Maximum likelihood/FRAN

Ruth Ripley

November 7, 2017

1. Introduction

The R function *maxlikec* (so named during development when I had other versions which did not call C!) is the function which controls the maximum likelihood simulations. It is used as an argument to the function *sienaModelCreate* and its name is stored as the element named FRAN of the model object. It has three (or 3+) modes: initial, to set up the C++ data structure, terminal (to tidy up) and ordinary (to do one complete simulation). (There is an extra mode for using multiple processors, which does some of the work of an initial call.) It uses C++ functions for most of its work.

The interface to C++ and to *robmon* is more or less the same as for *simstats0c*. The terminal call is the same as for *simstats0c* and is done by the function *terminateFRAN*. The initial call is process by the funciton *initializeFRAN*.

2. Initial call to maxlikec

Call *initializeFRAN*, which does the data set up in C++ as for *simstats0c* plus a call to C++ to initialise the chains.

Extra details beyond *simstats0c* processing:

1. Calculate *nrunMH* as the multiplication factor (stored on *x* or *z*) times the sum over dependent variables of the distances by wave. Store on *z*.
2. Store in C++ the parameters needed for maximum likelihood:
   (a) Values controlling length of permutation (on the input model)

   maximumPermutationLength
   minimumPermutationLength
   initialPermutationLength

   (b) Probabilities of the different MH steps (on the input model)

   insertDiagonalProbability
   cancelDiagonalProbability
   permuteProbability

insertPermuteProbability
deletePermuteProbability
insertRandomMissingProbability
deleteRandomMissingProbability

(c) Proportion of missing data for each period: calculate from the number of missing and non missing entries in each of the dependent variables, which are stored as attributes of the group (which is either input or created for convenience in *initializeFRAN*).

prmin
prmib

(d) Simple rates flag: TRUE unless any selected effects have type "rate" but are not basic rate effects.

3. **if** this is the first call, (i.e. to main process) **then**
   Set up a minimal chain: see section 2.1.
   Do a pre burnin for each chain: see section 2.2
   Do 500 normal MHsteps.
   Return minimal and post burnin chains as lists.
   Store the final chains on the model in C++, along with any intial and end state differences.
   else
   Copy the post burnin chains from the main process. Add them to FRANstore (for use in R, for debugging only?) and store on the model in C++.
   end if

2.1 Minimal Chain

for all periods **do**
for all variables **do**
for all actors **do**
if network variable **then**
for all alters present in out ties for one only of this period or the next **do**
create a ministep
end for
else {behavior}
If the values at either end are different, create enough plus or minus 1 steps to get from one to the other.
end if
end for
end for
In each case ignore structural links or values. We do not need them in the processing, although they should be added in before returning a chain to R.
TODO: this is not currently done.
if no constraints between the networks **then**
Create a chain from the ministeps in random order

else
repeat
for all ministeps **do**
Try to insert in a random position in the chain
If this fails, try to insert after any other for the same actor/alter combination in any network.
Finally try to insert before any other for the same actor and alter.
end for
until done as many loops as dependent variables
if any ministeps left **then**
stop with error
end if
end if
Initialize the variables for this period
Calculate the probabilities for the chain (section 3.2.9)
end for

2.2 Pre-burnin

repeat
do an insert diagonal step
until have rejected 5 steps
repeat
do an insert permute step with permutation length set to 1
until have rejected 5 steps

3. Simulation call

1. Copy over the updated parameters to the Model object.
2. Create a simulation object and set up, from stored values on the Model Object:

   simpleRates flag

   prmin Proportion of missing network data for this period

   prmib Proportion missing behavior data for this period

   chain for this period, terminal chain from last time.
3. Clear out the storage area for this chain.
4. Set up from the input parameters:

   addChainToStore

   needChangeContributions Set to 1 if either addChainToStore or needChangeContributions is 1 (not sure why!)

   returnChains Do we want the final chain returned

   returnDataFrame Return chain, if requested, as data frame rather than list.

deriv do we need to do derivatives (not in phase 2)

nrunMH Number of MH steps to do

5. Run Initialize method of the simulation object. This sets up the initial missing data lists (but no longer the initial state).

6. Reinstate the initial state by executing the vector of ministeps defining the differences. from the data.

7. If using multiple processes, we need to keep the same process for each wave, which happens by default in snow. Just don't try to use load-balancing.

8. Set up the probabilityArray (maybe I could not find anywhere to store this, but it also initialises the stores for acceptance and rejection statistics).

9. Calculate the chain probabilities (to use the new parameters)

10. Do *nrunMH* MH steps (section 3.1)

11. Set the flags *needScores* (always set to FALSE while doing the MH steps and TRUE at this point) and optionally *needDerivatives*.

