Landscape Ecology vol. 5 no. 1 pp 39-52 (1990)
SPB Academic Publishing bv, The Hague

Home range: A fractal approach

Craig Loehle
Environmental Sciences Section, Savannah River Laboratory, 773-42A, Westinghouse Savannah River Company, Aiken, SC 29808-0001

Keywords: home range, fractal, telemetry, foraging, behavior, spatial models

Abstract

Most current methods for describing animal home range assume that it may be represented as a Euclidean type shape such as a bell shaped curve or a closed polygon. Landscape ecology has increasingly shown that ecological objects are more often highly fragmented and irregular. A fractal approach to description of animal home range was thus developed. For each point where the animal was observed, a circle centered on this point was first laid down to represent the area searched for prey by the animal during a short time interval. In this way the behavior of the animal and differences between species can be represented. Next, a fine grid is laid over the map and the height of each grid square computed by the number of circles that overlap that square. Then, the fractal dimension of the resultant 3-D surface is calculated at several scales. From an analysis of data from a hawk, the existence of perching behavior can be inferred, as well as the observation that at coarse scales the hawk behavior is self-similar and resembles a random walk. The home range thus analyzed in no way resembles a closed figure such as a polygon because it is highly fragmented. Further analysis showed that the fractal measures are relatively insensitive to sample size and to measurement error. Code is included for performing the analyses.

Introduction

A great deal of work has been done on home range quantification and description. Most current methods, however, have fundamental difficulties. A completely new conception of the home range is developed here, based on fractal geometry, that is free of shape assumptions and that incorporates sample error.

Estimation of home range (territory) size and shape is useful for studies of territoriality, animal density, competition, and energetics (Harestad and Bunnel 1979, McNab 1963, Sanderson 1966, Schoener 1968, Swihart *et al.* 1988, Turner *et al.* 1969). A variety of methods have been developed for estimating home range, based on the probabilistic notion of the utilization distribution (UD), including the minimum convex polygon method (Southwood 1966), the bivariate normal method (Jennrich and Turner 1969, Samuel and Garton 1985), the Fourier transform method (Anderson 1982) and others (Dunn 1978, 1979, Dunn and Gipson 1977, Neu *et al.* 1974, Schoener 1981, Trevor-Deutsch and Hackett 1980, Turner *et al.* 1969, Van Winkle *et al.* 1973, Van Winkle 1975, Voigt and Tinline 1980). The widely divergent areas estimated by these different methods and the severe problems associated with sample size and irregular shape are documented by Anderson (1982), Ford and Krumme (1979), Jennrich and Turner (1969), and Schoener (1981). The minimum convex polygon method is very sensitive to sample size and outliers and can-

not deal with fragmented home ranges. Bivariate normal and similar measures make restrictive assumptions about shape and again are sensitive to outliers and cannot handle fragmented home ranges. The Fourier transform method (Anderson 1982) and Map index (Ford and Krumme 1979) have the advantage of providing (almost) assumption-free pictures of the home range, including irregularly shaped or fragmented ones. The disadvantages of these two methods are that 1) the picture drawn is scale-dependent (a smooth shape examined more closely may have holes) but the methods give no criteria for choosing a scale, and 2) the pictures of different home ranges must be compared by eye since shape is not described numerically. My objective is to use these two approaches as a starting point for a new method based on fractal analysis that provides numerical (as opposed to pictorial) descriptors of home range size, shape, and concentration, and incorporates the effect of measurement error.

Redefining home range

It is possible to redefine home range so that the above problems are largely overcome. Consider an animal moving about as it forages. At any given location it does not merely occupy a point, but activity searches the area within sight (hearing, smell) for food. This search radius may be small, as for a rodent, or huge, as for a hawk. While areas close by are searched more thoroughly than those somewhat farther, consider a circle to be representative of the nearby area searched over a short time interval. Food located at a farther location leads to movement to reach that location. The search radius is therefore that area within easy strike or bite of the forager. Note that all of the analysis that follows is also valid for point data (radius = 0), as will be shown.

Given a collection of sequential observations of animal location, we can draw a circle around each point representing the search area. This gives a collection of overlapping circles. To determine the utilization distribution (UD), we impose a grid over the overlapping circles. The frequency (height) within each square is the numer of circles that overlap into that square. The height of the UD now signifies intensity of foraging in a location, taking into account foraging efficiency. For the UD thus defined, we wish to know the shape and compactness or irregularity of it. A home range that is very compact may indicate use of a central den or the presence of distinct territorial boundaries. Very irregular home ranges or even ones with no central tendency may occur when an animal has no home but wanders or is even blown about by the wind. I take a specifically scale-dependent, fractal approach to quantifying the irregularity of the UD. This allows calculation of d for both the path taken by the animal (a tortuosity factor) as has been done by Dicke and Burrough (1988), and for the fragmentation of the patches used by the animal.

Fractals are coming into increased use in ecology (Krummel *et al.* 1987; Loehle 1983; Milne 1988; Morse *et al.* 1985; Palmer 1988; Phillips 1985; Tatsumi *et al.* 1989; Turner *et al.* 1989) as well as in other fields (Barnsley 1988; Burrough 1986; Håkanson 1978; Lovejoy 1987; Mandelbrot 1983; Pietgen and Saupe 1988). In ecology they are particularly useful for analyzing spatial patterns of landscapes, vegetation types, animal distributions, etc. Fractals enable one to quantify previously intuitive concepts such as "patch," "grain," "roughness," and "scale". One factor inhibiting their more rapid diffusion is that fractal analysis is computationally intensive – that is, algorithmic rather than analytic tools must be used. The appendices thus present a simple algorithm in the form of a FORTRAN and a Pascal subroutine for calculating d,. The algorithm is particularly suitable for landscape data.

There are a number of ways to quantify fractal dimensions. Some are appropriate for mathematical sets such as Mandelbrot sets or Cantor sets. Our concern here, however, is with the type of empirical spatial data often found in ecology, for which infinite resolution is not possible. Such data is typically represented by a map. Values on the map may be vegetation types, soil types, pollution levels, or animal herds. The fractal dimension for such a map tells us something about degree of fragmentation (heterogeneity) and pattern of fragmentation with scale.

