text{ axis}) \)

```r
# Test of Moran's I for 2nd PCoA axis
moran.test(x=hauss.pcoa$li[,2], listw=hauss.connectivity,
    alternative="greater", randomisation=TRUE)

hauss.pcoa2.mctest <- moran.mc(x=hauss.pcoa$li[,2],
    listw=hauss.connectivity, alternative="greater", nsim=1000)

hauss.pcoa2.mctest

# Output
Monte-Carlo simulation of Moran's I

data:  hauss.pcoa$li[, 2]
weights:  hauss.connectivity

number of simulations + 1: 1001
statistic = 0.0545, observed rank = 1001, p-value = 0.000999

alternative hypothesis: greater

# Plot the results
plot(hauss.pcoa2.mctest)  # Plot of density of permutations
moran.plot(x=hauss.pcoa$li[,2], listw=hauss.connectivity)  # PC plot
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
237/421
Second axis is surprisingly significant

- Tested hypothesis “greater” — there is significant positive autocorrelation — individuals are genetically similar over space.
Reading and limits on Moran’s I

- 1\textsuperscript{st} and 2\textsuperscript{nd} PC axes are orthogonal to each other, so they show different patterns.
- Moran’s I show single index over all data (whole space), but different processes might be going on on large or small scale — commonly negative autocorrelation on large scale (distant individuals are significantly dissimilar) and positive autocorrelation on small scale (individuals, which are close to each other are significantly similar).
- Single number isn’t always the best description of complex biological situation...
- Calculations using \texttt{moran.test} are done in connectivity network created by \texttt{chooseCN} — it’s crucial to select connectivity network reflecting biological features of studied species and its geographic situation.
Spatial Analysis of Principal Components (sPCA)

- Implemented in adegenet, see `adegenetTutorial("spca")`
- Analyzes matrix of relative allele frequencies of genotypes/populations and spatial weighting matrix
- The geographical matrix is usually (as for Moran’s I) created by `chooseCN()` — creates connectivity network among entities (genotypes/populations) — spatial coordinates are not directly used
- When using `chooseCN()`, look at the documentation and try various methods with changing settings to see differences

```r
# adegenet's toy dataset, Rupicapra rupicapra from French Alps
data(rupica)  # adegenet's toy dataset, Rupicapra rupicapra from French Alps
library(adespatial)  # Part of sPCA calculations are in adespatial

# Try various settings for chooseCN (type=X) - type 1-4 as there are identical coordinates (multiple sampling from same locality)
?chooseCN  # See for more details - select the best "type" for your data
chooseCN(xy=rupica$other$xy, ask=TRUE, type=5/6/7, plot.nb=TRUE, edit.nb=FALSE, ... )  # Play with settings little bit...
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R  
January 31 to February 3, 2022  
240/421
Calculations of sPCA

```r
hauss.spca <- spca(obj=hauss.genind, cn=hauss.connectivity, 
    scale=TRUE, scannf=TRUE)

# Plot eigenvalues of sPCA - global vs. local structure
barplot(height=hauss.spca$eig, main="Eigenvalues of sPCA", 
    col=spectral(length(hauss.spca$eig)))
legend("topright", fill=spectral(2), leg=c("Global structures", 
    "Local structures")) # Add legend
abline(h=0, col="gray") # Add line showing zero
print.spca(hauss.spca) # Information about sPCA
summary.spca(hauss.spca) # Summary of sPCA results
# Shows connectivity network, 3 different scores
# barplot of eigenvalues and eigenvalues decomposition
plot.spca(hauss.spca)
colorplot.spca(hauss.spca, cex=3) # Display of scores in color canals
title("sPCA - colorplot of PC 1 and 2 (lagged scores)", line=1, cex=1.5)
# Spatial and variance components of the eigenvalues
screeplot.spca(x=hauss.spca, main=NULL)
```
sPCA outputs I
Test if global/local structure is significant

```r
hauss.spca.glo <- global.rtest(X=hauss.genind$tab, listw=hauss.spca$lw, nperm=999)
hauss.spca.glo
plot(hauss.spca.glo)
hauss.spca.loc <- local.rtest(X=hauss.genind$tab, listw=hauss.spca$lw, nperm=999)
hauss.spca.loc
plot(hauss.spca.loc)
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

243/421
Map of genetic clines

```r
library(akima) # It is needed for manipulation with coordinates
# Transform the coordinates

hauss.spca.temp <- interp(other(hauss.genind)$xy[,1],
other(hauss.genind)$xy[,2], hauss.spca$ls[,1],
xo=seq(min(other(hauss.genind)$xy[,1]), max(other(hauss.genind)$xy[,1]), le=200),
max(other(hauss.genind)$xy[,1]), le=200),
duplicate="median")

# For 1st axis
image(x=hauss.spca.temp, col=spectral(100))
s.value(dfxy=hauss.genind$other$xy, z=hauss.pcoa$li[,1], add.p=TRUE,
csize=0.5, sub="PCoA - first PC", csub=2, possub="topleft")

# For 2nd axis
image(x=hauss.spca.temp, col=spectral(100))
s.value(dfxy=hauss.genind$other$xy, z=hauss.pcoa[['li']][,2],
add.p=TRUE, csize=0.5, sub="PCoA - second PC", csub=2, possub="topleft")
```
sPCA outputs II

```r
# Interpolated lagged score on a map
hauss.spca.annot <- function() {
  title("sPCA - interpolated map of individual scores")
  points(other(hauss.genind)$xy[,1], other(hauss.genind)$xy[,2])
}
filled.contour(hauss.spca.temp, color.pal=colorRampPalette(lightseasun(100)), pch=20, nlevels=100, key.title=title("Lagged
score 1"), plot.title=hauss.spca.annot())
```

![Interpolated lagged score on a map](attachment:image.png)
Loading plots — which alleles contribute the most?

```r
hauss.spca.loadings <- hauss.spca[["c1"]][,1]^2
names(hauss.spca.loadings) <- rownames(hauss.spca$c1)
loadingplot(x=hauss.spca.loadings, xlab="Alleles", ylab="Weight of the alleles", main="Contribution of alleles to the first sPCA axis")
boxplot(formula=hauss.spca.loadings~hauss.genind$loc.fac, las=3, ylab="Contribution", xlab="Marker", main="Contribution by markers into the first global score", col="gray")
```
Mantel test

- Originally created for biomedicine to test correlation between treatment and diseases
- “Only” correlation of two matrices — no biologically relevant underlying model — because of that it is heavily criticized (mainly in ecology)
- It is universal method usable for plenty of tasks
- Test of spatial and genetic relationships is probably one of few biologically relevant applications
- Package vegan (set of ecological tools) has implementation to test genetic similarity in various distance classes — not only overall result — very useful
- Distance matrix can be calculated as simple Euclidean geometric distance by `dist()`, or for larger areas by geodesic distance along Earth surface (e.g. `pegas::geod()`), or using special package like geosphere

```r
library(pegas) # Geodesic distance
```
Mantel test — isolation by distance

```r
# Geographical distance
hauss.gdist <- as.dist(m=geod(lon=hauss.genind$other$xy$lon, 
                          lat=hauss.genind$other$xy$lat), diag=TRUE, upper=TRUE)

# Mantel test
hauss.mantel <- mantel.randtest(m1=hauss.dist, m2=hauss.gdist, nrepet=1000)
hauss.mantel # See text output
plot(hauss.mantel, nclass=30)

# Libraries required by mantel.correlog:
library(permute)
library(lattice)
library(vegan)

# Different implementation of Mantel test testing distance classes
hauss.mantel.cor <- mantel.correlog(D.eco=hauss.dist, D.geo=hauss.gdist, 
                                      XY=NULL, n.class=0, break.pts=NULL, cutoff=FALSE, r.type="pearson", 
                                      nperm=1000, mult="holm", progressive=TRUE)
hauss.mantel.cor # See results for respective classes
summary(hauss.mantel.cor)
```
Molecular data in R

Vojtěch Zeisek (https://trapa.cz/)

January 31 to February 3, 2022

Introduction
R
Data
Alignment
Basic analysis
SNP
DAPC
Spatial analysis
Trees
Evolution
The end

Moran’s I
sPCA
Mantel test
Monmonier
Geneland
Plotting maps
Tasks

Mantel test outputs — strongly significant

```
hauss.mantel # See output
Monte-Carlo test
Call: mantel.randtest(m1 =
  hauss.dist, m2 =
  hauss.gdist, nrepet = 1000)
Observation: 0.35409
Based on 1000 replicates
Simulated p-value: 0.000999001
Alternative hypothesis: greater

Std.Obs  Expectation  Variance
7.61967545  0.001687140  0.0021389
```

# Plot correlogram (next slide)
plot(hauss.mantel.cor)

Vojtěch Zeisek (https://trapa.cz/)
Correlation (genetic similarity) in several distance classes (positive [up] in short distance [left], negative [down] in long [right]; [full] — significant, [empty] — not significant) — see ?mantel.correlog for details.
Mantel correlogram — text output

```r
hauss.mantel.cor # See the text output:
Mantel Correlogram Analysis
Call:
mantel.correlog(D.eco = hauss.dist, D.geo = hauss.gdist, XY = NULL,
              n.class = 0, break.pts = NULL, cutoff = FALSE, r.type = "pearson",
              nperm = 1000, mult = "holm", progressive = TRUE)
class.index   n.dist Mantel.cor Pr(Mantel) Pr(corrected)
D.cl.1    4.697165 532.000000 0.400101     0.0010     0.000999 ***
D.cl.2   14.091494  0.000000  NA       NA          NA
D.cl.3  23.485823  0.000000  NA       NA          NA
D.cl.4  32.880153  52.000000 0.079433     0.0490     0.048951 *
D.cl.5  42.274482 466.000000 -0.128236    0.0010    0.002997 **
D.cl.6  51.668811  0.000000  NA       NA          NA
D.cl.7  61.063140  36.000000 -0.027890    0.2547    0.254745
...
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Vojtěch Zeisek (https://trapa.cz/)  Molecular data in R  January 31 to February 3, 2022  251/421
```
Monmonier’s algorithm — genetic boundaries

- Finds boundaries of maximum differences between contiguous polygons of a tessellation
- Detects genetic boundaries among georeferenced genotypes (or populations)
- For more information see `adegenetTutorial("basics")`
- Requires every point to have unique coordinates — in case of population data it is better to work with populations, not individuals (but it is not ideal)
- It uses **Voronoi tessellation** — it is used by a lot of spatial analysis, especially by tools relying on Bayesian statistics (like Geneland, slide 57)

```r
# Calculates Monmonier's function (for threshold use 'd')
hauss.monmonier <- monmonier(xy=hauss.genpop$other$xy, dist=dist(hauss.genpop$tab), cn=chooseCN(hauss.genpop$other$xy, ask=FALSE, type=6, k=2, plot.nb=FALSE, edit.nb=FALSE), nrun=1)
coords.monmonier(hauss.monmonier) # See result as text
```
Voronoi tessellation

- In simplest case, all points have certain area and all points within this area are closer to the respective “main” point than to any other “neighbor” point.

- Extreme differences among size of areas make computational problems and results are unstable — this typically occurs when calculations are done on individual level and there are large distances among populations.

- The more similar sizes of polygons and the more even distribution of samples, the more stable and reliable results we get.
Plot genetic boundaries

```r
plot.monmonier(hauss.monmonier,
              add.arrows = FALSE, bwd=10,
              sub="Monmonier plot", csub=2)
points(hauss.genpop$other$xy, cex=2.5, pch=20, col="red")
text(x=haust.genpop$other$xy$lon, y=haust.genpop$other$xy$lat,
     labels=popNames(hauss.genpop), cex=3)
legend("bottomright",
       leg="Genetic boundaries\namong populations")
```

# For plotting see
?points
?text
?legend
Monmonier notes

• Sometimes it is needed to get rid of (random) noise in data. To do so use as parameter `dist` of `monmonier()` table from PCA (`pcaObject$li`) by something like:

```r
monmonier(..., dist=dudi.pco(d=dist.gene(x=GenindObject$tab), scannf=FALSE, nf=1)$li, ...)
```

• Generally (when dataset is bigger and more diverse) it is recommended to run it several times (parameter `nrun`) — there will be several iterations

• When running `monmonier(...)` when it asks for threshold of sorted local distances, try several values and see differences in output

• See `?plot.monmonier` for various graphical parameters to customize the plot

• Use `points()` to add for example colored symbols of samples and/or `text()` to add text labels
Introduction
R Data Alignment Basic analysis SNP DAPC Spatial analysis Trees Evolution The end

About Geneland

- For installation see slide 57
- Works with haploid and diploid co-dominant markers (microsatellites or SNPs)
- Spatially explicit Bayesian clustering
- Produces maps of distribution of inferred genetic clusters
- Relative complicated tool with various modeling options
- For more information see
  https://i-pri.org/special/Biostatistics/Software/Geneland/

```r
# Load needed libraries
library(PBSmapping) # Required to transform coordinates
library(Geneland)
# Graphical interface is available, we will use only command line...
Geneland.GUI()
```

Vojtěch Zeisek (https://trapa.cz/)
Geneland GUI

- Some tasks are easier in GUI, some in command line...
- Command line is great for its repeatability...
- Always read manual! It is not the simplest tool...

![Geneland GUI screenshot](image-url)
Loading and conversions of coordinates

```r
# Geneland requires specific coordinate space
# hauss.cord is DF, we need just plain matrix
hauss.geneland.coord <- as.matrix(hauss.coord)
colnames(hauss.geneland.coord) <- c("X", "Y")
attr(hauss.geneland.coord, "projection") <- "LL"
attr(hauss.geneland.coord, "zone") <- NA

hauss.geneland.coord.utm <- convUL(hauss.geneland.coord)
dim(hauss.geneland.coord)
hauss.geneland.coord
dim(hauss.geneland.coord.utm)

hauss.geneland.coord.utm # Final coordinates
```

```r
# Load data (only haploid or diploid data are supported)
# only plain table with alleles
hauss.geneland.data <- read.table(file= "https://soubory.trapa.cz/rcourse/haussknechtii_geneland.txt", na.string="-999", header = FALSE, sep="\t")
dim(hauss.geneland.data)
hauss.geneland.data
```
Before running MCMC

- Monte Carlo Markov Chains (MCMC) require usually millions of generations (iterations, \texttt{nit}) to find optimal solution
- Beginning (\(\sim 10\text{–}20\%\)) of the steps (\texttt{burnin}) use to be very unstable and useless for following analysis and it is discarded
- Geneland allows to set density of sampling among generations (\texttt{thinning}) — it is not necessary to sample (save) every generation

- Within millions of generations we can sample every 1000\text{–}10000^{th} generation
- Denser sampling produces smoother data, but can consume too much disk space...

Directory structure for Geneland:

```
  geneland/
        1  admixture
        2  admixture
        3  admixture
```
Settings and running MCMC

```r
hauss.geneland.nrun <- 5  # Set number of independent runs
hauss.geneland.burnin <- 100  # Set length of burnin chain
hauss.geneland.maxpop <- 10  # Set maximal K (number of populations)

# FOR loop will run several independent runs and produce output maps
# of genetic clusters - outputs are written into subdirectory within
# geneland directory (this has to exist prior launching analysis)
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/", hauss.geneland.irun, "/", sep="")  # paste is good especially for joining several texts
  # On Windows, remove following line and create subdirectories from
  # 1 to max K manually (creating subdirs in Windows in R is complicated)
  system(paste("mkdir ", hauss.geneland.path.mcmc))  # Creates subdirs
  # Inference - MCMC chain - see ?MCMC for details
  MCMC(coordinates=hauss.geneland.coord.utm, geno.dip.codom=hauss.geneland.data, path.mcmc=hauss.geneland.path.mcmc, delta.coord=0.001, varnpop=TRUE, npopmin=1, npopmax=hauss.geneland.maxpop, nit=10000, thinning=10, freq.model="Uncorrelated", spatial=TRUE)  # Loop continues on next slide
}
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Molecular data in R
January 31 to February 3, 2022
# Start of FOR loop is on previous page. In practice set much higher
# number of iterations (nit, millions), appropriate sampling (thinning, #
# thousands) and longer burnin. Post-process chains
PostProcessChain(coordinates=hauss.geneland.coord.utm, path.mcmc=hauss.
geneland.path.mcmc, nxdom=500, nydom=500, burnin=hauss.geneland.burnin)

# Output
# Simulated number of populations
Plotnpop(path.mcmc=hauss.geneland.path.mcmc, printit=TRUE,
  file=paste(hauss.geneland.path.mcmc, "/geneland-number_of_clusters
  .pdf", sep=""), format="pdf", burnin=hauss.geneland.burnin)
dev.off()  # We must close graphical device manually

# Map of estimated population membership
PosteriorMode(coordinates=hauss.geneland.coord.utm,
  path.mcmc=hauss.geneland.path.mcmc, printit=TRUE, format="pdf",
  file=paste(hauss.geneland.path.mcmc, "/geneland-map.pdf", sep=""))
dev.off()  # We must close graphical device manually
}

# End of FOR loop from previous slide
Estimate $F_{ST}$

```r
# Prepare list to record values of Fst for all runs
hauss.geneland.fstat <- list()

# Estimate Fst
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/",
                                    hauss.geneland.irun, "/", sep="")
  hauss.geneland.fstat[[hauss.geneland.irun]] <- Fstat.output(
    coordinates=hauss.geneland.coord.utm,
    genotypes=hauss.geneland.data,
    burnin=hauss.geneland.burnin, ploidy=2,
    path.mcmc=hauss.geneland.path.mcmc)
}

hauss.geneland.fstat # Print Fst output
```

- Probably one of few implementations of $F_{ST}$ using Bayesian statistics (and geographically explicit model)

Vojtěch Zeisek (https://trapa.cz/)
MCMC inference under the admixture model

```r
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/",
                                   hauss.geneland.irun, "/", sep="")
  hauss.geneland.path.mcmc.adm <- paste(hauss.geneland.path.mcmc,
                                          "admixture", "/", sep="")
  # On Windows, remove following line of code and create in each
  # result directory (from 1 to max K) new subdirectory "admixture"
  # (creating subdirs in Windows in R is complicated)
  system(paste("mkdir ", hauss.geneland.path.mcmc.adm))
  HZ(coordinates=hauss.geneland.coord.utm, geno.dip.codom=
      hauss.geneland.data, path.mcmc.noadm=hauss.geneland.path.mcmc,
      nit=10000, thinning=10,
      path.mcmc.adm=hauss.geneland.path.mcmc.adm)
}
```

• Currently, there is no much use for admixture results, at lest not without extra work...

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

263/421
Produce maps of respective inferred clusters

```r
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/",
    hauss.geneland.irun, "/", sep="")
  # Maps - tessellations
  PlotTessellation(coordinates=hauss.geneland.coord.utm,
    path.mcmc=hauss.geneland.path.mcmc, printit=TRUE,
    path=hauss.geneland.path.mcmc)
  for (hauss.geneland.irun.img in 1:hauss.geneland.maxpop) {
    dev.off() } # We must close graphical device manually
}
```

- Maps are produced as PS (PostScript) files in output directories
- Not every graphical software can handle PS (try for example GIMP)
- There are as many plots as was maximal K, but only those up to inferred number of clusters have some content (the others are empty)
Estimate frequencies of null alleles

```r
hauss.geneland.fna <- list()
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/", hauss.geneland.irun, "/", sep="")
  # Estimation
  hauss.geneland.fna[[hauss.geneland.irun]] <- EstimateFreqNA(path.mcmc=hauss.geneland.path.mcmc)
}
# See output
hauss.geneland.fna
```

- Each item of the `list` object `hauss.geneland.fna` (from 1 to number of runs) contains vector of estimated frequencies of null alleles for every locus.
- Estimation on null alleles (one from two alleles is missing, behaving like homozygote — `A-` instead of `AA`) is generally difficult, so this is nice and unique feature.
Determine which run is the best

```r
# Calculate average posterior probability
hauss.geneland.lpd <- rep(NA, hauss.geneland.nrun)
for (hauss.geneland.irun in 1:hauss.geneland.nrun) {
  hauss.geneland.path.mcmc <- paste("geneland/", hauss.geneland.irun, "/", sep="")
  hauss.geneland.path.lpd <- paste(hauss.geneland.path.mcmc, "log.posterior.density.txt", sep="")
  hauss.geneland.lpd[hauss.geneland.irun] <- mean(scan(hauss.geneland.path.lpd)[-1:hauss.geneland.burnin])
}
order(hauss.geneland.lpd, decreasing=TRUE) # Sorts runs according to decreasing posterior probability
# Run 5 is the best here
hauss.geneland.lpd # Here the runs are unsorted
# Run 5 wins
```

- We will use figures and $F_{ST}$ outputs only from the best run
- It is useful to keep all runs especially for comparison if there are different solutions with similar posterior probability

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

266/421
MCMC chain, number of clusters and their map

MCMC did not converge yet — too few generations, the most likely solution is K=4 followed by K=5. Final product is map of distribution of genetic clusters.
Map of posterior probability of belonging into cluster 1
When using Geneland, remember...

- Within `MCMC()`, there must be at least hundreds thousands or millions of generations (`nit`) and appropriate sampling (thousands or higher, `thinning` — not to fill whole disk)
- To analyze geo-referenced data with a non-spatial prior set in `MCMC()` `spatial=FALSE`
- To analyze non-spatial data remove parameter `coordinates` from `MCMC()` function
- To obtain structure-like plots, file `proba.pop.membership.indiv.txt` in Geneland output directory can be used as input file for `distruct`
- To use SNPs, ATCG bases must be recoded as `1, 2, 3, 4`, fixed alleles must be removed
- Geneland can handle only haploids and diploids (no ploidy mixing)
- When unsure, consult `manual`
Mapping overview

- There are plenty of options, only some basic options are shown...
- Some packages offer basic (state) maps
- It is possible to use various on-line maps
- Shapefiles (SHP) used in specialized GIS software can be used as base layer to plot points etc. to that map
- Interactive maps can be created with libraries like leaflet
- It is possible to plot to maps not only point and/or text labels, but also pie charts, bar charts, or e.g. plot phylogeny to the map
- Much more options are available in specialized mapping packages, especially with connections with GIS (e.g. QGIS or GRASS GIS)
Very basic mapping in R

```r
library(sp) # Load libraries
library(rworldmap) # Basic world maps
library(TeachingDemos) # To be able to move text little bit
library(RgoogleMaps) # Google and OpenStreetMaps
library(mapplots) # Plot pie charts
library(adegenet)

# Plot basic map with state boundaries within selected range
plot(x=getMap(resolution="high"), xlim=c(19,24), ylim=c(39,44), asp=1, lwd=1.5)
box() # Add frame around the map

# Plot location points
points(x=hauss.genpop@other$xy[['lon']], y=hauss.genpop@other$xy[['lat']],
pch=15:19, col="red", cex=4)

# Add text descriptions for points. Text is aside and with background
shadowtext(x=hauss.genpop@other$xy[['lon']], y=hauss.genpop@other$xy[['lat']],
labels=as.vector(popNames(hauss.genind)), col="black",
bg="white", theta=seq(pi/4, 2*pi, length.out=8), r=0.15, pos=c(1, 3, 2, 4, 4), offset=0.75, cex=1.5)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
# Insert legend

```r
legend(x="topright", inset=1/50, legend=c("He", "Oh", "Pr", "Ne", "Sk"),
    col="red", border="black", pch=15:19, pt.cex=2, bty="o", bg="lightgrey",
    box.lwd=1.5, cex=1.5, title="Populations")
```

---

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

272/421
# Basic help for Google maps

```r
?GetMap  
?PlotOnStaticMap
```

- For larger/complex maps Google requires **API key**
- See documentation and more options, it can use plenty of map resources (Bing maps, ...)
- Map is first downloaded with `GetMap()` into R object, and then plotted with `PlotOnStaticMap()` or with another function
- Many on-line map services do require (paid) API key...
# Download map
```r
gauss.gmap <- GetMap(center = c(lat = 41, lon = 21), size = c(640, 640),
  destfile = "gmap.png", zoom = 8, maptype = "satellite")
```

# Plot saved map, with extra data
```r
PlotOnStaticMap(MyMap = gauss.gmap, lat = gauss.genpop@other$xy[["lat"]],
  lon = gauss.genpop@other$xy[["lon"]], FUN = points, pch = 19, col = "blue",
  cex = 5)
```

```r
PlotOnStaticMap(MyMap = gauss.gmap, lat = gauss.genpop@other$xy[["lat"]],
  lon = gauss.genpop@other$xy[["lon"]], add = TRUE, FUN = points, pch = 19,
  col = "red", cex = 3)
```

```r
PlotOnStaticMap(MyMap = gauss.gmap, lat = gauss.genpop@other$xy[["lat"]],
  lon = gauss.genpop@other$xy[["lon"]], add = TRUE, FUN = text,
  labels = as.vector(popNames(gauss.genind)), pos = 4, cex = 3, col = "white")
```

# Google maps have their own internal scaling, adding of points by
# standard functions will not work correctly

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Molecular data in R
January 31 to February 3, 2022
274/421
Adding pie charts to map 1

```r
# Prepare matrix with some data (exemplary distribution of haplotypes)
hauss.pie <- cbind(c(20, 30, 15, 40, 10), c(30, 10, 25, 5, 45),
                  c(10, 20, 15, 5))
# Add row names according to populations
rownames(hauss.pie) <- popNames(hauss.genpop)
# Add column names according to data displayed
colnames(hauss.pie) <- c("HapA", "HapB", "HapC")
class(hauss.pie) # Check it is matrix
hauss.pie # See resulting matrix

# Plot basic map with state boundaries within selected range
plot(x=getMap(resolution="high"), xlim=c(20, 23), ylim=c(41, 42), asp=1, lwd=1.5)
box() # Add frame around the map

# Plot the pie charts
for (L in 1:5) {
  add.pie(z=hauss.pie[L,], x=as.vector(hauss.genpop@other$xy[["lon"]])[L],
          y=as.vector(hauss.genpop@other$xy[["lat"]])[L], labels=names(hauss.pie[L,]),
          radius=0.1, col=topo.colors(3))
}
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

275/421
Adding pie charts to map II

```r
add.pie # See more options
# Add population labels
text(
x=hauss.genpop$other$xy[["lon"]],
y=hauss.genpop$other$xy[["lat"]],
labels=as.vector(popNames(hauss.genind)),
col="red", cex=2)
```

• As usually, play with colours and display options...
# Prepare list to store recalculated coordinates
hauss.gmap.coord <- list()

# Calculation of coordinates to form required by Google Maps
for (LC in 1:5) {
  hauss.gmap.coord[[LC]] <- LatLon2XY.centered(MyMap=hauss.gmap, lat=as.vector(hauss.genpop$other$xy[["lat"]])[LC],
    lon=as.vector(hauss.genpop$other$xy[["lon"]])[LC], zoom=8)
}

hauss.gmap.coord # See result

PlotOnStaticMap(MyMap=hauss.gmap)

# Plot pie charts
for (LP in 1:5) {
  add.pie(z=hauss.pie[LP,], x=hauss.gmap.coord[[LP]]$newX,
    y=hauss.gmap.coord[[LP]]$newY, labels=names(hauss.pie[LP,]),
    radius=25, col=topo.colors(n=3, alpha=0.7))
}

# Alternative option to plot pie charts
for (LF in 1:5) {
  plotrix::floating.pie(xpos=hauss.gmap.coord[[LF]]$newX,
    ypos=hauss.gmap.coord[[LF]]$newY, x=hauss.pie[LF,], radius=30,
    col=heat.colors(n=3, alpha=0.5))
}
Pie charts on Google map II

```r
# Add population text labels
PlotOnStaticMap(MyMap = hauss.gmap, lat = hauss.genpop$other$sxy["lat"],
lon = hauss.genpop$other$sxy["lon"], add = TRUE, FUN = text,
labels = as.vector(popNames(hauss.genind)), cex = 2.5, col = "white")
```

![Map with pie charts](image-url)
Datasets from mapproj

```r
# Plot on data sets from mapproj package
library(maps) # Various mapping tools (plotting, ...)
# More detailed maps, but political boundaries often outdated, see
# https://CRAN.R-project.org/package=mapdata
library(mapdata)
library(mapproj)
# Convert latitude/longitude into projected coordinates
# Plot a map, check parameters
# Check among others "projection" and ?mapproject for its details
map(database="world", boundary=TRUE, interior=TRUE, fill=TRUE,
    col="lightgrey", plot=TRUE, xlim=c(16, 27), ylim=c(37, 46))
# If you'd use projection, use mapproject to convert also coordinates!
?mapproject # See for details
points(x=hauss.genpop$other$xy["lon"], y=hauss.genpop$other$xy["lat"],
    pch=15:19, col="red", cex=3)
```
Plotting on SHP files I