12. Do a final pass of the chain calculating probabilities, scores and optionally derivatives.

13. Store the current chain on the model, after converting the initial state to a vector of ministeps defining the differences.

14. Calculate the end state differences. The latter are not currently used: if they were to be carried over as an initial state for the next wave the chain would need correcting. They are returned to R with the chain.

15. Return:

    scores These are the simulated targets

    derivatives Returned as a lower triangle of the matrix. (eventually!)

    acceptance and rejection statistics by variable. Permutation steps are recorded under the first variable. *misdat* steps are separated out.

    chain Optional, based on *returnChains*. Can select format based on returnDataFrame or not.

3.1 MH steps

1. Choose a steptype with probability as in the array

2. Do the step

3. Store whether accepted or rejected

3.2 Individual Steps

3.2.1 Insert Diagonal

1. Choose a random ministep in the chain, including the option of the final one.
2. Set data to state just before this ministep.
3. Choose variable and actor as for forward simulation.
4. If actor not active, quit (should never occur)
5. Calculate proposal probability

$$
\begin{aligned}
p(\tilde{v}) &= \text{pr of chain after insert} \\
&= \text{pr (chain length R+1)} \prod_{r=1}^{R+1} (\ \text{pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
p(v) &= \text{pr of chain before insert} \\
&= \text{pr (chain length R)} \prod_{r=1}^{R} (\ \text{pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
u(\tilde{v}|v) &= \text{pr of proposing this insertion|not there} \\
&= \frac{\text{pr(propose an insert) pr(this var) pr(this actor)}}{(R+1)} \\
u(v|\tilde{v}) &= \text{pr of proposing deletion|there} = \frac{\text{pr(propose a deletion)}}{\text{pr(this diagonal)}} \\
\frac{p(\tilde{v})u(v|\tilde{v})}{p(v)u(\tilde{v}|v)} &= \frac{\text{pr (chain length R+1)} \prod_{r=1}^{R+1} (\ \text{pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \times}{\text{(pr chain length R)} \prod_{r=1}^{R} (\ \text{pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \\
&\qquad \frac{\text{pr(delete diagonal) pr(this diagonal)}\times}{\text{pr(insert diagonal)pr(this position) pr(this var) pr(this actor)}}
\end{aligned}
$$

6. The product of probabilities all cancel out except the probability of the diagonal step inserted, and further the variable and actor choices cancel out, leaving only the choice probability of the diagonal step.

3.2.2 Cancel Diagonal

1. If no diagonal ministeps, quit
2. Find a random diagonal ministep
3. Calculate the proposal probability: as for insert diagonal but upside down.
4. If proposal probability greater than a random uniform, do the deletion

3.2.3 Permute

1. Find a random ministep not equal to the final dummy step.
2. Interval to permute is from here for $c0$ steps, stopping at end.
3. If interval has one or less elements, quit
4. Get a permutation and check if valid
5. If not valid, quit
6. Calculate proposal probability

$$
\begin{aligned}
p(\tilde{v}) &= \text{pr of chain after permute} \\
&= \text{pr (chain length R } \prod_{r=1}^{R} \text{(pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \\
p(v) &= \text{pr of chain before permute} \\
&= \text{pr (chain length R) } \prod_{r=1}^{R} \text{(pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \\
u(\tilde{v}|v) &= \text{pr of proposing this permutation|not done} \\
&= \text{pr(propose a permute) pr(this interval)} \\
u(v|\tilde{v}) &= \text{pr of proposing reverse permutation|done} \\
&= \text{pr(propose a permute)pr(this interval)} \\
\frac{p(\tilde{v})u(v|\tilde{v})}{p(v)u(\tilde{v}|v)} &= \frac{\text{pr (chain length R) } \prod_{r=1}^{R} \text{(pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))}}{\text{pr (chain length R) } \prod_{r=1}^{R} \text{(pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))}}
\end{aligned}
$$

7. Calculations of choice probabilities only need to be done for the interval permuted.
8. If proposal probability greater than a random uniform, do the permutation.

3.2.4 Insert missing or CCP and permute

1. Select a random real ministep of the chain: *miniStepA*
2. Get the chain to the state before *miniStepA*
3. Choose a variable using relative rates, calculated as in *simstatsc*
4. **if** uponly or downonly for this variable **then** {logically here, as our variable and actor choices are separate}
   Exit
   end if[Why do we need to exit for missings too?]
5. For the variable, choose an actor proportional to their personal lambda.

6. **if** actor not active **then** {will have rate 0}
   Exit
   end if

7. Choose change as in forward simulation.

8. Quit if any of the conditions in section 3.2.8 hold

9. **if** chosen change corresponds to a missing value at either end of the period **then**
   set miniStepB to be the dummy at the end.
   else
   set miniStepB to be the place for the matching CCP: chosen randomly strictly between miniStepA and the next ministep for the same variable/actor/(alter).
   end if

10. Find the interval to permute: From miniStepA.next forwards for a maximum of *c0* steps. Stop at ministep-just-before-miniStepB if get that far.

