Random walk probability distributions

Murat Aytekin

Let δ be the distance a walker makes either to the right or to the left with equal probability within a unit period time τ. The probability that a walker will be at a distance

Taking the limit $\delta, \tau \to 0$ such that ${}^{\delta^2}/_{\tau} = 2D$, where D is the constant known as diffusion coefficient, the pdf for the location of the walker after time t is given as

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{\left(\frac{-x^2}{4Dt}\right)}$$

This equation is also the solution of the diffusion equation, $\frac{\partial p}{\partial t} = D\nabla^2 p$.

The second raw moment of position

$$E\left(X_t^2\right) = \int x^2 p\left(x,t\right) dx = 2Dt$$

For 2-dimensional random walk the probability distribution is

$$p(x,y,t) = \frac{1}{4\pi Dt} e^{\left(\frac{-\left(x^2+y^2\right)}{4Dt}\right)}$$

The mean squared distance is

$$E\left(R_t^2\right) = \int (x^2+y^2) p\left(x,y,t\right) dx = 4Dt$$

Biased random walk

Lets $\mathbf{u}$ be the average drift velocity (representing the bias) in 2-dimensional space and ∇ is the gradient vector.

$$\frac{\partial p}{\partial t} = -\mathbf{u}\nabla p + D\nabla^2 p$$

The pdf is then

$$p(x, y, t) = \frac{1}{4\pi Dt} e^{\left(\frac{-\left((x-\mathbf{u}t)^2+(y-\mathbf{u}t)^2\right)}{4Dt}\right)}$$

The mean squared distance

$$E\left(R_t^2\right) = \|\mathbf{u}\|^2 t + 4Dt$$

Matlab script 1 A Matlab script to generate 2-dimensional random walk process samples with a desired diffusion coefficient and directional bias.

```
function [x, y] = generate2DRandomWalk (D_given, samplingFreq, Ntrace, nT,
    BiasDir, BiasSpd_mean, BiasSpd_std)

% D_given: Desired diffusion coefficient
%      nT: Number of samples per trace
%  Ntrace: Number of tracks
% BiasDir: Bias direction (deg)
% BiasSpd_mean: Bias speed (unit/s)
% BiasSpd_std: Bias speed (unit/s)

dimensions = 2;                 % time interval in seconds
tau = 1/samplingFreq;

k = sqrt(D_given * dimensions * tau);  % standard deviation of each dimension
dx = k * randn(Ntrace,nT);        % x component of instantenous displacement
dy = k * randn(Ntrace,nT);        % y component of instantenous displacement

% For directional bias
if nargin == 7
  BiasSpdX = cosd(BiasDir).*(sqrt(BiasSpd_std).*randn(Ntrace,nT)+...
```

```
        BiasSpd_mean);
    BiasSpdY = sind(BiasDir).*(sqrt(BiasSpd_std).*randn(Ntrace,nT)+...
        BiasSpd_mean);
    dx = (BiasSpdX*tau) + dx;
    dy = (BiasSpdY*tau) + dy;
end

x = cumsum(dx,2);          % x component of displacement
y = cumsum(dy,2);          % y component of displacement
```

Estimation of diffusion coefficient

Lets assume that we have samples $\{x_k(t) \text{ and } y_k(t) \text{ s.t } t\epsilon[0,T_k]\}_{k=1,\dots N}$ from the random walk process. An estimate of $E\left(R_t^2\right)$ can be obtained by computing mean distance squared across all samples for a given t; $\hat{R}_t^2 = \left\langle x_t^2 + y_t^2 \right\rangle$. If we assume that our random walk model has no bias

$$D = \frac{\hat{R}_t^2}{4t} \tag{1}$$

This relationship constitutes the direct method of estimating diffusion coefficient with the strict assumption that there is no directional bias in the process.

Relation with normal distribution

At each time period $[0,t]$ the simple random walk positions are distributed as a Gaussian function:

$$p(x,y,t) = \frac{1}{4\pi Dt} e^{\left(\frac{-\left(x^2+y^2\right)}{4Dt}\right)}$$

Lets assume that the standard deviation of the process at each dimension is $\sigma_x = \sigma_y = \sigma$. The distances squared are distributed as so called Rayleigh distribution. The mean distance squared corresponds to this distribution's second raw moment which is $\mu_2 = 2\sigma^2$. Consequently, if one knows the standard deviation of the Gaussian distribution at all times t, the diffusion coefficient can be computed as

$$D = \frac{\hat{R}_t^2}{4t} = \frac{\sigma^2}{2t} \tag{2}$$

In some applications one may have the are area of the Gaussian border on the xy plane. The area of the ellipsoid corresponding to the radii of the standard deviations along the x and y axes then the diffusion coefficient is

$$D = \frac{\hat{R}_t^2}{4t} = \frac{\sigma^2}{2t} = \frac{\text{Area}}{4\pi t} \tag{3}$$

