[R-sig-eco] null models with continuous abundance data

Etienne Laliberté etiennelaliberte at gmail.com
Sun Jan 10 21:12:11 CET 2010

- Previous message: [R-sig-eco] Job Opening: Quantitatively savvy biologist
- Next message: [R-sig-eco] null models with continuous abundance data
- **Messages sorted by:** [date] [thread] [subject] [author]

Many thanks again Carsten for your answer.

>*From this interesting discussion, it seems that there are three*
different and somewhat distinct potential problems with null model approaches. Here's my attempt at summarizing them. Please correct me if I'm wrong.

1) non-randomness of null matrices: This can happen for example in sequential algorithms, where a null matrix may be quite similar to the previous one. I think this was the main point raised by Peter, and seems to be primarily linked to the nature of the algorithm (and not the constraints). For example, permatswap(..., method = "swap") vs. permatswap(..., method = "quasiswap"). His function summary.permat is a good way of checking this.

2) not enough unique matrices obtained: one should check how many unique null matrices are obtained. A small matrix with many constraints may lead to few unique null matrices. This does not imply that the algorithm has the problem described in #1 though. Having few unique matrices may make P-values obtained in tests meaningless. This was the main point raised by Carsten.

3) non-realism of null model: it is easy to build null models that are too null and biologically irrelevant. In that case it becomes very easy to detect significant patterns, and to some extent this could maybe be compared to a type-I error? Adding constraints is required to isolate the feature we want to test as much as possible, but too many constraints can sometimes make the test overly conservative (could maybe compare this to a type-II error?), and can increase the likelihood of problem #2.

I *think* this was the point Jari raised about r2dtable(), i.e. that is is too null and leads to null matrices with lower variance than the observed matrix because it tends to fill the zeros (and in doing so reduce the high abundances because marginal sums need to be respected), which has nothing to do with problem #1 or #2.

Feebdack would be appreciated. I think it's important not to confuse these different problems.

Cheers

Etienne

Le vendredi 08 janvier 2010 à 08:32 +0100, Carsten Dormann a écrit :
> *Dear Etienne,*
>
> *although certainly not as thorough as Peter/Jari's approach would be, I*
> *find that "problematic" null models become noticeable when you produce*

> only few of them (say 100 or so) and table (or plot the density) of
> their indices (say for example the Shannon diversity of entries or
> alike). What often happens with "problematic" null models is that they
> show only very few alternative "random" combinations, i.e. that I see 2
> to 5 humps of the density curve. That clearly indicates that only very
> few alternative combinations are possible and hence that the null model
> is too constrained.
> Using 1000s of null models might blur this picture a bit but should give
> the same idea.
>
> Jari, did I understand you right that think the r2dtable doesn't
> randomly and unbiasedly sample the space of possible combinations? You
> wrote about "too regular" matrices. I took r2dtable (or rather the
> Patefield algorithm) for granted, since it apparently is also under the
> hood of Fisher's test etc.
> That should be relatively easy to find out, for a given matrix, by
> creating endless numbers of null models and tabulating their unique
> frequencies. Ideally, this should show a uniform distribution.
>
> require(bipartite)
> data(Safariland)
> nulls <- r2dtable(10000, r=rowSums(Safariland), c=colSums(Safariland))
>
> # this takes a few minutes:
> nonsame <- 1:3
> for (i in 1:10000){
>   nonsame[i] <- sum(sapply(nulls, function(x) identical(x, nulls[[i]])))
> }
> table(nonsame-1)
> #   0
> #10000
>
> This means, there are no two identical random matrices. Encouraging (for
> this example)!
>
> HOWEVER, this does not mean that the whole sample space is covered.
> Furthermore, these null models may still be similarly "regular". But at
> least the r2dtable samples a large set of matrices. Maybe a value of 1e6
> to 1e9 would be better, and then some ordination of the matrices to
> visualise the coverage of the matrix space?
>
>
> Best wishes,
>
> Carsten
>
>
>
> On 07/01/2010 21:42, Etienne Laliberté wrote:
> > Many thanks Jari for your input.
> >
> > I'll have a look at this backtracking method and see how I could
> > implement it.
> >
> > Ensuring that the null matrices are indeed random is clearly good
> > advice, and I'd need to do this, but do you have any suggestions on how
> > to do this in practice? A copy of the script you've used to test Peter's
> > null model algorithms would be very useful.
> >
> > Thanks
> >
> > Etienne
> >
> >
> > Le jeudi 07 janvier 2010 à 22:24 +0200, Jari Oksanen a écrit :