- **Shapefile** (SHP) is common format for geographical data
- Get SHP files from [https://soubory.trapa.cz/rcourse/macedonia.zip](https://soubory.trapa.cz/rcourse/macedonia.zip) (from [https://download.geofabrik.de/europe/macedonia.html](https://download.geofabrik.de/europe/macedonia.html)) and unpack them into R working directory
- R working directory has to contain also respective DBF and SHX files (same name, only different extension)

```r
library(maptools)
library(rgdal)
dir() # Verify required files are unpacked in the working directory
# Check correct import by plotting all layers
macedonia_building <- readOGR(dsn="macedonia_buildings.shp")
plot(macedonia_building)
macedonia_landuse <- readOGR(dsn="macedonia_landuse.shp")
plot(macedonia_landuse)
```
macedonia_natural <- readOGR(dsn="macedonia_natural.shp")
plot(macedonia_natural)
macedonia_railways <- readOGR(dsn="macedonia_railways.shp")
plot(macedonia_railways)
macedonia_roads <- readOGR(dsn="macedonia_roads.shp")
plot(macedonia_roads)
macedonia_waterways <- readOGR(dsn="macedonia_waterways.shp")
plot(macedonia_waterways)
# Plot all layers into single image, add more information
plot(macedonia_building)
plot(macedonia_landuse, add=TRUE, col="darkgreen", fill=TRUE)
plot(macedonia_natural, add=TRUE, col="green", fill=TRUE)
plot(macedonia_railways, add=TRUE, col="brown", lty="dotted")
plot(macedonia_roads, add=TRUE, col="orange")
plot(macedonia_waterways, add=TRUE, col="blue", lwd=2)
# Now we can add few more data, next slide
# Add state boundaries

```r
plot(x = getMap(resolution = "high"), xlim = c(19, 24), ylim = c(39, 44), asp = 1, lwd = 5, add = TRUE) # Or e.g.
map(database = "world", boundary = TRUE, interior = TRUE, fill = FALSE, col = "red", add = TRUE, plot = TRUE, xlim = c(16, 27), ylim = c(37, 46), lwd = 5)
```

# Add sampling points

```r
points(x = hauss.genpop@other$xy[["lon"]], y = hauss.genpop@other$xy[["lat"]], pch = 15:19, col = "red", cex = 4)
```

# Add description of sampling points

```r
shadowtext(x = hauss.genpop@other$xy[["lon"]], y = hauss.genpop@other$xy[["lat"]], labels = as.vector(popNames(hauss.genind)), col = "black", bg = "white", theta = seq(pi/4, 2*pi, length.out = 8), r = 0.15, pos = c(1, 3, 2, 4, 4), offset = 0.75, cex = 1.5)
```

# Add legend

```r
legend(x = "topright", inset = 1/50, legend = c("He", "Oh", "Pr", "Ne", "Sk"), col = "red", border = "black", pch = 15:19, pt.cex = 2, bty = "o", bg = "lightgrey", box.lwd = 1.5, cex = 1.5, title = "Populations")
```
Plotting on SHP files IV and maps from mapproj
Practicing work with spatial data

Tasks

1. Try some spatial analysis with adegenet’s alpine *Rupicapra* dataset (see `?adegenet::rupica`), or cats from Nancy (see `?adegenet::nancycats`), or with any other (your) data.

2. Plot some of the above dataset, or some your data to map. Try several mapping options.

3. Which problems can you encounter when doing spatial analysis or plotting maps over large spatial scale (level of continents and above), close to equator or in polar regions?

4. What are possible problems when computing distances among sampling sites?

5. Search the Internet for another options how to plot maps in R which could be suitable for your data.

Note it has coordinates only for populations, not for individuals.
Manipulation, display and analysis of sets of trees
Work with individual trees and sets of trees, finding species trees from multiple gene trees

# Library
library(ape)
Working with phylogenetic trees in R I

- To import one or more tree(s) in **NEWICK** (*.tre, *.nwk,...) use **read.tree()**, for trees in **NEXUS** (*.nex, *.nexus,...) use **read.nexus()**

- To export tree(s) in NEWICK use **write.tree()**, in NEXUS use **write.nexus()**

- Another tree (ape’s class **phylo**) formats can be imported/exported/manipulated with e.g. functions from **treeio** package

- Some functions manipulating trees can work only with single tree (class **phylo**), some can work with multiple trees (class **multiPhylo**)

- **ape** has plenty of **.multiPhylo** functions to handle tree sets (e.g. **root.phylo** vs. **root.multiPhylo**)

- If the function needed does not work with multiple trees, use **lapply()** (see further examples)
Working with phylogenetic trees in R II

• Typical operations with trees in R
  • Calculation of individual phylogenetic trees — NJ, MP, ML, ...
  • Plotting, including various highlights and information on tips, nodes, labels, ...
  • Plotting two or more trees together to compare topologies (trees from various methods/genes, tree of plants vs. polynators, …)
  • Preparing trees for subsequent analysis (e.g. character evolution) — removal of outgroup(s), binding of trees (e.g. crown group from one gene and and stem lineages from another), removal of certain taxa, …
  • Comparison and evaluation of multiple gene trees, identification of trees with outlying topology
  • Construction of species tree or network from set of multiple gene trees
  • And more…

• Plenty of packages available...
• Users of ggplot2 can e.g. use ggtree
Manipulations

Introduction

Rooting and unrooting trees I

plot.phylo(hauss.nj)
print.phylo(hauss.nj)
tiplabels() # Shows tip numbers
# root.phylo accepts either tip number or tip label
# resolve.root=TRUE ensures root will be bifurcating
# (without this parameter it sometimes doesn't work)
# outgroup can be single value or vector of multiple tips
hauss.nj.rooted <- root.phylo(phy=hauss.nj, resolve.root=TRUE,
    outgroup=16) # Or
hauss.nj.rooted <- root.phylo(phy=hauss.nj, resolve.root=TRUE,
    outgroup="H16") # Or
hauss.nj.rooted <- root.phylo(phy=hauss.nj, resolve.root=TRUE,
    outgroup=c("H42", "H43"))
# root.multiPhylo() is an alias for root.phylo() for class multiPhylo
print.phylo(hauss.nj.rooted)
plot.phylo(hauss.nj.rooted)
# Rooting and unrooting trees II & swap clade

```r
# Check if it is rooted - returns TRUE/FALSE
is.rooted(hauss.nj.rooted)
is.rooted(hauss.nj)

# Root the tree interactively - click to selected tip
plot.phylo(hauss.nj)
hauss.nj.rooted <- root.phylo(phy=hauss.nj, interactive = TRUE)
plot.phylo(hauss.nj.rooted)

?unroot.phylo # Unroot the tree

# root(), unroot() and is.rooted() works with single or multiple trees
# (class phylo or multiPhylo)
# Rotate (swap) clade
# plot.phylo plots tree in exact order as it is in the phylo object
plot.phylo(hauss.nj)
nodelabels()

hauss.nj.rotated <- ape::rotate(phy=hauss.nj, node=74)
plot.phylo(hauss.nj.rotated)
nodelabels()
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Ladderize tree, drop fossil (extinct) tips

```r
plot.phylo(hauss.nj)
# Ladderize the tree - step-wise rotation of nodes
hauss.nj.ladderized <- ladderize(hauss.nj)
# Topology is unchanged, nodes are only rotated
plot.phylo(hauss.nj.ladderized)
# Drop "extinct" tips - those who don't reach end the tree
# tolerance is respective to the used metrics
plot.phylo(hauss.nj)
axisPhylo()
hauss.nj.fossil <- drop.fossil(phy=hauss.nj, tol=0.4)
plot.phylo(hauss.nj.fossil)
# See details
?drop.fossil
```
# Extract clade, part of tree

```r
# Plot source tree
plot.phylo(hauss.nj)

# See node labels (numbers) - needed for some tasks
nodelabels()

# Non-interactively extract clade
hauss.nj.extracted <- extract.clade(phy=hauss.nj, node=60)

# See new extracted tree
plot.phylo(hauss.nj.extracted)

# Interactive extraction
plot.phylo(hauss.nj)

# Select clade to extract by clicking on it
hauss.nj.extracted <- extract.clade(phy=hauss.nj, interactive=TRUE)

# See new extracted tree
plot.phylo(hauss.nj.extracted)

# Extract more options
extract.clade

# Does the opposite - keeps only selected tip(s)
keep.tip
```
Bind two trees into one

# Bind donor "y" tree to a given position of the "x" tree
?bind.tree  # See options
plot.phylo(hauss.nj.fossil)
nodelabels()
plot.phylo(hauss.nj.extracted)
nodelabels()
hausts.nj.bind <- bind.tree(x=hausts.nj.fossil, y=hausts.nj.extracted,
    where="root", position=0, interactive=FALSE)
plot.phylo(hauss.nj.bind)

# Bind two trees interactively
# Plot tree receiving the new one
plot.phylo(hauss.nj.fossil)
# Select where to bind new tree to
hausts.nj.bind <- bind.tree(x=hausts.nj.fossil, y=hausts.nj.extracted,
    interactive=TRUE)
plot.phylo(hauss.nj.bind)
Work with multiple trees

```r
# Read tree(s) in NEWICK format - single or multiple tree(s)
oxalis.trees <- read.tree("https://soubory.trapa.cz/rcourse/oxalis.nwk")

# See all trees
lapply(X=oxalis.trees, FUN=print.phylo)
plot.multiPhylo(x=oxalis.trees)

# See all trees
summary(oxalis.trees)
length(oxalis.trees)
names(oxalis.trees)

# Export trees in NEWICK format
write.tree(phy=oxalis.trees, file="trees.nwk")

# Export trees in NEXUS format
write.nexus(oxalis.trees, file="trees.nexus")

# Root all trees
oxalis.trees.rooted <- root.multiPhylo(phy=oxalis.trees,
                                        outgroup="O._fibrosa_S159", resolve.root=TRUE)

# See all trees
lapply(X=oxalis.trees.rooted, FUN=print.phylo)
plot.multiPhylo(x=oxalis.trees.rooted) # See all trees
```
Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022  294/421

Drop a tip

```r
plot.phylo(hauss.nj)
hauss.nj[["tip.label"]]
tiplabels()

hauss.nj.drop <- drop.tip(phy=hauss.nj, tip=47)  # Or
hauss.nj.drop <- drop.tip(phy=hauss.nj, tip="H31")  # Or
hauss.nj.drop <- drop.tip(phy=hauss.nj, tip=c("H18", "H29", "H31"))
plot.phylo(hauss.nj.drop)

# Drop a tip from multiPhylo
plot.multiPhylo(x=oxalis.trees)

# See tip labels
oxalis.trees[[1]][["tip.label"]]

oxalis.trees.drop <- lapply(X=oxalis.trees, FUN=drop.tip,  
tip="O._callosa_S15")
class(oxalis.trees.drop) <- "multiPhylo"
plot.multiPhylo(x=oxalis.trees.drop)
lapply(X=oxalis.trees.drop, FUN=print.phylo)
```
Branch lengths and ultrametricity of the tree

```r
# Check if the tree is ultrametric - is variance of distances of all
tips to node 0? It is required for some analysis
is.ultrametric(oxalis.trees)
# Fitting a chronogram to a phylogenetic tree whose branch lengths are
# in number of substitution per sites - force tree to be ultrametric
?chronos
# Compute branch lengths for trees without branch lengths
?compute.brlen
# Computes the branch lengths of a tree giving its branching times
# (aka node ages or heights)
?compute.brtime
```

- Class `multiPhylo` is just a list of `phylo` objects to store multiple trees — you can perform most of analysis on it as on `phylo`, commonly using `lapply` (afterward use `class(x) <- "multiPhylo"`
Maximum parsimony — theory

- **Maximum parsimony** finds optimal topology of the phylogenetic tree by minimizing of the total number of character-state changes.

- It minimizes homoplasy (convergent evolution, parallel evolution, evolutionary reversals).

- Very simple criterion, easy to score the tree, but not to find it — exhaustive search to explore all possible trees is realistic until ~9 taxa, branch-and-bound swapping (guaranteeing finding the best tree) until ~20 taxa, for more taxa heuristic search is needed — it doesn’t always guarantee to find the most probable (parsimonious) tree.

- To speed up calculations, initial tree (usually NJ — slide 191) is used to start the search.

- With rising performance of computers, it uses to be replaced maximum likelihood or Bayesian methods.

- Still underlying plenty of analysis, not only reconstruction of phylogeny.
Maximum parsimony — code and result

```r
library(phangorn)
# Conversion to phyDat for phangorn
gunnera.phydat <- as.phyDat(gunnera.mafft.ng)
#
# Prepare starting tree
gunnera.tre.ini <- nj(dist.dna(x=gunnera.mafft.ng, model="raw"))

?parsimony
# Parsimony details
parsimony(tree=gunnera.tre.ini, data=gunnera.phydat)
# Optimisation - returns MP tree
gunnera.tre.pars <- optim.parsimony(tree=gunnera.tre.ini, data=gunnera.phydat)
plot.phylo(x=gunnera.tre.pars, type="cladogram", edge.width=2)
```
Topographical distances among trees I — implementations I

• Comparing plenty of individual gene trees, finding different topologies, construction of consensual species tree topology

• Robinsons-Foulds distance in `phytools::multiRF`
  • The index adds 1 for each difference between pair of trees
  • Well defined only for fully bifurcating trees — if not fulfilled, some results might be misleading
  • Allow comparison of trees created by different methods
  • If the difference is very close to root, RF value can be large, even there are not much differences in the tree at all — `dist.multiPhylo` from package `distory` can be an alternative, although interpretation of that geodesic distance is sometimes not so straightforward as simple logic of RF

• Methods implemented in `ape::dist.topo` allow comparison of trees with polytomies (`method="PH85"`) or use of squared lengths of internal branches (`method="score"`)
Final matrices are commonly not Euclidean — may be problematic for usage in methods like PCA.

- Test it with `ade4::is.euclid`, can be scaled (forced to become Euclidean) by functions like `quasieuclid` or `cailliez` in `ade4` — carefully, it can damage meaning of the data.
- We get matrix of pairwise differences among trees (from multiple genes), we need display and analyze it.

Set of tools for identifying discordant phylogenetic trees are e.g. in package `kdetrees`.

Filtered trees (with removed outlying topologies) are input for further species tree reconstruction method.

Other approach are phylogenetic networks searching consensus among phylogenetic trees.

In any case, there are plenty of options how to display the differences among the trees.
Topographical distances among trees II
We have plenty of trees. How much are their topologies different?

