Package ltable 2.0.4. Part 5.

Ocheredko Oleksandr

Content:

FUNCTIONALITY

1. Constructs tables of counts and proportions out of data sets.
2. Inserts table into Excel and Word documents using clipboard, into LaTeX, HTML, Markdown and reStructuredText documents by the knitr::kable agency.
3. Molds table into acceptable for log-linear modeling data.frame, co.
4. Performs log-linear modeling.
5. Performs power analysis.

- This version is coded in R language exclusively to support across-systems portability.
- Log-linear and power analyses are enhanced with ability to model risks (rates) and relative risks (standardized ratios). Modelling survival data with interval censoring is also supported.
- MCMC stationarity diagnostic based on renewal theory is served by function *renewal()*.

MCMC stationarity diagnostic based on renewal theory

The basis of renewal theory is the definition of small sets as atoms with non-zero probabilities. Theory can serve the purpose of convergence control through small sets as in Robert,Ch.P.(1995) [1]. The use of small sets (hereafter I address them as states) are wider and they can be effectively applied as discretization technique.

The method has two main merits. Firstly, it is couched into strong mathematical reasoning and less influenced by analyst's experience. Secondly, the resultant solutions are not asymptotic, therefore not biased by inherent error of limit theorems.

Technique is based on secondary chains. First we code MCMC chain into states. Suppose we randomly choose state A_{10}. Secondary chain *(S_i)* is a fragment of original chain and includes chain of states between consecutive realizations of A_{10}. Given index *i*, it's fragment from the i^{th} instance of A_{10} to the $(i+1)^{th}$.

First of all, algorithm finds *nstat* that is the position in original chain when we are close enough (bounded by ϵ_1) to stationary distribution π_X, i.e.,

$$\|P_{x0}^{nstat} - \pi_X\| \leqslant \epsilon_1 \tag{1}$$

where $\|...\|$ is total variation norm. Therefore from position *nstat* on one may concider MCMC chain as being distributed almost from a stationary distribution π_X.

Secondly, we would like to achieve sufficiently small (bounded by ϵ_2) variance of the estimator *h(X)*. It can be done by finding the quantity *nVar* fulfilling the condition where $\ell = nVar - nstat$:

$$Var(\frac{1}{\ell}\sum_{nstat+1}^{nVar} h(X_k) - E_{\pi_X} h(X)) \leqslant \epsilon_2 \tag{2}$$

By finding ℓ we have a necessary length of original chain started at position *nstat* that satisfies sufficiently small variance of the estimator.

Conditional distribution of X|Y has the same information as marginal distribution of X given the knowledge on p(y):

$$\pi_X(x) = \int_y p(x)_{X|Y}(x|y) p_Y(y) dy$$

So one can use any state Y as the basis to approximate $\pi_X(x)$ of the chain. Useful lemma used by algorithm gives us conditional formulation of (1). Suppose that χ is a finite space and A_1 is a known atom in χ. If we denote $P_{A_1}^n(X)$ the conditional probability of having state X at position *n* of chain given

[1] Robert,Ch.P.(1995) Convergence Control Methods for Markov Chain Monte Carlo Algorithm. Statistical Science 10, 3.

state A at beginning (A_1) we have

$$\sum_{X\epsilon\chi} |P^n_{A_1}(X) - E_{\pi_X|A_1}| \leqslant 2P_x(\zeta^{(1)}_1 \geqslant n) + \sum_{j=0}^{n-1} P_x(\zeta^{(1)}_1 = j) * \left(\sum_{k=1}^{n-j-1} |P^k_{A_1}(A_1) - \pi_X(A_1) \right| \cdot P_{A_1}\left(\zeta^{(1)}_1 \geqslant n-j-k\right) + \pi_X(A_1) E_{A_1}(\zeta^{(1)}_1 - (n-j))_+ \Big) \tag{3}$$

Given atom A_1 for any state X in χ

$$\pi_X|A_1 = \pi_X(A_1) \sum_{n=0}^{\infty} P_{A_1}(X_n = X, \zeta^{(1)}_1 \geq n) \tag{4}$$

