生态群落多元分析 R 程序包 vegan 教程

Jari Oksanen

2015.6.10

摘要

本教程演示了生态群落数据多元分析 R 语言程序包 vegan 的排序分析方法，学习本教程之前您应了解 R 语言基本语法和群落排序的一般概念。vegan 包提供包括非度量多维尺度分析（NMDS）在内的所有基本排序方法。教程也介绍了包括邻近约束分析（CAP），冗余分析（RDA）和典范对应分析（CCA）等约束排序方法。vegan 包还提供环境变量拟合以及绘制排序图的函数。

目录

1 简介

本教程演示了生态群落数据多元排序分析的操作流程。首先讨论了非约束排序分析以及基于其结果的环境解释（此时环境因子不参与排序过程）。然后以典范对应分析（CCA）为例介绍了约束排序，类似地，邻近约束分析（constrained analysis of proximities，CAP）和冗余分析（RDA）等方法也可以进行相同的排序分析。最后，本教程介绍了不基于排序的物种-环境关系的分析方法以及群落的聚类分析。

教程中的案例是经过检验的：这是一个 Sweave 文档（基于 LaTeX 的 R 代码组织的文档）。源文件只包含文本和 R 命令：结果和统计图表的输出需要通过运行 Sweave 源文件时生成。本文档中 R 是 2015-06-09 的版本(vegan 2.3-0 版本)，其实 vegan 和 R 也在不断发展中。

教程主要涵盖了 vegan 包的排序方法，没有介绍 vegan 包其他同样重要的函数，例如生物多样性分析函数：多样性指数(diversity()、renyi()、fisher.alpha())、物种丰富度(specpool(), estimate())、物种积累曲线(specaccum())、物种丰富度模型(radfit()、fisherfit()、prestonfit())等。vegan 也不是唯一能进行生态群落多元分析的 R 语言工具。R 语言基础包有标准的统计工具，labdsv 包提供一些高级方法或替代函数对 vegan 进行了补充，ade4 是利用欧几里得方法进行生态学数据分析的包。

本教程仅介绍了基本的方法，并展示了常用案例的一般流程。我们认为排序方法主要是作为一种图形工具，所以没有花太多精力展示精确的数值，取而代之的是在绘图命令旁边穿插绘制结果。建议您参考本教程尝试自己的分析，探索不同的替代方案，并花时间去理解命令以及函数的输出结果。教程仅对函数作了简要的说明，如果想获得更全面的解释，请查看相应的帮助文档。需要注意本文档也只是为了应用相应的函数而写的，想获得更坚实的理论背景，最好的方法是查阅有关生态学群落排序方法的教科书或者学习相关的课程。

2 基本的排序方法

2.1 非度量多维尺度分析

非度量多维尺度分析（Non-metric multidimensional scaling, NMDS）可以用 MASS 包中的 isoMDS()函数实现，输入相异矩阵即可。vegan 的 vegdist()函数可以计算群落的相异矩阵，默认是 Bray-Curtis 相异系数，现在通常称为 Steinhaus 相似指数，在芬兰称为 Sorensen 指数。基本步骤如下:

```
> library(vegan)
> library(MASS)
```

```
> data(varespec)
> vare.dis <- vegdist(varespec)
> vare.mds0 <- isoMDS(vare.dis)

initial  value 18.026495
iter   5 value 10.095483
final  value 10.020469
converged
```

NMDS排序结构通过迭代来不断最小化应力函数（stress function），默认情况下是找到两个维度并使用度量尺度分析(cmdscale)作为初始结构进行调整。从跟踪(trace)信息中可以看出迭代过程(通过设置参数trace = F来隐藏迭代过程)。

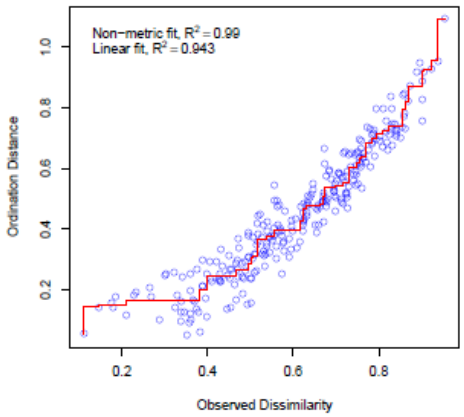
isoMDS()返回一个排序构建过程和应力函数的列表(item points, stress)。

S应力函数是一个拟合度统计量，是排序空间内对象结构与原始距离矩阵之间的相异程度的度量。

NMDS将观察到的群落差异非线性地映射到排序空间上，可以基于任何类型距离矩阵对对象进行排序。可以用MASS包的函数Shepard()或者vegan包的stressplot()函数来评估NMDS的结果（Shepard图）。

$$S = \sqrt{\frac{\sum_{i \neq j}[\theta(d_{ij}) - \tilde{d}_{ij}]^2}{\sum_{i \neq j} \tilde{d}_{ij}^2}}$$

```
> stressplot(vare.mds0, vare.dis)
```



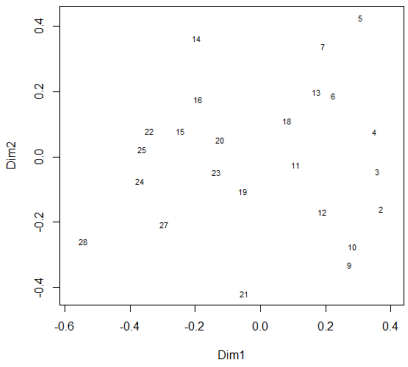
stressplot函数绘制了一个Shepard图，其中横坐标为原始距离，纵坐标为排序距离，用单调的折线拟合。此外，stressplot()显示了这两者距离相关性，如拟合度（goodness of fit）与应力函数的关系是$R^2 = 1 - S^2$。“fit-based R^2”是拟合值θ(d)和运算出的排序图上距离d之间的相关性，或者是折线和点之间的相关性。它应该是线性的，即使拟合有点弯曲，通常仍被称为“线性拟合”。这两个相关性都是基于Shepard图中的残差，但是它们的零模型有所不同。在线性拟合中，零模型是所有排序距离相等，拟合为一条水平直线。这听起来很合理，但是需要N-1维的N个点的零模型，而这个零模型在排序空间中是没有几何意义的。基本应力采用零模型，所有的观测都放在同一点上，这在几何上是可能的。注意，有时人们使用群落差异和排序距离之间的相关性。但是由于NMDS是一种非线性方法，因此这样做既危险又具有误导性：使用该准则，具有更多非线性关系的分

类将会出现更多错误。

vegan 的 ordiplot()函数可以用来绘制 NMDS 的结果：

> ordiplot(vare.mds0, type = "t")

相异矩阵没有物种的信息，因此只显示了样方信息（site）。



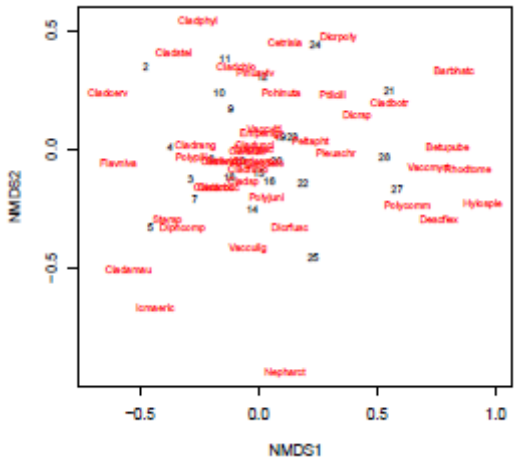
由于排序轴与初始相异结构之间存在非线性关系，使得 NMDS 在迭代时最优化应力函数非常困难：迭代容易陷入局部最优而不是全局最优。因此，建议进行多次独立的尝试，并在应力函数（stress）最小的类似解决方案中选择初始结构。这样做可能会很耗精力，好在 vegan 有 metaMDS()函数可以做到这一点。迭代过程输出很长，可以设置 trace = FALSE 来隐藏迭代过程显示，但通常我们希望有变化发生，尽管分析可能需要很长时间：

> vare.mds <- metaMDS(varespec, trace = FALSE)

> vare.mds

```
Call:
metaMDS(comm = varespec, trace = FALSE)

global Multidimensional Scaling using monoMDS

Data:     wisconsin(sqrt(varespec))
Distance: bray

Dimensions: 2
Stress:     0.1826
Stress type 1, weak ties
No convergent solutions - best solution after 20 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'wisconsin(sqrt(varespec))'
```

> plot(vare.mds, type = "t")



metaMDS()不需要单独计算相异矩阵，而是直接将原始数据矩阵作为输入。结果比以前更丰富，除了isoMDS()结果中的成分外还有很多其他成分：nobj, nfix, ndim, ndis, ngrp, diss, iidx, jidx, xinit, istart, isform, ities, iregn, iscal, maxits, sratmx, strmin, sfgrmn, dist, dhat, points, stress, grstress, iters, icause, call,model, distmethod, distcall, data, distance, converged, tries,engine, species。该函数将推荐的过程封装到一个命令中，所以到底发生了什么：

1. 一般生态群落数据比较离散，用平方根转换数据，然后进行Wisconsin双重标准化，或物种除以它们的最大值将数据均一化为相等的总数。这两个标准化通常可以提高排序的质量，但是我们通常在最初的分析中忘了考虑数据的转化问题。
2. 默认使用Bray-Curtis相异系数。
3. 运行多次独立的 isoMDS()，并在一定次数的尝试之后停止，或者在找到两个具有最小应力函数之后停止，返回了最佳的排序结果。
4. 旋转排序图，使样方坐标的最大方差位于第一轴上。
5. 对排序结果进行标准化，使一个单元对应于将群落相似性从重复相似性减半。
6. 函数发现物种排序轴为样方排序轴的加权平均值并将其扩大，使物种和样方排序轴具有相等的方差，可以使用 shrink = TRUE 撤消。

metaMDS()的帮助页面将提供更多细节，并解释函数使用过程。

2.2 群落相异性

非度量多维尺度分析（NMDS）是一种很好的排序方法，因为它可以使用具有生态学意义的方法来度量群落的差异。一个好的相异性测度与环境梯度的距离具有很好的秩关系。因为 NMDS 只使用秩信息，并且映射的秩在有序空间上是非线性的，故它能处理任意类型的非线性物种矩阵，并能有效、稳健地找到潜在梯度。

最自然的相异性测度是欧氏距离，欧氏距离是许多特征向量排序方法默认相异测度，它是物种空间中的距离。物种空间是指每个物种都是一个与所有其他物种正交的轴，而样方是这个多维超空间中的点。然而，欧氏距离是基于差值的平方，受单个最大方差影响较大。因此大多数有生态意义的相异系数是曼哈顿（Manhattan）类型的，使用的是差值而不是差值的平方。良好的相异指数的另一个特征是百分数系数：如果两个群落没有相同的物种，它们的相异指数最大，即为 1。另外，即使没有相同的物种，欧氏和曼哈顿相异指数也会随总体物种丰度的变化而变化。

vegan 包有 vegdist()函数可以计算各类相异系数，包括 Bray-Curtis、Jaccard 和 Kulczyński 指数。所有这些都属于曼哈顿类型，只使用一阶项（和以及差值），所有这些都按样方总数进行相对化，并在两个被比较群落之间没有相同物种时达到最大值 1。vegdist()函数是 R 基础 dist()函数的一个简易替代，这两个函数都可以用于相异系数的计算。

$$d_{jk} = \sqrt{\sum_{i=1}^{N}(x_{ij} - x_{ik})^2} \quad \text{Euclidean}$$

$$d_{jk} = \sum_{i=1}^{N}|x_{ij} - x_{ik}| \quad \text{Manhattan}$$

$$A = \sum_{i=1}^{N} x_{ij} \qquad B = \sum_{i=1}^{N} x_{ik}$$

$$J = \sum_{i=1}^{N} \min(x_{ij}, x_{ik})$$

$$d_{jk} = A + B - 2J \quad \text{Manhattan}$$

$$d_{jk} = \frac{A + B - 2J}{A + B} \quad \text{Bray}$$

$$d_{jk} = \frac{A + B - 2J}{A + B - J} \quad \text{Jaccard}$$

$$d_{jk} = 1 - \frac{1}{2}\left(\frac{J}{A} + \frac{J}{B}\right) \quad \text{Kulczyński}$$

众多的相异指数有许多令人困惑的方面：一是同样的指数名称却可能有完全不同的表达式：以右图曼哈顿相异系数的两种可选择的表达式为例。另一个复杂的问题是命名，vegdist()函数使用不够严格的口语化名字。vegan 的默认指数是 Bray（或 Bray-Curtis），但它可能应该被称为 Steinhaus 指数，几年前它的名字是 Czekanowski 指数（但现在这被认为是另一个指数），它也被称为 Sørensen 指数（但经常拼写错误）。严格来说，Jaccard 是二元（有-无）系数，但其定量版本在在 vegan 包称为 Ružička 指数。

这三个基本指标在梯度度量中被认为是较好的。此外，vegdist()函数还能够计算其他类型的距离矩阵：Morisita、Horn-Morisita、Raup-Cric、Binomial 和 Mountford 指数能够比较不同抽样单位的相异性，而 Euclidean、Canberra 和 Gower 指数具有更好的理论基础。

metaMDS()函数使用 Bray-Curtis 相异系数作为默认值，这通常是一个不错的选择。Jaccard （Ružička）指数具有完全相同秩的序，但具有更好的度量属性，可能应该作为首选。rankindex()函数在 vegan 中使用秩相关作为默认值，可用来研究哪个指数最好地沿着已知的梯度分离群落。下面的例子使用了 varechem 数据集中的所有环境变量，但都将其标准化为单位方差：是非线性相关的，但它们的秩的序是相同的，因此它们的秩相关也是相同的。一般来说，这三个推荐的指标是相当一致的。

```
> data(varechem)
```

> rankindex(scale(varechem), varespec, c("euc","man","bray","jac","kul"))

```
    euc     man    bray     jac     kul
 0.2396  0.2735  0.2838  0.2838  0.2840
```

许多教科书强调指标的度量性质，这些在某些方法中很重要，但在只使用信息的 NMDS 中就不重要了。这些度量属性简单来说是：

1. 如果两个样方相同，它们的距离为零；
2. 如果两个样方不同，它们的距离大于零；
3. 距离是对称的；
4. 两个样方之间最短的距离是直线，不受其他样方的影响。

$$\begin{aligned} &\text{for} \quad A = B \quad && d_{AB} = 0 \\ &\text{for} \quad A \neq B \quad && d_{AB} > 0 \\ & && d_{AB} = d_{BA} \\ & && d_{AB} \leq d_{Ax} + d_{xB} \end{aligned}$$

