On The Equitability Calculus RT/APO/07/8

August 29, 2008

J. Gergaud gergaud@n7.fr

1 Introduction

In ecology, we define the equitability or the evenness [1, 2, 3]. The objective of this report is to give mathematical results on the notion and to compute this index.

This report is organized as follows: we give the definition of the equitability and well pose the objective of the report in Section 2. In Section 3 we give and prove some mathematical results. Then, in Section 4, the MATLAB functions for computing the equitability are presented.

2 Definition and objective

We note

- S: the number of species;
- n_i: the number of individuals of the specie i;
- n: the total number of individuals, $n = \sum_{i=1}^{S} n_i$, namely the size of the sample ;
- p_i: the relative abundance of the specie i: $p_i = \frac{n_i}{n}$.

Definition 2.1. We define the equitability index or the equitability or the eveness index the quantities

$$J(p_1, \dots, p_S) = -\frac{\sum_{i=1}^{S} p_i \log(p_i)}{\log S},$$

where n and S are fixed.

Remark 2.2. We pose here $0 \log(0) = 0$.

Our target is, for a constant C fixed, to determine the values of the $(n_i)_i$ for which the equitability is equal to this constant C. In fact, we have to compute the set $J^{-1}(C)$. In practice we will have $n \leq 100$ et $S \leq 15$.

In fact the first question we have to solve is to know all the values that the function J can take. So we first consider the continuous case ($n = +\infty$), i.e. the case where the function J is defined on the simplex

$$P = \{p = (p_1, \dots, p_S) \in [0,1]^S | \sum_{i=1}^{S} p_i = 1\}.$$

Remark 2.3. The function $-k \sum_{i=1}^{S} p_i \log(p_i)$ defined on the simplex P is well know in physic and in the theory of information and is called the entropy **??**.

The Section 3 studies the mathematical properties of this continuous case, and then, in Section 4 we present the MATLAB functions:

- `equitcont(S,J,n)` which computes a solution of J(p)=S in the continuous case;
- `equit(n,S)` which computes the equitability function in the discrete case;
- `search_equit(NE,val)` which computes the $n_1 \leq n_2 \leq \ldots \leq n_S$ such that $J(n_1, \dots, n_S)$ is the nearest value of val.

3 Mathematical result

We consider in this Section that the function J is defined on the simplex P. The results we present here are well know (see for example [4] and [5]. The main result of this Section is the following proposition.

Proposition 3.1. The function J is onto the interval $[0, 1]$.

The figure 1 visualizes the function J when $S = 3$.

The property of the proposition 3.1 is important for the equitability index. In fact for an equitability index it is logical to have 0 if there is only one species and 1 if all the spcies have the same number of members.

For proving this proposition we 'll use two lemmas.

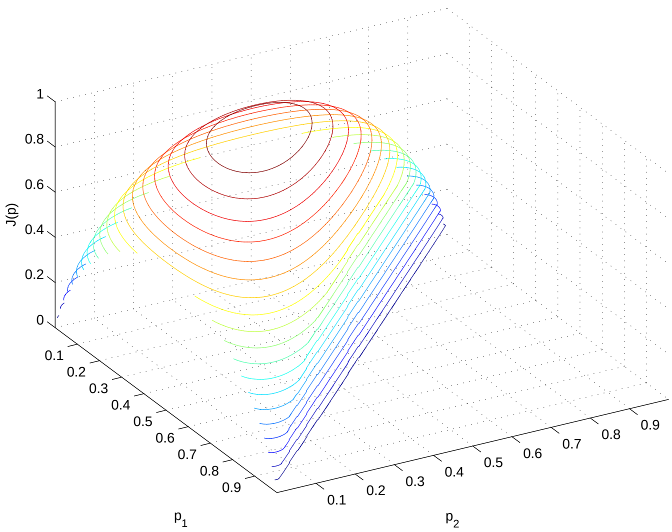

Figure 1: Function $J(p_1, p_2)$ ($p_3 = 1 - p_1 - p_2$) for $S = 3$

Lemma 3.2. The function J is on a compact interval $[a, b]$ of $\mathbf{R}$.

