vegan FAQ

Frequently Asked Questions on R package **vegan**

Jari Oksanen

Table of Contents

1 Introduction

1.1 What is vegan?

Vegan is an R package for community ecologists. It contains the most popular methods of multivariate analysis needed in analysing ecological communities, and tools for diversity analysis, and other potentially useful functions. **Vegan** is not self-contained but it must be run under R statistical environment, and it also depends on many other R packages. **Vegan** is free software (`http://www.gnu.org/philosophy/free-sw.html`) and distributed under GPL2 license.

1.2 What is R?

R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.

R has a home page at `http://www.R-project.org/`. It is free software (`http://www.gnu.org/philosophy/free-sw.html`) distributed under a GNU-style copyleft (`http://www.gnu.org/copyleft/copyleft.html`), and an official part of the GNU (`http://www.gnu.org/`) project ("GNU S").

1.3 How to obtain vegan and R?

Both R and latest release version of **vegan** can be obtained through CRAN. Unstable development version of **vegan** can be obtained through GitHub. Formerly **vegan** was developed in R-Forge, but after moving to GitHub the R-Forge repository may be out of date.

1.4 What R packages vegan depends on?

Vegan depends on the **permute** package which will provide advanced and flexible permutation routines for **vegan**. The **permute** package is developed together with **vegan** in GitHub.

Some individual **vegan** functions depend on packages **MASS**, **mgcv**, **cluster**, **lattice** and **tcltk**. These all are base or recommended R packages that should be available in every R installation. **Vegan** declares these as suggested or imported packages, and you can install **vegan** and use most of its functions without these packages.

Vegan is accompanied with a supporting package **vegan3d** for three-dimensional and dynamic plotting. The **vegan3d** package needs non-standard packages **rgl** and **scatterplot3d**.

1.5 What other packages are available for ecologists?

CRAN Task Views include entries like `Environmetrics`, `Multivariate` and `Spatial` that describe several useful packages and functions. If you install R package **ctv**, you can inspect Task Views from your R session, and automatically install sets of most important packages.

1.6 What other documentation is available for vegan?

Vegan is a fully documented R package with standard help pages. These are the most authoritative sources of documentation (and as a last resource you can use the force and the read the source, as **vegan** is open source). **Vegan** package ships with other documents which can be read with `vegandocs` command (documented in the **vegan** help). The documents included in the **vegan** package are

- **Vegan** `NEWS`
- **Vegan** `ChangeLog`.

- This document (`FAQ-vegan.pdf`).
- Short introduction to basic ordination methods in **vegan** (`intro-vegan.pdf`).
- Introduction to diversity methods in **vegan** (`diversity-vegan.pdf`).
- Discussion on design decisions in **vegan** (`decision-vegan.pdf`).
- Description of variance partition procedures in function `varpart` (`partitioning.pdf`).

Web documents outside the package include:

- `https://github.com/vegandevs/vegan`: **vegan** homepage.
- `http://cc.oulu.fi/~jarioksa/opetus/metodi/vegantutor.pdf`: **vegan** tutorial.

1.7 Is there a Graphical User Interface (GUI) for vegan?

Roeland Kindt has made package **BiodiversityR** which provides a GUI for **vegan**. The package is available at CRAN. It is not a mere GUI for **vegan**, but adds some new functions and complements **vegan** functions in order to provide a workbench for biodiversity analysis. You can install **BiodiversityR** using `install.packages("BiodiversityR")` or graphical package management menu in R. The GUI works on Windows, MacOS X and Linux.

1.8 How to cite vegan?

Use command `citation("vegan")` in R to see the recommended citation to be used in publications.

1.9 How to build vegan from sources?

In general, you do not need to build **vegan** from sources, but binary builds of release versions are available through CRAN for Windows and MacOS X. If you use some other operating systems, you may have to use source packages. **Vegan** is a standard R package, and can be built like instructed in R documentation. **Vegan** contains source files in C and FORTRAN, and you need appropriate compilers (which may need more work in Windows and MacOS X).