这些条件听起来都很自然，但并不是所有的相异系数都能满足。实际上，只有欧氏距离——可能还有 Jaccard 指数——满足这里讨论的相异系数的所有条件，并且是度量性质的。许多其他的相异系数满足前三个条件，并且是半度量性质的。

有一个学派说我们应该使用度量指标，并且首选的是欧氏距离。它们的缺点之一是没有固定的界限，但是没有相同物种的两个样方的相异系数可能会不同，甚至看起来比共有某些物种的两个样方更为相似。这可以通过标准化数据来解决。由于欧氏距离是基于平方差的，一个自然的变换是将样方标准化为相等的平方和，或者使用 decostand()函数将其标准化为向量范数：

> dis <- vegdist(decostand(varespec, "norm"), "euclid")

当两个样方之间没有相同物种时，其弦距离达到最大值 $\sqrt{2}$ 。另一个推荐的替代方法是 Hellinger 距离，它实际上是多度值先除以样方多度总和再取平方根后计算的欧式距离：

> dis <- vegdist(decostand(varespec, "hell"), "euclidean")

即使进行了标准化处理，欧式距离仍然具有良好的特性，除了用于转换数据之外。实际上即使使用其他指数，转换或标准化数据也常常很有用。如果在最小的非零丰度和最大丰度之间存在较大的差异，我们会想要减小这种差异，通常平方根变换足以平衡数据。Wisconsin 双重标准化往往提高了相异系数的梯度检测能力；这可以在 vegan 中使用函数 wisconsin()来实现。在这里，我们首先将所有物种数分别除以它们的最大值，然后将样方标准化到单位总量。在标准化之后，许多相异指数在排序上变得相同，在 NMDS 中应得到相同的结果。

Quadratic terms	
$J =$	$\sum_{i=1}^{N} x_{ij}x_{ik}$
$A =$	$\sum_{i=1}^{N} x_{ij}^2$
$B =$	$\sum_{i=1}^{N} x_{ik}^2$
Minimum terms	
$J =$	$\sum_{i=1}^{N} \min(x_{ij}, x_{ik})$
$A =$	$\sum_{i=1}^{N} x_{ij}$
$B =$	$\sum_{i=1}^{N} x_{ik}$
Binary terms	
$J =$	Shared species
$A =$	No. of species in j
$B =$	No. of species in k

		Site k	
		present	absent
Site j	present	a	b
	absent	c	d

$$J = a$$
$$A = a + b$$
$$B = a + c$$

在 vegan 包中，不仅可以使用 vegdist()函数，vegdist()与 R 的 dist()函数返回结构类似的相异矩阵，也可以使用标准 R 的 dist()函数以及其他包里的兼容函数，如 dsvdis()（labdsv 包）、daisy()（cluster 包）和 distance()（analogue 包），以及 vegan 包 betadiver()中的 β 多样性指数。此外，vegan 有 designdist()函数，支持用上面的 A、B 和 J 的符号来写它的方程，从而定义自己的相异系数，或者用二元数据，2*2 列联表表示法，其中 a 是在两个比较样方发现的物种数量，b 和 c 是只在其中一个样方发现的物种数量。以下三个方程定义了相同的 Sørensen 指数，用共有物种的数量除以比较样方的平均物种丰富度：

```
> d <- vegdist(varespec, "bray", binary = TRUE)

> d <- designdist(varespec, "(A+B-2*J)/(A+B)")

> d <- designdist(varespec, "(b+c)/(2*a+b+c)", abcd=TRUE)
```

Betadiver()函数在 vegan 中定义了更多的二元相异指数。

大多数已发表的相异指数可表示为 designdist 公式。但是，在现有函数中使用固定的替代方法会更容易也更安全：在手动编写相异指数方程时很容易出错。

2.3 比较排序图：普氏旋转（Procrustes rotation）

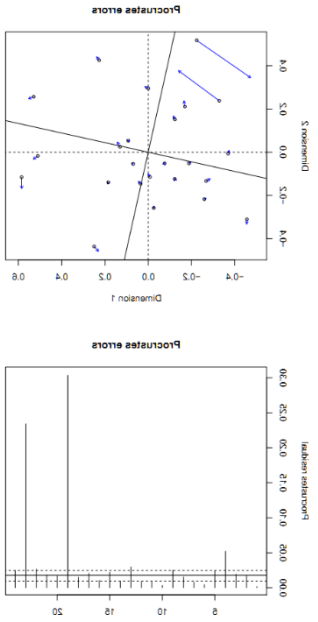
排序结果一般用排序图表示，比较两种排序方法构成的排序图是一个比较直观的方法。两个排序结果可能非常相似，但是由于轴的方向和比例略有不同，它们的差别很难观察到。实际上在 NMDS 中，虽然 metaMDS()使用简单的方法来确定最后三个组分，但是轴的标尺、排序、比例和位置都还不确定。比较排序的最好方法是使用普氏旋转(Procrustes rotation)。Procrustes rotation 使用均匀的比例

尺(膨胀或收缩)，通过旋转来尽量减少两个顺序之间的平方差。vegan 包有 procrustes()函数可以进行 Procrustes 分析。

metaMDS()比默认的 isoMDS()提供更多的信息。

```
> tmp <- wisconsin(sqrt(varespec))
> dis <- vegdist(tmp)
> vare.mds0 <- isoMDS(dis, trace = 0)
> pro <- procrustes(vare.mds, vare.mds0)
> pro

Call:
procrustes(X = vare.mds, Y = vare.mds0)

Procrustes sum of squares:
0.156
```

```
> plot(pro)
```



在这种情况下，差异相当小，主要表现在两个点上。我们可以使用 identify()函数通过人机交互式来识别的这些点，或者只用来做一张残差图：

```
> plot(pro, kind = 2)
```

描述性的统计是在 Procrustes 图中“Procrustes sum of squares”，或者“squared arrows 之和”。Procrustes rotation 是不对称的，统计量会随着排序轴旋转而变化。当参数 symmetric = TRUE 时，这两种解都首先被缩放为单位方差，并且会得到一个更独立、更对称的统计量(通常称为 Procrustes m^2)。

2.4 特征向量法

非度量多维尺度分析是一项艰巨的任务，因为任何一种差异度量都可以使用，而差异可以非线性地映射到排序空间。如果我们只接受特定类型的差异，并进行线性映射，排序就会成为一项简单的旋转和投影任务。在这种情况下，我们可以使用特征向量方法。主成分分析（PCA）和对应分析（CA）是群落排序中最重要的特征向量方法。此外，主坐标分析（又称为度量尺度分析，MDS）经常被使用。PCA 基于欧氏距离，CA 基于卡方距离，主坐标可以使用任意距离（如果用欧氏

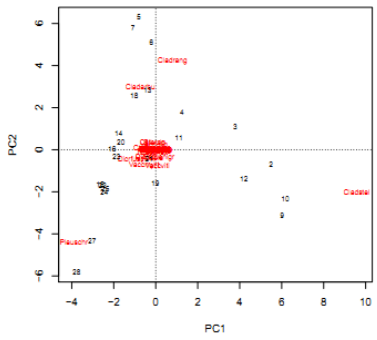
距离，则相当于 PCA）。

PCA 是一种标准的统计方法，可以用 R 的基础函数 prcomp()或 princomp()实现。CA 虽然不那么普及，但是也有几个函数可以运行。本教程展示如何使用 vegan 包里的函数 rda()和 cca()来做特征向量方法的排序分析，这两个函数名称实际上是为约束排序而设计的。

主成分分析可按以下方式进行：

```
> vare.pca <- rda(varespec)
> vare.pca
Call: rda(X = varespec)

              Inertia Rank
Total            1826
Unconstrained    1826   23
Inertia is variance

Eigenvalues for unconstrained axes:
PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8
983 464 132  74  48  37  26  20
(Showed only 8 of all 23 unconstrained eigenvalues)
```

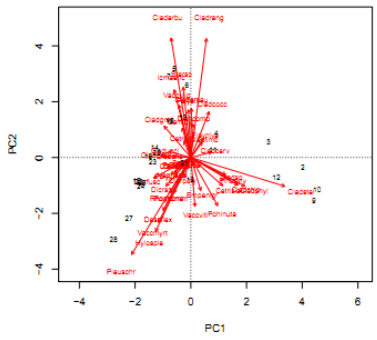
> plot(vare.pca)



输出结果表明总惯量（total inertia）为 1826，在 vegan 包中，惯量是数据“变差（variation）“的通用术语。23 个（秩）特征值之和等于总惯量。换句话说，这个解决方案将总方差分解成线性成分。我们可以很容易得出方差之和等于总惯量：

> sum(apply(varespec, 2, var))

```
[1] 1826
```



apply()将函数 var 或方差进行列(物种)求和，惯量是所有物种方差之和。换句话说，惯量各自“解释”了一定比例的总方差。第一轴“解释”总方差中的比例 983/1826=53.8%。

标准排序图函数对物种和样方使用点数或标签。有些人更喜欢对 PCA 中的物种使用双序图箭头，对样方也可以这样。有一个特定用于线性模型的 biplot()函数可以实现这一目的：

> biplot(vare.pca, scaling = -1)

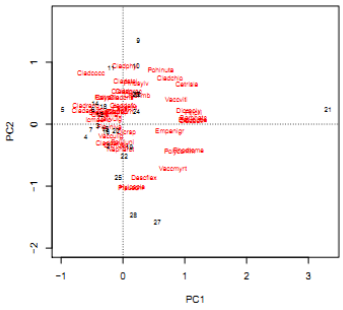
图中我们指定了 scaling=-1，结果只有在被访问时才被重新设定标尺，并且我们可以灵活地改变 plot()、biplot()以及其他函数的 scaling 参数。负值意味着物种坐标是物种得分除以物种标准差，因此优势种和稀有种将远离原点。

物种排序看上去有些不令人满意：只有 Cladina 和 Pleurozium Schreberi 很明显，其他物种都挤在原点处。之所以会这样是因为惯量是方差，也只有方差大的优势种才值得解释（但我们可以通过设置 scaling=-1 将其隐藏）。将所有物种标准化为单位方差，或使用相关系数而不是协方差，以得到更为平衡的排序：

> vare.pca <- rda(varespec, scale = TRUE)

> vare.pca

```
Call: rda(X = varespec, scale = TRUE)

              Inertia Rank
Total              44
Unconstrained      44   23
Inertia is correlations

Eigenvalues for unconstrained axes:
 PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8
8.90 4.76 4.26 3.73 2.96 2.88 2.73 2.18
(Showed only 8 of all 23 unconstrained eigenvalues)
```



现在惯量是变量自相关系数总和，一个变量与自身的相关性是 1，因此总惯量等于（物种）变量的数量。特征向量的排序或总数与以前相同。最大可能排序轴数由数据的维数定义：它比最少的物种数或地点数小 1。

> dim(varespec)

```
[1] 24 44
```

如果有相似的物种或地点，秩甚至会在这个基础上降低。

由第一轴解释的比例与前面的 PCA 相比有所下降。这是自然而然的，因为之前我们只需要“解释”有方差大的优势种，但现在我们还不得不平等地解释所有物种。我们不应该盲目地看百分比，而应该看我们得到的结果。

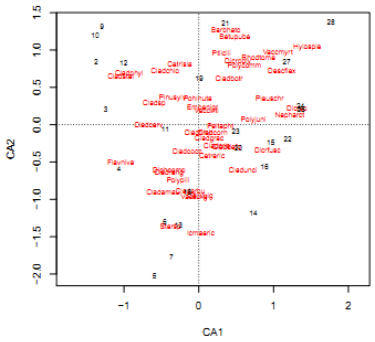
对应分析（CA）与 PCA 非常相似：

> vare.ca <- cca(varespec)

> vare.ca

```
Call: cca(X = varespec)

              Inertia Rank
Total            2.08
Unconstrained    2.08   23
Inertia is mean squared contingency coefficient

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
0.525 0.357 0.234 0.195 0.178 0.122 0.115 0.089
(Showed only 8 of all 23 unconstrained eigenvalues)
```

> plot(vare.ca)



这里惯量被称为均方列联系数。CA 是基于卡方距离的，惯性是一个标准化单位总数的数据矩阵的卡方统计量：

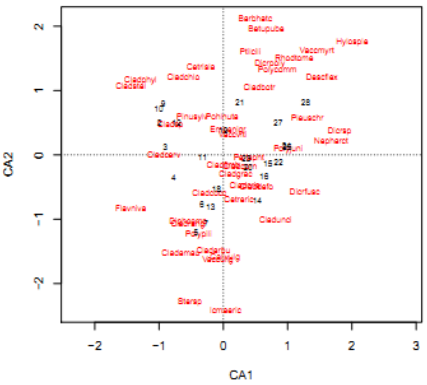
> chisq.test(varespec/sum(varespec))

```
        Pearson's Chi-squared test

data:  varespec/sum(varespec)
X-squared = 2.1, df = 990, p-value = 1
```

这里不应该关注p值，在这它存在误导性，上述结果中的卡方即为上面所讲的惯量。

对应分析是一种加权平均方法。在上图中物种坐标是样方坐标的加权平均值。对于不同标尺的结果，我们可以用物种坐标的加权平均值来表示样方坐标:

> plot(vare.ca, scaling = 1)



我们已经在PCA中看到scaling = 3或对称标尺的例子。另外两种方式表示物种是样方的加权平均值scaling = 2或样方是物种的加权平均值(scaling =1)。当我们取加权平均值时，平均值的范围从最初的值损耗。损耗系数与CA的特征值相等，最大值为1。

2.5 去趋势对应分析（DCA）

相对于主成分分析（PCA），CA是一种更好的、更为强健的群落分类方法。由于较长的生态梯度，该方法存在一些缺陷或错误。针对于此人们提出了去趋势

对应分析。

- 单一的长梯度以曲线或弧的形式出现在坐标中(弓形效应)：解决方法是使后一个轴的均值与前一个轴相等，从而去掉后一个轴的趋势。
- 在梯度极值处，样方的分布比在中心处更为密集：解决办法是重新调整坐标轴，使其与物种坐标的方差相等。
- 稀有物种可能对结果存在较大影响:解决方案是降低稀有物种的权重。

