The Stata Journal (2009)
9, Number 4, pp. 547–570

Bootstrap assessment of the stability of multivariable models

Patrick Royston
Hub for Trials Methodology Research
MRC Clinical Trials Unit and University College London
London, UK
pr@ctu.mrc.ac.uk

Willi Sauerbrei
Institute for Medical Biometry and Medical Informatics
Freiburg University Medical Center
Freiburg, Germany

Abstract. Assessing the instability of a multivariable model is important but is rarely done in practice. Model instability occurs when selected predictors—and for multivariable fractional polynomial modeling, selected functions of continuous predictors—are sensitive to small changes in the data. Bootstrap analysis is a useful technique for investigating variations among selected models in samples drawn at random with replacement. Such samples mimic datasets that are structurally similar to that under study and that could plausibly have arisen instead. The bootstrap inclusion fraction of a candidate variable usefully indicates the importance of the variable. We describe Stata tools for stability analysis in the context of the `mfp` command for multivariable model building. We offer practical guidance and illustrate the application of the tools to a study in prostate cancer.

Keywords: st0177, mfp, fracpoly, mfpboot, mfpboot_bif, pmbeval, pmbevalfn, pmbstabil, continuous covariates, fractional polynomials, multivariable modeling, stability, bootstrap, bagging

1 Introduction

As pointed out in chapter 8 of Royston and Sauerbrei (2008), an assessment of the (in)stability of a multivariable model is important but is rarely done in practice. Model instability occurs when selected predictors—and for multivariable fractional polynomial (MFP) modeling, functions of continuous predictors—are sensitive to a small change in the data. This article provides Stata tools for stability analysis accompanied by practical guidance.

For many years, bootstrap resampling has been suggested as a tool to study the stability of multivariable models (e.g., see Chen and George [1985] and Sauerbrei and Schumacher [1992]). Including a search for suitable functional forms for continuous variables substantially increases the number of potential models and the accompanying instability. The stability of MFP models was investigated by Royston and Sauerbrei

(2003). Typically, many hundreds or even thousands of bootstrap replications are required for a thorough analysis, although much useful information may be gleaned by using a smaller number, such as 100. With a typical dataset, many different models are selected by MFP (using Stata's `mfp` command) across bootstrap samples. The resulting bootstrap inclusion fractions (BIFs) provide information on the relative importance of each predictor. The BIF is defined as the proportion (or percentage) of bootstrap replications in which a given variable or type of function is selected by MFP. Because dependencies (correlations) among variables are commonplace, to give insight into the structure of selected models, BIFs for pairs of variables should also be considered (Sauerbrei and Schumacher 1992). Second- and higher-order dependencies among BIFs may be studied in detail by log-linear modeling (Royston and Sauerbrei 2003), but we do not consider that topic further here.

The BIFs for simplified classes of functions of continuous variables (for example, a variable excluded, or requiring a linear or nonlinear function) and plots of fractional polynomial (FP) functions estimated in bootstrap samples may help one to assess the functional form required for a continuous predictor. If the sample size is "too small", `mfp`'s default linear function is often selected even if the underlying function is nonlinear. One useful technique for summarizing functional form is "bagging" or bootstrap aggregation of functions (Breiman 1996a). Selected FP functions are averaged across bootstrap samples to give a more stable estimate of the underlying function and a more realistic impression of the uncertainty associated with that function. Bagging may be regarded as a type of model uncertainty analysis (e.g., see Buckland, Burnham, and Augustin [1997] and Augustin, Sauerbrei, and Schumacher [2005]). Some postestimation tools for bagging FP functions are included here.

Finally, quantitative measures to assess stability of functions are described and exemplified.

2 Using the bootstrap to explore model stability

It is not our intention to repeat all the details of the methods already published in Royston and Sauerbrei (2003) and Royston and Sauerbrei (2008) (chapter 8 and section 10.6). We recommend reading either Royston and Sauerbrei (2003) or chapters 6 and 8 of Royston and Sauerbrei (2008) beforehand. Here we give a taste of how the bootstrap is used with model selection, but we mainly concentrate on the Stata software and on examples.

2.1 Selecting variables within bootstrap samples

An appropriate method for studying model stability is nonparametric bootstrap sampling. A random sample with replacement is taken from the numbers $1, \ldots, n$, which index the observations. The complete observations (i.e., the response, covariates, and in time-to-event data, the censoring indicator) associated with the selected indices comprise the bootstrap sample.

A collection of B bootstrap samples may be used to explore variations among possible models for the original dataset. Suppose multivariable model building with k covariates is to be done. In the simplest case of the linear model, for which the inclusion or exclusion of variables is the only model selection issue, the algorithm is as follows:

1. Draw a bootstrap sample of size n.
2. Apply the model selection procedure.
3. For each covariate x_j $(j = 1, \ldots, k)$, record whether x_j is selected in the model.
4. Repeat steps 1–3 a large number, B, of times.
5. Summarize the results.

The results comprise a matrix with B rows and k columns. The ith row and jth column element is an indicator variable, I_{ij}, taking the value 1 if x_j was selected in the ith bootstrap sample and 0 if not. The BIF for x_j equals $(1/B)\sum_{i=1}^{B} I_{ij}$. A reasonable range for B is 100 to 1,000 or more.

2.2 Assessing the importance of a variable

It is assumed that each replication, being a random sample from the observations in the study, reflects the underlying structure of the data. Important variables should therefore be included in most of the bootstrap replications. The BIF may be used as a criterion for the importance of a variable. A variable that is approximately uncorrelated with others and that is just significant at level α in the full model is selected in about half of bootstrap samples. If its p-value in the full model is less than α, then the BIF is larger than 0.5, and if the p-value is very small, then the BIF tends toward 1.

A difficulty with the BIF as a criterion of the importance of a variable is how to cope with correlated variables. Often, only one variable of a correlated set is selected in a particular bootstrap replication. Sauerbrei and Schumacher (1992) considered the inclusion frequencies of all possible pairs of variables. In an extreme example, one of two highly correlated variables may always be selected, but the BIF of each variable may only be about 50%. Failure to recognize the correlation between the inclusion frequencies of two variables, together with the aim of building simpler models, may result in inappropriately eliminating both variables from the model. For a detailed discussion of strategies for model building based on BIFs, see Sauerbrei and Schumacher (1992). Furthermore, higher-dimensional dependencies among the inclusion frequencies often occur. Sauerbrei and Schumacher (1992) considered only two-dimensional inclusion frequencies in up to 1,000 bootstrap replications, whereas for a detailed investigation of higher-dimensional relationships, more replications are required; see, for example, Royston and Sauerbrei (2003).