1.10 Are there binaries for devel versions?

R-Forge runs daily tests on the devel package, and if passed, it builds source package together with Windows binaries. However, the R-Forge may be out of date, because **vegan** is mainly developed in GitHub. You can install R-Forge packages within R with command `install.packages("vegan", repos="http://r-forge.r-project.org/")`. If you use GUI menu entry, you must select or define the R-Forge repository.

1.11 Can I use vegan in Mac?

Yes, you can, and **vegan** binaries are available for Mac through CRAN. However, in some cases you may need to install extra tools packages available in MacOS tools pages: If you use function such as `orditkplot` that need `Tcl/Tk` you may need to install `tcltk` package. No Mac binaries of development versions are available in any repository we know.

1.12 How to report a bug in vegan?

If you think you have found a bug in **vegan**, you should report it to **vegan** maintainers or developers. The preferred forum to report bugs is GitHub. The bug report should be so detailed that the bug can be replicated and corrected. Preferably, you should send an example that causes a bug. If it needs a data set that is not available in R, you should send a minimal data set as well. You also should paste the output or error message in your message. You also should specify which version of **vegan** you used.

Bug reports are welcome: they are the only way to make **vegan** non-buggy.

Please note that you shall not send bug reports to R mailing lists, since **vegan** is not a standard R package.

1.13 Is it a bug or a feature?

It is not necessarily a bug if some function gives different results than you expect: That may be a deliberate design decision. It may be useful to check the documentation of the function to see what was the intended behaviour. It may also happen that function has an argument to switch the behaviour to match your expectation. For instance, function `vegdist` always calculates quantitative indices (when this is possible). If you expect it to calculate a binary index, you should use argument `binary = TRUE`.

1.14 Can I contribute to vegan?

Vegan is dependent on user contribution. All feedback is welcome. If you have problems with **vegan**, it may be as simple as incomplete documentation, and we shall do our best to improve the documents.

Feature requests also are welcome, but they are not necessarily fulfilled. A new feature will be added if it is easy to do and it looks useful, or if you submit code.

If you can write code yourself, the best forum to contribute to vegan is GitHub.

2 Ordination

2.1 I have only numeric and positive data but vegan still complains

You are wrong! Computers are painfully pedantic, and if they find non-numeric or negative data entries, you really have them. Check your data. Most common reasons for non-numeric data are that row names were read as a non-numeric variable instead of being used as row names (check argument `row.names` in reading the data), or that the column names were interpreted as data (check argument `header = TRUE` in reading the data). Another common reason is that you had empty cells in your input data, and these were interpreted as missing values.

2.2 Can I analyse binary or cover class data?

Yes. Most **vegan** methods can handle binary data or cover abundance data. Most statistical tests are based on permutation, and do not make distributional assumptions. There are some methods (mainly in diversity analysis) that need count data. These methods check that input data are integers, but they may be fooled by cover class data.

2.3 Why dissimilarities in vegan differ from other sources?

Most commonly the reason is that other software use presence–absence data whereas **vegan** used quantitative data. Usually **vegan** indices are quantitative, but you can use argument `binary = TRUE` to make them presence–absence. However, the index name is the same in both cases, although different names usually occur in literature. For instance, Jaccard index actually refers to the binary index, but **vegan** uses name `"jaccard"` for the quantitative index, too.

Another reason may be that indices indeed are defined differently, because people use same names for different indices.

2.4 Why NMDS stress is sometimes 0.1 and sometimes 10?

Stress is a proportional measure of badness of fit. The proportions can be expressed either as parts of one or as percents. Function `isoMDS` (**MASS** package) uses percents, and function `monoMDS` (**vegan** package) uses proportions, and therefore the same stress is 100 times higher in `isoMDS`. The results of `goodness` function also depend on the definition of stress, and the same `goodness` is 100 times higher in `isoMDS` than in `monoMDS`. Both of these conventions are equally correct.

