マイクロアレイデータによる予後予測モデル構築における .632+推定量によるエラー率の補正とROC曲線解析

伊 藤　陽 一[1,2]、西 本 尚 樹[2]、江 口 菜 弥 帆[2]
1. 北海道大学大学院医学研究科 臨床統計学分野
2. 北海道大学探索医療教育研究センター

The .632+ estimator for the error rate of the prognostic model with microarray data and the ROC curve analysis

Yoichi M. Ito[1,2], Naoki Nishimoto[2], Namiho Eguchi[2]
1. Department of Biostatistics, Hokkaido University Graduate School of Medicine
2. Center for Translational Research, Hokkaido University

要旨

マイクロアレイデータによる予後予測モデル構築では、通常単回帰によって、候補遺伝子をスクリーニングし、その候補遺伝子を用いて重回帰モデルによるモデル構築を行うのが一般的である。しかし、Simon ら (2003)で示されているように、この手順は、エラー率をかなり過少評価することが知られており、Simon らは、スクリーニングの段階も含めたクロスバリデーションを推奨している。

当該データのみを用いて行うクロスバリデーションの一手法として、Efron ら(1997)の.632+推定量が挙げられる。この方法は、当該データを用いて推定した過少評価を伴うエラー率と、ブートストラップサンプルを用いて推定されたエラー率の重み付き平均を求めることによって、エラー率の過少評価を補正するものである。

本報告では、.632+推定量を、スクリーニングの段階を含む予後予測モデル構築に適用し、構築した予後予測モデルにおける補正済み ROC 曲線を描出するプログラムを紹介する。

キーワード：マイクロアレイデータ解析、予後予測モデル、.632+推定量、ROC 曲線解析

はじめに

近年、細胞における遺伝子発現量を大量に測定できるマイクロアレイデータによって、対象疾患患者の予後を予測するモデルを構築することが行われているが、説明変数となる遺伝子数が非常に多いため、モデルのオーバフィットが問題となることがある。そこで通常は、結果変数とよく相関する遺伝子候補に絞り込むことが行われ、このプロセスはFeature selection と呼ばれている。Feature selection を伴って構築されたモデルの予測精度の評価を行う場合、どの段階までをクロスバリデーションの対象とするのかが問題となる。Simon らは、Feature selection を伴って構築されたモデルの予測精度評価におけるクロスバリデーショ

ンのバイアスを、シミュレーション研究によって示している[1)]。まず、遺伝子数6000、患者数20として、ランダムにデータを発生させる。20人の患者のうち10人を第1群、残りの10人を第2群として、発生させた遺伝子発現プロファイルによって分類することを考える。ランダムに発生させたデータであるので、期待される予測精度は0.5である。Simonらは、このシミュレーションデータを用いて、クロスバリデーションを行わない場合(in-sample error: $\overline{err}$)、Feature selectionを行った後にLeave-One-Out Cross Validation (LOOCV;一人の対象者を除いてモデルを構築し、そのモデルで除いた対象者について予測を行い、予測精度を評価する方法、この操作を全対象者について繰り返して平均を取る)を行う場合、Feature selectionのプロセスもLOOCVに含める場合の3通りに関して、誤分類する人数が何人となるか検討を行った。その結果、クロスバリデーションを行わなかった場合には、98.2%が誤分類する人数が0人となり、Feature selectionを行った後にLOOCVを行った場合でも、90.2%が誤分類する人数が0人となった。一方で、Feature selectionの段階もLOOCVに含めた場合は、誤分類する人数の中央値は11人となり、期待される誤分類人数（10人)に近くなった。したがって、クロスバリデーションを行う際には、Feature selectionのプロセスを含めることが必須と考えられる。

LOOCVは予測精度の推定に関してバイアスがないことが知られているが、対象者数が少ない場合、その推定値のバラツキが大きいことが問題となる場合がある。例えば、対象者数が20人しかいなかった場合にLOOCVで評価すると、一人の結果が変わっただけで正確性の推定値が5%も変化してしまうのである。そこでEfronらは、Bootstrapサンプリングを用いた.632 推定量および.632+推定量を提案している[2,3)]。

