Package: tidyMacro (via r-universe)

November 28, 2026

Type Package

Title Structural Vector Autoregressions and Local Projections via 'C++'

Version 1.1.2

Date 2026-10-22

Encoding UTF-8

Description Estimation and identification of structural vector autoregressions and local projections, with a compiled 'C++' backend. Shocks can be identified by short-run (recursive) restrictions, long-run restrictions following Blanchard and Quah (1989) <https://www.jstor.org/stable/1827924>, maximum forecast error variance share following Uhlig (2005) <doi:10.1016/j.jmoneco.2004.05.007>, external instruments (proxy SVAR) following Mertens and Ravn (2013) <doi:10.1257/aer.103.4.1212> and Kaenzig (2021) <doi:10.1257/aer.20190964>, heteroskedasticity across regimes, and sign or narrative sign restrictions. Local projections follow Jorda (2005) <doi:10.1257/0002828053828518> and cover instrumental variables, state-dependent and nonlinear specifications, panel data as in Mian, Sufi and Verner (2017) <doi:10.1093/qje/qjx017>, and the difference-in-differences estimator of Dube, Girardi, Jorda and Taylor (2025) <doi:10.1002/jae.70000>. Inference uses residual, wild and moving-block bootstraps, heteroskedasticity and autocorrelation consistent standard errors, and one-way or two-way cluster-robust standard errors.

License MIT + file LICENSE

Imports Rcpp, tibble, ggplot2, tidyr, dplyr, rlang, tidyselect

Suggests readxl, sandwich, testthat (>= 3.0.0), parallel

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 4.1.0)

LazyData true

URL https://github.com/muhsinciftci/tidyMacro

BugReports https://github.com/muhsinciftci/tidyMacro/issues

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Muhsin Ciftci [aut, cre, cph]

Maintainer Muhsin Ciftci <farmer.muhsin@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-27 22:09:26 UTC

RemoteUrl https://github.com/cran/tidyMacro

RemoteRef HEAD

RemoteSha b20b54858e7d6713c0fc2016d71f0d8eccaed19a

Contents

tidyMacro-package 4
ACR2019 5
ADRR2018 6
BankingDeregulation 8
BeaudryPortier2014 9
Bloom2009 10
CBS2022 11
DemocracyGrowth 12
fAICBIC 13
fBootstrapBQ 14
fBootstrapChol 15
fBootstrapCholCorrected 17
fBootstrapHDIV 18
fBootstrapHetero 20
fBootstrapIVInvertible 22
fBootstrapIVMBB 23
fBootstrapIVRecover 25
fBootstrapMax 27
fBootstrapUhlig 29
fBootstrapVAR 30
fBQIRF 31
fCheckRestrictions 33
fCheckStability 34
fCholeskyIRF 35
fCompanionMatrix 36
fFEVDChol 37
fFEVDIV 38
fGenerateQ 39
fGenerateVARData 40
fGetBands 41

fGetShock . . . 42
fHDIV . . . 43
fHDShock . . . 44
fHeteroIRF . . . 45
fHistDec . . . 46
fIRF . . . 47
fLagMakerMatrix . . . 50
fLP . . . 51
fLPDID . . . 53
fLPIV . . . 55
fLPPanel . . . 57
fMaxIRF . . . 60
fMBBVAR . . . 61
fMSW . . . 62
fNLLP . . . 63
fOLS . . . 67
fPlotHistDec . . . 68
fPlotIRF . . . 70
fPlotLPDID . . . 72
fPlotNLLP . . . 73
fPlotVarDec . . . 74
fPolyConvolve . . . 75
fSetTidyMacroTheme . . . 76
fSignRestr . . . 77
fSpectralFEVD . . . 80
fSVARSteps . . . 81
fTestInvertibility . . . 82
fTestRecoverability . . . 84
fThemeTidyMacro . . . 85
fUhligIRF . . . 86
fUhligMaxShare . . . 87
fVAR . . . 88
fVARX . . . 89
fWoldIRF . . . 90
Gali1999 . . . 91
GK2015 . . . 92
GK2015_2 . . . 93
JordaTaylor2025 . . . 94
JordaTaylor2025IV . . . 95
Kaenzig2021 . . . 96
MianSufiVerner2017 . . . 97
MianSufiVerner2018 . . . 98
predict.fNLLP . . . 99
scale_color_tidyMacro . . . 100
scale_fill_tidyMacro . . . 101
TenreyroThwaites2016 . . . 102
tidy.fLP . . . 103
tidy.fLPPanel . . . 104

tidy.fNLLP 105
tidyMacro_colors 106
Uhlig2005 107

Index **109**

`tidyMacro-package` *tidyMacro: Structural Vector Autoregressions and Local Projections via 'C++'*

Description

Estimation and identification of structural vector autoregressions and local projections, with a compiled 'C++' backend. Shocks can be identified by short-run (recursive) restrictions, long-run restrictions following Blanchard and Quah (1989) https://www.jstor.org/stable/1827924, maximum forecast error variance share following Uhlig (2005) doi:10.1016/j.jmoneco.2004.05.007, external instruments (proxy SVAR) following Mertens and Ravn (2013) doi:10.1257/aer.103.4.1212 and Kaenzig (2021) doi:10.1257/aer.20190964, heteroskedasticity across regimes, and sign or narrative sign restrictions. Local projections follow Jorda (2005) doi:10.1257/0002828053828518 and cover instrumental variables, state-dependent and nonlinear specifications, panel data as in Mian, Sufi and Verner (2017) doi:10.1093/qje/qjx017, and the difference-in-differences estimator of Dube, Girardi, Jorda and Taylor (2025) doi:10.1002/jae.70000. Inference uses residual, wild and moving-block bootstraps, heteroskedasticity and autocorrelation consistent standard errors, and one-way or two-way cluster-robust standard errors.

Author(s)

Maintainer: Muhsin Ciftci <farmer.muhsin@gmail.com> [copyright holder]

Authors:

- Muhsin Ciftci <farmer.muhsin@gmail.com> [copyright holder]

See Also

Useful links:

- https://github.com/muhsinciftci/tidyMacro
- Report bugs at https://github.com/muhsinciftci/tidyMacro/issues

ACR2019 *Arias, Caldara & Rubio-Ramirez (2019) Data with the Romer-Romer Instrument*

Description

Monthly US data for the six-variable monetary policy VAR of Arias, Caldara and Rubio-Ramirez (2019), together with the Romer and Romer (2004) narrative monetary policy shock used as an external instrument. Covers January 1965 - December 2007.

Usage

```
ACR2019
```

Format

A data frame with 516 rows and 8 columns:

Date Month start date (`Date`), 1965-01 - 2007-12

Real GDP Monthly real GDP, level (interpolated to monthly)

GDP Deflator Monthly GDP deflator, level

Commodity Price Idx. Commodity price index, level

Total Reserves Total reserves, level

Non-Borrowed Reserves Non-borrowed reserves, level

Fed. Funds Rate Federal funds rate, in percent

RR Romer-Romer narrative monetary policy shock, in percentage points. `NA` before January 1969, where the series does not exist.

Details

The first six series are the same variables as `Uhlig2005` and `ADRR2018`, which makes the three directly comparable. They are stored as levels rather than logs: the replication code takes `100 * log()` of all but the funds rate before estimation, and that transformation is left explicit rather than baked into the object.

`RR` is a narrative instrument, so the 118 exact zeros inside its observed span are genuine months without an identified policy change, not missing data. The 48 leading `NA` values form one contiguous block, so the instrument aligns with a single sub-sample of the VAR.

Obtained from the replication package of Braun and Bruggemann (2023) (`https://github.com/r-a-braun/SVAR-IVSR`), file `dataset_ACR.mat`, which carries the Arias, Caldara and Rubio-Ramirez dataset. Note that Braun and Bruggemann identify the model with zero restrictions and sign restrictions on both B and $A_0 = B^{-1}$ in a proxy-augmented Bayesian setup; `fSignRestr` implements the different scheme of Cesa-Bianchi and Sokol (2022), so this object supports their data, not a replication of their results.

References

Arias, J. E., Caldara, D., & Rubio-Ramirez, J. F. (2019). The systematic component of monetary policy in SVARs: An agnostic identification procedure. *Journal of Monetary Economics*, 101, 1-13. doi:10.1016/j.jmoneco.2018.07.011

Braun, R., & Bruggemann, R. (2023). Identification of SVAR Models by Combining Sign Restrictions With External Instruments. *Journal of Business & Economic Statistics*, 41(4), 1077-1089. doi:10.1080/07350015.2022.2104857

Romer, C. D., & Romer, D. H. (2004). A New Measure of Monetary Shocks: Derivation and Implications. *American Economic Review*, 94(4), 1055-1084. doi:10.1257/0002828042002651

See Also

fSignRestr, Uhlig2005, ADRR2018

Examples

```
data("ACR2019")

y <- ACR2019 |>
  dplyr::mutate(dplyr::across(-c(Date, `Fed. Funds Rate`, RR),
                              \(x) 100 * log(x))) |>
  dplyr::select(-Date, -RR) |>
  as.matrix()

# Shock 1 is pinned by the instrument, so `sign` has k - 1 columns.
SIGN <- matrix(0, 6, 5)
SIGN[, 1] <- c(-1, -1, 0, 0, 0, -1)   # a conventional demand shock

fit <- fSignRestr(y, p = 12, c = 0, sign = SIGN, nsteps = 48,
                  ndraws = 100, sr_hor = 6,
                  instrument = list(Z = as.matrix(ACR2019$RR)), seed = 42,
                  n_threads = 2)
```

ADRR2018 *Antolin-Diaz & Rubio-Ramirez (2018) Data*

Description

Monthly US data used to replicate the narrative sign-restriction VAR of Antolin-Diaz and Rubio-Ramirez (2018), covering January 1965 - November 2007. The variables are those of Uhlig2005; the sample runs four years longer.