Many studies have used transect data to quantify d_f because it is relatively easy to collect and analyze

(e.g., Burrough 1985; Lam 1990; Palmer 1988; Phillips 1985). Extrapolation of transect data to whole maps is problematical, however, and use of all the data contained in a map would be preferable. A commonly used method that does this is the **area-perimeter** ratio (e.g., Krummel ***et al.*** 1987; Turner ***et al.*** 1989). For a given vegetation type (for example) as one successively reclassifies the grid squares on a map at larger scales the area-perimeter ratio changes as

$$A = kp^{2/d}$$

where A is area of the type, P is perimeter, and d is the dimensionality. A single estimate of d using this method implicitly assumes self-similarity. Since d may change with scale for landscape data **(Burrough 1986),** self similarity cannot be assumed a priori. Thus the grid squares must be reclassified at various scales and d recomputed. This can be easily done by a GIS system, but not everyone has access to such a system or wishes to learn the mechanisms of their use for a single application. Furthermore, area perimeter ratios can not be used for 3-D maps such as elevation maps.

A method which may be used with transect data, flat maps, and 3-D maps is box-counting. **Box-**counting uses boxes (cubes) at different scales to assess the pattern of occupation of space by the object of interest. Several authors have used box-counting for environmental data (e.g., **Håkanson** 1978; Morse ***et al.*** 1985) but the analyses have been done by hand – e.g., by counting grid squares on a transparency laid over a map. **Lovejoy** ***et al.*** (1987) used their radar apparatus to collect 3-D weather data at different scales. Tatsumi ***et al.*** (1989) used an image processor to process photographs of root systems. The image processor used pixels of different sizes and created a block image of the root system from which d_f was computed using 2-D box-counting. This technique is limited to 2-D images. All of these box-counting examples either require special hardware or were done by hand. The methods presented below overcome these limitations.

Consider a square map in which every pixel is of type X (a vegetation type, or whatever). At the level of one division along the X-axis (the whole map is covered by one box), there is one square unit containing X. When the X-axis is subdivided into 2, there are 4 squares of type X (Fig. 1). In general

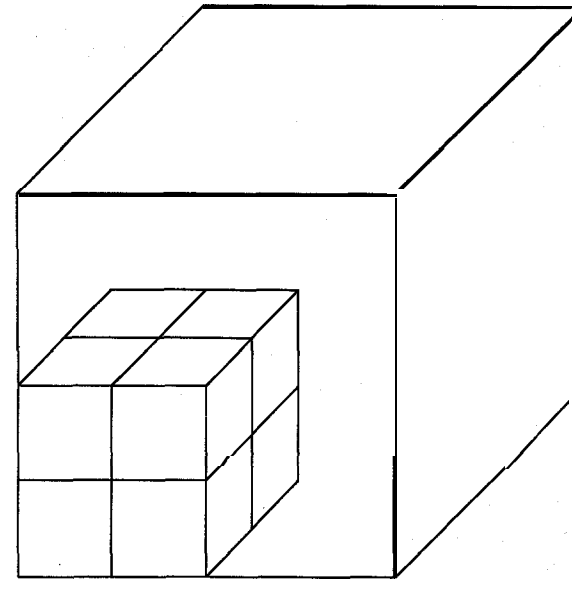

Fig. 1. Nesting convention for box counting.

$$\text{boxes} = t^d$$

where t is tick marks dividing up the axes. The dimension d of the type X is the power oft. In this example d = 2, X fills the plane, and X is not fractal. Consider a second case, type Y in which Y is a narrower linear feature. In this case d = 1. If the map is a single point or empty, d = 0. Regions of the map that occupy space (have a height) but are fragmented will have $1 < d < 3$ at various scales as measured by cubes occupied at different scales. Those that are fragmented into tiny widely scattered fragments will have $0 < d < 1$. In these last two cases d is fractal. It should be emphasized that this single algorithm can find d for trail data (e.g., an animal trail or river), transect data, scattered point data, patch-type data, and elevation-type (3-D) data, thus obviating the need to use several different (possibly incommensurate) methods.

The box counting algorithm is here implemented in both FORTRAN and Pascal versions. It calculates d directly at various scales. The algorithm uses nested boxes of increasing size to find d. A **simplis-**

tic approach would be to just imbed the data set in a 3-D x-y-z- coordinate system. Computation for such an approach is trivial. However, memory requirements quickly can become outrageous. For example, a 2048 x 2048 x 2048 pixel 3-D map of type BYTE (one character of storage at each location) requires 8 Gigabytes of memory. A different approach is thus needed. I therefore represent the elevation (or concentration, etc.) at each (x,y) location as an integer value. By thus requiring single-valued functions, memory requirements are cut to $2x^{2/3}$ where x is number of bytes required by an (x,y,z) representation. Nesting the computations also allows each data point to only be accessed (computed) once, which increases speed considerably. The animal location data are loaded into RASTER as follows. Given a set of observed animal locations and the search radius size, a grid is laid over the map. Each grid point is checked to see if it is in at least one of the circles (see Appendix III for methods). The area A is calculated as the number of grid points times the area each grid represents. This process is repeated with successively finer grids until the area converges to some criterion (say .5% difference between successive estimates). This grid is the finest scale used for the fractal calculations.

Note that as search radius r goes to zero, one is back to working with point data, but fractal measures are still appropriate. The pattern and scale of patchiness can still be calculated as illustrated below, as long as the grid is fine enough that most grid squares only contain one point (a height of 1) at the finest scale. In such a case d_f at the finest scale approaches zero unless many points are piled right on top of each other.