本報告では、Feature selection を伴うモデル構築における.632+推定量を用いたエラー率の補正と、ROC曲線解析を行う。

.632+推定量

.632+推定量の原型となった.632推定量では、Bootstrap法を用いた補正を行っている。Bootstrap法とは、ある確率変数が従う分布が、現在得られているデータの分布(経験分布)で近似できるという仮定のもと、現在得られているデータの復元抽出に基づいて推論を行う方法であり、抽出されたデータはBootstrapサンプルと呼ばれる。.632推定量では、まず元のサンプルサイズと同じ回数の復元抽出を行い、1回目のBootstrapサンプルとし、このデータを用いて予測モデルを構築する。構築されたモデルを用いて、Bootstrapサンプルとして抽出されなかった対象者の予測を行い、予測精度(エラー率)を評価する。この操作を繰り返し、平均を取ることによって、Bootstrapに基づくエラー率の推定値とする。この推定値はleave-one-out bootstrap estimate（$\overline{Err}^{(1)}$)と呼ばれている。Bootstrapサンプルの作り方から分かるように、このサンプルでは一人以上の対象者のデータが重複している。したがって、このデータに基づくと、重複した対象者のデータを重視したモデルが構築されることになる。このため、LOOCVと比較して、エラー率は高めに推定されることになる。そこで、.632推定量($\overline{Err}^{(.632)}$)では下式のように、低めに推定されるin-sample errorとの重み付き平均を取ることで、バイアスの減少を図っている。

$$\overline{Err}^{(.632)} = .368 \cdot \overline{err} + .632 \cdot \overline{Err}^{(1)}$$

直感的には.632という重みはleave-one-out bootstrap estimateの実質的なサンプルサイズに比例したものとなっていると解釈できる。復元抽出を行った場合、ある対象者のデータが少なくとも1回以上抽出される確率は、元のサンプルサイズnが大きいとき以下のようになるからである。

$$1-\left(1-\frac{1}{n}\right)^{n} \cong 1-e^{-1} = 1-.368 = .632$$

Breimen らは.632推定量は、極めてOverfitが強いときに、エラー率を過少評価してしまうことを示した[4]。これを受けて、Efron はこの過少評価を補正する.632+推定量を提案した[3]。まず、無情報エラー率γを定義する。ある疾患の再発の有無を予測することを想定し、サンプルにおける観測再発割合を$\hat{p}$、予測モデルによる予測再発割合を$\hat{q}$とする。すると、無情報エラー率の推定値$\hat{\gamma}$は下式で定義される。

$$\hat{\gamma} = \hat{p}(1-\hat{q}) + \hat{q}(1-\hat{p})$$

ここで、相対オーバーフィッティング率$\hat{R}$を以下のように定義する。

$$\hat{R} = \frac{\overline{Err}^{(1)} - \overline{err}}{\gamma - \overline{err}}$$

この$\hat{R}$を用いて.632+推定量は以下のように定義される。

$$\overline{Err}^{(.632+)} = (1-\hat{w})\cdot\overline{err} + \hat{w}\cdot\overline{Err}^{(1)}$$

$$\hat{w} = \frac{.632}{1-.368\hat{R}}$$

重み$\hat{w}$は$\hat{R}=0$のとき.632となり、$\hat{R}=1$のとき1となる。つまり、leave-one-out bootstrap estimate（$\overline{Err}^{(1)}$）が無情報エラー率γと等しければ、.632+推定量は無情報エラー率γと等しくなり、$\overline{Err}^{(1)}$がin-sample error（$\overline{err}$）に近づくにしたがって、.632+推定量も$\overline{err}$に近くなるように構成されている。実際には、$\overline{err}$と$\overline{Err}^{(1)}$と$\hat{\gamma}$の関係によっては、$\hat{R}$が0から1の範囲を外れてしまう問題があるため、$\hat{R}$は以下のように補正されている。

$$\overline{Err}^{(1)'} = \min\left(\overline{Err}^{(1)}, \hat{\gamma}\right)$$

$$\hat{R}' = \begin{cases} \left(\overline{Err}^{(1)'} - \overline{err}\right) \Big/ \left(\hat{\gamma} - \overline{err}\right) & if\ \overline{Err}^{(1)}, \hat{\gamma} > \overline{err} \\ 0 & otherwise \end{cases}$$

