# 二段階二肢選択CVMにおける提示額数・配布部数設計に関するS-PLUS関数*

寺脇 拓†

以下に示すS-PLUS[1]の関数群は，モンテカルロ実験により，二段階二肢選択CVMにおいて，提示額数・配布部数の選択が，WTP推定の誤差にどのような影響を与えるかを分析するためのものである．この関数により，提示額の数を変えたとき，そして標本サイズを変えたときに，二段階二肢選択CVMのパラメトリック推定法，ノンパラメトリック推定法それぞれのもとで推定されるWTP推定量のバイアス，分散，MSEがどのように変化するかをみることができる．但し，次の制約がある．

- 真の分布は，ワイブル分布か対数ロジステック分布のいずれかに限られる．
- 二段階目の提示額については，$k$ 番目の初期提示額を $B^k$，その二段階目のより高い提示額を $B^{ku}$，より低い提示額を $B^{kd}$ とすると，$B^{ku} = B^{k+1}$，$B^{kd} = B^{k-1}$ となるように設定される．
- 初期提示額数を $M$ として，提示額は $M + 2$ 等確率分位点を用いて設計される[2]．
- 初期提示額当たりの配布数は全て同一と設定される．例えば，標本サイズを100，初期提示額数を4とするとき，初期提示額当たりの配布数は，全ての初期提示額について25となる．
- パラメトリック推定法は，WTPの分布を対数ロジスティック分布としたもの，対数正規分布としたもの，ワイブル分布としたものの三種類に限られる．以下，これらをそれぞれ対数ロジスティックモデル，対数正規モデル，ワイブルモデルと呼ぶ．
- ノンパラメトリック推定法は，寺脇 [1] によるものである．以下，これをノンパラメトリックモデルと呼ぶ．

作業手順は次のようである．

1. 表1に表される10の関数を定義する．
2. `dbdc.mont.exp.1(M, N, B, model, p.gamma, p.lambda, trun.v)` の関数によりモンテカルロ実験を実行し，結果として作成されるデータセットをSオブジェクトに格納する．各引数のもつ意味は次の通り，
   - `M`：反復数
   - `N`：標本サイズ
   - `B`：初期提示額の数（最大18）．
   - `model`：真の分布をワイブル分布と仮定するとき1，対数ロジスティック分布と仮定するとき2．
   - `p.gamma`, `p.lambda`：仮定される分布のパラメータ

---

*Version: 平成13年5月20日 

†立命館大学経済学部
E-mail: masala.terawaki@nifty.ne.jp
URL: http://www.taku-t.com/

[1]S-PLUSは Insightful Corporation（2001年より，MathSoft Inc. から社名変更）の製品である．計算は，日本語版のVersion4.0J for Windowsで行っている．日本でのS-PLUSの入手方法については，(株) 数理システムのホームページ `http://www.msi.co.jp/`を参照されたい．

[2]初期提示額数を $M$ のとき，最大初期提示額のより高い二段階目の提示額，最小初期提示額のより低い二段階目の提示額を含めると，提示額数は合計 $M + 2$ となる．

– ワイブル分布の密度関数

$$f(WTP) = \frac{\texttt{p.gamma}}{\texttt{p.lambda}} \left( \frac{WTP}{\texttt{p.lambda}} \right)^{\texttt{p.gamma}-1} \exp\left\{ - \left( \frac{WTP}{\texttt{p.lambda}} \right)^{\texttt{p.gamma}} \right\}$$

– 対数ロジスティック分布の密度関数

$$f(WTP) = \frac{\texttt{p.gamma} \exp(-\texttt{p.lambda}) WTP^{\texttt{p.gamma}-1}}{\{1 + \exp(-\texttt{p.lambda}) WTP^{\texttt{p.gamma}}\}^2}$$

- `trun.v`：分布の切断値．当然，最高提示額よりも大きな値として設定される必要がある．切断しない場合は `Inf` となるが，ノンパラメトリック生存関数が収束しない場合，その平均値は無限大となることに注意．