Proof
The simplex P is connected and J is continuous, thus $J(P)$ is a connected set of $\mathbf{R}$ (i.e. an interval). Moreover P is compact, so $J(P)$ is also compact. $\square$

Lemma 3.3. The nonlinear programming problem

$$(\mathcal{P}) \begin{cases} Max \quad J(p) \\ p \in P, \end{cases}$$

has a solution and only one $p^\star = (1/S, \ldots, 1/S)$.

Proof
The lemma 3.2 involves that the optimization problem $(\mathcal{P})$ has a solution that it is equivalent to the following problem $(\mathcal{Q}_\mathcal{S})$

$$(\mathcal{Q}_S) \begin{cases} Min \quad f(p) = \sum_{i=1}^{S} p_i \log(p_i) \\ p \in P. \end{cases}$$

We are going to prove the result by induction. For $S = 2$, we have $f(p_1, p_2) = p_1 \log(p_1) + (1 - p_1) \log(1 - p_1) = g(p_1)$ and the function g is strictly convex on $[0, 1]$ and $g'(p_1) = 0$ for $p_1 = 1/2$. Then, we immediately obtain the solution $p^\star = (1/2, 1/2)$. So the result is true for $S = 2$.

We suppose now that the result is true for S and we prove that it is always true for $S+1$. That is why we first study the optimization problem

$$(\mathcal{Q}'_{S+1}) \begin{cases} Min \quad f(p) = \sum_{i=1}^{S} p_i \log(p_i) \\ 0 < p_i < 1 \quad \text{for all} \quad i \\ \sum_{i=1}^{S+1} p_i = 1. \end{cases}$$

This optimization problem is a convex problem and the Khun et Tucker conditions are necessary and sufficient. The Lagrangien is here

$$L(p, \lambda) = \sum_{i=1}^{S+1} p_i \log(p_i) + \lambda(\sum_{i=1}^{S+1} p_i - 1).$$

In fact, we can take $\lambda_0 \neq 0$ because the derivative of the constraint is not 0. So to solve the problem $(\mathcal{Q}'_{S+1})$ is equivalent to solve

$$\begin{cases} \frac{\partial L}{\partial p_1} = \log(p_1) + 1 + \lambda & = & 0 \\ & \vdots & \\ \frac{\partial L}{\partial p_{S+1}} = \log(p_{S+1}) + 1 + \lambda & = & 0 \\ \sum_{i=1}^{S+1} p_i = 1. \end{cases}$$

We then immediately obtain that the solution of $(\mathcal{Q}'_{S+1})$ is $(1/(S+1); \ldots, 1/(S+1))$.

Now, if the solution $p^\star$ of the problem $(\mathcal{P})$ is not in the constraints of the problem $(\mathcal{Q_{S+\infty}}')$ then there exists an index i_0 for which $p^\star_{i_0} = 0$. In this case $p^\star$ is a solution of a problem $(\mathcal{Q}_S)$, for which by induction the solution is $(1/S, \ldots, 1/S)$. But

$$\sum_{i=1}^{S} (1/S) \log(1/S) = -\log(S),$$

and we immediately conclude.

$\square$

It is possible now to prove the proposition 3.1.

Proof
Because of the lemma 3.2, it is sufficient to demonstrate that $a = 0$ and $b = 1$. It is obvious that the function J is always non negative and that the function is null if and only if $p = (0, \ldots, 1, \ldots, 0)$, namely p is a vertex of the simplex P.

For the value of b, it is sufficient to apply the lemma 3.3. In fact the maximum value is obtained for $p^\star = (1/S, \ldots, 1/S)$. Thus

$$b = J(1/S, \ldots, 1/S) = 1$$

. $\square$

4 Matlab functions

4.1 Installation

For use the MATLAB functions you need the optimization toolbox of MATLAB.