上述这些处理方法被包含在 decorana() 函数中，它是一个与Mark Hill源程序同名的准确可靠的函数。

> vare.dca <- decorana(varespec)

> vare.dca

```
Call:
decorana(veg = varespec)

Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                DCA1  DCA2   DCA3   DCA4
Eigenvalues     0.524 0.325 0.2001 0.1918
Decorana values 0.525 0.157 0.0967 0.0608
Axis lengths    2.816 2.205 1.5465 1.6486
```

> plot(vare.dca, display="sites")

decorana()函数仅有四个轴（DCA1- DCA4），与上面的CCA类似，特征值被作为加权平均值的损耗系数。“Decorana values”是源程序的返回值被称为作为“eigenvalues”——我不知道它们可能的含义，不推荐这样使用它们。大多数情况下，人们会对轴长进行注释，一些时候被称为“梯度长度”，但是这个词源很模糊：这些并不是梯度，而是坐标。通常当轴长小于2个单位时，数据被认为是线性的，这时应该使用PCA。这只是研究者经验之谈，并没有研究验证其可靠性，通常与较短梯度的PCA相比，CA分析更好。

当前的数据集是不是很离散（生态梯度较短），DCA的影响不是很大。但在具有明显弓形效应的异质性数据中，变化通常很剧烈。在许多情况下重新确定标尺可能比去趋势有更大的影响。

默认的分析中并没有降低稀有物种的权重，有关所需参数，请参阅帮助页面。实际上降低权重是一个独立的函数，可以与CCA一起使用。

有一种学派认为DCA是非约束排序很好的替代方法，然而这是一个不恰当的描述，含有歧义，应该避免这样理解。

2.6 排序图

在本教程中我们已经看到了许多排序图，它们有一个共同的特性：杂乱不堪，标签难以阅读。

由于我们必须把大量的标签放在一个小的图中，因此排序图很难画清楚，而且通常不可能展示所有的标签。在本章中，我们将讨论如何生成更清晰的排序图。为此，我们必须深入了解vegan包中的绘图函数，并需要学会如何更好地控制默认参数。

vegan包中的排序函数有其独立的绘图部分，它提供一些简单的绘图类型。如decorana()的结果由plot.decorana()函数显示，它在后台调用plot()函数。我们也可以使用ordiplot()函数，它也可以与许多非vegan排序函数一起工作，但是默认使用散点而不是文本标签。plot.decorana()(或ordiplot())函数将绘图分为三个步骤:

1.绘制一个带有坐标轴的空白区域，但没有表示样方或物种的符号。

2.使用函数、文本或点向空白区域添加物种信息。默认情况下，该函数在小数据集中将使用文本标签，在大数据集中使用散点。

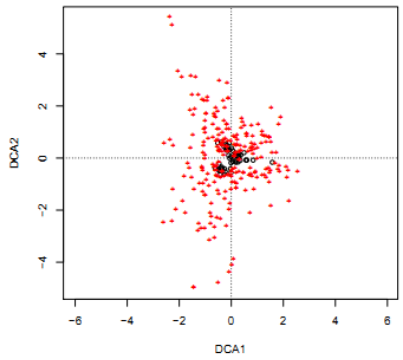
3.以类似的方式添加样方信息。

为了更好的控制这些绘图区域，我们必须不断重复上述步骤：绘制一个空白区域，然后根据需要添加样方/物种信息。在这一章中，我们研究了一个较难绘图的案例：绘制BCI（Barro Colorado Island）数据集的排序图。

```
> data(BCI)
```

这是一个难绘图的数据集：它有225个物种，没有办法把它们都清楚地标记出来——除非我们使用非常大的绘图区域和字体尺寸很小的文本。我们只能展示一个物种或一小部分的散点。首先使用decorana()函数初步绘制默认的排序图：

```
> mod <- decorana(BCI)

> plot(mod)
```



使用上述数据绘图时，存在变量名较长的问题：

> names(BCI)[1:5]

```
[1] "Abarema.macradenia"    "Acacia.melanoceras"
[3] "Acalypha.diversifolia" "Acalypha.macrostachya"
[5] "Adelia.triloba"
```

数据集使用完整的物种名称，无法在排序图中插入这些名称。

因此需要使用vegan包中make.cepnames()函数去缩写拉丁名：

> shnam <- make.cepnames(names(BCI))

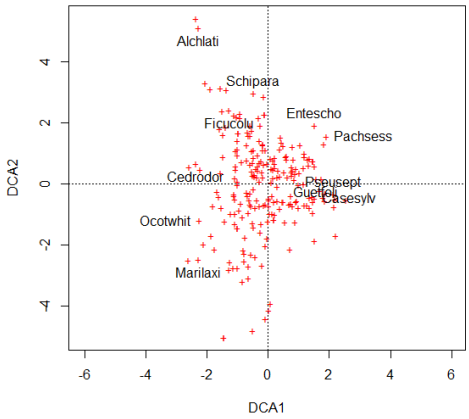
> shnam[1:5]

```
[1] "Abarmacr" "Acacmela" "Acaldive" "Acalmacr" "Adeltril"
```

选择标记物种最简单的方式是使用人机交互identify()函数：当单击某个点附近，它的标签将出现在单击的一侧，单击鼠标右键完成这个物种的标记，当你的鼠标存在问题时，可以单击键盘esc键完成操作。

> pl <- plot(mod, dis="sp")

所有vegan排序图都将返回到一个隐藏的ordiplot对象，该对象包含关于所绘制点的全部信息。可以展示这个隐藏的结果，并将其用作识别的输入。以下是一些极端物种的选择标签：



> identify(pl, "sp", labels=shnam)

vegan包中有一个“排序文本或点”的orditorp()函数。只有在不覆盖之前的标签的情况下，这个函数才会标记某个点。如果某个点不能用文本标记，它将会被标记为一个点。项目要么从边缘到中心的顺序处理，要么按优先级递减顺序处理。以下是按物种的丰富度进行的优先排序:

> stems <- colSums(BCI)

> plot(mod, dis="sp", type="n")

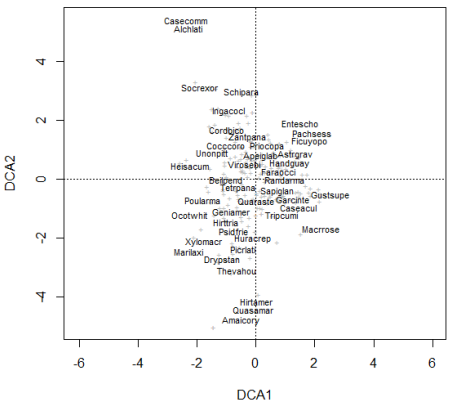
> sel <- orditorp(mod, dis="sp", lab=shnam, priority=stems, pcol = "gray", pch="+")

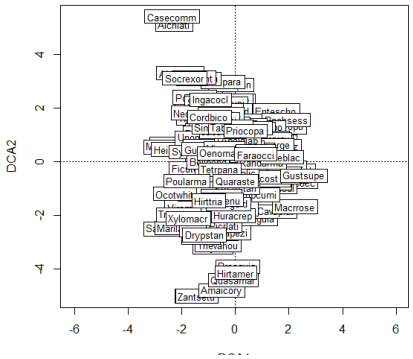
我们还可以通过设置xlim和ylim来放大坐标图的某些部分，从而可以看到更多的细节。

orditorp() 的另一种替代方法是使用 ordilabel()函数，它在不透明的标签上绘制文本，覆盖在下面的其他标签之上，尽管其不能显示所有标签，但最上面的标签是可读的。参数优先级的工作原理与orditorp()类似，可以选择最重要的标签进行展示:

> plot(mod, dis="sp", type="n")

> ordilabel(mod, dis="sp", lab=shnam, priority = stems)





最后，还有利用点和标签的函数 ordipointlabel()。这些点位于固定的位置，但是标签是迭代定位的，以最小化它们的重叠。对于 ordipointlabel()函数，BCI 数据集物种名称依然太多了，但是在很多情况下它都很有用。

除了这些自动功能之外，函数 orditkplot()还允许编辑绘图。它有固定位置的点和标签，可以用鼠标拖动到更合适的地方。该函数使用不同于普通 R 图形的工具集(Tcl/Tk)，但结果可以传递给标准 R 绘图函数进行编辑或直接保存为图形文件。此外，可以使用 orditkplot()编辑 ordipointlabel()控制输出。

函数 identify()、orditorp()、ordilabe()l 和 ordipointlabe()l 可以提供一种快速简便的方法来检查排序结果。通常我们需要更好地控制图形，并选择标记的物种。在这种情况下，我们可以首先绘制一个空的图(type = "n")，然后在排序 text 和 points 函数中使用 select 参数。select 参数可以是一个能列出所选项索引的数字向量。这些索引显示在标识函数中，标识函数可用于帮助选择点。或者 select 可以是一个逻辑向量，它所选的是为 TRUE 的对象。这样一份清单是由 orditorp()函数后台控制的。你不能直接从方法中看到不可见的结果，但是你可以像我们在第一个 orditorp()调用中所做的那样捕捉结果，并使用这个向量作为完全控制图形的基础。在这个例子中，第一项是：

> sel[1:14]

```
Abarmacr Acacmela Acaldive Acalmacr Adeltril Aegipana Alchcost
   FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
Alchlati Alibedul Allopsil Alseblac Amaicory Anacexce Andiiner
    TRUE    FALSE    FALSE    FALSE     TRUE     TRUE    FALSE
```

3 使用环境因子被动解释排序轴

通常结合研究样方的生态知识或关于物种生态特征来解释排序结果，但一般

最好使用外部环境变量来解释。有许多方法可以将环境信息叠加到排序图上。最简单的方法之一是根据环境变量(plot()函数中的参数 cex)更改绘图字符的大小。vegan 包有一些有用的函数可以拟合环境变量。

3.1 矢量拟合

最常用的解释方法是将环境向量拟合到排序图上，拟合的向量是带有解释的箭头：

- 箭头方向环境变量变化最快的方向，通常称为梯度方向。
- 箭头的长度表示环境因子与排序轴的相关性，通常称为梯度强度。

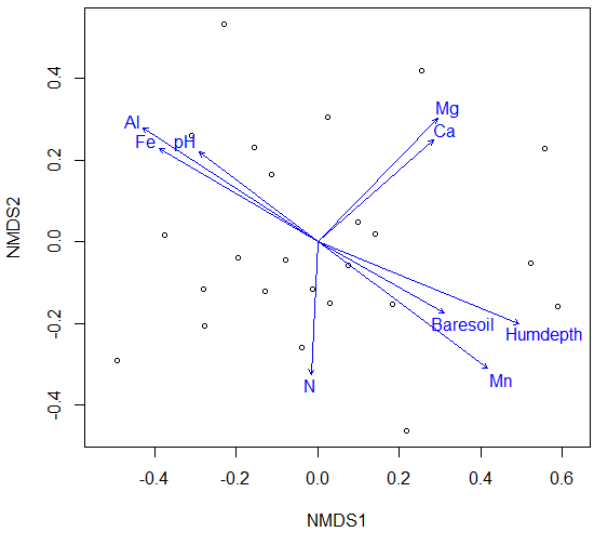
利用函数 envfit()对环境向量进行拟合比较简单，该示例使用了前面的 NMDS 排序结果和 varechem 数据集中的环境变量：

```
> data(varechem)
> ef <- envfit(vare.mds, varechem, permu = 999)
> ef
```

```
***VECTORS

           NMDS1  NMDS2   r2 Pr(>r)
N         -0.057 -0.998 0.25  0.046 *
P          0.620  0.785 0.19  0.103
K          0.767  0.642 0.18  0.132
Ca         0.685  0.728 0.41  0.006 **
Mg         0.633  0.775 0.43  0.002 **
S          0.191  0.982 0.18  0.118
Al        -0.872  0.490 0.53  0.002 **
Fe        -0.936  0.352 0.44  0.004 **
Mn         0.799 -0.602 0.52  0.001 ***
Zn         0.617  0.787 0.19  0.098 .
Mo        -0.903  0.430 0.06  0.519
Baresoil   0.925 -0.380 0.25  0.049 *
Humdepth   0.933 -0.360 0.52  0.002 **
pH        -0.648  0.762 0.23  0.072 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Permutation: free
Number of permutations: 999
```

前两列给出环境因子向量的方向余弦值，r^2 给出了相关系数。对于绘图，坐标轴应该缩放（scaled）r^2 的平方根。plot 函数自动执行此操作，可以使用坐标(ef，“vector”)提取缩放值。显著性(Pr>r)或 P 值基于数据的随机置换：如果得到与随机置换的数据一样 R^2，说明置换检验的结果是不显著的。

使用 plot()函数将拟合向量添加到排序图中，使用参数 p.max 将绘图限制为最重要的变量。与往常一样，在帮助页面中能找到更多可以调整的参数。




```
> plot(vare.mds, display = "sites")
> plot(ef, p.max = 0.1)
```

3.2 曲面拟合

向量拟合是一种普遍的方法，它提供了一种能同时显示大量环境变量的简洁方法。然而它默认排序轴和环境因子之间是线性关系：只需要知道方向和强度。考虑到生态学的研究场景，这一假定可能并不总是合适的。

函数 ordisurf()将环境变量的曲面与排序轴匹配。在 mgcv 包的 gam()函数中采用广义可加模型。函数 gam()在二维排序空间中采用薄板样条插值算法（thinplate splines），通过广义交叉验证自动选择平滑程度。如果响应确实是线性的且向量合适，则拟合曲面是一个梯度平行于箭头的平面，并且拟合轮廓是与箭头垂直的等距平行线。

在下面的示例介绍了两个新的 R 函数：

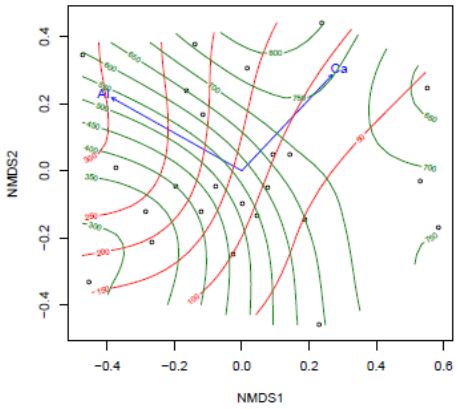
函数 envfit()可以通过公式接口调用。公式有一个特殊的波形字符(~)，左边为排序结果，右边为参与拟合的环境因子。此外，必须定义包含拟合变量的数据的名称。