.632+推定量を用いたROC曲線の描出方法

再発予測モデルのROC曲線解析を行うためには、構築されたモデルによって再発と判断する閾値ごとの感度と特異度を推定する必要がある。.632+推定量では、予測モデルのエラー率のみが推定されるので、感度と特異度を推定するために、観測された再発の有無で層別し、層ごとにエラー率の推定を行う。また、モデルによって再発と判断する閾値の指標としては、ロジスティック重回帰によってモデル構築を行っているため、特定の遺伝子の発現量ではなく、モデルによって予測された再発確率を用いる。In-sample errorを推定する場合の予測再発確率のセット(閾値の集合)と、leave-one-out bootstrap estimateを推定する場合の予測

再発確率のセットは異なるため、In-sample error を推定する場合の予測再発確率のセットに全ての Bootstrap サンプルにおける予測再発確率のセットを加えたデータを用いて、再発の有無を判断し、感度と特異度の推定を行う。

解析対象データと ROC 曲線解析の結果

解析対象データは、子宮体癌 60 例における癌細胞の遺伝子発現で、遺伝子数は 18401 、結果変数は再発で再発割合は 22/60=36.67%である。Feature selection として、各遺伝子の発現量を用いて、ロジスティック単回帰を行い、$p<0.001$ となった遺伝子を候補遺伝子とした。選択された候補遺伝子群を用いて、ステップワイズ変数選択法を伴うロジスティック重回帰によりモデルを構築した。変数選択の基準は、組み入れ、除外ともに $p<0.05$ である。

Bootstrap サンプルに対しても同様の手順でモデル構築を行った。Feature selection の前で、Bootstrap sampling が行われているので、各 Bootstrap サンプルごとに、選択された候補遺伝子群や最終的に構築される予後予測モデルが異なる点に注意が必要である。

最終的に、どの遺伝子を用いたモデルを採用すべきかについては、in-sample error を推定する際に用いたモデルの遺伝子を採用する方法[1)]と、Bootstrap サンプルにおいて、より多く選択された遺伝子を採用する方法が考えられる[5)]。

図 1 に in-sample error の場合の ROC 曲線とその AUC を、図 2 には leave-one-out bootstrap estimate の場合の ROC 曲線とその AUC を示す。

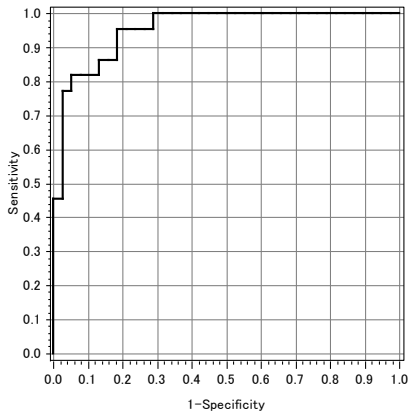

図 1. in-sample error (AUC= 0.953)

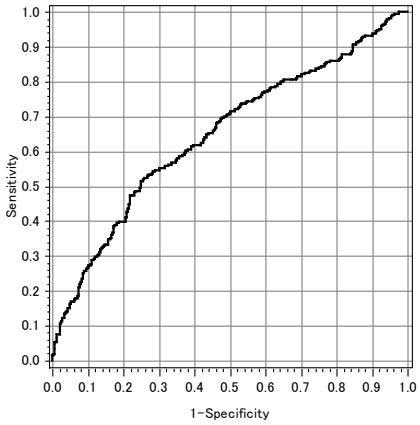

図 2. leave-one-out bootstrap estimate (AUC=0.648)

Leave-one-out bootstrap estimate の場合の ROC 曲線の予測精度が、著しく低下していることが分かる。

次に、図 3 に.632 推定量で補正した場合の ROC 曲線とその AUC を、図 4 に.632+推定量で補正した場合の ROC 曲線とその AUC を示す。

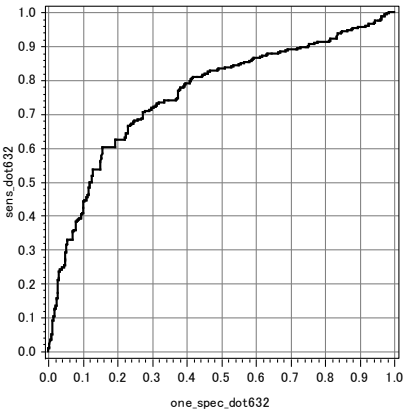

図 3. The .632 estimator (AUC= 0.758)