Here $\mathsf{Area} = 2\pi\sigma_x\sigma_y$. Note that with this definition now we can relax the assumption that the standard deviation of the random walk process along the x and y axes do not have to be equal. Naturally, this situation can be extended to the cases where the two Gaussian processes that define x and y axes can be correlated. That is, $\sigma^2 = \sigma_x^2 + \sigma_y^2 + \rho\sigma_x\sigma_y$, where ρ is the correlation coefficient. In this case, it is always possible to find the two independent axes that will allow us to write $\sigma^2 = \sigma_1^2 + \sigma_2^2$. Here, σ_1 and σ_2 represent the standard deviation along the two principle component axes. Hence, the area can be written in a more generally form as $\mathsf{Area} = 2\pi\sigma_1\sigma_2$.

It is important to emphasize that, for a random walk with correlated dimensions, the principle axes are not expected to change direction for different time intervals. If such a change is observed, then it is natural to assume that the processes that give rise to the random walk are not stationary, but have additional time dependent terms, which by itself may be an interesting descriptive feature of the underlying natural phenomenon.

Matlab script 2 Matlab script to estimate diffusion coefficient with 3 different ways described.

```
function [D1, D2, D3, bias] = estimateDiffCoefficients (x, y, samplingFreq)

% x, y: horizontal and vertical traces of drift samples
%   D1: diffusion coefficient obtained from the slope of the regression
%       line between <d^2> and deltaT.
%   D2: diffusion coefficient obtained from the slope of the regression
%       line between Gaussian area and deltaT.
%   D3: diffusion coefficient obtained from the slope of the regression
%       line after removing bias term using PCA.
% BiasMeanX, BiasMeanY: Directional bias
%
% example:
% [x, y] = generate2DRandomWalk (41, 1000, 100, 1000, 60, 10, .02);
% [D1, D2, D3, bias] = estimateDiffCoefficients (x, y, 1000);
% % plot estimated bias direction
% plot(atan(bias.biasMeanY./bias.biasMeanX).*180/pi) %


dimensions = 2;

[Ntrace, nT] = size(x);
tm = (0:nT-1)./samplingFreq;    % create a time vector for plotting
```

```
Dx = cell(nT,1);
Dy = cell(nT,1);

for dt = 1:nT-1
    dx = x(:,1+dt:end) - x(:,1:end-dt);
    dy = y(:,1+dt:end) - y(:,1:end-dt);            % pool displacements

    Dx{dt+1} = [Dx{dt+1} dx(:)'];
    Dy{dt+1} = [Dy{dt+1} dy(:)'];
end

% displacements at zero time
dispSquared(1) = 0;
dispSquared(1) = 0;
area_unbiased(1) = 0;

for dt = 2:nT
    % method-1
    dispSquared(dt) = mean(Dx{dt}.^2 + Dy{dt}.^2);

    % method-2 finding std along the principle axes
    n = length(Dx{dt});
    mat = [Dx{dt}; Dy{dt}];
    sigmas = sqrt(eig(mat*mat')./n);
    area(dt) = 2*prod(sigmas);

    % method-3 (removing the mean (bias))
    bias.biasMeanX(dt-1) = mean(Dx{dt});
    bias.biasMeanY(dt-1) = mean(Dy{dt});
    bias.biasStdX(dt-1) = std(Dx{dt});
    bias.biasStdY(dt-1) = std(Dy{dt});

    mat = [Dx{dt} - bias.biasMeanX(dt-1); ...
        Dy{dt} - bias.biasMeanY(dt-1)];

    sigmas = sqrt(eig(mat*mat')./n);
    area_unbiased(dt) = 2*prod(sigmas);
end

% compute diffusion coefficent
ind = 1:round(nT*.5);
```

```
slp1 = regress(dispSquared(ind)', tm(ind)');
D1 = slp1/(2*dimensions);

slp2 = regress(area(ind)', tm(ind)');
D2 = slp2/(2*dimensions);

slp3 = regress(area_unbiased(ind)', tm(ind)');
D3 = slp3/(2*dimensions);
fprintf(['\n\nEstimated diffusion coefficients are '...
    'D1=%4.2f, D1=%4.2f, D1=%4.2f\n\n'], D1,D2,D3)
```
