FEP – the forecast evaluation package for *gretl*

Artur Tarassow and Sven Schreiber*

version 2.3, October 2019

Abstract

The `FEP` function package for the *gretl* program is a collection of functions for computing different types of forecast evaluation statistics as well as tests. For ease of use a common scripting interface framework is provided, which is flexible enough to accommodate future additions. Most of the functionality is also accessible through a graphical dialog window interface within gretl. This documentation explains the usage and capabilities as well as providing some econometric background where necessary. An illustration with expert forecasts of euro area growth is also provided.

Contents

*Tarassow: atecon@posteo.de,
https://sites.google.com/site/arturtarassow;
Schreiber: IMK Düsseldorf and Free University Berlin, econ.svens.de.

1 Introduction

The `FEP` function package is a collection of gretl functions for computing different types of forecast evaluation statistics as well as tests. The FEP package currently comprises the functions listed in Table 1. In Section 6 we explain the easy usage from gretl's menu-driven interface (GUI), but until then we focus on the scripting way of performing the analysis. A function package can be downloaded and installed simply by invoking the install command:[1]

```
install FEP
```

Then, in each work session, as with all function packages the FEP package needs to be loaded by:

```
include FEP.gfn
```

2 The applyFCtests() function

This is a convenient top-level function which calls the other test functions in the background as needed. The standard workflow is to define the needed gretl bundle[2] with your input data and choices, and then call this function with the appropriate string code. Here's an example with the Mincer-Zarnowitz test:

[1] In future gretl versions (>2018c) this will be replaced by 'pkg install'.

[2] A "bundle" is a flexible gretl datatype which is basically a collection of other gretl objects. See Section 10.7 of the gretl user guide (as of November 2017).

Table 1: Included functions

#	Function	Code	Description
1	applyFCtests()		Top-level function
2	doDMtest()	DM	Diebold-Mariano regression based test of forecast accuracy
3	doGWtest()	GW	Giacomini-White regression based test of forecast accuracy
4	doCWtest()	CW	Clark-West regression based test of forecast accuracy of nested models
5	doMZtest()	MZ	Mincer-Zarnowitz regression based test of optimal (point) forecasts
6	doHPtest()	HP	Holden-Peel extension of the Mincer-Zarnowitz regression based test
7	doCGtest()	CG	Campbell-Ghysels sign or signed rank (Wilcoxon type) tests for unbiasedness or efficiency of forecasts
8	doEKTtest()	EKT	Elliott, Komunjer & Timmermann test of forecast rationality under flexible loss
9	doDLtest()	DL	Diebold-Lopez direction of change test
10	doKS()	KS	Computes the (Hanssen-) Kuipers Score for forecasts of binary outcomes
11	doPTtest()	PT	Pesaran-Timmermann test of market timing
12	DrawLoss()		draws a single loss function and its associated confidence interval
13	DrawLoss2()		draws two loss functions and their associated confidence intervals
14	getLoss()		calculate the (forecast error) loss series
15	doPS() / probscore()	PS	calculate the log and quadratic probability scores for forecasts of binary outcomes

```
bundle b
series b.y = growth # assuming 'growth' exists in current workfile
series b.fc = myforecast # ditto
applyFCtests(&b, "MZ")
```

The first three lines set up the bundle and put some relevant data in it.[3] Note that the names "y" and "fc" need to match exactly and are case sensitive. See Table 2 for the possible and/or necessary bundle elements and their names, which depends on the test that you wish to perform. It is possible to store additional elements in the bundle that are not used by a certain test. Therefore you can set up the bundle once and then pass it around as an input to various tests.

In the last line some things are noteworthy: The applyFCtests() function does not have any return value, so the function call stands for itself without any assignment to a variable. The results of the tests are instead added to the bundle ("b" here) that is passed to the function. To actually enable the function to change the input bundle we need to pass it in so-called pointer form, which just means to prepend the ampersand character: "&b".[4] Finally, a string code must be specified to indicate which test should be run, where the possible codes are given in Table 1. There appear also some supplementary functions in that table which do not have a string code; those functions have to be called directly and cannot be accessed through applyFCtests(). On the other hand, several string codes may be included in a single call to applyFCtests(), separated by spaces.

If the function is called like this, then there will typically be some printed output showing the test results. The details depend on the respective background function, see the corresponding documentation below. The other possibility to access the results is to inspect the new members that are added to the bundle. Again, see the documentation below to learn which new members each function adds to the bundle.

3 Forecast descriptive statistics

3.1 Calculate the (forecast error) loss

The function getLoss() helps to calculate the forecast losses (disutilities) implied by the given forecast errors, according to various loss function assumptions such as lin-lin, square, linex, double linex. See Table 3.

[3] If you're obsessed with saving lines of code, you might instead use something like the following: `bundle b = defbundle("y",growth, "fc",myforecast)`.

[4] See "Function programming details", Section 13.4 of the gretl user guide (as of November 2017).

Table 2: Bundle members needed for each function

	CG	DL	DM	GW	EKT*	HP	MZ	KS	PT	PS
series inputs ([ ]: optional)										
Realizations	y		y	y	y	y	y			
Forecast values	fc		f1, f2	f1, f2	fc	fc	fc			
Forecast errors (replaces y, fc)	(E)				(E)					
Binary indicator		yup						yup	yup	yup
Binary FC of yup		fcup						fcup	fcup	
FC of yup prob.										pfc
Further tested	[CGX]									
list inputs										
Exog. regressors				[z]		z				
Instruments					z					
scalar inputs (mostly integer / all are optional)										
FC horizon			[fhor]	[fhor]						
Loss type			[loss]	[loss]	[loss]					
Lag (efficiency test)	[k]									
Bootstrap iter.						[nboot]	[nboot]			
Robust switch						[robust]	[robust]		[robust]	
Initial shape					[a0]					
Conditional				[cond]						
Verbosity	[verb]	[verb]	[verb]	[verb]	[verb]	[verb]	[verb]	[verb]	[verb]	
string input (optional)										
Loss drawing					[lossdraw]					

Notes:

robust	can be 0 (default) or 1 (use HAC/robust SE);
loss	can be 1 (U-shape, default) or 2 (V-shape);
cond	can be 0 (unconditional test, default) or 1 (conditional test)
verb	can be 0 (no details, default) or 1 (print details);
lossdraw	can be "no" (default), "display", or consist of path + filename;
fcup	is binary, not a probability;
k	can be 0 (default, no test on lags) or a positive (not too large) integer; note that only a single lag is tested;
a0	can be between 0 and 1 (non-integer, default 0.5).

*: EKT refers to the doEKTtest() variant. For the matrix-based variant doEKTtest_matrix() see the source code.

Table 3: Forecast error loss

Function	`getLoss(matrix fce, string LF[null], matrix param[null], matrix realiz[null])`	
Description	Calculate the (forecast error) loss	
Return type	matrix	
Function arguments	matrix *fce*	T by k matrix of forecast errors; if *realiz* is given, *fce* is understood as the forecasts themselves
	string *LF* (optional)	specify loss function: "ll" (or "linlin"), "qq" (or "quadquad"), "sq" (or "square", default), "abs", "linex" (Varian), "dlinex" (or "dle") for double linex (Granger)
	matrix *param* (optional)	loss function parameter(s), default 0.5; *param* must be a 1-element or 2-element vector
	matrix *realiz* (optional)	matrix of realizations
Output	Returns a matrix with forecast error losses, of the same (T x k) dimension as the input matrix.	
Reference	Varian [1975], Granger [1999]	

Notes: About input format for *param:* For gretl versions until 2017a scalar values must explicitly be provided as a pseudo matrix (e.g {0.4}) while for later versions, gretl accepts a scalar as a 1x1 matrix.