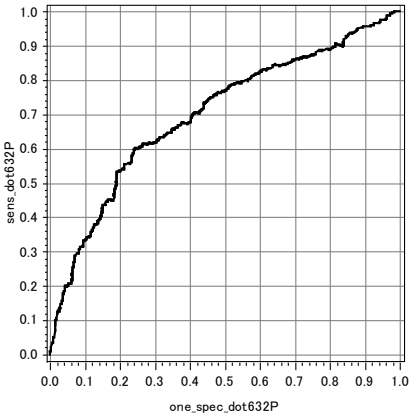

図 4. The .632+ estimator (AUC= 0.700)

.632 推定量で補正することによって、in-sample error の場合と leave-one-out bootstrap estimate の場合の中間に ROC 曲線が入ってくることが分かる。また、.632+推定量で補正した場合には、.632 推定量で補正した場合と比較して、若干予測精度が低下している。また、$\hat{R}'$ の推定値が再発を判断する閾値ごとに変化するため、階段状の曲線にはならないという特徴がある。

考察

本報告では、. 632 推定量によるエラー率の補正を行った場合の ROC 曲線解析の実例の紹介と解析プログラムの提示を行った。本稿で提示した解析プログラムは、Feature selection を行う方法や、モデルを構築する方法の変更を行いたい場合には、解析プログラム中の該当のプロシジャを変更すれば良いだけなので、応用上、一定の柔軟性を有しているものと思われる。また、各 Bootstrap サンプルにおいて、選択された遺伝子候補群と推定されたモデルの遺伝子 ID とパラメータ推定値は、外部ライブラリに candidate データセットおよび model データセットとして保存されるため、事後的に検討が可能である。

Bootstrap sampling の回数については、本プログラムでは Efron の推奨する 50 回を採用しているが、感度と特異度を推定するために、データを再発群と非再発群に分けてエラー率の推定を行っているため、十分な推定精度を確保できていない可能性がある。解析時間に余裕がある場合には、増やした方が良いかもしれない。その場合にはプログラム冒頭のマクロ変数 nboot の数値を変更すればよい。