数据框中的变量对 R 会话不可见，除非将数据框载入(attached)会话界面。我们可能不想让所有的变量都对会话可见，因为可能存在同义名称，而且在某些分析中可能使用了同名的错误变量。可以使用 with()函数使给定的数据框仅对指定的命令可见。

现在举例说明。我们对选定的变量进行向量拟合，并在同一个图中添加拟合曲面。

```
> ef <- envfit(vare.mds ~ Al + Ca, varechem)
```

```
> plot(vare.mds, display = "sites")
> plot(ef)
> tmp <- with(varechem, ordisurf(vare.mds, Al, add = TRUE))
> with(varechem, ordisurf(vare.mds, Ca, add = TRUE, col = "green4"))
```



函数 ordisurf()返回拟合的 gam 的结果。如果我们保存这个结果，就像我们在第一次使用铝(Al)做拟合时那样，我们可以将它用于进一步的分析，比如统计检验和新值的预测。举个例子，fitted (ef)（本例中应为 fitted (tmp)）给出站点的实际拟合值。

3.3 因子变量

前面讨论的都是连续型环境因子变量，因子型环境变量亦可以加到排序空间中。在排序图中因子分类数据质心（centroids）是因子变量自然选择的结果，并且 R^2 可用作衡量拟合优度的统计量。“显著性(significance)”可以通过置换检验实现，就像在向量拟合中一样。变量可以定义为 R 中的因子(factors)，它们将被相应地处理，不需要任何特殊技巧。

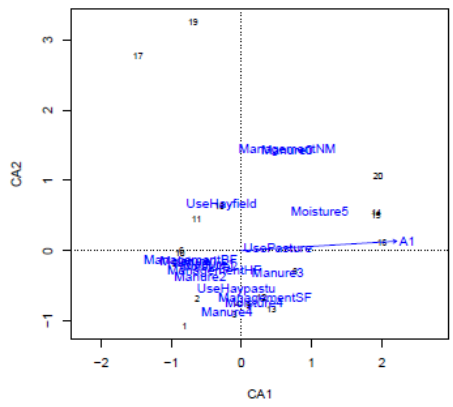
举一个例子，我们将检查沙丘草甸数据（dune meadow），它有几个类变量。函数 envfit()也适用于因子：

```
> data(dune)
> data(dune.env)
> dune.ca <- cca(dune)
> ef <- envfit(dune.ca, dune.env, permutations = 999)
> ef
```

```
> plot(dune.ca, display = "sites")
```

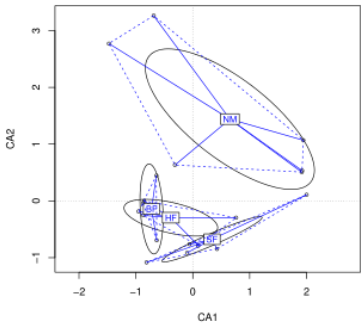
> plot(ef)

```
***VECTORS

      CA1    CA2   r2 Pr(>r)
A1 0.9980 0.0606 0.31   0.04 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Permutation: free
Number of permutations: 999

***FACTORS:

Centroids:
               CA1   CA2
Moisture1    -0.75 -0.14
Moisture2    -0.47 -0.22
Moisture4     0.18 -0.73
Moisture5     1.11  0.57
ManagementBF -0.73 -0.14
ManagementHF -0.39 -0.30
ManagementNM  0.65  1.44
ManagementSF  0.34 -0.68
UseHayfield  -0.29  0.65
UseHaypastu  -0.07 -0.56
UsePasture    0.52  0.05
Manure0       0.65  1.44
Manure1      -0.46 -0.17
Manure2      -0.59 -0.36
Manure3       0.52 -0.32
Manure4      -0.21 -0.88

Goodness of fit:
             r2 Pr(>r)
Moisture   0.41  0.008 **
Management 0.44  0.001 ***
Use        0.18  0.078 .
Manure     0.46  0.006 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Permutation: free
Number of permutations: 999
```



因子质心的名称是由因子的名称和因子水平的名称组合而成的。现在排序轴显示了因子水平的质心，R^2 值代表因子拟合优度。默然参数绘制图形并不美观，可以使用§2.6 中的技巧使图更清楚，但显然并非所有因素在解释时都是必要的。

Vegan 包有几个显示图形因素的函数。函数 ordihull()为分类中的对象绘制椭圆（enclosing convex hull），ordisider()将对象组合到它们的(加权的)分类质心，ordiellipse()为类标准视图、标准错误或置信区绘制椭圆。这个例子显示了前一个排序中的类型，并在 ordisider()中自动标记分组:



> plot(dune.ca, display = "sites", type = "p")

> with(dune.env, ordiellipse(dune.ca, Management, kind = "se", conf = 0.95))

> with(dune.env, ordispider(dune.ca, Management, col = "blue", label= TRUE))

> with(dune.env, ordihull(dune.ca, Management, col="blue", lty=2))

对应分析(CA)是一种加权排序方法，vegan 种的函数 envfit()和 ordisurf()将进行加权拟合，除非用户指定了相等的权重。

4 约束排序

在非约束排序中，我们首先找到主成分，然后把这种变化与所观察到的环境变化联系起来。在约束排序中，我们不希望显示所有甚至大部分的成分变化，而只显示由环境变量或解释变量引起的变化。约束排序通常称为“典范”排序，但这个名称具有误导性：这个方法中没有什么特别典范的。之所以使用这个名称，是因为有一种特殊的统计方法，典范相关(canonical correlations)：它们是两个矩阵之间的关联，被认为是彼此依赖的对称分析。而约束排序是非对称的：分析中有“自变量”或约束变量和“因变量”或群落数据。约束排序与多元线性模型有关。

Vegan 包有三种约束排序方法，它们都是基本排序方法的约束版本：

- 函数 capscale()中的邻近约束分析(Constrained analysis of proximities，CAP)与度量尺度(cmdscale)对应，它可以处理任何不同的度量，并执行线性映射。
- 函数 rda()中的冗余分析(RDA)与主成分分析对应，它基于欧氏距离的线性映射。
- 函数 cca()中的典范对应分析(CCA)与对应分析对应，它基于卡方距离，并进行加权线性映射。

我们已经使用了 rda()和 cca()函数进行非约束排序：如果不使用约束排序，它们将作为特殊情况执行基本的非约束排序。

这三种 vegan 函数非常相似。下面的例子主要使用 cca()，其他方法用法类似。实际上它们的输出结构类似，并且都继承了彼此的属性。由于历史原因，cca()是基本方法，rda()继承了它的特性。函数 capscale()直接继承了 rda()，并通过这个继承 cca()。许多函数与所有这些方法是共同的，并且只在方法偏离其先前的方法时才有特定的功能。在 vegan 2.3-0 版本中，为这些方法定义了以下类函数:

- cca: add1, alias, anova, as.mlm, biplot, bstick, calibrate, coef,deviance, drop1, eigenvals, extractAIC, fitted, goodness, model.frame,model.matrix, nobs, permutest, plot, points, predict, print, residuals, RsquareAdj, 坐标, screeplot, simulate, stressplot, summary, text, tolerance, weights
- rda: as.mlm, biplot, coef, deviance, fitted, predict, RsquareAdj,坐标, simulate, stressplot, weights
- capscale: fitted, print, RsquareAdj, simulate, stressplot.

其中许多方法一般情况下很少用到。

4.1 模型说明

定义约束模型的推荐方法是使用模型公式表达式。公式有一个特殊的字符~，左边给出群落数据矩阵的名称，右边给出约束变量的方程式，此外还需给出在何处查找约束变量的数据集。如 cca()对数据集 varespec 受 Al、K 和 P 的约束:

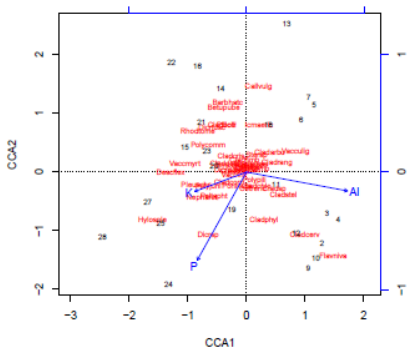
```
> vare.cca <- cca(varespec ~ Al + P + K, varechem)
> vare.cca
Call: cca(formula = varespec ~ Al + P + K, data =
varechem)

              Inertia Proportion Rank
Total           2.083      1.000
Constrained     0.644      0.309    3
Unconstrained   1.439      0.691   20
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1  CCA2  CCA3
0.362 0.170 0.113

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8
0.350 0.220 0.185 0.155 0.135 0.100 0.077 0.054
(Showed only 8 of all 20 unconstrained eigenvalues)
```

cca()输出类似于非约束排序，将总惯量分解为约束分量和非约束分量。有三

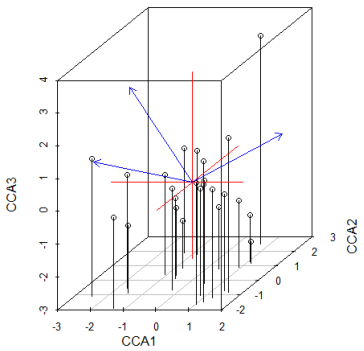
个约束条件，约束分量的秩为 3。非约束分量的秩为 20，在之前的分析中（CA）为 23。秩与排序轴的数目相同：有 3 个约束排序轴和 20 个非约束排序轴。在一些案例中，约束变量的秩可能小于其数量：一些约束变量之间是相互依赖的（共线性较强），分析当中它们还存在其他的别名，并且其有效的信息将会和结果一起输出。从总的惯量（列联系数均方）当中计算部分约束变量的惯量是非常普遍的。然而，总惯量在 CCA 分析当中没有明确的意义，同时这部分的惯量也无法明确得到。在 RDA 分析过程中，约束变量的惯量有可能是方差或相关性的比例。在 RDA 中具有明确的意义，但是即便这样，大部分的总惯量也有可能是随机噪声引起的。因此，更应该关注结果而不是这些惯量的比例。

基本的绘图工作和之前一样：

> plot(vare.cca)。



图中可以看出与环境因子被动拟合的向量有着相似的形式：箭头的指向是环境因子变化的梯度方向，它的投影长度代表着变量在排序轴上的解释度。矢量将在满秩解中具有单位长度，但它们将会被投影到图中使用的平面上。



vegan3d包提供了一个原始的3D绘制功能ordiplot3d()来显示所有箭头的总长度：

> library(vegan3d)

> ordiplot3d(vare.cca, type = "h")

使用函数ordirgl()，可以使用鼠标通过旋转和放大来检查3D动态图。

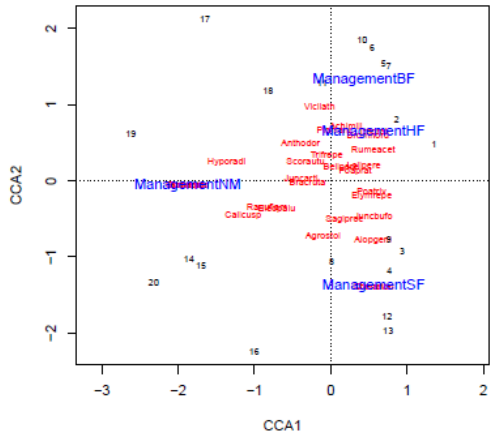
这个公式的的接口操作同样也适用于因子变量：

> dune.cca <- cca(dune ~ Management, dune.env)

> plot (dune.cca)

> dune.cca

```
Call: cca(formula = dune ~ Management, data = dune.env)

              Inertia Proportion Rank
Total           2.115      1.000
Constrained     0.604      0.285    3
Unconstrained   1.511      0.715   16
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1  CCA2  CCA3
0.319 0.182 0.103

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8   CA9  CA10
0.447 0.203 0.163 0.135 0.129 0.095 0.079 0.065 0.050 0.043
 CA11  CA12  CA13  CA14  CA15  CA16
0.039 0.024 0.018 0.009 0.008 0.004
```



因子变量主要有四个水平(BF, HF, NM, SF)。R的内部将这四个因子水平当作三个约束变量（有时称为虚拟变量。约束条件的应用如下：

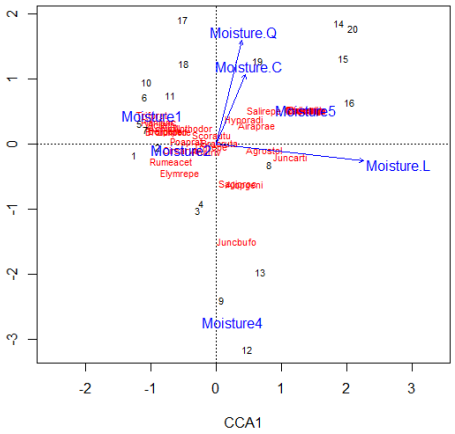
	ManagementHF	ManagementNM	ManagementSF
SF	0	0	1
BF	0	0	0
HF	1	0	0
NM	0	1	0

我们不需要三个变量来表达四个因子水平：如果有个数字1在一个列当中，那么观察对象就属于该水平。如果一整列都是0的话，观察对象属于被忽略的水平。基本的画图功能展现的是分组的质心而不是因子向量。

除了这些普通的因子外，R同样也能识别有序的因子。dune.env中的湿度变量就被定义为一个有序的四个水平的因子变量。在这个例子中约束条件就有些不一样。

	Moisture.L	Moisture.Q	Moisture.C
1	-0.6708	0.5	-0.2236
2	-0.2236	-0.5	0.6708
4	0.2236	-0.5	-0.6708
5	0.6708	0.5	0.2236

R使用多项式约束：线性项相当于把湿度当作连续性变量，二次项Q和立方项C表示的是非线性的特征，有四个明显的因子水平，并且约束变量的数目要比因子水平少一个。因子无论是有序的还是无序的，排列的构造、特征值和秩都不会发生改变，但是结果中呈现的因子可能不同。




```
> vare.cca <- cca(dune ~ Moisture, dune.env)
> plot(vare.cca)
```

在排序图中显示了因子水平的质心，也显示了约束变量的中心。如果我们能够将有序因子改变成连续性的向量，那么只有线性效果的箭头才是最重要的。如果响应的变量是非线性的，则二次项（和立方项）的箭头将会很长。

