S-PLUS・S+NUOPT でのプログラム

・データの生成

```
tmp.data.name = "koka.hajime"
tmp.data = get(tmp.data.name)
tmp.data = as.integer(tmp.data[,1])
tmp.data = data.frame ( times =
        as.integer(names(table(tmp.data))),sales=as.integer(table(tmp.data)))
tmp.data = merge ( data.frame ( times = seq ( from = 1 , to =
        max ( tmp.data$times ))), tmp.data,by.x = " times ", by.y=" times ", all=T )
tmp.data$sales [is.na(tmp.data$sales)] = 0
```

● 既存モデルのパラメータ推定および実データへの当てはめ

・S＋NUOPT での最小二乗法の定式化

```
min.res = function(times, sales){
        Time <- Set()
        param <- Set(1:4)

        timelength <- length(times)

        t <- Element(set = Time)
        j <- Element(set = param)

        times <- Parameter(list(1:timelength, times), index = t)
        sales <- Parameter(list(1:timelength, sales), index = t)

        beta <- Variable(index = param)
        beta[1] >= 0
        beta[2] >= 0
        beta[2] + beta[3] >= 0
        beta[4] >= 0
```

```
            ((beta[3]*beta[4]*times[t+1,t<timelength])/(beta[2]+beta[3])) +
                    (beta[4]/(beta[1]-beta[2]-beta[3]))*(((beta[3]-beta[1])/beta[1])*
                    (1-exp(-beta[1]*times[t+1,t<timelength])) +
                    ((beta[1]*beta[2])/(beta[2]+beta[3])^2)*(1-exp(-(beta[2]+beta[3])*
                    times[t+1,t<timelength]))) >=
                    ((beta[3]*beta[4]*times[t,t<timelength])/(beta[2]+beta[3])) +
                    (beta[4]/(beta[1]-beta[2]-beta[3]))*(((beta[3]-beta[1])/beta[1])*
                    (1-exp(-beta[1]*times[t,t<timelength])) +
                    ((beta[1]*beta[2])/(beta[2]+beta[3])^2)*
                    (1-exp(-(beta[2]+beta[3])*times[t,t<timelength])))

          r <- Expression(index = t)

           r[t] ~ sales[t] - ((beta[3]*beta[4]*times[t])/(beta[2]+beta[3])) -
                    (beta[4]/(beta[1]-beta[2]-beta[3]))*(((beta[3]-beta[1])/beta[1])*
                    (1-exp(-beta[1]*times[t])) +
                    ((beta[1]*beta[2])/(beta[2]+beta[3])^2)*(1-exp(-(beta[2]+beta[3])*times[t])))

          obj <- Objective(type="minimize")
          obj ~ Sum(r[t]*r[t],t)

          beta[1] ~ 0.07
          beta[2] ~ 0.5
          beta[3] ~ 0.5
          beta[4] ~ 0.1

}


pred = function(beta, tmp, k=0){
    SumOfSales = sum(tmp[,2])
    times = tmp[,1]

    y2 = (beta[1]*(1-k)*SumOfSales/(beta[1]-beta[2]-beta[3])) * ( -exp(-beta[1]*times) +
        exp(-(beta[2]+beta[3])*times))
```

```
    y4 = ((beta[2]*k + beta[3])*SumOfSales/(beta[2]+beta[3])) +
        ((beta[3]*(1-k)*SumOfSales)/(beta[2]+beta[3])*(beta[1]-beta[2]-beta[3]))*
        ((beta[2]+beta[3])*exp(-beta[1]*times) - beta[1]*exp(-(beta[2]+beta[3])*times))
        y = y2 + y4
    }


predd = function(beta, tmp, k=0){
        SumOfSales = sum(tmp[,2])
        times = tmp[,1]

        Y = (beta[3]*beta[4]*times/(beta[2]+beta[3])) +
            (beta[4]/(beta[1]-beta[2]-beta[3]))*(((beta[3]-beta[1])/beta[1])*(1-exp(-beta[1]*times))
            + ((beta[1]*beta[2])/(beta[2]+beta[3])^2)*(1-exp(-(beta[2]+beta[3])*times)))

        Y = SumOfSales*Y
}
```