参考文献

[1] Simon R. Radmacher MD. Dobbin K. McShane LM. Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification. Journal of the National Cancer Institute. 95(1):14-8, 2003.
[2] Efron B. Estimating the error rate for a prediction rule: Improvement on cross-validation. Journal of the American Statistical Association. 78(382):316-31, 1983.
[3] Efron B. Tibshirani R. Improvements on cross-validation: The .632+ bootstrap method. Journal of the American Statistical Association. 92(438):548-60, 1997.
[4] Breiman L. Friedman J. Olshen R. Stone C. Classification and Regression Trees, Pacific Grove, CA: Wadsworth, 1984.
[5] Michiels S. Koscielny S. Hill C. Prediction of cancer outcome with microarrays: a multiple random validation strategy. Lancet. 365(9458):488-92, 2005.

```
/*--- マイクロアレイデータによる予後予測モデル構築における.632+推定量によるエラー率の補正とROC曲線解析のためのSASプログラム
@ SASユーザー総会 in 神戸 2011                    29Jul2011
Copyright © 2011 Yoichi M. Ito
Department of Biostatistics, Graduate School of Medicine, Hokkaido University.
---*/

libname ext "D:¥SASUsersGroup";

/*--- BootStrap Sampling ---------------------------------
&nsample. サンプルサイズ
&ngene.   遺伝子数
&nboot.   ブートストラップ回数

data001   遺伝子発現データセット

  cens      再発の有無(有 cens=0, 無 cens=1)
  c1-c18401 正規化後の遺伝子発現量(18401個の遺伝子)
  no        被験者ID

boot1
 -boot&nboot. Bootstrapサンプルデータセット

  cens      再発の有無(有 cens=0, 無 cens=1)
  c1-c18401 正規化後の遺伝子発現量(18401個の遺伝子)
  no        被験者ID

  cens2     Bootstrapサンプルとして選ばれた場合は欠測
            Bootstrapサンプルとして選ばれなかった場合は
              censの値を代入して、censは欠測にする

  no2       BootstrapサンプルにおけるID
-------------------------------------------------------*/
%let nsample=60;
%let ngene=18401;
%let nboot=50;

/*---------------- Bootstrap sampling -----------------*/
%macro boot;
%do i=1 %to &nboot.;
data test;
  do i=1 to &nsample.;
  no=int(1+ranuni(0)*&nsample.);
  output;end;
run;

proc sort data=test;
  by no;
run;

data test2;
  merge test ext.data001;
  by no;
run;

data ext.boot&i.;
  set test2;
  if i=. then do; cens2=cens; cens=.; end; /* cens2の処理 */
  no2=_n_;
run;

%end;
%mend;
%boot;

/*-------------- leave-one-out bootstrap --------------*/
/* 予測モデルの候補遺伝子のデータセットを用意 */
%macro looboot;
data ext.candidate;
  stop;
run;

data ext.model;     /* 予測モデルのデータセットを用意 */
  stop;
run;

/* bootstrapサンプルにおける予測確率のデータセット
  を用意 */
data ext.looboot;
  stop;
run;

%do i=1 %to &nboot;

data data100;
  set ext.boot&i.;
  call symput('n', put(_n_, best.)); /* OBS数の取得 */
run;

proc sort data=data100;
  by no2 cens2 no cens;
run;

/* 解析用データセットに編集 */
proc transpose data=data100 out=data101;
  var c1-c&ngene.;
  by  no2 cens2 no cens;
run;

data data102;          /* 遺伝子IDの付与(変数名geneid) */
  do i=1 to &n.;
  do geneid=1 to &ngene.;
  output;
  end;end;
run;

data data103;                  /* 解析用データセット */
  merge data101 data102;
  rename col1=exp;
run;

proc sort data=data103;
  by geneid;
run;

/* 現在の繰り返し回数の出力 */
%put The number of repeat is &i.;

/* logの出力を抑制(解析の高速化) */
proc printto log=_dummy_;run;

ods listing close;

ods output ParameterEstimates=est001;

/* logistic単回帰によって候補遺伝子をスクリーニング */
proc logistic data=data103;
  model cens=exp /;
  by geneid;
run;

ods listing;

proc printto;run;               /*logの出力抑制の解除*/
```

```
/* p<0.001以下の遺伝子に限定する */
data est002;
  set est001;
  where  Variable="exp" and probchisq<0.001;
  variable = 'COL'||left(_n_);
  boot = &i.;
  call symput('n2', variable); /* 候補遺伝子数のカウント */
run;

data est003;
  set est002;
  keep geneid variable;
run;

data data104;
  merge est003 data103;
  by geneid;
run;

data data105;
  set data104;
  if variable^="";
  drop  _name_;
run;

proc sort data=data105;
  by  no2    cens    cens2  no   geneid;