例えば，形状パラメータ 0.8，尺度パラメータ 22026.47 のワイブル分布を仮定し，初期定額数を 4，標本サイズ 100，切断値を 100000 としたときの実験を 10 回繰り返すとき，そしてその結果を S データフレームオブジェクト `wei.4.100` に格納するとき，コマンドは次のようになる．

```
wei.4.100 <- dbdc.mont.exp.1(10, 100, 4, 1, 0.8, 22026.47, 100000)
```

3. 最後に，`result.dbdc.mont.exp.1(X, model, p.gamma, p.lambda)` の関数により，モンテカルロ実験結果を要約する．各引数のもつ意味は次の通り，
   - `X`：`dbdc.mont.exp.1` により作成された S データフレームオブジェクト
   - `model`：真の分布をワイブル分布と仮定するとき `1`，対数ロジスティック分布と仮定するとき `2`.
   - `p.gamma`, `p.lambda`：仮定される分布のパラメータ．`dbdc.mont.exp.1` と同様．

結果，ノンパラメトリックモデル（`Nonp`），対数ロジスティックモデル（`logis`），対数正規モデル（`norm`），ワイブルモデル（`Wei`）ごとに，平均値推定量の平均値（`mean.mean`），分散（`mean.var`），標準偏差（`mean.s.d.`），バイアス（`mean.bias`），$10^6$ で除した MSE（`mean.mse-6`），そして中央値推定量の平均値（`median.mean`），分散（`median.var`），標準偏差（`median.s.d.`），バイアス（`median.bias`），$10^6$ で除した MSE（`median.mse-6`）がそれぞれ計算される．

## 引用文献

[1] 寺脇拓「都市近郊農業の外部経済効果の計測－二段階二肢選択 CVM における WTP 分布のノンパラメトリック推定－」『農業経済研究』，第 69 巻第 4 号，1998，pp.201~212.

