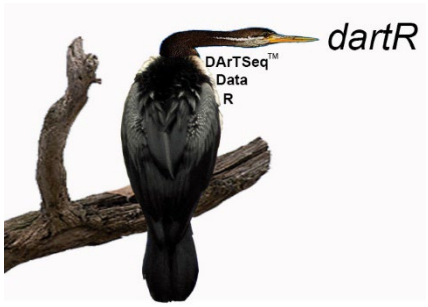
Wildlife Genomics@UC

UNIVERSITY OF CANBERRA

SNP Analysis using dartR



Installing dartR

Version 4

Copies of the latest version of this tutorial are available from:

The Institute for Applied Ecology
University of Canberra ACT 2601
Australia

Email: arthur.georges@biomatix.com.au

Citation: Gruber, B., Mijangos, J.L and Georges, A. (2025). Installing dartR. Version 4. Institute for Applied Ecology, University of Canberra, Canberra, ACT, 2617.

dartR is a collaboration between the University of Canberra, CSIRO and Diversity Arrays Technology, and is supported with funding from the ACT Priority Investment Program, CSIRO and the University of Canberra.

Contents

Installing dartR 5
For WINDOWS 5
For MACOS 5
For LINUX (Ubuntu 20.04) 6
About dartRverse 7
Rationale 7
Installing dartRverse 8
Installing additional dartR packages 8
Installing from GitHub repositories 9
Using legacy dartR 10
Using dartRverse 10
Testing your installation 11
Help 11

Installing dartR

For WINDOWS

1. Download and install **R** from **here**.
2. Download and install **RStudio** from **here**.
3. Download and install the latest version of **Rtools** from **here**. **Rtools** is a toolchain bundle used for building R packages from source (those that need compilation of C/C++ or Fortran code).
4. Install **dartRverse**; see the specific section for this step below.

For MACOS

1. Download and run the **R** installer from **here**.
2. Download and open the **RStudio app** from **here**. You must drag the application to the Applications folder to install it.
3. Install **Command-Line tools**. Open the **Terminal app** and type:

```
xcode-select --install
```

4. Install **Homebrew** by typing in the **Terminal app** the code below. **Homebrew** is a management system that simplifies software installation on macOS and Linux.

```
/bin/bash -c "$(curl -fsSL
    https://raw.githubusercontent.com/Homebrew/ins
    tall/HEAD/install.sh)"
```

5. Add **Homebrew** to your PATH **if prompted**. After installing **Homebrew**, you might see that the last message, "==> Next steps:" describes further instructions. Follow these instructions and type them in the **Terminal app**. These instructions should be something similar to the following:

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"'
    >> /Users/m1/.zprofile

eval "$(/opt/homebrew/bin/brew shellenv)"
```

6. Install **GNU Compiler Collection** (GCC) by typing in the **Terminal app**:

```
brew install gcc
```

7. Check the version of **GCC** you installed in the previous step by typing in the **Terminal app**:

```
gcc --version
```

8. Make a **symlink** (a symbolic link that points to another file or folder on your computer) from your GCC version to gcc using the **Terminal app**.

Replace the X in the code below with the GCC version you checked in the previous step.

```
ln -s gcc-X gcc
```

9. Install **GDAL** using Homebrew. **GDAL** is a translator library for raster and vector geospatial data. In the first command, we install the headers files first to avoid **GDAL** fail.

```
brew install gdal --HEAD
```

```
brew install gdal
```

10. Open **RStudio.** If you get warning messages such as **Setting LC_CTYPE failed, using "C"** or the error: **tar: Failed to set default locale**, type the command below in the **R console** and then restart your RStudio session. Find more information **here.**

```
system('defaults write org.R-project.R force.LANG
    en_US.UTF-8')
```

11. Create a makevars file and put the path to libgfortran into FLIBS. Open **RStudio** and type in the **R console:**

```
dir.create('~/.R')
```

```
write.table("FLIBS=`gfortran -print-search-dirs |
    grep '^libraries:' | sed 's|libraries: =||' |
    sed 's|:| -L|g' | sed 's|^|-L|'`",
    file='~/.R/Makevars', col.names = FALSE,
    row.names = FALSE, quote = FALSE)
```

12. Install **dartRverse**; see the specific section for this step below.

For LINUX (Ubuntu 20.04)

1. Install **R** by following **these instructions**.

2. Install the **r-base-dev** package. Open **Terminal** and type the code below. This package compiles R packages from source (those that need compilation of C/C++ or Fortran code).

