DSC 2003 Working Papers
(Draft Versions)
http://www.ci.tuwien.ac.at/Conferences/DSC-2003/

Fractals and Statistics: an R package called `ifs`

Stefano Maria Iacus, Davide La Torre
Department of Economics
University of Milan
Via Conservatorio, 7
I-20122 Milan - Italy
stefano.iacus@unimi.it
davide.latorre@unimi.it

Iterated Function Systems (IFSs) were born in mid eighties (see Hutchinson, 1981 and Barnsley and Demko, 1985) as applications of the theory of discrete dynamical systems and as useful tools to build fractals and other similar sets. Some possible applications of IFSs can be found in image processing theory (see e.g. the recent work by Forte and Vrscay, 1994), in the theory of stochastic growth models and in the theory of random dynamical systems. In particular they come to great popularity when they have been proposed as the 'definitive' method of images compression. We do not discuss such aspect here but the reader can refer, for example, to http://www.faqs.org/faqs/compression-faq/.

We propose to apply (affine) IFSs to statistics. In particular, we are concerned with the problem of distribution function estimation and related quantities like the characteristic and density function.

The idea behind the IFSs is the following. Suppose one wants to approximate an object f, i.e. a function, that is a point in some complete metric space (E, d). The aim is to construct a contractive operator $T : E \to E$ with a unique fixed point $\tilde{f}$, i.e. $T\tilde{f} = \tilde{f}$, in such a way that $d(f, \tilde{f})$ is minimum. In our case E will be the space of distribution functions on a compact set $[a, b]$ and d is the sup-norm distance. Alternatively, one can consider to start from the space of measures on $[a, b]$ and use the Hutchinson metric (for details see Iacus and La Torre, 2002). From now on, we will consider the compact set $[a, b] = [0, 1]$ without lost of generality. We introduce

the IFS operator as follows: for any distribution function $G \in E$

$$TG(x) = \sum_{i=1}^{N} p_i G\left(w_i^{-1}(x)\right)$$

where w_i are affine maps ($w_i(x) = a_i + s_i \cdot x$), and w_i^{-1}, the inverse function of w_i, are increasing and continuous. The p_i are the IFS coefficients that are essentially a probability distribution, i.e. $\sum_i p_i = 1$ and $p_i \geq 0$. Given a target distribution function F, w_i and p_i depend on F. The number of coefficients and maps N depend on the quality of the approximation one desires, in the sense that it is always possible to choose $N = N_\varepsilon$ such that for the fixed point $\tilde{F}$ of T one has $d(\tilde{F}, F) < \varepsilon$.

In the IFS theory it is always a problem to choose the p_i and the maps w_i. For a fixed target F (i.e. an image to compress) usually, a set of maps w_i is chosen and the p_i are determined as solutions of some minimization procedure. In our case the target F is not known, thus the problem is to find a method to express the maps w_i and/or the coefficients p_i of T in terms of the sample data, say $(x_1, x_2, \ldots, x_n)$. Two different approaches are available at present: for a fixed number N, choose the so-called wavelets maps (see below) and then use the sample moments $\mathbf{m} = (m_1, \ldots, m_M)$, $M > N$, to build a quadratic form

$$\mathbf{p}^t Q \mathbf{p} + \mathbf{b}^t \mathbf{p} + c$$

where $Q = Q(\mathbf{m})$, $b = b(\mathbf{m})$ and $c = c(\mathbf{m})$. The solution of the minimization problem for Q is the vector $\mathbf{p}$ of the p_i's. This quadratic form measures the distance between the moments of the IFS and the vector of sample moments $\mathbf{m}$. Two sets of wavelet maps are of interest (see Iacus and La Torre, 2002). One of these sets is constructed as follows: let i^* be such that $N = \sum_{i=1}^{i^*} 2^i$ and define

$$\mathcal{W}_1 = \{w_1 = h_{11}, w_2 = h_{12}, w_3 = h_{21}, \ldots, w_6 = h_{24}, w_7 = h_{31}, \ldots, w_N = h_{i^* 2^{i^*}}\}$$

with

$$h_{ij} = \frac{x + j - 1}{2^i}, \quad i = 1, 2, \ldots, i^* \quad j = 1, 2, \ldots, 2^i \,.$$

In this approach, the maps are always fixed *a priori*. The second approach consists in using the so called *quantile* maps defined as follows

$$\mathcal{W}_q = \{w_i(x) = (\hat{q}_{i+1} - \hat{q}_i)x + \hat{q}_i, i = 1, 2, \ldots, N\} \,.$$

where $\hat{q}_i$ are the sample quantiles of order $1/N$ and the coefficients p_i are all equal to $1/N$. In this case, the p_i are fixed *a priori*.