Table 4: Draw losses

Function	`DrawLoss(int p, scalar aT, scalar V, string fpath[null])`
Description	Draw (forecast error) loss
Return type	none (void)
Function arguments	int *p:* loss function type, 1 = lin-lin, 2 = quad-quad scalar *aT:* loss function shape parameter $\alpha \in [0, 1]$ scalar V: estimated variance of *aT* string *fpath* (optional): null = display figure (default) or provide complete "path+file name" to store figure
Output	displays plot (or saves to file path)
Function	`DrawLoss2(int p, scalar aT1, scalar V1, scalar aT2, scalar V2, string fpath[null])`
Description	Draw two (forecast error) losses
Return type	none (void)
Function arguments	int *p:* loss function type, 1 = lin-lin, 2 = quad-quad scalar *aT1:* loss function shape parameter $\alpha \in [0, 1]$ scalar $V1$: estimated variance of *aT1* scalar *aT2:* loss function shape parameter $\alpha \in [0, 1]$ scalar $V2$: estimated variance of *aT2* string *fpath* (optional): null = display figure (default) or provide complete "path+file name" to store figure
Output	displays plot (or saves to file path)

3.2 Draw some loss functions

Draw a single (DrawLoss) or two (DrawLoss2) loss functions and their associated confidence intervals. See Table 4.

3.3 Tools for binary outcomes

3.3.1 The Kuipers Score (KS)

The (Hanssen-) Kuipers Score (KS) is used to evaluate binary outcomes. Let $f = \{0, 1\}$ be a forecast of the binary variable $y = \{0, 1\}$. The KS is defined as the difference between the probability of detection (POD) and the probability of false detection (POFD).[5] The POD is the proportion of times where $y = 1$ is correctly predicted. The POFD is the proportion of times where $y = 1$ is wrongly predicted. Thus, KS is defined as $KS = POD - POFD$. See Table 5 for the function interface and elements.

[5]It seems that this terminology is not universal. Sometimes the POD might be called hit rate, whereas usually the hit rate denotes something else. Similarly with the POFD and the false alarm rate.

Table 5: Kuipers Score

Function	`doKS(bundle *b)`	
Description	compute the Kuipers Score	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *yup*	binary-valued series of actuals that takes the value of unity for ups, otherwise zero
	series *fcup*	binary-valued forecast that takes the value of unity for ups, otherwise zero
	scalar *verb*	0 = no printout, 1 = print details
Output	The following new elements are stored into the model bundle:	
	scalar *KSstat*:	Kuipers score
	matrix *KSmat*:	matrix holding all classifications
Reference	Pesaran [2015, p. 396]	

Table 6: Probability scores

Function	`probscore(matrix y, matrix Pr)`
Description	computes the log (LPS) and quadratic (QPS) probability scores
Return type	matrix
Function arguments	matrix *y*: binary-valued vector of actuals that takes the value of unity for ups and otherwise zero matrix *Pr*: vector of forecast probabilities
Output	2-element (1 by 2) vector with QPS, LPS

3.3.2 Probability scores – doPS() and probscore()

The probscore() function computes forecast accuracy statistics used for probability forecasts. The observed outcome is still binary, but in contrast to the Kuipers score the forecast here is a continuous probability. See Table 6. The doPS() function is just a thin wrapper around probscore that is harmonized with the evaluation test functions in the package; see Table 7.

3.4 Tools for point forecasts – ForecastMetrics()

The ForecastMetrics() function comprises the computation of various evaluation metrics for point forecasts. For computation of five standard measures, we apply gretl's built-in function fcstats(). In total twelve measures are currently implemented – for details see Table 8). For further readings we refer to Hyndman and Koehler [2006]. Details on the ForecastMetrics() function can be found in Table 9.

Table 7: Probability scores wrapper

Function	`doPS(bundle *b)`	
Description	computes the log (LPS) and quadratic (QPS) probability scores	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *yup*	binary-valued series of actuals that takes the value of unity for ups, otherwise zero
	series *pfc*	probability forecast (between 0 and 1) that *yup* takes the value of unity
Output	The following new elements are stored into the model bundle:	
	scalar *qps*	quadratic probability score
	scalar *lps*	log probability score

4 Evaluation of individual forecasts

4.1 Mincer-Zarnowitz test of forecast unbiasedness – doMZtest()

Define the h-step ahead forecast made in t as $f_{t+h\,|\,t}$ and the actual outcome as y_{t+h}. Mincer&Zarnowitz suggest to run the following regression:

$$y_{t+h} = \beta_0 + \beta_1 f_{t+h|t} + u_{t+h}$$

Unbiasedness is viewed as a joint test of $\beta_0 = 0$ and $\beta_1 = 1$. (H0: Forecast is unbiased and efficient. H1: Forecast is biased and/or inefficient.) Usually the corresponding test statistics is compared with asymptotic critical values from the F-distribution.

However, remaining serial correlation in u_{t+h} yields inefficient estimates. Also the small sample properties are unknown. To account for these two potential sources of inefficiency, the user can compute HAC robust standard errors as well as bootstrap p-values. An intercept will be automatically inserted.

See Table 10 for a synopsis.

4.2 Holden-Peel test of forecast efficiency – doHPtest()

The Mincer-Zarnowitz regression can be extended to include another regressor (or a whole vector of additional regressors), Z_t, such that (assuming for simplicity that Z_t is a scalar value)

$$y_{t+h} = \beta_0 + \beta_1 f_{t+h|t} + \beta_2 Z_t + u_{t+h} \qquad .$$

The hypothesis to test for a strong form of unbiasedness involves the null $\beta_0 = 0$, $\beta_1 = 1$ and $\beta_2 = 0$. An intercept will be automatically inserted.

Table 8: Computations of point forecast evaluation measures

Measure	**Abbreviation**	**Details**
Mean error	ME	$mean(y_t - f_t)$
Root mean squared error	RMSE	$\sqrt{(mean\left((y_t - f_t)^2\right)}$
Mean absolute error	MAE	$mean(\lvert y_t - f_t \rvert)$
Mean percentage error	MPE	$mean\left(\frac{y_t - f_t}{y_t} \times 100\right)$
Mean absolute percentage error	MAPE	$mean\left(\frac{\lvert y_t - f_t \rvert}{y_t} \times 100\right)$
Median absolute percentage error	MdAPE	$median\left(\frac{\lvert y_t - f_t \rvert}{y_t} \times 100\right)$
Root mean square percentage error	RMSPE	$\sqrt{mean\left(\left[\frac{y_t - f_t}{y_t} \times 100\right]^2\right)}$
Root median square percentage error	RMdSPE	$\sqrt{median\left(\left[\frac{y_t - f_t}{y_t} \times 100\right]^2\right)}$
Symmetric mean absolute percentage error	sMAPE	$mean\left(\frac{\lvert y_t - f_t \rvert}{y_t + f_t} \times 200\right)$
Symmetric median absolute percentage error	sMdAPE	$median\left(\frac{\lvert y_t - f_t \rvert}{y_t + f_t} \times 200\right)$
Symmetric mean absolute absolute percentage error	sMAAPE	$mean\left(\frac{\lvert y_t - f_t \rvert}{\lvert y_t + f_t \rvert} \times 200\right)$
Symmetric median absolute absolute percentage error	sMdAAPE	$mean\left(\frac{\lvert y_t - f_t \rvert}{\lvert y_t + f_t \rvert} \times 200\right)$