本例只介绍了公式模式的最简单的用法，可以通过这个公式转变约束变量，还可以定义交互作用以及使用多项式约束等等。然而，模型的交互以及多项式可能很难得到解释。

4.2 置换检验

所用因子的重要性可以通过置换检验来评估：通过多次随机调换被检验元素的的位置计算统计值，构造经验分布再进行假设检验。当置换中约束的惯量比观察到的约束惯量低时，则认为约束变量是显著的。

最简单运行置换检验的方法就是使用vegan包中的anova()函数（注意，R语言中方差分析的函数也是anova()，二者名称一样但用法完全不同。关于此函数的详情，可使用?anova.cca()查看帮助而不是?anova()，不然弹出的将会是方差分析的帮助文档））：

```
> anova(vare.cca)
```

```
Permutation test for cca under reduced model
Permutation: free
Number of permutations: 999
Model: cca(formula = dune ~ Moisture, data = dune.env)
          Df ChiSquare      F Pr(>F)
Model      3   0.62831 2.2536  0.003 **
Residual 16   1.48695
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

输出结果中 Model 部分表明约束排序因子，残差(Residual)表明排序的非约束成分，卡方(ChiSquare)是相应的惯量，而自由度(Df)是相应的秩。检验统计量 F 或者时更加正确的“pseudo-F"被定义为它们的比值。不应该关注它的数值或者是自由度的数量，因为”pseudo-F"与真实的 F 值没有任何关系，唯一评估它的方法是置换检验的结果。在这样简单的模型中，我们在检验中可以直接使用惯性，但是“pseudo-F"和“partialled"只有在更加复杂的模型中才有意义。

$$F = \frac{0.628/3}{1.487/16} = 2.254$$

在anova()函数中对置换的次数没有规定，这个函数是贪婪的：当函数不确定时会持续地进行置换而不管最终p值的大小是否低于或者超过标准值（通常是0.05）。

如果在置换中从未达到观察的惯量，则函数将会在200次置换之后停止，如果经常超出范围，会在置换100次后停止。接近邻近水平时，可能需要上千次置换。当情况明确时，用这种方法能够很快地进行计算，但是在不确定的情况下计算的时间会持续很长。如果要进行固定数量的迭代，则必须在anova()调用中指定或直接使用基础函数permutest.cca。

除了全模型一起的约束变量外，我们也可以通过设置参数来分析单个变量或者排序轴。以下的命令在连续(“Type I”) 中分别分析所有的变量:

```
> mod <- cca(varespec ~ Al + P + K, varechem)
> anova(mod, by = "term", step=200)
Permutation test for cca under reduced model
Terms added sequentially (first to last)
Permutation: free
Number of permutations: 999

Model: cca(formula = varespec ~ Al + P + K, data = varechem)
         Df ChiSquare    F Pr(>F)
Al        1     0.298 4.14  0.001 ***
P         1     0.190 2.64  0.007 **
K         1     0.156 2.17  0.022 *
Residual 20     1.439
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

所有的变量均与残差对比不受置换次数的影响，由于检验是按顺序进行的，所以变量的顺序会影响结果，除非变量之间是不相关的。在这个例子当中，所有变量的置换数目一样。

每一个因子的检验统计量（卡方）之和与整体检验中的模型检验统计量相同。“type III"测试分析从包含所有其他项的模型中删除每项的边际效应：

```
> anova(mod, by = "margin", perm=500)
```

```
Permutation test for cca under reduced model
Marginal effects of terms
Permutation: free
Number of permutations: 999

Model: cca(formula = varespec ~ Al + P + K, data = varechem)
         Df ChiSquare    F Pr(>F)
Al        1     0.312 4.33  0.001 ***
P         1     0.168 2.34  0.018 *
K         1     0.156 2.17  0.029 *
Residual 20     1.439
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

边际效应与因子的顺序无关，但相关因子的 P 值会更高(“更坏”)。在全模型检验中，测试统计量的总和并不等于模型测试统计量，除非这些项不相关。

我们还可以要求对各个轴进行检验：

> anova(mod, by="axis", perm=1000)

```
Permutation test for cca under reduced model
Marginal tests for axes
Permutation: free
Number of permutations: 999

Model: cca(formula = varespec ~ Al + P + K, data = varechem)
          Df ChiSquare    F Pr(>F)
CCA1       1     0.362 5.02  0.001 ***
CCA2       1     0.170 2.36  0.008 **
CCA3       1     0.113 1.57  0.131
Residual 20     1.439
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

4.3 模型建立

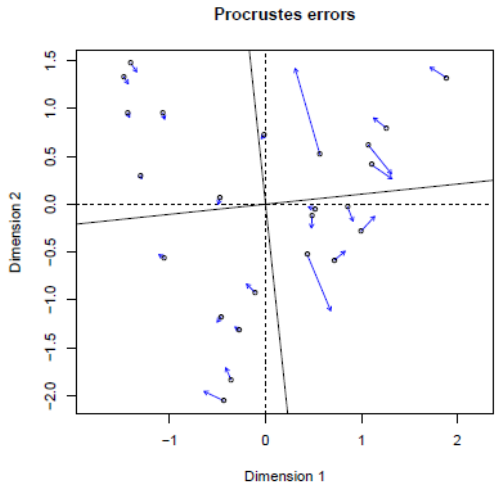
使用所有解释变量同时执行约束排序是比较普遍的。增加约束的解释数量实际上意味着放松约束：约束排序会变得更类似于非约束的排序。如果不做约束排序，最好使用非约束排序与向量拟合(或曲面拟合)，这样可以检测我们没有观察到的环境变量的解释部分。在受约束排序中，最好将约束的数量减少到仅有的几个，例如 3-5 个。

我们不鼓励同时使用所有可能的环境变量作为约束。但是，在公式模式中仍有用将所有变量都参与分析的快捷方式：

```
> mod1 <- cca(varespec ~ ., varechem)
> mod1
Call: cca(formula = varespec ~ N + P + K + Ca + Mg + S
+ Al + Fe + Mn + Zn + Mo + Baresoil + Humdepth + pH,
data = varechem)

              Inertia Proportion Rank
Total           2.083      1.000
Constrained     1.441      0.692   14
Unconstrained   0.642      0.308    9
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1  CCA2  CCA3  CCA4  CCA5  CCA6  CCA7  CCA8  CCA9 CCA10
0.439 0.292 0.163 0.142 0.118 0.089 0.070 0.058 0.031 0.013
CCA11 CCA12 CCA13 CCA14
0.008 0.007 0.006 0.005

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8    CA9
0.1978 0.1419 0.1012 0.0708 0.0533 0.0333 0.0189 0.0151 0.0095
```

这个结果可能与非约束排序的结果非常相似：

```
> plot(procrustes(cca(varespec), mod1))
```



我们应该减少约束的数量，找到重要的环境变量。原则上，约束排序与固定的实验模式配套，因此所有筛选变量的做法都是有危险的：可能有几个可供选择的模型，它们几乎一样好；数据的微小变化会导致所选模型的巨大变化；使用自适应策略可能没有通向最佳模型的路径。建模有其历史：不同的初始步骤可能导致非常不同的最终模型；显著性检验是有偏的，因为模型选择是以表现最佳为标准的。

这里展示了 vegan 如何使用标准的 R 函数 step()自动筛选变量。该函数采用 Akaike 信息准则(AIC)作为选择准则。AIC 是一种惩罚型拟合度测度：拟合度可以通过残差惯量（非约束部分）比例活动，惩罚度也就是解释变量的个数。原则上，AIC 是基于排序分析没有的似然对数值。然而，deviance() 函数将约束排序

中的非约束特征根转化为卡方值。在高斯模型中，这种偏差被当作平方和来处理。如果解释变量为连续定量变量，可以根据变量对约束特征根（惯量）的贡献来筛选变量。但如果解释变为为因子，情况是比较棘手的，因为因子会增加用于惩罚的自由度，但如何设置惩罚的大小是个难题。step()函数在帮助深入了解数据方面可能仍然很有用，但不应盲目地信任它（或根本不信任它），而只应将其视为建模中的一种辅助工具。

在冗长的介绍之后，使用 step()函数比解释它的工作原理简单得多。我们需要给出最初使用的模型以及模型空间。为此，我们需要另一个公式技巧：只有 1 作为约束的公式定义了一个非约束的模型。我们必须这样定义它，以便可以在最初的非约束模型中添加新的项。在建模中使用的 AIC 不是基于一个坚定的理论，因此我们也要求在每一步进行置换检验。在理想情况下，最终模型中所有包含的解释变量都应该是重要的，而所有被排除的解释变量都应该是不重要的。我们可以使用 formula 函数从拟合模型中提取最终的公式。下面的示例从一个非约束的模型 mod0 开始，然后逐步实现最简约的模型 mod1：

```
> mod0 <- cca(varespec ~ 1, varechem)
> mod <- step(mod0, scope = formula(mod1), test = "perm")
Start:  AIC=130.31
varespec ~ 1

           Df    AIC      F Pr(>F)
+ Al        1 128.61 3.6749  0.005 **
+ Mn        1 128.95 3.3115  0.005 **
+ Humdepth  1 129.24 3.0072  0.005 **
+ Baresoil  1 129.77 2.4574  0.020 *
+ Fe        1 129.79 2.4360  0.020 *
+ P         1 130.03 2.1926  0.030 *
+ Zn        1 130.30 1.9278  0.060 .
<none>       130.31
+ Mg        1 130.35 1.8749  0.075 .
+ K         1 130.37 1.8609  0.050 *
+ Ca        1 130.43 1.7959  0.060 .
+ pH        1 130.57 1.6560  0.115
+ S         1 130.72 1.5114  0.120
+ N         1 130.77 1.4644  0.170
+ Mo        1 131.19 1.0561  0.415
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=128.61
varespec ~ Al

           Df    AIC      F Pr(>F)
+ P         1 127.91 2.5001  0.005 **
+ K         1 128.09 2.3240  0.010 **
+ S         1 128.26 2.1596  0.015 *
+ Zn        1 128.44 1.9851  0.030 *
+ Mn        1 128.53 1.8945  0.025 *
<none>       128.61
+ Mg        1 128.70 1.7379  0.085 .
```

```
    + N          1 128.85 1.5900  0.065 .
    + Baresoil   1 128.88 1.5670  0.090 .
    + Ca         1 129.04 1.4180  0.190
    + Humdepth   1 129.08 1.3814  0.205
    + Mo         1 129.50 0.9884  0.435
    + pH         1 129.63 0.8753  0.575
    + Fe         1 130.02 0.5222  0.875
    - Al         1 130.31 3.6749  0.010 **
    ---
    Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    Step:  AIC=127.91
    varespec ~ Al + P

              Df    AIC      F Pr(>F)
+ K            1 127.44 2.1688  0.040 *
<none>           127.91
+ Baresoil     1 127.99 1.6606  0.070 .
+ N            1 128.11 1.5543  0.135
+ S            1 128.36 1.3351  0.255
+ Mn           1 128.44 1.2641  0.280
+ Zn           1 128.51 1.2002  0.300
+ Humdepth     1 128.56 1.1536  0.320
- P            1 128.61 2.5001  0.020 *
+ Mo           1 128.75 0.9837  0.435
+ Mg           1 128.79 0.9555  0.430
+ pH           1 128.82 0.9247  0.555
+ Fe           1 129.28 0.5253  0.875
+ Ca           1 129.36 0.4648  0.895
- Al           1 130.03 3.9401  0.005 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step:  AIC=127.44
varespec ~ Al + P + K
```

```
           Df    AIC      F Pr(>F)
<none>        127.44
+ N         1 127.59 1.5148  0.115
+ Baresoil  1 127.67 1.4544  0.120
+ Zn        1 127.84 1.3067  0.290
+ S         1 127.89 1.2604  0.230
- K         1 127.91 2.1688  0.035 *
+ Mo        1 127.92 1.2350  0.280
- P         1 128.09 2.3362  0.025 *
+ Mg        1 128.17 1.0300  0.400
+ Mn        1 128.34 0.8879  0.545
+ Humdepth  1 128.44 0.8056  0.570
+ Fe        1 128.79 0.5215  0.920
+ pH        1 128.81 0.5067  0.855
+ Ca        1 128.89 0.4358  0.910
- Al        1 130.14 4.3340  0.005 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> mod

Call: cca(formula = varespec ~ Al + P + K, data =
varechem)

              Inertia Proportion Rank
Total           2.083      1.000
Constrained     0.644      0.309    3
Unconstrained   1.439      0.691   20
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1  CCA2  CCA3
0.362 0.170 0.113

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8
0.350 0.220 0.185 0.155 0.135 0.100 0.077 0.054
(Showed only 8 of all 20 unconstrained eigenvalues)
```

我们用先前一直用的类似的模型结束（现在你就会明白为什么一开始用这个模型了）。AIC 值是根据残差计算来的，并且每增加一个参数，惩罚自由度为 2。每一次迭代 AIC 都会涉及到所有可能的增加(+)或减少(-)，并且变量都按 AIC 的排列列出。当出现<none>或当前模型排在顶端的时候，步骤停止。

step()函数模型的建立是易变的，模型的建立策略可以改变最终的模型。如果我们从最大的模型（mod1）开始，得到的最终模型就会不一样。

```
> modb <- step(mod1, scope = list(lower = formula(mod0), upper = formula(mod1)), trace = 0)
> modb

Call: cca(formula = varespec ~ P + K + Mg + S + Mn + Mo
+ Baresoil + Humdepth, data = varechem)

              Inertia Proportion Rank
Total           2.083      1.000
Constrained     1.117      0.536    8
Unconstrained   0.967      0.464   15
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1  CCA2  CCA3  CCA4  CCA5  CCA6  CCA7  CCA8
0.401 0.249 0.149 0.127 0.088 0.066 0.025 0.013

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8    CA9
0.2582 0.1881 0.1193 0.1020 0.0879 0.0609 0.0446 0.0278 0.0269
  CA10   CA11   CA12   CA13   CA14   CA15
0.0165 0.0136 0.0082 0.0066 0.0037 0.0024
```

这个模型的 AIC 值为 127.89，比先前的模型（127.44）要高（坏）。我们通过抑制追踪运算过程可以减少输出的页数，但 step 函数会自动添加计算的历史：

```
> modb$anova

  Step Df Deviance Resid. Df Resid. Dev   AIC
1      NA       NA         9       1551 130.1
2 - Fe  1    115.2        10       1667 129.8
3 - Al  1    106.0        11       1773 129.3
4  - N  1    117.5        12       1890 128.8
5 - pH  1    140.4        13       2031 128.5
6 - Ca  1    141.2        14       2172 128.1
7 - Zn  1    165.3        15       2337 127.9
```