```r
library(gplots)
library(corrplot)
library(phytools)

# Compute matrix of topological distances among phylogenetic trees
?dist.topo # See details of available computing methods
oxalis.trees.d <- dist.topo(x=oxalis.trees, method="score")

# Basic information about the distance matrix
dim(as.matrix(oxalis.trees.d))
head.matrix(as.matrix(oxalis.trees.d))
```

- There are more options how to display the differences and identify (and possibly exclude) outlying trees — heatmap, PCoA, hierarchical clustering (e.g. package `dbscan`), ...
- Sources of incongruencies among trees: low-quality DNA/laboratory mistake, problem with alignment/gene tree reconstruction, gene duplication and paralogy (e.g. in polyploids), ILS, HGT, ... — such problems must be inspected

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Topographical distances among trees III

Post process the matrix and plot it

- There are several methods for calculating distance matrices among the trees — some take branch lengths into account, some only topology.

- There are plenty of heatmap functions, like `heatmap`, `heatmap2`. (from `gplots`), `heatmap.plus` (archived), and more...

```r
# Create heat maps using heatmap.2 function from gplots package
heatmap.2(as.matrix(oxalis.trees.d), Rowv=FALSE, Colv="Rowv",
          dendrogram="none", symm=TRUE, scale="none", na.rm=TRUE, revC=FALSE,
          col=rainbow(15), cellnote=round(as.matrix(oxalis.trees.d), digits=2),
          notecex=1, notecol="white", trace="row", linecol="black",
          labRow=names(oxalis.trees), labCol=names(oxalis.trees), key=TRUE,
          keysize=2, density.info="density", symkey=FALSE, main="Correlation matrix of topographical distances",
          xlab="Trees", ylab="Trees")
```

Vojtěch Zeisek (https://trapa.cz/)
Topographical distances among trees IV

Calculate Robinsons-Foulds distance matrix among trees and plot it

- **phytools::multiRF** can handle **multiPhylo** objects

```r
# Robinsons-Foulds distance
oxalis.trees.d.rf <- multiRF(oxalis.trees)

# Add names of columns and rows
colnames(oxalis.trees.d.rf) <- names(oxalis.trees)
rownames(oxalis.trees.d.rf) <- names(oxalis.trees)

# Create heatmap using corrplot function from corrplot package
corrplot(corr = oxalis.trees.d.rf, method = "circle", type = "upper",
          col = rainbow(15), title = "Correlation matrix of topographical distances",
          is.corr = FALSE, diag = FALSE, outline = TRUE,
          order = "alphabet", tl.pos = "lt", tl.col = "black")

corrplot(corr = oxalis.trees.d.rf, method = "number", type = "lower",
add = TRUE, col = rainbow(15), title = "Correlation matrix of topographical distances",
          is.corr = FALSE, diag = FALSE, outline = FALSE,
          order = "alphabet", tl.pos = "ld", cl.pos = "n")
```

Vojtěch Zeisek (https://trapa.cz/)

Topographical distances among trees V — the matrices
PCoA from distance matrices of topographical differences among trees — the code

PC plots help to identify outliers — trees with noticeably different topology

```r
# Test if the distance matrix is Euclidean or not
library(ade4)
is.euclid(distmat=oxalis.trees.d, plot=TRUE)[1] TRUE # If FALSE, we can use e.g. quasieuclid() to make it Euclidean
# Calculate the PCoA
oxalis.trees.pcoa <- dudi.pco(d=oxalis.trees.d, scannf=TRUE, full=TRUE)
# Plot PCoA and add kernel densities
s.label(dfxy=oxalis.trees.pcoa$li)
s.kde2d(dfxy=oxalis.trees.pcoa$li, cpoint=0, add.plot=TRUE)
# Add histogram of eigenvalues
add.scatter.eig(oxalis.trees.pcoa[["eig"]], 3,1,2, posi="topleft")
# Add title to the plot
title("PCoA of matrix of pairwise trees distances")
scatter(x=oxalis.trees.pcoa, posieig="topleft") # Alternative plotting PCA
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022 304/421
PCoA from distance matrices of topographical differences among trees — the plot

- Of course, think what could cause observed difference...
  - Problem in lab?
  - Paralog? ILS? HGT? ...
  - Try to BLAST the gene

```r
# See current state
oxalis.trees

# Remove outlying trees
oxalis.trees[c("T471", "T639", "T654")]<-NULL

# See after removal
oxalis.trees
```
**kdetrees — identification of outlying trees**

- Distance-based method of identification of trees with significantly different topology
- Function `kdetrees` has plenty of options...
- Parameter `k` sets threshold for trees to be removed — it requires repeated running with different `k` (and plotting the figures) to decide which trees to remove and which to keep

```r
# Load library
library(kdetrees)

# kdetrees # See options
# Run main function - play with parameter k
oxalis.kde <- kdetrees(trees=oxalis.trees, k=0.4, 
  distance="dissimilarity", topo.only=FALSE, greedy=TRUE)

# See results
plot(oxalis.kde)
hist(oxalis.kde)
```

Vojtěch Zeisek (https://trapa.cz/)
kdetrees — identification of outlying trees II
Manipulations

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

308/421
Consensus tree

- Very basic method simply collapsing all nodes with lower than required (typically 50–75%) identity
- There is no underlying model, only comparison of topologies

```r
# Consensus tree (50% rule)
oxalis.tree.con <- consensus
  (oxalis.trees.rooted, p=0.5,
    check.labels=TRUE)
print.phylo(oxalis.tree.con)
# Plot the tree
plot.phylo(oxalis.tree.con,
  edge.width=2, label.offset=0.3)
axisPhylo(side=1)
# What a nice tree... :-P
```
Species tree — all trees must be ultrametric

```r
# Chronos scale trees
oxalis.trees.ultra <- lapply
  (X=oxalis.trees.rooted,
   FUN=chronos, model="correlated")
class(oxalis.trees.ultra) <- "multiPhylo"

# Mean distances
oxalis.tree.sp.mean <- speciesTree
  (x=oxalis.trees.ultra, FUN=mean)

# Plot the tree
plot.phylo(oxalis.tree.sp.mean,
   edge.width=2, label.offset=0.01)
edgelabels(text=round(oxalis.
   tree.sp.mean[["edge.length"]],
   digits=2), frame="none",
   col="red", bg="none")
axisPhylo(side=1)
```
library(phytools)

?mrp.supertree # See details

oxalis.tree.sp <- mrp.supertree
tree = oxalis.trees.rooted,
method = "optim.parsimony",
rooted = TRUE

print.phylo(oxalis.tree.sp)

plot.phylo(oxalis.tree.sp,
edge.width = 2,
label.offset = 0.01)

axisPhylo(side = 1)

# Similar function

?phangorn::superTree

# Coalescence model to handle

# multiple individuals per species

?phangorn::coalSpeciesTree
library(phangorn)

oxalis.tree.net <- consensusNet(oxalis.trees.rooted, prob = 0.25)

plot(x = oxalis.tree.net, planar = FALSE, type = "2D", use.edge.length = TRUE,
     show.tip.label = TRUE, show.edge.label = TRUE, show.node.label = TRUE,
     show.nodes = TRUE, edge.color = "black", tip.color = "blue") # 2D

plot(x = oxalis.tree.net, planar = FALSE, type = "3D", use.edge.length = TRUE,
     show.tip.label = TRUE, show.edge.label = TRUE, show.node.label = TRUE,
     show.nodes = TRUE, edge.color = "black", tip.color = "blue") # 3D
# Prepare list of trees to show
```r
hauss.nj.trees <- list(hauss.nj, hauss.nj.bruvo, hauss.nj.rooted)
hauss.nj.trees <- lapply(X=hauss.nj.trees, FUN=compute.brlen)
hauss.nj.trees <- lapply(X=hauss.nj.trees, FUN=chronos)
class(hauss.nj.trees) <- "multiPhylo"
```

# The trees should be (otherwise plotting works, but may be ugly)...  
is.rooted.multiPhylo(hauss.nj.trees)  # rooted,  
is.ultrametric.multiPhylo(hauss.nj.trees)  # ultrametric and  
is.binary.multiPhylo(hauss.nj.trees)  # binary bifurcating.

# Plotting has various options, play with it
```r
phangorn::densiTree(x=hauss.nj.trees, direction="downwards", scaleFactor=TRUE, col=rainbow(3), width=5, cex=1.5)  # See next slide
densiTree(x=hauss.nj.trees, direction="upwards", scaleFactor=TRUE, width=5)
densiTree(x=hauss.nj.trees, scaleFactor=TRUE, width=5, cex=1.5)
```

# Compare this option with similar on following slide
```r
?phangorn::densiTree
?phytools::densityTree
```
Density tree II
Density tree III

```r
phytools::densityTree(trees=oxalis.trees.ultra, fix.depth=TRUE,
                      use.gradient=TRUE, alpha=0.5, lwd=4) # Probably to much noise... :-?
phytools::densityTree(trees=oxalis.trees.ultra[1:3], fix.depth=TRUE,
                      use.gradient=TRUE, alpha=0.5, lwd=4) # Nice selection
phytools::densityTree(trees=oxalis.trees.ultra[c(2,4,6,7)],
                      fix.depth=TRUE, use.gradient=TRUE, alpha=0.5, lwd=4) # Nice selection
```
Kronoviz — see all trees on same scale

```r
kronoviz(x = oxalis.trees.rooted,
          layout = length(oxalis.trees.rooted),
          horiz = TRUE)

# Close graphical device to
# cancel division of plotting
# device
dev.off()
```

- The plot can be very long and it can be hard to see details
- But one can get impression if all trees are more or less in same scale (have comparable length) or not

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
# Compare two trees

```r
# Compare topology of the species trees - basically outputs TRUE/FALSE
all.equal.phylo(oxalis.tree.sp, oxalis.tree.sp.mean, use.edge.length=FALSE)
?all.equal.phylo # Use to see comparison possibilities

# Plot two trees with connecting lines
# We need 2 column matrix with tip labels

tips.labels <- matrix(data=c(sort(oxalis.tree.sp[['tip.label']])),
                      sort(oxalis.tree.sp.mean[['tip.label']])), nrow=length(oxalis.tree.sp[['tip.label']]), ncol=2)

# Draw a tree - play with graphical parameters and use rotate=TRUE
# to be able to adjust fit manually

cophyloplot(x=ladderize(oxalis.tree.sp), y=ladderize(oxalis.tree.sp.mean),
            assoc=tips.labels, use.edge.length=FALSE, space=60, length.line=1, gap=2,
            type="phylogram", rotate=TRUE, col="red", lwd=1.5, lty=2)
title("Comparing the trees
Parsimony super tree
Species tree")

legend("topleft", legend="Red lines
connect tips", text.col="red",
        cex=0.75, bty="n", x.intersp=-2, y.intersp=-2)
```

Vojtěch Zeisek (https://trapa.cz/)
Cophyloplot comparing two trees

- **cophyloplot()** has not any optimization to plot the lines
- Automatic plot is usually not perfect — there use to be unneeded crossing lines — `rotate=TRUE` is recommended to can fix this manually by clicking to the nodes
- **cophyloplot()** has similar parameters like `plot.phylo()` — play with it and/or adjust in graphical editor
- Other options are in package `dendextend`

- **ladderize()** pre-sorts tips in the tree — it can help to reduce crossings

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022

318/421
Alternative implementation — `phytools::cophylo`

```r
?cophylo # See options

# Prepare the object for plotting
oxalis.cophylo <- cophylo(tr1=oxalis.tree.sp, tr2=oxalis.tree.sp.mean,
  assoc=(cbind(sort(oxalis.tree.sp$tip.label),
    sort(oxalis.tree.sp$tip.label))), rotate=TRUE)

plot.cophylo(x=oxalis.cophylo, lwd=2, link.type="curved") # Plot it

title("Comparison of species tree (left) and parsimony supertree (right)")
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022
Change orientation of plots

- `plot.phylo()` has plenty of possibilities to influence – check `?plot.phylo`, `?par`, `?points`, ...

```r
plot.phylo  # check it for various possibilities what to influence
par(mfrow=c(1, 2))  # Plot two plots in one row
plot.phylo(x=hauss.nj, type="cladogram", use.edge.length=FALSE,
            direction="rightwards")
plot.phylo(x=hauss.nj, type="cladogram", use.edge.length=FALSE,
            direction="leftwards")
dev.off()  # Close graphical device to cancel par() settings
```
```r
# Load tree in text format
trape <- read.tree(text = "((Homo, Pan), Gorilla);")

# Plot the tree
plot.phylo(x = trape,
            show.tip.label = FALSE)

# Add colored tip labels
tiplabels(trape[["tip.label"]],
          bg = c("white", "black", "white"),
          col = c("black", "white", "black"), cex = 2)

# Add colored node labels
nodelabels(text = c("6.4 Ma", "5.4 Ma"),
           frame = "circle",
           bg = "yellow")

add.scale.bar() # Add scale bar

# Note vectors for tip/node labels
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Introduction

Manipulations

R

Data

Alignment

Basic analysis

SNP

DAPC

Spatial analysis

Trees

Evolution

Tasks

The end

Trees tasks

Tasks

1. Import/export some tree(s), preferably your data you are working with. Try various plotting options.

2. Try various displays of some tree(s) from phytools dataset (see

   \texttt{data(package="phytools")}).

3. Use some method to analyze Apicomplexa trees (see \texttt{?kdetrees::apicomplesa}) or some your data. Find discordant trees and try to construct species tree from the set of gene trees.