> >
> >> On 07/01/2010 21:48, "Etienne Laliberté"<etiennelaliberte at gmail.com> wrote:
> >>
> >>
> >>> Many thanks again Carsten.
> >>>
> >> Yes, you're right that care must be taken to
> >>
> >>> ensure that a decent number
> >>>
> >> of unique random matrices must be obtained. I
> >>
> >>> don't think it would be a
> >>>
> >> problem in my case given that transforming my
> >>
> >>> continuous abundance data
> >>>
> >> to count by
> >>
> >> mat2<- floor(mat * 100 / min(mat[mat>
> >>
> >>> 0]))
> >>>
> >> can quickly lead to a very large number of individuals (hundreds
> >>
> >>> of
> >>>
> >> thousands if not> million in some cases). The issue then becomes
> >>
> >>> mostly
> >>>
> >> one of computing speed, but that's another story.
> >>
> >> Following your
> >>
> >>> suggestion, I came up with a simple (read naïve) R
> >>>
> >> algorithm that starts with
> >>
> >>> a binary matrix obtained by commsimulator
> >>>
> >> then fills it randomly. It seems to
> >>
> >>> work relatively well, though it's
> >>>
> >> extremely slow.
> >>
> >> In addition, because the
> >>
> >>> algorithm fills the matrix one individual at a
> >>>
> >> time, I sometimes ran into the
> >>
> >>> problem of one individual which had
> >>>
> >> nowhere to go because "ressources" at the
> >>
> >>> site (i.e. the row sum) was
> >>>
> >> already full. In that case, this individual
> >>
> >>> disappears, i.e. goes
> >>>
> >> nowhere. This is a bit drastic and clearly sub-optimal,