Another useful measure is concentration which separates out the extent to which the search circles overlap from the extent of fragmentation, which are somewhat confounded in d_f. If we define concentration C as relative to our basic search radius, then C describes how much the animal moves around relative to how large an area it searches in a short interval. For example, a raptor could have several perching trees some meters apart, but since it can search such a large area in any given interval, there is a large amount of overlap between areas searched from the several perches. Thus C is given by

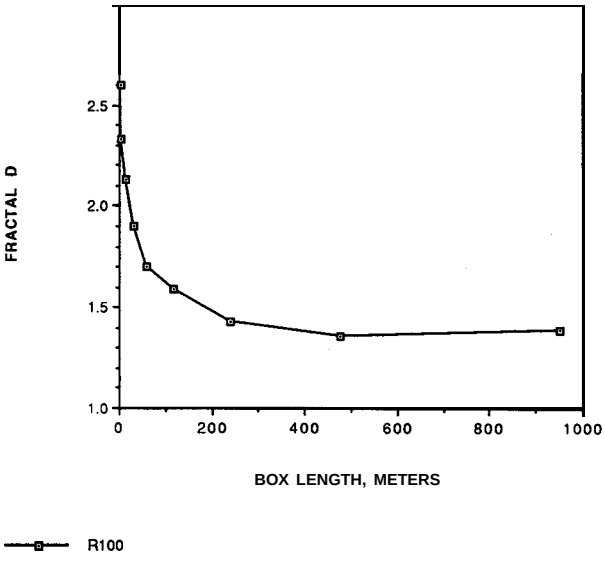

Fig. 2. d_f as a function of scale for hawk data, n = 123, r = 100.

$$C = \frac{A}{n\pi r^2}$$

where A is the area of the home range (2-D projection of the UD) as calculated above, n is the total number of observations, and r is the search radius. C = 1 for a set of completely non-overlapping circles and approaches 0 for a stationary predator for n large. For point data (r = 0) C is undefined.

An example

The methods discussed above are illustrated with real data for a radio-collared hawk (observed 4 km west of Newcomb, N.Y.) provided by Paul F. Steblein. Hawk 5013 was used, with 123 observations made at less than 0.5 km distance. Measurement error was estimated at a few meters. A search radius of 100 meters was chosen as a first approximation to the area within which a hawk could successfully catch prey. The home range analysis for this hawk (Fig. 2) shows a very well-behaved fractal pattern. Below 200 meters box size, d_f rises smoothly from 1.4 to 2.6 at successively smaller scales. At these scales we are quantifying the pattern of overlapping

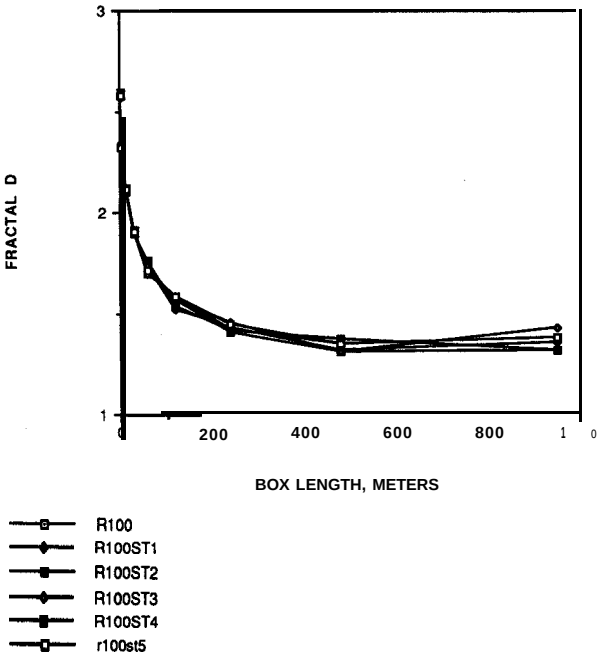

Fig. 3. d_f as a function of scale for data in Figure 2 with stochastic simulation of measurement error effects.

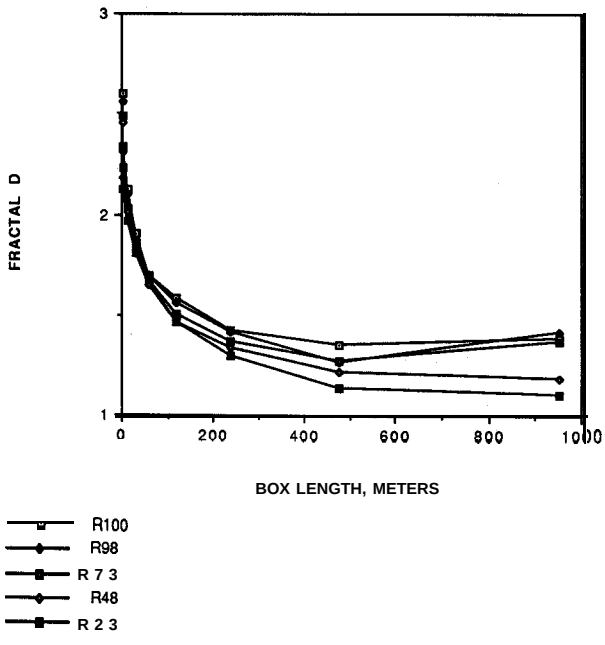

Fig. 4. d_f as a function of scale for sample sizes from 23 to 123.

l-unit high circles resulting from time spent on and near preferred perches. The values from 1.4 to 2.6 indicate considerable relief and fragmentation (irregularity). The concentration C = .42, indicating considerable overlap in the areas searched (largely due to perching). Above 200 meters box size, we see the curve flatten out with d_f between 1.4 and 1.5. At these scales vertical relief due to the degree of overlap in this data is not evident, so the UD appears as it would be looking straight down on it from above. The d_f values between 1.4 and 1.5 indicate a quite fragmented object in the plane and the flatness of the d_f line indicates self-similarity. This means that over these scales a smaller piece of the UD closely resembles a larger view of it, in a statistical sense. A value of d_f around 1.5 indicates white-noise Brownian motion, so we can say that the hawk's movement (exclusive of perches) resembles a random walk.

The robustness of this analysis with respect to measurement error was analyzed by Monte Carlo simulation of point estimation error. The nominal set of data points was modified by the addition of random noise uniformly distributed over [-80, +80] displacement for both X and Y coordinates, giving a mean center-to-center displacement of around 9.5 m over the five stochastic runs. This degree of measurement error is probably realistic for the hawk data. The result of this analysis is shown in Fig. 3. With this degree of measurement error, we see that the pattern evident in Fig. 2 is virtually unchanged and the degree of spread is quite acceptable. Thus a fractal measure of home range shape is rather insensitive to measurement error.

The effect of sample size was next evaluated. The full 123 observations was compared to d_f calculated with the first 98, first 73, 48, and 23 observations. Results (Fig. 4) show that the 23 and 48 sample sizes are inadequate, but above this the sample size does not significantly affect the pattern. This robustness to sample size results from the self-similar nature of the space usage. This points out the other desirable property of using fractal analyses: robustness to sample intensity.