Table 9: Point forecast measures

Function	`ForecastMetrics(series y, list fc)`
Description	Computes various evaluation statistics for point forecasts
Return type	matrix
Function arguments	series *y*: series of actuals list fc: list of *k* alternative forecasts
Output	12 by k matrix with 12 measures for k different forecasts

Table 10: Mincer-Zarnowitz test

Function	`doMZtest(bundle *b)`	
Description	computes the Mincer-Zarnowitz test regression of unbiasedness	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	actual observations
	series *fc*	h-step ahead forecast
	scalar *robust*	0 = no robust SEs (default), 1 = compute HAC SEs
	scalar *nboot*	0 = no bootstrap (default), or number of bootstrap iterations
	scalar *verb*	0 = no printout (default), 1 = print details
Output	The following new elements are stored into the model bundle:	
	scalar *MZstat*	test statistics
	scalar *MZpval*	p-value
Reference	Mincer and Zarnowitz [1969]	

Table 11: Holden-Peel test

Function	`doHPtest(bundle *b)`	
Description	computes the Holden-Peel variant of the Mincer-Zarnowitz test regression of unbiasedness	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	actual observations
	series *fc*	h-step ahead forecast
	list *z*	gretl list of additional control variables
	scalar *robust*	0 = no robust SEs (default), 1 = compute HAC SEs
	scalar *nboot*	0 = no bootstrap (default), or number of bootstrap iterations
	scalar *verb*	0 = no printout (default), 1 = print details
Output	The following new elements are stored into the model bundle:	
	scalar *HPstat*	test statistics
	scalar *HPpval*	p-value
Reference	Holden and Peel [1990]	

Again the user can compute HAC robust standard errors and/or bootstrap p-values; see Table 11.

4.3 Campbell-Ghysels efficiency tests – doCGtest()

Here we understand "efficiency" in a broad sense, comprising unbiasedness for example.

Let us define the one-period forecast errors as $e_{1t} = y_{t+1} - f_{t+1|t}$. An indicator function indicates whether the forecast error is positive or negative such that $u(z) = 1$ if $z \geq 0$ and $u(z) = 0$ otherwise. The test statistics of the sign test of unbiasedness of forecast errors is

$$S = \sum_{t=1}^{T} u(e_{1t}),$$

where T is the number of available forecast errors. While the signed rank test (see below) is defined as

$$W = \sum_{t=1}^{T} u(e_{1t}) R_{1t}^{+}$$

with R_{1t}^{+} referring to the rank of each forecast error when $|e_{1t}|$,..., $|e_{1T}|$ are placed in ascending order. The forecast errors are independent with zero median. The sign statistic S is binomially distributed with $Bi(T, 0.5)$. Under the additional assumption of symmetrically distributed forecast errors around zero, the test statistics W follows a Wilcoxon signed rank distribution. Note that this test is performed once

Table 12: Campbell-Ghysels sign and signed rank

Function	`doCGtest(bundle *b)`	
Description	Campbell-Ghysels sign and signed rank (Wilcoxon-type) tests for unbiasedness or efficiency of forecasts	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	actual observations
	series *fc*	forecast
	(or series *E*	forecast error)
	scalar *verb*	0 = no printout (default), 1 = print details
	series *CGX* (optional)	variable for orthogonality (see also *k*)
	scalar *k*	set $k > 0$ to test efficiency with respect to information at lag *k* (own lag of forecast error if *X* is absent), else $k = 0$ (or omit) to test for unbiasedness
Output	The following new elements are stored into the model bundle:	
	scalar *CGSIGNstat* and *CGWILCstat*	test statistic
	scalar *CGSIGNpval* and *CGWILCpval*	p-value
Reference	Campbell and Ghysels [1995], Campbell and Ghysels [1997], Dufour [1981]	

Notes: The calculation of CGWILCpval, the p-value of the Wilcoxon-type signed rank tests, depends on the WSRpvalue function in the special package "extra" for gretl ($\geq$v0.41). Please install that package manually from the gretl package repository if the automatic loading fails.

you set $k = 0$ (or leave it out, as this is the default; see Table 12).

This test idea can also be employed to test for serial correlation in the forecast errors. Construct the product series $Z^k_{1t} = e_{1t}e_{1(t-k)}$ with $k \geq 1$, and compute the statistics:

$$S_k = \sum_{t=k+1}^{T} u(Z^k_{1t}) \quad \text{and} \quad W_k = \sum_{t=k+1}^{T} u(Z^k_{1t})R^+_{2t}$$

where R^+_{2t} is the signed rank of the product Z^k_{1t}, $t = 1, \ldots, T$. These location tests were proposed by Dufour [1981]. Serial correlation in the forecast errors will move the centre of their product away from zero. The sign statistic S_k is binomially distributed with $Bi(T - k, 0.5)$. The test statistics W_k follows a Wilcoxon signed rank distribution of size $T - k$. Note, both tests on serial correlation require to set the option $k > 0$, and the necessary product series Z^k_{1t} will be constructed automatically.

Lastly, one can use this framework to assess whether the forecast has made efficient use of the available information represented by the series X in t. For this,

one needs to construct the product series $Z_t^k = e_{ht} X_{t-k}^c$ with $k > 0$ based on the recursively re-centered series $X_t^c = X_t - \text{median}(X_1, X_2, \ldots, X_t)$.[6] This way of re-centering requires, however, that the series X_t is stationary and has no trend.

The sign and signed rank statistics are given by

$$S_{ok} = \sum_{t=k+1}^{T} u(Z_t^k) \quad \text{and} \quad W_{ok} = \sum_{t=k+1}^{T} u(Z_t^k) R_{1t}^+,$$

noting that the ranks used refer to the forecast errors, not to Z_t^k, to obtain a statistic with a known and valid distribution; see Campbell and Ghysels [1995, pp. 3] or Campbell and Ghysels [1997, p. 560] for a discussion.[7] This orthogonality test can be achieved by passing the series as CGX in the bundle. See Table 12.

4.4 Elliott, Komunjer, and Timmermann test with asymmetric loss – doEKTtest()[8]

This is a test for forecast rationality that allows for asymmetric loss. As a side product the parameters of a general class of loss functions can be estimated.

Elliott et al. [2005] propose a flexible class of loss functions, the so-called EKT loss function:

$$L(e) = [\alpha + (1 - 2\alpha)\mathbf{I}(e < 0)] \, |e|^p, \quad \alpha \in [0, 1]$$

where $\mathbf{I}$ is an indicator function which equals one if the forecast error $e = y - f < 0$, and otherwise zero. Asymmetric loss is given for $\alpha \neq 0.5$, and where values exceeding 0.5 indicate greater aversion to positive forecast errors. Parameter p is a positive integer. Setting $p = 1$ results in a lin-lin loss function with possibly different slopes under asymmetric loss. The quadratic loss function is given for the special setting $p = 2$ and $\alpha = 0.5$. Asymmetric quadratic loss functions are produced under $p = 2$ and $\alpha \neq 0.5$.