4. Browse \url{http://blog.phytools.org/} for a while. Do you find there some interesting/useful display? Try something with your or test data.
# Reconstruction of evolution of traits

## Evolution
- PIC
- Autocorrelation
- pPCA
- Decomposition
- PGLS
- GEE
- Phylosignal
- Ancestral state
- Phenogram
- Tasks

**Adapted from a presentation by Vojtěch Zeisek (https://trapa.cz/)**
Overview of methods and questions of reconstruction of evolution of traits I

- Testing if there is correlation between evolution of two or more characters (if they evolve together)
- Testing if there is correlation between one character and phylogenetic history (if trait changes follow evolution)
- Reconstruction of ancestral states of character
- For some methods, taxonomic level can be taken into account (if there is significant evolutionary signal on the trait evolution on e.g. level of genus or family)
- Generally available for continuous as well as discrete characters (not all methods)
- Some methods can handle more observations per accession
- There are various methods how to display everything
- Methods and models are highly debated in the literature
Overview of methods and questions of reconstruction of evolution of traits II

• Different experts commonly disagree what is the best method...
• General methods are not usable everywhere (e.g. evolution of genome size must take into account polyploidization — chromEvol)
• Usage is better to be consulted with some relevant expert
• This is very difficult chapter by meaning of how to find the best method to analyze particular data...
• Always read manual and original papers explaining the methods
Phylogenetic independent contrast

- When analyzing comparative data takes phylogeny into account
- If we assume that a continuous trait evolves randomly in any direction (i.e. following Brownian motion model), then the “contrast” between two species is expected to have a normal distribution with mean zero, and variance proportional to the time since divergence

```r
data(shorebird, package="caper") # Load training data
caper::shorebird # See it
head(shorebird.data) # See the data part
shorebird.tree # See the phylogeny
plot.phylo(shorebird.tree)
# The tree must be fully bifurcating for most of methods
shorebird.tree <- multi2di(phy=shorebird.tree)
plot.phylo(shorebird.tree) # See result
```
PIC and its plotting

```r
shorebird.pic.fmass <- pic(x=shorebird.data[["F.Mass"]], phy=
  shorebird.tree, scaled=TRUE, var.contrasts=FALSE, rescaled.tree=FALSE)
shorebird.pic.eggmass <- pic(x=shorebird.data[["Egg.Mass"]], phy=
  shorebird.tree, scaled=TRUE, var.contrasts=FALSE, rescaled.tree=FALSE)
# Plot a tree with PIC values
plot.phylo(x=shorebird.tree, edge.width=2)
nodelabels(round(shorebird.pic.fmass, digits=3), adj=c(0, -0.5),
  frame="none", col="red")
nodelabels(round(shorebird.pic.eggmass, digits=3), adj=c(0, 1),
  frame="none", col="red")
add.scale.bar()
# Plot PIC
plot(x=shorebird.pic.fmass, y=shorebird.pic.eggmass, pch=16, cex=1.5)
abline(a=0, b=1, lty=2) # x=y line
# Correlation of PIC of body mass and longevity
cor(x=shorebird.pic.fmass, y=shorebird.pic.eggmass, method = "pearson")
[1] 0.879019
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022
Plot of PIC (on the tree) and test the correlation

```r
# Correlation test
cor.test(x = shorebird.pic.fmass, y = shorebird.pic.eggmass, 
  alternative = "greater", method = "pearson")
```
# Linear model of both PICs

```
lm(formula = shorebird.pic.fmass ~ shorebird.pic.eggmass)
```

Coefficients:

```
(Intercept)    shorebird.pic.eggmass
2.636           9.110
```

# Because PICs have expected mean zero - such linear regressions should
# be done through the origin (i.e. the intercept is set to zero)

```
lm(formula = shorebird.pic.fmass ~ shorebird.pic.eggmass - 1)
```

Coefficients:

```
(Intercept)    shorebird.pic.eggmass
2.636           9.110
```

# Permutation procedure to test PIC

```
lm(origin(formula = shorebird.pic.fmass ~ shorebird.pic.eggmass, nperm = 1000)
```

Coefficients and parametric test results

```
      Coefficient Std_error    t-value  Pr(>|t|)
shorebird.pic.eggmass     9.14518   0.60274  15.173 < 2.2e-16 ***
```
Intraspecific variation

- `pic.ortho()` requires list of measurements (numeric vectors) for all taxa — their lengths can differ.
- We must for each `list` item provide numeric vector (construct by something like `values <- list(taxon1, taxon2, taxon3, ...)`), otherwise usage is same as with `pic()` — plotting, tests, etc.
- Transforming data into required form can require considerable work...

```
# See help
?pic.ortho
```
Phylogenetic autocorrelation

- Autocorrelation coefficient to quantify whether the distribution of a trait among a set of species is affected or not by their phylogenetic relationships
- In the absence of phylogenetic autocorrelation, the mean expected value of $I$ and its variance are known — it is thus possible to test the null hypothesis of the absence of dependence among observations

```r
# Let's choose weights as $w_{ij} = 1/d_{ij}$, where the $d$'s is the distances
# measured on the tree - cophenetic() calculates cophenetic distances
# can be just cophenetic(shorebird.tree) or some other transformation
shorebird.weights <- 1/cophenetic(shorebird.tree)
# See it
class(shorebird.weights)
head(shorebird.weights)
# Set diagonal to 0
diag(shorebird.weights) <- 0
```
Testing of Moran’s I

# Calculate Moran's I
# Significant, but super small, positive phylogenetic correlation
Moran.I(x=shorebird.data[["M.Mass"]], weight=shorebird.weights,
  alternative="greater")
Moran.I(x=shorebird.data[["F.Mass"]], weight=shorebird.weights,
  alternative="greater")

# Test of Moran's with randomisation procedure
library(ade4)
?gearymoran
# For all characters significant, but very small
gearymoran(bilis=shorebird.weights, X=shorebird.data[,2:5], nrepet=1000)
# Test of Abouheif designed to detect phylogenetic autocorrelation
# in a quantitative trait - in fact Moran's I test using a particular
# phylogenetic proximity between tips
library(adephylo)
abouheif.moran(x=shorebird.data[,2:5], W=shorebird.weights,
  method="oriAbouheif", nrepet=1000, alter="greater")
Correlogram to visualize results of phylogenetic autocorrelation analysis

```r
# Data
data(carnivora) # Loads training data set
head(carnivora) # Look at the data

# Calculate the correlogram
?correlogram.formula
carnivora.correlogram <- correlogram.formula
  (formula = SW ~ Order/SuperFamily/Family/Genus, data = carnivora)
carnivora.correlogram # See results

# Calculate the correlogram - test for both body masses
carnivora.correlogram2 <- correlogram.formula
  (formula = SW + FW ~ Order/SuperFamily/Family/Genus, data = carnivora)
carnivora.correlogram2 # See results

# Plot it
plot(x = carnivora.correlogram, legend = TRUE, test.level = 0.05, col = c("white", "black")) # Plot it

# Plot it - test for both body masses - two or one graph(s)
plot(x = carnivora.correlogram2, lattice = TRUE, legend = TRUE, test.level = 0.05)
plot(x = carnivora.correlogram2, lattice = FALSE, legend = TRUE, test.level = 0.05)
```

Vojtěch Zeisek (https://trapa.cz/)
Correlograms of SW and SW+FW (in one or two graphs) depending on taxonomic level with marked significance
Phylogenetic principal component analysis

PCA corrected for phylogeny

- It requires as input phylogenetic tree and respective comparative data
- Phylogenetic component is removed from the data, then classical PCA is calculated
- Very useful for comparison of plenty of characters
- Together with nodes (taxa), PCA scores for PC axes are plotted — not the taxa — it shows trends of character evolution on the tree, not positions of taxa in PC space
- Other graphs show global vs. local structure, eigenvalues decomposition and positions of characters in virtual space (if they correlate or not)
- From package adephylo by Jombart et al. 2010
- It doesn’t contain any test, it is more method of data exploration or dealing with big data sets, it is not for verifying hypothesis
Phylogenetic principal component analysis — the code

```r
# Library needed to create phylo4d object required by ppca
library(adephylo)

# Calculate pPCA
shorebird.ppca <- ppca(x=phylo4d(x=shorebird.tree, shorebird.data[,2:5]),
  method="patristic", center=TRUE, scale=TRUE, scannf=TRUE, nfposi=1,
  nfnega=0)

print(shorebird.ppca) # Print results # See summary information
summary(shorebird.ppca)

# See PCA scores for variables on phylogenetic tree
scatter(shorebird.ppca)

# See decomposition of pPCA eigenvalues
screeplot(shorebird.ppca)

# Plot pPCA results - global vs. local structure, decomposition of pPCA
# eigenvalues, PCA plot of variables and PCA scores for variables on
# phylogenetic tree
plot(shorebird.ppca)
```

Plot pPCA results - global vs. local structure, decomposition of pPCA eigenvalues, PCA plot of variables and PCA scores for variables on phylogenetic tree.
Decomposition of topographical distances

table.phylo4d(x=phylo4d(x=shorebird.tree, tip.data=treePart(x=shorebird.tree, result="orthobasis")), treetype="cladogram")
Prepare toy data set (the variable)

```r
# Generate some random variable
library(geiger)
shorebird.eco <- sim.char(phy=shorebird.tree, par=0.1, nsim=1,
    model="BM")[,1]
?sim.char # See it for another possibilities to simulate data

# Names for the values
names(shorebird.eco) <- shorebird.tree[["tip.label"]]
shorebird.eco # See it
```

- `sim.char()` creates an array (we keep only numeric vector of 1\textsuperscript{st} simulation — [,,1]) of simulated characters, with `model="BM"` under \textit{Brownian motion}.

- Many methods compare \textbf{names} of character values with `tip.label` slot of the tree to pair character values with correct taxa:
  - Otherwise values must be ordered in same way as in `tip.label` slot
  - Always check manual for respective function and all data!
Orthonormal decomposition - phylogenetic eigenvector regression

```r
anova(lm(shorebird.eco ~ as.matrix(orthobasis.phylo(x=shorebird.tree, method="patristic")[,1:2])))
```

Analysis of Variance Table

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.matrix...</td>
<td>2</td>
<td>226.61</td>
<td>113.306</td>
<td>54.045</td>
<td>8.918e-15 ***</td>
</tr>
<tr>
<td>Residuals</td>
<td>68</td>
<td>142.56</td>
<td>2.097</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```r
anova(lm(shorebird.data[["M.Mass"]]) ~ as.matrix(orthobasis.phylo(x=shorebird.tree, method="patristic")[,1:2])))
```

- Significant result — significant phylogenetic inertia (phylogenetic effect) — the tendency for traits to resist evolutionary change despite environmental perturbations
- `orthobasis.phylo()` return matrix, which is linear transformation of cophenetic distances — columns 1 and 2 can be used to calculate phylogenetic variance — it can be used to calculate linear regression
Orthonormal decomposition of variance of a quantitative variable on an orthonormal basis

```r
orthogram(x=shorebird.eco, tre=shorebird.tree, nrepet=1000,
alter="two-sided")
?orthogram # See another calculation possibilities
orthogram(x=shorebird.data[["M.Mass"]], tre=shorebird.tree, nrepet=1000,
alter="two-sided")
```

- Analyses one quantitative trait
- Do not confuse with `ade4::orthogram` — similar, but require data in little bit different form, marked as deprecated and replaced by the `adephylo` version
- It returns results of 5 non-parametric tests associated to the variance decomposition
- Procedure decomposes data matrix to separate phylogeny and phenotype to see if there is significant signal
Observed value is out of permutations — significant inertia of the trait to phylogeny
Phylogenetic Generalized Least Squares

- Model-based testing if there is significant correlation between two traits (after removing the phylogenetic component)
- `nlme::gls` fits a linear model using **generalized least squares**
- Functions `corBlomberg`, `corBrownian`, `corMartins` and `corPagel` from `ape` package create correlation matrix of evolution of continuous character according to the given tree

```r
library(nlme)
summary(gls(model = F.Mass ~ Egg.Mass, data = shorebird.data,
           correlation = corBrownian(value = 1, phy = shorebird.tree)))
```

Generalized least squares fit by REML

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Std. Error</th>
<th>t-value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-2.384233</td>
<td>73.54508</td>
<td>-0.032419</td>
<td>0.9742</td>
</tr>
<tr>
<td>Egg.Mass</td>
<td>9.145185</td>
<td>0.60274</td>
<td>15.172606</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Implementation in caper package

```r
library(caper) # Load needed library
data(shorebird) # Load training data, see ?shorebird.data
# Calculate the model
shorebird.pglsls <- pglsl(formula = shorebird.data[["F.Mass"]]) ~
    shorebird.data[["Egg.Mass"]], data = comparative.data(phy =
    shorebird.tree, data = as.data.frame(cbind(shorebird.data[["F.Mass"]],
    shorebird.data[["Egg.Mass"]], shorebird.data[["Species"]])),
    names.col = V3, vcv = TRUE))
# See the result
summary(shorebird.pglsls)
# See the plot of observer and fitted values
plot(shorebird.pglsls)
abline(a = 0, b = 1, col = "red")
# ANOVA view of the model
anova(shorebird.pglsls)
# Akaike's information criterion (smaller = better)
AIC(shorebird.pglsls)
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Results of PGLS

- `p guts()` uses maximum likelihood to test for phylogenetic signal
- The signal is clearly presented
- Usually, tuning the model (possible data transformations and/or changing model parameters) is necessary to find the best model — AIC helps (lower is better)
- See `caper manual` for details
Generalized Estimating Equations

- Extension of GLM for correlated data, usage is similar
- It is possible to use phylogeny or correlation matrix (typically based on phylogeny)

```r
# Calculate the model
compar.gee(formula=shorebird.data[["F.Mass"]]
     ~ shorebird.data[["Egg.Mass"]], phy=shorebird.tree)

# or with correlation matrix
compar.gee(formula=shorebird.data[["F.Mass"]]
     ~ shorebird.data[["Egg.Mass"]], corStruct=corMartins(value=1,
     phy=shorebird.tree, fixed=TRUE))

# for corStruct there are similar functions corBlomberg, corMartins,
# corPagel, corBrownian - see manuals for differences
```
Call: `compar.gee(formula = shorebird.data[['F.Mass']] ~ shorebird.data[['Egg.Mass']], phy = shorebird.tree)

Number of observations: 71

Model:
- Link: identity
- Variance to Mean Relation: gaussian

QIC: 449593.7

Summary of Residuals:
- Min: -121.577831
- 1Q: -48.195170
- Median: -32.598282
- 3Q: -2.168055
- Max: 295.586322

Coefficients:
- Estimate: -2.384233
- S.E.: 41.120255
- t: -0.057981
- Pr(T > |t|): 9.545638e-01

- S.E.: 0.337003
- t: 27.136762
- Pr(T > |t|): 1.003642e-13

Estimated Scale Parameter: 6515.203

"Phylogenetic" df (dfP): 16.3298
Phylogenetic signal

- Direct consequence of the evolution of trait depends on evolution — if trait variation is driven by environment, phylogenetic signal is 0
- Blomberg’s K statistic of phylogenetic signal

```r
library(picante)
# It requires named vector of trait values
shorebird.mmass <- shorebird.data[["M.Mass"]]
names(shorebird.mmass) <- rownames(shorebird.data)
# Bloomberg's K statistics
Kcalc(x=shorebird.mmass, phy=shorebird.tree, checkdata=TRUE)
# Test with permutations
phylosignal(x=shorebird.mmass, phy=shorebird.tree, reps=1000, checkdata=T)
```

- Blomberg’s values of 1 correspond to a Brownian motion process, which implies some degree of phylogenetic signal or conservatism
- K values closer to zero correspond to a random or convergent pattern of evolution, while K values greater than 1 indicate strong phylogenetic signal and conservatism of traits
Analyze multiple traits in once

```r
# multiPhylosignal requires data frame of numerical values
multiPhylosignal(x = shorebird.data[,2:5], phy = shorebird.tree, reps = 1000)

K PIC.variance.obs PIC.variance.rnd.mean PIC.variance.Pic... 
M.Mass 1.089419 5.139489e+02 3.278559e+03 0.000999001... 
F.Mass 1.092474 6.316701e+02 3.993696e+03 0.000999001... 
Egg.Mass 1.139078 4.160653e+00 2.783029e+01 0.000999001... 
Cl.size 0.673523 8.397776e-03 3.323744e-02 0.000999001...
```

# See help
?phylosignal
?multiPhylosignal
When there are vectors with standard errors of measurements

- Functions for testing of phylogenetic signal do not work with more measurements per taxon
  - Currently, the only possibility is `phylosig()` which is able to work with SE (user must prepare this vector from the data manually; from e.g. `plotrix::std.error`)
  - `phylosig()` can be used as an alternative to `phylosignal()` — the functions are similar in basic usage

```r
library(phytools)
?phylosig  # See for details

# Test for phylogenetic signal (here without SE)
phylosig(tree=shorebird.tree, x=shorebird.mmass, method="K", test=TRUE, nsim=1000)
plot(phylosig(tree=shorebird.tree, x=shorebird.mmass, method="K", test=TRUE, nsim=1000))
phylosig(tree=shorebird.tree, x=shorebird.mmass, method="lambda", test=TRUE)
```
Alternative testing for phylogenetic signal with GLM

- It is possible to use intercept-only (model/formula will be something like `variable ~ 1` instead of `variable1 ~ variable2`) GLM to quantify phylogenetic signal in trait.
- It is tricky to select the best correlation structure — AIC can help with selections (`AIC(pglS(...))`).