> >>
> >>> but it's the
> >>>
> >> only way I could quickly think of yesterday to prevent the
> >>
> >>> algorithm
> >>>
> >> from getting stuck. The consequence is that often, the total number
> >>
> >>> of
> >>>
> >> individuals in the null matrix is a bit less than total number
> >>
> >>> of
> >>>
> >> individuals in the observed matrix.
> >>
> >> Etienne,
> >>
> >> This is the same problem as filling up a binary matrix: you get stuck before
> >> you can fill in all presences. The solution in vegan::commsimulator was
> >> "backtracking" method: when you get stuck, you remove some of the items
> >> (backtrack to an earlier stage of filling) and start again. This is even
> >> slower than initial filling, but it may be doable. The model application of
> >> backtracking goes back to Gotelli& Entsminger (Oecologia 129, 282--291;
> >> 2001) who had this for binary data, but the same idea is natural for
> >> quantitative null models, too. The vegan vignette discusses some details of
> >> implementation which are valid also for quantitative filling.
> >>
> >> I would like to repeat the serious warning that Carsten Dormann made: you
> >> better be sure that your generated null models really are random. Péter
> >> Sólymos developed some quantitative null models for vegan, but we found in
> >> our tests that they were not random at all, but the constraints we put
> >> produced peculiar kind of data with regular features although there was
> >> technically no problem. Therefore we removed code worth of huge amount of
> >> developing work as dangerous for use. I do think that also the r2dtable
> >> approach is more regular and less variable (lower variance) than the
> >> community data, and you very easily get misleadingly significant results
> >> when you compare your observed community against too regular null models.
> >> This is something analogous to using strict Poisson error in glm or gam when
> >> the data in fact are overdispersed to Poisson. Be very careful if you want
> >> to claim significant results based on quantitative null models. Would I be a
> >> referee or an editor for this kind of ms, I would be very skeptical ask for
> >> the proof of the validity of the null model.
> >>
> >> Cheers, Jari Oksanen
> >>
> >>
> >>> I don't see this as being a
> >>> real
> >>>
> >> problem with the large matrices I'll deal with though, but
> >>
> >>> definitely
> >>>
> >> something to be aware of.
> >>
> >> If you're curious, here's the simple
> >>
> >>> algorithm, as well as some code to
> >>>
> >> run a test below. I'd be very interested to
> >>
> >>> compare it to Vazquez's
> >>>

```
> >> algorithm!
> >>
> >> Thanks again
> >>
> >> Etienne
> >>
> >> ###
> >>
> >> nullabun<-
> >>
> >>> function(x){
> >>>
> >>     require(vegan)
> >>     nsites<- nrow(x)
> >>     nsp<- ncol(x)
> >>
> >>
> >>> rowsum<- rowSums(x)
> >>>
> >>     colsum<- colSums(x)
> >>     nullbin<- commsimulator(x,
> >>
> >>> method = "quasiswap")
> >>>
> >>     rownull<- rowSums(nullbin)
> >>     colnull<-
> >>
> >>> colSums(nullbin)
> >>>
> >>     rowsum<- rowsum - rownull
> >>     colsum<- colsum - colnull
> >>
> >>
> >>> ind<- rep(1:nsp, colsum)
> >>>
> >>     ress<- rep(1:nsites, rowsum)
> >>     total<-
> >>
> >>> sum(rowsum)
> >>>
> >>     selinds<- sample(1:total)
> >>        for (j in 1:total){
> >>
> >>
> >>> indsel<- ind[selinds[j]]
> >>>
> >>           sitepot<- which(nullbin[, indsel]>  0)
> >>
> >>
> >>> if (length(ress[ress %in% sitepot])>  0 ){
> >>>
> >>              sitesel<-
> >>
> >>> sample(ress[ress %in% sitepot], 1)
> >>>
> >>              ress<- ress[-which(ress ==
> >>
> >>> sitesel)[1] ]
> >>>
> >>              nullbin[sitesel, indsel]<- nullbin[sitesel, indsel]
> >>
> >>> + 1
> >>>
> >>            }
> >>        }
```

```
> >> return(nullbin)
> >> }
> >>
> >>
> >> ### now let's test the
> >>
> >>> function
> >>>
> >> # create a dummy abundance matrix
> >> m<-
> >>
> >>> matrix(c(1,3,2,0,3,1,0,2,1,0,2,1,0,0,1,2,0,3,0,0,0,1,4,3), 4,
> >>> 6,
> >>>
> >> byrow=TRUE)
> >>
> >> # generate a null matrix
> >> m.null<- nullabun(m)
> >>
> >> # compare
> >>
> >>> total number of individuals
> >>>
> >> sum(m) #30
> >> sum(m.null) #29: one individual got
> >>
> >>> "stuck" with no ressources...
> >>>
> >> # to generate more than one null matrix, say
> >>
> >>> 999
> >>>
> >> nulls<- replicate(n = 999, nullabun(m), simplify = F)
> >>
> >> # how many unique
> >>
> >>> null matrices?
> >>>
> >> length(unique(nulls) ) # I found 983 out of 999
> >>
> >> # how many
> >>
> >>> individuals in m?
> >>>
> >> sum(m) # there are 30
> >>
> >> # how bad is the problem of
> >>
> >>> individuals getting stuck with no ressources
> >>>
> >> in null matrices?
> >> sums<-
> >>
> >>> as.numeric(lapply(nulls, sum, simplify = T))
> >>>
> >> hist(sums) # the vast majority
> >>
> >>> had 29 or 30 individuals
> >>>
> >>
> >> Le jeudi 07 janvier 2010 à 10:34 +0100, Carsten
> >>
> >>> Dormann a écrit :
> >>> Hi Etienne,
> >>>
```