To install the MATLAB functions:

1. download the file `equit.tar` from `www.enseeiht.fr/~gergaud/research`;
2. extract the file `equit.tar` (`tar xvf equit.tar` command for UNIX and LINUX system). All the MATLAB functions are now in the directory equit.

4.2 Continuous case

We describe in this Subsection all the MATLAB functions we need for computing a value of the set $J^{-1}(C)$ in the continuous case. And then we present an example of one execution.

Remark 4.1. In general $J^{-1}(C)$ is a smooth manifold of dimension $S-2$ (a loop for $S = 3$ 1). So it contains an infinite number of points.

The `equitcont` function computes an approximation of a point of $J^{-1}(C)$ by solving the following optimization problem.

$$(Pe)\left\{\begin{array}{l} Min \quad (\sum_{i=1}^{S} p_i \log(p_i) - C\log(S))^2 \\ 0 \leq p_i \leq 1 \quad \text{for all} \quad i \\ \sum_{i=1}^{S} p_i = 1. \end{array}\right.$$

It also furnishes an S-tuple $(n_1, \ldots, n_S)$ such that $(p_1, \ldots, p_S)$ is near $(p_1^\star, \ldots, p_S^\star)$, with $p_i = n_i/n$ and $n = \sum_{i=1}^{S} n_i = n$.

4.2.1 equitcont

```
% function [pstar,fval,nstar]=equitcont(S,C,n)
% Script to compute one element pstar  of the set J^{-1}(C)
% -----------------------------------------------------------
% Input parameters:
% S: number of species
% C: constant C
% n: sample size
% Output parameter:
% pstar: a point such that J(pstar) = C
% fval: (sum(p_i*ln(p_i)) +C*ln(S))^2
% nstar: =(nstar_1,...nstar_S) near n*pstar such that
% sum(nstar)=n
%============================================================
% The following optimization problem is solved
% Min (sum(p_i*ln(p_i)) +J*ln(S))^2
% 0<=p_i<=1
% sum(p_i)=1
%============================================================
% Author: Gergaud Joseph
% Date: may 2007
% University of Toulouse
% ENSEEIHT-IRIT-CNRS (UMR CNRS 5505)
% www.enseeiht.fr/~gergaud/research
%============================================================
%
% Initialisation
function [pstar,fval,nstar]=equitcont(S,C,n)
A=[]; b=[];
Aeq=ones(1,S); beq=1;
LB=zeros(S,1); LU=ones(S,1);
p0=[ones(S-1,1)*(1/(2*(S-1)));1/2];
options=optimset('Display','iter');
% Optimization
[pstar,fval,exitflag,output,lambda] = ...
fmincon(@f,p0,A,b,Aeq,beq,LB,LU,[],options,S,C);
% Compute nstar
nstar=round(pstar*n);
i=S;
if (sum(nstar)>n),
  while (sum(nstar)~=n),
    if (nstar(i)~=0),
       nstar(i)=nstar(i)-1;
    end;
    i=i-1;
  end;
else
  while (sum(nstar)~=n),
    if nstar(i)~=0,
      nstar(i)=nstar(i)+1;
    end;
    i=i-1;
  end;
end;
disp('f(n_1,...,n_S)')
disp(f(nstar/n,S,C))
```

4.3 f.m

```
% function fval=f(p,S,C)
% Cost function
% Input parameter
% p: value of p
% S: number of species
% C: constant C
%=================================
% Gergaud
% may 2007
% University of Toulouse
% ENSEEIHT-IRIT-INP (UMR CNRS 5505)
%
function fval=f(p,S,C)
fval=(sum(plnp(p))+C*log(S))^2;
```

