

The extra package (a collection of various convenience functions for hansl programming)

The GRETl team*

October 2018, release 0.5, this is still small but growing...

Contents

1	Usage	1
2	Functions working without a dataset	2
2.1	commute	2
2.2	eliminate	2
2.3	duplicate	2
2.4	nearPSD	2
2.5	scores2x2	2
2.6	sepstr2arr	3
2.7	truncnorm	3
2.8	zeroifclose	3
2.9	WSRcritical	4
2.10	WSRpvalue	4
2.11	powerset	4
3	Functions requiring a dataset	4
3.1	gap_filler	4
3.2	winsor	5
4	Authors	5
5	Changelog	5

1 Usage

This package is intended for hansl scripting, not for gretl's GUI. (But of course other contributed function packages that make use of functions in extra.gfn can provide GUI access for themselves.)

The usual one-time requirement is to do `pkg install extra.zip` to get a copy on the local system (or install it via gretl's graphical mechanism), and then in the respective hansl script have a line `include extra.gfn`.

*Currently co-ordinated by Sven Schreiber.

Note that functions that are exact lookalikes of Matlab functions do not live here, but would go into the `matlab_utilities` package.

2 Functions working without a dataset

2.1 commute

Arguments: matrix `A`, int `m`, int `n` (optional), bool `post` (optional)

Return type: matrix

Returns A premultiplied by K_{mn} (the commutation matrix; more efficient than explicit multiplication). In particular, `commute(vec(B), rows(B), cols(B))` gives $vec(B')$. The optional argument n defaults to m (giving $K_{mm} = K_m$). If the optional arg `post` is non-zero, then does post-multiplication ($A \times K_{mn}$).

2.2 eliminate

Arguments: matrix `vecA`

Each column of the input `vecA` is assumed to come from the operation `vec(A)` on a square matrix, thus `rows(vecA)` must be a square number. Returns `vech(A)`, which is the result of pre-multiplying `vec(A)` with the "elimination" matrix L_m . If `vecA` has several columns, each column is treated separately as described above (and the results stacked side-by-side).

Return type: matrix

2.3 duplicate

Arguments: matrix `vechA`

The input is a vector assumed to come from an operation like `vech(A)`. Returns `vec(A)`, which is the result of pre-multiplying `vech(A)` with the "duplication" matrix D_m . If `vechA` has several columns, each column is treated separately as described above (and the results stacked side-by-side).

Return type: matrix

2.4 nearPSD

Arguments: matrix pointer `*m`, scalar `epsilon` (optional)

Return type: scalar

Forces the matrix m into the positive semi-definite region. Algorithm ported from "DomPazz" in Stackoverflow, apparently mimicking the `nearPD()` function in R. Because of re-scaling (to correlation matrix), the `epsilon` criterion value should implicitly apply to the correlation-based eigenvalues. The return value 0 or 1 indicates whether `m` was altered or not.

2.5 scores2x2

Arguments: matrix `in`, bool `verbose` (optional)

Return type: matrix

Computes some standard score measures for a 2×2 contingency table of the form:

		Observed	
		1	0
Predicted	1	h(its)	f(alse)
	0	m(iss)	z(eros)

and $n = h + f + m + z$ (total observations). Returns a column vector with the following elements:

1. POD / prob of detection = $h / (h + m)$
2. POFD / prob of false detection = $f / (f + z)$
3. HR / hit rate = $(h + z) / n$
4. FAR / false alarm rate = $f / (h + f)$
5. CSI / critical success index = $h / (h + f + m)$
6. OR / odds ratio = $h * z / (f * m)$
7. BIAS / bias score = $(h + f) / (h + m)$
8. TSS / true skill stat = $POD - POFD$
The TSS is also known as the Hanssen-Kuipers score, and is $= h / (h + m) - f / (f + z)$.
9. HSS / Heidke skill score = $2 * (h * z - f * m) / ((h + m) * (m + z) + (h + f) * (f + z))$
10. ETS / equitable threat score = $(h * z - f * m) / ((f + m) * n + (h * z - f * m))$
11. PRC / precision = $h / (h + f)$
12. FSC / F-Score = $2 * (PRC * POD) / (PRC + POD)$
The F-Score can also be expressed as $2 * h / (1 + h + m)$.

The input is always sanitized by taking the upper 2x2 part, using absolute values, and integer-ization. Warnings are issued if `verbose` is 1.

2.6 sepstr2arr

Transforms comma-separated string to an array of strings. (Retired as of v0.5, use the extended functionality of `gretl`'s built-in `strsplit` function.)