> >>> I'm afraid that swap.web cannot easily
> >>> accommodate this constraint.
> >>> Diego Vazquez has used an alternative approach
> >>> for this problem, but I
> >>> haven't seen code for it (it's briefly described in
> >>> his Oikos 2005
> >>> paper). While swap.web starts with a "too-full" matrix and
> >>> then
> >>> downsamples, Diego starts by allocating bottom-up. There it should be
> >>>
> >>> relatively easy to also constrain the row-constancy of connectance.
> >>>
> >>> I
> >>> can't promise, but I will hopefully have this algorithm by the end
> >>> of next
> >>> week (I'm teaching this stuff next week and this is one of the
> >>> assignments).
> >>> If so, I'll get it over to you.
> >>>
> >>> The problem that already arises with
> >>> swap.web for very sparse matrices
> >>> is that there are very few unique
> >>> combinations left after all these
> >>> constraints. This will be even worse for
> >>> your approach, and plant
> >>> community data are usually VERY sparse. Once you
> >>> have produced the
> >>> null models, you should assure yourself that there are
> >>> hundreds
> >>> (better thousands) of possible randomised matrices. Adding just
> >>> one
> >>> more constraint to your null model (that of column AND row constancy
> >>>
> >>> of 0s) will uniquely define the matrix! Referees may not pick it up,
> >>> but it
> >>> may give you trivial results.
> >>>
> >>> Best wishes,
> >>>
> >>> Carsten
> >>>
> >>>
> >>> Vázquez,
> >>> D. P., Melián, C. J., Williams, N. M., Blüthgen, N., Krasnov,
> >>> B. R., Poulin,
> >>> R., et al. (2007). Species abundance and asymmetric
> >>> interaction strength in
> >>> ecological networks. Oikos, 116, 1120-1127.
> >>>
> >>> On 06/01/2010 23:57, Etienne
> >>> Laliberté wrote:
> >>>
> >>>> Many thanks Carsten and Peter for your suggestions.
> >>>>
> >>>>
> >>>
> >>>> commsimulator indeed respects the two contraints I'm interested in, but>
> >>>> only allows for binary data.
> >>>>
> >>>> swap.web is *almost* what I need, but
> >>>>
> >>> only overall matrix fill is kept
> >>>
> >>>> constant, whereas I want zeros to move
> >>>>

> >>> only between rows, not between
> >>>
> >>>> both columns and rows. In others words, if
> >>>>
> >>> the initial data matrix had
> >>>
> >>>> three zeros for row 1, permuted matrices
> >>>>
> >>> should also have three zeros
> >>>
> >>>> for that row.
> >>>>
> >>>> I do not doubt that
> >>>>
> >>> Peter's suggestions are good, but I'm afraid they
> >>>
> >>>> seem a bit overly
> >>>>
> >>> complicated for my particular problem. All I'm after
> >>>
> >>>> is to create n
> >>>>
> >>> randomly-assembled matrices from an observed species
> >>>
> >>>> abundance matrix to
> >>>>
> >>> compare the observed functional diversity of the
> >>>
> >>>> sampling sites to a null
> >>>>
> >>> expectation. To be conservative, this requires
> >>>
> >>>> that I hold species
> >>>>
> >>> richness constant at each site, and keep row and
> >>>
> >>>> column marginals fixed.
> >>>>
> >>>
> >>>> Could swap.web or permatswap(..., method = "quasiswap") be easily
> >>>>
> >>>>
> >>> tweaked to accomodate this? The only difference really is that matrix
> >>>
> >>>> fill
> >>>>
> >>> should be kept constant *but also* be constrained within rows.
> >>>
> >>>> Thanks
> >>>>
> >>> again for your help.
> >>>
> >>>> Etienne
> >>>>
> >>>> Le 7 janvier 2010 08:17, Peter
> >>>>
> >>> Solymos<solymos at ualberta.ca> a écrit :
> >>>
> >>>>
> >>>>
> >>>>> Dear Etienne,
> >>>>>
> >>>>>
> >>>> You can try the Chris Hennig's prablus package which have a parametric
> >>>>

> >>>>>
> >>> bootstrap based null-model where clumpedness of occurrences or
> >>>
> >>>>>
> >>> abundances (this might allow continuous data, too) is estimated from
> >>>
> >>>>> the
> >>>>>
> >>> site-species matrix and used in the null-model generation. But
> >>>
> >>>>> here, the
> >>>>>
> >>> sum of the matrix will vary randomly.
> >>>
> >>>>> But if you have
> >>>>>
> >>> environmental covariates, you might try something more
> >>>
> >>>>> parametric. For
> >>>>>
> >>> example the simulate.rda or simulate.cca functions in
> >>>
> >>>>> the vegan package,
> >>>>>
> >>> or fit multivariate LM for nested models (i.e.
> >>>
> >>>>> intercept only, and with
> >>>>>
> >>> other covariates) and compare AIC's, or use
> >>>
> >>>>> the simulate.lm to get
> >>>>>
> >>> random numbers based on the fitted model. This
> >>>
> >>>>> way you can base you
> >>>>>
> >>> desired statistic on the simulated data sets, and
> >>>
> >>>>> you know explicitly
> >>>>>
> >>> what is the model (plus it is good for continuous
> >>>
> >>>>> data that you have).
> >>>>>
> >>> By using the null-model approach, you implicitly
> >>>
> >>>>> have a model by
> >>>>>
> >>> defining constraints for the permutations, and
> >>>
> >>>>> p-values are
> >>>>>
> >>> probabilities of the data given the constraints (null
> >>>
> >>>>> hypothesis), and
> >>>>>
> >>> not probability of the null hypothesis given the data
> >>>
> >>>>> (what people
> >>>>>
> >>> usually really want).
> >>>
> >>>>> Cheers,
> >>>>>
> >>>>> Peter