The distribution of inclusion frequencies critically depends on the sample size (or in time-to-event data, the number of events) and the significance level used in the selection procedure. Nevertheless, some understanding of the distribution of inclusion frequencies of variables and their relationships is a useful complement to the model-building process. It can be an important criterion when selecting a model (Sauerbrei 1999; Sauerbrei, Holländer, and Buchholz 2008).

2.3 Assessing model stability for functions

Selecting or not selecting a predictor in a multivariable model is a simpler consideration than selecting a functional form for a continuous covariate. Allowing FP functions of continuous predictors in bootstrap replications greatly increases the number and types of candidate models available. Furthermore, one may consider how to define similarities or differences between the functions and how to summarize them across bootstrap replications. BIFs alone are inadequate for assessing the effect of a continuous covariate. For example, in some replications a variable may be excluded, and in others it may be included as linear, and in still others it may be included as a nonlinear, monotonic, or nonmonotonic FP function.

Methods for summarizing variation between curves and a measure of curve instability were suggested by Royston and Sauerbrei (2003). The methods are briefly described and exemplified later in this article.

3 Using the mfpboot program

The main program described here, `mfpboot`, is designed to carry out stability analysis of variable selection and FP function selection with Stata's `mfp` program. The reader is assumed to have some familiarity with `mfp` and the concepts of model building it embodies. A recent introduction to the topic may be found in Sauerbrei, Royston, and Binder (2007).

The output from `mfpboot` looks slightly different for Stata 11 than for earlier versions of Stata and is illustrated here for Stata 11. The program works with Stata 8 and all later versions.

We defer presenting the formal syntax diagram of `mfpboot` at this stage, preferring to give two motivating examples of using `mfpboot` with a real dataset. The first example concentrates on variable selection and the BIF (no FP modeling is done), whereas the second example is more general in that FP modeling of continuous covariates is included.

3.1 Prostate cancer data

Stamey et al. (1989) studied potential predictors of prostate-specific antigen (PSA) in 97 patients with prostate cancer. The aim was to see which factors or combinations of factors were associated with a raised PSA level. Clinical observations were made around the time of a surgical operation in which the entire prostate gland is removed. Please see table A.6 of Royston and Sauerbrei (2008) for further details.

The response variable, log PSA level, is continuous. There are seven candidate variables for multivariable modeling, six of which are continuous; see table 1.

Table 1. Description of the prostate cancer data; the sample size is 97

Name	Type	Description
`age`	Continuous	Age
`svi`	Binary	Seminal vessel invasion (Y/N)
`pgg45`	Continuous	Percent Gleason score 4 or 5
`cavol`	Continuous	Cancer volume, ml
`weight`	Continuous	Prostate weight, g
`bph`	Continuous	Amount of benign prostatic hyperplasia, g
`cp`	Continuous	Amount of capsular penetration, g
`lpsa`	Continuous	Log PSA concentration [response]

The Spearman correlation matrix among the variables is given in table 2.

Table 2. Spearman correlation coefficients between the variables in the prostate cancer data

Name	`age`	`svi`	`pgg45`	`cavol`	`weight`	`bph`	`cp`	`lpsa`
`age`	1							
`svi`	0.13	1						
`pgg45`	0.27	0.48	1					
`cavol`	0.19	0.56	0.50	1				
`weight`	0.40	0.17	0.17	0.29	1			
`bph`	0.34	−0.09	0.10	0.01	0.49	1		
`cp`	0.15	0.62	0.66	0.66	0.18	0.02	1	
`lpsa`	0.22	0.57	0.52	0.70	0.45	0.17	0.52	1

We recommend always examining the Spearman correlation matrix before embarking on multivariable model building. It provides valuable information on the likelihood of predictive information being shared among variables and could even result in a priori omission of some variables among clusters of highly correlated sets.

Note the positive correlation between the response, `lpsa`, and every covariate, and some quite strong correlations among the covariates themselves. We would therefore expect that a parsimonious multivariable model would not need to include all seven predictors. The strongest predictor (univariately) is `cavol`.

(*Continued on next page*)

3.2 First analysis: All effects assumed linear

Analysis with mfp

We apply `mfp` to select a model at a conventional significance level of 0.05 for each predictor, assuming all effects are linear (signified by the `df(1)` option):

```
. use prostate_ca
(Prostate cancer data)

. mfp, select(0.05) df(1): regress lpsa age svi pgg45 cavol weight bph cp

Deviance for model with all terms untransformed =   214.267, 97 observations
  (output omitted)
Fractional polynomial fitting algorithm converged after 2 cycles.

Transformations of covariates:

-> gen double Ipgg4__1 = pgg45-24.3814433 if e(sample)
-> gen double Icavo__1 = cavol-7.000824803 if e(sample)
-> gen double Iweig__1 = weight-41.30953479 if e(sample)

Final multivariable fractional polynomial model for lpsa
--------------------------------------------------------------------
    Variable |    -----Initial-----          -----Final-----
             |   df     Select   Alpha    Status    df    Powers
-------------+------------------------------------------------------
         age |    1     0.0500   0.0500     out      0
         svi |    1     0.0500   0.0500     in       1     1
       pgg45 |    1     0.0500   0.0500     in       1     1
       cavol |    1     0.0500   0.0500     in       1     1
      weight |    1     0.0500   0.0500     in       1     1
         bph |    1     0.0500   0.0500     out      0
          cp |    1     0.0500   0.0500     out      0
--------------------------------------------------------------------

      Source |       SS       df       MS              Number of obs =      97
-------------+------------------------------           F(  4,    92) =   31.34
       Model |  73.7759955     4  18.4439989           Prob > F      =  0.0000
    Residual |  54.1416631    92  .588496339           R-squared     =  0.5767
-------------+------------------------------           Adj R-squared =  0.5583
       Total |  127.917659    96  1.33247561           Root MSE      =  .76714

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         svi |   .5842647   .2477803     2.36   0.020     .0921517    1.076378
    Ipgg4__1 |    .006692   .0031353     2.13   0.035     .0004651     .012919
    Icavo__1 |   .0634745   .0123785     5.13   0.000     .0388897    .0880592
    Iweig__1 |   .0164586   .0041656     3.95   0.000     .0081854    .0247318
       _cons |   2.351897   .0945757    24.87   0.000     2.164061    2.539732
------------------------------------------------------------------------------
Deviance:  218.713.
```

Four variables are selected: `svi`, `pgg45`, `cavol`, and `weight`. The explained variation for this model is 0.58. Standard backward elimination, using the command

```
. stepwise, pr(0.05): regress lpsa age svi pgg45 cavol weight bph cp
```

finds the same model.

A word of explanation of the columns labeled `Select` and `Alpha` in the above output may be helpful. They correspond to the `mfp` options `select()` and `alpha()`. The first controls the significance level that is applied to selecting variables by backward elimination. Here `select(0.05)` was used, resulting in the selection of variables nominally significant at the 5% level. The second determines the significance level for selecting FP functions of continuous variables, the default being `alpha(0.05)`. We applied the `mfp` option `df(1)`, meaning that unless otherwise specified, all predictors are to be treated as linearly related to the outcome. No FP functions are required, and the `alpha()` option is inactive. Nevertheless, the value(s) of `alpha()` is still displayed in the summary table.