run;

proc transpose data=data105 out=data106;
  var exp;
  by no2    cens    cens2   no;
run;

/* logの出力を抑制(解析の高速化) */
proc printto log=_dummy_;run;

ods listing close;
ods output ParameterEstimates=est004;

/* logistic重回帰によって予測モデルを構築 */
/* 予測遺伝子の組み入れ・除外基準 p<0.05 */
proc logistic data=data106;
  model  cens(event='0') =  col1 - &n2.
/selection=s sle=0.05 sls=0.05 ;
output out=result001 p=pred;
run;

ods listing;

proc printto;run;              /* logの出力抑制の解除 */

proc sort data=est004;
  by variable;
run;

proc sort data=est003;
  by variable;
run;

data est005;
  merge est003 est004;
  by variable;
  if df^=.;
  boot=&i.;
run;

data result002;
  set result001;
  drop col1- &n2.;
  boot=&i.;
run;


/* 予測モデルの候補遺伝子のデータセットに蓄積 */
data ext.candidate;
  set ext.candidate est002;
run;

/* 予測モデルのデータセットに蓄積 */
data ext.model;
  set ext.model est005;
run;

/* bootstrapサンプルにおける予測確率のデータセットに
    蓄積 */
data ext.looboot;
  set ext.looboot result002;
run;
%end;

%mend;

%looboot;

/*---------- in-sample error ----------*/
proc sort data=ext.data001;
  by no cens;
run;

/* 解析用データセットに編集 */
proc transpose data=ext.data001 out=data002;
  var c1-c&ngene.;
  by  no cens;
run;

data data003;           /* 遺伝子IDの付与(変数名geneid) */
  do i=1 to &nsample.;
  do geneid=1 to &ngene.;
  output;
  end;end;
run;

data data004;                      /* 解析用データセット */
  merge data003 data002;
  rename col1=exp;
run;

proc sort data=data004;
  by geneid;
run;

/* logの出力を抑制(解析の高速化) */
proc printto log=_dummy_;run;

ods listing close;
ods output ParameterEstimates=est101;

proc logistic data=data004;
  model cens=exp;
  by geneid;
run;

ods listing;

proc printto;run;              /* logの出力抑制の解除 */
```

```
/* p<0.001以下の遺伝子に限定する */
data est102;
  set est101;
  where  Variable="exp" and probchisq<0.001;
  variable = 'COL'||left(_n_);
  call symput('n2', variable); /* 候補遺伝子数のカウント */
run;

data est103;
  set est102;
  keep geneid variable;
run;

data data104;
  merge est103 data004;
  by geneid;
run;

data data105;
  set data104;
  if variable^="";
  drop  _name_;
run;

proc sort data=data105;
  by  no cens  geneid;
run;

proc transpose data=data105 out=data106;
  var exp;
  by no cens;
run;

/* logの出力を抑制(解析の高速化) */
proc printto log=_dummy_;run;

ods listing close;
ods output ParameterEstimates=est004;
proc logistic data=data106;
  model  cens(event='0') =  col1 - &n2.
/selection=s sle=0.05 sls=0.05 ;
output out=result001 p=pred;
run;

ods listing;

proc printto;run;              /* logの出力抑制の解除 */

proc sort data=est004;
  by variable;
run;

proc sort data=est103;
  by variable;
run;

data est005;
  merge est103 est004;
  by variable;
  if df^=.;
  boot=.;       /* Bootstrapサンプルではないことを明示 */
run;

data result002;
  set result001;
  drop col1- &n2.;
  boot=.;
run;

data ext.cand_insample;
  set est102;
run;

data ext.model_insample;
  set est005;
run;

data ext.pred_insample;
  set result002;
run;

/*-------- ROC曲線解析のためのデータセット作成 --------*/
/* bootstrap sampleにおいてsampleとして選ばれなかった
sampleの予測再発確率を取得 */
data data401;
  set ext.looboot;
  where cens=.;
run;

/* in-sampleにおける予測再発確率を取得 */
data data402;
  set ext.pred_insample;
  cens2=cens;
run;

/* ROC評価症例数のカウント */
data data403;
  set data401 data402;
  call symput('n3', left(put(_n_,best.)));
run;

proc sort data=data403; /* 予測再発確率でソート */
  by pred;
run;

/* predの各値を閾値とした場合の感度・特異度の計算のための
データセット */
data data404;
  set data403;
  array roc[&n3.] roc1-roc&n3.;
  array correct[&n3.] correct1-correct&n3.;
  do i=1 to &n3.;
  /* predの各値を閾値とした場合の0,1判断 */
  if i>=_n_ then roc[i]=1;
            else roc[i]=0;
/*predの各値を閾値とした場合の0,1判断と再発の観測値との一
致判断 */
  correct[i]=(cens2=roc[i]);
  end;
  no3=_n_;