Usage

```
ADRR2018
```

Format

A data frame with 515 rows and 7 columns:

Date Month start date (`Date`), 1965-01 - 2007-11

Real GDP Log real GDP, interpolated to monthly frequency

GDP Deflator Log GDP deflator, interpolated to monthly frequency

Commodity Price Idx. Log commodity price index

Total Reserves Log total reserves

Non-Borrowed Reserves Log non-borrowed reserves

Fed. Funds Rate Federal funds rate, in percent

Details

The paper keeps Uhlig's sign restrictions and adds two narrative constraints tied to the Volcker announcement of October 1979: the monetary policy shock was positive that month (a Type 1, sign restriction), and it was the dominant contributor to the unexpected move in the funds rate (a Type 2, dominance restriction). Both are imposed by `fSignRestr` through its `narrative` argument.

Estimation follows footnote 8 of the paper and uses no constant, the data having been demeaned. Unlike `Uhlig2005`, the log-level series are used as stored, without rescaling by 100.

Data taken from the replication files of the VAR Toolbox of Ambrogio Cesa-Bianchi (https://github.com/ambropo/VAR-Toolbox).

References

Antolin-Diaz, J., & Rubio-Ramirez, J. F. (2018). Narrative Sign Restrictions for SVARs. *American Economic Review*, 108(10), 2802-2829. doi:10.1257/aer.20161852

See Also

`fSignRestr`, `Uhlig2005`

Examples

```
data("ADRR2018")

y <- ADRR2018 |> dplyr::select(-Date) |> as.matrix()

SIGN <- matrix(0, 6, 6)
SIGN[, 1] <- c(0, -1, -1, 0, -1, 1)

# Volcker, October 1979: positive MP shock that dominates the funds-rate
# forecast error (variable 6).
narr <- list(
  sign = list(shock = 1, period = as.Date("1979-10-01"), sign = 1),
  dom  = list(shock = 1, period = as.Date("1979-10-01"), var = 6)
)

fit <- fSignRestr(y, p = 12, c = 0, sign = SIGN, nsteps = 60, ndraws = 100,
```

```
sr_hor = 6, narrative = narr, dates = ADRR2018$Date,
seed = 42, n_threads = 2)
```

`BankingDeregulation` *US Banking Deregulation and the Labor Share*

Description

Annual state-level panel (46 US states, 1970–1996) on the private labor share and staggered banking deregulation, from the replication package of Dube, Girardi, Jordà and Taylor (LP-DiD). Treatment is absorbing.

Usage

```
BankingDeregulation
```

Format

A data frame with 1,242 rows and 12 columns:

id Integer state identifier

state State abbreviation

year Year, 1970–1996

lshare Private labor share (compensation / private GSP)

bank Inter-state banking deregulation in force (0/1)

branch Intra-state branching deregulation in force (0/1)

grgsp Growth rate of gross state product

corptax State corporate tax rate

unionmem Union membership share

popgrowth Population growth

lnunemp Log unemployment rate

lnhpi Log house price index

References

Dube, A., Girardi, D., Jordà, Ò., & Taylor, A. M. (2025). A Local Projections Approach to Difference-in-Differences. *Journal of Applied Econometrics*, 40(7), 741–758. doi:10.1002/jae.70000

Examples

```
data("BankingDeregulation")
m <- fLPDID(lshare ~ bank, data = BankingDeregulation,
            panel_id = c("id", "year"), treat = "bank",
            post = 9, pre = 9, n_threads = 2)
```

`BeaudryPortier2014` *Beaudry & Portier (2014) Data*

Description

Quarterly US data used to identify news shocks following Beaudry and Portier (2014) and Barsky and Sims (2011/2012). The news shock is identified as the shock with no contemporaneous effect on TFP that maximises the long-run forecast error variance of TFP.

Usage

```
BeaudryPortier2014
```

Format

A data frame with rows and 4 columns:

logTFP Log total factor productivity (Fernald index, base approximately 100)

logSP500 Log real S&P 500 stock price

logConsumption Log real private consumption

logHours Log total hours worked in the business sector

Details

Data sourced from the replication package of Beaudry and Portier (2014), available at doi:10.3886/E113916V1. The VAR is estimated with $p = 2$ lags and a constant. The news shock is ordered second (after TFP) and identified via the max-share criterion over a 40-quarter horizon.

References

Beaudry, P., & Portier, F. (2014). News-driven business cycles: Insights and challenges. *Journal of Economic Literature*, 52(4), 993–1074.

Barsky, R. B., & Sims, E. R. (2012). Information, animal spirits, and the meaning of innovations in consumer confidence. *American Economic Review*, 102(4), 1343–1377.

Examples

```
data("BeaudryPortier2014")
head(BeaudryPortier2014)
y <- BeaudryPortier2014 |>
  dplyr::select(logTFP, logSP500, logConsumption, logHours) |>
  as.matrix()
```

`Bloom2009` *Bloom (2009) Data*

Description

Monthly US data used to replicate the uncertainty shock VAR of Bloom (2009). Data sourced from https://nbloom.people.stanford.edu/research, covering July 1962 – June 2013.

Usage

```
Bloom2009
```

Format

A data frame with 612 rows and 9 columns:

Date Month start date (`Date`), 1962-07 – 2013-06

SP500 Log S&P 500 stock price index, in percent

FFR Federal funds rate, in percent

WAGE Log average hourly earnings (manufacturing), in percent

CPI Log Consumer Price Index, in percent

HOURS Average weekly hours in manufacturing

EMPL Log employment in manufacturing, in percent

INDPRO Log industrial production (manufacturing), in percent

UNCERT Uncertainty indicator: equals 1 in months coinciding with one of 17 identified uncertainty events (e.g. Cuban Missile Crisis, Black Monday), 0 otherwise.

Details

The uncertainty shock is identified via Cholesky ordering with `UNCERT` ordered first. The 17 event dates follow Bloom (2009) Table 1.

References

Bloom, N. (2009). The impact of uncertainty shocks. *Econometrica*, 77(3), 623–685.

Examples

```
data("Bloom2009")
head(Bloom2009)
y <- Bloom2009 |> dplyr::select(-Date) |> as.matrix()
```

CBS2022 *Cesa-Bianchi and Sokol (2022) Data*

Description

Monthly US macroeconomic and financial data with two external instruments for monetary policy and central bank information shocks.

Usage

```
CBS2022
```

Format

A data frame with 486 rows and 8 columns:

Date Month start date, July 1979 through December 2019

lnRGDP Log real GDP

CPI Log Consumer Price Index

i_1YR One-year interest rate, stored as a decimal rate

EBP Excess bond premium, in percentage points

BondYield Corporate bond yield, stored as a decimal rate

MPshockSign Sign-identified monetary policy shock instrument

CBIshockSign Sign-identified central bank information shock instrument

Details

The instruments are jointly observed from February 1990 through December 2016. Missing values are stored as `NA`; genuine zeros are retained.

See Also

`fSignRestr`

Examples

```
data("CBS2022")
head(CBS2022)
```

`DemocracyGrowth` *Democracy and Growth (ANRR / DDCG Panel)*

Description

Annual country-level panel on democracy and GDP per capita from Acemoglu, Naidu, Restrepo and Robinson (2019), as shipped in the Dube-Girardi- Jordà-Taylor LP-DiD replication package. Treatment (democracy) is non-absorbing: countries democratize and revert.

Usage

```
DemocracyGrowth
```

Format

A data frame with 5 columns:

id Integer country identifier (World Bank code)

country Country name

year Year

lgdp Log GDP per capita, multiplied by 100

dem Democracy indicator (0/1; `NA` where undefined)

References

Acemoglu, D., Naidu, S., Restrepo, P., & Robinson, J. A. (2019). Democracy Does Cause Growth. *Journal of Political Economy*, 127(1), 47–100.

Dube, A., Girardi, D., Jordà, Ò., & Taylor, A. M. (2025). A Local Projections Approach to Difference-in-Differences. *Journal of Applied Econometrics*, 40(7), 741–758. doi:10.1002/jae.70000

Examples

```
data("DemocracyGrowth")
m <- fLPDID(lgdp ~ dem + l(lgdp, 1:4), data = DemocracyGrowth,
            panel_id = c("id", "year"), treat = "dem",
            post = 30, pre = 20, nonabsorbing = TRUE, L = 20, ccc = 1,
            n_threads = 2)
```

fAICBIC *Determine Optimal Lag Order for VAR Model Using Information Criteria*

Description

Fits VAR models for lag orders 1 through pmax and returns the lag length that minimises each of three information criteria.

Usage

```
fAICBIC(y, pmax, c, exog = NULL)
```

Arguments

y	A T x N matrix of endogenous variables (T observations, N series).
pmax	An integer specifying the maximum VAR lag order to consider.
c	An integer (0 or 1) indicating whether to include a constant term (1 = include, 0 = exclude).
exog	An optional T x M matrix of exogenous variables. Default is NULL.

Details

For each candidate lag order p the residual covariance Sigma = (e'e)/T is computed and the three criteria are evaluated as:

AIC = log(det(Sigma)) + 2 * (p*N^2 + N + M) / T

BIC = log(det(Sigma)) + log(T) * (p*N^2 + N + M) / T

HQ = log(det(Sigma)) + 2 * log(log(T)) * (p*N^2 + N + M) / T

where M is the number of exogenous variables.