在变量前向选择过程中变量 Al 是是第一个被选出来的，但是在后面的后向选择过程中却是第二个被去除的。变量 Al 和其他的解释变量有很强的相关性。这一点在查看全模型 mod1 的方差膨胀因子（vif）时更明显：

```
> vif.cca(mod1)

       N        P        K       Ca       Mg        S       Al
   1.982    6.029   12.009    9.926    9.811   18.379   21.193
      Fe       Mn       Zn       Mo Baresoil Humdepth       pH
   9.128    5.380    7.740    4.320    2.254    6.013    7.389
```

经验法则：当 vif＞10 时表示一个变量强烈依赖于其他变量且没有独立信息。换句话说，这个变量可能不应该被去除，而另一些可选择的变量应该被去除。在前向选择的模型中所有变量的包括 Al 的 vif 值都很小。

```
> vif.cca(mod)

    Al      P      K
1.012 2.365 2.379
```

4.4 线性组合和加权平均

在约束排序中有两种样方坐标:

1. 线性组合坐标 lc：约束变量的线性组合。
2. 加权平均坐标 wa：物种坐标的加权平均。

这两种坐标是很相似的，它们的(加权)相关性称为物种-环境相关性:

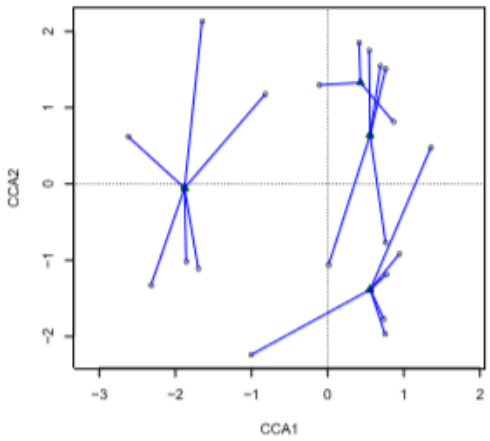
```
> spenvcor(mod)

   CCA1    CCA2    CCA3
0.8555 0.8133 0.8793
```

相关系数对于单个偏离正常的极值非常敏感，就像上述例子中由于轴 3 有一些极值，所以其有“最好”的相关性，用特征根来描述 lc 坐标和 wa 坐标之间的相似性更加合适。

排序图到底用 lc 坐标还是 wa 坐标来作为样方坐标，人们意见不一。vegan 包更倾向于 wa 坐标，而 CCA 的主要商业软件更倾向于 lc 坐标。vegan 包附带了一份单独的文件（“vegan FAQ”），该文件更详细地解释了这个问题，但我也会在这里讨论这个主题。

更喜欢 WA 坐标的实际原因是，它们不受环境变量中的随机误差影响。所有的生态观测都有随机误差，因此最好使用能够抵抗这种变化的分数。另一点是，我认为 LC 坐标是一种约束结果:分值只依赖于环境变量，而群落组成并不影 LC 坐标。WA 坐标是基于群落组成的坐标，但是它们尽可能地与约束排序的结果相似。当使用单个因子变量作为约束时，这种相似性尤为明显: LC 坐标在因子的每个水平内都恒定，并且落在同一点。WA 坐标不同水平具有不同的坐标。

vegan 包有一个做图函数 ordispider 可以将 WA 坐标和 LC 坐标组合。



将 WA 坐标与相应的 LC 坐标相结合。单因子约束：

```
> dune.cca <- cca(dune ~ Management, dune.env)
> plot(dune.cca, display = c("lc", "wa"), type = "p")
> ordispider(dune.cca, col="blue")
```

其解释与判别分析相似: LC 坐标给出了预测值的质心，
WA 坐标给出了预测值。对于不同的类，组之间没有重叠。一般来说，ordisider 给出的线段的长度是物种环境相关的可视化图像。

4.5 双序图箭头和环境标定

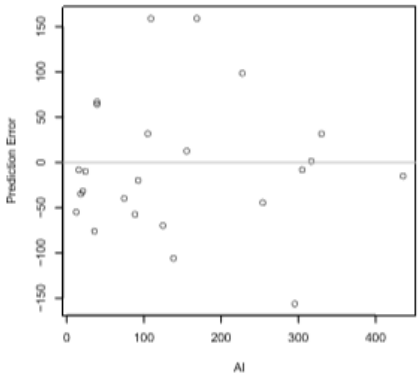
双序图箭头是约束排序图的重要组成部分。箭头是基于 LC 坐标和环境变量的(加权)相关。这些箭头的长度已经被标准化，当这些箭头被投影到 2D 坐标图上时，如果它们离开平面，它们看起来会更短。

在 vegan 中，双序图箭头总是以类似的方式进行，而不考虑位置或物种的缩放(scaling)。当默认 scaling=2 时，双序图箭头与样方的关系最准，但当 scaling=1 时，它们与物种的关系更好。

双序图箭头的标准解释是，一个样方点应该垂直投影到箭头上，该点的位置为预测变量的值。箭头从环境变量的(加权)均值开始，值向箭头方向增大，向相反方向减小。然后我们仍然要算出变化量的单位。Calibrate.cca 函数在 vegan 包中自动执行此操作。让我们检查带有三个约束变量的 step 函数的结果：

```
> pred <- calibrate(mod)
> head(pred)
```

```
          Al     P      K
18 103.219 25.64  80.57
15  30.661 47.25 190.90
24  32.105 72.80 208.34
27   7.178 64.44 241.89
23  14.321 38.50 125.73
19 136.568 54.39 182.60
```

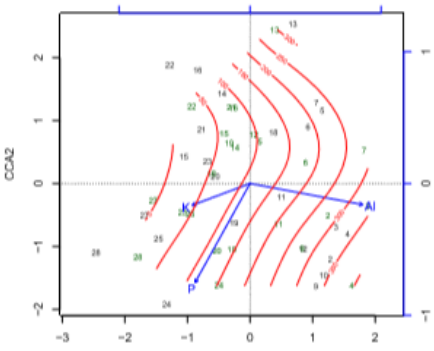


实际上，这不是基于双序图箭头，而是基于约束排序中的回归系数。双序图箭头只能看作是一种直观的近似。该拟合以完全约束秩作为默认值，并对所有约束同时进行拟合。这个例子绘制了预测的残差图

```
> with(varechem, plot(Al, pred[,"Al"] - Al, ylab="Prediction
      Error"))
> abline(h=0, col="grey")
```

vegan 包提供了基于 mgcv 包中 gam 函数的 ordisurf 函数，可以自动检测所需的平滑的阶数，可以用来检验双序图法线性假设。函数进行加权拟合，模型应与箭头拟合中使用的模型一致。在我们的例子中，铝（Al）是三个约束环境因子中最重要的一个。现在我们应该将模型与 lc 坐标相匹配，就像箭头一样：

```
> plot(mod, display = c("bp", "wa", "lc"))
> ef <- with(varechem, ordisurf(mod, Al, display = "lc", add =
TRUE))
```



结果和我们预想的不一样:我们得到的是曲线而不是垂直于 Al 箭头的平行线。在这种情况下，我们似乎不能使用线性投影。线性投影实际上是可行的，但只是在完全约束排序（full constrained rank）下，或者在三维空间中。当我们把多维的数据投影到一个平面上时，就会有畸变。一旦我们有两个以上约束轴，投影就变得不好用——但有时投影可能工作得相当好。在这种情况下，磷（P）会显示一个线性响应的曲面，尽管它在模型中没有 Al 那么显著。

4.6 条件模型或偏模型

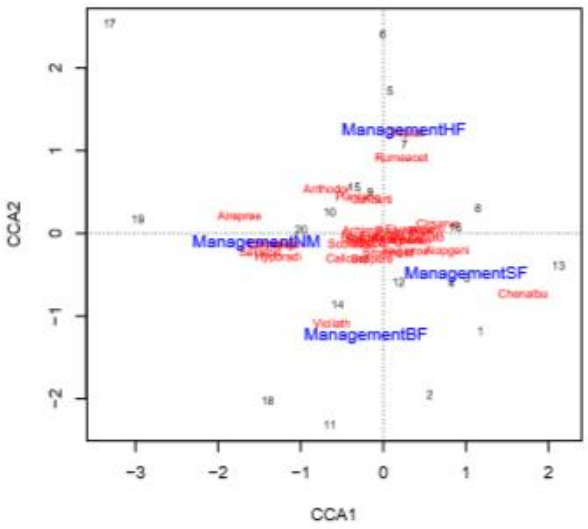
在约束其他变量之前，可以先将一些环境变量的影响从排序中去除。这种分析被称为以变量为条件的分析，也就是说，剔除某些变量引起的变化后的分析是偏分析。这些条件变量通常是“随机的”或背景变量，它们的影响在基于“固定的”或有趣的变量的分析中被删除。

在 vegan 中，约束排序公式模式可以包含一个 Condition 参数，参数是指定一个或多个解释变量，在使用其他变量约束之前，这些解释变量的影响将从分析中删除。举个例子，让我们

来看看去除 Moisture 引起的自然变化后，Management 的效应是什么:

> dune.cca <- cca(dune ~ Management + Condition(Moisture), dune.env)
> plot(dune.cca)
> dune.cca

```
Call: cca(formula = dune ~ Management +
Condition(Moisture), data = dune.env)

              Inertia Proportion Rank
Total           2.115      1.000
Conditional     0.628      0.297    3
Constrained     0.374      0.177    3
Unconstrained   1.113      0.526   13
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.2278 0.0849 0.0614

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8   CA9  CA10
0.350 0.152 0.125 0.110 0.092 0.077 0.059 0.048 0.037 0.022
 CA11  CA12  CA13
0.021 0.011 0.008
```



将总惯量分解为条件惯量、约束惯量和剩余无约束惯量三部分。我们以前用 Management 作为唯一的约束来拟合模型，在那种情况下，约束惯性明显高于当前这个模型。似乎不同的 Management 是在不同的自然条件下引起的，我们之前归因于 Management 的变化可能是由于降雨量引起的。

我们可以在条件模型中对 Management 进行置换检验，也可以单独进行检验：

> anova(dune.cca, perm.max = 2000)

```
Permutation test for cca under reduced model
Permutation: free
Number of permutations: 999

Model: cca(formula = dune ~ Management + Condition(Moisture), data = dune.env)
         Df ChiSquare    F Pr(>F)
Model     3     0.374 1.46  0.034 *
Residual 13     1.113
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> anova(cca(dune ~ Management, dune.env))

Permutation test for cca under reduced model
Permutation: free
Number of permutations: 999

Model: cca(formula = dune ~ Management, data = dune.env)
         Df ChiSquare    F Pr(>F)
Model     3     0.604 2.13  0.001 ***
Residual 16     1.511
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

在作为单一解释变量的时候，Management 似乎非常重要，但消除由 Moisture 引起的变化后，情况就不那么明显了。

anova 函数（就像 vegan 包中的任何置换检验一样）可以进行局部的置换检验，以便置换只在分层抽样或因子变量水平内进行：

> with (dune.env, anova(dune.cca, strata = Moisture))

```
Permutation test for cca under reduced model
Blocks:  strata
Permutation: free
Number of permutations: 999

Model: cca(formula = dune ~ Management + Condition(Moisture), data = dune.env)
         Df ChiSquare    F Pr(>F)
Model     3     0.374 1.46  0.008 **
Residual 13     1.113
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

条件模型或偏模型是变差分解（variation partitioning）的基础。在某些情况下，这会给出有意义的结果，但是环境变量组对于无偏分解应该是非线性独立的。如果环境变量组具有多项式依赖性，则惯量的某些分量甚至可能变为负值（这是不可能的）。这种更高级的依赖关系几乎肯定会出现在大量变量和大量组中。然而，varpart 函数只能执行在两到四个解释变量的分解。

5 相异性与环境

我们已经讨论了非约束排序的环境解释和环境约束的典范排序。这两种方法

都多维空间的降维技术，并且主要检验前面几维功效。有时我们也希望在不使用排序的情况下，或在整个多维空间内分析植被与环境的关系。通常这些方法在分析中使用相异矩阵。Vegan 包中推荐的方法是 adonis 函数，它使用距离矩阵实现多元方差分析（multivariate analysis of variances using distance matrices）。函数 adonis 可以处理连续和因子预测因子。Vegan 包中的其他方法包括多反应置换法（multiresponse permutation procedure，MRPP）和相似性分析法（anosim）。这两个都只处理定性预测因子，而且它们不如 adonis 函数稳定。

5.1 adonis：基于相异矩阵的多元方差分析

函数 adonis 为变差的来源划分不同的相异分区，并使用置换检验来验证这些分区的显著性。由于使用欧氏距离，其结果与 RDA 及其方差分析置换检验中的结果相似，但 adonis 可以处理任何相异矩阵对象。

下面的案例使用 adonis 函数研究沙丘草地数据（dune meadow）中 Management 管理类之间的 beta 多样性。我们将 beta 多样性定义为物种面积曲线的斜率，或是 Arrhenius 模型的指数 z，在该研究地区中物种数量 S 取决于研究区域的大小 X。对于样点的成对比较，可以从两个样点之间共享的物种数量（a）和每个样点特有的物种数量（b 和 c）中找到斜率 z。一般认为，z≈0.3 意味着随机抽样变异性，只有较高的值才表示真正的系统偏差。Arrhenius 模型的指数 z 可以直接用函数 betaddiver 获得，它也提供了许多其他的成对 beta 多样性指数。

```
> betad <- betadiver(dune, "z")
```

函数 adonis 可以使用公式模式，使用数据可以是相异矩阵，也可以是数据框，如果是直接使用数据框，adonis 函数是调用 vegdist 函数来计算相异矩阵。

```
> adonis(betad ~ Management, dune.env, perm=200)
```

```
Call:
adonis(formula = betad ~ Management, data = dune.env, permutations = 200)

Permutation: free
Number of permutations: 200

Terms added sequentially (first to last)

           Df SumsOfSqs MeanSqs F.Model    R2 Pr(>F)
Management  3      1.24   0.412    2.36 0.307   0.01 **
Residuals  16      2.79   0.174         0.693
Total      19      4.03                 1.000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

模型可能更复杂，如果有几个因子，则进行方差分析的置换检验：