```r
# Examples of usage of GLS for testing of phylogenetic signal
summary(gls(model=shorebird.mmass ~ 1, correlation=corBrownian(value=1, phy=shorebird.tree)))
summary(pglS(formula=shorebird.mmass ~ 1, data=comparative.data(phy=shorebird.tree, data=as.data.frame(cbind(shorebird.data[["M.Mass"]], shorebird.data[["Species"]])), names.col="V2, vcv=TRUE")))
```
# Load library
library(ape)

# Loading data
# Ackerly & Donoghue (1998) https://doi.org/10.1086/286208
data(maples, package="adephylo")

# Process the phylogenetic tree
# maples data provide tree as plain text in NEWICK, must be imported
# into the phylo object
maples.tree <- read.tree(text=maples[["tre"]])
maples.tree
plot.phylo(maples.tree)

# For plenty of analysis it must be fully resolved (bifurcating),
# rooting and ultrametricity often helps
is.binary.phylo(maples.tree)
is.rooted.phylo(maples.tree)
is.ultrametric.phylo(maples.tree)
# See the character matrix

```r
head(maples[['tab']])
maples.data <- maples[['tab']][,.1:30]
head(maples.data)
summary(maples.data)
```

# Maples mature height (m)

```r
maples.height <- maples[['tab']]['MatHt']
names(maples.height) <- rownames(maples[['tab']])
maples.height
```

# Maples seed size (mg)

```r
maples.sdsz <- maples[['tab']]['SdSz']
names(maples.sdsz) <- rownames(maples[['tab']])
maples.sdsz
```

# Maples leaf + petiole length (mm)

```r
maples.lfpt <- maples[['tab']]['LfPt']
names(maples.lfpt) <- rownames(maples[['tab']])
maples.lfpt
```
Ancestral state reconstruction

- By default `ape::ace()` performs estimation for continuous characters assuming a Brownian motion model fit by maximum likelihood
- `ace()` can handle continuous as well as discrete data

```r
library(ape)

# See for possible settings
maples.height.ace <- ace(x=maples.height, phy=maples.tree, 
                         type="continuous", method="REML", corStruct=corBrownian(value=1, 
                         phy=maples.tree))

# See result - reconstructions are in ace slot - to be plotted on 
# nodes - 1st column are node numbers
maples.height.ace

# Plot it
plot.phylo(x=maples.tree, lwd=2, cex=0.75)
tiplabels(maples.height, adj=c(1, 0), frame="none", col="blue", cex=0.75)
nodelabels(round(maples.height.ace[["ace"]], digits=1), frame="none", 
            col="red", cex=0.75)
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022 354/421
Ancestral state reconstructions of primates body weights

- ACE returns long numbers — truncate them by e.g.
  \[
  \text{round(x=..., digits=3)} \quad (x \text{ is vector with ACE values})
  \]

```r
# Other implementations are
# available in packages geiger, phangorn, ape, phytools, ...
# Parsimony based method
ape::MPR
# For continuous characters using
# Maximum Likelihood
geiger::fitContinuous
# For continuous characters using
# Markov Chain Monte Carlo
geiger::fitContinuousMCMC
# For discrete characters, various
# models available
geiger::fitDiscrete
# Marginal reconstruction of the
# ancestral character states
phangorn::ancestral.pml
```
Implementation in phytools

```r
# ML estimation of continuous trait
?fastAnc
tree = maples.tree,
x = maples.height,
vars = FALSE, CI = TRUE

maples.height.fa <- fastAnc(tree = maples.tree, x = maples.height, vars = FALSE, CI = TRUE)

maples.height.fa

tiplabels(maples.height, adj = c(1, 0), frame = "none", col = "blue", cex = 0.75)
```

# ACE for Brownian evolution
# with directional trend

```r
?anc.trend
tree = maples.tree,
x = maples.height, maxit = 1000000

maples.height.ac <- anc.trend(tree = maples.tree, x = maples.height, maxit = 1000000)

maples.height.ac
```

# ACE for Brownian evolution
# using likelihood

```r
?anc.ML
tree = maples.tree,
x = maples.height, maxit = 1000000

maples.height.ml <- anc.ML(tree = maples.tree, x = maples.height, maxit = 1000000)
```

Vojtěch Zeisek (https://trapa.cz/)

Molecular data in R

January 31 to February 3, 2022

356/421
Bayesian ancestral character estimation I

1: `anc.Bayes`
2: `maples.height.bayes <- anc.Bayes(tree=maples.tree, x=maples.height,`  
   `ngen=1000000) # Use more MCMC generations`
3: `maples.height.bayes`  
4: `# Get end of ancestral states from Bayesian posterior distribution`  
5: `# (it should converge to certain values)`  
6: `tail(maples.height.bayes[['mcmc']])`  
7: `maples.height.bayes[['mcmc']][10001,3:18]`  
8: `# Get means of ancestral states from Bayesian posterior distribution`  
9: `colMeans(maples.height.bayes[['mcmc']][2001:nrow(`  
10: `maples.height.bayes[['mcmc']],as.character(1:`  
11: `maples.tree$Nnode+length(maples.tree$tip.label))])`  
12: `# Plot the ancestral states from posterior distribution`  
13: `# (it should converge to certain values)`  
14: `plot(maples.height.bayes)`  
15: `# Plot the tree and reconstructed ancestral states`  
16: `plot.phylo(x=maples.tree, edge.width=2, cex=2)`
**Bayesian ancestral character estimation II**

```r
maples.height.bayes # Mean ancestral states from posterior distribution
     18   19   20   21   22   23   24   25 ...
 2.659527 2.662223 2.932293 2.482104 2.456929 2.510414 2.692636 2.919908 ...

tail(maples.height.bayes[["mcmc"]]) # See end of the table
gen   sig2  18   19   20 ...  33    logLik
     ...     ...    ... ...
 10000 999900 0.1860167 2.284352 2.680593 2.962142 ...
 10001 1000000 0.1834573 2.315086 2.048144 2.755104 ...
maples.height.bayes[["mcmc"]][[10001,3:18]] # See end of the table - values
     18   19   20   21   22   23   24 ...
 2.659527 2.662223 2.932293 2.482104 2.456929 2.510414 2.692636 2.919908 ...

# Get means of ancestral states from Bayesian posterior distribution
colMeans(maples.height.bayes[["mcmc"]][[2001:nrow(maples.height.bayes[["mcmc"]]),as.character(1:length(maples.tree$Nnode)+length(maples.tree$tip.label))]])
     18   19   20   21   22   23   24   25 ...
 2.659527 2.662223 2.932293 2.482104 2.456929 2.510414 2.692636 2.919908 ...
```

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))

Molecular data in R

January 31 to February 3, 2022
Bayesian ancestral character estimation III

Likelihood of Bayesian posterior probability and the tree with reconstructions
Continuous map

```r
?contMap
contMap(tree=maples.tree, x=maples.height)
# Change colors with setMap()
maples.contmap <- setMap(
x=contMap(tree=maples.tree, x=maples.height),
colors=c("white", "black"))
plot(maples.contmap)
# See ?par for more settings
```
Display more characters on a tree in a table

```
library(adephylo)
table.phylo4d(x=phylo4d(x=maples.tree, tip.data=maples.data),
  treetype="cladogram", symbol="circles", scale=FALSE, ratio.tree=0.5)
table.phylo4d(x=phylo4d(x=maples.tree, tip.data=maples.data),
  treetype="cladogram", symbol="circles", scale=TRUE, ratio.tree=0.5)
```
**Phenogram**

Vertical axis shows character values

```r
phenogram(tree=maples.tree, x=maples.height, ftype="i", colors="red", main="Maples adult heights")
fancyTree(tree=maples.tree, type="phenogram95", x=maples.height, ftype="i", main="95-percentile of Maples adult heights")
```
Display 2 continuous characters in space and 3D tree connecting them

# 2 characters on 2 axis
phylomorphospace(
  tree = shorebird.tree,
  X = shorebird.data[,2:3],
  label = "horizontal", lwd = 2)

# 3D - 3 characters in a rotating cube
phylomorphospace3d(
  tree = shorebird.tree,
  X = shorebird.data[,2:4],
  label = TRUE)
Combine phenograms and ancestral state reconstructions

# 3 characters on 2 axis and
# ancestral state reconstruction
# for all of them
fancyTree(tree=shorebird.tree,
  type="scattergram",
  X=shorebird.data[,2:4],
  res=500, ftype="i")
# See manuals for more settings
?fancyTree
?phenogram
?phylomorphospace
?phylomorphospace3d
?contMap
?setMap
?par
?plot

Vojtěch Zeisek (https://trapa.cz/)
Plotting traits on trees — code

```r
# See options for plotting functions
?plotTree.wBars  # There are more variants available
?dotTree
?plotSimmap

# Plot the trees
# Tip labels with bars with length proportional to character values
plotTree.wBars(tree=shorebird.tree, x=shorebird.mmass, tip.labels=TRUE)
# Tip labels with dots with size proportional to character values
dotTree(tree=shorebird.tree, x=shorebird.mmass, tip.labels=TRUE,
        type="cladogram")
```
Plotting traits on trees — plots

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
Evolutionary tasks

Tasks

1. **Browse** [http://blog.phytools.org/](http://blog.phytools.org/) and see if you find some interesting method to display your data. If so, try it with any suitable data.

2. **See relevant training datasets**
   ```
   data(package=c("adephylo", "caper", "geiger","phytools"))
   ```
   and select some training data set (or use your own data) to try at least 1–2 of the above methods.
Process more data

Not all combinations and possibilities were shown...

Tasks

1. Try to do some analysis with another introduced toy data
2. Try some of the introduced analysis with your own custom imported data

Remember...

- Working code can be easily recycled to process another data in similar way
- R is always moving forward — new and new options are arising — be opened for news and search them on the Internet
- Previous examples are not covering all possibilities...
- It is crucial to be able to edit the introduced commands to be able to handle your data
- Check help pages of the functions for more options what to do with your data
Final topics
General remarks about graphics, introduction to scripting, documentation and help resources, overview of packages

The end
Graphics
GitHub
Scripts
MetaCentrum
Functions
Loops
If-else branching
Solving problems
Resources
Summary
The end
Direct saving of plots to disk

Useful e.g. if plot should be bigger than screen, requires special settings, if done in batch, script, etc.

```r
# Output figure will be saved to the disk as OutputFile.png
png(filename="OutputFile.png", width=720, height=720, bg="white")
# Here can go any number of functions making plots...
plot(...) # Whatever...
# When using plotting commands, nothing is shown on the screen
# The final plot(s) will be saved by:
dev.off() # Closes graphical device - needed after use of plotting
    # functions png(), svg(), pdf(), ... followed by any
    # function like plot() to write the file(s) to the disk
filename="OutFiles_%03d.png" # Returns list of files named
    # OutFiles_001.png, OutFiles_002.png, ...
    # Useful for functions returning more
    # graphs.
?png # These functions have various possibilities to set size, whatever.
?svg # Exact possibilities of all 3 functions vary from system to system
?pdf # according to graphical libraries available in the computer.
```
Graphical packages

- Basic plotting functions in R are very limited...
  - The usage is simple, but anything more complicated requires extensive coding (plenty of examples were shown in the course)...
  - It can be tricky to get desired figure — some magic use to be needed...

- There are plenty of graphical packages...

- Advanced functions we used internally by used packages are `lattice` (web), `gplot` and `ggplot2` (web)
  - They have enormous possibilities, it is large topic for another long course...

- `par()` sets graphical parameters for following plots (splitting into panes, style of lines, points, text — see `pch`, `lwd`, `lty`, `cex`, `mai`, `mar`, `mfcol`, `mfrow`, ...) — see help pages...