Analysis with mfpboot

We now perform a stability analysis of the same model-selection procedure with the `mfpboot` command with $B = 100$ replicates:

```
. mfpboot, select(0.05) df(1) clear outfile(mfpboot1) replicates(100) seed(101):
> regress lpsa age svi pgg45 cavol weight bph cp
Dry run of mfp ...
mfp, noscaling df(1) select(0.05) center(age:63.86598,pgg45:24.38144,
> cavol:7.000825,weight:41.30953,bph:2.644845,cp:2.362371):
> regress lpsa age svi pgg45 cavol weight bph cp
Bootstrapping ...
0 10 20 30 40 50 60 70 80 90 100
Results from original data and 100 bootstrap replicates saved to mfpboot1.
```

In this example, `mfpboot` has two options to control model selection with `mfp`, namely, `select(0.05)` and `df(1)`, and four options of its own, i.e., `clear`, `outfile(mfpboot1)`, `replicates(100)`, and `seed(101)`. We included `seed(101)` for reproducibility of the results. The display ("dry run") of the `mfp` command that `mfpboot` executes in each bootstrap sample is to assist debugging and to show exactly what `mfp` is doing. Here the `df(1)` option overrides the default setting of 4 degrees of freedom for each predictor implicitly given by `dfdefault(4)`. The values shown in the `center()` option have been calculated by `mfp` and are the means of the continuous predictors. Note that "bootstrap replicate 0" denotes the original data; `mfp` is applied to the original data and the results are stored in the first observation of the output file (`mfpboot1.dta`).

Analyzing the output file

`mfpboot` creates an output file—here `mfpboot1.dta` with 101 records—with one record (the first) for the analysis of the original data, and the rest for the analysis of each bootstrap sample. It contains $2k + 2$ variables, where k is the number of predictors. Included are two generic variables: `i`, which denotes the original data when `i` equals 0 and the bootstrap replication number when `i` is greater than 0, and `b0`, the regression intercept. A summary of `mfpboot1.dta`, excluding the first observation, is as follows:

```
. use mfpboot1, clear
. summarize if i > 0
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           i |       100        50.5    29.01149          1        100
       agep1 |         8           1           0          1          1
       ageb1 |         8   -.0348865    .0081541  -.0458779  -.0235269
       svip1 |        79           1           0          1          1
       svib1 |        79    .7812214    .2016457   .4089902   1.292439
-------------+--------------------------------------------------------
     pgg45p1 |        66           1           0          1          1
     pgg45b1 |        66    .0094244    .0024546    .005666   .0162177
     cavolp1 |       100           1           0          1          1
     cavolb1 |       100    .0772394    .0180375   .0444745    .142585
    weightp1 |        79           1           0          1          1
-------------+--------------------------------------------------------
    weightb1 |        79     .017017    .0047068   .0082254   .0276669
       bphp1 |        36           1           0          1          1
       bphb1 |        36    .0986751     .021038   .0599876   .1398226
        cpp1 |        22           1           0          1          1
        cpb1 |        22   -.0917698    .0456209  -.1538938   .0838325
-------------+--------------------------------------------------------
          b0 |       100    2.335615    .1029856   2.117865   2.554383
```

In this example, in which only inclusion or exclusion of a variable is entertained, each predictor in the MFP model generates two variables, *varname*`p1` and *varname*`b1`. For example, `age` has generated `agep1` and `ageb1`. `agep1` contains the first FP transformation applied to `age`. Because we have specified `df(1)` (i.e., linear) for all variables, `agep1` is always 1 and there is no `agep2`. `ageb1` is the estimated regression coefficient for the FP1 transformation of `age` (here just of `age` minus its mean) in bootstrap replications in which `age` is selected. Similar considerations apply to the other predictors. Because `svi` is binary, it will always be treated as linear in any model.

The summary of the *`p1` or the *`b1` variables immediately reveals the BIFs. For example, `agep1` has eight nonmissing values; therefore, the BIF for `age` is 8/100 or 8%. Clearly, `age` is not an important variable. As expected, the selected variables (`svi`, `pgg45`, `cavol`, `weight`) all have BIFs greater than 50%, and `cavol` has a BIF of 100%. The only unselected variable that might be considered for inclusion in a final model is `bph` with a BIF of 36%, but 36% is not convincing evidence that `bph` is important (see further evidence below).

The BIFs may be economically displayed using the supplied `mfpboot_bif` command:

```
. use mfpboot1
. mfpboot_bif
        age:      8   8.00
        svi:     79  79.00
      pgg45:     66  66.00
      cavol:    100 100.00
     weight:     79  79.00
        bph:     36  36.00
         cp:     22  22.00
```

Tabulation of inclusion indicators for each pair of predictors (using the `tab2` command, results not shown) reveals quite a strong negative association between inclusion of `weight` and `bph`—the Spearman correlation between the inclusion indicators is −0.69:

```
. generate byte weighti = !missing(weightp1) if i > 0
(1 missing value generated)
. generate byte bphi = !missing(bphp1) if i > 0
(1 missing value generated)
. tabulate weighti bphi

           |         bphi
   weighti |         0          1 |     Total
-----------+----------------------+----------
         0 |         0         21 |        21 
         1 |        64         15 |        79 
-----------+----------------------+----------
     Total |        64         36 |       100 
```

In all 21 replications in which `weight` is omitted, `bph` seems to substitute for it. In only 15% of replications are both variables selected. When `weight` is included, `bph` is added to the model in only 15/79 (19%) of replications. The table lends support for including only one of the two variables, and `weight`, being the stronger predictor, is the obvious choice.

3.3 Second analysis: FP modeling of continuous predictors

Analysis with mfp

The `mfp` command is identical to before, except that the `df(1)` option is removed to activate the default of 4 degrees of freedom for each continuous predictor, equivalent to allowing maximum complexity FP2.

The result (details not shown) is that three predictors were selected: `svi`, `cavol`(0), and `weight`(1). The notation `cavol`(0) means that an FP1 transformation with power 0 (i.e., a log transformation) was selected by `mfp` for this variable. A linear function of `weight` was selected. The explained variation for the model is 63%, some 5 percentage points higher than for the four-variable linear model.

