Fra: Marit og Lars
Dato: 8.7.09

Simulate tracks

This note describes how to simulate synthetic tracks. The simulation algorithms are the most basic, with few parameters and short simulation algorithms. This may be used in testing different algorithms. We also include how this may be combined with intensity tracks.

Simulation of point tracks.

Let X_i be the distances between the points. X_1 is the distance from start of track to first point. The simulation algorithm includes clusters.

For i =1,2,3,… until end of track
With probability p, let X_i ~expon (λ_A), and with probability 1-p, let X_i ~expon (λ_B)

expon (λ_A) means exponentially distributed (i.e. density $\lambda_A \exp(-\lambda_A x)$) with parameter λ_A. There are three parameters:

λ_A determines distance between clusters,
λ_B determines distance within cluster and
p determines number of points in cluster.

If p=1, all points are independent and expected distance between points is $1/\lambda_A$ and median $\ln(2)/\lambda_A$.

If p<1, there are clusters. The number of points in the cluster is geometrically distributed. The expected number of points in the cluster is 1/p. Hence, p=1, means one point in each cluster, i.e. all points are independent. The expected distance between points in a cluster is $1/\lambda_B$ and median $\ln(2)/\lambda_B$. We should have $\lambda_A < \lambda_B$. The expected distance between last point in a cluster and first points in the next cluster is $1/\lambda_A$.

Simulation of segment tracks.

Assume overlap between segments

We may use the same algorithm as point tracks where X_i is the distance between centre points of segments. If the length of the segment is at the form 2n, we define the centre point to be the n'th base pair in the segment.

The length or the logarithm of the length of each segment may be uniformly distributed in (L_A, L_B). This gives 5 parameters

λ_A determines distance between clusters,
λ_B determines distance within cluster,

p determines number of points in cluster, and
(L_A, L_B) specifies the length of the segments.

Assume no-overlap between segments

We may use the same algorithm as above, but now let X_i be the distance between the segments. (i.e. from end of one segment to start of the following.)

Simulation of function tracks.

One level
Let the values of the function track be f_i . Assume first only an AR(1) process with the following algorithm:

$f_0 \sim N(a, \sigma^2/b)$
For i =1,2,3,…. until end of track
$f_i = a + b\,(f_{i-1} - a) + \varepsilon_i$ where $\varepsilon_i \sim N(0, \sigma^2)$

This gives values that varies around the level a. Each value is $N(a, \sigma^2/b)$. The correlation between neighboring values is $b\,\sigma^2$.

Several levels
In many cases it is more realistic that the function values vary around several values. We define k states and the function track is an AR(1) process when the track remains within one state.

Let P be a probability matrix where element $p_{i,j}$ is the probability to move from state i to state j, and let p_i^M be the marginal distribution. Theory for Markov chains implies that the vector p^M is an eigenvector of P with 1 as an eigenvalue. In practice, we will probably first specify the marginal distribution p^M. Then we will find a transition matrix P with the given eigenvector and eigenvalue. The expected number of base pairs in a state is $1/(1-p_{i,i})$. Hence, we will probably choose the diagonal elements in P close to 1.
One alternative for such a matrix is described below. For this matrix there is equal probability for all the other states when leaving a state:

For i =1,2,3,…,n
Set $p_{i,i}$ as a probability slightly less than 1
Set $p_{i,j} = (1- p_{i,i})\, p_i^M / (1-p_i^M)$ for $j \neq i$.

Let S_i be the state for base pair i. The following algorithm describes a function track with several levels.

For i =1,2,3,… until end of track

If (i=1) then <Simulate S_1 proportional with p^M>

else <Simulate S_i proportional with $p_{S\_i-1,>}$

Set $k=S_i$

If new state $g_{i-1} \sim N(a, \sigma_k^2/b_k)$ else $g_{i-1} = f_{i-1}$

$f_i = a_k + b_k (g_{i-1} - a_k) + \varepsilon_i$ where $\varepsilon_i \sim N(0, \sigma^2_k)$

The parameters are

a_k mean value in state k

b_k determines AR process in state k

σ_k determines noise term in state k

P transition matrix

Simulation with control track

Here we will show how to simulate correlated tracks that may be used when testing the control track option. This will be illustrated using GC-content and melting function, but the approach is general.

It is well known that both the melting function and the probability for an exon start increases with increasing GC content. The question is whether there is higher correlation between the melting function and exon start than the indirect correlation through the GC content. Or formulated slightly differently, do we know more about the exon start points knowing both the GC content and the melting function than only knowing the GC content.

Let f^{GC} be the function track of smoothed GC content. This is assumed to be data. Simulate a melting function track f^M by

$f^M = \alpha f^{GC} + (1-\alpha) f^l$

where α is a constant, f^{GC} is assumed known and f^l is a "noise" track independent of all the other tracks and simulated by one of the algorithms above.