・S+NUOPT による実データへの当てはめとパラメータの算出

```
nuopt.options(maxitn = 200, method ="trust", scaling="on", eps=1e-06)
result.problem = System(model = min.res, tmp.data$times,
    cumsum(tmp.data$sales)/sum(tmp.data$sales))
result.solution = solve(result.problem)
se = as.vector(result.solution$objective)


beta
```

・実際購買と推定結果のグラフ（図 3・図 4）

```
beta = as.double(result.solution$variables$beta$current)
plot(tmp.data$times,tmp.data$sales,type="l",col=1)
lines(tmp.data$times,pred(beta,tmp.data,k=0)/20,type="l",col=2)
```

```
plot(tmp.data$times,cumsum(tmp.data$sales),type="l",col=1)
lines(tmp.data$times,predd(beta,tmp.data,k=0),type="l",col=2)
```

・算出される結果

```
NUMBER_OF_VARIABLES                                   4
NUMBER_OF_FUNCTIONS                                 211
PROBLEM_TYPE                               MINIMIZATION
METHOD                                     TRUST_REGION
<preprocess begin>..........<preprocess end>
<iteration begin>
    res=1.1e+001 .... 1.5e-001 .... 2.6e-003 .... 3.8e-004 .... 3.9e-004 ....
        1.3e-003 .... 8.4e-004 .... 1.3e-003 .... 1.1e-005 .... 8.3e-007
<iteration end>
STATUS                                          OPTIMAL
VALUE_OF_OBJECTIVE                         0.2166693456
ITERATION_COUNT                                      45
FUNC_EVAL_COUNT                                      77
FACTORIZATION_COUNT                                 154
RESIDUAL                               8.281828955e-007
ELAPSED_TIME(sec.)                                 1.98

> beta
[1] 1.55333854 0.32685592 0.02756388 0.05574682
```

※ここで，VALUE_OF_OBJECTIVE の値が誤差二乗和（最小二乗誤差）の値である．

- ワイブル分布のモデルのパラメータ推定および実データへの当てはめ

・S＋NUOPT での最小二乗法の定式化

```
sales = as.integer(get(dataname)[,1])
sales = sales - min(sales) + 1

tmp = data.frame(times = as.integer(names(table(sales))),sales = as.integer(table(sales)))
model.data = merge(data.frame(times =
    seq(from=1,to=max(sales))),tmp,by.x="times",by.y="times",all=T)
```

```
model.data$sales[is.na(model.data$sales)] = 0
model.data$sales = cumsum(model.data$sales)

min.res = function(sales, times, mark.size){
    sales = sales/mark.size
    Time <- Set()
    param <- Set(1:2)
    timelength <- length(times)
    t <- Element(set = Time)
    j <- Element(set = param)

    sales <- Parameter(list(1:timelength,sales),index = t)
    times <- Parameter(list(1:timelength,times),index = t)

    beta <- Variable(index = param)

    beta[j] >= 0
    1-exp(-(times[t]/beta[1])^beta[2]) >= 0
    1-exp(-(times[t+1,t<length(times)]/beta[1])^beta[2]) >=
        1-exp(-(times[t,t<length(times)]/beta[1])^beta[2])

    r <- Expression(index = t)
    r[t] ~ sales[t] - (1-exp(-(times[t]/beta[1])^beta[2]))

    obj = Objective(type="minimize")
    obj ~ Sum(r[t]*r[t],t)

    beta[1] ~ 0.1
    beta[2] ~ 1
}

min.res2 = function(sales, times, mark.size, first.beta){
    sales = sales/mark.size
    Time <- Set()
    param <- Set(1:2)
    timelength <- length(times)
```

```
    t <- Element(set = Time)
    j <- Element(set = param)

    sales <- Parameter(list(1:timelength,sales),index = t)
    times <- Parameter(list(1:timelength,times),index = t)

    beta <- Variable(index = param)
    beta[j] >= 0
    1-exp(-(times[t]/beta[1])^beta[2]) >= 0
    1-exp(-(times[t+1,t<length(times)]/beta[1])^beta[2]) >=
         1-exp(-(times[t,t<length(times)]/beta[1])^beta[2])

    r <- Expression(index = t)
    r[t] ~ sales[t] - (1-exp(-(times[t]/beta[1])^beta[2]))

    obj = Objective(type="minimize")
    obj ~ Sum(r[t]*r[t],t)

    beta[1] ~ first.beta[1]
    beta[2] ~ first.beta[2]
}
```