Value

A list with three integer elements: aic — optimal lag length by Akaike IC; bic — optimal lag length by Bayesian IC; hq — optimal lag length by Hannan-Quinn IC.

See Also

fVAR

Examples

```
set.seed(123)
y      <- matrix(rnorm(400), ncol = 2)
result <- fAICBIC(y, pmax = 4, c = 1)
result$aic
result$bic
```

`fBootstrapBQ` *Bootstrap Blanchard-Quah Long-Run Identified Impulse Response Functions*

Description

Bootstrap Blanchard-Quah Long-Run Identified Impulse Response Functions

Usage

```
fBootstrapBQ(
  y,
  var_result,
  nboot,
  horizon,
  conf = 90,
  conf2 = 68,
  bootscheme = "residual",
  cumulate = as.integer(c()),
  scaling = NULL,
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

`y`	T x N matrix of original endogenous variables.
`var_result`	List from `fVAR()`.
`nboot`	Number of bootstrap replications.
`horizon`	Maximum IRF horizon.
`conf`	Confidence level in percent (e.g. 68).
`conf2`	Secondary confidence level in percent (default 68).
`bootscheme`	`"residual"` or `"wild"`.
`cumulate`	Integer vector (1-based) of variable indices whose IRFs should be cumulated along the horizon. Typically used when the VAR is estimated in first differences and level responses are required.
`scaling`	Optional numeric vector of length 2. First element is the variable index (1-based) for normalisation; second is the shock size. Default NULL (no normalisation).
`exog`	Optional T x M matrix of exogenous variables. Must be provided if the original VAR was estimated with exogenous variables; must be omitted otherwise.
`n_threads`	Number of OpenMP threads. 0 = all cores minus one. Default 0.
`verbose`	Logical. If `TRUE`, report the number of threads used. Default `FALSE`, so the function is silent.

Details

Implements the residual or wild bootstrap for BQ long-run identified VARs. In each replication the full BQ identification is re-computed: $C_1 = \sum_h \Psi_h$, $D_1 = \text{chol}(C_1 \Sigma C_1')^\top$, $K = C_1^{-1} D_1$. Selected variables are then cumulated if the VAR is estimated in differences.

Value

List with elements:

bootbq N x N x (horizon+1) x nboot array of bootstrapped BQ IRFs

upper N x N x (horizon+1) upper confidence bands

lower N x N x (horizon+1) lower confidence bands

boot_beta N x (Np+c) x nboot array of bootstrapped coefficients

conf Band levels in percent, `c(conf, conf2)`

References

Blanchard, O. J., & Quah, D. (1989). The dynamic effects of aggregate demand and supply disturbances. *American Economic Review*, 79(4), 655–673.

Gali, J. (1999). Technology, employment, and the business cycle: Do technology shocks explain aggregate fluctuations? *American Economic Review*, 89(1), 249–271.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 12)
C1         <- apply(wold, c(1, 2), sum)
D1         <- t(chol(C1 %*% var_result$sigma %*% t(C1)))
point_irf  <- fBQIRF(wold, solve(C1, D1))

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapBQ(y, var_result, nboot = 50, horizon = 12,
                     conf = 68, bootscheme = "residual",
                     cumulate = c(1L, 2L), n_threads = 2)
fPlotIRF(point_irf, bands = boot, shock = 1)
```

`fBootstrapChol` *Bootstrap Cholesky Identified Impulse Response Functions*

Description

Bootstrap Cholesky Identified Impulse Response Functions

Usage

```
fBootstrapChol(
  y,
  var_result,
  nboot,
  horizon,
  conf = 90,
  conf2 = 68,
  bootscheme = "residual",
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N matrix of original endogenous variables.
var_result	List from fVAR().
nboot	Number of bootstrap replications.
horizon	Maximum IRF horizon.
conf	Outer confidence level in percent (e.g. 90).
conf2	Inner confidence level in percent (e.g. 68).
bootscheme	"residual" or "wild".
exog	Optional T x M matrix of exogenous variables. Must be provided if the original VAR was estimated with exogenous variables; must be omitted otherwise.
n_threads	Number of OpenMP threads. 0 = all cores minus one. Default 0.
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.

Value

List with elements:

bootchol N x N x (horizon+1) x nboot array of bootstrapped Cholesky IRFs

upper N x N x (horizon+1) outer upper confidence bands

lower N x N x (horizon+1) outer lower confidence bands

upper2 N x N x (horizon+1) inner upper confidence bands

lower2 N x N x (horizon+1) inner lower confidence bands

boot_beta N x (Np+c) x nboot array of bootstrapped coefficients

conf Band levels in percent, c(conf, conf2)

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapChol(y, var_result, nboot = 50, horizon = 12,
                       n_threads = 2)
dim(boot$upper)
```

`fBootstrapCholCorrected` *Bootstrap Bias-Corrected Cholesky Identified Impulse Response Functions*

Description

Bootstrap Bias-Corrected Cholesky Identified Impulse Response Functions

Usage

```
fBootstrapCholCorrected(
  y,
  var_result,
  nboot1,
  nboot2,
  horizon,
  conf = 90,
  conf2 = 68,
  bootscheme = "residual",
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

`y`	T x N matrix of original endogenous variables.
`var_result`	List from `fVAR()`.
`nboot1`	Number of first-pass replications for bias estimation.
`nboot2`	Number of second-pass replications for confidence bands.
`horizon`	Maximum IRF horizon.
`conf`	Outer confidence level in percent (e.g. 90).
`conf2`	Inner confidence level in percent (e.g. 68).

bootscheme	"residual" or "wild".
exog	Optional T x M matrix of exogenous variables. Must be provided if the original VAR was estimated with exogenous variables; must be omitted otherwise.
n_threads	Number of OpenMP threads. 0 = all cores minus one. Default 0.
verbose	Logical. If TRUE, report the number of threads used and the bias-correction summary. Default FALSE.

Value

List with elements:

bootchol N x N x (horizon+1) x nboot2 array of bootstrapped Cholesky IRFs

upper N x N x (horizon+1) outer upper confidence bands

lower N x N x (horizon+1) outer lower confidence bands

upper2 N x N x (horizon+1) inner upper confidence bands

lower2 N x N x (horizon+1) inner lower confidence bands

boot_beta N x (Np+c) x nboot2 array of bootstrapped coefficients

conf Band levels in percent, c(conf, conf2)

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapCholCorrected(y, var_result, nboot1 = 50, nboot2 = 50,
                                horizon = 12, n_threads = 2)
dim(boot$upper)
```

---

fBootstrapHDIV *Bootstrap Confidence Bands for the IV Historical Decomposition*

Description

Computes MBB bootstrap uncertainty bands for the companion-form historical decomposition of an IV-identified structural shock. In each replication the VAR is re-estimated, the IV identification is re-run, and the companion-form HD recursion is applied to the bootstrap draw. Bands are recentered around the point estimate following Kaenzig (2021).

Usage

```
fBootstrapHDIV(
  y,
  var_result,
  Z,
  s,
  nboot,
  blocksize,
  adjustZ,
  adjustu,
  policyvar = 1L,
  conf = 90,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N matrix of original endogenous variables.
var_result	List from fVAR.
Z	Instrument matrix aligned to the proxy sample.
s	N x 1 unit-normalised point-estimate structural impact vector.
nboot	Number of bootstrap replications.
blocksize	MBB block size.
adjustZ	Integer vector c(start, end) selecting the proxy-sample rows of Z (1-based).
adjustu	Integer vector c(start, end) selecting the proxy-sample rows of the residuals (1-based).
policyvar	Integer (1-based) index of the IV policy variable. Default 1.
conf	Confidence level in percent (e.g. 90 for 90% CI). Default 90.
n_threads	OpenMP threads. 0 = all cores minus one. Default 0.
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.

Value

A list with elements:

HDshock	(T-p) x N point-estimate HD matrix.
upper	(T-p) x N upper confidence bands.
lower	(T-p) x N lower confidence bands.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# An instrument correlated with the first reduced-form residual
u    <- var_result$residuals
Z    <- matrix(u[, 1] + rnorm(nrow(u), sd = 0.5), ncol = 1)
rows <- c(1, nrow(u))
s    <- c(1, sum(Z * u[, 2]) / sum(Z * u[, 1]))

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapHDIV(y, var_result, Z, s = s, nboot = 50, blocksize = 10,
                       adjustZ = rows, adjustu = rows, policyvar = 1,
                       n_threads = 2)
names(boot)
```

`fBootstrapHetero` *MBB Bootstrap for Heteroskedasticity-Based VAR Identification*

Description

Moving-block bootstrap confidence bands for structural IRFs identified via proxy IV restricted to OPEC announcement months (Kaenzig 2021, treatment group only). Residuals and proxy are re-sampled jointly.