For those uncomfortable with using the search radii, the home range was analyzed as point data. Stochastic runs (i.e., with the same measurement

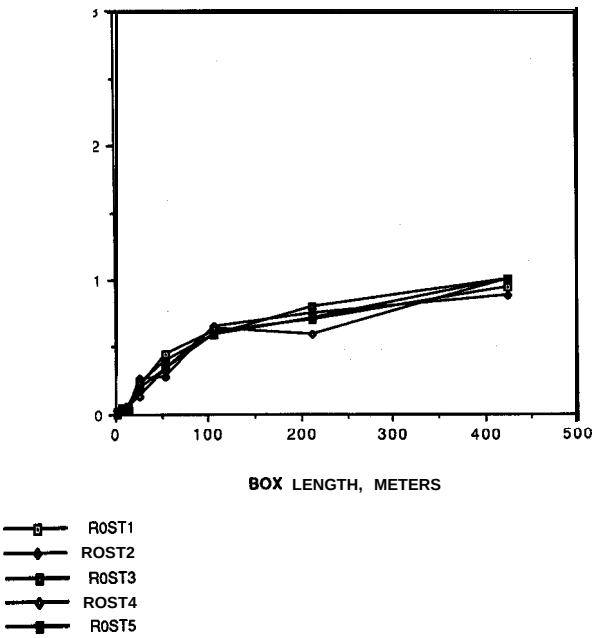

Fig. 5. d_f as a function of scale for point data (r = 0) with repeat Monte Carlo runs shown to evaluate measurement error.

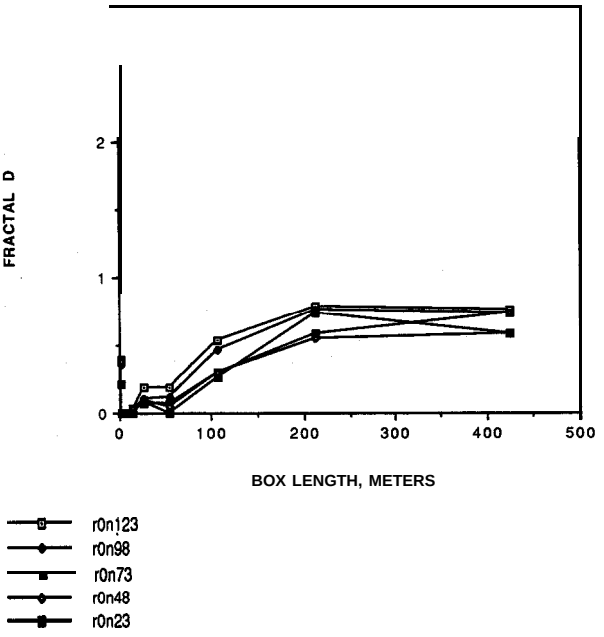

Fig. 6. d_f as a function of scale for point data (r = 0) as affected by sample size.

error as used above) are shown in Fig. 5. The d_f curve rises from zero (point data have d = 0) up to around .7– .8. Note, however, that self-similarity is much less evident, concentration of use on favored perches is not evident at all, and the measurement error causes more divergence in the curve, particularly in our ability to see the self-similarity. With no measurement error (Fig. 6) the perching behavior is barely evident at the lower left, and the sample size effect is more pronounced than when r = 100. In this case the n = 73 sample size becomes inadequate. Thus the use of point data is overall much less robust than the use of search circles overlaid on each point.

Discussion

The methods presented represent a new method for quantifying animal movement patterns. Observational point data is converted into short-interval search areas. These overlapping search areas are overlaid to form 3-D utilization distributions. The UD's are analyzed for concentration of use and fractal dimensions to quantify extent of fragmentation of home range, self-similarity, and concentration of usage. Algorithms to compute these quantities are presented. The effect of location estimation error is evaluated using Monte Carlo simulation and it is concluded that the descriptors are very robust with respect to this type of error. Sample size adequacy can also be addressed with these methods, which are fairly robust with respect to sample size.

The analyses conducted above suggest that the concept of home range as a bounded space within which an animal may be found (like a suburban property line) is not really biologically meaningful. Within the area used by an animal some areas are never used while others are used repeatedly. All of this kind of detail is lost if we just draw a line around the points where the animal has been seen. Fractal measures capture many features of the pattern of habitat usage. Furthermore, converting point data to search areas incorporates, in a natural way, the grain of the habitat as the animal perceives it and the differences in habitat usage due to differences between species. By placing home ranges of a number of animals on a larger map, a plot can be

made of all scales from perches on up to the landscape level. One could thereby analyze habitat usage patterns at various scales simultaneously. Overlap of adjacent animal territories could also be easily studied using these methods.

The methods presented here need extensive testing with a variety of species before all the subtleties of this kind of analysis will be worked out. Nevertheless, the ease of use and ease of interpretation, fewness of assumptions, and robustness to measurement error and sample size recommend these methods for more extensive evaluation.

Acknowledgments

This work was carried out under contract DE-AC09-88SR18035 with the U.S. Department of Energy, with whom the copyright remains. I would like to thank T. Gilman of the Scientific Computations Section of SRL for graphics programming assistance. Paul Steblein of the New York State Museum Science Service, Albany, N.Y. graciously provided the hawk data.

References

Altmann, S.A. and Altmann, J. 1970. Baboon ecology. University of Chicago Press. Chicago, Illinois, USA.

Anderson, D.J. 1982. The home range: a new nonparametric estimation technique. Ecology 63: 103-l 12.

Barnsley, M. 1988. Fractals Everywhere. Academic Press, N.Y.

Burrough, P.A. 1986. Principles of geographical information systems for land resources assessment. Clarendon Press,' Oxford.

Dicke, M. and Burrough, P.A. 1988. Using fractal dimensions for characterizing tortuosity of animal trails. Physiolog. Ent. 13: 393-398.

Dunn, J.E. 1978. Computer programs for the analysis of radio telemetry data in the study of home range. U. Ark. Stat. Lab. Tech. Rept. 7.

Dunn, J.E. 1979. A complete test for dynamic territorial interaction. U. Ark. Stat. Lab. Tech. Rept. 10.

Dunn, J.E. and Gipson, P.S. 1977. Analysis of radio telemetry data in studies of home range. Biometrics 33: 85–101.