表 1:  S-PLUS 関数

```
 1 :  ######## S-PLUS functions for analysing the influence of sample size
 2 :  ######## and the number of bids on the error of WTP estimation
 3 :  ######## in double-bounded dichotomous choice CVM
 4 :  ######## (C) 2001 Taku TERAWAKI
 5 :  ######## E-MAIL: masala.terawaki@nifty.ne.jp
 6 :  ######## URL: http://www.taku-t.com/
 7 :  ##### likelihood functions in parametric model #####
 8 :  ### log-logistic dist.  ###
 9 :  DBDCL.obj <- function(r, X)
10 :  {
11 :          W2H <- (log(X$BIDU) - r[1])/r[2]
12 :          W <- (log(X$BID) - r[1])/r[2]
13 :          W2L <- (log(X$BIDD) - r[1])/r[2]
14 :          P.yy <- log(1 - plogis(W2H))
15 :          P.yn <- log(plogis(W2H) - plogis(W))
16 :          P.ny <- log(plogis(W) - plogis(W2L))
17 :          P.nn <- log(plogis(W2L))
18 :          obj <- X$Y1 * X$Y2 * P.yy + X$Y1 * (1 - X$Y2) * P.yn + (1 - X$Y1) *
19 :                  X$Y2 * P.ny + (1 - X$Y1) * (1 - X$Y2) * P.nn
20 :  - sum(obj)
21 :  }
22 :  ### log-normal dist.  ###
23 :  DBDCP.obj <- function(r, X)
24 :  {
25 :          W2H <- (log(X$BIDU) - r[1])/r[2]
26 :          W <- (log(X$BID) - r[1])/r[2]
27 :          W2L <- (log(X$BIDD) - r[1])/r[2]
28 :          P.yy <- log(1 - pnorm(W2H))
29 :          P.yn <- log(pnorm(W2H) - pnorm(W))
30 :          P.ny <- log(pnorm(W) - pnorm(W2L))
31 :          P.nn <- log(pnorm(W2L))
32 :          obj <- X$Y1 * X$Y2 * P.yy + X$Y1 * (1 - X$Y2) * P.yn + (1 - X$Y1) *
33 :                  X$Y2 * P.ny + (1 - X$Y1) * (1 - X$Y2) * P.nn
34 :  - sum(obj)
35 :  }
36 :  ### weibull dist.  ###
37 :  DBDCW.obj <- function(r, X)
38 :  {
39 :          W2H <- (log(X$BIDU) - r[1])/r[2]
40 :          W <- (log(X$BID) - r[1])/r[2]
41 :          W2L <- (log(X$BIDD) - r[1])/r[2]
42 :          P.yy <- log(exp( - exp(W2H)))
43 :          P.yn <- log( - exp( - exp(W2H)) + (exp( - exp(W))))
44 :          P.ny <- log( - exp( - exp(W)) + (exp( - exp(W2L))))
45 :          P.nn <- log(1 - exp( - exp(W2L)))
46 :          obj <- X$Y1 * X$Y2 * P.yy + X$Y1 * (1 - X$Y2) * P.yn + (1 - X$Y1) *
47 :                  X$Y2 * P.ny + (1 - X$Y1) * (1 - X$Y2) * P.nn
48 :  - sum(obj)
49 :  }
50 :  ##### the functions for calculating representative values in nonparametric model #####
51 :  ### mean ###
52 :  nonpara.mean.wtp <- function(bids, est.prob)
53 :  {
54 :          m <- length(est.prob) - 1
55 :          area <- 0
56 :          for(i in 1:m) {
57 :                  area <- area + ((bids[i + 1, ] - bids[i, ]) * (est.prob[
58 :                          i + 1] + est.prob[i]))/2
59 :          }
60 :          area <- as.data.frame(area)
61 :          area
62 :  }
63 :  ### median ###
64 :  nonpara.median.wtp <- function(bids, est.prob)
65 :  {
```

```
 66 :          m <- length(est.prob) - 1
 67 :          for(i in 1:m) {
 68 :                  if(est.prob[i] <= 0.5)
 69 :                          break
 70 :                  median.wtp <- ifelse(est.prob[i + 1] == est.prob[i], 1, ((
 71 :                          bids[i + 1, ] - bids[i, ])/(est.prob[i + 1] -
 72 :                          est.prob[i]) * (0.5 - est.prob[i])) + bids[i, ])
 73 :          }
 74 :          median.wtp <- as.data.frame(median.wtp)
 75 :          median.wtp
 76 :  }
 77 :  ##### survival functions in parametric model ####
 78 :  ### log-logistic dist.  ###
 79 :  param.surv.l <- function(Z, Xmax, cons, sigm)
 80 :  {
 81 :          plogis((cons - log(Z))/sigm)/(1 - plogis((cons - log(Xmax))/sigm)
 82 :                  )
 83 :  }
 84 :  ### log-normal dist.  ###
 85 :  param.surv.n <- function(Z, Xmax, cons, sigm)
 86 :  {
 87 :          pnorm((cons - log(Z))/sigm)/(1 - pnorm((cons - log(Xmax))/sigm))
 88 :  }
 89 :  ### weibull dist.  ###
 90 :  param.surv.w <- function(Z, Xmax, cons, sigm)
 91 :  {
 92 :          exp( - exp((log(Z) - cons)/sigm))/(1 - exp( - exp((log(Xmax) -
 93 :                  cons)/sigm)))
 94 :  }
 95 :  ##### the function for a monte carlo experiment #####
 96 :  dbdc.mont.exp.1 <- function(M, N, B, model, p.gamma, p.lambda, trun.v)
