付録 A. データフレームから確率を推定する関数 est

A.1. 確率の推定方法

関数 est の目的は、今月入居状態であった部屋が来月も入居状態である確率 p と、今月は空室状態であった部屋が来月も空室状態である確率 q を推定することである。関数 est による p, q の推定方法の概略は以下のとおりである。

1. 物件ごとにデータフレームを分割する
2. 次の変数を定める
   - 今月入居状態であった部屋が先月も入居状態である回数を保存する変数 *p.count*
   - 今月は空室状態であった部屋が先月も空室状態である回数を保存する変数 *q.count*
   - 先月は入居状態であった部屋が、今月は空室である回数を保存する変数 *mp.count*
   - 先月が空室状態であった部屋が、今月は入居状態である回数を保存する変数 *mq.count*
3. 各変数の条件に該当するデータの数を数えて、各変数に代入する（部屋ごとの推定は行なっていない）
4. 次式で確率を推定する

$$p = \frac{p.count}{p.count + mp.count} \quad q = \frac{q.count}{q.count + mq.count}$$

A.2. 関数 est のコード

```
est <- function(dframe) #引数 dframe = データフレーム型の元データ{
   cat("--- Start Estimation of Probability --- \n")

   #unique 関数で、dframe の 4 列目 (物件コード) から
   #重複のない物件コードベクトルを取り出す
   uni.code <- unique(dframe[, 4])

   for(i in uni.code) {
      #物件コードごとのデータフレームを取り出す
      tmp.frame <- dframe[dframe[, 4] == as.integer(i),  ]

      #変数の初期化
      #p.count     先月が入居状態で、今月も入居状態である回数を保存する変数
      #mp.count    先月が入居状態で、今月は空室である回数を保存する変数
      #q.count     先月が空室状態で、今月も空室状態である回数を保存する変数
      #mq.count    先月が空室状態で、今月は入居状態である回数を保存する変数
      #SumOfP      p.count + mp.count の値を保存
      #SumOfQ      q.count + mq.count の値を保存
      p.count <- q.count <- mp.count <- mq.count <- SumOfP <- SumOfQ <- 0
```

```
#check は、先月の部屋がどの行に存在するかを探すための変数
check <- tmp.frame[, 14] - 1000
check[(check %% 100000) < 1000] <- check[(check %% 100000) < 1000] - 100000 + 12000

for(j in seq(along = check)) {
   #先月のデータ位置を示すフラグ
   fragment <- tmp.frame[, 14] == check[j]


   if(length(which(fragment)) > 0) {
      #fragment で、先月のデータに当たる行が存在するかどうかを確認

      #今月が空室でないときの処理
      if(tmp.frame[j, 6] != "nobody") {
         if(tmp.frame[fragment, 6] != "nobody") {
            p.count <- p.count + 1
         }
         else {
            mp.count <- mp.count + 1
         }
      }
      #今月が空室だったときの処理
      else if(tmp.frame[j, 6] == "nobody") {
         if(tmp.frame[fragment, 6] == "nobody") {
            q.count <- q.count + 1
         }
         else {
            mq.count <- mq.count + 1
         }
      }
   }
}
#SumOfP と SumOfQ の算出
SumOfP <- p.count + mp.count
SumOfQ <- q.count + mq.count
if(SumOfP == 0) {
   p.count <- NA
}
else {
   p.count <- p.count/SumOfP
}
#p.count,q.count に確率 p,q を上書き
```

```
      if(SumOfQ == 0) {
         q.count <- NA
      }
      else {
         q.count <- q.count/SumOfQ
      }
      #値の表示
      cat(i, ",", p.count, ",", q.count, "\n", sep = "")
   }
   cat(" ---                End                    ---\n")
}
```

付録 B. 契約価値算定シミュレーション関数 sim.option

B.1. シミュレーションの概略

NPV の推定にはモンテカルロシミュレーションを使っている。入居・空室状態を表すベクトル *room.inuse* を考える。入居状態の部屋に対し、一様乱数を生成し、p 値を超えないものに引き続き入居状態の属性を代入する。また、空室状態の部屋に対して、同様に q 値を超えないものは空室状態の属性を代入するので、q 値を超えるものに入居状態の属性を代入している。

cash は家賃が保存されているベクトルである。*room.inuse* について、入居状態に 1 を、空室状態に 0 を対応させるならば、それぞれのベクトルの要素を掛け合わせて合計をとることで、その期の収益を得ることができる。第 j 期に対して、キャッシュフロー流列を表現するベクトルの第 j 番目に合計を割引率で割り引いたものを代入し、それをシミュレーション期間の長さ分実行することにより、割引キャッシュフロー列を生成できる。割引キャッシュフロー列の合計を取れば、それが NPV である。