The unknown loss function parameter α is estimated by means of a linear instrumental variable (IV) approach. The computation of the linear IV estimator $\hat{\alpha}$ is done iteratively. Using the same notation as in Elliott et al. [2005] we set the initial weighting matrix S to be the identity matrix I_d, where d refers to the number of

[6] "Recursive" in the sense that for the calculation of the median in each period t only realizations up to t are used, not all observations from the forecast evaluation sample. Otherwise the centering would contradict the real-time information flow and would not be feasible for actual forecasting.

[7] When X_t is strictly exogenous without any feedback occurring from the forecast errors to future realizations of X the ranks of Z_t^k could also be used, and this would then indeed equal a Wilcoxon signed rank test on Z_t^k. This variant is not implemented, however, because the assumption appears very strong and hardly relevant in practice.

[8] For historical reasons and backward compatibility the doEKTtest function up to FEP version 2.1 has operated on matrices. The preferred interface now is based on a bundle and series as described in this document. The doEKTtest_series function has fulfilled that role since version 2.0. Starting with version 2.3 the doEKTtest function will follow the preferred interface (be an alias for doEKTtest_series), and a new wrapper function doEKTtest_matrix will provide the old access for those who need it.

instruments used. Based on the initial S one can compute the initial $\hat{\alpha}_1$ which in turn can be used to update the precision matrix $S^{-1} = S^{-1}(\hat{\alpha}_1)$. These steps are repeated until some convergence criteria is met.

The user can use the framework to (i) test for symmetric loss, and (ii) for rationality. As $\hat{\alpha}$ is asymptotically normal and its variance is identical to the one obtained by the standard GMM estimator, one can test the null of symmetry, $H_0 : \alpha = 0.5$, against the alternative, $H_0 : \alpha \neq 0.5$, by a t-test.

For a given p_0 the user can test for rationality if the number of instruments $d > 1$. Testing for over-identification by means of a J-type test provides a joint test for rationality of the forecasts. The EKT approach allows to test for rationality (i) either for an unknown α or (ii) by imposing the restriction of symmetry $\alpha = 0.5$.

See Table 13 for the synopsis, and note that the instruments z should not contain an intercept, it will be added automatically.

4.5 Pesaran-Timmermann test of market timing – doPTtest()

While the Kuipers score just computes the difference between the hit rate, H, and the false alarm rate, F, it is not a proper statistical test. Pesaran and Timmermann [1992, PT] have proposed a simple test on market timing using binary outcomes. The basic idea is to test whether predicted ups are independent of actual ups or not.

Let $f = \{0, 1\}$ be a forecast of the binary variable $Y = \{0, 1\}$. Denote the corresponding time series of binary predictions as x_t and actual realizations of "ups" (or unity values) as y_t. How the forecaster obtains the forecasts x_t –e.g. through a model of a latent variable in the background– is irrelevant here.

The PT test statistic can be approximated by the t-ratio of the β_1 coefficient of the following OLS regression

$$y_t = \beta_0 + \beta_1 x_t + u_t.$$

Under the null hypothesis that predictions and realizations are independently distributed, the restriction $\beta_1 = 0$ holds. The test statistic follows asymptotically a standard normal distribution. A rejection of the null hypothesis indicates predictive failure. Serial correlation in the errors, u_t, are likely to occur but can be dealt with by using Bartlett weights to compute HAC standard errors.

A second –non-regression based– approach to compute the test instead is to rely on the correlation coefficient between forecasts and predictions, ρ. The test statistics is computed by $\rho \times \sqrt{T}$ where T refers to the number of forecasts available. The test statistic also follows a standard normal distribution asymptotically and cannot be robustified.

See Table 14 for a synopsis.

Table 13: Elliott, Komunjer and Timmermann test

Function	`doEKTtest_series(bundle *b) <doEKTtest(bundle *b) from v2.2>`	
Description	Elliott, Komunjer and Timmermann test for forecast rationality that allows for asymmetric loss	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	actual observations
	series *fc*	h-step ahead forecast
	(or series *E*	forecast error)
	list *z*	instruments
	scalar *a0*	initial value of shape parameter *alpha* (between 0 and 1, default 0.5)
	scalar *loss*	Loss function type: 1 = U-shape (default), 2 = V-shape
	scalar *verb*	0 = no printout (default), 1 = print details
	string *lossdraw*	"no" = no draw (default), "display" = immediate plot, "Path+filename"
Output	The following new elements are stored into the model bundle:	
	scalar *alpha*	estimated shape parameter
	scalar *V*	estimated variance of *alpha*
	scalar *SymTest*	test statistics of test for symmetric loss function
	scalar *SymPval*	p-value of test for symmetric loss function
	scalar *RatTest*	test statistics of test for rationality conditional on estimated *alpha*
	scalar *RatPv*	p-value of test for rationality conditional on estimated *alpha*
	scalar *RatTest05*	test statistics of test for rationality conditional on symmetry (as if alpha = 0.5)
	scalar *RatPv05*	p-value of test for rationality conditional conditional on symmetry (as if alpha = 0.5)
Reference	Elliott et al. [2005]	

Table 14: Pesaran-Timmermann test of market timing

Function	`doPTtest(bundle *b)`	
Description	Pesaran-Timmermann test of market timing based on binary outcomes	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *yup*:	binary-valued series of actuals that takes the value of unity for ups, otherwise zero
	series *fcup*:	binary-valued forecast that takes the value of unity for ups otherwise zero
	scalar *robust*:	0 = correlation-based PT test, 1 = regression-based with HAC robust SEs
	scalar *verb*:	0 = no printout, 1 = print details
Output	The following new elements are stored into the model bundle:	
	scalar *PTstat*	test statistic
	scalar *PTpval*	p-value
Reference	Pesaran and Timmermann [1992], Pesaran [2015, p. 398]	

4.6 Diebold-Lopez direction-of-change test – doDLtest()

Performs the Diebold-Lopez direction of change test which is in principle just Pearson's χ- square test. H0: The direction-of-change forecast has no value meaning that the forecasts and realizations are independent. H1: The direction-of-change forecast has some value. (The side output of *DLinfo* is closely related to the Kuipers score, namely by adding unity to it.)

See Table 15 for a synopsis.

5 Evaluation and comparison of multiple forecasts

5.1 Diebold-Mariano test – doDMtest()

First, note that there also exists a dedicated contributed gretl function package called "DiebMar.gfn" by Giulio Palomba. It only partly overlaps with the features of the doDMtest() function, so it might be useful for you, too.

The Diebold-Mariano (DM) test of equivalent expected loss takes explicitly into account the underlying loss function as well as sampling variation in the average losses.

We define the h-step ahead forecast error and its associated loss function by $e_{t+h|t}$ and $L(e_{t+h|t})$, respectively. The loss differential of two non-nested forecasts for observation $t+h$ is given by $d_{12,t+h|t} = L(e_{1,t+h|t}) - L(e_{2,t+h|t})$. Specifically, one can test the null hypothesis of equal predictive accuracy

$$H_0: \quad E(d_{12,t+h|t}) = 0$$

Table 15: Diebold-Lopez test

Function	`doDLtest(bundle *b)`	
Description	Diebold-Lopez test for predictive accuracy of a direction-of-change forecast	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *yup*	binary-valued series of actuals that takes the value of unity for ups, otherwise zero
	series *fcup*	binary-valued forecast that takes the value of unity for ups otherwise zero
	scalar *verb*	0 = no printout (default), 1 = print details
Output	The following new elements are stored into the model bundle:	
	scalar *DLstat*	test statistic
	scalar *DLpval*	p-value
	scalar *DLinfo*	additional info value
Reference	Diebold and Lopez [1996]	

against either an one-sided alternative or against a two-sided alternative. Under the null, the test statistics is $DM = \bar{d}_{12,t+h|t}/\widehat{\sigma}_{\bar{d}_{12,t+h|t}} \rightarrow N(0,1)$ where $\bar{d}_{12,t+h|t} = F^{-1}\sum_{j=1}^{F} d_{12,j,t+h|t}$ refers to the sample mean loss differential and $\widehat{\sigma}_{\bar{d}_{12,t+h|t}}$ is a consistent estimate of its standard deviation based on F forecasts available.