Ford, R.G. and Krumme, D.W. 1979. The analysis of space use patterns. Journal of Theoretical Biology 76: 125-155.

Håkanson, L. 1978. The length of closed geomorphic lines. Mathem. Geol. 10: 141-167.

Harestad, A.S. and Bunnel, F.L. 1979. Home range and body weight – a reevaluation. Ecology 60: 389-402.

Jennrich, R.I. and Turner, F.B. 1969. Measurement of non-circular home range. Journal of Theoretical Biology 22: 227-237.

Krummel, J.R., Gardner, R.H., Sugihara, G., O'Neill, R.V. and Coleman, P.R. 1987. Landscape patterns in a disturbed environment. Oikos 48: 321-324.

Lam, N.S.-N. 1990. Description and measurement of Landsat TM images using fractals. Phot. Eng. & Remote Sens. 56: 187-195.

Loehle, C. 1983. The fractal dimension and ecology. Speculations in Science and Technology 6: 313-342.

Lovejoy, S., Schertzer, D. and Tsonis, A.A. 1987. Functional box-counting and multiple elliptical dimensions in rain. Science 235: 1036-1038.

MacDonald, D.W., Ball, F.G. and Hough, N.G. 1980. The evaluation of home range size and configuration using radio tracking data. *In* A Handbook on biotelemetry and radio tracking, pp 405-424. Edited by C.J. Amlaner and D.W. MacDonald. Pergamon Press, Oxford, United Kingdom.

Mandelbrot, B.B. 1983. The Fractal Geometry of Nature. W.H. Freeman and Co., N.Y.

McNab, B.K. 1963. Bioenergetics and the determination of home range size. American Naturalist 97: 133-140.

Milne, B.T. 1988. Measuring the fractal geometry of landscapes. App. Math. Comput. 27: 67-79.

Morse, D.R., Lawton, J.H., Dodson, M.M. and Williamson, M.H. 1985. Fractal dimension of vegetation and the distribution of arthropod body lengths. Nature 314: 731-733.

Neu, C.W., Byers, C.R. and Peek, J.M. 1974. A technique for analysis of utilization – availability data. J. Wildl. Manage. 38: 541-545.

Palmer, M.W. 1988. Fractal geometry: a tool for describing spatial patterns of plant communities. Vegetatio 75: 91-102.

Phillips, J.D. 1985. Measuring complexity of environmental gradients. Vegetatio 64: 95– 102.

Pietgen, H.O. and Saupe, D. 1988. The Science of Fractal Images. Springer Verlag, N.Y.

Samuel, M.D. and Garton, E.O. 1985. Home range: a weighted normal estimate and tests of underlying assumptions. J. Wildl. Manage. 49: 513-519.

Sanderson, G.C. 1966. The study of mammal movements: A review. J. Wildl. Manage. 30: 215-235.

Schoener, T.W. 1968. Sizes of feeding territories among birds. Ecology 49: 123-141.

Schoener, T.W. 1981. An empirically based estimate of home' range. Theoretical Population Biology 20: 28 l-325.

Sharpe, P.J.H., Walker, J., Penridge, L.K., Wu, H. and Rykiel Jr, E.J. 1986. Spatial considerations in physiological models of tree growth. Tree Physiology 2: 403-421.

Southwood, T.R.E. 1966. Ecological methods. Methuen, London. United Kingdom.

Swihart, R.K., Slade, N.A. and Bergstrom, B.J. 1988. Relating body size to the rate of home range use in mammals. Ecology 63: 393-399.

Trevor-Deutsch, B. and Hackett, D.F. 1980. An evaluation of several grid trapping methods by comparison with radio telemetry in a home range study of the eastern chipmunk (*Tamias striatus* L.). *In* A Handbook on Biotelemetry and Radio Tracking, pp 375-385. Edited by C.J. Amlaner and D.W. MacDonald. Pergamon Press, Oxford, United Kingdom.

Turner, F.B., Jennrich, R.I. and Weintraub, J.D. 1969. Home ranges and the body size of lizards. Ecology 50: 1076-1081.

Turner, M.G., Constanza, R. and Sklar, F.H. 1989. Methods to evaluate the performance of spatial simulation models. Ecol. Model. 48: l-18.

Van Winkle, W. 1975. Comparison of several probabilistic home-range models. Journal of Wildlife Management 39: 118-123.

Van Winkle, W., Martin, D.C. and Sebetich, M.J. 1973. A home-range model for animals inhabiting an ecotone. Ecology 54: 205-209.

Voigt, D.R. and Tinline, R.R. 1980. Strategies for analyzing radio tracking data. *In* A Handbook on Biotelemetry and Radio Tracking, pp 387-404. Edited by C.J. Amlaner and D.W. MacDonald. Pergamon Press, Oxford, United Kingdom.

Appendix I
FORTRAN box-counting

***FORTRAN** version*

The input is an array named RASTER. It is assumed that RASTER is a 2048 x 2048 INTEGER *2 array with origin at (1,1). Each grid location must have on input an integer value between 0 and 99, representing the height of the map at that point (representing elevation, animal numbers, biomass, etc., appropriately scaled). If RASTER is INTEGER (4-byte), then four times as much storage space will be used, but greater vertical resolution is attainable. To analyze a flat map (vegetation type data, soil type data), all that is necessary is to enter all pixels of the type of interest in l's and everything else as zeros. The estimate of d_f will come out between 0 and 2. At the finest resolution, the number of boxes is simply the height summed over all pixels. At coarser scales, volumes filled determine the number of boxes. For example, for a 2 x 2 box the volume must be greater than 8 (2 x 2 x 2) to count two cubes vertically, greater than 24 to count four cubes, etc. Computations are multiply-nested, with the innermost loop being the finest scale (pixel by pixel). The dimension d_f is defined by the slope $\Delta x/\Delta y$ (from the slope of a log-log plot of boxes as a function of scale), where Ax is log (B_{i-1})-log (B_i), where log is natural log, (B_i) is # boxes and $_i$ is the power of 2, and Ay is log (S_i)-log (S_{i-1}) where S_i is the number of steps along the map edge. This is a discrete estimate of d_f as one moves from one scale to another. This algorithm can handle any value of d_f between 0 and 3. Note that it assumes the map is conpact: that is, flat on the bottom and with no caves or floating objects, like checkers stacked on the squares of a checkerboard. Caves and floating objects are prevented on input by the integer height-at-each-pixel input format. Negative heights will cause errors. As Palmer (1988) notes, ecological data is likely to be fractal (d is not integer) but not self-similar because d varies with scale. The box counting algorithm presented here allows evaluation of this effect.