Analysis with mfpboot

The `mfpboot` command is identical to before, except that, as with the `mfp` command above, the `df(1)` option is removed to allow FP2 or FP1 functions to be selected if sufficiently supported by the data:

(*Continued on next page*)

```
. use prostate_ca, clear
(Prostate cancer data)
. mfpboot, select(0.05) clear outfile(mfpboot2) replicates(100) seed(101):
> regress lpsa age svi pgg45 cavol weight bph cp
Dry run of mfp ...
mfp, noscaling select(0.05) center(age:6.386598,pgg45:.2538144,cavol:.7000825,
> weight:.4130953,bph:2.644845,cp:.2362371): regress lpsa age svi pgg45 cavol
> weight bph cp
Bootstrapping ...
0 10 20 30 40 50 60 70 80 90 100
Results from original data and 100 bootstrap replicates saved to mfpboot2.
```

The output file (`mfpboot2.dta`) has additional variables to allow for possible FP2 transformations. The predictor `age`, for example, is represented by `agep1` and `ageb1` (as before), and `agep2` and `ageb2`. If, in a given bootstrap replication, an FP1 function was selected, then `agep1` holds the corresponding power (1 if a linear function was selected) and `agep2` has a missing value. If an FP2 function was selected, then `agep1` and `agep2` store the respective FP2 powers. If `age` was dropped, then `agep1`, `agep2`, `ageb1`, and `ageb2` all have missing values. `ageb1` and `ageb2` store the relevant regression coefficients.

Analyzing the output file

The BIFs from the `mfpboot` analysis may be obtained using the `mfpboot_bif` program, as before:

```
. use mfpboot2, clear
. mfpboot_bif
          age:     19  19.00
          svi:     86  86.00
        pgg45:     38  38.00
        cavol:    100 100.00
       weight:     85  85.00
          bph:     31  31.00
           cp:     31  31.00
```

Interestingly, the distinction between included and excluded variables is sharper than before. The selected variables (`svi`, `cavol`, and `weight`) have BIFs of 85% or more. The excluded variables have BIFs of 31% or less.

By using `mfpboot_bif` with the `term(2)` option to pick up the *`p2` variables, we can also obtain the BIFs for FP2 functions:

```
. use mfpboot2
. mfpboot_bif, term(2)
          age:      3   3.00
variable svip2 not found
        pgg45:     10  10.00
        cavol:     35  35.00
       weight:     10  10.00
          bph:     13  13.00
           cp:     17  17.00
```

Because `svi` is binary, it has only a linear term (`svip1`, which is 1 when `svi` is selected and missing otherwise)—no `svip2` variable is found. No continuous predictor has an FP2 BIF of greater than 35%, suggesting no clear need for an FP2 function for any of them.

The strongest predictor is `cavol` (its BIF is 100%). A tabulation of the selected FP1 and FP2 powers for this variable in a multivariable context is as follows:

```
. tabulate cavolp1 cavolp2 if i > 0, missing
           |                        cavolp2
   cavolp1 |        -1         .5          1          2          3          . |     Total
-----------+------------------------------------------------------------------+----------
        -2 |         1          3          3          0          0          0 |         7 
        -1 |         0          0          9          5          3          4 |        21 
       -.5 |         0          0          1          1          7         11 |        20 
         0 |         0          0          0          0          2         37 |        39 
        .5 |         0          0          0          0          0          9 |         9 
         1 |         0          0          0          0          0          4 |         4 
-----------+------------------------------------------------------------------+----------
     Total |         1          3         13          6         12         65 |       100 
```

The most commonly selected model, with a BIF of 37%, is FP1(0), and that is the obvious choice if one FP model is required. However, several other FP1 and FP2 models are selected in different bootstrap replications. We explore the implications of the variability among selected functions for the shape of the function of `cavol` in the next section.

3.4 Examining the functions

Following use of `mfpboot`, it is of interest to investigate variation among the (partial) predicted functions for a particular variable. A program called `pmbeval` that works with the curve summary data saved by `mfpboot` is designed to make this easier.

Consider `cavol`, which is selected in 100% of bootstrap replications. However, the FP1(0) model (log transformation) is chosen in only 37%, so there appears to be some uncertainty about the functional form for `cavol` within the bootstrap samples.

The following code uses `pmbeval` to extract the fitted functions of `cavol` in the original data (variable `v0`) and in all 100 bootstrap samples (variables `v1`–`v100`) and plots the first 50 of them against `cavol`:

```
. use mfpboot2
. pmbeval, clear xvar(cavol) rawdata(prostate_ca)
. line v1-v50 cavol, ytitle("Fitted values of lpsa") legend(off)
```

The resulting plot is shown in figure 1.

(*Continued on next page*)

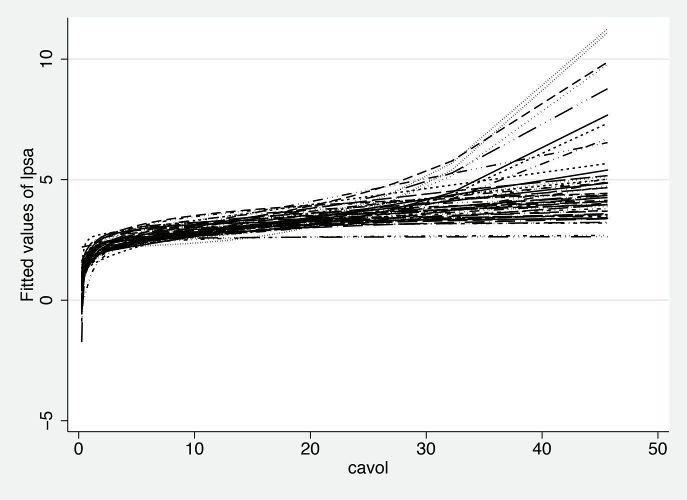

Figure 1. Estimated functions of `cavol` in the first 50 bootstrap replications

Let us represent the model fit in a given bootstrap replication as $\widehat{\beta}_0 + \widehat{f}(\texttt{cavol}) + \widehat{\beta}\mathbf{z}$, where $\widehat{f}(\texttt{cavol})$ denotes the fitted FP function of `cavol` that is selected and $\widehat{\beta}\mathbf{z}$ refers to other covariates or FP functions of them that enter the model. The curves in figure 1 are plots of $\widehat{\beta}_0 + \widehat{f}(\texttt{cavol})$ against `cavol`. It appears that the functional form is rather stable (well defined) for `cavol` less than about 25 ml but much less certain beyond that.