However, due to serial correlation in the forecast errors, we compute alternatively the following regression-based version using OLS combined with HAC robust standard errors:

$$d_{12,t+h|t} = \beta + u_t$$

where u_t is an *i.i.d.* zero-mean error term. The null of equal forecast accuracy between the two point forecasts is defined as $\beta = 0$. The test statistics, t_{DM}, follows asymptotically a standard normal distribution.

Harvey et al. [1997] have suggested the following small-sample corrected degress-of-freedom adjusted t-statistics

$$t_{HLN} = [1 + F^{-1}(1 - 2 \times h) + F^{-2}h \times (h-1)]^{0.5} \times t_{DM}$$

where F and h refer to the number of forecasts and the h-step forecast horizon. See Table 16 for a synopsis.

At the moment we only support two loss functions, namely symmetric linear (V-shape) and symmetric quadratic (U-shape) loss but in principle other loss functions could be applied and may be incorporated into the package in future versions.

Table 16: Diebold-Mariano test

Function	`doDMtest(bundle *b)`	
Description	Diebold-Mariano test for predictive accuracy, computation regression-based with HAC robust SEs	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	realized values
	series *f1*	forecast values of model 1
	series *f2*	forecast values of model 2
	scalar *fhor*	forecast horizon (default 1)
	scalar *loss*	Loss function type: 1 = U-shape (default), 2 = V-shape
	scalar *verb*	0 = print no details (default), 1 = print details
Output	The following new elements are stored into the model bundle:	
	series *L*	loss differentials
	scalar *DMstat*	test statistics (not small-sample adjusted)
	scalar *DMpvaln*	p-value based on standard normal
	scalar *DMpvalt*	p-value based on t-distribution
	scalar *DMpvalssc*	p-value using small-sample correction
Reference	Diebold and Mariano [1995], Harvey et al. [1997]	

5.2 Giacomini-White test – doGWtest()

The Giacomini-White (GW) test is a test on equal conditional predictive ability. Given that the population parameters are not known and must be estimated, it is the actual future loss that is of interest to the forecaster, rather than that based on some population value. The function doGWtest() implements two test approaches: (i) The unconditional test restricts attention to the forecast model whereas (ii) the conditional approach allows the evaluation of the *forecasting method*, which includes the model, the estimation procedure and the possible choice of estimation window.

The conditional test for forecasting performance tests whether the loss-differential between two forecasts is conditional on current information, z_t. The null hypothesis is then altered to

$$E\left[d_{12,t+h|t}(\hat{\beta}_{1,t}, \hat{\beta}_{2,t}, y_{t+h})|z_t\right] = 0$$

where z_t is the information set (in Gretl terms a list of potential variables) available at time t. In case the user selects the conditional approach, per default the lagged loss differential, $d_{12,t-h}$, which is available in period t for the $t+h$ forecast, enters the information set. Additional conditioning variables may be included into the list 'z'.

At the moment we only support two loss functions, namely symmetric linear (V-shape) and symmetric quadratic (U-shape) loss but in principle other loss func-

Table 17: Giacomini-White test

Function	`doGWtest(bundle *b)`	
Description	Giacomini-White test on equal conditional predictive ability, computation regression-based	
Return type	none (void)	
Function arguments	Pointer to the model bundle. This bundle includes as members:	
	series *y*	realized values
	series *f1*	forecast values of model 1
	series *f2*	forecast values of model 2
	scalar *fhor*	forecast horizon (default 1)
	scalar *loss*	Loss function type: 1 = U-shape (default), 2 = V-shape
	scalar *cond*	0 = unconditional test (default), 1 = conditional test
	list *z*	series to condition on (only if $cond = 1$)
	scalar *verb*	0 = print no details (default), 1 = print details
Output	The following new elements are stored into the model bundle:	
	series *L*	loss differentials
	scalar *GWstat*	test statistics (not small-sample adjusted)
	scalar *GWpval*	p-value based on χ^2-distribution
Reference	Giacomini and White [2006]	

tions could be applied and may be incorporated into the package in future versions.

5.3 Clark-West test – doCWtest()

TBW

6 Menu-driven (GUI) usage

To install the FEP package using the GUI one follows the usual steps for contributed function packages: Open the function package list window for example via the menu File / Function packages / On server, then find the FEP entry and click the install button (or mouse right-click and choose Installation in the context pop-up menu).

For the precise meaning of the inputs to the respective functions please see the function documentation above, but ideally, using this package from gretl's menu interface should be mostly self-explanatory. From the menus you can only execute one test at a time.

Figure 1 shows the layout of the central window where choices and specifications are entered. In order to keep the number of argument fields in this window within reasonable bounds, some fields have different overloaded meanings

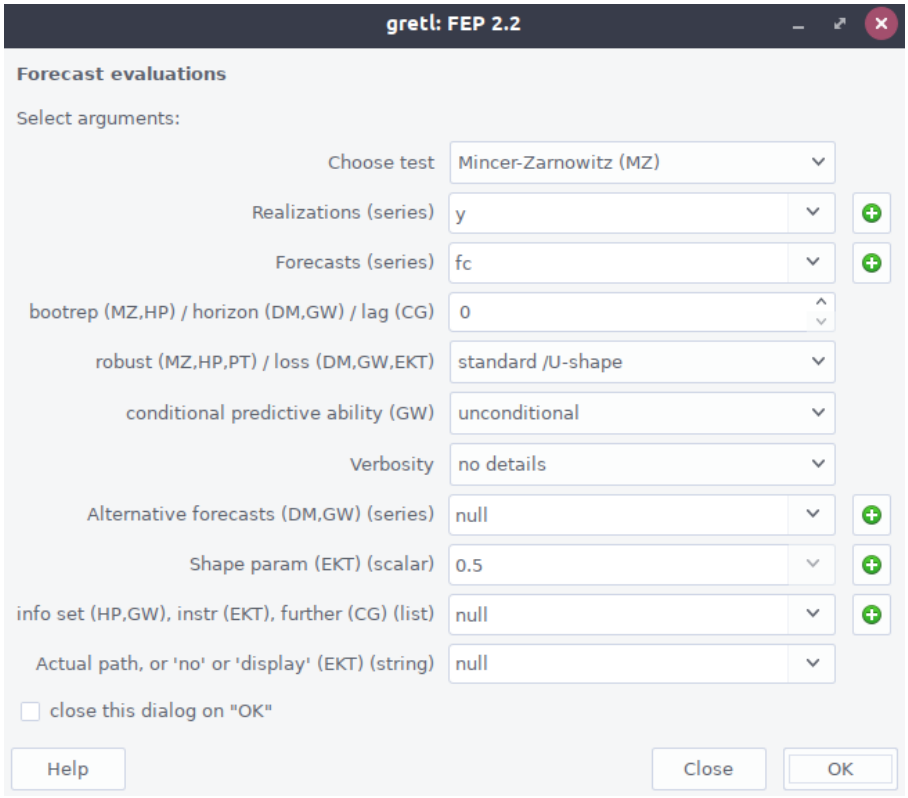

Figure 1: Screenshot of the FEP graphical interface (under Ubuntu Linux)

depending on which test is chosen. For example, the penultimate entry field expects a gretl list, and this input is relevant for the HP, DL, EKT, and CG variants.[9]

The output from executing the function from the GUI is presented to the user in a new window, mostly simply with the printed test result. At the top of that output there is a toolbar including "save" and "bundle" icons. The save button allows you to save first the textual output, and secondly the whole produced bundle to the current gretl session. The bundle button in turn gives you direct access to the various produced bundle members which can also be saved.[10]

Note that only the actual test functions are choosable from the GUI, not the helper functions from Section 3.