4.4 plnp.m

```
% function f=plnp(p)
% vectoriel function which computes pln(p)
% Input parameter:
% p: vector of positive or null reals
% Output parameter:
% f: f_i = p_i*ln(p_i)
%         and 0 if p_i=0
%============================================
% Gergaud
% may 2007
% University of Toulouse
% ENSEEIHT-IRIT-INP (UMR CNRS 5505)
function f=plnp(p)
p(find(p==0))=eps;
f=p.*log(p);
a=eps*log(eps);
f(find(f==a))=0;
```

4.4.1 Example of an execution

```
>> [pstar,fval,nstar]=equitcont(4,0.8,30)
Warning: Large-scale (trust region) method does not currently solve this type of problem,
switching to medium-scale (line search).
> In /appli/matlab6r13/toolbox/optim/fmincon.m at line 213
  In /home/gergaud/Optim/Ouin/equitcont.m at line 22
```

```
                              max                     Directional   First-order
 Iter F-count        f(x)   constraint    Step-size    derivative   optimality Procedure
    1      12  0.000137399           0          0.5       0.00797        0.15
    2      18  4.44072e-05    1.11e-16            1      0.000241       0.0271
    3      24  4.43634e-08           0            1     -2.98e-06       0.0102  Hessian modified
    4      30  5.58095e-10           0            1      1.09e-08     4.21e-05  Hessian modified
Optimization terminated successfully:
 Magnitude of directional derivative in search direction
  less than 2*options.TolFun and maximum constraint violation
  is less than options.TolCon
Active Constraints:
     1

f(n_1,...n_S)
   1.1704e-05


pstar =

    0.1326
    0.1326
    0.1326
    0.6023


val =

   5.5810e-10


nstar =

     4
     4
     4
    18

>> diary
```

4.5 Discrete case

We are here in the discrete case, so the number of the species S and the size of the sample n are fixed. Thus the equitability can only takes a finite number of values. We give in this subsection the MATLAB function `equit(n,S)` which computes the mapping $(0 < n_1, \dots, n_S) \mapsto J(n_1/n, \dots, n_S/n)$ for all the possible values $(n_1 \leq \dots \leq n_S$ such that $\sum_{i=1}^{S} n_i = n$, and the function `search_equit(NE,C)` which computes for a constant `C` fixed two sets of integer numbers $n_1 \leq \dots \leq n_S$ and $n'_1 \leq \dots \leq n'_S$, which verify $\sum_i n_i = n$ and $\sum_{i=1}^{S} n'_i = n$, and such that $J(n_1/n, \dots, n_S/n) \leq C \leq J(n'_1/n, \dots, n'_S/n)$.

4.5.1 `equit.m`

This function computes for S and n fixed all the possible values $0 < n_1 \leq \ldots \leq n_S$ such that $\sum_i n_i = n$ and the value of the corresponding equitability. The result is in an array which contains in its last column the value of the equitability in an increase order.

```
% This function computes in an increase order the equitabilities
% for a n-sample and S species.
%
% function res=equit(n,S)
%
% Input parameter:
% n = sum of the ni;
% S = number of species;
% Output parameter
% res = each line of res containts the values of 1<=n1<=n2<=...<=nS and
%       the last column contains the value of the equitability
%       function at this point.
%       This array is such that the equitability is in an increase order.
%==================================================================
% Author: J. Gergaud
% Date: march 2007
% University of Toulouse
% INP-ENSEEIHT-IRIT (UMR CNRS 5505)
%==================================================================
function res=equit(n,S)
res = generation(1,n,S);       % generation computes by induction all the
%                                possible value of 0<n_1<=...<=n_S such that
%                                sum(n_1,...n_S)=n
if n<S,
  disp('bad input parameter n<S'),
  res=[];
else
  P = res/n;
  P = P.*log(P);
  % P(find(isnan(P)==1)) = 0;      % 0*log(0) -> 0, not used here because 0<n_1
  E = - sum(P,2)/log(S);
  res = [res E];
  [A,I] = sort(res(:,S+1));
  res = res(I,:);
end
```