2.5 I get zero stress but no convergent solutions in `metaMDS`

Most common reason is that you have too few observations for your NMDS. For `n` observations (points) and `k` dimensions you need to estimate `n*k` parameters (ordination scores) using `n*(n-1)/2` dissimilarities. For `k` dimensions you must have `n > 2*k + 1`, or for two dimensions at least six points. In some degenerate situations you may need even a larger number of points. If you have a lower number of points, you can find an undefined number of perfect (stress is zero) but different solutions. Conventional wisdom due to Kruskal is that you should have `n > 4*k + 1` points for `k` dimensions. A typical symptom of insufficient data is that you have (nearly) zero stress but no two convergent solutions. In those cases you should reduce the number of dimensions (`k`) and with very small data sets you should not use `NMDS`, but rely on metric methods.

It seems that local and hybrid scaling with `monoMDS` have similar lower limits in practice (although theoretically they could differ). However, higher number of dimensions can be used

in metric scaling, both with `monoMDS` and in principal coordinates analysis (`cmdscale` in **stats**, `wcmdscale` in **vegan**).

2.6 Zero dissimilarities in isoMDS

Function `metaMDS` uses function `monoMDS` as its default method for NMDS, and this function can handle zero dissimilarities. Alternative function `isoMDS` cannot handle zero dissimilarities. If you want to use `isoMDS`, you can use argument `zerodist = "add"` in `metaMDS` to handle zero dissimilarities. With this argument, zero dissimilarities are replaced with a small positive value, and they can be handled in `isoMDS`. This is a kluge, and some people do not like this. A more principal solution is to remove duplicate sites using R command `unique`. However, after some standardizations or with some dissimilarity indices, originally non-unique sites can have zero dissimilarity, and you have to resort to the kluge (or work harder with your data). Usually it is better to use `monoMDS`.

2.7 I have heard that you cannot fit environmental vectors or surfaces to NMDS results which only have rank-order scores

Claims like this have indeed been at large in the Internet, but they are based on grave misunderstanding and are plainly wrong. NMDS ordination results are strictly metric, and in **vegan** `metaMDS` and `monoMDS` they are even strictly Euclidean. The method is called "non-metric" because the Euclidean distances in ordination space have a non-metric rank-order relationship to community dissimilarities. You can inspect this non-linear step curve using function `stressplot` in **vegan**. Because the ordination scores are strictly Euclidean, it is correct to use **vegan** functions `envfit` and `ordisurf` with NMDS results.

2.8 Where can I find numerical scores of ordination axes?

Normally you can use function `scores` to extract ordination scores for any ordination method. The `scores` function can also find ordination scores for many non-**vegan** functions such as for `prcomp` and `princomp` and for some **ade4** functions.

In some cases the ordination result object stores raw scores, and the axes are also scaled appropriate when you access them with `scores`. For instance, in `cca` and `rda` the ordination object has only so-called normalized scores, and they are scaled for ordination plots or for other use when they are accessed with `scores`.

2.9 How the RDA results are scaled?

The scaling or RDA results indeed differ from most other software packages. The scaling of RDA is such a complicated issue that it cannot be explained in this FAQ, but it is explained in a separate pdf document on "Design decision and implementation details in vegan" that you can read with **vegan** command `vegandocs("decision")`.

2.10 cca fails with "data.frame expected" or ""site.env" missing"

This is not a **vegan** error message, but it comes from the `cca` function in the **ade4** package. There is an unfortunate name clash, and if you have loaded **ade4** after **vegan**, the **ade4** version of `cca` will mask the **vegan** version. You can use the **vegan** version using command `vegan::cca()`. If you do not need package **ade4**, you can detach it with command `detach(package:ade4)`.