・S+NUOPT による実データへのあてはめとパラメータの算出

```
nuopt.options(maxitn = 150,method="trust",scaling="on")
result.problem = System(model = min.res,model.data$sales,model.data$times,length(sales))
result.solution = solve(result.problem)
result.lst = list(dataname = dataname,beta = as.vector(result.solution$variables$beta$current),
    variance=as.vector(result.solution$objective)/length(sales),cumsale=length(sales),
    objective=as.vector(result.solution$objective))

result.lst = list(dataname = dataname,beta = as.vector(result.solution$variables$beta$current),
    variance=as.vector(result.solution$objective)/length(sales),cumsale=length(sales),
    objective=as.vector(result.solution$objective))
times = seq(from=1,to=max(sales))
pred = length(sales)*(1-exp(-(times/result.lst$beta[1])^result.lst$beta[2]))
```

result.lst

・実際購買と推定結果のグラフ（図 5・図 6）

```
plot(model.data$times[-1],diff(model.data$sales),type="l",
      xlab="time",ylab="Cumulative unit sales",ylim=c(0,max(diff(model.data$sales))))
lines(times[-1],diff(pred),type="l",col=6)
title(main = dataname)

plot(model.data$times,model.data$sales,type="l",xlab="time",ylab="Cumulative unit sales",ylim=
      c(0,max(model.data$sales)))
lines(times,pred,type="l",col=6)
title(main = dataname)
```

・算出される結果

```
NUMBER_OF_VARIABLES                              2
NUMBER_OF_FUNCTIONS                            211
PROBLEM_TYPE                          MINIMIZATION
METHOD                                TRUST_REGION
<preprocess begin>..........<preprocess end>
<iteration begin>
   res=2.0e+000 .... 1.0e+000 .... 4.9e-002 .... 1.2e+000 .... 1.1e+000 ....
         9.4e-001 .... 1.9e-001 ... 2.5e-011
<iteration end>
STATUS                                     OPTIMAL
VALUE_OF_OBJECTIVE                    0.2432036668
ITERATION_COUNT                                 34
FUNC_EVAL_COUNT                                 50
FACTORIZATION_COUNT                             66
RESIDUAL                          2.547597494e-011
ELAPSED_TIME(sec.)                            0.10
```

```
> result.lst
$dataname:
[1] "koka.hajime"
```

```
$beta:
[1] 98.062341    1.183934

$variance:
[1] 0.0003948111

$cumsale:
[1] 616

$objective:
[1] 0.2432037
```

※ここで，VALUE_OF_OBJECTIVE の値が誤差二乗和（最小二乗誤差）の値である．

- 指数分布のモデルの実データへの当てはめ

・パラメータの推定

```
lm.fit = lm(log(sales+1e-5)~times,data=tmp.data)

predddd = exp(predict(lm.fit))
```

・実際購買と推定結果のグラフ（図 7）

```
plot(model.data$times[-1],diff(model.data$sales),type="l",xlab="time",
    ylab="Cumulative unit sales",ylim=c(0,max(diff(model.data$sales))))

lines(predddd,col=6)
```