```
sudo apt-get install r-base-dev
```

3. Install the various packages that are required in LINUX by typing in Terminal:

```
sudo apt install libcurl4-openssl-dev libssl-dev
    libfontconfig1-dev libxml2-dev libharfbuzz-dev
    libfribidi-dev libfreetype6-dev libpng-dev
    libtiff5-dev libjpeg-dev gdal-bin proj-bin
    libgdal-dev libproj-dev libgmp3-dev jags
```

4. Download **RStudio** from **here**.

5. Install **RStudio** using the code below in **Terminal** specifying the location of the file that you downloaded in the previous step, for example:

```
sudo gdebi '/home/vdiuser/Downloads/rstudio-
    2022.07.1-554-amd64.deb'
```

6. Indicate where **RStudio** should search for the **R** executable. Locate the **R** executable first, then use the environment variable RSTUDIO_WHICH_R to specify the executable location and add it to the profile file (the shell script file that gets executed after login). Enter the following command in **Terminal** after updating the location of your specific R executable. Note that ~ is a shortcut to the current user's home directory.

```
echo 'export RSTUDIO_WHICH_R=/opt/R/4.1.3/bin/R'
    >> ~/.profile
```

7. Source the **profile** file to update the change from the previous step. Type in **Terminal**:

```
source ~/.profile
```

8. Open **RStudio** by typing in **Terminal** (not from the desktop Applications menu):

```
rstudio
```

9. Install **dartRverse**; see the specific section for this step below.

About dartRverse

Rationale

We have developed a new suite of packages called **dartRverse** that will replace the previous **dartR** package, which will no longer be supported.

The **dartRverse** is meant to be a 1:1 replacement of **dartR**, and only in very few instances will minor changes be needed to update previous code. For example, we reworked the "output" of all functions, now being able to save figures in specified folders, which can be quickly recovered and manipulated for further tweaking.

The main reasons for splitting **dartR** into several packages and developing the **dartRverse** suit of packages was the difficulty of maintaining **dartR** because of limits imposed by CRAN, but also limits due to the long time to test new functions when compiling the package. Therefore, we were forced to split the package into several smaller packages. This has several advantages and no disadvantages in the best of all cases.

- Easier maintenance.
- Faster development.
- Less dependencies on other packages.
- Easier to become a contributor.
- Have your own package developed that can be branded as part of the **dartRverse**.

The main paradigm was also that for our users, nothing (or at as little as possible) needs to change, and all existing code and scripts should still work. Also, the installation process was meant to be straightforward, and finally, the coexistence between **dartR** and **dartRverse** packages should be possible (for the interim until **dartR** will no longer be supported).

Installing dartRverse

```
# Install the necessary Bioconductor packages
  install.packages("devtools")
  install.packages("BiocManager")
  BiocManager::install("SNPRelate")
# Install dartRverse (dartRverse) & core (dartR.base,
    dartR.data)
  install.packages("dartRverse")
```

The **dartRverse** package is the first package to be installed, and its only purpose is to support the installation of the other **dartRverse** packages. It will also be the first port of call to provide links to tutorials and other documentation.

If everything works well, this should install two more packages from the core version of **dartRverse**, namely **dartR.base** and **dartR.data**. Those core packages have all the main functions of the old **dartR** package that deals with input, conversion, reporting and data filtering. Also, some base functions to analyse data (e.g., PCoA and Fst) are included.

Once the **dartRverse** package has been installed, we can load the package to check which part of the **dartRverse** has been installed.

```
library(dartRverse)
```

Installing additional dartR packages

Now, we can install the additional packages that are part of the **dartRverse**. You can install all or only the one you want, depending on your needs.

For example, if you are interested in additional functions to analyse population structure (e.g. run STRUCTURE or FastStructure) you can install the **dartR.popgen** package. If you are interested in functions that support data simulation, you can install the **dartR.sim** package.

You can check which packages are available and which of them you have installed by typing:

```
dartRverse::dartRverse_install()
```

The currently available packages are:

- **dartR.sim** (simulate SNP data).
- **dartR.popgen** (run population genetic analysis).
- **dartR.spatial** (run landscape genetic analysis).
- **dartR.captive** (estimate relatedness, support captive breeding).
- **dartR.sexlinked** (identify sex-linked markers).

So, to install the **dartR.sim** simply type:

```
install.packages("dartR.sim")
```

Installing from GitHub repositories

We make all of our packages available via CRAN. This is because CRAN packages follow stringent testing before they are allowed to be uploaded to CRAN and, therefore, are more likely to contain fewer errors than packages available on other repositories. Nevertheless, we also make our packages available during development via GitHub.

You can find the repositories under: **https://github.com/green-striped-gecko/dartR.base** [for the dartR.base package] and equivalent for the other packages.

You might want to install a package from GitHub because you want to use the latest version of the package. However, you should be aware that the packages on GitHub are not tested and, therefore, might contain errors. Also, the packages on GitHub might be updated daily and, therefore, might change without notice.

We use different branches, reflecting different stages of development and maturity.

- main (the main branch, which is equivalent to the current CRAN version)
- dev (development branch, which has the latest functions that might be in the next CRAN version but have not been tested yet)
- dev_name (These are branches of our leading developers used for testing and developing new functions. Installing functions from here might cause problems and should only be done if you know what you are doing)

dartRverse supports the installation of the GitHub version of the packages using the following syntax:

```
dartRverse_install(package = "dartR.base", rep =
    "github", branch = "dev")
```

This installs the dev branch of **dartR.base** from GitHub. All main and dev branches are tested to see if they can be installed (and some additional error checks via): **https://github.com/green-striped-gecko/dartRverse**

Please note that you should provide the package repository (github/cran), the branch (main, dev, dev_name) and version number in case you want to report a

bug. You can use the GitHub methods to report issues or our Google Groups Forum: **https://groups.google.com/d/forum/dartr**.

Owing to CRAN requirements, even after installing the **dartRverse** using this approach, some of the functions require additional installation of packages. This is not to tease users; we want to ensure the packages do not fall over because of an incompatibility of one function (often, users do not use all functions anyway). We are working on a command that installs all packages in one go for those users who want to have all functions available. However, this is not a priority at the moment. If you want to use a function that is unavailable, please install the required package. If you are unsure which package is required, please check the function's documentation or ask on our Google Groups Forum. We are happy to help.

Using legacy dartR

For whatever reason, you might want to use legacy **dartR** instead of the **dartRverse** packages [e.g. for initial testing]. The good news is that both packages can be installed next to each other, but you must ensure you detach the other package. This can be done in Rstudio via the package tab by unticking the packages. Please note that the order in which you untick the packages is essential. So, first the non-core packages (dartR.sim, dartR.popgen, dartR.captive, dartR.spatial), then **dartRverse** followed by the core packages dartR.base and finally dartR.data. If you do not follow the correct order, you get a warning message, and it will not detach the package. If you want to use code, type:

```
detach("package:dartR.sim", unload = TRUE)
detach("package:dartR.sexlinked", unload = TRUE)
detach("package:dartR.spatial", unload = TRUE)
detach("package:dartR.captive", unload = TRUE)
detach("package:dartR.popgen", unload = TRUE)
detach("package:dartRverse", unload = TRUE)
detach("package:dartR.base", unload = TRUE)
detach("package:dartR.data", unload = TRUE)
```

Now, you can load and use the legacy **dartR** package as before.

```
library(dartR)
```

To unload **dartR**, you can use the Packages tab as described before or the following code:

```
detach("package:dartR", unload = TRUE)
detach("package:dartR.data", unload = TRUE)
```

Using dartRverse

To use **dartRverse**, you can load the package and use it as before.

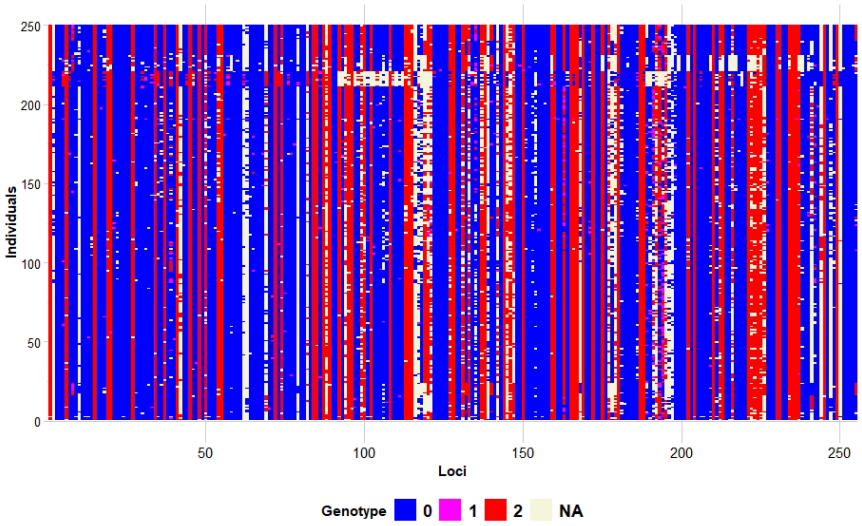
```
library(dartRverse)
```

Testing your installation

You can test your installation by running a function on the test dataset, for example

```
gl.smearplot(testset.gl)
```

to yield a plot as follows:



Help

You can post questions via Google group dartR if you are having trouble. You will need to join the group.

https://groups.google.com/forum/#!forum/dartr

If you are a developer and wish to contribute to the development of dartR, you should link through RStudio to

http://github.com/green-striped-gecko/dartR

develop locally and push to the /dev version. Raise any issues you believe require attention.