11. Shorten the interval by stopping if you find any duplicated non diagonal variable/actor/alter combinations.

12. Get a permutation and check whether the resulting chain is valid, calculating the chain probabilities as you go.

13. Quit if not valid

14. Otherwise calculate the proposal probability.

For a CCP:

$$
\begin{aligned}
p(\tilde{v}) &= \text{ pr of chain after insertions} \\
&= \text{pr (chain length R+2)} \prod_{r=1}^{R+2} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
p(v) &= \text{ pr of chain before insertions} \\
&= \text{ pr (chain length R)} \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
u(\tilde{v}|v) &= \text{pr of proposing insertions|not there} \\
&= \text{pr(insert permute) pr(these positions) pr(this var)}\times \\
&\quad \text{pr(this actor) pr(this choice)} \\
u(v|\tilde{v}) &= \text{pr of proposing deletions|there} \\
&= \text{pr(delete permute) pr(not select missing) pr(this CCP)} \\
\frac{p(\tilde{v})u(v|\tilde{v})}{p(v)u(\tilde{v}|v)} &= \frac{\text{pr (chain length R+2)} \prod_{r=1}^{R+2} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \times}{\text{(pr chain length R)} \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))}} \\
&\quad \frac{\text{pr(delete permute) pr(not select missing) pr(this CCP)}\times}{\text{pr(insert permute) pr(these positions) pr(this var)}} \\
&\quad \frac{1}{\text{pr(this actor) pr(this choice)}}
\end{aligned}
$$

For a missing data insertion:

$$
\begin{aligned}
p(\tilde{v}) &= \text{ pr of chain after insertion} \\
&= \text{ (pr chain length R+1)} \prod_{r=1}^{R+1} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
p(v) &= \text{ pr of chain before insertions} \\
&= \text{ (pr chain length R)} \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \\
u(\tilde{v}|v) &= \text{pr of proposing insertion|not there} \\
&= \text{pr(insert permute) pr(this position) pr(this var)}\times \\
&\quad \text{pr(this actor) pr(this choice)} \\
u(v|\tilde{v}) &= \text{pr of proposing deletions|there} \\
&= \text{pr(delete permute) pr(select missing) pr(this missing one)} \\
\frac{p(\tilde{v})u(v|\tilde{v})}{p(v)u(\tilde{v}|v)} &= \frac{\text{(pr chain length R+1)} \prod_{r=1}^{R+1} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{)}) \times}{\text{(pr chain length R)} \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))}} \\
&\quad \frac{\text{pr(delete permute) pr(select missing) pr(this missing)}\times}{\text{pr(insert permute) pr(this position) pr(this var)}} \\
&\quad \frac{1}{\text{pr(this actor) pr(this choice)}}
\end{aligned}
$$

The probability that the chain contains R events is either a Poisson with

mean the constant rate parameter:

$$\exp(-n\alpha(t_2 - t_1))\frac{(n\alpha(t_2 - t_1))^R}{R!} \quad (1)$$

or approximated by a Gaussian with mean the sum of the reciprocal rates and variance the sum of the squares of the reciprocal rates.

In each case the product terms will cancel for the part of the chain before the first insertion and after the final insertion (if there is one).

It is necessary to know the number of CCP's which will exist after the insertions: the only reliable way I found was to do the insertions and find out.

15. If this probability is bigger than a random uniform, do the insertion(s).

3.2.5 Delete missing or CCP and permute

1. Decide whether to delete a CCP or a ministep corresponding to a missing observation (referred to as a missing ministep in what follows). Do the latter with probability *prmin* + *prmib* (This does not seem sensible, I would suggest using the mean rather than the sum. In an extreme case you could have the sum greater than 1).
2. **if** deleting a missing ministep **then**
   Decide on network or behavior using *prmin:prmib*.
   Pick a random missing ministep from network or behavior.
   If there are none of this type, quit.
   else
   Pick a random CCP
   end if

   Pick a random missing ministep or CCP as appropriate. If the former, if it happens to be the last step in the chain, quit.
3. Set ministepA to be the ministep after the random missing or the first of the CCP.
4. **if** deleting a missing ministep **then**
   set miniStepB to be the dummy at the end.
   else
   set miniStepB to be the ministep after the second of the CCP.
   end if
5. Find the interval to permute: From miniStepA.next forwards for a maximum of *c0* steps. Stop at ministep-just-before-miniStepB if get that far.
6. Shorten the interval by stopping if you find any duplicated non diagonal variable/actor/alter combinations.