Usage

```
fBootstrapHetero(
  y,
  var_result,
  Z,
  indsR1,
  adjustu,
  nboot = 1000L,
  blocksize = 0L,
  hor = 48L,
  nvar = 1L,
  scale = 10,
  conf = 90,
  conf2 = 68,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N data matrix.
var_result	List returned by fVAR().
Z	Proxy matrix (T1 x 1), aligned to adjustu.
indsR1	Integer 0/1 vector of length T1 (1 = treatment month).
adjustu	Integer vector c(start, end) (1-based).
nboot	Bootstrap replications.
blocksize	MBB block size (0 = automatic 5.03*T^0.25).
hor	IRF horizon.
nvar	1-based normalization variable.
scale	Shock size.
conf	Primary confidence level (e.g. 90).
conf2	Secondary confidence level (e.g. 68).
n_threads	OpenMP threads (0 = all cores minus one).
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.

Value

A list with upper, lower, upper2, lower2, point (each N x hor+1) and conf, the band levels c(conf, conf2). point is the plug-in estimate on the original sample; the bootstrap supplies the bands only.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

u      <- var_result$residuals
Z      <- matrix(u[, 1] + rnorm(nrow(u), sd = 0.5), ncol = 1)
rows   <- c(1, nrow(u))
indsR1 <- as.integer(abs(Z[, 1]) > stats::median(abs(Z[, 1])))

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapHetero(y, var_result, Z, indsR1, rows, nboot = 50,
                         blocksize = 10, hor = 12, nvar = 1, scale = 1,
                         n_threads = 2)
dim(boot$upper)
```

`fBootstrapIVInvertible` *Bootstrap IRFs for the Invertible Case (Wild Bootstrap)*

Description

Implements the wild bootstrap with Rademacher signs for standard SVAR-IV. The same Rademacher vector is applied to both the VAR residuals and the raw instrument; at each draw the structural impact vector is re-estimated by IV regression.

Usage

```
fBootstrapIVInvertible(
  y,
  instr,
  var_result,
  nboot,
  p,
  c,
  hor,
  cumu,
  conf = 90,
  conf2 = 68,
  n_threads = 0L
)
```

Arguments

`y`	T x N matrix of endogenous variables.
`instr`	Length-T raw instrument vector (before cleaning).
`var_result`	List from `fVAR`.
`nboot`	Number of bootstrap replications.
`p`	VAR lag order.
`c`	Integer (0/1): include intercept.
`hor`	Impulse-response horizon.
`cumu`	1-indexed integer vector of variable positions to cumulate.
`conf`	Primary confidence level (e.g. 90). Default 90.
`conf2`	Secondary confidence level (e.g. 68). Default 68.
`n_threads`	OpenMP threads. 0 (default) uses all cores minus one.

Value

List with N x (hor+1) matrices: upper, lower (at `conf`), upper2, lower2 (at `conf2`), median.

See Also

fBootstrapIVRecover, fGetBands

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 1, c = 1)

# A length-T instrument (the first observation is lost to the lag)
u     <- var_result$residuals
instr <- c(0, u[, 1] + rnorm(nrow(u), sd = 0.5))

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapIVInvertible(y, instr, var_result, nboot = 50, p = 1,
                               c = 1, hor = 12, cumu = integer(),
                               n_threads = 2)
dim(boot$median)
```

---

fBootstrapIVMBB *Bootstrap IV Impulse Response Functions using Moving Block Bootstrap*

Description

Computes bootstrap confidence bands for instrumental variable identified impulse response functions using the moving block bootstrap to preserve temporal dependence in residuals and instruments.

Usage

```
fBootstrapIVMBB(
  y,
  var_result,
  Z,
  nboot,
  blocksize,
  adjustZ,
  adjustu,
  policyvar,
  horizon,
  conf = 90,
  conf2 = 68,
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N matrix of endogenous variables
var_result	List output from fVAR containing beta, residuals, p, c
Z	Matrix of instrumental variables with K columns; rows adjustZ are used. Zeros are treated as months without an instrument observation.
nboot	Integer number of bootstrap replications
blocksize	Integer block size for moving block bootstrap
adjustZ	Integer vector of length 2: start and end row indices (1-based, inclusive) for instrument alignment.
adjustu	Integer vector of length 2: start and end row indices (1-based, inclusive) for residual alignment.
policyvar	Integer index (1-based) of the policy variable.
horizon	Integer maximum impulse response horizon.
conf	Confidence level in percent (default 90).
conf2	Secondary confidence level in percent (default 68).
exog	Optional matrix of exogenous variables (T x M). Default NULL.
n_threads	Integer number of threads for parallel computation. Default is 0 (uses all available cores minus one, with a minimum of one). Set to 1 for single-threaded execution. If OpenMP is not available, automatically falls back to single-threaded execution.
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.

Details

This function implements the moving block bootstrap for IV-identified SVARs, following the replication code of Kaenzig (2021). The first stage regresses the policy variable residual on the instrument(s) and a constant, and the second stage regresses the other residuals on the fitted values and a constant. The normalization sets the policy variable shock to have unit impact on itself.

Blocks are drawn from all VAR residuals jointly with the instrument, which is set to zero outside the rows selected by adjustu; the identification step uses those rows only. Non-zero instrument values are centred so that they keep their sample mean. A resample with fewer than 15 non-zero instrument values in the identification window is drawn again.

The function uses OpenMP for parallel computation when available, speeding up bootstrap iterations. Each bootstrap replication and percentile computation is independent and can be parallelized. If OpenMP is not available, the function automatically falls back to single-threaded execution.

Value

A list containing:

- upper: N x (horizon+1) matrix of upper confidence bands
- lower: N x (horizon+1) matrix of lower confidence bands

- upper2: N x (horizon+1) matrix of secondary upper bands
- lower2: N x (horizon+1) matrix of secondary lower bands

  All four bands are bootstrap percentiles recentred on the point estimate, percentile minus bootstrap median plus `point`, so `point` always lies inside them.
- point: N x (horizon+1) matrix of impulse responses on the original sample, computed as the Wold coefficients times the proxy-SVAR impact vector, exactly as in `fHeteroIRF()` and `fMSW()`. The bootstrap supplies the bands only, never the point estimate.
- conf: band levels in percent, `c(conf, conf2)`

Examples

```
set.seed(123)
y <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# An instrument correlated with the first reduced-form residual
u <- var_result$residuals
Z <- matrix(u[, 1] + rnorm(nrow(u), sd = 0.5), ncol = 1)
rows <- c(1, nrow(u))

# Few draws here to keep the example quick; use 1000 or more in practice.
result <- fBootstrapIVMBB(y, var_result, Z, nboot = 50, blocksize = 10,
                          adjustZ = rows, adjustu = rows,
                          policyvar = 1, horizon = 12, conf = 68,
                          n_threads = 2)
fPlotIRF(result, varnames = c("y1", "y2"))
```

`fBootstrapIVRecover` *Bootstrap IRFs for the Recoverable Non-Invertible Case*

Description

Implements the iid joint-resampling bootstrap for generalized SVAR-IV when the shock is non-invertible but recoverable (Forni, Gambetti & Ricco 2022). At each draw the VAR residuals and the instrument noise are resampled jointly, the instrument is regenerated from the DGP $z(t) = \delta(F)e(t) + v(t)$, and $\Psi(L)$ is re-estimated.

Usage

```
fBootstrapIVRecover(
  y,
  instr,
  var_result,
  noise,
  delta,
  nboot,
```

```
  p,
  c,
  r,
  hor,
  cumu,
  conf = 90,
  conf2 = 68,
  n_threads = 0L
)
```

Arguments

y	T x N matrix of endogenous variables.
instr	Length-T raw instrument vector (before cleaning).
var_result	List from fVAR.
noise	Length-(T-p-r) noise residuals v_t from the invertibility regression.
delta	Length-$N(r+1)+1$ coefficient vector (intercept + current and r future VAR residual coefficients) from the invertibility regression.
nboot	Number of bootstrap replications.
p	VAR lag order.
c	Integer (0/1): include intercept.
r	Number of future residuals used in the invertibility test.
hor	Impulse-response horizon.
cumu	1-indexed integer vector of variable positions to cumulate.
conf	Primary confidence level (e.g. 90). Default 90.
conf2	Secondary confidence level (e.g. 68). Default 68.
n_threads	OpenMP threads. 0 (default) uses all cores minus one.

Value

List with N x (hor+1) matrices: upper, lower (at conf), upper2, lower2 (at conf2), median.

See Also

fBootstrapIVInvertible, fGetBands

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 1, c = 1)

# Instrument built from current and one-period-ahead residuals (r = 1),
# with the first-stage coefficients delta and noise v_t
r     <- 1
u     <- var_result$residuals
```

```
n_eff <- nrow(u) - r
delta <- c(0, 1, 0.2, 0.45, 0.1)
noise <- rnorm(n_eff, sd = 0.5)
u_f   <- cbind(u[seq_len(n_eff), ], u[2:(n_eff + 1), ])
instr <- numeric(nrow(y))
instr[2:(nrow(y) - r)] <- cbind(1, u_f) %*% delta + noise

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapIVRecover(y, instr, var_result, noise, delta, nboot = 50,
                            p = 1, c = 1, r = r, hor = 12,
                            cumu = integer(), n_threads = 2)
dim(boot$median)
```

---

fBootstrapMax *Bootstrap Long-Run Maximum-Response Impulse Responses*

Description

Computes residual-bootstrap confidence bands for the identification used by fMaxIRF. The corrected variant first estimates coefficient bias, shrinking the correction if needed for VAR stability, and then bootstraps the bias-corrected VAR.

Usage

```
fBootstrapMax(
  y,
  var_result,
  nboot,
  horizon,
  var_idx,
  conf = 90,
  conf2 = 68,
  cumulate = as.integer(c()),
  scaling = NULL,
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)

fBootstrapMaxCorrected(
  y,
  var_result,
  nboot1,
  nboot2,
  horizon,
  var_idx,
  conf = 90,
```

```
  conf2 = 68,
  cumulate = as.integer(c()),
  scaling = NULL,
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N matrix of endogenous variables.
var_result	Fitted VAR returned by fVAR.
nboot	Number of bootstrap replications.
horizon	Maximum response horizon; horizon zero is included.
var_idx	Index (1-based) of the variable whose last-horizon response is maximised.
conf	Confidence level in percent (default 90).
conf2	Secondary confidence level in percent (default 68).
cumulate	Integer vector of variable indices (1-based) whose responses should be cumulated. Defaults to no cumulation.
scaling	Optional length-two numeric vector. Responses are divided by the impact response of variable scaling[1] multiplied by scaling[2], when that product is nonzero. Default NULL.
exog	T x M matrix of exogenous variables, required when the original VAR included them; otherwise NULL.
n_threads	Number of OpenMP threads. Zero uses all available cores minus one, with a minimum of one. Without OpenMP, uses one.
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.
nboot1	Number of first-pass replications used to estimate bias.
nboot2	Number of second-pass replications used for the bands.