```
> adonis(betad ~ A1*Management, dune.env, perm = 200)
```

```
Call:
adonis(formula = betad ~ A1 * Management, data = dune.env, permutations = 200)

Permutation: free
Number of permutations: 200

Terms added sequentially (first to last)

              Df SumsOfSqs MeanSqs F.Model    R2 Pr(>F)
A1             1      0.65   0.655    4.13 0.163  0.005 **
Management     3      1.00   0.334    2.11 0.249  0.055 .
A1:Management  3      0.47   0.156    0.99 0.117  0.458
Residuals     12      1.90   0.158         0.472
Total         19      4.03                 1.000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

5.2 组的同质性和 beta 多样性

Adonis 函数研究了组内平均值的差异，而函数 betadisper 研究了组同质性的差异。函数 adonis 类似于多元方差分析，betadaver 类似于 levene 对方差齐性的 levene 检验。

以下案例继续前面部分的分析，并检查 beta 多样性。函数只能使用一个因子作为自变量，且不能用公式模式，因此需要 attach 数据框或使用 with 语句使因子对函数可用：

> mod <- with(dune.env, betadisper(betad, Management))

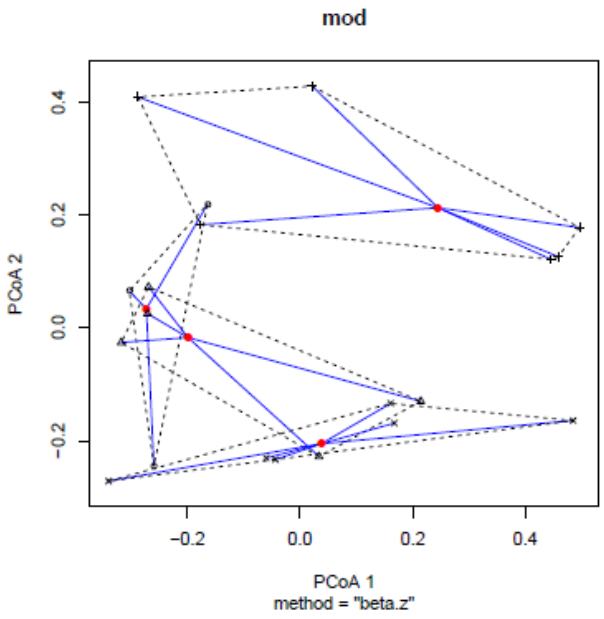
> mod

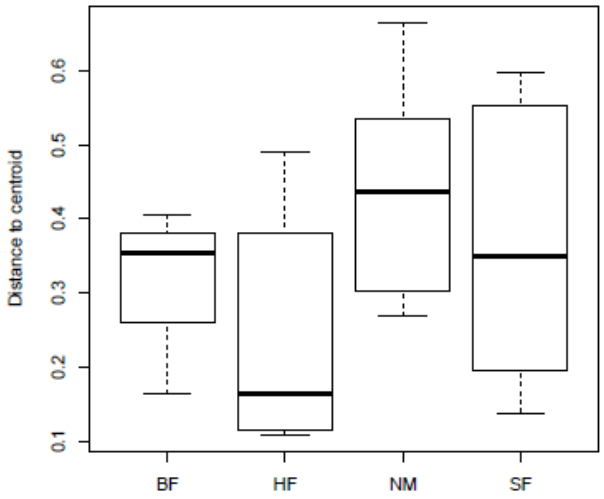
```
        Homogeneity of multivariate dispersions

Call: betadisper(d = betad, group = Management)

No. of Positive Eigenvalues: 12
No. of Negative Eigenvalues: 7

Average distance to median:
   BF    HF    NM    SF
0.308 0.251 0.441 0.363

Eigenvalues for PCoA axes:
PCoA1 PCoA2 PCoA3 PCoA4 PCoA5 PCoA6 PCoA7 PCoA8
1.655 0.887 0.533 0.374 0.287 0.224 0.161 0.081
```

该函数具有用于图形显示的绘图和箱线图的结果。

> plot(mod)

> boxplot(mod)





模型拟合的显著性可以使用标准参数方差分析或置换试验（permutest）进行分析：

> anova(mod)

```
Analysis of Variance Table

Response: Distances
          Df Sum Sq Mean Sq F value Pr(>F)
Groups     3  0.104  0.0348    1.26   0.32
Residuals 16  0.443  0.0277
```

> permutest(mod)

```
Permutation test for homogeneity of multivariate
dispersions
Permutation: free
Number of permutations: 999

Response: Distances
          Df Sum Sq Mean Sq    F N.Perm Pr(>F)
Groups     3  0.104  0.0348 1.26    999   0.31
Residuals 16  0.443  0.0277
```

此外，还可以使用 Tukey HSD 检验来分析各组之间的两两差异：

> *TukeyHSD(mod)*

```
    Tukey multiple comparisons of means
      95% family-wise confidence level

  Fit: aov(formula = distances ~ group, data = df)

  $group
            diff       lwr    upr  p adj
  HF-BF -0.05682 -0.40452 0.2909 0.9651
  NM-BF  0.13256 -0.20409 0.4692 0.6791
  SF-BF  0.05547 -0.28119 0.3921 0.9643
  NM-HF  0.18938 -0.09891 0.4777 0.2752
  SF-HF  0.11229 -0.17600 0.4006 0.6862
  SF-NM -0.07709 -0.35197 0.1978 0.8523
```

5.3 Mantel 检验

Mantel 检验是比较两组的相异矩阵的相关性。由于 N 个对象之间存在 N（N-1）/2 的相异对，所以常规的显著性检验是不适用的。虽然 Mantel 开发了渐近检验统计量，但 *vegan* 包中的 mantel 函数使用的是置换检验。

在这个例子中，我们研究了位于意大利北部城市—的地衣群落（varespec）与环境的相关性。我们可以在简单排序后使用了后加环境因子拟合。然而，排序轴和环境因子可能是非线性关系，所以我们现在尝试使用 mantel 函数。首先，我们对环境变量进行主成分分析，然后计算第一主成分的相异矩阵，我们使用标准的 R 函数 *prcomp* 进行分析，当然 *princomp* 函数或 *rda* 也可以达到同样的效果，而 *vegan* 包中的 *scores* 函数都能跟这些方法匹配。以下对群落相异矩阵计算使用与之前 MetaMDS 的相同标准化。

> *pc <- prcomp(varechem, scale = TRUE)*

> *pc<- 坐标(pc, display = "sites", choices = 1:4)*

> *edis <- vegdist(pc, method = "euclid")*

> vare.dis <- vegdist(wisconsin(sqrt(varespec)))

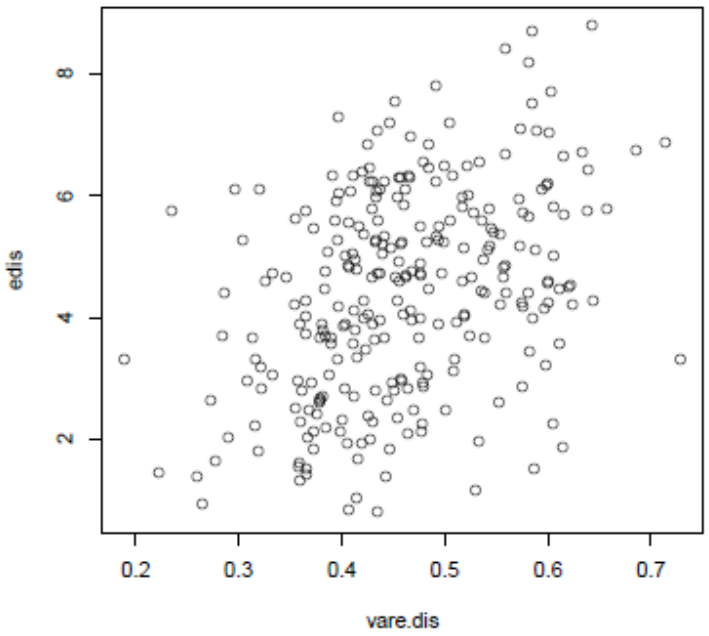
> mantel(vare.dis, edis)

```
Mantel statistic based on Pearson's product-moment correlation

Call:
mantel(xdis = vare.dis, ydis = edis)

Mantel statistic r: 0.381
      Significance: 0.001

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.147 0.186 0.218 0.250
Permutation: free
Number of permutations: 999
```

我们可以在主成分分析中使用所选的环境变量，或者我们可以直接使用标准化的环境变量而不进行主成分分析—体验不同。*bioenv* 函数为排序和环境比较选择最优子集提供了一个有趣的选择方案。还有一个偏 *Mantel* 检验，我们可以从分析中消除第三组相异矩阵的影响，但其结果通常很难解释。

Mantel 函数没有诊断绘图函数。但是，可以直接绘制两个相异矩阵散点图：

> plot(vare.dis, edis)



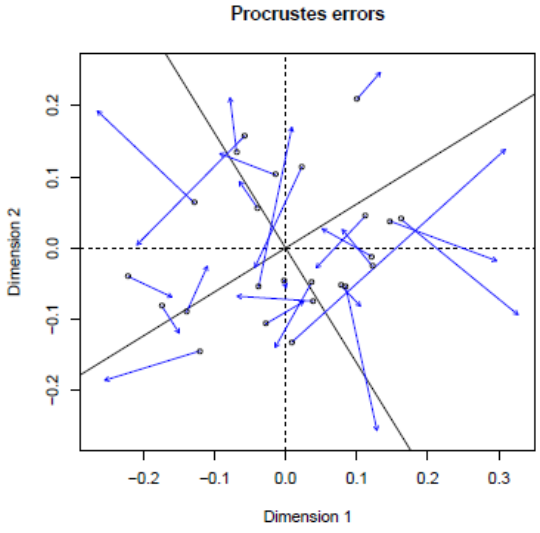
如果这种关系或多或少是单调的，甚至是线性的和正相关，那么一切都是好的。在空间模型中，我们甚至可以观察到表示空间聚集的单峰模型。

5.4 Protest:普鲁克检验

普鲁克检验或 protest 比较使用对称的普鲁克分析来比较两种排序。它是 Mantel 检验的替代版本，但使用的是降维空间，而不是完全相异矩阵。我们可以重复先前的分析，但是现在比较 metaMDS 排序结果和环境变量 PCA 分析前两轴。

> pc <- scores(pc, choices = 1:2)
> pro <- protest(vare.mds, pc)
> plot(pro)
> pro

```
Call:
protest(X = vare.mds, Y = pc)

Procrustes Sum of Squares (m12 squared):        0.533
Correlation in a symmetric Procrustes rotation: 0.683
Significance:  0.001

Permutation: free
Number of permutations: 999
```



$$r = \sqrt{1 - m^2}$$

其显著性是通过置换检验来评估。现在的统计量是由对称的普鲁克残差 m^2 导出的普鲁克相关系数 r。对于同样的相异矩阵，普鲁克相关明显高于 Mantel 相关。因为在较低维数中，我们去除数据中的干扰，这可能解释更高的相关性(也可

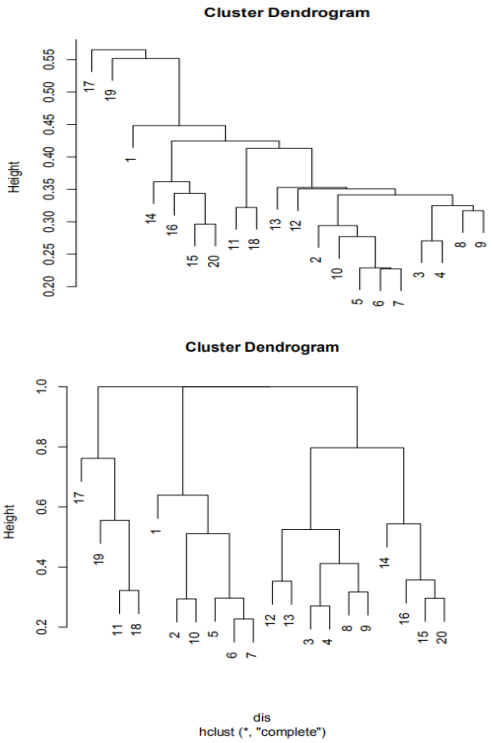
能通过不同的方法来计算相关性)。然而，这两种方法都同样都是显著的。显著性通常不是一个重要的概念：在大数据集中，即使与随机性有很小的偏差也可能具有很高的显著性。Protest 函数提供了图形演示（“普鲁克叠加图”），这在评估结构之间的一致性方面可能更有用。

protest（作为普通的普鲁克分析）经常用于评估不同群落排序之间的相似之处。这就是所谓的同余分析（analysis of congruence）

6 分类

vegan 包主要用于排序和多样性分析，并且对分类的支持很少。这还有其他几个具有更广泛分类函数的 R 包。在群落生态分析的包中，由 Dave Roberts 编写的 *labdsv* 包在分类函数方面尤为突出。

本章描述了在群落生态学中执行对许多群落生态学家来说足够简单的分类任务。



6.1 聚类分析

层次聚类可以使用标准的 R 函数 *hclust* 完成。此外，还有其他几个聚类包，其中一些可能与 *hclust* 兼容。*hclust* 函数需要一个距离矩阵作为输入对象。

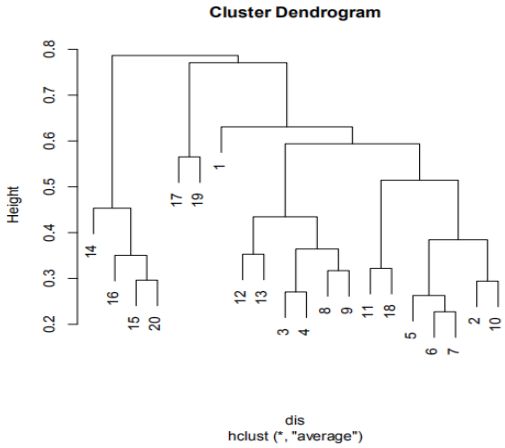
hclust 函数提供了几种可供选择的聚类策略。在群落生态学中，普遍使用的是“单连接聚类（最近邻体法）”、“完全链接聚类（最远邻体法）”和“平均距离聚类”。下面一一举例说明：

```
> dis <- vegdist(dune)
> clus <- hclust(dis, "single")
> plot(clus)
```