7 Illustrative example

In the following an applied example using survey-based forecasts for annual growth of euro area (EA) real GDP is presented. Forecasts are averages based on surveys among professional forecasters, fc_t, conducted by the Economist magazine and obtained through the Macrobond database. The realizations, y_t, are from the AMECO

[9]This will be transferred internally to the various function arguments "z", "ref", or "CGX", respectively.

[10]The concrete images of those icons will depend on your operating system and/or your desktop theme, but they are the first two buttons from the left.

database provided by the European Commission. Both series' values are reported in the data appendix A, and Figure 2 depicts the forecast errors, $e_t = y_t - fc_t$. For the following code to work we assume that a gretl annual time-series datafile is loaded with the series named `y` and `fc`.

In the following the main (but not all) functions of the FEP package are applied. Assuming the package is already installed on the local machine, at the beginning of the session the package must be loaded with:

```
include FEP.gfn
```

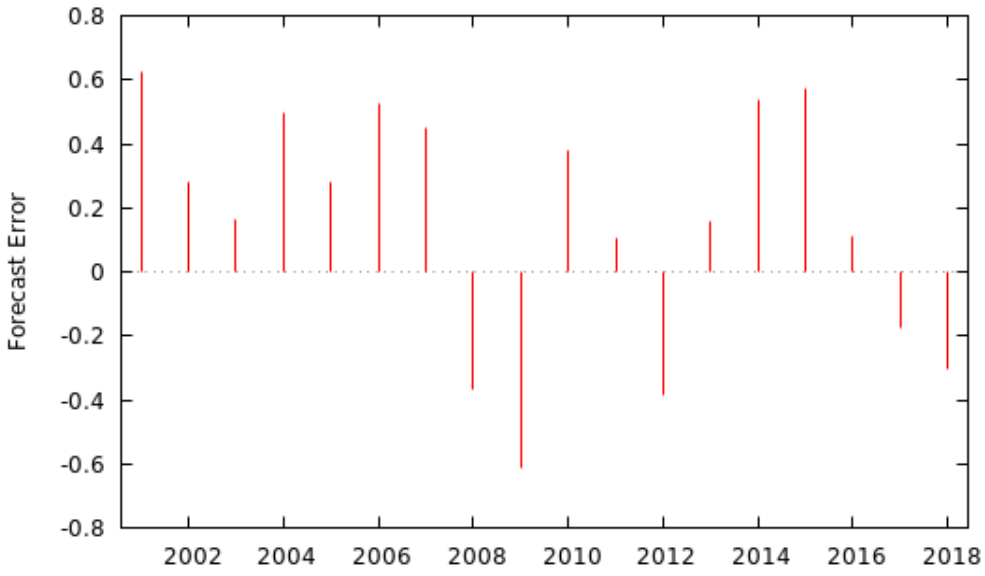

Figure 2: Forecast error for euro area growth of real GDP

7.1 Tests of unbiasedness

We run the parametric Mincer-Zarnowitz test on forecast unbiasedness using bootstrap p-values (999 iterations, *iid*) and HAC standard errors, see Section 4.1. First an empty bundle is defined, then the series and parameters are added:

```
bundle b = null
series b.y = y   # realizations
series b.fc = fc # forecasts
b.nboot = 999
b.robust = 1 # 1= HAC robust VCV
b.verb = 1   # 1= print details
applyFCtests(&b, "MZ") # call the top-level function
```

Yields as output:

```
*********************************************
*** Mincer & Zarnowitz test
*** on forecast unbiasedness
Method: Approach using HAC robust VCV.
Bootstrap p-value using 999 iterations.
H0: forecasts are unbiased
Test stat.: 5.6758
p-value.:   0.3874
*********************************************
```

As can be seen, the null hypothesis cannot be rejected at standard levels which indicates that the forecast errors are unbiased. (Please bear in mind that your exact numerical values may differ slightly due to different initializations of the bootstrap. To avoid this you would have to give gretl's random number generator a seed: `set seed <whatever>`)

A non-parametric alternative test approach for forecast unbiasedness was proposed by Campbell and Ghysels, as discussed in Section 4.3. Assuming that the previous code has been executed and thus `b`, `b.y`, `b.fc` and `b.verb` are already defined, the CG test can be performed as follows:[11]

```
applyFCtests(&b, "CG")
printf "P-value Wilcoxon-type rank sign test = %.3f\n", b.CGWILCpval
printf "P-value sign test = %.3f\n", b.CGSIGNpval
```

Yields as output:

```
*****************************************************
You selected the test(s) of unbiasedness.
*****************************************************
P-value Wilcoxon-type rank sign test = 0.122
P-value sign test = 0.096
```

These nonparametric tests do not provide strong evidence against unbiasedness, either. However, if a researcher assumed an underlying asymmetric distribution the only adequate test would be the nonparametric sign test, which might be considered borderline significant here.[12]

[11] Formatting the number printout as `%.3f` in the `printf` command is already slightly advanced, rounding to three decimal digits. A straightforward alternative that usually also works fine for floating point numbers is `%g`.

[12] The empirical skewness in this sample is -0.6. A skewed distribution would obviously imply non-Gaussianity; the Shapiro-Wilk test of the null of a normal distribution yields a p-value of 0.096, the Doornik-Hansen test in turn produces 0.152, thus somewhat borderline again.

7.2 Tests of efficiency

The Holden-Peel test is a parametric test of efficiency, see Section 4.2. We simply use lagged forecasts, fc_{t-1}, as the conditional variable. The HP test with bootstrap HAC standard errors is easily executed by:

```
list b.z = fc(-1) # use lagged forecast as conditional variable
applyFCtests(&b, "HP")
```

Yields as output:

```
**********************************************
*** Holden & Peel test
*** on forecast efficiency
Method: Approach using HAC robust VCV.
Bootstrap p-value using 999 iterations.
H0: forecasts are efficient
Test stat.: 8.1013
p-value.:   0.2763
**********************************************
```

Hence, this parametric test result does not indicate any issue with forecast efficiency conditional on fc_{t-1}.

A non-parametric version is also provided by the CG test approach which can be used as follows. Note that here `k` already specifies the lag to be applied to `CGX`, so the series for CGX should be given contemporaneously (here: `fc`), not with a lag (not: `fc(-1)`); otherwise errors due to missing values may occur.

```
b.k = 1
series b.CGX = fc
applyFCtests(&b, "CG")
printf "P-value Wilcoxon-type rank sign test = %.3f\n", b.CGWILCpval
printf "P-value sign test = %.3f\n", b.CGSIGNpval
```

Yields as output:

```
******************************************************
You selected the test(s) of orthogonality at lag 1.
******************************************************
P-value Wilcoxon-type rank sign test = 0.109
P-value sign test = 0.210
```

Thus, the non-parametric test results broadly agree with the parametric ones, although we might add that for lag 2 the result is quite different. (The reader is invited to verify this statement using the FEP package and the data in the appendix.)