Affine IFSs are very simple and tractable objects. In fact, if T is the IFS with some fixed point $\tilde{F}$ then the Fourier transform of T has a fixed point which is the Fourier transform of $\tilde{F}$. Thus, for an affine IFS its Fourier transform is itself an IFS that can be written as

$$B\phi(t) = \sum_{k=1}^{N} p_k e^{-ita_k} \phi(s_k t)$$

where now B is the operator on the space of characteristic functions. So, once the p_i and w_i are available, one has at the same time a distribution function and a Fourier transform estimator. Thus, using classical Fourier analysis, one can deduce a density function estimator (assuming that this exists) such as

$$\hat{f}(x) = \frac{1}{2\pi} \sum_{k=-m}^{+m} \tilde{\phi}(k) e^{ikx} \tag{1}$$

where $\tilde{\phi}$ is the fixed point of B.

Properties of the IFS estimator and perspective usage

Figure 1 shows the fractal nature of the IFS: one can notice that the IFS is built as rescaled copies of itself.

IFS estimators of both approaches (quantile and wavelet) have good properties for small sample size as they are in general smoother than the empirical distribution function (e.d.f). Asymptotically they are equivalent to the e.d.f and it is possible to establish a Glivenko-Cantelli type theorem and law of iterated logarithm results (see Iacus and La Torre, 2002).

The self-similarity of the IFS estimator allows very interesting extensions in fields where the e.d.f. it is not usually an appropriate estimator. Consider for example to have a distribution function on $[a, b]$ where data are observed after a censoring on both sides, i.e. we can only observe data on a subset $[c, d] \subset [a, b]$. The IFS is still able to reconstruct a distribution function on the whole $[a, b]$ by copying and rescaling itself on the intervals $[a, c]$ and $[d, b]$. This arbitrary approximation will be better, in many cases, then the one based on the e.d.f. which returns 0 on $[a, c]$ and 1 on $[d, b]$.

To use IFS in practice, once the w_i and p_i are given, to obtain the estimate it is only necessary to iterate the functional starting from any point of the space. In particular, for the distribution functions one can start from the uniform distribution on $[a, b]$ and iterate till convergence. It appears that in general the convergence is rather fast. Indeed, 3 to 5 iterations are sufficient.

In Figure 2 an application to Old Faithful gayser data is presented. The IFS density estimator is plotted against the usual kernel one. This example is presented to show the ability of the IFS to discriminate the two subpopulations.

What's inside the ifs package?

The `ifs` package for R is currently available on `CRAN`. Apart from the R interface, the internal source C code is easily portable to other languages. Anyway, once you have loaded the package (`library(ifs)`) there is very few to get the IFS estimate. Suppose that F is the unknown distribution function with density f assuming that it exists. The very first thing you can do is to run the examples. The main function of the package is

```
ifs(x, p, s, a, k = 5)
```

which accepts in input the point x where to evaluate the estimator of $F(x)$, `p` is the vector of coefficients, and `s` and `a` are the vectors containing the coefficients of the affine maps $w_i(x) = a_i + s_i \cdot x$. The number of iterations is `k` by default set to 5. As you see, this function doesn't know about estimation, it simply generates the fixed point of an IFS. This means that maps and coefficient have to be generated from the data by using one of the two approaches presented above. The function `ifs` starts iterating from the uniform distribution. If you don't trust us about the speed of convergence and the unicity of the fixed point, you can provide your own initial distribution function using the `ifs.flex(x, p, s, a, k = 5, f = NULL)` function. It is the same as `ifs` where the argument `f` is a user defined distribution function. To setup the parameters `p`, `a` and `s` one can use the function

```
ifs.setup(y, maps = "quantile", qtl)
```

where `y` is the vector containing the sample data and `maps` is the switch that can assume three values: `quantile`, `wl1` and `wl2`. The last two are for the wavelet IFS. The argument `qtl` is only considered by the function if you do not provide the vector `y` because it is intended to work directly with the quantiles provided by the user in the vector `qtl`. It is ignored if `maps` is not set to `quantile`.