2.11 Ordination fails with "Error in La.svd"

Constrained ordination (`cca`, `rda`, `capscale`) will sometimes fail with error message `Error in La.svd(x, nu, nv): error code 1 from Lapack routine 'dgesdd'.`

It seems that the basic problem is in the `svd` function of `LAPACK` that is used for numerical analysis in R. `LAPACK` is an external library that is beyond the control of package developers and R core team so that these problems may be unsolvable. It seems that the problems with the `LAPACK` code are so common that even the help page of `svd` warns about them

Reducing the range of constraints (environmental variables) helps sometimes. For instance, multiplying constraints by a constant < 1. This rescaling does not influence the numerical results of constrained ordination, but it can complicate further analyses when values of constraints are needed, because the same scaling must be applied there. We can only hope that this problem is fixed in the future versions of R and `LAPACK`.

2.12 Variance explained by ordination axes.

In general, **vegan** does not directly give any statistics on the "variance explained" by ordination axes or by the constrained axes. This is a design decision: I think this information is normally useless and often misleading. In community ordination, the goal typically is not to explain the variance, but to find the "gradients" or main trends in the data. The "total variation" often is meaningless, and all proportions of meaningless values also are meaningless. Often a better solution explains a smaller part of "total variation". For instance, in unstandardized principal components analysis most of the variance is generated by a small number of most abundant species, and they are easy to "explain" because data really are not very multivariate. If you standardize your data, all species are equally important. The first axes explains much less of the "total variation", but now they explain all species equally, and results typically are much more useful for the whole community. Correspondence analysis uses another measure of variation (which is not variance), and again it typically explains a "smaller proportion" than principal components but with a better result. Detrended correspondence analysis and nonmetric multidimensional scaling even do not try to "explain" the variation, but use other criteria. All methods are incommensurable, and it is impossible to compare methods using "explanation of variation".

If you still want to get "explanation of variation" (or a deranged editor requests that from you), it is possible to get this information for some methods:

- Eigenvector methods: Functions `rda`, `cca` and `capscale` give the variation of conditional (partialled), constrained (canonical) and residual components, but you must calculate the proportions by hand. Function `eigenvals` extracts the eigenvalues, and `summary(eigenvals(ord))` reports the proportions explained in the result object `ord`. Function `RsquareAdj` gives the R-squared and adjusted R-squared (if available) for constrained components. Function `goodness` gives the same statistics for individual species or sites (species are unavailable with `capscale`). In addition, there is a special function `varpart` for unbiased partitioning of variance between up to four separate components in redundancy analysis.
- Detrended correspondence analysis (function `decorana`). The total amount of variation is undefined in detrended correspondence analysis, and therefore proportions from total are unknown and undefined. DCA is not a method for decomposition of variation, and therefore these proportions would not make sense either.
- Nonmetric multidimensional scaling. NMDS is a method for nonlinear mapping, and the concept of of variation explained does not make sense. However, 1 - stress^2 transforms nonlinear stress into quantity analogous to squared correlation coefficient. Function `stressplot` displays the nonlinear fit and gives this statistic.

2.13 Can I have random effects in constrained ordination or in `adonis`?

No. Strictly speaking, this is impossible. However, you can define models that respond to similar goals as random effects models, although they strictly speaking use only fixed effects.

Constrained ordination functions `cca`, `rda` and `capscale` can have `Condition()` terms in their formula. The `Condition()` define partial terms that are fitted before other constraints and can be used to remove the effects of background variables, and their contribution to decomposing inertia (variance) is reported separately. These partial terms are often regarded as similar to random effects, but they are still fitted in the same way as other terms and strictly speaking they are fixed terms.

Function `adonis` evaluates terms sequentially. In a model with right-hand-side `~ A + B` the effects of `A` are evaluated first, and the effects of `B` after removing the effects of `A`. Sequential tests are also available in `anova` function for constrained ordination results by setting argument `by = "term"`. In this way, the first terms can serve in a similar role as random effects, although they are fitted in the same way as all other terms, and strictly speaking they are fixed terms.