 97 :  {
 98 :  ## M : the number of iteration
 99 :  ## N : sample size
100 :  ## B : the number of initial bids (which is allowed up to 18)
101 :  ## model :  model=1 means weibull model, model=2 means loglogistic model.
102 :  ## p.gamma, p.lambda :  the parameters in the following density functions
103 :  ## ---weibull model
104 :  ## f(x)={p.gamma/p.lambda*(x/p.lambda)^(p.gamma-1)}*{exp-(x/p.lambda)^p.gamma}
105 :  ## ---loglogistic model
106 :  ## f(x)={p.gamma*exp(-p.lambda)*x^(p.gamma-1)}/{1+exp(-p.lambda)*x^p.gamma}^2
107 :  ## trun.v :  the truncation value of survival functions ; the value must be more
108 :  ## than the highest bid.  also if the model has no truncation value,
109 :  ## then trun.v=Inf
110 :          cv.datum <- matrix(0, N, 6)
111 :          cv.datum <- as.data.frame(cv.datum)
112 :          var.name <- c("BIDD", "BID", "BIDU", "Y1", "Y2", "TWTP")
113 :          dimnames(cv.datum)[[2]] <- var.name
114 :          bid.vector <- matrix(0, B + 2, 1)
115 :          bid.vector <- as.data.frame(bid.vector)
116 :          mont.mean <- matrix(0, M, 4)
117 :          mont.mean <- as.data.frame(mont.mean)
118 :          mont.median <- matrix(0, M, 4)
119 :          mont.median <- as.data.frame(mont.median)
120 :          num.obs <- matrix(0, 4, B + 4)
121 :          num.obs <- as.data.frame(num.obs)
122 :          est.prob <- matrix(0, B + 4, 3)
123 :          est.prob <- as.data.frame(est.prob)
124 :          mont.est.prob <- matrix(0, M, B + 4)
125 :          mont.est.prob <- as.data.frame(mont.est.prob)
126 :          est.coef.objec <- matrix(0, M, 9)
127 :          est.coef.objec <- as.data.frame(est.coef.objec)
128 :          check.vec <- matrix(0, B + 1, 1)
129 :          check.vec <- as.data.frame(check.vec)
130 :          name.col <- c("Mean.Nonp", "Mean.logis", "Mean.norm", "Mean.Wei",
131 :                  Medi.Nonp, "Medi.logis", "Medi.norm", "Medi.Wei",
132 :                  a.loigs, "b.logis", "a.norm", "b.norm", "a.wei",
133 :                  b.wei, "like.logis", "like.norm", "like.wei")
134 :          bids.nonp.name <- c("p1", "p2", "p3", "p4", "p5", "p6", "p7",
135 :                  p8, "p9", "p10", "p11", "p12", "p13", "p14", "p15",
```

```
                p16, "p17", "p18", "p19", "p20", "p21", "p22")
## Mean.Nonp :  the estimated mean by nonparametric method
## Mean.logis :  the estimated mean by parametric method with loglogistic model
## Mean.norm :  the estimated mean by parametric method with lognormal model
## Mean.Wei :  the estimated mean by parametric method with weibull model
## Medi.Nonp :  the estimated median by nonparametric method
## Medi.logis :  the estimated median by parametric method with loglogistic model
## Medi.norm :  the estimated median by parametric method with lognormal model
## Medi.Wei :  the estimated median by parametric method with weibull model
## a.loigs, b.logis :  the parameter estimates in loglogistic model
## a.norm, b.norm :  the parameter estimates in lognormal model
## a.wei, b.wei :  the parameter estimates in weibull model
## like.logis :  maximum log-likelihood in loglogistic model
## like.norm :  maximum log-likelihood in lognormal model
## like.wei :  maximum log-likelihood in weibull model
## p1-p22 :  the estimates of survival probability in nonparametric model
        bids.nonp.name <- bids.nonp.name[1:(B + 4)]
        name.col <- c(name.col, bids.nonp.name)
        mean.logis <- p.lambda/p.gamma
        var.logis <- 1/p.gamma
        for(i in 1:(B + 2)) {
                bid.vector[i, ] <- if(model == 1) qweibull(i/(B + 3),
                                p.gamma, p.lambda) else exp(qlogis(i/(B +
                                3), mean.logis, var.logis))
        }
        for(i in 1:M) {
                for(j in 1:N) {
                        gen.elem <- rlogis(1, mean.logis, var.logis)
                        gen.data <- exp(gen.elem)
                        num.bid.vec <- ifelse(B == 1, 2, sample(2:(B + 1),
                                1))