- Most important low-level functions are `points`, `lines`, `text`, `abline`, `legend`, `axis`, `axes`, `arrows`, `box` — see help pages...
Install package from GitHub

- **GitHub** is currently probably the most popular platform to host development of open-source projects — plenty of R packages are there
- **Git** is version controlling system — it traces changes among all versions — absolutely crucial for any software development
- Normal stable version of package is installed from repository as usual, but sometimes it can be useful to get latest developmental version (e.g. when it fixes some bug and new release is not available yet)

```r
# Needed library
require(devtools)
dev_mode(on = TRUE)

# Install selected package from GitHub (user/project)
install_github("thibautjombart/adegenet")

# when finished go back to normal version
dev_mode(on = FALSE)
```
R script and its running from command line I

- R script is just plain `TXT` file with `.r` (e.g. `myscript.r`) extension containing list of R commands
- Mark all user comments with `#` on the beginning
- In command line (Linux/macOS/Windows/...) use
  - `Rscript myscript.r` to work **interactively** — all output is written to the terminal (screen; as usual), user can be asked for some values, ...
  - `R CMD BATCH myscript.r` to let it run **non-interactively** — all output is written into file `myscript.Rout`, terminal (screen) is clean and user can not influence the script anyhow — e.g. on MetaCentrum — be sure the script doesn’t require user input and works correctly
- Script ends when there is any error or on the end of the file
- When working on both Windows and macOS/Linux, take care about end of lines, and in case of usage of accented characters (e.g. for labels) also about encoding
Windows and UNIX (Linux, macOS, ...) have different internal symbol for new line

Use UNIX command line utilities `dos2unix myscript.r` or `unix2dos myscript.r` to get correct ends of lines for target system

Linux and macOS use to use UTF-8, Windows use regional encoding, e.g. Czech CP-1250 — use advanced text editor (slide 11) to convert the encoding, or use some command line tool, like `iconv`
CESNET and MetaCentrum

- **CESNET** (češky) is organization of Czech universities, Academy of Science and other organizations taking care about Czech backbone Internet, one of world leading institutions of this type
- CESNET provides various services (češky)
  - Massive computations — MetaCentrum (češky)
  - Large data storage (češky)
  - FileSender (češky) to be able to send up to 1.9 TB file
  - Cloud (češky) — computing (HPC) cloud similar to e.g. Amazon Elastic Compute Cloud (EC2), Google Compute Engine or Microsoft Azure
  - ownCloud (češky) to backup and/or sync data across devices (default capacity is 100 GB, user may ask for more) — similar to e.g. Dropbox, Google Drive or Microsoft OneDrive
- Check my special course [https://soubory.trapa.cz/linuxcourse/](https://soubory.trapa.cz/linuxcourse/)
Running R tasks on MetaCentrum

- There are only some R packages, to get more create own package library and use it in scripts (see e.g. `/software/R/4.0.0/gcc/lib/R/library/`)

- Be careful about paths!

- In the `metacentrum.sh` script load R module `add R-4.0.0-gcc` and start there R script as usually `R CMD BATCH script.r`

  1. Login to selected front node via SSH
  2. Create somewhere new directory for R packages `mkdir rpkgs` (or use default `~/R/`)
  3. Start R `R` and install all R packages needed for the task — install them into the `rpkgs` directory `install.packages(pkgs=..., lib="rpkgs")`
  4. In the R script `*.r` load the packages from the `rpkgs` directory `library(package=..., lib.loc="/storage/.../rpkgs")`
  5. Ensure all needed outputs are saved from the R script
OnDemand
Applications in web browser

- It allows to run selected interactive application (e.g. RStudio) in web browser
- Applications start in /storage/brno3-cerit/home/$USER/ — ensure to have everything needed there
Jupyter Notebook

- Web service allowing to record code as well as its output for languages like BASH, R, Python, ...
- Convenient for recording and sharing code, interactive work, ...
- Use Jupyter Hub for MetaCentrum users
  - Data are in /storage/brno2/home/USER/
- Available also as part of OnDemand (previous slide), or bit experimental CERIT hub, which allows to select custom storage and also custom docker image (see documentation)
Simple function

- Functions pack sets of commands for more comfortable repeated usage
- People more interested in R programming need to check special courses and/or documentation

```r
# General syntax:
MyFunction <- function (x, y) {
  # Any commands can be here...
  x + y
}
# Use as usually:
MyFunction(5, 8)
MyFunction(1, 4)
MyFunction(x=4, y=7)
MF <- MyFunction(9, 15)
MF # See it works
```
Simple loop — for cycles

- Loops repeat one task given number of times
- Variable $i$ has changing value for every repetition — useful for working with indexes (within lists, matrices, ...)
- It is possible to use variables or numeric output of functions in `from:to` expression — this is very variable
- In `for` loop we know in advance the number of repetitions (cycles), in `while` loop (next slide) we don’t

```r
# Simplest loop - print value of "i" in each step
# "i" is commonly used for various indexing
for (i in 1:5) { print(i) }
[[1]] 1  # This is the value of "i"
[[1]] 2
[[1]] 3
[[1]] 4
[[1]] 5
```

Vojtěch Zeisek (https://trapa.cz/)
For and while loops

```r
# In every step modify value of variable "X" (add 1 to previous value)
X <- 0 # Set initial value
for (i in 1:10) {
  # Any commands can be here...
  print("Loop turn") # Some message for user
  print(i) # Print number of turn - note how it is increasing
  X <- X+i # Rise value of "X" by current value of "i" (previous line)
  print(paste("Variable value:" , X)) } # Print current value of "X"
for (i in 10:5) { print(i) } # Can be descending...
# Work on each item of a list object
# Print length of each sequence in gunnera.dna
for (L in 1:length(gunnera.dna)) {
  print(length(gunnera.dna[[L]])) }
# While loop - it is done while the condition is valid
# While value of "Q" is < 5 (starting from 0), print it and add 1
Q <- 0
while (Q < 5) { print(Q <- Q+1) }
```

Vojtěch Zeisek (https://trapa.cz/)
Molecular data in R
January 31 to February 3, 2022
If-else branching I

- Basic method of branching the code — **if** the condition is met, **then** one branch is followed, **else** — in any other case — the other branch of the code is executed

- **else** part can be missing — the code is executed **only if** the condition is met

```r
XX <- seq(from=-3, to=6.5, by=0.1)
XX
YY <- c()
for (II in 1:length(XX)) {
    if(XX[II] <= 2) {
        # Executed for XX <= 2
        YY[II] <- XX[II]^2
    } else {
        # if(XX[II] > 2) # Executed for XX > 2
        YY[II] <- 6-XX[II]
    }
}
YY # See next two slides for the end of the example
```
If-else branching II

```r
# Plot from example from previous slide
plot(XX, YY) # See the result
# Or (different possibility to get very same result)
# Note "XX" is reused from the previous slide
CC <- function(AA) {
  if(AA <= 2) {
    # Executed for XX <= 2
    BB <- AA^2
  } else {
    # Executed for XX > 2
    BB <- 6 - AA
  }
  return(BB) # The output value
}
CC # Previously, "YY" contained values to plot made by the for loop,
    # here "CC" contains function to by used by sapply() when plotting
plot(sapply(XX, CC)) # See the result
# The plot (same for both ways how to do it) is on next slide
```
Output of the if-else branching example
Most common problems and their solutions I

- Something was not found (object, function, file, ...)
  - Check spelling of all methods, parameters, etc.
  - Check all paths (slide 86)
  - Check if all required objects were correctly created in previous steps
  - Check if all required libraries are loaded

- Unknown parameter, method, etc.
  - Check spelling of all parameters, consult manual pages
  - Check if all required libraries are loaded

- Graphics is not plotted correctly
  - Graphical window is too small (common problem with RStudio on screen with low resolution)
    — try to enlarge plotting window/pane
  - Reset graphical settings from some previous plot(s) by (repeated) calling of `dev.off()`

- R does nothing (but CPU is not extensively used)
  - R is waiting for some user input
Most common problems and their solutions II

• If command line starts with +, previous line was not completed correctly (e.g. missing closing bracket ) — check syntax, add it and hit Enter

• Some functions show plots and ask user for decision what to do (e.g. DAPC, slide 219) — write the answer into command line or special window and hit Enter

• Some functions are not (without extra work) usable on all operating systems, some don’t work correctly in GUI
  • Check manual and/or some on-line forum (slide 389 and onward)

• R and packages are more or less changing from version to version
  • Old methods can became outdated and not working anymore
  • Check release notes and change logs for new versions, manual pages and on-line forums (slide 389 and onward)
  • Generally, follow news for your topic (appropriate mailing list, ...)
  • Unmaintained packages are archived, new created...
**How to ask for help**

- **Never ever** ask simple silly lazy questions you can quickly find in manual or web
- People on mailing lists and forums respond voluntarily in their spare free time — do not waste it — be polite, brief and informative
- Be as specific and exact as possible
  - Write *exactly* what you did ("It doesn’t work!” is useless...)
  - Copy/paste your commands and their output, especially error messages — they are keys to solve the problem
  - Try to search web for the error messages (or their parts)
  - Try to provide minimal working example — add at least part of your data (if applicable) so that the problem is reproducible
  - Specify version(s) of R/packages, operating system and/or another important details — authors will commonly insist on newest versions: add outputs of `sessionInfo()` and `packageVersion("PackageName")`
  - Try to find forum most appropriate for your question (does package have dedicate forum?)
How to ask for help II

- **R is free as freedom of speech — not as free beer!**
  - As soon as you don’t pay for support, you can’t blame anyone for lack of responses
  - There are plenty of reasons some package/function doesn’t work — usage/data author didn’t expect, unsupported operating system, author’s mistake, user’s mistake, ...
  - Authors wish their software to be useful — constructive feedback, reporting bugs and wishes is welcomed, but it must be provided in the way useful for the developer

- **R functions commonly lack control of input data — error messages are returned by internal functions**
  - They are not straightforward
  - It requires some training and experience to be quickly able to find what is going on
  - Always carefully read error messages and think about them

- Imagine you should answer — which information do you need?
Where to look for the help I

Question must have certain form!

Before asking, ensure your question is in answerable form — slide 387.

• Sloppily asked question can’t be answered at all...
• Check documentation, manuals and search the Internet before asking

• R homepage https://www.r-project.org/ and packages https://CRAN.R-project.org/web/packages/ (with documentation and links)
• R phylogeny mailing list https://stat.ethz.ch/mailman/listinfo/r-sig-phylo
• R genetics mailing list https://stat.ethz.ch/mailman/listinfo/r-sig-genetics
Where to look for the help II

- Bioconductor home page https://bioconductor.org/ and support forum https://support.bioconductor.org/
- Adegenet help mailing list https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/adegenet-forum and GitHub page https://github.com/thibautjombart/adegenet/wiki
- Poppr forum https://groups.google.com/g/poppr/about
- R help mailing list https://stat.ethz.ch/mailman/listinfo/r-help
- R announce mailing list https://stat.ethz.ch/mailman/listinfo/r-announce
- R ecology mailing list https://stat.ethz.ch/mailman/listinfo/r-sig-ecology
Where to look for the help III

- R at StackOverflow StackExchange (for programmers)
  https://stackoverflow.com/questions/tagged/r
- R at CrossValidated StackExchange (for statisticians, mathematicians, etc.)
  https://stats.stackexchange.com/questions/tagged/r
- Biostars — general bioinformatics forum https://www.biostars.org/
- Biology — general forum about biology at StackExchange
  https://biology.stackexchange.com/
- Bioinformatics at StackExchange
  https://bioinformatics.stackexchange.com/
- Do not hesitate to ask on the forum or contact author of package with which you have problem, preferably through some public forum or mailing list, they usually respond quickly and helpfully... — they wish their packages to be working and useful
- Uncle Google is your friend here ("how to XXX in R")...
Citations

• To correctly cite R launch `citation()` and see information there — it is slightly different for every version of R

• Cite used packages — launch `citation("PackageName")` — if this information is missing, go to its manual page and/or homepage and find the information there

• Most of packages implementing methods are created by scientists — they like to be cited :-)

• Packages/functions commonly provide various methods to calculate desired task — check function’s help page (`?FunctionName`) and find references there and cite them accordingly

• Check original papers to fully understand respective method
Further reading
The most important books for our topics

Emmanuel Paradis
Analysis of Phylogenetics and Evolution with R, second edition
Springer, 2012
http://ape-package.ird.fr/APER.html

Michael J. Crawley
The R Book, second edition
Wiley, 2012

Paurush Praveen Sinha
Bioinformatics with R Cookbook
Packt Publishing, 2014

Anthony R. Ives
Mixed and Phylogenetic Models: A Conceptual Introduction to Correlated Data
Leanpub, 2018
https://leanpub.com/correlateddata (free to read on-line)
Learning resources I

- R homepage [https://www.r-project.org/](https://www.r-project.org/) and packages [https://CRAN.R-project.org/web/packages/](https://CRAN.R-project.org/web/packages/) (with documentation and links)
- Books about R [https://www.r-project.org/doc/bib/R-books.html](https://www.r-project.org/doc/bib/R-books.html)
- List of R documentation [https://CRAN.R-project.org/manuals.html](https://CRAN.R-project.org/manuals.html)
- Bioconductor help pages [https://master.bioconductor.org/help/](https://master.bioconductor.org/help/)
- R phylogenetics wiki [https://www.r-phylo.org/wiki/Main_Page](https://www.r-phylo.org/wiki/Main_Page)
- R phylogenetics at CRAN [https://CRAN.R-project.org/web/views/Phylogenetics.html](https://CRAN.R-project.org/web/views/Phylogenetics.html)
- Integrated documentation search [https://www.rdocumentation.org/](https://www.rdocumentation.org/)
- Better interface to R and packages documentation and integrated search [https://rdrr.io/](https://rdrr.io/)
Learning resources II

- RForge package repository https://r-forge.r-project.org/ (with documentation)
Learning resources III

- Adegenet web [https://adegenet.r-forge.r-project.org/] and GitHub page [https://github.com/thibautjombart/adegenet/wiki]
- APE home page [http://ape-package.ird.fr/]
- Information and manual about pegas [http://ape-package.ird.fr/pegas.html]
- Phytools [https://phytools.org/], its blog [http://blog.phytools.org/] and GitHub page [https://github.com/liamrevell/phytools]
- Poppr documentation [https://grunwaldlab.github.io/poppr/reference/poppr-package.html]
- Population Genetics in R [https://popgen.nescent.org/] by Kamvar et al.
Learning resources IV

- Phangorn resources [https://CRAN.R-project.org/package=phangorn](https://CRAN.R-project.org/package=phangorn)
- The R journal [https://journal.r-project.org/](https://journal.r-project.org/)
- R-bloggers — aggregation of R blogs [https://www.r-bloggers.com/](https://www.r-bloggers.com/)
- R on The Molecular Ecologist [https://www.molecularecologist.com/category/software/r/](https://www.molecularecologist.com/category/software/r/)
Learning resources V

- R tutorial [https://www.r-tutor.com/](https://www.r-tutor.com/)
- Cookbook for R [http://www.cookbook-r.com/](http://www.cookbook-r.com/)
- R for open big data [https://ropensci.org/](https://ropensci.org/)
- Statistics with R [http://zoonek2.free.fr/UNIX/48_R/all.html](http://zoonek2.free.fr/UNIX/48_R/all.html)
- ggplot2 (the most powerful graphical library used by many packages) information [https://ggplot2.tidyverse.org/](https://ggplot2.tidyverse.org/)
- plyr documentation [https://plyr.had.co.nz/](https://plyr.had.co.nz/) — manipulation with data (split-apply-combine)
Learning resources VI