The `ifs.setup` returns a list of vectors that are the vector of empirical moments `m`, the coefficients `a` and `s` and the number of maps `n`. You can pass these parameters as inputs to `ifs`. A more speedy way to obtain the IFS estimator is the function `IFS(y, k = 5, q = 0.5, f = NULL, n = 512, maps = c("quantile", "wl1", "wl2"))`. In this function `y` is the vector of sample observations, `k` is the number of iterations, `n` is the number of points where the estimator is evaluated and `q` is the proportion of quantiles you want to use, where the number of quantiles used will be `length(y) * q`. This function returns a list of the x and y coordinates that can be used to plot the estimator. For example try: `y<-rbeta(50,.5,.1); plot(IFS(y))`.

If you want to use the wavelet IFS estimator you have firstly to build the quadratic form and then minimize it. Use the function `setQF(m, s, a, n = 10)` to build the quadratic form. Here `m` is the vector of moments, `s` and `a` are as above and `n` is the number of coefficient to use. This is different from above because `s` and `a` can have length bigger than `n`. The problem is that the dimension of the quadratic form depends on this `n` and consequently the minimization problem for it. In output this function returns a list of two objects: the matrix `Q` and the vector `b` of the quadratic form $x'Qx + b'x$. Once these are available you can use `ifs.cf(Q,b)` to obtain the parameters p_i. This function is rather far from being optimized. The problem is that $x'Qx + b'x$ has to be minimized over the simplex $\sum_i x_i = 1$ and $x_i \geq 0$. In fact, the `quadprog` package cannot be used as `Q` is not positive definite. Thus, inside `ifs.cf` we use the constrained version of `optim` with `L-BFGS-B` method and we minimize the function

```
fr <- function(x)  t(x) %*% Q %*% x + b %*% x + (1-sum(x))^2
```

i.e. we use a penalization approach. This guarantees, due to the constraints, that the minimum of `fr` coincides with the minimum of the quadratic form when `sum(x)==1` is `TRUE` (the converse does not hold in general).

The last set of functions concerns the Fourier transform of the IFS and its inverses (both density function and distribution function). The function `ifs.FT` evaluates the Fourier transform at some point x. The usual parameters should be passed as arguments. The function `ifs.setup.FT(m, p, s, a, k = 2, cutoff)` is used to build the coefficients of the anti Fourier transform. The parameters `p`, `s`, `a` and `k` are as above with `m` is the number of coefficients to calculate. The `cutoff` parameter is used to determine the number of significant coefficients to use in the anti Fourier transform (e.g. in equation (1)). If you do not specify it the cutoff is set to $2/(n+1)$, where n is the sample size. Given the cutoff, then the following rule of thumb is used:

"if $\left|\tilde{\phi}(j+1)\right|^2$ and $\left|\tilde{\phi}(j+2)\right|^2 <$ `cutoff` *then use the first* j *coefficients"*.

This rule has been applied to the data in Figure 2.

The function `ifs.setup.FT` returns a vector of Fourier coefficients `b` and the scalar `nterms` that is the number of significant coefficients to use according to the above rule. The functions `ifs.pf.FT(x,b,nterms)` and `ifs.df.FT(x,b,nterms)` evaluate respectively the probability distribution function and the density function at point `x` given the Fourier coefficients `b` using `nterms` coefficients. The last two functions: `IFS.pf.FT` and `IFS.df.FT` are the analogues of `IFS`. Given a vector of sample data `y`, they return a list of `n` coordinates (x, y) that can be used to plot the estimated distribution function and density curves. The number of iterations is `k` and the maps are chosen according to the argument `maps`.