We can examine the function and its uncertainty further by using `pmbeval` to compute the mean and a pointwise 90% data interval, i.e., the estimated 5th and 95th centiles of `cavol`, for the fitted values across bootstrap samples. (For acceptable precision, a 95% interval would require $B > 100$):

```
. use mfpboot2, clear
. pmbeval if i > 0, clear xvar(cavol) rawdata(prostate_ca) centiles(5 95) mean
```

We specify `i > 0` to exclude the results from the original data from the calculations. `pmbeval` places three new variables into the dataset: `_mean` containing the mean fitted values, and `_c1` and `_c2` containing the 5th and 95th pointwise centiles. We may compare these results graphically with those from the selected model. We refit the latter using `fracpoly` and compute the required pointwise confidence interval for the fitted function:

```
. use prostate_ca, clear
(Prostate cancer data)

. fracpoly, center(cavol weight: mean): regress lpsa cavol 0 weight svi
-> gen double Iweig__1 = weight-41.30953479 if e(sample)
-> gen double Icavo__1 = ln(X)+.3565571219 if e(sample)
   (where: X = cavol/10)

      Source |       SS       df       MS              Number of obs =      97
-------------+------------------------------           F(  3,    93) =   53.36
       Model |  80.9134282     3  26.9711427           Prob > F      =  0.0000
    Residual |  47.0042304    93  .505421833           R-squared     =  0.6325
-------------+------------------------------           Adj R-squared =  0.6207
       Total |  127.917659    96  1.33247561           Root MSE      =  .71093

------------------------------------------------------------------------------
        lpsa |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    Icavo__1 |    .540209   .0744859     7.25   0.000     .3922947    .6881233
    Iweig__1 |    .014159   .0038967     3.63   0.000     .0064209     .021897
         svi |   .6794446   .2080709     3.27   0.002     .2662572    1.092632
       _cons |   2.653265   .1064653    24.92   0.000     2.441846    2.864684
------------------------------------------------------------------------------
Deviance:   205.00.

. fracpred f, for(cavol)

. fracpred s, for(cavol) stdp

. gen lb = f - 1.645 * s

. gen ub = f + 1.645 * s
```

For compatibility with what `mfp` does by default, in the above code, it is necessary to add the option `center(cavol weight: mean)` to `fracpoly` to adjust `weight` and `cavol` to their means, but not to adjust the binary variable `svi`.

Figure 2 compares the bootstrap mean (bagged) function for `cavol` with the curve from the original MFP analysis.

(*Continued on next page*)

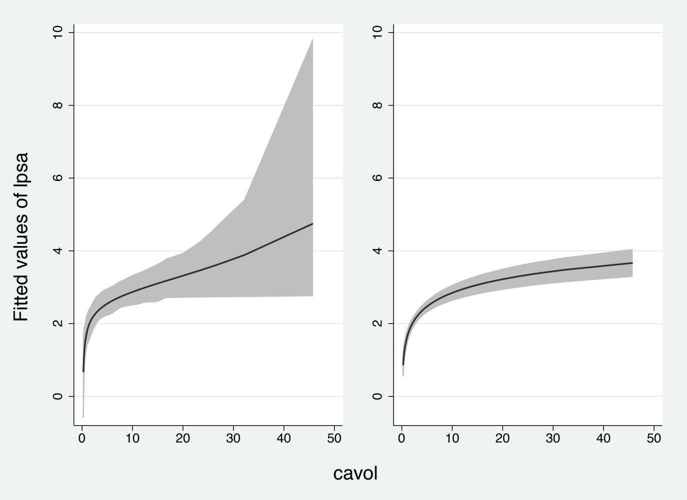

Figure 2. Left panel: Bagged function of `cavol` with 90% data interval. Right panel: Estimated FP1 function of `cavol` in the original data, with 90% pointwise confidence interval (conditional on selected model).

The shape of the functions is similar for lower values of `cavol` but the bagged function increases more for higher values. A striking difference is the amount of additional uncertainty revealed by the bootstrap approach.

4 Syntax

`mfpboot` is a prefix command. Some of its options are native to itself and the others are for `mfp`. The general syntax is

`mfpboot,` *mfpboot_options* [*mfp_options*]`:` *regression_cmd* [*yvar1* [*yvar2*]]
xvarlist [*if*] [*in*] [*weight*] [`,` *regression_cmd_options*]

See section 6.1 for available *mfpboot_options*.

See [R] **mfp** for available *mfp_options*.

regression_cmd may be `clogit`, `cnreg`, `glm`, `intreg`, `logistic`, `logit`, `mlogit`, `nbreg`, `ologit`, `oprobit`, `poisson`, `probit`, `qreg`, `regress`, `rreg`, `stcox`, `stcrreg`, `streg`, or `xtgee`; see the corresponding manual entry in the *Base Reference Manual*.

All weight types supported by *regression_cmd* are allowed; see [U] **11.1.6 weight**.

`mfpboot_bif` [, `term(`#`)` `generate`]

`pmbeval` is a postestimation command with the following syntaxes:

First syntax

`pmbeval` [*if*] [*in*], `clear` `rawdata(`*filename*`)` `xvar(`*xvarname*`)`
[`centiles(`*numlist*`)` `mean` `sd` `n` `standardize`]

Second syntax

`pmbeval` [*if*] [*in*], `saving(`*newfilename*`)` `rawdata(`*filename*`)` `xvar(`*xvarname*`)`
[`standardize`]

`pmbevalfn` [*if*] [*in*], `clear` `rawdata(`*filename*`)` [`standardize`]

`pmbstabil` [*if*] [*in*] [, `by(`*varname*`)` `conditional` `trunc(`#`)`]

5 Description

The programs compute fitted FP functions and stability measures following the use of `mfpboot`. A "replication" denotes a bootstrap sample.

`mfpboot_bif` computes the BIF for each predictor in a dataset created by `mfpboot`. The BIF for a given variable is the proportion of bootstrap replications in which the variable entered the model selected by `mfp`.

`pmbeval` (first syntax) evaluates the FP function of the covariate *xvarname* for each available replication and saves the results to the workspace, replacing the current data. The values of *xvarname* are read from *filename* and are also saved to the workspace. The resulting data are suitable for plotting the fitted functions.

`pmbeval` (second syntax) evaluates the FP function of the covariate *xvarname* for each available replication and saves the results to *newfilename*. The values of *xvarname* are read from *filename*. The data saved in *newfilename* are in a format suitable for stability analysis using `pmbstabil` but are not intended for plotting.

`pmbevalfn` evaluates the linear predictor for the model from each replication and saves the results to the workspace, replacing the current data. The current data are the contents of the file created by `mfpboot`, which must be loaded first. The values of the covariates are supplied in *filename*. Usually, *filename* is the file holding the data on which `mfpboot` was originally run, although another file may be used instead if it contains appropriate data.

`pmbstabil` computes instability measures V, D^2, T, and R^2 for a continuous covariate for which the fitted values were created by `pmbeval` (second syntax) and which have been loaded into the workspace.

6 Options

6.1 Options for mfpboot

`clear` is required and signifies willingness for the data in the workspace to be replaced.

`outfile(`*outfilename*`)` is required and specifies the name of the new file to be created holding the summaries of the MFP model from each bootstrap replicate. If *outfilename*`.dta` exists, an error is raised unless the `replace` option is used to allow the file to be replaced.