7.3 Tests of (a)symmetric loss and forecast rationality

The framework proposed by Elliott et al. [2005] extends parts of the previous analysis to the case of asymmetric loss functions, cf. Section 4.4. We will use lagged forecast errors as an additional instrument apart from an intercept for illustration, and assume a quad-quad loss function. (Strictly speaking, the explicit specification of `b.loss` is redundant, since the quadratic U-shape is the default, see Table 2.)

```
b.loss = 1                    # Loss function (1=quad-quad, 2=lin-lin)
series e = y - fc # forecast error
series e1 = e(-1) # lagged forecast error
b.z = e1                 # instrumental variable
smpl e e1 --no-missing       # avoid mismatch
b.a0 = 0.5                    # Initial value of shape param aT
string b.lossdraw = "display"
applyFCtests(&b, "EKT")
smpl full
```

Yields as output:

```
Number of iterations before convergence = 8
*********************************************
Test for Symmetry: H0: aT=0.5 vs. H1: aT!=0.5
Estim. alpha =        0.236
Test stat. =         -2.41
P-value =        0.0158
*********************************************
*********************************************
Rationality Test
Estim. alpha =        0.236
J-statistics =         1.46
P-value =        0.227
*********************************************
*********************************************
Rationality Test
Alpha fixed to 0.5
J-statistics =         7.28
P-value =        0.0262
*********************************************
```

The null of symmetric loss can be rejected at the 5% level, and after imposing symmetry anyway one would indeed reject the null of forecast rationality at the 5%

level. However, under the estimated asymmetric loss function one cannot reject the null of forecast rationality. Here the suggested conclusion is that the forecasts may be regarded as "rational", but only if one departs from the assumption of a symmetric forecast error loss function.

The estimated $\hat{\alpha}$ is 0.236 which indicates greater aversion to negative forecast errors. This is perhaps not surprising considering the sequence of positive forecast errors in the first part of the sample. Apart from the reported test results, a plot of the estimated loss function is returned (see Figure 3). The upper and lower lines represent a confidence band due to the estimation uncertainty of the asymmetry parameter $\hat{\alpha}$.

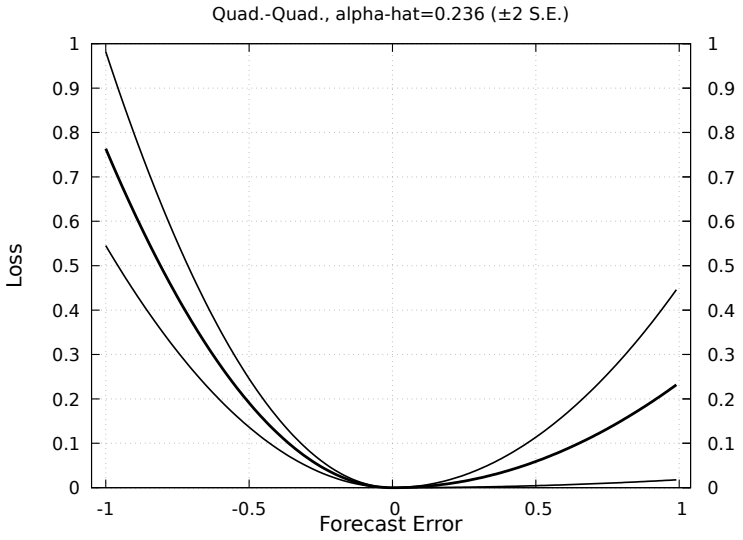

Figure 3: Estimated loss function using the EKT approach

7.4 Forecast comparison

The Diebold-Mariano approach tests for equal predictive accuracy of two competing forecasts (Section 5.1). We compare the current forecast for simplicity with the pre-year (naive) forecast assuming linear loss. The test is called by:

```
b.f1 = fc    # FC 1 series
b.f2 = y(-1) # FC 2 series: naive pre-year realization
b.loss = 2   # 1="U-shape", 2="V-shape"
applyFCtests(&b, "DM")
```

Yields as output:

```
*********************************************
*** Diebold & Mariano (1995) test
*** on equal forecast accuracy
```

```
Loss: V-shape linear.
Forecast horizon: 1
H0: forecasts are equally accurate
Test stat.: -2.5611
p-value (stand. normal): 0.0104
p-value (t-distributed): 0.0209
p-value (small-sample): 0.0178
********************************************
```

The null hypothesis can be rejected at the 5% significance level for all three versions computed. The negative test statistic (-2.56) indicates that the "real" survey-based forecast is more accurate compared to the naive one.

7.5 Directional forecast

Apart from point forecasts, directional forecasts may be of interest. The *Kuipers score* (KS) is a widely applied simple statistic to summarize directional forecasts, see Section 3.3.1. The following lines first compute the dummy variables `yup` and `fcup` which take the value of one if the period-change in realized (or forecast) values is positive (otherwise zero), and then execute the KS computation.

```
series b.yup = (diff(y) > 0)
series b.fcup = (diff(fc) > 0)
applyFCtests(&b, "KS") # could also use doKS(&b)
```

Yields as output:

```
************************
*** Kuipers Score ***
Hit Rate        = 1.000
False Alarm Rate = 0.100
Kuipers Score   = 0.900
************************
```

Accordingly, the forecasts have a perfect hit rate of 1 meaning that all "ups" are correctly predicted. The false alarm rate is only 10% such that the KS statistic is 0.9 which indicates a good directional forecast performance.

Finally, the *Diebold-Lopez* test (Section 4.6) tests whether a directional forecast is significantly different from a coin flip. For reasons of backwards compatibility with older FEP versions the bundle for the DL test currently must not contain other non-binary `y` and `fc` series, and therefore we cannot re-use the previously defined bundle. Instead we build a new bundle here.

The positive test statistic of 1.88 indicates that the actual forecast outperforms a pure coin flip which is statistically confirmed by the p-value.

```
bundle bDL = null
bDL.verb = 1
series bDL.yup = (diff(y) > 0)
series bDL.fcup = (diff(fc) > 0)
applyFCtests(&bDL, "DL")
```

Yields as output:

```
*********************************************
*** Diebold & Lopez test
*** on directional change
H0: y(t) and fc(t) are independent
Info-value: 1.8750
Test stat.: 13.3875
p-value.: 0.0003
*********************************************
```

In this context the Pesaran-Timmermann test described in Section 4.5 may be seen as an alternative test for directional forecasts.

References

B. Campbell and E. Ghysels. Federal budget projections: A nonparametric assessment of bias and efficiency. *The Review of Economics and Statistics*, 77(1):17 – 31, 1995.

Bryan Campbell and Eric Ghysels. An Empirical Analysis of the Canadian Budget Process. *Canadian Journal of Economics*, 30(3):553–76, August 1997. URL `https://ideas.repec.org/a/cje/issued/v30y1997i3p553-76.html`.

Francis X Diebold and Jose A. Lopez. Forecast evaluation and combination. In G. S. Maddala and C. R. Rao, editors, *Handbook of Statistics*, volume 14 of *Statistical Methods in Finance*, pages 241–268. 1996.

Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 20(1):134–144, 1995.

