The Gretl `fsreg` function package for running the forward stagewise regression

Artur Tarassow

Version 0.1

Changelog

- Version 0.1 (August, 2020)
  - initial release

Contents

1 Introduction

Selecting the relevant predictors among a many potential ones is a crucial task. Neglecting relevant predictors may lead to inconsistent parameter estimates while considering irrelevant regressors yields an inefficient estimator. Furthermore, considering highly correlated predictors into a standard least square regression most probably suffers from multicollinearity issues. Lastly, standard least squares cannot handle the case when the number of observations, T, exceeds the number of potential regressors, k.

So called shrinkage and/ or selection estimators such as Ridge or Lasso among others, are known to handle such issues. Another estimation approach marks the so called forward stagewise regression approach (FSREG henceforth). FSREG follows a simple strategy for constructing a sequence of sparse regression estimates: Initially set all coefficients to zero, and iteratively update the coefficient (by a small amount, depending on the *learning rate* ϵ) of the variable that achieves the maximal absolute correlation with the current residual. *Learning* from the residuals has some connection to Boosting. However, this procedure also has some interesting connection to the Lasso under some conditions (REFs). As the step size goes to zero, the sequence of forward stagewise estimates exactly coincides with the lasso path. While, this equivalence holds outside of least squares regression, currently we only support minimization of squared loss (RMSE).

2 The FSREG algorithm

The FSREG algorithm works as follows:[1]

1. Start with $r = y$ and $\beta_1 = \beta_2 = \ldots = \beta_k = 0$.

2. Find the predictor x_j most correlated with r.

3. Update the j-th predictor $\beta_j^i \leftarrow \beta_j^{i-1} + \delta_j$ where $\delta_j = \epsilon \times \text{sign} < r^{i-1}, x_j >$

4. Update the residuals $r^i \leftarrow r^{i-1} - \delta_j \times x_j$ and repeat steps 2 and 3 many times.

Here y, β, ϵ, $< r, x_j >$ and i refer to the endogenous variable, the unknown regression coefficients, the learning rate, the correlation between the current residuals and the j-th regressor and the i-th iteration. The learning rate ϵ is a hyper-parameter and set to a fixed constant (e.g., $\epsilon = 0.01$). The only computations intense task is to compute the correlation between r and the all k predictors. We make use of Gretl's `mcorr()` function for this which is written in C and hence computation is very fast.

The idea behind the stagewise updates is simple: at each iteration greedily select the predictor j that has the largest absolute inner product (or correlation, for standardized variables) with the residual. As the residuals refer to the yet unexplained part of the model, we are searching for any variable that still has some information content for explaining 'something' left unexplained.

Given the greediness that only a single predictor is selected each iteration, updating the coefficient of variable j by a large amount is problematic. Instead, the parameter ϵ slows down the learning process only changing the coefficient by a tiny amount. Thus, many iterations are required to yield reasonable parameter estimates among a large set of potential predictor variables. Figure 1 illustrates the coefficient path of the coefficient estimates over 2000 iterations.

Early stopping rules are important for two reasons: First, one tries to avoid over-fitting, and second it may be unnecessary to run $\bar{N}$ iterations if no improvement (in terms of model fit) can be seen after $N \ll \bar{N}$ iterations. The implemented early stopping strategy checks the absolute correlation $| < r, x_j^i > |$: In case the absolute correlation does not improve for n (e.g., $n = 30$)

[1] Also note, that this is analogous to least squares boosting, with the number of trees equal to the number of predictors predictors.