`keepalso(`*varlist*`)` causes the variables in *varlist* to be stored in the file specified by `saving()`, along with the standard variables stored there. See also the `saving()` option.

`nodryrun` prevents `mfpboot` from doing a test run of the `mfp` command. By default, a test run is done before the bootstrap procedure starts. The aim is to detect syntax errors or other issues in the underlying `mfp` command. It is strongly recommended that you do not skip this check. Once the `mfp` command is working on the original data, the production run of `mfpboot` can be done with the `nodryrun` option.

`replace` allows *outfilename*`.dta` to be overwritten if it already exists.

`replicates(`*#*`)` sets the number of bootstrap replicates to *#*. If $\# = 0$, then only results from the model for the original data are saved. The default is `replicates(100)`.

`saving(`*datafilename*`)` saves the bootstrap samples to a new file called *datafilename*. By default, only *yvar1* [*yvar2*] and *xvarlist* (and if Cox regression is used, *deadvar*) are saved, but see the `keepalso()` option for how to extend the list of saved variables.

`seed(`*#*`)` sets the random-number seed to *#*. This option is intended to ensure reproducibility of the bootstrap samples.

6.2 Options for mfpboot_bif

`term(`*#*`)` refers to the FP term. The default is `term(1)`, meaning to compute the BIF for the first term of the FP function of a predictor (the only term, if the variable was modeled as FP1 or linear). Specifying `term(2)` would compute the BIF for the FP2 term and would indicate the fraction of bootstrap replicates in which MFP selected an FP2 function for each variable.

`generate` generates new variables indicating whether each FP term in the dataset has been selected (new variable taking on the value 1) or not (taking on the value 0). The new variables are named by appending `i`# to the name of the original variable, where # is 1 (for the FP1 or linear term) or 2 (for the FP2 term).

6.3 Options for pmbeval (both syntaxes)

`rawdata(`*filename*`)` is required and specifies the name of the file that holds the desired values of *xvarname*. This file is typically the original data file used with `mfpboot`, but it need not be.

`xvar(`*xvarname*`)` is required and specifies the name of the covariate whose function is to be evaluated.

`standardize` standardizes the fitted values for each curve to have mean zero.

6.4 Options for pmbeval (first syntax only)

`clear` is required and signifies willingness for the data in the workspace to be replaced.

`centiles(`*numlist*`)` calculates and saves centiles of the fitted curves across replications at the observed values of *xvarname*. The required centiles are listed in *numlist*.

`mean` calculates and saves the mean of the fitted curves across replications at the observed values of *xvarname*.

`sd` calculates and saves the standard deviation of the fitted curves across replications at the observed values of *xvarname*.

`n` calculates and saves the frequencies of the observed values of *xvarname*.

6.5 Options for pmbeval (second syntax only)

`saving(`*newfilename*`)` is required and specifies the name of a new file to hold values of *xvarname* and its fitted functions suitable for stability analysis using `pmbstabil`. If *newfilename* already exists, it is replaced without warning.

6.6 Options for pmbevalfn

`clear` is required and signifies willingness for the data in the workspace to be replaced.

`rawdata(`*filename*`)` is required and specifies the name of the file that holds the desired values of *xvarname*. This file is typically the original data file used with `mfpboot`, but it need not be.

`standardize` standardizes each linear predictor to have mean zero.

6.7 Options for pmbstabil

`by(`*varname*`)` reports instability measures for two complementary subsets of the bootstrap replications:

1. Replications in which *varname* has nonzero, nonmissing values;
2. Replications in which *varname* is either zero or missing.

`conditional` calculates instability measures conditional on *xvarname* entering the model. The default is to compute unconditional measures assuming that $f(xvarname) = 0$ when *xvarname* does not enter.

`trunc(`#`)` specifies that $100 \times \#\%$ of the most extreme observations be dropped. # can be 0 or less than 1. The default is `trunc(0)`.

7 Instability measures for functions

As discussed by Royston and Sauerbrei (2008), selecting an FP function of a continuous covariate across bootstrap replications accesses a much wider range of candidate models than just finding the best-fitting MFP model in the original data. How should similarities or differences between such functions be defined and summarized? As already noted, using BIFs alone is insufficient. To supplement BIFs, Royston and Sauerbrei (2003) proposed some measures of variation between curves that we present in abbreviated form below. The key components of the approach were used to assess the instability of an additive spline model (Binder and Sauerbrei 2009). However, although the measures have been in existence for some years, there is little accumulated experience of their use. One motivation for making such tools available to the Stata community is to facilitate further research into them.

We present additional analyses of the prostate cancer data as an example.

7.1 Measures of curve instability

Each bootstrap replication in which a continuous covariate x enters the model generates a straight line or curve representing one estimate of the function. Variation in the intercept term across replications, although relevant to the calculation of bootstrap-based confidence intervals exemplified earlier in this paper, is of no interest in the present context and may appropriately be removed by standardization, as follows. Let $\widehat{f}_b\,(x)$ be the estimated function of x in the bth bootstrap replication. The standardized function $\widetilde{f}_b\,(x)$ is given by

$$\widetilde{f}_b\,(x) = \widehat{f}_b\,(x) - \frac{1}{n}\sum_{i=1}^{n} \widehat{f}_b(x_i) \tag{1}$$

where x_i is the value of x in the ith observation of the original dataset.

To summarize graphically the variation among functions in different replications, Royston and Sauerbrei (2003) calculated the cross-sectional mean (i.e., $q^{-1} f_{\text{bag}}(x)$, see below), together with an uncertainty band comprising, say, the 5th and 95th centiles of $\widetilde{f}_b(x)$ across the bootstrap replications. The mean and uncertainty band are plotted against x at sample values x_i.

Let us call the multivariable model of interest, i.e., that estimated on the original data, the reference model. Royston and Sauerbrei (2003) assessed the instability of the reference model according to the variability among the bootstrap-generated curves, $\widetilde{f}_b(x)$ $(b = 1, \ldots, B)$. Different sources of variability are present:

- Uncertainty associated with estimating the functional form—many different functional forms are available in the FP class;
- Uncertainty reflecting the "confounding" influence of other predictors, which may or may not happen to be selected, and for which the selected functional forms differ;
- Random variation around the curve.

Royston and Sauerbrei (2003) defined the following univariate measures for a continuous x of interest.