References

- Barnsley, M.F. and Demko, S. (1985), Iterated function systems and the global construction of fractals, *Proc. Roy. Soc. London, Ser A*, **399**, 243-275.
- Forte, B. and Vrscay, E.R. (1994), Solving the inverse problem for function/image approximation using iterated function systems, I. Theoretical basis, *Fractal*, **2**, 3, 325-334.
- Hutchinson, J. (1981), Fractals and self-similarity, *Indiana Univ. J. Math.*, **30**, 5, 713-747.
- Iacus, S.M. and La Torre, D. (2002), On fractal distribution function estimation and applications, *submitted*, available in the "inst" directory of the ifs package source archive.

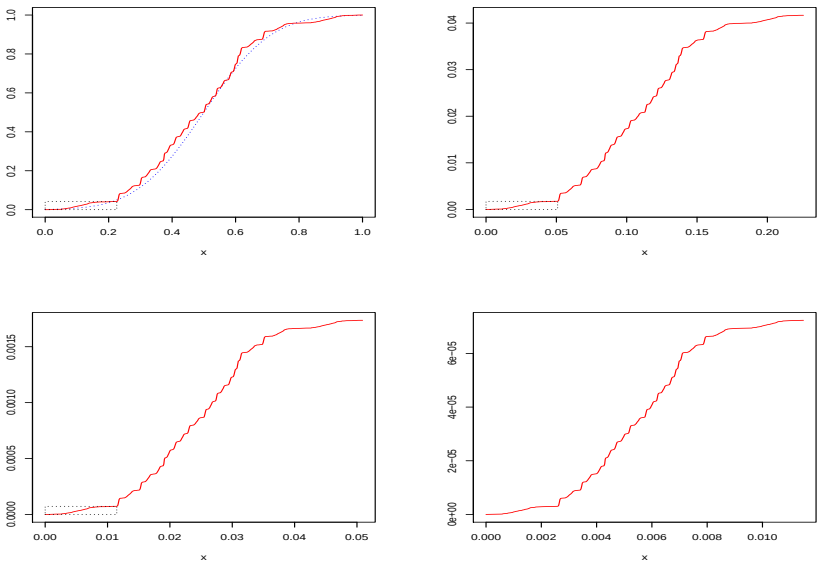

Figure 1: The fractal nature of the (quantile) IFS distribution function estimator. The dotted line is the underlying truncated Gaussian distribution. The dotted rectangle is to represent the area zoomed in the next plot (left to right, up to down). The dotted boxes are in the order: $[0, \hat{q}_2] \times [0, \hat{q}_2]$, $[0, \hat{q}_2^2] \times [0, \hat{q}_2^2]$ and $[0, \hat{q}_2^3] \times [0, \hat{q}_2^3]$.

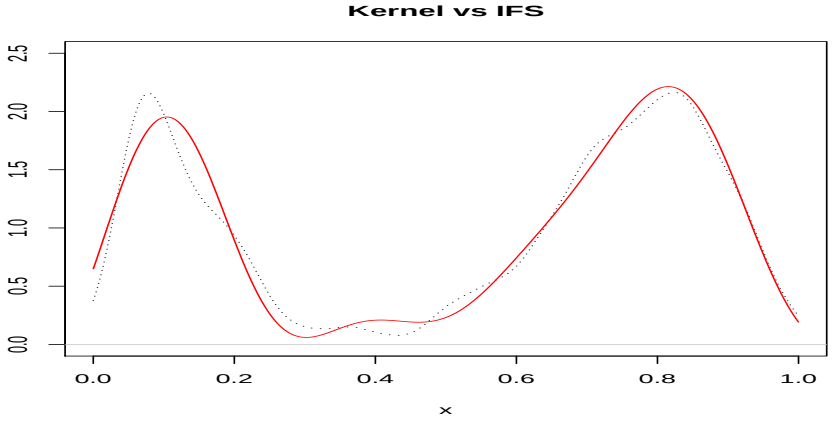

Figure 2: Old Faithful gayser data rescaled on [0,1]. Dotted line is the kernel estimator (bw=0.03, kernel=Gaussian), solid line is the IFS-Fourier expansion estimator (iterated 2 times, 26 Fourier coefficients).