有些人更喜欢这种单连接聚类（最近邻体法），因为它在概念上与最小生成树相关，而最小生成树可以很好地用序号表示，并且能够发现数据的不连续性。然而，这种方法会让大组容易吸收新的成员。

> cluc <- hclust(dis, "complete")

> plot(cluc)



有些人更喜欢完全链接聚类（最远邻体法），因为它能形成紧凑的集群。然而，这在一定程度上是这种方法的一种假像：不允许集群增长，因为最远距离连接法准则会被违反。

> clua <- hclust(dis, "average")

> plot(clua)

有些人(包括我)更喜欢平均距离聚类，因为这似乎是前两个极端之间的折衷，而且在分组中更中立。有几种另外的方法可以松散地连接到“均值连接”家族。word 的方法在出版物中似乎很受欢迎。它试图使聚集中的方差最小化。默认的“均值”方法是一种通常被称 *upgma* 的方法，它在过去的遗传学中很流行。

所有这些聚类方法都是聚合模式。他们首先将两个最相似的站点组合在一起，然后，他们将点到点或组，或将组到组进行组合。融合标准各不相同，所有图表中的纵坐标表示融合程度，这些数字因方法而异，但它们都基于与范围相同的差异矩阵：

> range(dis)

[1] 0.2273 1.0000

在所有例子中，第一次融合是在相同的两个最相似的位点之间，并且具有相同的最小相异性。在 complete linkage 聚类中，最后一次融合将两个差异最大组结合在一起。在 single linkage 聚类中，融合水平始终处于群体间最小的差异，且所融合的水平远低于 complete linkage 聚类的水平。Average linkage 聚类在组中心点之间进行融合，其融合水平在前两个树之间。两点之间的估计差异在于它们在树中融合的水平。函数 cophenetic 用来度量聚类树上样方之间的图上差异性—函数的名称反映了数值分类学中的聚类历史。cophenetic 相关性是测量原始相异性与树估计相异性之间的相似性。下面是做上面三个聚类的 cophenetic 相关:

> cor(dis, cophenetic(clus))

[1] 0.6602

> cor(dis, cophenetic(cluc))

[1] 0.6707

> cor(dis, cophenetic(clua))

[1] 0.8169

近似差异性与排序执行相同的任务，并且 Average linkage 聚类有着最好的表现。

6.2 类的显示和解释

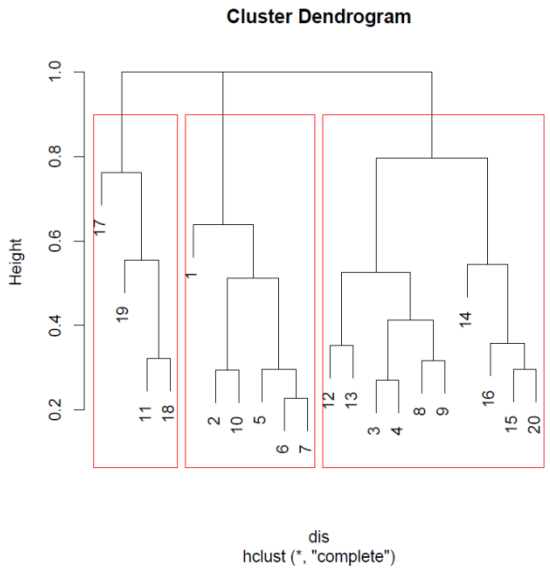
聚类分析执行层次聚类，其结果可以在多个有点的层次上检验：极端情况是每个点单独一组，或者所有点都属于同一个聚类。我们通常希望在一定的层次上检验聚类，将其作为一个不分层次的、由一定数量的聚类组成的系统来检验。聚类的分组确定是通过在一定的融合水平上对树进行切割，从而得到所需的聚类数目。

基于 R 提供了函数 rect.hclust 来可视化确定分组聚类树，函数 cutree 提供具有一定类数的分类向量：

> plot(cluc)

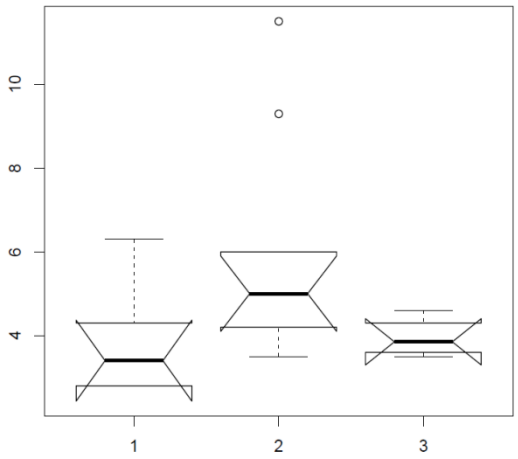
> rect.hclust(cluc, 3)

> grp <- cutree(cluc, 3)



分类向量可以变为因子变量作为他用。检验群落分类的好坏的一种自然方法是看看分组是否很好地预测未用于聚类的外部环境变量。Dune 数据中唯一的连续变量是 A1，看看分组下的箱线图：

> boxplot(A1 ~ grp, data=dune.env, notch = TRUE)



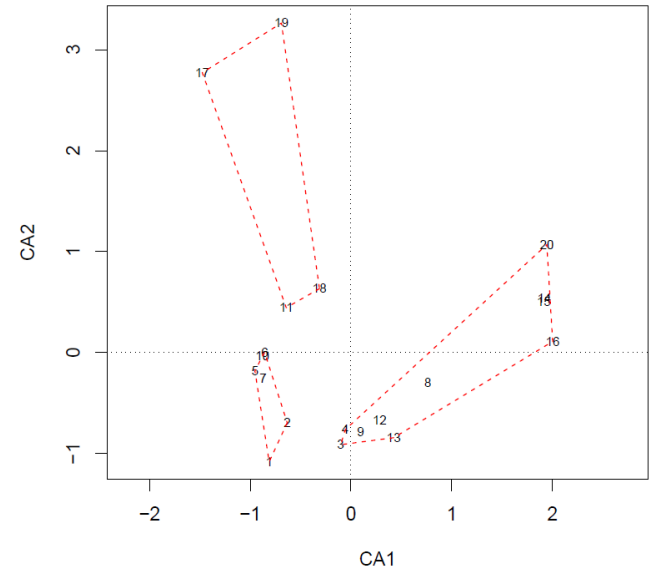
如果愿意的话，我们可以使用所有带有因子的常规统计方法，例如函数 lm 或 AOV 来正式检验聚类的分之间连续环境变量“显著性”。分类也可以与外部因子变量进行比较。然而，vegan 并没有为此提供任何工具。

聚类结果可以在排序图中显示。可以使用几个vegan函数：ordihull, ordispider, ordiellipse. 我们将只以第一个为例：

> ord <- cca(dune)

> plot(ord, display = "sites")

> ordihull(ord, grp, lty = 2, col = "red")

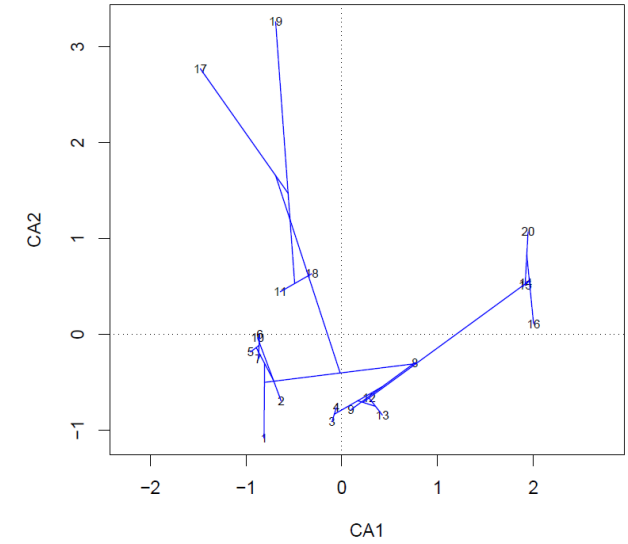


有时有人说，排序中的重叠类可以为聚类做交叉检查：如果排序图中的集群看起来不同，那么(两者)分析可能都是足够的。但是，如果聚类簇有重叠，聚类分析仍然可以是好的。可能需要三个或更多轴才能显示多变量类结构。此外，排序和分类可能使用不同的标准。在我们的示例中，CA 使用加权的卡方距离，而聚类使用可能完全不同的 Bray-Curtis 相异系数。通过使用动态 3D 图形，函数 ordirgl 及其关联函数 orglSpider 可用于检验分类。

vegan 包有 ordicluster 函数，它可以在排序中添加聚类树：

> plot(ord, display="sites")

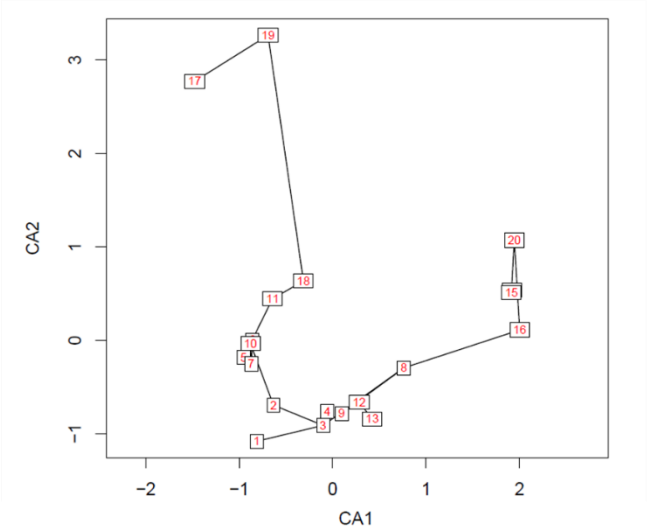
> ordicluster(ord, cluc, col="blue")



Single linkage 聚类是排序图中最常用的方法。Single linkage 聚类算法在聚类算法中有其独特之处，因为它总是将点到点结合在一起：只有最近的点才能被识别，并且没有使用同组聚类其他成员的信息。然而，聚类树隐藏了下面这些信息：它只显示聚类之间的融合，而不显示哪些是被连接的实际点。连接各个点的树称为最小生成树(MST)。在图论中，“树”是一个没有循环的连通图，“生成树”是连接所有点的树，最小生成树是连接段的总长度最短的树。vegan 中的 spantree 函数能找到了这棵树，并且它有一个线函数将树覆盖到排序上：

> mst <- spantree(dis, toolong = 1)

> plot(mst, ord=ord, pch=21, col = "red", bg = "yellow", type = "t")



在这里用到相异指数中，距离=1 意味着与两个样点没有任何共同之处。函

数 spantree 将这些最大的相异性看作是丢失的数据，而不使用它们来构建树。如果由于这些缺失值而无法连接所有点，则结果将由断开的连接的生成树组成。在图论中，这被称为“森林”。MST 有时用于交叉检查排序：如果树是线性的，排序可能是好的。弯曲的树可能表示弓形或马蹄形效应，而凌乱的树则表示排序不当，或需要更多的维度。然而，结果往往是难以解释的。

6.3 分类群落表

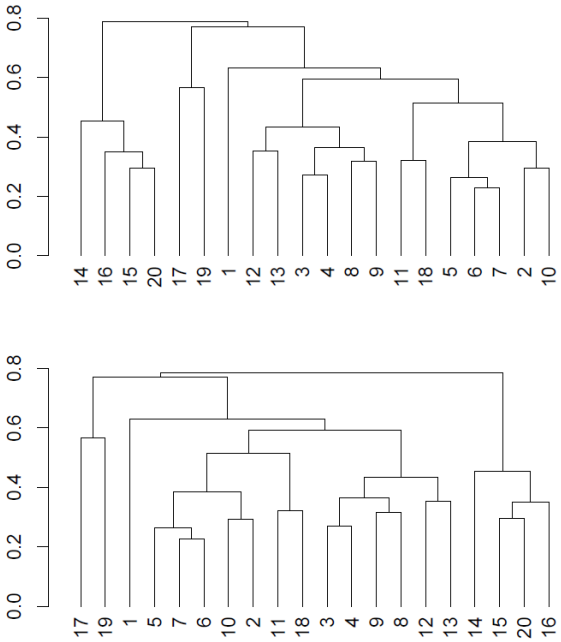
分类的目的往往是制作一个分类的群落表。为了达到此目的，应将位点和物种都整理好，使表格看起来有条理。最初的聚类可能不是理想的结构，因为在聚类树中，位点的排序并不严格。您可以使用任何分支并围绕它的节点旋转，聚类是相同的。树形绘制算法使用启发式规则使树看起来美观，但这种排序可能不是结构化群落表的最优排序。

基于 R 有一个名为 dendrogram 的通用树类，它被用作任何树状表示的公共基础。这个类有一个函数，可以根据某个外部变量重新排序一棵树。通过 as.dendrogram 函数，hclust 结果可以改为 dendrogram 树，并且可以通过 reorder 函数来重新排序。Dune 数据中唯一的连续变量是 A1，这可用于排列树。但是，对于结构良好的群落表，我们使用了另一个技巧：CA 是一种排序方法，它能最优地将表构造成对角线结构，我们可以使用它的第一个轴重新排序树：

```
> wa <- scores(ord, display = "sites", choices = 1)
> den <- as.dendrogram(clua)
> oden <- reorder(den, wa, mean)
```

结果确实发生了变化，除了叶子的顺序之外，可能需要花一些功夫才能看到这两棵树真的是相同的。

```
> op <- par(mfrow=c(2,1), mar=c(3,5,1,2)+.1)
> plot(den)
> plot(oden)
> par(op)
```



vegan 中的 vegemite 函数制作紧凑的植被样方表。它可以使用一个参数来安排位点（和物种，如果可能的话）。这个参数可以是用于排列位点的向量，也可以是排序结果，也可以是 hclust 结果或 dendrogram 对象。

后记：

vegan 包自开发以来已经有接近两万次的引用，不断更新的 vegan 包函数也正在成为数量生态学分析和挖掘数据必备的工具。翻译本教程的初衷是方便国内生态学从学从业人员使用 R 语言分析自己的数据，使入门的同行对 vegan 工具包特别是 vegan 所提供的群落生态学分析思路有一个轮廓的认识。

初稿由中国科学院大学 2019 年 R 语言课程选课研究生张剑坛、黄依依、赖媛、蒋芬、刘洋、易佳慧、黄龙、李俊杰、彭雨璇、张梓良、李学红、辛佩琦、李玉辉、丁聪等同学翻译，由周运来和赖江山修改。