Let $f_{\text{ref}}(x)$ be the estimated function of x in the reference model, standardized as in (1). Let $f_{\text{ref}}(x) = 0$ if x is not selected in the reference model, and let $\widetilde{f}_b(x) = 0$ if x is not included in the model selected in replication b. For a sample S of replications, a bootstrap summary of the function is obtained using Breiman's (1996a) bagged (bootstrap-aggregated) estimator

$$f_{\text{bag}}(x) = \frac{1}{|S|} \sum_{b \in S} \widetilde{f}_b(x)$$

where $|S|$ denotes the number of members of S. By considering the decomposition,

$$\begin{aligned} T(x) &= \frac{1}{|S|} \sum_{b \in S} \left\{ \widetilde{f}_b(x) - f_{\text{ref}}(x) \right\}^2 \\ &= \frac{1}{|S|} \sum_{b \in S} \left\{ \widetilde{f}_b(x) - f_{\text{bag}}(x) \right\}^2 + \left\{ f_{\text{bag}}(x) - f_{\text{ref}}(x) \right\}^2 \end{aligned}$$

which may be written as

$$T(x) = V(x) + D^2(x)$$

the total variation, $T(x)$, of bootstrapped functions around the reference function is seen to be the sum of the within-subset variance, $V(x)$, and the squared deviation, $D^2(x)$, between the bagged and reference curves. Large values of $D^2(x)$ may indicate that the shapes of the reference curve and the bagged function differ. The function $V(x)$ quantifies the random variation and other contributions to variability of the individual curves $\widetilde{f}_b\left(x\right)$ around the bagged function. A derived measure that may carry useful information is the explained variation, $R^2\left(x\right)$,

$$R^2\left(x\right) = 1 - D^2\left(x\right)/T\left(x\right) = V\left(x\right)/T\left(x\right) = V\left(x\right)/\left\{V\left(x\right) + D^2\left(x\right)\right\}$$

A large value of $R^2\left(x\right)$ (i.e., near 1) indicates that the variation between the reference and bagged functions is small compared with the total amount of variation between the reference function and the bootstrap estimates. The interpretation is that the reference function is in good agreement with the bagged estimates, presumably the best estimate that the bootstrap can provide.

We also consider the conditional variance, $V_{\text{cond}}\left(x\right)$, of the bootstrap functions in the subset S_x in which x enters the model,

$$V_{\text{cond}}\left(x\right) = \frac{1}{\left|S_x\right|}\sum_{b\in S_x}\left\{\widetilde{f}_b\left(x\right) - \frac{1}{q}f_{\text{bag}}\left(x\right)\right\}^2$$

where $q = \left|S_x\right|/\left|S\right|$ is the BIF for x with respect to S. Division by $q \leq 1$ scales $f_{\text{bag}}\left(x\right)$ appropriately to allow for exclusion of the replicates in which $\widetilde{f}_b\left(x\right) = 0$. When x is always selected (i.e., when $q = 1$), then $V\left(x\right)$ and $V_{\text{cond}}\left(x\right)$ are identical, whereas for uninfluential variables (when q is low), $V\left(x\right)$ and $V_{\text{cond}}\left(x\right)$ may be very different.

$V\left(x\right)$, $D^2\left(x\right)$, $V_{\text{cond}}\left(x\right)$, and $T\left(x\right)$ are all functions of x. Summary measures may be obtained by averaging them over the empirical distribution function of x in the original sample,

$$\begin{aligned}
V &= \frac{1}{n}\sum_{i=1}^{n} V(x_i)\\
D^2 &= \frac{1}{n}\sum_{i=1}^{n} D^2(x_i)\\
V_{\text{cond}} &= \frac{1}{n}\sum_{i=1}^{n} V_{\text{cond}}(x_i)\\
T &= V + D^2\\
R^2 &= V/T
\end{aligned}$$

An advantage of averaging over the empirical distribution function of x, as opposed to, say, integrating the measures smoothly over the range of x, is that the results are concentrated on the region of x with the highest density. Outlying, "wild" function estimates at extremes of x tend to be downweighted, although they may still be influential.

7.2 Example

We reanalyze the prostate data to show how the stability functions $V\left(x\right)$, $D^2\left(x\right)$, and $R^2\left(x\right)$ may be calculated using `pmbeval`. Graphs of these against x are often informative. We then demonstrate how the summary measures V, D^2, and R^2 are computed using `pmbstabil`.

The following code shows the necessary calculations:

```
. // Compute and save f_ref, the reference function, standardized
. use mfpboot2, clear

. pmbeval if i == 0, clear xvar(cavol) rawdata(prostate_ca) standardize
(Prostate cancer data)

Data now comprise 93 observations for cavol.

. rename v0 f_ref

. save f_ref, replace
file f_ref.dta saved

. // Compute and save f_bag = _mean, V = _sd^2 and frequencies for each x,
> standardized
. use mfpboot2

. pmbeval if i > 0, clear xvar(cavol) rawdata(prostate_ca) standardize mean sd n
(Prostate cancer data)

Data now comprise 93 observations for cavol.

. rename _mean f_bag

. // Merge with f_ref
. merge 1:1 _n using f_ref

    Result                           # of obs.
    -----------------------------------------
    not matched                             0
    matched                                93  (_merge==3)
    -----------------------------------------

. drop _merge

. // Calculate V(x), D^2(x) and R^2(x)
. gen V = _sd^2

. gen Dsq = (f_ref - f_bag)^2

. gen Rsq = V / (Dsq + V)

. gen sV = sqrt(V)

. gen sDsq = sqrt(Dsq)

. drop _sd

. // Expand dataset to n observations
. expand _freq
(4 observations created)

. summarize V Dsq Rsq

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           V |        97    .0747896    .3606356   .0113916   3.524102
         Dsq |        97    .0155182    .0939165   7.14e-08   .9215094
         Rsq |        97    .8587237    .1070781   .6547148   .9999946
```

The first call to `pmbeval` computes the standardized reference function f_{ref}(`cavol`), whose coefficients and FP powers are stored in the first observation of `mfpboot2.dta`, indexed by `i==0`. The second call to `pmbeval` computes the mean f_{bag}(`cavol`), stored in `_mean`; the standard deviation, stored in `_sd`; and the frequency of each observed value of `cavol`, stored in `_freq`. V(`cavol`) is simply the square of `_sd`. Following some simple data management, D^2(`cavol`) is calculated via `gen Dsq = (f_ref - f_bag)^2`.

Figure 3 is a plot of $V(x)$, $D^2(x)$, and $R^2(x)$ against `cavol`.

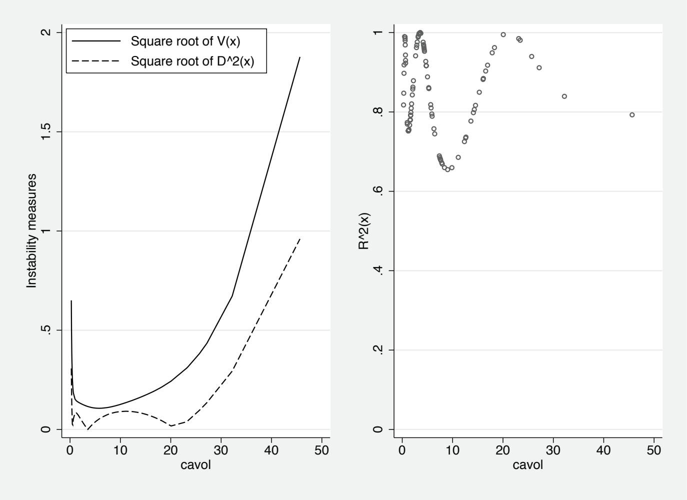

Figure 3. Left panel: Instability functions $V(x)$ and $D^2(x)$. To increase legibility, the square root of each measure has been plotted. Right panel: Explained-variation function $R^2(x)$. Functions are plotted against $x =$ `cavol`.