7. Get a permutation and check whether the resulting chain is valid, calculating the chain probabilities as you go.
8. Quit if not valid
9. Calculate the proposal probabilities: same as for insert permute but upside down.
10. If proposal probability is greater than a random uniform, do the deletion(s).

3.2.6 Insert initial missing

1. If no missing values quit.
2. Select a random missing option (variable, actor, plus alter if applicable).
3. **if** behavior **then**
   Select up or down at random, but if this goes outside range, try the other way.
   end if
4. Select ministepA, a random ministep not after the first occurrence of this option, if any. The balancing ministep will be inserted before this ministep.
5. Check that it is valid to alter the initial values of the network by a toggle or the reverse of the direction selected, and insert the new ministep before ministepA. Note that you do not need to enforce upOnly and downOnly on the change in the initial state.
6. If not valid, and a behavior option, try the reverse change (unless you know it won't be accepted because you have already been forced to change it!)
7. If not valid, quit
8. Proposal probability for behavior direction should be set to 1 unless a genuine choice was accepted, when it should be 0.5.

9. Calculate the proposal probability:

$$
\begin{aligned}
p(\tilde{v}) &= \text{ pr of chain after changes} \\
&= \text{ pr(new val of miss item) pr (chain length R+1)} \times \\
&\quad \prod_{r=1}^{R+1} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \\
p(v) &= \text{ pr of chain before changes} \\
&= \text{ pr(orig val of miss item)pr (chain length R)} \times \\
&\quad \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \\
u(\tilde{v}|v) &= \text{pr of proposing changes|not there} \\
&= \text{pr(insert missing) pr(this missing option) pr(direction) pr(ministep here)} \\
u(v|\tilde{v}) &= \text{pr of proposing undoing changes|there} \\
&= \text{pr(delete missing) pr(this missing option)} \\
\frac{p(\tilde{v})u(v|\tilde{v})}{p(v)u(\tilde{v}|v)} &= \frac{\text{pr (chain length R+1) } \prod_{r=1}^{R+1} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))} \times}{\text{pr (chain length R) } \prod_{r=1}^{R} (\text{ pr(var}_r\text{)pr(actor}_r\text{)pr(choice}_r\text{))}} \\
&\quad \frac{\text{pr(delete missing) pr(new val)}}{\text{pr(insert missing) pr(direction) pr(ministep here) pr(orig val)}}
\end{aligned}
$$

10. Probabilities for the different values of the missing item are set to the proportion of non-missing values in the variable at the start of the period which are equal to the value in question.

11. The product of probabilities of ministeps will cancel out after ministepA.

12. If proposal probability is greater than a random uniform, do the changes.

3.2.7 Delete initial missing

1. If no missing values, quit

2. Select a random missing option.

3. If there are no ministeps for this option, quit

4. If a behavior option, calculate what the direction probability would have been at insertion.

5. Calculate how many places could have been chosen for the position at insertion.

6. Check that doing a reverse change to the initial value and deleting the ministep is valid.

7. If not valid, exit

8. Calculate proposal probability. Same as insert missing, but upside down.

9. If proposal probability is greater than a random uniform, do the changes.

3.2.8 quits

if diagonal **then**
  Exit
end if
if network and structurally fixed link **then** {we exclude these: need to sort this out }
  Exit
end if
if behavior and structurally fixed **then** {will have rate 0}
  Exit
end if
if behavior and goes over end **then** {will be diag or down as not permissible change}
  Exit
end if
if same variable/actor/(alter) as miniStepA **then**
  Exit
end if

3.2.9 Update chain probabilities

This reinitialises the data, but no longer the initial state.

3.3 Calculation of scores and derivatives

Scores are as for method of moments. For the rates, set τ equal to 1 over the number of real events in the chain. (Not sure about structurals here!)

Using similar notation, with $v_{rk_1k_2}$ the derivative of the score corresponding to$(\theta_{k_1}, \theta_{k_2})$ in the r-th chain,

for a basic rate parameter, (λ), for a dependent variable, not the total here, just the per actor rate

$$v_{rkk} = \sum_{m=1}^{M} \text{nbr of non-structurally determined links in the chain}/\lambda_m^2$$

for other parameters

$$v_{rk_1k_2} = \sum_{m=1}^{M} \left[\left(\sum_{a,\mathit{delta}_a} s_{i,a\delta k_a} p_{i,a\delta} \right) \left(\sum_{b,\mathit{delta}_b} s_{i,b\delta k_b} p_{i,a\delta} \right) - \sum_{a,\mathit{delta}} s_{i,a\delta k_1} p_{i,a\delta} s_{i,a\delta k_2} \right]$$

with derivative matrix

$$D_{jk} = \frac{1}{R} \sum_{r=1}^{R} v_{rjk}$$

Note that rate effects are uncorrelated with all other effects.