Jean-Marie Dufour. Rank tests for serial dependence. *Journal of Time Series Analysis*, 2(3):117–128, 1981. ISSN 1467-9892. doi: 10.1111/j.1467-9892.1981.tb00317.x. URL `http://dx.doi.org/10.1111/j.1467-9892.1981.tb00317.x`.

Graham Elliott, Ivana Komunjer, and Allan Timmermann. Estimation and testing of forecast rationality under flexible loss. *Review of Economic Studies*, 72:1107–1125, 2005.

Raffaella Giacomini and Halbert White. Tests of Conditional Predictive Ability. *Econometrica*, 74(6):1545–1578, 2006. URL https://ideas.repec.org/a/ecm/emetrp/v74y2006i6p1545-1578.html.

C.W.J. Granger. Outline of forecast theory using generalized cost functions. *Spanish Economic Review*, 1:161–173, 1999.

David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291, 1997. URL https://ideas.repec.org/a/eee/intfor/v13y1997i2p281-291.html.

K Holden and David Peel. On Testing for Unbiasedness and Efficiency of Forecasts. *The Manchester School of Economic & Social Studies*, 58(2):120–27, 1990. URL http://EconPapers.repec.org/RePEc:bla:manch2:v:58:y:1990:i:2:p:120-27.

Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. URL https://ideas.repec.org/a/eee/intfor/v22y2006i4p679-688.html.

J. Mincer and V. Zarnowitz. The evaluation of economic forecasts. In J. Mincer, editor, *Economic Forecasts and Expectations: Analysis of Forecasting Behavior and Performance*, pages 1 – 46. NBER, NBER, 1969.

M. Hashem Pesaran. *Time Series and Panel Data Econometrics*. Number 9780198759980 in OUP Catalogue. Oxford University Press, 2015. ISBN ARRAY(0x79444480). URL https://ideas.repec.org/b/oxp/obooks/9780198759980.html.

M Hashem Pesaran and Allan Timmermann. A Simple Nonparametric Test of Predictive Performance. *Journal of Business & Economic Statistics*, 10(4):561–65, 1992. URL https://ideas.repec.org/a/bes/jnlbes/v10y1992i4p561-65.html.

H.R. Varian. A Bayesian approach to real estate assessment. In S.E. Fienberg and A. Zellner, editors, *Studies in Bayesian Econometrics and Statistics in Honor of Leonard J. Savage*, pages 195–208. North-Holland, Amsterdam, 1975.

A Data

For replication purposes we reproduce the data used in the illustration in Section 7. For the definitions see the text; source Macrobond and AMECO.

	fc	y
2000		
2001	1.5	2.123350
2002	0.7	0.980211
2003	0.5	0.661216
2004	1.8	2.299620
2005	1.4	1.679013
2006	2.7	3.228410
2007	2.6	3.049257
2008	0.8	0.429603
2009	-3.9	-4.514502
2010	1.7	2.081707
2011	1.5	1.605639
2012	-0.5	-0.886516
2013	-0.4	-0.240252
2014	0.8	1.336558
2015	1.5	2.070988
2016	1.7	1.811255
2017	2.4	2.227502
2018	2.4	2.099555

B Changelog

- version 2.3 (October 2019)
  - Implement experimental version of Clark-West test on equal predictive accuracy of nested models.
  - New function ForecastMetrics() for computing various time-series related measures for the evaluation of point forecasts.
  - As previously announced, rename doEKTtest_series to doEKTtest and doEKTtest to doEKTtest_matrix to harmonize the interfaces. This is backwards-incompatible, but the minimal needed change is to follow the renaming to doEKTtest_matrix (or adapt your used interface).

  - remove several deprecated aliases: doCGWILCtest (use: doCGtest), doCGRANKtest (use: doCGtest), doKStest (use: doKS), doKGtest (use: doHPtest)
  - Remove deprecated compatibility handling of y and fc in doDLtest. (Need to use the documented labels "yup" and "fcup".)

- version 2.21 (August 2019)
  - replace deprecated isnull() function
  - refactoring code

- version 2.2 (December 2018)
  - add Giacomini-White test on conditional predictive ability
  - switch to new sample data set comprising annual EA GDP growth realizations and forecasts by the Economist Poll of Forecasters
  - In case the forecast horizon is chosen to be $fhor = 0$ for both the DM- and GW-Test, respectively, return a warning.

- version 2.11 (March 2018)
  - internal change (related to the 'strstr' function) to make FEP compatible with the next gretl version 2018a
  - extended and reformatted help (this document)

- version 2.1 (February 2018)
  - doCGtest():
    * re-centering of the CGX series for the orthogonality test is done properly (recursively) now
    * rewrite the entire CG test apparatus based on new function CamDufStats (referring to Campbell & Dufour 1995, not yet in references), properly taking into account the necessary different variants of constructing the ranks
    * remove the method switch CGmeth and always calculate both variants
  - add print-out option for doDLtest, doHPtest and doMZtest
  - fix bug and simplify the computation of the DL test

- version 2.0 (December 2017)
  - switch the documentation to this PDF file
  - harmonize the functions' interfaces, in a hopefully backwards-compatible way (old behavior not documented anymore)

  - introduce more default values (b.nboot = 0, b.fhor = 1, b.verb = 0...)
  - allow more than one test spec in applyFCtests()
  - For functions that deal with FC errors E, infer the respective series from the bundle input of realizations y and forecasts fc if E isn't directly given.
  - add probscore() and doPS()
  - deprecate doKStest() in favor of doKS() because it is not a test
  - deprecate doKGtest() in favor of doHPtest() because the reference is Holden-Peel
  - fix bug with the CG tests for $k > 1$
  - deprecate doCGRANKtest() and doCGWILCtest() in favor of single doCGtest() with method switch (also CGRANK didn't really have ranks in it); and switch to gretl built-in nonparametric test (difftest)

- version 1.3 (July 2017)
  - add a GUI wrapper
  - fix another bug with p-value calculation in the PT case
  - DL test: additional info value now in <bundle>.DLinfo, test stat in <bundle>.DLstat (backward incompatible change)

- version 1.25 (12.06.2017): Correct a bug in the computation of the p-value of the Pesaran-Timmermann test when using HAC S.E.

- version 1.24 (22.05.2017)
  - Add the getLoss() function for computing forecast error losses (written by Sven Schreiber)
  - no specific data type for usage of the FEP package required now

- version 1.23 (09.05.2017): Some small improvements in both DrawLoss() and DrawLoss2() (thanks to Sven Schreiber for suggestions)

- version 1.22 (08.03.2017)
  - corrected: Pesaran-Timmermann test should be a 1-sided not 2-sided alternative
  - DrawLoss() (related to doEKTtest) now also plots the 2 S.E. of the estimated loss function based on the estimated (a-)symmetry parameter alpha
  - add the function DrawLoss2() which can be called manually (see sample script) to plot jointly 2 loss functions as well as its associated 2 S.E.

- version 1.21 (26.01.2017): return classification matrix Ksmat computed for the Kuipers Score
- version 1.20 (15.01.2017)
  - Replace the DiebMar.gfn package (vers. 1.0) for computation of Diebold-Mariano test by own regression-based procedure using HAC robust S.E.
  - *** NOTE: The function's usage has changed from vers. 1.1 -> 1.2!
- version 1.10 (30.12.2016)
  - Correction of small bug in doDMtest() which occured for loss>2
  - add Kuipers Score
  - add Pesaran-Timmermann Test (1992)
- version 1.01 (24.11.2016): In doKGtest() the linear restriction was not correctly set.