4.6 generation.m

This function computes by induction all the possible values $n_1 \leq \ldots \leq n_S$ such that $\sum_{i=1}^{S} n_i = 1$.

```
% This function computes by induction all the numbers n1,...,nS such that
% 1) n0<=n1<=n2<=...<=nS
% 2) sum(n1,...,nS)=n
%
% function N = generation(n0,n,S)
%
% n0 = first value;
% n  = sum of the ni;
% S  = Number of species;
% N  = Matrix which contains all the possibilities of number n1<=...<=nS
%      such that sum(n1,...nS)=n.
% Remark: put n0=1 for the first call.
%==========================================================================
% Author: J. Gergaud
% Date: march 2007
% University of Toulouse
% INP-ENSEEIHT-IRIT (UMS CNRS 5505)
function N = generation(n0,n,S)
N=[];
if S==2,
  for n1=n0:n/2,
    n2 = n-n1;
    N = [N ; n1 n2];
  end;
else
  for n1=n0:n/S,
    r = generation(n1,n-n1,S-1);
    [li,col] = size(r);
 %   if (li~=0),
    N = [N ; ones(li,1)*n1 r];
  end;
end;
```

4.7 search_equit

This function computes the nearest values of the equitabilities from a fixed value.

```
% This function computes the nearest values of the equitability from the
% value c.
%
% function nie =search_equit(NE,C)
%
% Input parameter
% NE  = matrix of equitabitilies which have been generated
%       by the function equit;
% C = value from which we search the nearest value of the equitability.
% Output parameter
% nie   = [n11 ... n1S e1
%          n21 ... n2S e2] such that: e1 <= C < e2
%=========================================================================
% Author: J. Gergaud
% Date: march 2007
% University of Toulouse
% INP-ENSEEIHT-IRIT (UMR CNRS 5505)
%=========================================================================
function nie = search_equit(NE,C)
%
% test the value of C
if (C<0) | (C>1),
  disp('The value of C is not in [0,1]')
else
  if (NE(1,end) > C),
    nie = NE(1,:);
  else
    l = find(NE(:,end) <= C);
    if length(l) == size(NE,1),
    % Case max(NE(:,S+1) <= C <= 1
      if NE(l(end),end) == C,
    l = find(NE(:,end) == C);
    nie = NE(l,:);
  else
    l1 = find(NE(:,end) == NE(l(end),end));
        nie = NE(l1(:),:);
  end;
    else
      if NE(l(end),end) == C,
    l = find(NE(:,end) == C);
    nie = NE(l,:);
  else
    l1 = find(NE(:,end) == NE(l(end),end));
    l2 = find(NE(:,end) == NE(l(end)+1,end));
        nie = NE([l1(:) ; l2(:)],:);
  end;
    end;
  end;
end;
```

4.8 Examples of executions

Remark 4.2. before using the `search_equit.m` function it is necessary to compute the array which contains the equitability with the function `equit.m`.

```
NE=equit(20,3)

NE =

    1.0000    1.0000   18.0000    0.3590
    1.0000    2.0000   17.0000    0.4717
    1.0000    3.0000   16.0000    0.5579
    2.0000    2.0000   16.0000    0.5817
    1.0000    4.0000   15.0000    0.6257
    2.0000    3.0000   15.0000    0.6650
    1.0000    5.0000   14.0000    0.6791
    1.0000    6.0000   13.0000    0.7200
    2.0000    4.0000   14.0000    0.7298
    3.0000    3.0000   14.0000    0.7453
    1.0000    7.0000   12.0000    0.7498
    1.0000    8.0000   11.0000    0.7693
    1.0000    9.0000   10.0000    0.7789
    2.0000    5.0000   13.0000    0.7799
    3.0000    4.0000   13.0000    0.8069
    2.0000    6.0000   12.0000    0.8173
    2.0000    7.0000   11.0000    0.8433
    3.0000    5.0000   12.0000    0.8535
    2.0000    8.0000   10.0000    0.8587
    2.0000    9.0000    9.0000    0.8637
    4.0000    4.0000   12.0000    0.8650
    3.0000    6.0000   11.0000    0.8871
    4.0000    5.0000   11.0000    0.9078
    3.0000    7.0000   10.0000    0.9089
    3.0000    8.0000    9.0000    0.9197
    4.0000    6.0000   10.0000    0.9372
    5.0000    5.0000   10.0000    0.9464
    4.0000    7.0000    9.0000    0.9545
    4.0000    8.0000    8.0000    0.9602
    5.0000    6.0000    9.0000    0.9713
    5.0000    7.0000    8.0000    0.9835
    6.0000    6.0000    8.0000    0.9912
    6.0000    7.0000    7.0000    0.9977
```

```
>> search_equit(NE,0.8)

ans =

    2.0000    5.0000   13.0000    0.7799
    3.0000    4.0000   13.0000    0.8069

>>
```