STEP1 To turn (4) to practicality they elaborated 3 obvious inequalities:

$$\|P^n_{A_1}(A_1) - \pi_X(A_1)\| \leqslant M_1 r_1^{-n} \tag{5}$$

$$P_{A_1}(\zeta^{(1)}_1 \geqslant n) \leqslant M_2 r_2^{-n} \tag{6}$$

$$P_x(\zeta^{(1)}_1 = n) \leqslant M_3 r_3^{-n} \tag{7}$$

After applying (5-7) to inequality (3) they arrived at (8)

$$\frac{2M_3 r_3^{1-n}}{r_3 - 1} + \frac{\pi_X(A_1) M_2 M_3 r_3 (r_3^{-n} - r_2^{-n}}{(r_2 - 1)(r_2 - r_3)} + \frac{M_1 M_2 M_3}{(r_2 - r_1)} \left(\frac{r_1 r_3 (r_3^{-n} - r_1^{-n})}{(r_1 - r_3)} + \frac{r_2 r_3 (r_3^{-n} - r_2^{-n})}{(r_3 - r_2)} \right) \leqslant \epsilon_1 \tag{8}$$

Using (8) one can ascertain the starting point in chain satisfying (1) with given precision bound of ϵ_1

STEP2 Besides starting point in chain we want to have the length of the chain to satisfy (2). Instrumental idea is to use (2) strightforwardly to estimate ℓ. I devided chain on secondary chains (S_i) and applied the equality of variance formulation for iid z:

$$Var(\sum(z_1, z_2, ...z_n)) = Var(z_1) + Var(z_2) +, ..., Var(z_n)$$

But instead of variances we add up sums of squares (SS), which by same logic are independent across secondary chains. We have as many components as there are secondary chains. Finally we can choose lenght of chain that satisfies (2), i.e.

$$\ell = \sqrt{\frac{\sum_{k=1}^{nVar} SS_k}{\epsilon_2}} \tag{9}$$

Estimation of $r_1, r_2, r_3, M_1, M_2, M_3$ is based on function nloptr() rendered by <nloptr> package, my gratitude goes to Steven G. Johnson, Aymeric Stamm and team members.

Call supports several arguments: *renewal(x, StatesNum=10, Astate=NULL, nForStart = 3000, epsilon1=0.05, epsilon2=0.05)*

- *x* is the name of the numeric vector containing original chain
- *StatesNum* is positive integer that indicates the number of states to classify chain values into, max value is 100; to choose *StatesNum* one should consider the adequate number of unique intervals chain values can be suitably represented by
- *Astate* is positive integer from 1 to *StateNum* that defines state to base calculation on; there may be small differences in results under different base states, see examples for details; by default it is median state value
- *nForStart* is positive integer that indicates the maximum distance from beginning of chain to consider in finding *nstat* value
- *epsilon1* is value of closeness to stationary distribution at position *n* of chain, it is used to find *nstat*, see (1)
- *epsilon2* is upper bound of variance of estimator, it is used to find *nVar*, see (2)

EXAMPLES

```
require(ltable)
data(tdata, package="ltable")
res1<-MCLogLin(Counts~smoker+contraceptive+
      tromb +contraceptive*tromb, data=
      tdata, draw=5000, burnin=500)
```

```
renewal(res1[,1], StatesNum=10, Astate=1)
```

```
[1] 2400  694
```

```
renewal(res1[,1], StatesNum=10, Astate=5)
```

```
[1] 1674  549
```

```
renewal(res1[,1], StatesNum=10, Astate=10)
```

```
[1] 2400  489
```

```
renewal(res1[,14], StatesNum=10, Astate=2)
```

```
[1] 1516  908
```

```
renewal(res1[,14], StatesNum=10, Astate=6)
```

```
[1] 2400 1084
```

```
renewal(res1[,14], StatesNum=10, Astate=9)
```

```
[1] 2400  962
```

```
renewal(res1[,1], StatesNum=50, Astate=1)
```

```
[1] 2400 3141
```

```
renewal(res1[,1], StatesNum=50, Astate=5)
```

```
[1] 2400 3141
```

```
renewal(res1[,1], StatesNum=50, Astate=10)
```

```
[1] 2400 3141
```

```
renewal(res1[,14], StatesNum=50, Astate=2)
```

```
[1] 2400 5437
```

```
renewal(res1[,14], StatesNum=50, Astate=6)
```

```
[1] 2400 5295
```

```
renewal(res1[,14], StatesNum=50, Astate=9)
```

```
[1] 2400 5485
```