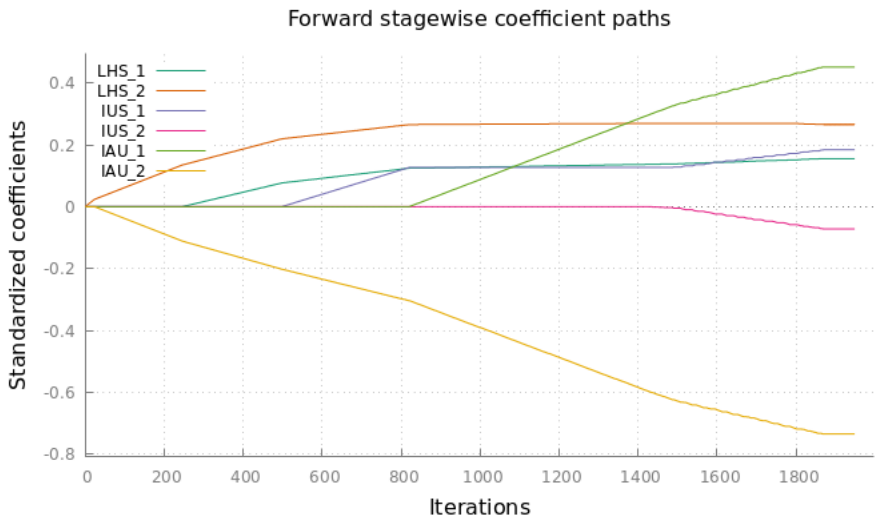

Figure 1: Coefficient paths

iterations, we assume that the coefficient estimates have converged and stop the algorithm. Figure 2 illustrates the development of the development of absolute correlation between the residuals and the selected variables.

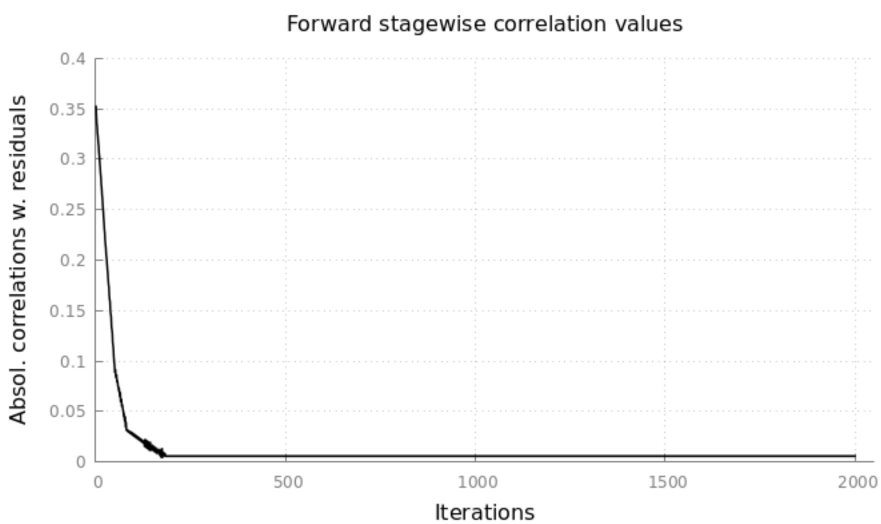

Figure 2: Correlation dynamics

3 Install and load the package

The `fsreg` package is publicly available on the gretl server. The package must be downloaded once, and loaded into memory each time gretl is started.

```
clear
set verbose off

pkg install fsreg          # Download package (only once need)
include fsreg.gfn          # Load the package into memory
help fsreg                 # Open the help file
```

4 Example

For illustration we use the “mroz” cross-sectional data set comprising 753 observations from a study of income dynamics. The endogenous variable is named `wage` and refers to the a wife’s 1975 average hourly earnings (in 1975 dollars). The data set includes additional 24 potential predictors.

The sample script opens the data set, sets the right-hand-side list of predictors and computes standard least square estimates.

```
open mroz.gdt --quiet
list RHS = const dataset
RHS -= wage             # drop lhs variable
ols LHS RHS              # run ols as benchmark
```