Assuming that the map grid is fine enough, the algorithm should be able to handle any type of spatial data once the user has loaded it into RASTER. It uses about 8 Megabytes of memory. If the user modifies this for larger or smaller maps, care must be taken that (MAXGRID) is a power of 2. A larger map may be analyzed by adding outer loops to the calculation.

Tests were conducted of FRACTAL. Long objects one pixel wide gave d_f = 1. to .98 at all scales. Smooth circles or squares gave d_f = 2 except at very coarse scales. Single or a few scattered points gave d_f = 0. A stepped pyramid (100 x 100 base, 1 unit high, upper left 40 x 40 base 10 units high, upper left 10 x 10 base 40 units high) gave the following output:

# STEPS	LN # STEPS	# BOXES	LN #BOXES	D_f
	0.000	19003.	9.852	
2	0.693	2378.	7.774	2.998
4	1.386	349.	5.855	2.768
8	2.079	58.	4.060	2.589
16	2.773	13.	2.565	2.158
32	3.466	7.	1.946	0.893
64	4.159	4.	1.386	0.807
128	4.852	1.	0.000	2.000
256	5.545	1.	0.000	0.000

Thus at the finest scale this object is correctly measured as d = 3 and is smooth. Then it is viewed as a spikey object (d ≃ 2.5). At still coarser scales d ≈ 2 (it is seen as a *flat* smooth object) and at the coarsest scale it becomes a point. This is all as we expect so the program is verified.

Appendix II
Pascal box-counting

Pascal version

A limitation of the FORTRAN version is that it must be reprogrammed for larger or smaller maps. A Pascal version overcomes this problem. The Pascal version uses recursion instead of nested loops. It thus does not need any modification for larger (up to 2^{12} pixels on a side) or smaller maps, except for redefining the data array size. The output table from this differs slightly from the FORTRAN version. Testing of it gave similar results to the above tests of the FORTRAN version. The Pascal version is also useful for work on PC's where a FORTRAN compiler may not be available.

```
      SUBROUTINE  FRACTAL
C     CALCULATES 3-D FRACTAL COVERAGE USING BOX COUNTING
C     IT IS ASSUMED THAT ANY INTRUSION INTO A CUBE COUNTS
C ---------------- COMMON:  PERIVAR ----------------------------------------
      INTEGER MAXGRID
      PARAMETER (MAXGRID=2048)
      INTEGER*2 RASTER(MAXGRID,MAXGRID)
      COMMON /PERIVAR/ RASTER
C -------------------------------------------------------------------------
      INTEGER IX,IY,IX2,IY2,IX4,IY4,IX8,IY8,IX16,IY16,IX32,IY32
      INTEGER IX64,IY64,IX128,IY128,IX256,IY256,IX512,IY512
      REAL COUNT(11),SUM(11),SLOPE(11),LCOUNT(11),ADD,TOTAL
      INTEGER ISTEPS(11),L,I


      TOTAL=0.
      DO 100 I=1,11
        COUNT(I)=0.
        SUM(I)=0.
        ISTEPS(I)=2**I
100   CONTINUE
      SUM(8)=0.
      DO 512 IX512=0,ISTEPS(11)-ISTEPS(8),ISTEPS(8)
      DO 512 IY512=0,ISTEPS(11)-ISTEPS(8),ISTEPS(8)
          SUM(7)=0.
          DO 256 IX256=0,ISTEPS(7),ISTEPS(7)
          DO 256 IY256=0,ISTEPS(7),ISTEPS(7)
            SUM(6)=0.
            DO 128 IX128=0,ISTEPS(6),ISTEPS(6)
            DO 128 IY128=0,ISTEPS(6),ISTEPS(6)
              SUM(5)=0.
              DO 64 IX64=0,ISTEPS(5),ISTEPS(5)
              DO 64 IY64=0,ISTEPS(5),ISTEPS(5)
                SUM(4)=0.
                DO 32 IX32=0,ISTEPS(4),ISTEPS(4)
                DO 32 IY32=0,ISTEPS(4),ISTEPS(4)
                  SUM(3)=0.
                  DO 16 IX16=0,ISTEPS(3),ISTEPS(3)
                  DO 16 IY16=0,ISTEPS(3),ISTEPS(3)
                    SUM(2)=0.
                    DO 8 IX8=0,ISTEPS(2),ISTEPS(2)
                    DO 8 IY8=0,ISTEPS(2),ISTEPS(2)
                      SUM(1)=0.
                      IX4 MAKES A 4X4 LOOP (2X2 +2X2 +2X2 +2X2)
                      DO 4 IX4=0,ISTEPS(1),ISTEPS(1)
                      DO 4 IY4=0,ISTEPS(1),ISTEPS(1)
                        ADD=0.
                        DO 2 IX2=1,ISTEPS(1)
                        DO 2 IY2=1,ISTEPS(1)
                          IX=IX2+IX4+IX8+IX16+IX32+IX64+IX128+IX256+
     1                       IX512
                          IY=IY2+IY4+IY8+IY16+IY32+IY64+IY128+IY256+
     1                       IY512
                          ADD=ADD+RASTER(IX,IY)
2                       CONTINUE
                        TOTAL=TOTALtADD
                        SUM(1)=SUM(1)+ADD
                        COUNT(1)=COUNT(1)+JINT(ADD/FLOAT(ISTEPS(1))
     1                     **3)
                        IF(AMOD(ADD,FLOAT(ISTEPS(1))**3).GT.0.)
     1                     COUNT(1)=COUNT(1)+1
4                     CONTINUE
                      COUNT(2)=COUNT(2)+JINT(SUM(1)/FLOAT(ISTEPS(2))
```

```
     1                          **3)
                              IF(AMOD(SUM(1),FLOAT(ISTEPS(2))**3).GT.0.)
     1                          COUNT(2)=COUNT(2)+1
                            SUM(2)=SUM(2)+SUM(1)
     8                      CONTINUE
                           COUNT(3)=COUNT(3)+JINT(SUM(2)/FLOAT(ISTEPS(3))
     1                          **3)
                            IF(AMOD(SUM(2),FLOAT(ISTEPS(3))**3).GT.0.)
     1                          COUNT(3)=COUNT(3)+1
                            SUM(3)=SUM(3)+SUM(2)
    16                      CONTINUE
                          COUNT(4)=COUNT(4)+JINT(SUM(3)/FLOAT(ISTEPS(4))
     1                          **3)
                            IF(AMOD(SUM(3),FLOAT(ISTEPS(4))**3).GT.0.)
     1                          COUNT(4)=COUNT(4)+1
                            SUM(4)=SUM(4)+SUM(3)
    32                      CONTINUE
                         COUNT(5)=COUNT(5)+JINT(SUM(4)/FLOAT(ISTEPS(5))**3)
                         IF(AMOD(SUM(4),FLOAT(ISTEPS(5))**3).GT.0.)
     1                          COUNT(5)=COUNT(5)+1
                         SUM(5)=SUM(5)+SUM(4)
    64                   CONTINUE
                       COUNT(6)=COUNT(6)+JINT(SUM(5)/FLOAT(ISTEPS(6))**3)
                       IF(AMOD(SUM(5),FLOAT(ISTEPS(6))**3).GT.0.)
     1                          COUNT(6)=COUNT(6)+1
                       SUM(6)=SUM(6)+SUM(5)
   128                 CONTINUE
                     COUNT(7)=COUNT(7)+JINT(SUM(6)/FLOAT(ISTEPS(7))**3)
                     IF(AMOD(SUM(6),FLOAT(ISTEPS(7))**3).GT.0.)
     1                          COUNT(7)=COUNT(7)+1
                     SUM(7)=SUM(7)+SUM(6)
   256               CONTINUE
                  COUNT(8)=COUNT(8)+JINT(SUM(7)/FLOAT(ISTEPS(8))**3)
                  IF(AMOD(SUM(7),FLOAT(ISTEPS(8))**3).GT.0.)COUNT(8)=COUNT(8)+1
                  SUM(8)=SUM(8)+SUM(7)
   512            CONTINUE


                  WRITE(6,*)
     1'               #STEPS        LN #STEPS        #BOXES    LN   #BOXES   Df'
                  LCOUNT(1)=LOG(COUNT(1))
                  L=1
                  WRITE(6,2000)L,LOG(FLOAT(L)),TOTAL,LOG(TOTAL)
                  SLOPE(1)=(LOG(TOTAL)-LCOUNT(1))/LOG(FLOAT(ISTEPS(1)))
     C            NOTE ABOVE THAT LOG(1)=0
                  WRITE(6,2000)ISTEPS(1),LOG(FLOAT(ISTEPS(1))),COUNT(1),
     1              LCOUNT(1),SLOPE(1)
                  DO 1000 L=2,8
                    LCOUNT(L)=LOG(COUNT(L))
                    SLOPE(L)=(LCOUNT(L-1)-LCOUNT(L))/(LOG(FLOAT(ISTEPS(L)))-
     1                LOG(FLOAT(ISTEPS(L-1))))
                    WRITE(6,2000)ISTEPS(L),LOG(FLOAT(ISTEPS(L))),COUNT(L),
     1                LCOUNT(L),SLOPE(L)
  2000            FORMAT(4X,I7,7X,F9.3,3X,F11.0,3X,F8.3,3X,F7.3)
  1000            CONTINUE
                  RETURN
                  END
```