  drop _name_ _level_;
run;

proc sort data=data404;
  by no3 cens2 boot;
run;

proc transpose data=data404 out=data405;
  var correct1-correct&n3.;
  by no3 cens2 boot;
run;

/*---- in-sample errorにおける感度・特異度の計算 ------*/
proc means data=data405 noprint;
  class cens2 _name_;
  var col1;
  output out=data406 mean=correct;
  where boot=.;
run;
```

```
data data407;
  set data406(where=(_type_=3));
  roc_id = input(compress(_name_,"correct"), best.);
run;


proc sort data=data407;
  by roc_id cens2;
run;

proc print data=data407;run;

proc transpose data=data407(drop=_name_) out=data408;
  var correct;
  by roc_id;
run;

data data409;
  set data408;
  sensitivity=col1;
  one_spec=1-col2;
  label one_spec="1-Specificity"
        sensitivity="Sensitivity";
run;


/*--- in-sample errorにおけるROC曲線の描出 ---*/
filename gout "D:¥SASUsersGroup¥ROC.emf";
goptions reset=all device=emf gsfname=gout;
proc gplot data=data409;
  plot sensitivity * one_spec/vref=0 to 1 by 0.1 href=0 to
    1 by 0.1 vaxis=axis1 haxis=axis2;
  axis1 length=8.8cm order=(0 to 1 by 0.1) label=(h=1 a=90)
        value=(h=1);
  axis2 length=8.8cm order=(0 to 1 by 0.1) label=(h=1)
        value=(h=1);
  symbol i=join v=none c=black w=2;
run;


/*-- leave-one-out bootstrapにおける感度・特異度の計算 --*/
proc means data=data405 noprint;
  class cens2 _name_ boot;
  var col1;
  output out=data501 mean=correct;
  where boot^=.;
run;

proc means data=data501 noprint;
  class cens2 _name_ ;
  var correct;
  output out=data502 mean=correct;
  where _type_=7;
run;

data data503;
  set data502(where=(_type_=3));
  roc_id = input(compress(_name_,"correct"), best.);
run;

proc sort data=data503;
  by roc_id cens2;
run;

proc transpose data=data503(drop=_name_) out=data504;
  var correct;
  by roc_id;
run;

data data505;
  set data504;
  sensitivity=col1;
  one_spec=1-col2;
  label one_spec="1-Specificity"
        sensitivity="Sensitivity";
run;


/*--- leave-one-out bootstrapにおけるROC曲線の描出 ---*/
filename gout "D:¥SASUsersGroup¥ROC_boot.emf";
goptions reset=all device=emf gsfname=gout;
proc gplot data=data505;
  plot sensitivity * one_spec/vref=0 to 1 by 0.1 href=0 to
1 by 0.1 vaxis=axis1 haxis=axis2;
  axis1 length=8.8cm order=(0 to 1 by 0.1) label=(h=1 a=90)
value=(h=1);
  axis2 length=8.8cm order=(0 to 1 by 0.1) label=(h=1)
value=(h=1);
  symbol i=join v=none c=black w=2;
run;


/*---------------- .632+Estimatorの算出 --------------*/
proc means data=data404;
  var cens2;
  output out=data601 mean=p1;  /* 観測再発割合の推定 */
  where boot=.;
run;

proc transpose data=data404 out=data602;
  var roc1-roc&n3.;
  by no3 cens2 boot;
run;

/* predの各値を閾値とした場合の予測再発割合の推定 */
proc means data=data602 noprint;
  class _name_;
  var col1;
  output out=data603 mean=q1;
  where boot=.;
run;

/* predの各値を閾値とした場合の行IDの生成 */
data data604;
  set data603(where=(_type_=1));
  roc_id = input(compress(_name_,"roc"), best.);
run;

proc sort data=data604;
  by roc_id;
run;

data data605;
  /*  観測再発割合を全OBSに適用 */
  if _n_=1 then set data601(keep=p1);
  set data604;
  gamma=p1*(1-q1)+(1-p1)*q1;          /* gammaの推定 */
  drop  _NAME_    _TYPE_    _FREQ_;
run;

/* leave-one-out bootstrapにおける感度・特異度の取得 */
data data606;
  set data505;
  rename sensitivity=sens_boot one_spec=one_spec_boot;
  drop _NAME_     COL1      COL2;
run;
```

```
/* leave-one-out bootstrapにおけるpredの各値を閾値とした場合の正解率の推定 */
proc means data=data501 noprint;
  class _name_;
  var correct;
  output out=data701 mean=correct_boot;
  where _type_=3;
run;

/* predの各値を閾値とした場合の行IDの生成 */
data data702;
  set data701(where=(_type_=1));
  roc_id = input(compress(_name_,"correct"), best.);
  keep roc_id correct_boot;
run;

/* in-sample errorにおけるpredの各値を閾値とした場合の正解率の推定 */