> >>>>>
> >>>>>
> >>>>>
> >>> Péter Sólymos
> >>>
> >>>>> Alberta Biodiversity Monitoring Institute
> >>>>> Department
> >>>>>
> >>> of Biological Sciences
> >>>
> >>>>> CW 405, Biological Sciences Bldg
> >>>>> University
> >>>>>
> >>> of Alberta
> >>>
> >>>>> Edmonton, Alberta, T6G 2E9, Canada
> >>>>> Phone:
> >>>>>
> >>> 780.492.8534
> >>>
> >>>>> Fax: 780.492.7635
> >>>>>
> >>>>>
> >>>>>
> >>>>> On Wed, Jan 6,
> >>>>>
> >>> 2010 at 2:18 AM, Carsten Dormann<carsten.dormann at ufz.de> wrote:
> >>>
> >>>>>
> >>>>>
> >>>
> >>>>>> Hi Etienne,
> >>>>>>
> >>>>>> the double constraint is observed by two
> >>>>>>
> >>> functions:
> >>>
> >>>>>> swap.web in package bipartite
> >>>>>>
> >>>>>>
> >>>>>>
> >>> and
> >>>
> >>>>>> commsimulator in vegan (at least in the r-forge
> >>>>>>
> >>> version)
> >>>
> >>>>>> Both build on the r2dtable approach, i.e. you have,
> >>>>>>
> >>> as you propose, to turn
> >>>
> >>>>>> the low values into higher-value integers.
> >>>>>>
> >>>>
> >>>>>
> >>>>>> The algorithm is described in the help to swap.web.
> >>>>>>
> >>>>>>
> >>>>
> >>>>>
> >>>>>> HTH,
> >>>>>>
> >>>>>> Carsten
> >>>>>>
> >>>>>>

> >>>>>>
> >>>>>>
> >>>>>>
> >>> On 06/01/2010 08:55, Etienne Laliberté wrote:
> >>>
> >>>>>>
> >>>>>>
> >>>>>>> Hi,
> >>>>>>>
> >>>>
> >>>>>>
> >>>>>>> Let's say I have measured plant biomass for a total of 5
> >>>>>>>
> >>> species from 3
> >>>
> >>>>>>> sites (i.e. plots), such that I end with the
> >>>>>>>
> >>> following data matrix
> >>>
> >>>>>>> mat<- matrix(c(0.35, 0.12, 0.61, 0,
> >>>>>>>
> >>> 0, 0.28, 0, 0.42, 0.31, 0.19, 0.82,
> >>>
> >>>>>>> 0, 0, 0, 0.25), 3, 5, byrow =
> >>>>>>>
> >>> T)
> >>>
> >>>>>>> dimnames(mat)<- list(c("site1", "site2", "site3"),
> >>>>>>>
> >>> c("sp1", "sp2",
> >>>
> >>>>>>> "sp3", "sp4", "sp5"))
> >>>>>>>
> >>>>>>> Data is
> >>>>>>>
> >>> therefore continuous. I want to generate n random community
> >>>
> >>>>>>> matrices
> >>>>>>>
> >>> which both respect the following constraints:
> >>>
> >>>>>>> 1) row and
> >>>>>>>
> >>> column totals are kept constant, such that "productivity" of
> >>>
> >>>>>>> each
> >>>>>>>
> >>> site is maintained, and that rare species at a "regional" level
> >>>
> >>>>>>> stay
> >>>>>>>
> >>> rare (and vice-versa).
> >>>
> >>>>>>> 2) number of species in each plot
> >>>>>>>
> >>> is kept constant, i.e. each row
> >>>
> >>>>>>> maintains the same number of zeros,
> >>>>>>>
> >>> though these zeros should not stay
> >>>
> >>>>>>> fixed.
> >>>>>>>
> >>>>>>> To
> >>>>>>>