The OLS output is:

```
m: OLS, using observations 1-753
Dependent variable: LHS

                 coefficient    std. error    t-ratio    p-value
  --------------------------------------------------------------
  const           4.10278       2.37643        1.726     0.0847   *
  taxableinc     1.31268e-05   2.74615e-05   0.4780    0.6328
  federaltax      2.12780e-05   9.20550e-05    0.2311    0.8173
  hsiblings      0.000386927   0.0355359     0.01089   0.9913
  hfathereduc     0.0139611     0.0285757      0.4886    0.6253
  hmothereduc    0.0396817     0.0276745     1.434     0.1520
  siblings       0.0694937     0.0358372     1.939     0.0529   *
  lfp             3.45220       0.263461      13.10      2.25e-35 ***
  hours          0.00111994    0.000152678   7.335     5.91e-13 ***
  kidsl6         0.0423818     0.179218      0.2365    0.8131
  kids618        0.0329783     0.0703111     0.4690    0.6392
  age            0.00589606    0.0226630     0.2602    0.7948
  educ            0.216125      0.0492212      4.391     1.30e-05 ***
  wage76          0.543352      0.0463328     11.73      3.26e-29 ***
  hhours         0.000484660   0.000170424   2.844     0.0046   ***
  hage            0.00327033    0.0217384      0.1504    0.8805
  heduc          0.0511196     0.0359926     1.420     0.1560
  hwage          0.123328      0.0380489     3.241     0.0012   ***
  faminc          3.36999e-05   1.52480e-05    2.210     0.0274   **
  mtr            4.76933       2.38306       2.001     0.0457   **
  mothereduc     0.0282278     0.0299406     0.9428    0.3461
  fathereduc     0.0220970     0.0285042     0.7752    0.4385
  unemployment   0.0156837     0.0262039     0.5985    0.5497
  largecity       0.00659037    0.179593       0.03670   0.9707
  exper           0.00247946    0.0121007      0.2049    0.8377

Mean dependent var   2.374565   S.D. dependent var   3.241829
Sum squared resid    3351.283   S.E. of regression   2.145556
R-squared            0.575954   Adjusted R-squared   0.561974
F(24, 728)           41.19977   P-value(F)           2.5e-118
Log-likelihood      1630.588   Akaike criterion     3311.176
Schwarz criterion    3426.777   Hannan-Quinn         3355.711

Excluding the constant, p-value was highest for variable 3 (hsiblings)
```

Next, we run the forward stagewise regression by calling the `fsreg()` function:

```
bundle B = fsreg(LHS, RHS)
print_fsreg_results(B)      # Print estimation results
```

The regression results are as follows (note that inference is not supported, yet!):

```
Info: No improvement in correlation for the last 50 iterations.
 Early stopping applies.

Forward-stagewise regression results (no inference)
-------------------------------------------------------

                coefficient    std. error   z    p-value
  ------------------------------------------------------
  const          2.88386           NA        NA     NA
  taxableinc    -1.36310e-05       NA        NA     NA
  hmothereduc   -0.0483092         NA        NA     NA
  siblings      -0.0700839         NA        NA     NA
  lfp            3.27042           NA        NA     NA
  hours         -0.000930155       NA        NA     NA
  educ           0.213255          NA        NA     NA
  wage76         0.535865          NA        NA     NA
  hhours        -0.000272163       NA        NA     NA
  heduc         -0.0536584         NA        NA     NA
  hwage         -0.0766289         NA        NA     NA
  faminc         2.65937e-05       NA        NA     NA
  mtr           -3.88264           NA        NA     NA
  mothereduc    -0.0481345         NA        NA     NA

  Number of iterations = 2068
  Correl. w. residuals = -0.0329421
  S.E. of regression = 2.12071
  R-squared = 0.572159
```

The estimator converged after 2068 iterations and selected only 14 out of 25 potential predictors. The last correlation coefficient between the final residuals and some predictor was found being close to zero (-0.033). Even though only 14 predictors are selected being relevant, the R^2 statistics is of similar size of the OLS equivalent. Also the standard error of the residuals is slightly smaller with 2.12 compared to the OLS-based value of 2.14.

5 Using the GUI

Instead of scripting, one may access the `fsreg` procedure by means of the Gretl GUI. Once the package is installed and loaded, simply open the menu "*Model -> Other linear models -> Forward Stagewise*". This a menu window as depicted in Figure 3.

6 Public functions and parameter values

The following public functions currently exist.