- Leaflet for R [https://rstudio.github.io/leaflet/](https://rstudio.github.io/leaflet/)
- Learning R blog [https://learnr.wordpress.com/](https://learnr.wordpress.com/)
- Quick-R learning resource [https://www.statmethods.net/](https://www.statmethods.net/)
- Visualizing and annotating phylogenetic trees with ggtree [https://4va.github.io/biodatasci/r-ggtree.html](https://4va.github.io/biodatasci/r-ggtree.html)
- Uncle Google is your friend ("how to XXX in R")...
- R packages commonly contain vignettes (tutorials) — list them by `vignette()` and load selected by `vignette("VignetteName")`
- And finally: Reading documentation is not wasting of time! ;-)

Vojtěch Zeisek ([https://trapa.cz/](https://trapa.cz/))
Packages we used... I

We used following packages — but not all functions — explore them for more possibilities

- **ade4**: multivariate data analysis and graphical display (enhancements: ade4TkGUI — GUI, adegraphics — extra graphical functions, commonly used internally)
- **adegenet**: exploration of genetic and genomic data
- **adephylo**: multivariate tools to analyze comparative data
- **adespatial**: multiscale spatial analysis of multivariate data
- **akima**: cubic spline interpolation methods for irregular and regular grids data
- **ape**: analysis of phylogenetics and evolution
- **BiocManager**: access the Bioconductor project package repository
- **caper**: phylogenetic comparative analysis
- **corrplot**: graphical display of a correlation or general matrix
- **devtools**: package development tools, access to GitHub
Packages we used... II

We used following packages — but not all functions — explore them for more possibilities

- `gee`: generalized estimation equation solver
- `geiger`: fitting macroevolutionary models to phylogenetic trees
- `Geneland`: stochastic simulation and MCMC inference of structure from genetic data
- `ggplot2`: data visualizations using the Grammar of Graphics
- `gplots`: plotting data
- `hierfstat`: estimation and tests of hierarchical F-statistics
- `ips`: interfaces to phylogenetic software
- `kdetrees`: non-parametric method for identifying potential outlying observations in a collection of phylogenetic trees
- `lattice`: Trellis graphics, with an emphasis on multivariate data
- `mapdata`: supplement to maps, larger and/or higher-resolution databases
Packages we used... III

We used following packages — but not all functions — explore them for more possibilities

- mapplots: extra map plotting, pie charts and more
- mapproj: converts latitude/longitude into projected coordinates
- maps: draws geographical maps
- maptools: manipulating and reading geographic data
- nlme: fits and compares Gaussian linear and nonlinear mixed-effects models
- PBSmapping: spatial analysis tools
- pegas: population and evolutionary genetics analysis
- phangorn: phylogenetic analysis
- philentropy: over 40 optimized distance and similarity measures for comparing probability functions
- phylobase: phylogenetic structures and comparative data
Packages we used... IV
We used following packages — but not all functions — explore them for more possibilities

- **phyloch**: interfaces and graphic tools for phylogenetic data
- **phytools**: phylogenetic analysis, comparative biology, graphics
- **picante**: integrates phylogeny and ecology
- **plotrix**: various labeling, axis and color scaling functions
- **poppr**: genetic analysis of populations with mixed reproduction
- **raster**: reading, writing, manipulating, analyzing and modeling of gridded spatial data
- **rentrez**: interface to the NCBI, allowing to search databases like GenBank
- **rgdal**: bindings to the Geospatial Data Abstraction Library and access to projection/transformation operations library
- **RgoogleMaps**: interface to query the Google server for static maps and uses the map as a background image to overlay plots
Packages we used... V

We used following packages — but not all functions — explore them for more possibilities

- **Rmpi**: interface (wrapper) to MPI (used for parallel processing)
- **rworldmap**: mapping global data (and extra data in rworldxtra)
- **seqinr**: exploratory data analysis and data visualization for biological sequence
- **shapefiles**: read and write ESRI shapefiles
- **snow**: simple parallel computing
- **sos**: searches contributed R packages
- **sp**: classes and methods for spatial data
- **spdep**: spatial dependence: weighting schemes, statistics and models
- **splancs**: display and analysis of spatial point pattern data
- **StAMPP**: statistical analysis of mixed ploidy populations
## Packages we used... VI

We used following packages — but not all functions — explore them for more possibilities

- **TeachingDemos**: demonstrations for teaching and learning, enhanced plotting of text
- **tripack**: constrained two-dimensional Delaunay triangulation
- **vcfR**: import/export, basic checking and manipulations of VCF
- **vegan**: community ecology
Another interesting packages (we did not use)... 1

For your own explorations...

- **adhoc**: ad hoc distance thresholds for DNA barcoding identification
- **addTaxa**: adding missing taxa to phylogenies
- **admixr**: interface for running ADMIXTOOLS
- **apex**: analysis of multiple gene data
- **apTreeshape**: analysis of phylogenetic tree topologies
- **BAMMtools**: analyzing and visualizing complex macroevolutionary dynamics on phylogenetic trees
- **betapart**: partitioning beta diversity into turnover and nestedness components
- **Biodem**: biodemography
- **Biostrings**: string matching algorithms, and other utilities, for fast manipulation of large biological sequences or sets of sequences
Another interesting packages (we did not use)... II

For your own explorations...

- **convevol**: quantifies and assesses the significance of convergent evolution
- **corHMM**: analysis of binary character evolution
- **DAMOCLES**: maximum likelihood of a dynamical model of community assembly
- **dbscan**: implementation of several density-based algorithms (**DBSCAN**, **OPTICS**, etc.)
- **DDD**: diversity-dependent diversification
- **dendextend**: extending dendrogram objects, comparing trees
- **distory**: geodesic distance between phylogenetic trees
- **diversitree**: comparative phylogenetic analysis of diversification
- **diveRsity**: calculation of both genetic diversity partition statistics, genetic differentiation statistics, and locus informativeness for ancestry assignment
- **dplyr**: various manipulations with data frames
Another interesting packages (we did not use)... III

For your own explorations...

- **ecodist**: dissimilarity-based functions for ecological analysis, spatial and community data
- **evobiR**: comparative and population genetic analysis
- **factoextra**: extract and visualize the results of multivariate data analyses
- **fields**: tools for spatial data
- **genetics**: population genetics
- **genotypeR**: design of genotyping markers from VCF files, output of markers for multiplexing on various platforms, various QA/QC and analysis
- **geomorph**: geometric morphometric analysis of 2D/3D landmark data
- **geosphere**: spherical trigonometry for geographic applications — distances and related measures for angular (longitude/latitude) locations
- **ggtree**: visualization and annotation of phylogenetic trees (documentation)
Another interesting packages (we did not use)... IV

For your own explorations...

- **HardyWeinberg**: statistical tests and graphics for HWE
- **heatmaply**: interactive cluster heat maps
- **HMPTrees**: models, compares, and visualizes populations of taxonomic tree objects
- **hwde**: models and tests for departure from HWE and independence between loci
- **HyPhy**: macroevolutionary phylogenetic analysis of species trees and gene trees
- **IRanges**: infrastructure for manipulating intervals on sequences
- **iteRates**: iterates through a phylogenetic tree to identify regions of rate variation
- **knitr**: general-purpose tool for dynamic report generation
- **LDheatmap**: graphical display, as a heat map, of measures of pairwise linkage disequilibrium between SNPs
- **LEA**: landscape and ecological association studies
Another interesting packages (we did not use)... V

For your own explorations...

- **leaflet**: interactive web maps with the JavaScript Leaflet library
- **Linarius**: dominant marker analysis with mixed ploidy levels
- **markophylo**: markov chain models for phylogenetic trees
- **MASS**: functions and data sets for venables and ripley’s MASS
- **MCMCglmm**: MCMC generalized linear mixed models
- **microseq**: microbial sequence data analysis (using tibble)
- **MINOTAUR**: multivariate visualization and outlier analysis
- **MonoPhy**: visualization and exploration of monophyletic clades on a tree
- **MPSEM**: modeling phylogenetic signals using eigenvector maps
- **muscle**: multiple sequence alignment with MUSCLE
Another interesting packages (we did not use)... VI

For your own explorations...

- **mvMORPH**: multivariate comparative tools for fitting evolutionary models to morphometric data
- **mvtnorm**: multivariate normal and t probabilities
- **onemap**: molecular marker data from model (backcrosses, F2 and recombinant inbred lines) and non-model systems (outcrossing species), constructions of genetic maps
- **OpenStreetMap**: plotting OpenStreetMap maps (various layers)
- **ouch**: Ornstein-Uhlenbeck models for evolution along a phylogenetic tree
- **OUwie**: analysis of evolutionary rates in an OU framework
- **paleoPhylo**: assess how speciation, extinction and character change contribute to biodiversity
- **paleotree**: paleontological and phylogenetic analysis of evolution
Another interesting packages (we did not use)... VII

For your own explorations...

- **paleoTS**: analyze paleontological time-series
- **ParallelStructure**: running analysis in the population genetics software STRUCTURE
- **pastis**: phylogenetic assembly with soft taxonomic inferences
- **PBD**: protracted birth-death model of diversification
- **pcadapt**: PCA and search for loci responsible for the grouping (no support for mixing ploidy levels), uses VCF
- **PCPS**: principal coordinates of phylogenetic structure
- **permute**: restricted permutation designs
- **Phybase**: read, write, manipulate, simulate, estimate, and summarize phylogenetic trees (gene trees and species trees)
- **phyclust**: phylogenetic clustering
Another interesting packages (we did not use)... VIII

For your own explorations...

- **phyloclim**: integrating phylogenetics and climatic niche modeling
- **PHYLOGR**: manipulation and analysis of phylogenetically simulated data sets and phylogenetically based analysis using GLS
- **phyloland**: models a space colonization process mapped onto a phylogeny
- **phylolm**: phylogenetic linear models and phylogenetic generalized linear models
- **phyloTop**: calculating and viewing topological properties of phylogenetic trees
- **phylotools**: supermatrix for DNA barcodes using different genes
- **plotly**: creates interactive web graphics
- **plyr**: splitting, applying and combining data
- **pmc**: phylogenetic Monte Carlo
Another interesting packages (we did not use)… IX

For your own explorations...

- **polyfreqs**: Gibbs sampling algorithm to perform Bayesian inference on biallelic SNP frequencies, genotypes and heterozygosity in a population of autopolyploids
- **polysat**: polyploid microsatellite analysis
- **RandomFields**: simulation of Gaussian fields (+ RandomFieldsUtils)
- **radiator**: RADseq data exploration, manipulation and visualization
- **rCharts**: interactive JS charts
- **RColorBrewer**: ColorBrewer palettes
- **rdryad**: access for Dryad web services
- **reshape2**: restructure and aggregate data
- **rMaps**: interactive maps
- **RPHAST**: interface to PHAST software for comparative genomics
Another interesting packages (we did not use)... X

For your own explorations...

- **RMesquite**: interoperability with Mesquite
- **Rphylib**: interface for PHYLIP
- **Rsamtools**: BAM, FASTA, BCF and tabix file import and manipulations
- **rwty**: tests, visualizations, and metrics for diagnosing convergence of MCMC chains in phylogenetics
- **sangeranalyseR**: analysis of Sanger sequence
- **sensiPhy**: sensitivity analysis for phylogenetic comparative methods, statistical and graphical methods that estimate and report different types of uncertainty
- **seqLogo**: sequence logos for DNA sequence alignments
- **SigTree**: identify and visualize significantly responsive branches in a phylogenetic tree
Another interesting packages (we did not use)... XI

For your own explorations...

- **SimRAD**: simulate restriction enzyme digestion, library construction and fragments size selection to predict the number of loci expected from most of the RAD and GPS approaches
- **SNPassoc**: SNPs-based whole genome association studies
- **SNPRelate**: parallel computing toolset for relatedness and principal component analysis of SNP data
- **snpStats**: classes and statistical methods for large SNP association studies
- **spatstat**: spatial point pattern analysis
- **splits**: delimiting species and automated taxonomy at many levels of biological organization
- **strap**: stratigraphic analysis of phylogenetic trees, palaeontology
- **strataG**: analyzing stratified population genetic data by vast range of methods, very powerful
Another interesting packages (we did not use)... XII

For your own explorations...

- **stringi** and **stringr**: character string processing, internally used by many packages
- **SYNCSA**: analysis of metacommunities based on functional traits and phylogeny of the community components
- **taxize**: taxonomic information from around the web
- **TESS**: simulation of reconstructed phylogenetic trees under tree-wide time-heterogeneous birth-death processes and estimation of diversification parameters under the same model
- **tidysq**: tidy approach to analysis of biological sequences
- **tmap**: various thematic maps
- **treebase**: discovery, access and manipulation of TreeBASE phylogeny
- **treeio**: read, parse and write various tree formats
- **TreePar**: estimating birth and death rates based on phylogeny
Another interesting packages (we did not use)... XIII
For your own explorations...

- **TreeSearch**: search for phylogenetic trees that are optimal using a user-defined criterion
- **TreeSim**: simulating phylogenetic trees
- **treespace**: exploration of distributions of phylogenetic trees
- **UpSetR**: visualizations of intersecting sets using a novel matrix design, along with visualizations of several common set, element and attribute related tasks
- **VariantAnnotation**: annotation of genetic variants (useful to filter VCF)
- **XVector**: representation and manipulation of external sequences

And more... R is continuously evolving and new packages are arising...
Orientation in so many packages...

- ...is not easy...
- Many methods are implemented in more packages
  - Quality and richness of implementations may vary a lot...
  - Same methods in different packages may require data in different formats/R classes (conversion use to be simple — but always see respective documentation)
- Anyone can create and submit R package...
  - Plenty of packages to choose from...
  - No restrictions (apart basic technical requirements in repositories) — quality may be variable...
- Follow news on R sites, mailing lists, journal articles introducing new packages, etc.
- Be open for new tools, explore, try, share your experience
The methods are over

• We went in more or less details through plenty of methods to work with molecular data to analyze phylogeny, population genetics, evolution and so on in R
• There are many more methods to try...
• It is nearly impossible to go in reasonable time through all relevant R tools — a lot of space for you
Our course is over...

...I hope it was helpful for You...

...any feedback is welcomed...

...happy R hacking...

... any final questions?

Typesetting using X\LaTeX \! on openSUSE GNU/Linux, January 29, 2022.