Value

A list with bootmax, an N x (horizon+1) x nboot array; upper, lower, upper2, and lower2, each an N x (horizon+1) matrix; and boot_beta, an N x K x nboot coefficient array, where K is the number of regressors; and conf, the band levels c(conf, conf2). The corrected variant stores nboot2 draws.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapMax(y, var_result, nboot = 50, horizon = 12,
```

```
                        var_idx = 1, n_threads = 2)
dim(boot$upper)
```

---

fBootstrapUhlig *Bootstrap Uhlig Maximum-Share Impulse Responses*

Description

Computes residual-bootstrap confidence bands for the identification used by `fUhligIRF`. The corrected variant first estimates coefficient bias, shrinking the correction if needed for VAR stability, and then bootstraps the bias-corrected VAR.

Usage

```
fBootstrapUhlig(
  y,
  var_result,
  nboot,
  horizon,
  idx,
  conf = 90,
  conf2 = 68,
  cumulate = as.integer(c()),
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)

fBootstrapUhligCorrected(
  y,
  var_result,
  nboot1,
  nboot2,
  horizon,
  idx,
  conf = 90,
  conf2 = 68,
  cumulate = as.integer(c()),
  exog = NULL,
  n_threads = 0L,
  verbose = FALSE
)
```

Arguments

y	T x N matrix of endogenous variables.

var_result	Fitted VAR returned by fVAR.
nboot	Number of bootstrap replications.
horizon	Maximum response horizon; horizon zero is included.
idx	Index (1-based) of the variable whose forecast error variance contribution is maximised.
conf	Confidence level in percent (default 90).
conf2	Secondary confidence level in percent (default 68).
cumulate	Integer vector of variable indices (1-based) whose responses should be cumulated. Defaults to no cumulation.
exog	T x M matrix of exogenous variables, required when the original VAR included them; otherwise NULL.
n_threads	Number of OpenMP threads. Zero uses all available cores minus one, with a minimum of one. Without OpenMP, uses one.
verbose	Logical. If TRUE, report the number of threads used. Default FALSE, so the function is silent.
nboot1	Number of first-pass replications used to estimate bias.
nboot2	Number of second-pass replications used for the bands.

Value

A list with bootuhlig, an N x (horizon+1) x nboot array; upper, lower, upper2, and lower2, each an N x (horizon+1) matrix; and boot_beta, an N x K x nboot coefficient array, where K is the number of regressors; and conf, the band levels c(conf, conf2). The corrected variant stores nboot2 draws.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# Few draws here to keep the example quick; use 1000 or more in practice.
boot <- fBootstrapUhlig(y, var_result, nboot = 50, horizon = 12,
                        idx = 1, n_threads = 2)
dim(boot$upper)
```

---

fBootstrapVAR *Bootstrap VAR Model*

Description

Generates a bootstrap pseudo-sample from a fitted VAR model using either residual (iid) resampling or a wild (Rademacher) bootstrap scheme.

Usage

```
fBootstrapVAR(y, fVAR_result, bootscheme = "residual")
```

Arguments

y	T x N matrix of original endogenous variables.
fVAR_result	List from fVAR containing beta, residuals, p, and c.
bootscheme	String indicating the bootstrap method: "residual" for iid resampling of residuals, or "wild" for a Rademacher wild bootstrap.

Details

The recursion is

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + e_t^*$$

where e_t^* is either a randomly resampled residual (residual scheme) or the t-th residual multiplied by a Rademacher sign (wild scheme).

Value

A list with two elements: ynext — T x N bootstrapped data matrix (first p rows copied from y); rademacher — T_eff-length vector of Rademacher signs used in the wild scheme (all zeros for residual scheme).

See Also

fBootstrapChol, fVAR

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# One artificial sample drawn from the estimated VAR
y_star <- fBootstrapVAR(y, var_result, bootscheme = "wild")
dim(y_star$ynext)
```

---

fBQIRF *Compute Blanchard-Quah (BQ) Impulse Response Functions*

Description

Compute Blanchard-Quah (BQ) Impulse Response Functions

Usage

```
fBQIRF(wold, K, scaling = NULL)
```

Arguments

`wold`	Wold representation cube (N x N x horizon+1), where `wold[,,h]` contains the Wold IRF at horizon h
`K`	N x N contemporaneous impact matrix from BQ identification, obtained by solving C(1) K = D, where D is the lower Cholesky factor of the long-run covariance matrix.
`scaling`	Optional numeric vector of length 2. The first element specifies the variable index (1-based) used for normalisation, and the second element specifies the shock size. When omitted no normalisation is applied.

Details

Computes structural impulse response functions under the Blanchard-Quah (1989) long-run identification scheme. For each horizon h the structural IRF is computed as:

$$IRF_h = \Psi_h \cdot K$$

where Ψ_h is the Wold representation at horizon h and K is the contemporaneous impact matrix implied by the long-run restrictions.

When `scaling` is supplied the entire cube is divided by $IRF_0(\text{scaling}[1], \text{scaling}[1])$ / scaling[2], normalising the impact response of the selected variable to `scaling[2]`.

Value

A cube (N x N x horizon+1) of long-run identified impulse response functions.

References

Blanchard, O. J., & Quah, D. (1989). The dynamic effects of aggregate demand and supply disturbances. *American Economic Review*, 79(4), 655–673.

Examples

```
set.seed(123)
y    <- matrix(rnorm(400), ncol = 2)
VAR  <- fVAR(y, p = 2, c = 1)
wold <- fWoldIRF(VAR, horizon = 20)

# Map the long-run Cholesky factor to contemporaneous impacts
C1 <- apply(wold, c(1, 2), sum)
D1 <- t(chol(C1 %*% VAR$sigma %*% t(C1)))
K  <- solve(C1, D1)
bqirf <- fBQIRF(wold, K)
bqirf[, 1, 1]

# With normalisation: unit shock to variable 1
bqirf_norm <- fBQIRF(wold, K, scaling = c(1, 1))
```

`fCheckRestrictions` *Check Sign Restrictions for One Shock*

Description

For a candidate orthonormal rotation Q applied to the Cholesky IRFs, tests whether the impulse responses for a given shock satisfy the imposed sign restrictions at all specified horizons.

Usage

```
fCheckRestrictions(irf, shock, restr, hor_vec)
```

Arguments

`irf`	N x N x H cube of structural IRFs (e.g. `cholirf %*% Q` slice-by-slice).
`shock`	Integer (1-indexed). Which column of `irf` to examine.
`restr`	Numeric row vector of length N. Sign restrictions per variable: `+1` (must be positive), `-1` (must be negative), `0` (unrestricted).
`hor_vec`	Integer vector of horizons to check (1-indexed, where 1 = impact).

Details

Implements the restriction-checking step of the algorithm in Rubio-Ramirez, Waggoner & Zha (2010). Restrictions with `restr[i] = 0` are treated as "generic" (no restriction) and are ignored when counting satisfied constraints.

Value

Integer scalar:

- `+1` all restrictions pass at every checked horizon
- `0` all restrictions fail at every horizon; flipping the sign of this shock column would fix them all
- `-1` mixed result; this draw of Q must be discarded

See Also

`fGenerateQ`, `fSignRestr`

Examples

```
set.seed(123)
y        <- matrix(rnorm(600), ncol = 3)
VAR      <- fVAR(y, p = 2, c = 1)
wold     <- fWoldIRF(VAR, horizon = 20)
cholirf  <- fCholeskyIRF(wold, t(chol(VAR$sigma)))
```

```
Q   <- fGenerateQ(3)
irf <- array(NA_real_, dim = dim(cholirf))
for (h in seq_len(dim(irf)[3])) irf[, , h] <- cholirf[, , h] %*% Q

# Shock 3 should lower variables 1 and 2 and raise variable 3 on impact
fCheckRestrictions(irf, shock = 3, restr = c(-1, -1, 1), hor_vec = 1L)
```

`fCheckStability` *Check VAR Stability*

Description

Check VAR Stability

Usage

```
fCheckStability(var_model)
```

Arguments

`var_model`	A list containing VAR estimation results with elements: • beta: Coefficient matrix • c: Integer indicator for intercept (1 if intercept, 0 otherwise) • p: Integer lag order

Value

Called for its side-effect of printing to the console. Returns `NULL` invisibly.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
fCheckStability(var_result)
```

`fCholeskyIRF` *Compute Cholesky Impulse Response Functions*

Description

Compute Cholesky Impulse Response Functions

Usage

```
fCholeskyIRF(wold, S)
```

Arguments

`wold`	Wold representation cube (N x N x horizon)
`S`	Cholesky factor matrix (N x N), lower triangular

Details

This function computes structural impulse response functions using the Cholesky decomposition identification scheme. For each horizon h, the structural IRF is computed as:

$$IRF_h = \Psi_h \cdot S$$

where Ψ_h is the Wold representation at horizon h and S is the Cholesky factor (lower triangular) of the covariance matrix of reduced-form residuals.