                        bid.d <- bid.vector[num.bid.vec - 1, ]
                        bid.m <- bid.vector[num.bid.vec, ]
                        bid.u <- bid.vector[num.bid.vec + 1, ]
                        cv.datum[j, 1] <- bid.d
                        cv.datum[j, 2] <- bid.m
                        cv.datum[j, 3] <- bid.u
                        cv.datum[j, 4] <- if(bid.m <= gen.data) 1 else 0
                        cv.datum[j, 5] <- if((bid.m <= gen.data & bid.u <=
                                gen.data) | (bid.m > gen.data & bid.d <=
                                gen.data)) 1 else 0
                        cv.datum[j, 6] <- gen.data
                }
## Terawaki's nonparametric model ##
                for(k in 1:B) {
                        num.obs[1, k + 2] <- nrow(cv.datum[cv.datum[,
                                Y1] == 0 & cv.datum[, "Y2"] == 0 &
                                cv.datum[, "BID"] == bid.vector[k + 1, ],
                                ])
                        num.obs[2, k + 2] <- nrow(cv.datum[cv.datum[,
                                Y1] == 0 & cv.datum[, "Y2"] == 1 &
                                cv.datum[, "BID"] == bid.vector[k + 1, ],
                                ])
                        num.obs[3, k + 2] <- nrow(cv.datum[cv.datum[,
                                Y1] == 1 & cv.datum[, "Y2"] == 0 &
                                cv.datum[, "BID"] == bid.vector[k + 1, ],
                                ])
                        num.obs[4, k + 2] <- nrow(cv.datum[cv.datum[,
                                Y1] == 1 & cv.datum[, "Y2"] == 1 &
                                cv.datum[, "BID"] == bid.vector[k + 1, ],
                                ])
                }
                est.prob[1, 1] <- 1
                est.prob[1, 2] <- 1
                est.prob[1, 3] <- est.prob[1, 1]/est.prob[1, 2]
                for(k in 1:(B + 2)) {
                        est.prob[k + 1, 1] <- num.obs[4, k] + num.obs[3,
                                k + 1] + num.obs[4, k + 1] + num.obs[2, k +
                                2] + num.obs[3, k + 2] + num.obs[4, k + 2
                                ]
```

```
206 :                         est.prob[k + 1, 2] <- sum(num.obs[, k]) + sum(
207 :                                 num.obs[, k + 1]) + sum(num.obs[, k + 2])
208 :                         est.prob[k + 1, 3] <- est.prob[k + 1, 1]/est.prob[
209 :                                 k + 1, 2]
210 :                 }
211 :                 est.prob[B + 4, 1] <- 0
212 :                 est.prob[B + 4, 2] <- 1
213 :                 est.prob[B + 4, 3] <- est.prob[B + 4, 1]/est.prob[B + 4,
214 :                         2]
215 :                 for(k in 1:50) {
216 :                         for(l in 1:(B + 1)) {
217 :                                 check.vec[l, ] <- if(est.prob[l + 1, 3] >=
218 :                                 est.prob[l + 2, 3]) 0 else 1
219 :                         }
220 :                         if(sum(check.vec[, 1]) == 0)
221 :                                 break
222 :                         for(t in 2:(B + 3)) {
223 :                                 est.prob[t, 1] <- if(est.prob[t, 3] <
224 :                                 est.prob[t + 1, 3]) est.prob[t, 1] +
225 :                                 est.prob[t + 1, 1] else est.prob[t, 1
226 :                                 ]
227 :                                 est.prob[t, 2] <- if(est.prob[t, 3] <
228 :                                 est.prob[t + 1, 3]) est.prob[t, 2] +
229 :                                 est.prob[t + 1, 2] else est.prob[t, 2
230 :                                 ]
231 :                                 est.prob[t, 3] <- est.prob[t, 1]/est.prob[
232 :                                 t, 2]
233 :                                 est.prob[t + 1, 1] <- if(est.prob[t, 3] <
234 :                                 est.prob[t + 1, 3]) est.prob[t, 1] +
235 :                                 est.prob[t + 1, 1] else est.prob[t +
236 :                                 1, 1]