All permutation tests in **vegan** are based on the **permute** package that allows constructing various restricted permutation schemes. For instance, you can set levels of `plots` or `blocks` for a factor regarded as a random term.

A major reason why real random effects models are impossible in most **vegan** functions is that their tests are based on the permutation of the data. The data are given, that is fixed, and therefore permutation tests are basically tests of fixed terms on fixed data. Random effect terms would require permutations of data with a random component instead of the given, fixed data, and such tests are not available in **vegan**.

2.14 Is it possible to have passive points in ordination?

Vegan does not have a concept of passive points, or a point that should only little influence the ordination results. However, you can add points to eigenvector methods using `predict` functions with `newdata`. You can first perform an ordination without some species or sites, and then you can find scores for all points using your complete data as `newdata`. The `predict` functions are available for basic eigenvector methods in **vegan** (`cca`, `rda`, `decorana`, for an up-to-date list, use command `methods("predict")`). You also can simulate the passive points in R by using low weights to row and columns (this is the method used in software with passive points). For instance, the following command makes row 3 "passive": `dune[3,] <- 0.001*dune[3,]`.

2.15 Class variables and dummies

You should define a class variable as an R `factor`, and **vegan** will automatically handle them with formula interface. You also can define constrained ordination without formula interface, but then you must code your class variables by hand.

R (and **vegan**) knows both unordered and ordered factors. Unordered factors are internally coded as dummy variables, but one redundant level is removed or aliased. With default contrasts, the removed level is the first one. Ordered factors are expressed as polynomial contrasts. Both of these contrasts explained in standard R documentation.

2.16 How are environmental arrows scaled?

The printed output of `envfit` gives the direction cosines which are the coordinates of unit length arrows. For plotting, these are scaled by their correlation (square roots of column `r2`). You can see the scaled lengths of `envfit` arrows using command `scores`.

The scaled environmental vectors from `envfit` and the arrows for continuous environmental variables in constrained ordination (`cca`, `rda`, `capscale`) are adjusted to fill the current graph.

The lengths of arrows do not have fixed meaning with respect to the points (species, sites), but they can only compared against each other, and therefore only their relative lengths are important.

If you want change the scaling of the arrows, you can use `text` (plotting arrows and text) or `points` (plotting only arrows) functions for constrained ordination. These functions have argument `arrow.mul` which sets the multiplier. The `plot` function for `envfit` also has the `arrow.mul` argument to set the arrow multiplier. If you save the invisible result of the constrained ordination `plot` command, you can see the value of the currently used `arrow.mul` which is saved as an attribute of `biplot` scores.

Function `ordiArrowMul` is used to find the scaling for the current plot. You can use this function to see how arrows would be scaled:

```
sol <- cca(varespec)
ef <- envfit(sol ~ ., varechem)
plot(sol)
ordiArrowMul(scores(ef, display="vectors"))
```

2.17 I want to use Helmert or sum contrasts

vegan uses standard R utilities for defining contrasts. The default in standard installations is to use treatment contrasts, but you can change the behaviour globally setting `options` or locally by using keyword `contrasts`. Please check the R help pages and user manuals for details.

2.18 What are aliased variables and how to see them?

Aliased variable has no information because it can be expressed with the help of other variables. Such variables are automatically removed in constrained ordination in **vegan**. The aliased variables can be redundant levels of factors or whole variables.

Vegan function `alias` gives the defining equations for aliased variables. If you only want to see the names of aliased variables or levels in solution `sol`, use `alias(sol, names.only=TRUE)`.

2.19 Plotting aliased variables

You can fit vectors or class centroids for aliased variables using `envfit` function. The `envfit` function uses weighted fitting, and the fitted vectors are identical to the vectors in correspondence analysis.