Simulate the point track T_i^E for exons start correlated by the smoothed GC content by first simulating a new function track f^E by

$f^E = \beta f^{GC} + \gamma f^l$

where β and γ are constants, f^{GC} is assumed known and f^l is the same noise term as in the melting function. Define a probability $p(f_i^E)$ with a parameter f_i^E such that there is an exon start at base pair i with probability $p(f_i^E)$, i.e. $P(T_i^E=1)= p(f_i^E)$. We could have defined separate noise terms in f^M and f^E, but this would not make the model better since we could then replace $p(f_i^E)$ with the expected probability when these additional noise terms are not known.

In order to specify this process, it is necessary to specify the functions f^l, the parameters α, β, and γ, in addition to the probability function p(x).

If we assume f_i^{GC} and f_i^1 are independent and normal distributed with variance σ_{GC}^2 and σ_1^2 respectively, then the covariance is as follows:

$CoV(f_i^{GC}, f_i^E) = \alpha\beta\ \sigma_{GC}^2$, $\quad\quad$ $CoV(f_i^M, f_i^E) = \alpha\beta\ \sigma_{GC}^2 + (1-\alpha)\ \gamma\ \sigma_1^2$

From the specification above it is clear that the additional value of knowing f^M in addition to f^{GC} is connected to the track $\gamma\ f^1$.

Quantification of information from function track when evaluating point tracks

We need to quantify the information gained from one or several function tracks (i.e. f^{GC} and f^M) when evaluating a point track (i.e. T^E). Then we may quantify the increased information from knowing both the melting function and GC content compared to only the GC content.

As a measure we use the expected values for the probability for a point in the point track T_i^E in the base pairs where there is a point i.e. $T_i^E = 1$. We use the notation $E_{TiE=1}\{P(T_i^E=1)\}$, $E_{TiE=1}\{P(T_i^E=1|\ f_i^{GC})\}$, and $E_{TiE=1}\{P(T_i^E=1|\ f_i^M, f_i^{GC})\}$ for these expected value probabilities. These three expectations are the expected value of an exon start point in the base pairs where there are exon starts, when using no additional information, when using information about GC-content, and when using information about both the GC content and the melting function, respectively. If the melting function f^M gives additional information, the last expected value is larger. Notice that we in this example study each base pair separately. We do not use any spatial information in the model except that both $f^{GC}{}_i$ and $f^M{}_i$ depend on values in several base pairs close to base pair i.

Example

The following example is given in order to illustrate the concept. Assume all the function tracks have expectation 0 and let $p(x)=\min\{1,a\exp(bx)\}$. We may simulate the expected probabilities $E_{TiE=1}\{P(T_i^E=1)\}$, $E_{TiE=1}\{P(T_i^E=1|\ f_i^{GC})\}$, and $E_{TiE=1}\{P(T_i^E=1|\ f_i^M, f_i^{GC})\}$. Notice that in our model $E_{TiE=1}\{P(T_i^E=1|\ f_i^M, f_i^{GC})\} = E_{TiE=1}\{P(T_i^E=1|\ f_i^1, f_i^{GC})\}$ since knowing both f_i^M and f_i^{GC} implies knowing both f_i^1 and f_i^{GC}. This implies that we do not need to consider α in our study. We set $\sigma_{GC} = \sigma_1 = 1$.

β	γ	a	b	$E_{TiE=1}\{P(T_i^E=1)\}$	$E_{TiE=1}\{P(T_i^E=1\|\ f_i^{GC})\}$	$E_{TiE=1}\{P(T_i^E=1\|\ f_i^M, f_i^{GC})\}$
0.3	0.2	0.05	2	0.065	0.096	0.114
0.3	0.2	0.03	2	0.039	0.058	0.068
0.3	0.2	0.01	4	0.029	0.109	0.184
0.3	0.2	0.05	10	0.129	0.450	0.761
0.6	0	0.05	2	0.101	0.315	0.315
0.6	0.2	0.05	1	0.061	0.090	0.094
0.3	0.2	0.05	2	0.065	0.095	0.113
0.3	0.5	0.05	2	0.098	0.135	0.305

Notice that knowing GC content increases the probabilities, and that knowing the melting function in addition increases them even more. The probabilities increase when b increases. Knowing GC content increases the probabilities most for large β values and further improvement when knowing both GC and melting is obtained with a large γ values. The expected probabilities may be too large, but this is easy to scale differently.

We use the following algorithm in R

```
fGC<-rnorm(n,sd=sdGC)    # n is number of base pairs
f1<-rnorm(n,sd=sd1)
ps2GC<-0
psGC<-0
ps2GC1<-0                # Sum of all probabilities squared
psGC1<-0                 # Sum of all probabilities
for (i in 1:n) {
   p<-a*exp(b*(be*fGC[i]+ga*f1)) # vector with p for all n f1 values
   p<-pmin(p,en)                 # p=min(p',1)
   s2<-sum(p**2)
   s<-sum(p)
   ps2GC1<-ps2GC1+s2             # add all p**2 values
   psGC1<-psGC1+s                # add all p values
   s<-s/n                        # average for all f1 values
   ps2GC<-ps2GC+s**2             # add all s**2 values, average over f1
   psGC<-psGC+s                  # add all s values
}
E<-psGC1/(n**2)         # average for all probabilities
EGC<-ps2GC/psGC         # average knowing only GC, averages over f1
EGC1<-ps2GC1/psGC1      # average knowing both GC and f1
```