proc means data=data405 noprint;
  class cens2 _name_;
  var col1;
  output out=data703 mean=correct_naive;
  where boot=.;
run;

/* predの各値を閾値とした場合の行IDの生成 */
data data704;
  set data703(where=(_type_=1));
  roc_id = input(compress(_name_,"correct"), best.);
  keep roc_id correct_naive;
run;

proc sort data=data702;
  by roc_id;
run;

proc sort data=data704;
  by roc_id;
run;

/*--- 感度特異度の.632推定量および.632+推定量の推定 ---*/
data data801;
  merge data409(drop=_NAME_ COL1 COL2) data606 data605
data702 data704;

sens_dot632=1-(.368*(1-sensitivity)+.632*(1-sens_boot));
  one_spec_dot632=.368*(one_spec)+.632*(one_spec_boot);
  error_boot=1-correct_boot;
  error_naive=1-correct_naive;
  error_boot2=min(error_boot,gamma);
  if error_boot>error_naive and gamma>error_naive then
R2=(error_boot2 - error_naive)/(gamma - error_naive);
                                                  else
R2=0;
  w=.632/(1-.368*R2);
  sens_dot632P=1-((1-w)*(1-sensitivity)+w*(1-sens_boot));
  one_spec_dot632P=(1-w)*(one_spec)+w*(one_spec_boot);
run;

/*--- .632推定量を用いたROC曲線の描出 ---*/
filename gout "D:¥SASUsersGroup¥dot632.emf";
goptions reset=all device=emf gsfname=gout;
proc gplot data=data801;
  plot sens_dot632 * one_spec_dot632/vref=0 to 1 by 0.1
    href=0 to 1 by 0.1 vaxis=axis1 haxis=axis2;
  axis1 length=8.8cm order=(0 to 1 by 0.1) label=(h=1 a=90)
        value=(h=1);
  axis2 length=8.8cm order=(0 to 1 by 0.1) label=(h=1)
        value=(h=1);
  symbol i=join v=none c=black w=2;
run;

/*--- .632+推定量を用いたROC曲線の描出 ---*/
filename gout "D:¥SASUsersGroup¥dot632P.emf";
goptions reset=all device=emf gsfname=gout;
proc gplot data=data801;
  plot sens_dot632P * one_spec_dot632P/vref=0 to 1 by 0.1
    href=0 to 1 by 0.1 vaxis=axis1 haxis=axis2;
  axis1 length=8.8cm order=(0 to 1 by 0.1) label=(h=1 a=90)
        value=(h=1);
  axis2 length=8.8cm order=(0 to 1 by 0.1) label=(h=1)
        value=(h=1);
  symbol i=join v=none c=black w=2;
run;


/*--- 各ROC曲線推定方法におけるAUCの推定 ---*/
data data901;
  set data801;
  retain pre_sens      pre_one_spec      auc
         pre_sens_boot pre_one_spec_boot auc_boot
         pre_sens_632  pre_one_spec_632  auc_632
         pre_sens_632P pre_one_spec_632P auc_632P;
  if _n_=1 then do; pre_sens=sensitivity;
                    pre_one_spec=one_spec; auc=0;
                    pre_sens_boot=sens_boot;
                    pre_one_spec_boot=one_spec_boot;
                    auc_boot=0;
                    pre_sens_632=sens_dot632;
                    pre_one_spec_632=one_spec_dot632;
                    auc_632=0;
                    pre_sens_632P=sens_dot632P;
                    pre_one_spec_632P=one_spec_dot632P;
                    auc_632P=0;
                end;
           else do; auc = auc +(pre_sens+sensitivity)
                          *(pre_one_spec-one_spec)/2;
                auc_boot =auc_boot+
                (pre_sens_boot+sens_boot)
                *(pre_one_spec_boot-one_spec_boot)/2;
                auc_632 = auc_632 +
                (pre_sens_632+sens_dot632)
                *(pre_one_spec_632-one_spec_dot632)/2;
                auc_632P = auc_632P+
                (pre_sens_632P+sens_dot632P)
                *(pre_one_spec_632P-one_spec_dot632P)/2;
             pre_sens=sensitivity; pre_one_spec=one_spec;
             pre_sens_boot=sens_boot;
             pre_one_spec_boot=one_spec_boot;
             pre_sens_632=sens_dot632;
             pre_one_spec_632=one_spec_dot632;
             pre_sens_632P=sens_dot632P;
             pre_one_spec_632P=one_spec_dot632P;
                end;
run;

proc print data=data901;  /* 最後の行の数値を参照のこと */
  var auc auc_boot auc_632 auc_632P;
run;
```