The plot of $V(x)$ is dominated by the largest observation at `cavol` = 45.65 ml. The associated variability between bootstrap replications is clearly seen in figure 1. Similarly, the difference between $f_{\rm bag}$ and $f_{\rm ref}$ is maximal at `cavol` = 45.65 ml, reflected in the largest value of $D^2(x)$ (also see figure 2). By contrast, the relative measure $R^2(x)$ fluctuates around its mean of 0.859.

The mean values of $V(x)$ and $D^2(x)$, i.e., the summary measures V and D^2, are 0.0748 and 0.01552, respectively. The values may be obtained more directly with `pmbstabil`, as follows:

```
. use mfpboot2, clear
. pmbeval, saving(cavol2) xvar(cavol) rawdata(prostate_ca) standardize
(Prostate cancer data)
file cavol2.dta saved
93 variables and 101 observations based on cavol saved to file cavol2.
. use cavol2
. pmbstabil
|S| = 100, N = 97, T = .09030785, V = .07478962, D^2 = .01551823,
> R^2 = .82816302
```

Note that the second syntax of `pmbeval` is used here. Without the smallest and largest values of `cavol`, restricted using the `trunc()` option, the values are as follows:

```
. pmbstabil, trunc(0.011)
|S| = 100, N = 95, T = .03996372, V = .03481303, D^2 = .00515069,
> R^2 = .87111594
```

`trunc(.011)` deletes 1.1% of the observations of the sample of size 97, i.e., the most extreme observation at each end of the distribution. As a result, the summary instability measures are considerably reduced and R^2 is increased.

8 Discussion

We have presented tools for applying the bootstrap to model selection with `mfp` and evaluating the output analytically and graphically. The `mfpboot` program is flexible in the sense that if the analyst wishes to impose an assumption of linearity on all continuous variables, all that is required is to specify the `df(1)` option of `mfpboot`. Variables are then selected by backward elimination only.

When calculating instability measures for continuous variables whose BIF is less than 100%, there is a choice between ignoring replications in which a variable was excluded and taking the estimated function as zero. The default in `pmbstabil` is to compute unconditional measures, i.e., assuming that $f(x)$ is zero when x does not enter. The alternative, `conditional`, ignores such replications. The issue is discussed in Royston and Sauerbrei (2003) and Royston and Sauerbrei (2008, 199). Furthermore, additional work on suitable measures for assessing (dis)agreement between functions is needed.

We have not compared the "full" MFP model with reduced MFP models in which variables and functions are selected at some chosen significance level. In the prostate data, it so happens (rather unusually) that the full MFP model gives more stable estimates of the function for `cavol` than do the selected models. The FP2 reference function for `cavol` in the full MFP model turns out to be similar to the bagged function, so the D^2 measure is small. Further investigation of the instability of a full MFP model is described by Royston and Sauerbrei (2008, 195–196).

In conclusion, we strongly recommend bootstrap investigation of model (in)stability. As has been pointed out more than once in the literature, particularly by Breiman (1996b), selecting a model according to the statistical significance of its variables in an adaptive algorithm is a fragile process that ignores many other possible models that may fit as well as or even better than the single chosen one. Selecting functions as well as variables increases the potential instability. The use of `mfpboot` should alert the analyst to these issues and enable a better-informed choice of a "final" model.

9 References

Augustin, N., W. Sauerbrei, and M. Schumacher. 2005. The practical utility of incorporating model selection uncertainty into prognostic models for survival data. *Statistical Modelling* 5: 95–118.

Binder, H., and W. Sauerbrei. 2009. Stability analysis of an additive spline model for respiratory health data by using knot removal. *Journal of the Royal Statistical Society, Series C* 58: 577–600.

Breiman, L. 1996a. Bagging predictors. *Machine Learning* 24: 123–140.

———. 1996b. Heuristics of instability and stabilization in model selection. *Annals of Statistics* 24: 2350–2383.

Buckland, S. T., K. P. Burnham, and N. H. Augustin. 1997. Model selection: An integral part of inference. *Biometrics* 53: 603–618.

Chen, C.-H., and S. L. George. 1985. The bootstrap and identification of prognostic factors via Cox's proportional hazards regression model. *Statistics in Medicine* 4: 39–46.

Royston, P., and W. Sauerbrei. 2003. Stability of multivariable fractional polynomial models with selection of variables and transformations: A bootstrap investigation. *Statistics in Medicine* 22: 639–659.

———. 2008. *Multivariable Model-building: A Pragmatic Approach to Regression Analysis Based on Fractional Polynomials for Modelling Continuous Variables.* Chichester, UK: Wiley.

Sauerbrei, W. 1999. The use of resampling methods to simplify regression models in medical statistics. *Journal of the Royal Statistical Society, Series C* 48: 313–329.

Sauerbrei, W., N. Holländer, and A. Buchholz. 2008. Investigation about a screening step in model selection. *Statistics and Computing* 18: 195–208.

Sauerbrei, W., P. Royston, and H. Binder. 2007. Selection of important variables and determination of functional form for continuous predictors in multivariable model building. *Statistics in Medicine* 26: 5512–5528.

Sauerbrei, W., and M. Schumacher. 1992. A bootstrap resampling procedure for model building: Application to the Cox regression model. *Statistics in Medicine* 11: 2093–2109.

Stamey, T. A., J. N. Kabalin, J. E. McNeal, I. M. Johnstone, F. Freiha, E. A. Redwine, and N. Yang. 1989. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. Radical prostatectomy treated patients. *Journal of Urology* 141: 1076–1083.

About the authors

Patrick Royston is a medical statistician with more than 30 years of experience. He has a strong interest in biostatistical methodology and in statistical computing and algorithms. At present, he works in clinical trials and related research issues in cancer. Currently, he is focusing on problems of model building and validation with survival data, including prognostic factors studies, on parametric modeling of survival data, on multiple imputation of missing values, and on novel trial designs.

Willi Sauerbrei has worked for more than two decades as an academic biostatistician. He has extensive experience of randomized trials in cancer, with a particular concern for breast cancer. Having a long-standing interest in modeling prognosis and a PhD thesis in issues in model building, he has more recently concentrated on model uncertainty, meta-analysis, treatment–covariate interactions, and time-varying effects in survival analysis.