> >>> deal with continuous data, my initial idea was to transform the
> >>>
> >>>>>>>
> >>> continuous data in mat to integer data by
> >>>
> >>>>>>>> mat2<-
> >>>>>>>
> >>> floor(mat * 100 / min(mat[mat>   0]))
> >>>
> >>>>>>>> where multiplying
> >>>>>>>
> >>> by 100 is only used to reduce the effect of rounding
> >>>
> >>>>>>>> to nearest
> >>>>>>>
> >>> integer (a bit arbitrary). In a way, shuffling mat could now
> >>>
> >>>>>>>> be seen
> >>>>>>>
> >>> as re-allocating "units of biomass" randomly to plots. However,
> >>>
> >>>>>>>
> >>> doing so results in a matrix with large number of "individuals" to
> >>>
> >>>>>>>
> >>> reshuffle, which can slow things down quite a bit. But this is only part
> >>>
> >>>>> of the problem.
> >>>>>
> >>>>>>>> My main problem has been to find an
> >>>>>>>
> >>> algorithm that can actually respect
> >>>
> >>>>>>>> constraints 1 and 2. Despite
> >>>>>>>
> >>> trying various R functions (r2dtable,
> >>>
> >>>>>>>> permatfull, etc), I have not
> >>>>>>>
> >>> yet been able to find one that can do
> >>>
> >>>>>>>> this.
> >>>>>>>
> >>>>>>>
> >>>>>>>
> >>> I've had some kind help from Peter Solymos who suggested that I try the
> >>>
> >>>>> aylmer package, and it's *almost* what I need, but the problem is that
> >>>>>
> >>>>>>>
> >>>>>>>> their algorithm does not allow for the zeros to move within the
> >>>>>>>
> >>> matrix;
> >>>
> >>>>>>>> they stay fixed. I want the number of zeros to stay constant
> >>>>>>>
> >>> within each
> >>>
> >>>>>>>> row, but I want them to move freely betweem columns.
> >>>>>>>
> >>>>>
> >>>>>>>
> >>>>>>>> Any help would be very much appreciated.
> >>>>>>>
> >>>>>>>>

> >>>>>>>
> >>> Thanks
> >>>
> >>>>>>>
> >>>>>>>
> >>>>>>>
> >>>>>> --
> >>>>>> Dr. Carsten
> >>>>>>
> >>> F. Dormann
> >>>
> >>>>>> Department of Computational Landscape Ecology
> >>>>>>
> >>>>>>
> >>> Helmholtz Centre for Environmental Research-UFZ
> >>>
> >>>>>> Permoserstr. 15
> >>>>>>
> >>>>>
> >>>> 04318 Leipzig
> >>>>
> >>>>>> Germany
> >>>>>>
> >>>>>> Tel: ++49(0)341 2351946
> >>>>>>
> >>>>>
> >>>> Fax: ++49(0)341 2351939
> >>>>
> >>>>>> Email: carsten.dormann at ufz.de
> >>>>>>
> >>>>>>
> >>> internet: http://www.ufz.de/index.php?de=4205
> >>>
> >>>>>>
> >>>>>>
> >>> _____
> >>>
> >>>>>> R-sig-ecology mailing
> >>>>>>
> >>> list
> >>>
> >>>>>> R-sig-ecology at r-project.org
> >>>>>>
> >>>>>>
> >>> https://stat.ethz.ch/mailman/listinfo/r-sig-ecology
> >>>
> >>>>>>
> >>>>>>
> >>>>>>
> >>>
> >>>>
> >>>>
> >>>>
> >>> --
> >>> Dr. Carsten F. Dormann
> >>> Department of
> >>> Computational Landscape Ecology
> >>> Helmholtz Centre for Environmental
> >>> Research-UFZ
> >>> Permoserstr. 15
> >>> 04318 Leipzig
> >>> Germany
> >>>
> >>> Tel: ++49(0)341
> >>> 2351946

> >>> Fax: ++49(0)341 2351939
> >>> Email: carsten.dormann at ufz.de
> >>> internet:
> >>> http://www.ufz.de/index.php?de=4205
> >>>
> >>
> >>
> >
> >
>

--
Etienne Laliberté
==========
School of Forestry
University of Canterbury
Private Bag 4800
Christchurch 8140, New Zealand
Phone: +64 3 366 7001 ext. 8365
Fax: +64 3 364 2124
www.elaliberte.info

- Previous message: [R-sig-eco] Job Opening: Quantitatively savvy biologist
- Next message: [R-sig-eco] null models with continuous abundance data
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the R-sig-ecology mailing list