The Cholesky identification imposes a recursive structure on the contemporaneous relationships between variables, with the ordering of variables determining the causal structure.

Value

A cube containing: Cholesky structural impulse response functions (N x N x horizon)

Examples

```
set.seed(123)
y     <- matrix(rnorm(400), ncol = 2)
VAR   <- fVAR(y, p = 2, c = 1)
wold  <- fWoldIRF(VAR, horizon = 10)
S     <- t(chol(VAR$sigma))        # lower-triangular factor
chol_irf <- fCholeskyIRF(wold, S)
chol_irf[, 1, 1]                   # impact responses to shock 1
```

`fCompanionMatrix` *Compute VAR Companion Matrix*

Description

Compute VAR Companion Matrix

Usage

```
fCompanionMatrix(beta, c, p)
```

Arguments

`beta`	Coefficient matrix. Dimensions: (Np+c+M) x N, where N is the number of variables, p is the lag order, c is 1 if intercept included (0 otherwise), and M is the number of exogenous variables
`c`	Integer indicator for intercept (1 if intercept included, 0 otherwise)
`p`	Integer lag order of the VAR model

Details

This function constructs the companion form matrix of a VAR(p) model. The companion matrix representation allows the VAR(p) to be written as a VAR(1) system, which is useful for various computations including impulse response functions and forecasting.

When exogenous variables are present, they are excluded from the companion matrix as they do not have a dynamic feedback structure in the VAR system.

The companion matrix has the structure:

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I_N & 0 & \cdots & 0 & 0 \\ 0 & I_N & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_N & 0 \end{bmatrix}$$

Value

A list containing:

- comp: Companion matrix (Np x Np)
- N: Number of variables in the VAR system

Examples

```
set.seed(123)
# VAR(2) with 3 variables, no intercept
beta <- matrix(rnorm(18), 6, 3)
comp <- fCompanionMatrix(beta, c = 0, p = 2)
dim(comp$comp)

# VAR(2) with 3 variables, an intercept and 2 exogenous regressors
beta_int <- matrix(rnorm(27), 9, 3)
fCompanionMatrix(beta_int, c = 1, p = 2)$comp[1:3, 1:3]
```

`fFEVDChol` *Forecast Error Variance Decomposition for Cholesky-Identified VARs*

Description

Computes the forecast error variance decomposition (FEVD) showing the proportion of forecast error variance of each variable attributable to each structural shock identified via Cholesky decomposition.

Usage

```
fFEVDChol(chol_irf, shock = 0L)
```

Arguments

`chol_irf`	N x N x (horizon+1) cube of Cholesky impulse response functions from `fCholeskyIRF`. Each slice chol_irf[,,h] contains the responses at horizon h. The horizon is automatically inferred from the cube dimensions.
`shock`	Integer (1-indexed). If provided, returns only the N x horizon matrix for that specific shock (matching MATLAB `variance_decomp(irf, shock)`). If 0 (default), returns the full N x N x horizon cube for all shocks.

Details

The forecast error variance decomposition measures the proportion of the h-step ahead forecast error variance of variable i that is attributable to each structural shock j. For Cholesky-identified VARs:

$$FEVD(i,j,h) = \frac{\sum_{s=0}^{h} [\Psi_s P]_{ij}^2}{\sum_{k=1}^{N} \sum_{s=0}^{h} [\Psi_s P]_{ik}^2}$$

Value

A list containing:

fevd	If shock = 0: N x N x horizon cube where fevd[i,j,h] is the share of forecast error variance of variable i at horizon h explained by shock j. If shock > 0: N x horizon matrix for the specified shock.

See Also

fCholeskyIRF, fWoldIRF, fPlotVarDec, fFEVDIV

Examples

```
set.seed(123)
y        <- matrix(rnorm(400), ncol = 2)
VAR      <- fVAR(y, p = 2, c = 1)
wold     <- fWoldIRF(VAR, horizon = 20)
chol_irf <- fCholeskyIRF(wold, t(chol(VAR$sigma)))

# All shocks (a 3-D array), then one shock (an N x horizon matrix)
vardec   <- fFEVDChol(chol_irf)$fevd
vardec_1 <- fFEVDChol(chol_irf, shock = 1)$fevd
colSums(vardec[1, , ])   # shares of variable 1 sum to one at every horizon
```

---

fFEVDIV *Forecast Error Variance Decomposition for IV-Identified Shocks*

Description

Computes the forecast error variance decomposition (FEVD) for instrumental variable identified structural shocks. The function scales shocks to unit variance and calculates the contribution of a specific shock to the forecast error variance of each variable over time.

Usage

```
fFEVDIV(s, S, wold, N, hor, sigma, u, T1, p)
```

Arguments

s	N x 1 vector of structural impact coefficients
S	N x N lower triangular Cholesky factor of the residual covariance matrix
wold	N x N x (hor+1) cube of Wold impulse response functions
N	Integer number of variables in the VAR system
hor	Integer maximum forecast horizon
sigma	N x N residual covariance matrix
u	(T-p) x N matrix of VAR residuals
T1	Integer effective sample size (T-p)
p	Integer lag order of the VAR model

Details

The FEVD measures the proportion of the h-step ahead forecast error variance of each variable that is attributable to the identified structural shock.

The function performs the following steps: 1. Normalizes the structural shock to have unit variance 2. Computes unit variance impulse responses 3. Recovers the unit variance shock series 4. Computes total forecast error variance at each horizon 5. Computes variance due to identified shock 6. Calculates FEVD as the ratio

The FEVD values are proportions between 0 and 1, where higher values indicate that the identified shock explains a larger share of the forecast error variance.

Value

A list containing: - scaler: N x 1 vector of scaling factors for unit variance normalization - ivirf_scaled: N x (hor+1) matrix of unit variance impulse responses - oil_news_unit_var: 1 x T1 vector of unit variance shock series - check_uv: Scalar variance check (should be close to 1) - denom: N x (hor+1) matrix of total forecast error variances - fFEVDIV: N x (hor+1) matrix of FEVD shares (between 0 and 1)

See Also

`fWoldIRF` for computing Wold impulse responses, `fVAR` for VAR estimation

Examples

```
set.seed(123)
y    <- matrix(rnorm(600), ncol = 3)
VAR  <- fVAR(y, p = 2, c = 1)
wold <- fWoldIRF(VAR, horizon = 20)

# An impact vector from an external instrument, normalised on variable 1
s     <- c(1, 0.5, 0.3)
sigma <- VAR$sigma
result <- fFEVDIV(s, t(chol(sigma)), wold, N = 3, hor = 20,
                  sigma = sigma, u = VAR$residuals,
                  T1 = nrow(VAR$residuals), p = 2)
result$fFEVDIV[1, 1:5]
```

---

fGenerateQ *Draw a Random Orthonormal Matrix (RZW 2010)*

Description

Generates an N x N orthonormal matrix Q (QQ' = Q'Q = I) by applying QR decomposition to a matrix of i.i.d. standard normals and normalising the diagonal of R to be positive, yielding a unique decomposition.

Usage

```
fGenerateQ(N)
```

Arguments

N	Integer. Dimension of the square matrix.

Details

Draw an N x N matrix M of independent standard normals, compute the QR decomposition M = QR, then set Q[,i] = -Q[,i] whenever R[i,i] < 0. The resulting Q is uniformly distributed on the Stiefel manifold (Haar measure), as required for sign-restriction identification.

Value

An N x N orthonormal matrix.

References

Rubio-Ramirez, J. F., Waggoner, D. F., & Zha, T. (2010). Structural vector autoregressions: Theory of identification and algorithms for inference. *Review of Economic Studies*, 77(2), 665–696.

Examples

```
set.seed(123)
Q <- fGenerateQ(3)
round(t(Q) %*% Q, 10)  # the identity matrix
```

---

fGenerateVARData *Generate VAR Data*

Description

Recursively generates VAR data from initial values, coefficients, and residuals. Optimised for maximum performance and memory efficiency.

Usage

```
fGenerateVARData(y, p, c, beta, residuals)
```

Arguments

y	T x N matrix of initial/historical endogenous variables.
p	Integer lag order of the VAR model.
c	Integer indicator for intercept (1 if included, 0 otherwise).
beta	Coefficient matrix: (Np+c) x N if c = 1, or Np x N if c = 0.
residuals	(T-p) x N matrix of residuals/shocks to add.

Details

The recursion follows:

$$y_t = c + A_1 y_{t-1} + \cdots + A_p y_{t-p} + e_t$$

Value

T x N matrix of generated VAR data. The first p rows are copied from y; the remaining T-p rows are generated recursively.

Examples

```
set.seed(123)
y_init    <- matrix(rnorm(200), ncol = 2)
beta      <- matrix(rnorm(10), 5, 2)
residuals <- matrix(rnorm(196), 98, 2)
y_new     <- fGenerateVARData(y_init, p = 2, c = 1, beta, residuals)
dim(y_new)
```

---

fGetBands *Compute Confidence Bands from Bootstrap IRF Cube*

Description

Computes upper, lower, and median bands from a 3-D bootstrap IRF array along the third (bootstrap) dimension. Percentiles follow R's `quantile(type = 7)`.

Usage

```
fGetBands(bootirf, conf = 90)
```

Arguments

`bootirf`	N x (hor+1) x nboot cube of bootstrapped IRFs.
`conf`	Confidence level (e.g. 68 for 68% band). Upper and lower quantiles are $(50 + conf/2)\%$ and $(50 - conf/2)\%$. Default 90.

Value

List with three N x (hor+1) matrices: upper, lower, median.