4.9 Limits of the programs

We give in table 1 the CPU time for computing the equitability (function equit.m) and in table 2 the number of different values we obtain for the equitability. All the results have been obtained on a biprocessor (3Ghz for each processor)computer with 2Giga RAM under Linux system (Ubunto).

$S : n$	10	20	30	40	50	60	70
2	0.053923	0.000865	0.00026	0.000254	0.00031	0.000317	0.000379
3	0.003888	0.000725	0.001127	0.001648	0.002328	0.003243	0.004136
4	0.000489	0.001691	0.003803	0.007475	0.012949	0.019859	0.029417
5	0.000653	0.002796	0.008504	0.020432	0.042168	0.0782	0.13712
6	0.000673	0.003512	0.014376	0.041954	0.10271	0.2226	0.44159
7	0.000809	0.003981	0.01908	0.067326	0.19579	0.48734	1.0823
8	0.001161	0.003971	0.022721	0.092923	0.30659	0.85511	2.1351
9	0.001793	0.003516	0.023524	0.11054	0.41577	1.2947	3.5559
10	0.002606	0.003505	0.023169	0.11982	0.50835	1.7493	5.2538
11	0.003393	0.002798	0.021322	0.12248	0.56599	2.1341	6.9167
12	0.004024	0.002231	0.01884	0.11771	0.59915	2.4254	8.3718
13	0.004942	0.001784	0.016006	0.10885	0.59658	2.5756	9.5772
14	0.005655	0.001535	0.013947	0.099786	0.57128	2.653	10.4503
15	0.006103	0.001305	0.012114	0.086157	0.53734	2.6178	10.7505

Table 1: CPU time with respect to n and S for the function `equit.m`

$S:n$	10	20	30	40	50	60	70
2	5	10	15	20	25	30	35
3	8	33	75	133	208	300	408
4	9	64	206	478	920	1575	2484
5	7	84	377	1115	2611	5260	9542
6	5	90	532	1945	5427	12692	26207
7	3	82	618	2738	8946	23961	55748
8	2	70	638	3319	12450	37638	97539
9	1	54	598	3589	15224	51294	146520
10	1	42	530	3590	16928	62740	195491
11	0	30	445	3370	17475	70515	237489
12	0	22	366	3036	17084	74287	268079
13	0	15	290	2637	15988	74331	285373
14	0	11	229	2241	14499	71509	290071
15	0	7	176	1861	12801	66634	284054

Table 2: Number of different values of the equitability with respect to n and S for the function `equit.m`

Bibliography

[1] A.E. Magurran. Measuring Biological diversity. Blackwed, 2004.

[2] E.C. Pielou. An introduction to mathematical ecology. Wiley-interscience, 1969.

[3] E.C. Pietou. Ecological diversity. Wiley-interscience, 1975.

[4] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol. 27, pp. 379–423 and 623–656, July, October, 1948.

[5] D.E. Catlin. Estimation, Control, and the Discrete Kalman Filter. Springer-Verlag, Applied Mathematical Sciences series, number 71, 1989.