Listing of RECUR.PAS, page 1 at 21:02 Monday 1/29/90

```
 1        {$R+}
 2        uses Dos;
 3        TYPE
 4           SUMVAR=ARRAY[1..12] OF LONGINT;
 5           GRIDVAR=ARRAY[1..128,1..128] OF INTEGER:
 6
 7        VAR  DATA:TEXT;
 a
 9
10        PROCEDURE RECUR(VAR GRID:GRIDVAR; VAR TEMPSUM:LONGINT; VAR
          SUM:SUMVAR;
11            VAR INDEX:INTEGER; BOXSIZ:lONGINT; VAR
              GRIDX,GRIDY:SUMVAR);
12
13        VAR TEMPX,TEMPY,IX,IY:INTEGER;
14            HOLD:LONGINT;
15
16        BEGIN
17        INDEX:=INDEX+1;
18        IF(BOXSIZ>2)
19          THEN BEGIN
20               BOXSIZ:=BOXSIZ DIV 2;
21                HOLD:=0;
22                FOR IY:=0 TO 1 DO
23                  FOR IX:=0 TO 1 DO
24                    BEGIN
25                    GRIDX[INDEX]:=IX*BOXSIZ;
26                    GRIDY[INDEX]:=IY*BOXSIZ;
27                    RECUR(GRID,TEMPSUM,SUM,INDEX,BOXSIZ,GRIDX,GRIDY);
28                    HOLD:=HOLD+TEMPSUM;
29                    END;
30                SUM[INDEX]:=SUM[INDEX]+(HOLD DIV
                  (BOXSIZ*BOXSIZ*BOXSIZ*8));
31                IF((HOLD MOD (BOXSIZ*BOXSIZ*BOXSIZ*8))>0)
32                  THEN SUM[INDEX]:=SUM[INDEX]+1;
33                TEMPSUM:=HOLD;
34              END
35          ELBE BEGIN
36                TEMPX:=0;
37                FOR IX:=1 TO 12 DO
38                   TEMPX:=TEMPX+GRIDX[IX];
39                TEMPY:=0;
40                FOR IY:=1 TO 12 DO
41                   TEMPY:=TEMPY+GRIDY[IY];
42                TEMPSUM:=0;
43                FOR IY:=1 TO 2 DO
44                  FOR IX:=1 TO 2 DO
45                    BEGI
46                    TEMPSUM:=TEMPSUM+GRID[TEMPX+IX,TEMPY+IY]:
47                    END;
48                SUM[INDEX]:=SUM[INDEX]+(TEMPSUM DIV
                  (BOXSIZ*BOXSIZ*BOXSIZ\\:,,,
49                IF((TEMPSUM MOD (BOXSIZ*BOXSIZ*BOXSIZ))>0)
50                   THEN SUM[INDEX]:=SUM[INDEX]+1;
```

```
51          SUM[INDEX+1] :=SUM[INDEX+1]+TEMPSUM;
52          END;
53
54      INDEX:=INDEX-1;
55      END;
56        (*MAIN  PROGRAM*)
57      VAR SUM,GRIDX,GRIDY,POWER:SUMVAR;
58            GRID:GRIDVAR;
59            INDEX,IX,IY:INTEGER;
60            TEMPSUM,BOXSIZ   :  LONGINT;
61            RUN,RISE,DUMMY1,DUMMY2:REAL;
62      BEGIN
63      ASSIGN(DATA,'RECUR.OUT');
64      REWRITE(DATA);
65      FOR IY:=1 TO 12 DO
66        BEGIN
67        GRIDX[IY]:=0;
68        GRIDY[IY]:=0;
69        SUM[IY]:=0;
70        END;
71      POWER[1]:=1;
72      FOR IY:=2 TO 12 DO
73        POWER[IY]:=POWER[IY-1]*2;
74      BOXSIZ:=128;
75      (*THIS  IS  A  TEST  SET  OF  INPUT  DATA*)
76      FOR IY:=1 TO BOXSIZ DO
77        FOR IX:=1 TO BOXSIZ DO
78          BEGIN
79          GRID[IX,IY]:=0;
80          END; (*END INITIALIZING GRID*)
81      (*TRY  A  PYRAMID*)
82      FOR IY:=1 TO 30 DO
83        FOR IX:=1 TO 30 DO
84        BEGIN
85        GRID[IX,IY]:=1;
86        END:
87      FOR IY:=1 TO 15 DO
88        FOR IX:=1 TO 15 DO
89        BEGIN
90        GRID[IX,IY]:=12;
91        END;
92      FOR IY:=1 TO 4 DO
93        FOR IX:=1 TO 4 DO
94        BEGIN
95        GRID[IX,IY]:=23;
96        END;
97      INDEX:=0;
98      RECUR(GRID,TEMPSUM,SUM,INDEX,BOXSIZ,GRIDX,GRIDY);
99      INDEX:=2;
100     WRITELN(DATA,'     #STEPS       LN  #STEPS        #BOXES      LN
        #BOXES     Df');
101     WHILE(SUM[INDEX]>0)DO
102       BEGIN
103       DUMMY1:=SUM[INDEX];
```

```
104             DUMMY2:=SUM[INDEX-1];
105             RISE:=LN(DUMMY1)-LN(DUMMY2);
106             RUN:=LN(l.O*POWER[INDEX])-LN(l.O*POWER[INDEX-l]):
107             W
                RITELN(DATA,(POWER[INDEX-1]):10,LN(1.0*POWER[INDEX-1]):16,
108                  SUM[INDEX-1]:14,LN(DUMMY2):12,
109                    (RISE/RUN):6:2);
110             INDEX:=INDEX+1;
111         END;
112     DUMMY2:=SUM[INDEX-1];
113     WRITELN(DATA,(POWER[INDEX-1]):10,LN(1.0*POWER[INDEX-1]):16,SUM[
        INDEX-1]:14,
114         LN(DUMMY2):12);
115     CLOSE(DATA);
116 END.
```