Examples

```
set.seed(123)
# 200 bootstrap draws of a 2-variable IRF over horizons 0 to 12
bootirf <- array(rnorm(2 * 13 * 200), c(2, 13, 200))
bands   <- fGetBands(bootirf, conf = 68)
bands$median[, 1:3]
```

fGetShock *Recover Structural Shock Series (Stock-Watson 2018)*

Description

Recover Structural Shock Series (Stock-Watson 2018)

Usage

```
fGetShock(residuals, sigma_full, s, shockSize = 1, normalize = "unit")
```

Arguments

residuals	A T x N matrix of VAR residuals.
sigma_full	An N x N residual covariance matrix (full sample).
s	An N x 1 structural impact vector.
shockSize	Scalar normalisation applied in "unit" mode (default 1). Set to 10 to express shocks in "10% impact" units, consistent with IRF scaling. Ignored when normalize = "sd".
normalize	Character string selecting the normalisation mode.

- "unit" (default): divides by $s'\Sigma^{-1}s$ and scales by shockSize. Use when s is unit-normalised (first element = 1).
- "sd": returns $U\Sigma^{-1}s$ directly. Use when s is already sd-normalised.

Value

A T x 1 numeric vector of recovered structural shocks.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# Impact vector normalised so the shock moves variable 1 by one unit
S     <- t(chol(var_result$sigma))
s     <- S[, 1] / S[1, 1]
shock <- fGetShock(var_result$residuals, var_result$sigma, s)
head(shock)
```

fHDIV *Companion-Form Historical Decomposition for IV-Identified Shock*

Description

Computes the companion-form historical decomposition attributing each observation of all VAR variables to the IV-identified structural shock, following Stock and Watson (2018).

Usage

```
fHDIV(residuals, sigma, s, beta, c, p)
```

Arguments

residuals	(T-p) x N matrix of VAR residuals from fVAR.
sigma	N x N residual covariance matrix (e.g. var_result$sigma).
s	N x 1 unit-normalised structural impact vector (first element = 1).
beta	(Np+c) x N VAR coefficient matrix from fVAR.
c	Integer intercept indicator (1 = include, 0 = exclude).
p	Integer VAR lag order.

Value

A list with element:

HDshock	(T-p) x N matrix. Column j is the contribution of the IV shock to variable j at each point in time.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

S  <- t(chol(var_result$sigma))
s  <- S[, 1] / S[1, 1]
hd <- fHDIV(var_result$residuals, var_result$sigma, s,
            var_result$beta, c = 1, p = 2)
dim(hd$HDshock)
```

fHDShock *Historical Decomposition of a Structural VAR*

Description

Splits each observed series into the contribution of every structural shock plus the deterministic and initial-condition terms. The components sum back to the data over the estimation sample.

Usage

```
fHDShock(x, B = NULL, y = NULL, beta = NULL, p = NULL, c = NULL, exog = NULL)
```

Arguments

x	Either an "fSignRestr" object from fSignRestr, or a T x N numeric matrix of endogenous variables. When an object is supplied, beta, p and c are taken from it and only y or B need overriding.
B	Structural impact matrix. Defaults to the Fry-Pagan draw Bfp when x is an "fSignRestr" object, because it is an accepted model draw; the element-wise median Bmed need not be. Any invertible B reconstructs the data, since residuals and deterministic terms are computed from the same beta.
y	Optional T x N matrix of endogenous variables, required when x is not a matrix and the data are not recoverable from it.
beta	Reduced-form coefficient matrix, [const | lags | exog]. Defaults to the coefficients of the Fry-Pagan draw, beta_all[, , fp_index], when x is an "fSignRestr" object, so that B and beta come from the same draw (the OLS coefficients when inference = 0).
p	Integer lag order.
c	Integer intercept indicator, 1 or 0.
exog	Optional T x M matrix of exogenous regressors.

Value

A list with shock (a T x N x N array: time, variable, shock), init, const, exo and endo, the last being the reconstructed data.

See Also

fSignRestr, fPlotHistDec

Examples

```
data("Uhlig2005")
y <- Uhlig2005 |>
  dplyr::mutate(dplyr::across(-c(Date, `Fed. Funds Rate`), \(v) 100 * v)) |>
  dplyr::select(-Date) |> as.matrix()

SIGN <- matrix(0, 6, 6)
SIGN[, 1] <- c(0, -1, -1, 0, -1, 1)

fit <- fSignRestr(y, p = 12, c = 1, sign = SIGN, nsteps = 24,
                  ndraws = 100, sr_hor = 6, seed = 42, n_threads = 2)
hd <- fHDShock(fit, y = y)

# Components reproduce the data over the estimation sample.
max(abs(hd$endo[-(1:12), ] - y[-(1:12), ]))
```

`fHeteroIRF` *Heteroskedasticity-Based VAR Identification (Proxy IV, Treatment Months)*

Description

Identifies a structural shock using the external instrument restricted to high-variance (treatment) months, following the proxy-SVAR approach of Kaenzig (2021). The structural impact vector is estimated via OLS of the centred residuals on the centred proxy, restricted to OPEC announcement months (`indsR1 == 1`).

Usage

```
fHeteroIRF(var_result, Z, adjustu, indsR1, hor, nvar, scale = 10)
```

Arguments

`var_result`	List returned by `fVAR()`.
`Z`	Proxy matrix (T1 x 1), aligned to the identification window defined by `adjustu`.
`adjustu`	Integer vector `c(start, end)` (1-based) giving the rows of `var_result$residuals` corresponding to the proxy window.
`indsR1`	Integer 0/1 vector of length `adjustu[2]-adjustu[1]+1`; 1 = treatment month (e.g. OPEC announcement), 0 otherwise.
`hor`	IRF horizon.
`nvar`	1-based normalization variable index.
`scale`	Shock size for normalization (default 10).

Value

A list with `IRF` (N x hor+1) and `b1` (N-vector).

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

# A proxy and a 0/1 indicator for the high-variance (treatment) periods
u      <- var_result$residuals
Z      <- matrix(u[, 1] + rnorm(nrow(u), sd = 0.5), ncol = 1)
rows   <- c(1, nrow(u))
indsR1 <- as.integer(abs(Z[, 1]) > stats::median(abs(Z[, 1])))

irf <- fHeteroIRF(var_result, Z, rows, indsR1, hor = 12, nvar = 1,
                  scale = 1)
dim(irf$IRF)
```

`fHistDec` *Historical Decomposition of a VAR Variable*

Description

Decomposes a chosen variable's realisation into contributions from each structural shock, identified via an impact matrix K. Supports both Cholesky (lower-triangular K) and Blanchard-Quah (general K) identification. Mirrors the MATLAB `hist_decmp(y, beta, residuals, c, p, K, series)`.

Usage

```
fHistDec(y, fVAR, K, series)
```

Arguments

`y`	TxN numeric matrix of original (undemeaned) data.
`fVAR`	List returned by `fVAR()`, containing at minimum `beta`, `residuals`, `sigma`, `p`, `c`, and `n_exog`.
`K`	NxN structural impact matrix. For Cholesky identification pass `t(chol(sigma))`; for BQ identification pass the matrix returned by `solve(C1, D1)`.
`series`	Integer (1-indexed) selecting which variable to decompose.

Details

The structural shocks are recovered as

$$\varepsilon_t = K^{-1} u_t$$

where u_t are the reduced-form residuals. The structural MA representation is built by multiplying each Wold matrix by K:

$$\Theta_h = \Psi_h K$$

The contribution of shock j to variable i at time t is then the inner product of $\Theta_{0:t}[i, j]$ with the time-reversed structural shocks $\varepsilon_{t:-1:0,j}$:

$$HD(t, j) = \sum_{h=0}^{t} \Theta_h[i, j]\, \varepsilon_{t-h,j}$$

Value

A list with two elements:

`histdec`	(T-p) x N numeric matrix. Column `j` is the cumulative contribution of structural shock `j` to the chosen variable at each point in time.
`ystar`	(T-p) numeric vector of the demeaned realisation of the chosen variable (benchmark series for the plot).

See Also

`fVAR`, `fCholeskyIRF`, `fWoldIRF`, `fPlotHistDec`

Examples

```
set.seed(123)
y      <- matrix(rnorm(400), ncol = 2)
VAR    <- fVAR(y, p = 2, c = 1)
K_chol <- t(chol(VAR$sigma))
hd     <- fHistDec(y, VAR, K_chol, series = 1)
dim(hd)
```

---

fIRF *Impulse Responses in a Common Tidy Format*

Description

Converts the output of any identification scheme into one long tibble with the same columns, so that results can be plotted with `fPlotIRF`, combined with `rbind()`, or used in a hand-made **ggplot2** figure.

Usage

```
fIRF(x, ...)

