vegan FAQ

Frequently Asked Questions on R package vegan
Version of $Date: 2008-09-26 14:47:00 +0300 (Fri, 26 Sep 2008) $

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 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 distributed under a GNU-style copyleft, and an official part of the GNU 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 R-Forge.

1.4 What R packages vegan depends on?

Some vegan functions depend on packages `MASS`, `mgcv`, `cluster` and `lattice`. These all are recommended standard R packages that should be available in every R installation. In addition, some vegan functions `require` non-standard R packages. Vegan declares these packages only as suggested ones, and you can install vegan and use most of its functions without these packages. The non-standard packages needed by some vegan functions are:

- Package `ellipse` is needed by `ordiellipse`
- Package `scatterplot3d` is needed by `ordiplot3d`
- Package `rgl` is needed by `ordirgl` and `rgl.isomap`
- Package `tcltk` is needed by `orditkplot`

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 `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:

- `http://vegan.r-forge.r-project.org/`: vegan homepage.
- `http://cc.oulu.fi/~jarioksa/opetus/metodi/vegantutor.pdf`: vegan tutorial.
- http://wiki.r-project.org/: entry in R-Wiki.

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 Version numbering in vegan

From version 1.10-0, vegan is developed at R-Forge and there is a general progression of version numbers mixed with stable (at CRAN) and devel versions (at R-Forge).

Up to versions 1.8-7 and 1.9-34, vegan version numbers were of type x.y-z, where number y is even for stable release versions at CRAN and odd for unstable release versions at my personal homepage. Version 1.8-8 was a backport of bug fixes from the 1.10 series.

1.10 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, or want to use unstable devel versions, 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).

1.11 Are there binaries for devel versions?

R-Forge runs daily tests on the devel package, and if passed, it builds source package and Windows binaries. You can install those packages within R with command `install.packages("vegan", repos="http://r-forge.r-project.org/")`.

1.12 How to report a bug in vegan?

If you think you have found a bug in vegan, you should report it to me. The bug report should be so detailed that I can correct the bug. To correct a bug, I should be able to reproduce the buggy behaviour. Preferably, you should send me an example that causes a bug. If it needs a data set that is not available in R, you should send me minimal data set as well. You also should paste the output or error message in your message. You also should tell me 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.

There also is a bug reporting tool at R-Forge, but you need to register as a site user to report bugs (this is site policy).

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 problem with vegan, it may be as simple as incomplete documentation, and I'll do my 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 to me or in general.

Contributed code and functions are welcome and more certain to be included than mere requests. However, not all functions will be added, but I must judge them to be suitable for vegan. I also audit the code, and typically I edit the code in vegan style for easier maintenance. All included contributions will be credited. You can easily see that many vegan functions were contributed by other people, and they are listed as authors in the documentation.

2 Ordination

2.1 Can you 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.

2.2 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.

2.3 Zero dissimilarities in isoMDS

You can use argument `zerodist = "add"` in `metaMDS` or `metaMDSdist` to handle zero dissimilarities. With this argument, zero dissimilarities are replace with a small above zero 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).

2.4 Warnings of negative eigenvalues in capscale

Function `capscale` regularly and normally gives warnings of negative eigenvalues. These warnings are harmless, and `capscale` will ignore the axes with negative eigenvalues. The warnings are generated by underlying function `cmdscale` or metric multidimensional scaling (a.k.a. principal coordinates analysis), The metric MDS assumes that dissimilarities are metric, but most ecologically useful indices are semimetric. The warnings only concern the very last minor axes, and the axes with negative eigenvalues will be ignored in `capscale`. If the warnings are disturbing, you can use argument `add = TRUE` in `capscale` which implements "correction method 2" of Legendre & Legendre (1998, p. 434) in `cmdscale`.

You can get a warning about negative eigenvalues also with metric indices if you have deficit rank data. This happens, for instance, when number of species (columns) is lower than number of sites (rows), or if some sites are linear combinations of other sites. You can find the rank of the data using, for instance, vegan function `rda` which is identical to `capscale` with Euclidean distance.

2.5 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" 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. The `summary` gives the contributions of the axes. 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 unknown and undefined in detrended correspondence analysis, and therefore proportions from total also 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.

2.6 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,]`.

2.7 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.

You should never make your own dummy variables, but you should use standard R factors. R will internally change these factors into dummies in a consistent and correct way.

2.8 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.

2.9 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`, write `sol$CCA$alias`.

2.10 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.

2.11 Constrained permutations in vegan

You can constrain your permutations within `strata` or levels of factors. You can use stratified permutations in all `vegan` functions that use permutation, such as `adonis`, `anosim`, `anova.cca`, `mantel`, `mrpp`, `envfit` and `protest`.

Vegan has an alternative permutation function `permuted.index2` which allows restricted permutation designs for time series, line transects, spatial grids and blocking factors. Over time, the other functions that currently use the older `permuted.index` will be updated to use `permuted.index2`, but at the moment it is still experimental.

2.12 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.

2.13 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. However, here are some tricks you can use:

- 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 `identify` for ordination graphics, 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.

- 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`.
- Use `ordipointlabel` which uses points and text labels to the points, and tries to optimize the location of the text to minimize the overlap.
- 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.
- Most `plot` functions allow you to zoom to a part of the graph using `xlim` and `ylim` arguments to reduce clutter in congested areas.

3 Other analysis methods

3.1 Is there TWINSPAN?

No. It may be possible to port TWINSPAN to vegan, but it is not among my 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 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.