2.20 Restricted permutations in vegan

Vegan uses **permute** package in all its permutation tests. The **permute** package will allow restricted permutation designs for time series, line transects, spatial grids and blocking factors. The construction of restricted permutation schemes is explained in the manual page `permutations` in **vegan** and in the documentation of the **permute** package.

2.21 How to use different plotting symbols in ordination graphics?

The default ordination `plot` function is intended for fast plotting and it is not very configurable. To use different plotting symbols, you should first create and empty ordination plot with `plot(..., type="n")`, and then add `points` or `text` to the created empty frame (here `...` means other arguments you want to give to your `plot` command). The `points` and `text` commands are fully configurable, and allow different plotting symbols and characters.

2.22 How to avoid cluttered ordination graphs?

If there is a really high number of species or sites, the graphs often are congested and many labels are overwritten. It may be impossible to have complete readable graphics with some data sets. Below we give a brief overview of tricks you can use. Gavin Simpson's blog From the bottom of the heap has a series of articles on "decluttering ordination plots" with more detailed discussion and examples.

- Use only points, possibly with different types if you do not need to see the labels. You may need to first create an empty plot using `plot(..., type="n")`, if you are not satisfied with the default graph. (Here and below `...` means other arguments you want to give to your `plot` command.)
- Use points and add labels to desired points using interactive `identify` command if you do not need to see all labels.
- Add labels using function `ordilabel` which uses non-transparent background to the text. The labels still shadow each other, but the uppermost labels are readable. Argument `priority` will help in displaying the most interesting labels (see Decluttering blog).
- Use `orditorp` function that uses labels only if these can be added to a graph without overwriting other labels, and points otherwise, if you do not need to see all labels. You must first create an empty plot using `plot(..., type="n")`, and then add labels or points with `orditorp` (see Decluttering blog).
- Use `ordipointlabel` which uses points and text labels to the points, and tries to optimize the location of the text to minimize the overlap (see Decluttering blog).
- Ordination `text` and `points` functions have argument `select` that can be used for full control of selecting items plotted as text or points.
- Use interactive `orditkplot` function that lets you drag labels of points to better positions if you need to see all labels. Only one set of points can be used (see Decluttering blog).
- Most `plot` functions allow you to zoom to a part of the graph using `xlim` and `ylim` arguments to reduce clutter in congested areas.

2.23 Can I flip an axis in ordination diagram?

Use `xlim` or `ylim` with flipped limits. If you have model `mod <- cca(dune)` you can flip the first axis with `plot(mod, xlim = c(3, -2))`.

2.24 Can I zoom into an ordination plot?

You can use `xlim` and `ylim` arguments in `plot` or `ordiplot` to zoom into ordination diagrams. Normally you must set both `xlim` and `ylim` because ordination plots will keep the equal aspect ratio of axes, and they will fill the graph so that the longer axis will fit.

Dynamic zooming can be done with function `orditkplot`. You can directly save the edited `orditkplot` graph in various graphic formats, or you can export the graph object back to R and use `plot` to display the results.

3 Other analysis methods

3.1 Is there TWINSPAN?

No. It may be possible to port TWINSPAN to **vegan**, but it is not among the **vegan** top priorities. If anybody wants to try porting, I will be happy to help. TWINSPAN has a very permissive license, and it would be completely legal to port the function into R.

3.2 Why restricted permutation does not influence adonis results?

The permutation scheme influences the permutation distribution of the statistics and probably the significance levels, but does not influence the calculation of the statistics.

3.3 How is deviance calculated?

Some **vegan** functions, such as `radfit` use base R facility of `family` in maximum likelihood estimation. This allows use of several alternative error distributions, among them `"poisson"` and `"gaussian"`. The R `family` also defines the deviance. You can see the equations for deviance with commands like `poisson()$dev` or `gaussian()$dev`.

In general, deviance is 2 times log.likelihood shifted so that models with exact fit have zero deviance.