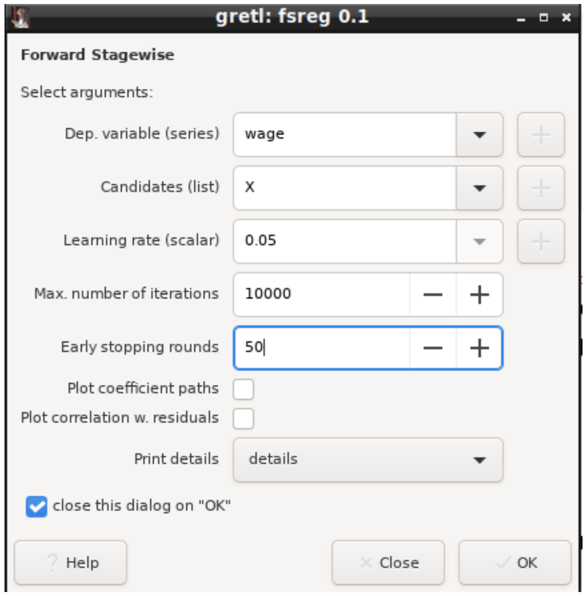

Figure 3: GUI access window

6.1 fsreg()

The `fsreg()` function marks the main function. The function arguments are:

```
fsreg(const series y, const list X, bundle opts[null])
```

Return type: `bundle`

Argument	Description
y	series, Endogenous variable
X	list, Non-empty list of predictor variables
opts	bundle, Pass parameters for controlling the algorithm (optional)

Return type: `bundle`

The returned bundle includes various which are listed in Table 1.

The additional parameters which can be passed by means of the `opts` bundle to `fsreg()` are shown in Table 2.

6.2 print_fsreg_results()

The `print_fsreg_results()` function takes the resulting bundle returned by the `fsreg()` function, and prints a summary of the estimation results. The argument is:

```
print_fsreg_results(const bundle B)
```

Return type: `void`

6.3 plot_rho_values()

For plotting the development of the remaining correlation with the residuals, simply call the `plot_rho_values()` function. It takes the resulting bundle returned by the `fsreg()` function. The argument is:

Parameter	Description
rho_values	Vector holding correlation coefficients with the residuals for each iteration
max_num_iterations	Number of the maximum iterations
actual_num_iterations	Actual number of iterations ran
eta	Learning rate
early_stopping_rounds	Number of iterations of no improvement before stopping
yname	String holding the name of the endogenous variable
Xnames_wo_constant	String array holding the names of all predictor variables without the constant
Xnames	String array holding the names of all predictor variables incl. the constant
X_final	List incl. finally selected predictors.
betas	Matrix holding the coefficient point estimates across iterations (rows) for each predictor (columns)
coeff_nonzero	k by 1 vector incl. the final coefficient point estimates for all selected predictors (non-zero coefficients)
coeff	n by 1 vector incl. the coefficient point estimates of all predictors (incl. zero coefficients)
with_constant	Boolean taking zero if the passed list X does not incl. an intercept, otherwise one
verbose	Integer indicating the level of verbosity
T	Number of effective (non-missing) observations.
yhat	Fitted values using final point coefficient estimates
uhat	Estimated final residuals
uhat_variance	Variance of the estimated final residuals
r2_qcorr	R-square based on quadratic correlation between actual y and fitted y

Table 1: Bundle content as returned by `fsreg()`.

```
plot_rho_values(const bundle B)
```

Return type: `void`

6.4 plot_coefficient_paths()

For plotting the development of the coefficients (coefficient paths), call the `plot_coefficient_paths()` function. It takes the resulting bundle returned by the `fsreg()` function. The argument is:

```
plot_coefficient_paths(const bundle B)
```

Return type: `void`

Parameter	**Data type**	**Description**	**Default value**
`eta`	scalar	Learning rate; $0 < \epsilon < 1$	0.05
`max_num_iterations`	int	Number of the maximum iterations	10000
`early_stopping_rounds`	int	Number of iterations of no improvement before stopping	50
`verbose`	bool	Print details or not: either 0 or 1	1 (True)

Table 2: Parameters which can be set through the optional bundle `opts`.