Appendix III
Subroutine iucircle

This routine takes **(x,y)** pair animal location data and loads it into RASTER so that it can be used in FRACTAL. A box of size **2*** RADIUS is placed around each **(x,y)** data point and all array elements in this box checked for coverage by the circle centered at **(x,y).** This saves considerable time over checking for each **ar**ray location all the circles to see if they overlap.

```
      SUBROUTINE    INCIRCLE(SCALE,NPOINTS,BLKS)
      REAL  SCALE,Z,Q,X,Y
      LOGICAL   FOUNDBLK
      INTEGER    K,IX,IY,NPOINTS,BLKS
C-------------------COMMON: MAXMIN-----------------------------------------
        REAL    XMAX,XMIN,YMAX,YMIN
        COMMON/MAXMIN/XMAX,XMIN,YMAX,YMIN
C ----------------- COMMON: PERIVAR -----------------------------------------
        INTEGER  MAXGRID
        PARAMETER (MAXGRID=3202)
        INTEGER*2  RASTER(MAXGRID,MAXGRID)
        COMMON /PERIVAR/  RASTER
C    ---------------------------------- ------------------------------------
C ----------------COMMON:    RPLOTVAR-------------------------------------
        REAL  RNEWX(175),RNEWY(175)
        INTEGER    RPOINTCNT,NTP,NSK,NP
        REAL  RADIUS,RXSCALE,RADOFF
        COMMON/RPLOTVAR/RADOFF,RADIUS,RXSCALE,RNEWX,RNEWY,RPOINTCNT,NTP,
     1    NSK,NP
C-------------------------------------------------------------------------

C       WE HAVE A BOX NPOINTS X NPOINTS IN SIZE THAT ENCLOSES ALL THE
C       CIRCLES.   WE SCAN EACH GRID POINT IN THIS BOX TO SEE IF IT IS IN
C       ANY CIRCLE.   SUM UP THE HEIGHT ON EACH GRID (# CIRCLES OVERLAPPING IT)-
C       WE ARE SCANNING A RECTANGLE, SO WE JUMP OUT WHEN WE PASS XMAX OR YMAX
C       WHICH ARE THE KNOWN LARGEST VALUES.   BLKS GIVES THE TOTAL NUMBER OF
C       CUBES  COUNTED.
      BLKS = 0
      X  =  XMIN-SCALE
      DO  110  IX=0,NPOINTS
         X  =  X+SCALE
         IF(X.GT.XMAX)GO TO 120
         Y  =  YMIN-SCALE
         DO  100  IY.=0,NPOINTS
            Y  =  Y+SCALE
            IF(Y.GT.YMAX)GO TO 110
            FOUNDBLK=.FALSE.
            DO 90  K=1,RPOINTCNT
               Z  =  RNEWX(K)-X
               Q  =  RNEWY(K)-Y
               IF((RADIUS+0.5*SCALE).GE.SQRT(Z*Z t Q*Q))THEN
                  FOUNDBLK=.TRUE.
                  RASTER(IX,IY)=RASTER(IX,IY)+1
               ENDIF
90          CONTINUE
         IF(FOUNDBLK)BLKS=BLKS+1
100      CONTINUE
110   CONTINUE
120   CONTINUE
      RETURN
      E N D
```