I demonstrate the functionality with results of applying *MCLogLin()* to data *tdata* saved to object *res1* of class "matrix" "array" that contains 5000 sampled values of 14 model parameters stored by columns. First parameter is *lambda1*, last is *phi*, counts heterogeneity factor. Data and parameters are described in Part 1. From the outputs one can ascertain minor variability dependent on chosen state as the basis for calculus. Conspicuous augmentation in required length of chain relates to enlargement in number of considered states which is obvious. More states lead to more lengthy secondary chains with larger SS_i. Even more expressed effect is rendered by length of original chain. Let's look at the results of the same model but with samples of size 20000.

```
require(ltable)
data(tdata, package="ltable")
res2<-MCLogLin(Counts~smoker+contraceptive+
      tromb +contraceptive*tromb, data=
      tdata, draw=20000, burnin=500)
```

```
renewal(res2[,1], StatesNum=10, Astate=1)
```

```
[1] 2400 1374
```

```
renewal(res2[,1], StatesNum=10, Astate=5)
```

```
[1] 1668 1192
```

```
renewal(res2[,1], StatesNum=10, Astate=10)
```

```
[1] 2400 1403
```

```
renewal(res2[,14], StatesNum=10, Astate=2)
```

```
[1] 1518 2122
```

```
renewal(res2[,14], StatesNum=10, Astate=6)
```

```
[1] 2400 2675
```

```
renewal(res2[,14], StatesNum=10, Astate=9)
```

```
[1] 2400 2913
```

```
renewal(res2[,1], StatesNum=50, Astate=1)
```

```
[1] 2400 6894
```

```
renewal(res2[,1], StatesNum=50, Astate=5)
```

```
[1] 2400 6894
```

```
renewal(res2[,1], StatesNum=50, Astate=10)
```

```
[1] 2400 7083
```

```
renewal(res2[,14], StatesNum=50, Astate=2)
```

```
[1]  2400 15619
```

```
renewal(res2[,14], StatesNum=50, Astate=6)
```

```
[1]  2400 14876
```

```
renewal(res2[,14], StatesNum=50, Astate=9)
```

```
[1]  2400 14984
```

This regularity one can explain by STEP2 algorithm. The lengthier the chain the more secondary chains the more SS_i addends are there in nominator of (9). Should one worry about such dependency? Personally me, I don't think so. If criteria ϵ_1, ϵ_2 are satisfied one can use the chain starting from *nstat* of recommended length or lengthier. One should worry if the required length surpasses length of chain. This bears evidence on failure of the model to produce chain satisfying bound criterion of variance. One can not resolve the situation by supplying lengthier chain to *renewal()*, the result would not change.

Dependencies of *nstat* and ℓ on criteria ϵ_1, ϵ_2 for chain of parameter *lambda1* for sample size 5000 and basis state 5 are graphed below.

```
library(ggplot2)
library(grid)
nstat<-function(eps){
ltable::renewal(x=res1[,1], nForStart=3000,
                Astate=5, epsilon1=eps)[1]
}
chainlength<-function(eps){
ltable::renewal(x=res1[,1], nForStart=3000,
                Astate=5, epsilon2=eps)[2]
}
y<-seq(from=0.1,to=0.0001,by=-0.005)

x1<-sapply(y,  nstat)
x2<-sapply(y,  chainlength)
a<-ggplot(data.frame(epsilon1=y, nstat=x1),
          aes(nstat,epsilon1)) +geom_line()
b<-ggplot(data.frame(epsilon2=y, length=x2),
          aes(length,epsilon2)) +geom_line()
```

```
grid.newpage()
pushViewport(viewport(layout = grid.layout(2,1)))
print(a, vp = viewport(layout.pos.row = 1))
print(b, vp = viewport(layout.pos.row = 2))
```