237 :                                 est.prob[t + 1, 2] <- if(est.prob[t, 3] <
238 :                                 est.prob[t + 1, 3]) est.prob[t, 2] +
239 :                                 est.prob[t + 1, 2] else est.prob[t +
240 :                                 1, 2]
241 :                                 est.prob[t + 1, 3] <- est.prob[t + 1, 1]/
242 :                                 est.prob[t + 1, 2]
243 :                         }
244 :                 }
245 :                 last.bid <- if(est.prob[B + 2, 3] != est.prob[B + 3, 3]) (
246 :                                 (bid.vector[B + 2, ] - bid.vector[B + 1,
247 :                                 ])/(est.prob[B + 3, 3] - est.prob[B + 2,
248 :                                 3]) * (0 - est.prob[B + 3, 3])) +
249 :                                 bid.vector[B + 2, ] else trun.v
250 :                 est.prob[B + 4, 3] <- if(last.bid > trun.v) ((est.prob[B +
251 :                                 3, 3] - est.prob[B + 2, 3])/(bid.vector[B +
252 :                                 2, ] - bid.vector[B + 1, ]) * (trun.v -
253 :                                 bid.vector[B + 2, ])) + est.prob[B + 3,
254 :                                 3] else est.prob[B + 4, 3]
255 :                 last.bid <- if(last.bid > trun.v) trun.v else last.bid
256 :                 bids.nonp <- rbind(0, bid.vector, last.bid)
257 :                 est.prob.nonp <- est.prob[, 3]
258 :                 last.prob <- est.prob[B + 4, 3]
259 :                 mont.nonp.mean <- nonpara.mean.wtp(bids.nonp,
260 :                         est.prob.nonp)/(1 - last.prob)
261 :                 mont.nonp.median <- nonpara.median.wtp(bids.nonp,
262 :                         est.prob.nonp)
263 :                 mont.mean[i, 1] <- mont.nonp.mean
264 :                 mont.median[i, 1] <- mont.nonp.median
265 :                 mont.est.prob[i, ] <- as.vector(est.prob[, 3])
266 :         ## parametric model ##
267 :                 r.prob <- c(10, 0.5)
268 :                 name <- c("CONS", "SIGM")
269 :                 names(r.prob) <- name
270 :                 mle.log <- nlminb(start = r.prob, obj = DBDCL.obj, X =
271 :                         cv.datum)
272 :                 mle.nor <- nlminb(start = r.prob, obj = DBDCP.obj, X =
273 :                         cv.datum)
274 :                 mle.wei <- nlminb(start = r.prob, obj = DBDCW.obj, X =
275 :                         cv.datum)
```

```
276 :                 est.coef.objec[i, (1:2)] <- mle.log$parameters
277 :                 est.coef.objec[i, (3:4)] <- mle.nor$parameters
278 :                 est.coef.objec[i, (5:6)] <- mle.wei$parameters
279 :                 est.coef.objec[i, 7] <- mle.log$objective
280 :                 est.coef.objec[i, 8] <- mle.nor$objective
281 :                 est.coef.objec[i, 9] <- mle.wei$objective
282 :                 coef.cons.l <- est.coef.objec[i, 1]
283 :                 coef.sigm.l <- est.coef.objec[i, 2]
284 :                 coef.cons.n <- est.coef.objec[i, 3]
285 :                 coef.sigm.n <- est.coef.objec[i, 4]
286 :                 coef.cons.w <- est.coef.objec[i, 5]
287 :                 coef.sigm.w <- est.coef.objec[i, 6]
288 :                 param.mean.l <- integrate(param.surv.l, 0.1, last.bid,
289 :                         Xmax = last.bid, cons = coef.cons.l, sigm =
290 :                         coef.sigm.l)$integral
291 :                 param.mean.n <- integrate(param.surv.n, 0.1, last.bid,
292 :                         Xmax = last.bid, cons = coef.cons.n, sigm =
293 :                         coef.sigm.n)$integral
294 :                 param.mean.w <- integrate(param.surv.w, 0.1, last.bid,
295 :                         Xmax = last.bid, cons = coef.cons.w, sigm =
296 :                         coef.sigm.w)$integral
297 :                 param.median.l <- exp(coef.cons.l)
298 :                 param.median.n <- exp(coef.cons.n)
299 :                 param.median.w <- exp(coef.cons.w) * (( - log(0.5))^(
300 :                         coef.sigm.w))