B.2. 関数 sim.option のコード

```
#cash          家賃が代入されているベクトル 例. rep(c(4.8,4.9,5.0,5.1),rep(4,5))
#discount      NPV を算出するために使用する割引率
#p             est 関数で推定された確率 p
#q             est 関数で推定された確率 q
#n             シミュレーション年数
#create        シミュレーション回数
#assurance     保証割合 (本文中では r)

sim.option <- function(cash=...,discount=0.1,p=0.9,q=0.1,n=1,create=5,assurance=0.8)
{

#cash のベクトルの長さから、部屋数を算出
room.len <- length(cash)

#入居率 100%の状態での収益を算出
max.cash <- sum(cash)

#保証水準収益の算出
a.cash <- max.cash * assurance

#入居・退出状態の初期値を一様乱数で設定
room.inuse <- runif(room.len, min = 0, max = 1) > q

#cashflow    各期に発生する収益を保存するベクトル
cashflow <- vector(length = n * 12)
```

```
#dis.cashflow     各期に発生した収益を割り引いたものを保存するベクトル
dis.cashflow <- vector(length = n * 12)

#opt.cashflow     保証契約を考慮した収益を保存するベクトル
opt.cashflow <- vector(length = n * 12)

#dis.opt.cashflow  保証契約を考慮した収益を割り引いたものを保存するベクトル
dis.opt.cashflow <- vector(length = n * 12)

#NPV     シミュレーション回数分の NPV を保存するベクトル
NPV <- vector(length = create)

#opt.NPV     シミュレーション回数分の、保証契約を考慮した NPV を保存するベクトル
opt.NPV <- vector(length = create)

#SamplePasss     各期に発生した収益の履歴を保存するベクトル（後で matrix 型に変換)
SamplePass <- rep(0, n * 12)

#opt.SamplePasss     各期に発生した保証契約を考慮した収益の履歴を保存するベクトル（後で
matrix 型に変換)
opt.SamplePass <- rep(0, n * 12)

#シミュレーション回数分、実行する
for(i in seq(to = create)) {

        #n 年間のシミュレーションを開始する (n*12 回)
        for(j in seq(to = n * 12)) {

              #入退室の判定
              room.inuse[room.inuse==T] <- runif(room.len,min=0,max=1)[room.inuse==T] < p
              room.inuse[room.inuse==F] <- runif(room.len,min=0,max=1)[room.inuse==F] > q

              #入退室の判定からその期の収益を算出
              CF <- cashflow[j] <- sum(cash[room.inuse])

              #保証契約を考慮したときの収益を算出
              if(CF > as.integer(a.cash)) {
                      opt.cashflow[j] <- cashflow[j]
              }
              else {
                      opt.cashflow[j] <- a.cash
              }
```

```
                #収益の割引
                dis.cashflow[j] <- cashflow[j]/(1 + (discount/12))^j
                dis.opt.cashflow[j] <- opt.cashflow[j]/(1 + (discount/12))^j
        }

        #NPV と保証契約を考慮した NPV の算出
        NPV[i] <- sum(dis.cashflow)
        opt.NPV[i] <- sum(dis.opt.cashflow)

        #収益の履歴を保存 (データ発生数が 100 以上の場合、100 個まで)
        if(i == 1) {
                SamplePass <- cashflow
                opt.SamplePass <- opt.cashflow
        }
        else if(i <= 100) {
                SamplePass <- cbind(SamplePass, cashflow)
                opt.SamplePass <- cbind(opt.SamplePass, opt.cashflow)
        }
}
#最終的な結果をリスト形式で保存
lst <- list(NPV = NPV, Option.NPV = opt.NPV, SamplePass = as.data.frame(SamplePass),
opt.SamplePass = as.data.frame(opt.SamplePass))
lst
}
```

付録 C. その他

C.1. sim.option の実行

```
> lst <- sim.option(rep(c(4.7,4.8,4.9,5.0,5.1),rep(4,5)),
+ discount = 0.1,p = 0.833,q=0.750,n=10,create = 1000,assurance = 0.8)
> names(lst)
[1] "NPV"            "Option.NPV"     "SamplePass"     "opt.SamplePass"
> frm <- data.frame(未契約 = lst$NPV, 契約 = lst$Option.NPV)
```

C.2. ヒストグラム描画

```
> x <- c(frm$未契約,frm$契約)
> y <- rep(c("未契約","契約"),rep(1000,2))
> histogram(~x|y,layout=c(1,2),xlab="NPV(万)",ylab="割合")
```

C.3. 契約価値算出

```
> z <- frm$契約 - frm$未契約
> frm <- data.frame(frm, 契約価値 = z)
> histogram(frm$契約価値,xlab="契約価値",ylab="割合")
```

C.4. サンプルパスの作図

```
> tsplot(lst$SamplePass[,1],xlab="期間",ylab="CF(万)")
> for(i in 2:10){lines(lst$SamplePass[,i],col = i %% 12)}
> tsplot(lst$opt.SamplePass[,1],xlab="期間",ylab="CF(万)")
> for(i in 2:10){lines(lst$opt.SamplePass[,i],col = i %% 12)}
```