2.7 truncnorm

Arguments: `int n`, `scalar m`, `scalar sigma`, `scalar below`, `scalar above`
Return type: `matrix`

Generates n truncated normal random values. Specify mean `m` and std.dev. `sigma`, and the left/right truncation values `below` and `above`. (Pass NA for any one of them to skip the respective truncation.) Returns a col vector of values.

2.8 zeroifclose

Arguments: `matrix pointer *m`, `scalar thresh` (optional)
Return type: `scalar`

Sets elements of `m` to zero if they are really close. The return value 0 or 1 indicates whether `m` was altered or not.

2.9 WSRcritical

Arguments: `int n`, `scalar prob` (optional), `bool forcenorm` (optional)

Concerns the distribution of Wilcoxon's signed rank test statistic for n trials (at least 4). Tries to find the critical values (low/hi) where the two-sided area to the outside is as close as possible to the given `prob` (default: 0.05). (Note that "outside" means including the critical values themselves in the exact/discrete case.) If we end up in the interior region not covered by the exact table (for `prob` far away from 0 and also from 1), we fall back to the normal approximation. Returned is col vector $\{\text{low}; \text{hi}; \text{epv}\}$, where `epv` is the actual probability mass (close to `prob` but not equal in general for small samples). 'low' and 'hi' can be non-integers in the normal approximation case. The normal approximation instead of the exact table values can be enforced with the `forcenorm` argument (default: zero, do not enforce).

Return type: `matrix`

See also the sister function `WSRpvalue`.

2.10 WSRpvalue

Arguments: `int n`, `scalar W`, `bool forcenorm` (optional)

Concerns the distribution of Wilcoxon's signed rank test statistic for n trials (at least 4), returns $P(X \geq W)$. In the interior region not covered by the exact table, the true value is $\geq 12.5\%$ (and $\leq 87.5\%$) according to the table used,¹ so typically based on such values H_0 would not be rejected. We fall back to the normal approximation in this region. In the extreme outer regions not explicitly covered by the table, the deviation from 0 or 1 will be smaller than $0.5\% = 0.005$. We return values 0.001 or 0.999 as an approximation here. The test statistic W should usually be an integer, but in case of bindings it could be fractional as well; in this case we also fall back to the normal approximation.

The normal approximation instead of the exact table values can be enforced with the `forcenorm` argument (default: zero, do not enforce).

Return type: `scalar`

See also the sister function `WSRcritical`.

2.11 powerset

Arguments: `strings S`

Computes the powerset of the input S , i.e. all possible combinations of the string elements in S . (Including the empty set / empty string `""`.) Each combination yields one string in the output array. Being a set, the ordering is not defined and arbitrary.

Return type: `strings (array)`

3 Functions requiring a dataset

3.1 gap_filler

Arguments: `series x`, `int method` (optional)

Return type: `series`

¹Source of the table: Wilfrid J Dixon and Frank J. Massey, Jr., Introduction to Statistical Analysis, 2nd ed. (New York: McGraw-Hill, 1957), pp. 443-444.

Simple convenience function to crudely get rid of missing values interspersed between valid observations. Apart from the first argument (`series`) accepts an integer parameter as second argument, whose meaning is: 0: do nothing, leave the gaps; 1: NAs are replaced with previous observations; 2: NAs are replaced with a linear interpolation. Returns the filled series.

Note that the function only replaces NAs between valid observations; therefore, if the origin series has missing values at the beginning or the end of the sample, they will be in the returned series too.

3.2 winsor

Arguments: series `x`, scalar `p` (optional), scalar `phi` (optional)

Return type: series

Returns a trimmed (“winsorized”) version of the series, where outliers are replaced with implicit threshold values. Truncation quantiles are determined according to relative tail frequencies `p` and `phi`. Default lower and upper frequencies are 0.05, but re-settable with `p`. Pass `phi` in addition to `p` for an asymmetric trimming, then `p` determines only the lower frequency and `phi` the upper.

4 Authors

- `gap_filler`, `commute`, `eliminate`, `duplicate`, `truncnorm`, `powerset`: Jack Lucchetti
- `nearPSD`, `zeroifclose`, `scores2x2`, `WSRcritical`, `WSRpvalue`: Sven Schreiber
- `winsor`: Sven Schreiber, original code JoshuaHe

5 Changelog

- October 2018: 0.5, fix small `commute` bug; retire `sepstr2arr`; add `powerset`, `eliminate`, `duplicate`
- February 2018: 0.41, allow non-integer input in `WSRpvalue`
- January 2018: 0.4, add `WSRcritical`, `WSRpvalue`
- December 2017: 0.3, add `scores2x2`; switch to pdf help document
- September 2017: 0.2, add `winsor`
- July 2017: initial release