301 :                 mont.mean[i, 2] <- param.mean.l
302 :                 mont.mean[i, 3] <- param.mean.n
303 :                 mont.mean[i, 4] <- param.mean.w
304 :                 mont.median[i, 2] <- param.median.l
305 :                 mont.median[i, 3] <- param.median.n
306 :                 mont.median[i, 4] <- param.median.w
307 :         }
308 :         RESULT <- cbind(mont.mean, mont.median, est.coef.objec,
309 :                 mont.est.prob)
310 :         names(RESULT) <- name.col
311 :         RESULT
312 : }
313 : ##### the function for summarizing a experiment result #####
314 : result.dbdc.mont.exp.1 <- function(X, model, p.gamma, p.lambda)
315 : {
316 : ## X : a S data object made by function of 'dbdc.mont.exp.1'
317 : ## model :  weibull model if model=1, loglogistic model if model=2
318 : ## p.gamma, p.lambda :  the parameters in the following density functions
319 : ## ---weibull model
320 : ## f(x)={p.gamma/p.lambda*(x/p.lambda)^(p.gamma-1)}*{exp-(x/p.lambda)^p.gamma}
321 : ## ---loglogistic model
322 : ## f(x)={p.gamma*exp(-p.lambda)*x^(p.gamma-1)}/{1+exp(-p.lambda)*x^p.gamma}^2
323 :         gen.stat.data.set <- X
324 :         true.surv.l <- function(x, a, b)
325 :         {
326 :                 1 - plogis( - b + a * log(x))
327 :         }
328 :         true.surv.w <- function(x, a, b)
329 :         {
330 :                 1 - pweibull(x, a, b)
331 :         }
332 :         mean.logis <- p.lambda/p.gamma
333 :         true.mean <- if(model == 1) integrate(true.surv.w, 0.1, Inf, a =
334 :                         p.gamma, b = p.lambda)$integral else integrate(
335 :                         true.surv.l, 0.1, Inf, a = p.gamma, b = p.lambda)$
336 :                         integral
337 :         true.median <- if(model == 1) p.lambda * log(2)^(1/p.gamma) else
338 :                         exp(mean.logis)
339 :         result.matrix <- matrix(0, 10, 4)
340 :         result.matrix <- as.data.frame(result.matrix)
341 :         name.col <- c("Nonp", "logis", "norm", "Wei")
342 : ## Nonp :  nonparametric model
343 : ## logis :  loglogistic model
344 : ## norm :  lognormal model
345 : ## Wei :  weibull model
```

```
346 :         name.row <- c("mean.mean", "mean.var", "mean.s.d.", "mean.bias",
347 :                 mean.mse-6, "median.mean", "median.var", "median.s.d.",
348 :                 median.bias, "median.mse-6")
349 :  ## mean.mean :  the mean of mean estimator
350 :  ## mean.var :  the variance of mean estimator
351 :  ## mean.s.d.  :  the standard deviation of mean estimator
352 :  ## mean.bias :  the bias of mean estimator
353 :  ## mean.mse-6:  the mse*10^(-6) of mean estimator
354 :  ## median.mean :  the mean of median estimator
355 :  ## median.var :  the variance of median estimator
356 :  ## median.s.d.  :  the standard deviation of median estimator
357 :  ## median.bias :  the bias of median estimator
358 :  ## median.mse-6:  the mse*10^(-6) of median estimator
359 :         for(i in 1:4) {
360 :                 result.matrix[1, i] <- mean(gen.stat.data.set[, i])
361 :                 result.matrix[2, i] <- var(gen.stat.data.set[, i])
362 :                 result.matrix[3, i] <- sqrt(var(gen.stat.data.set[, i]))
363 :                 result.matrix[4, i] <- result.matrix[1, i] - true.mean
364 :                 result.matrix[5, i] <- (result.matrix[4, i]^2 +
365 :                         result.matrix[2, i]) * 10^(-6)
366 :                 result.matrix[6, i] <- mean(gen.stat.data.set[, i + 4])
367 :                 result.matrix[7, i] <- var(gen.stat.data.set[, i + 4])
368 :                 result.matrix[8, i] <- sqrt(var(gen.stat.data.set[, i + 4
369 :                         ]))
370 :                 result.matrix[9, i] <- result.matrix[6, i] - true.median
371 :                 result.matrix[10, i] <- (result.matrix[9, i]^2 +
372 :                         result.matrix[7, i]) * 10^(-6)
373 :         }
374 :         dimnames(result.matrix) <- list(name.row, name.col)
375 :         result.matrix
376 :  }
```