## Default S3 method:
fIRF(
  x,
  bands = NULL,
  varnames = NULL,
  shocknames = NULL,
```

```
  cumulate = NULL,
  scale = 1,
  conf = NULL,
  model = NA_character_,
  ...
)

## S3 method for class 'list'
fIRF(
  x,
  varnames = NULL,
  shocknames = NULL,
  scale = 1,
  conf = NULL,
  model = NA_character_,
  ...
)

## S3 method for class 'fLP'
fIRF(x, variables = NULL, scale = 1, model = NA_character_, ...)

## S3 method for class 'fSignRestr'
fIRF(
  x,
  conf = NULL,
  varnames = NULL,
  shocknames = NULL,
  scale = 1,
  model = NA_character_,
  ...
)

## S3 method for class 'fIRF'
fIRF(x, scale = 1, model = NULL, ...)
```

Arguments

x	Point estimate, result list or fitted object (see Details).
...	Passed to methods.
bands	Bootstrap result with upper, lower and optionally upper2, lower2, in the same dimensions as x.
varnames	Character vector of variable names (length N).
shocknames	Character vector of shock names (one per shock).
cumulate	Variables (indices or names) whose *point* estimate is cumulated over horizons. Bootstrap bands are taken as given: fBootstrapBQ(cumulate = ...) already returns cumulated bands.
scale	Scalar, or one value per variable, multiplying the point estimate and the bands. A negative value flips the sign and keeps lower <= upper.

conf	Band levels in percent, outer band first. By default taken from the result (conf element) or c(90, 68). For fSignRestr, selects which stored levels to use.
model	Label for the model column; fPlotIRF() fills it from argument names when missing.
variables	For LP fits, the response variables to keep.

Details

Supported inputs:

- a point estimate (N x N x H array from fCholeskyIRF or fBQIRF, or N x H matrix for a single shock) with optional bands from fBootstrapChol, fBootstrapCholCorrected, fBootstrapBQ, fBootstrapMax or fBootstrapUhlig;
- the lists returned by fBootstrapIVMBB and fBootstrapHetero (both carry the sample estimate in point), and fMSW (Anderson-Rubin and delta-method bands);
- fLP, fLPIV and fLPPanel fits;
- fSignRestr fits (point = posterior median).

Value

A tibble of class "fIRF" with columns model, shock, variable, horizon, band, point, lower, upper.

See Also

fPlotIRF

Examples

```
set.seed(123)
y     <- matrix(rnorm(400), ncol = 2, dimnames = list(NULL, c("y1", "y2")))
fit   <- fVAR(y, p = 2, c = 1)
wold  <- fWoldIRF(fit, horizon = 12)
point <- fCholeskyIRF(wold, t(chol(fit$sigma)))
boot  <- fBootstrapChol(y, fit, nboot = 50, horizon = 12, n_threads = 2)

irf <- fIRF(point, bands = boot, varnames = colnames(y))
head(irf)
fPlotIRF(irf, shock = 1)
```

`fLagMakerMatrix` *Create Lagged Matrix*

Description

Creates a matrix of lagged values from a time series matrix. For each variable and each lag from 1 to p, this function creates a column containing the lagged values, removing the first p rows to align the lags properly.

Usage

```
fLagMakerMatrix(y, p)
```

Arguments

`y`	A numeric matrix of time series data with T rows (time periods) and N columns (variables)
`p`	An integer specifying the number of lags to create

Details

The function creates a lagged matrix where each column represents a lagged version of the original variables. The lags are organized by lag order first, then by variable. For example, with 2 variables and 2 lags, the column order would be: var1_lag1, var2_lag1, var1_lag2, var2_lag2.

Value

A numeric matrix with (T-p) rows and (N*p) columns. The columns are organized as: all N variables at lag 1, then all N variables at lag 2, etc. Column names are generated as "var_name_lag_k" where applicable.

Examples

```
y <- cbind(var1 = 1:10, var2 = 11:20)
x <- fLagMakerMatrix(y, 2)
head(x)

# A data frame must be converted first
df <- data.frame(var1 = 1:10, var2 = 11:20)
head(fLagMakerMatrix(as.matrix(df), 2))
```

fLP *Classical Local Projections (fixest-like syntax)*

Description

Estimates impulse responses via the Jordà (2005) local projection method. The syntax mirrors **fixest**: use a formula for the model, `..macroname` tokens to expand character-vector variable lists, and `l()` / `f()` operators to create lags and leads on-the-fly.

Usage

```
fLP(
  formula,
  data,
  horizons = 12L,
  shock,
  conf = 90,
  nw_lags = NULL,
  nw_offset = 1L,
  nw_prewhite = FALSE,
  store_full = FALSE,
  cumulative = FALSE,
  shock_size = c("unit", "sd"),
  n_threads = 0L
)
```

Arguments

`formula` A formula of the form `y ~ shock + controls`.

- **LHS**: one variable, or `c(y1, y2)` for multi-equation. LHS variables are *never* automatically placed on the RHS.
- **RHS**: exactly what you want in the regression. No automatic lag augmentation. Use `l()` / `f()` for that.
- **Macros**: write `..name` where `name` is a character vector in the calling environment.
- **Lag operator**: `l(var, k)` or `l(var, k1:k2)`.
- **Lead operator**: `f(var, k)` or `f(var, k1:k2)`.
- Macros inside lag/lead: `l(..controls, 1:3)`.

`data` A `data.frame` or `tibble`. Must contain all variables referenced in the formula (before lag/lead expansion). Rows are consecutive periods. The regressors must be observed on one block of rows. Outcomes may have `NA`: at each horizon h, each outcome uses the dates t whose regressors and y_{t+h} are observed (and y_{t-1} when `cumulative = TRUE`), as Stata does with `F.h y`. So y_{t+h} is used after the regressors end, and each outcome of `c(y1, y2)` has its own sample. An outcome must be observed on one block of rows; an interior gap is an error.

`horizons` Integer. Maximum horizon H, or the complete sequence `0:H`; projections run for $h = 0, 1, \ldots, H$. Default: 12.

`shock` Character. **Required.** Name of the shock/impulse variable (must appear on the RHS). This is the variable whose coefficient at each horizon is returned as the IRF.

`conf` Numeric in (0, 100), in percent. One or more confidence levels for bands, e.g. `90` or `c(68, 95)`. When more than one is supplied, `irfs_upper` and `irfs_lower` become named lists of matrices (one per level, keyed by the level as a string) and the plot functions (`fPlotIRF()` for LP fits, `fPlotNLLP()` for nonlinear fits) draw one ribbon per level. Bands are always rebuilt in R from `irfs_se`, so multi-level fits cost the same as single-level fits. Default: 90.

`nw_lags` Integer or `NULL`. Base Newey-West bandwidth. At horizon h the effective bandwidth is `max(nw_lags + h + nw_offset, 0)` (Bartlett kernel), capped at the horizon's observations minus one. Defaults to `floor(T^(1/3))`, with T the number of dates with every regressor observed.

`nw_offset` Integer. Shift applied to the NW bandwidth rule. Default `1L`, the Miranda-Agrippino & Ricco convention (bandwidth grows as $h + 1$). Set to `0L` to reproduce the classic Jordà (2005) rule, bandwidth = h (LP residuals at horizon h are MA(h) by construction). `fLP`'s `h` is 0-indexed (h=0 is the impact horizon), which makes this the exact equivalent of Cesa-Bianchi's VAR Toolbox (`LPmodel.m`: `OLSmodel(Y,X,0,hh-1)` with 1-indexed `hh`, so `hh-1 == h`) — needed to match that toolbox's SEs exactly, e.g. in the Jordà-Taylor (2025) replication.

`nw_prewhite` Logical. If `TRUE`, apply Andrews-Monahan (1992) VAR(1) prewhitening to the regression scores before the Newey-West sum (same bandwidth), shrinking the VAR(1) matrix when its spectral radius exceeds 0.97. Default `FALSE` (plain Newey-West, as in the VAR Toolbox).

`store_full` Logical. If `TRUE`, return full coefficient and standard-error matrices for every horizon. Defaults to `FALSE` to keep the result small and fast.

`cumulative` Logical. If `TRUE`, the LHS at horizon h is $y_{t+h} - y_{t-1}$ (cumulative response from $t-1$ to $t+h$). Default `FALSE`. When generated lags trim the leading rows, the outcome immediately before the first regressor date is still observed and is used as that date's long-difference base, so no observation is lost — matching `endo_lag1` in Cesa-Bianchi's `LPmodel.m`.

`shock_size` Character. Units of the reported impulse response. `"unit"` (default) = response to a one-unit shock. `"sd"` = response to a one-standard-deviation shock, obtained by z-scoring the shock over the estimation sample. These correspond to `LPopt.impact = 1` and `LPopt.impact = 0` respectively in Cesa-Bianchi's `LPmodel.m`; note the standard deviation is taken over the dates with every regressor observed (the lag-trimmed sample), not the raw input column, and is the same at every horizon.

`n_threads` Integer. Number of OpenMP threads for the horizon loop. `0` = all cores minus one. Default `0`.

Value

An object of class `"fLP"`: a list with `irfs`, `irfs_se`, `irfs_lower`, `irfs_upper` (`(H+1) x n_y`; lists of matrices for several `conf` levels), `nobs` (`(H+1) x n_y` observations per horizon and outcome),

with `store_full = TRUE` also `betas` and `ses`, plus the settings of the call.

Examples

```
data("JordaTaylor2025")

## 1. Explicit shock and controls
fLP(dLogCPI ~ RRShock + dSTIR + dLogGDP,
    data = JordaTaylor2025, horizons = 6, shock = "RRShock",
    n_threads = 2)

## 2. Lag operator — creates the lags from the data
fLP(dLogCPI ~ RRShock + l(dLogCPI, 1:3),
    data = JordaTaylor2025, horizons = 6, shock = "RRShock",
    n_threads = 2)

## 3. Macro expansion
controls <- c("dSTIR", "dLogGDP")
fLP(dLogCPI ~ RRShock + ..controls,
    data = JordaTaylor2025, horizons = 24, shock = "RRShock",
    n_threads = 2)

## 4. Macro inside the lag operator
vars <- c("dLogCPI", "dSTIR", "dLogGDP")
fLP(dLogCPI ~ RRShock + l(..vars, 1:3),
    data = JordaTaylor2025, horizons = 24, shock = "RRShock",
    n_threads = 2)

## 5. Several outcomes at once (one regression each)
fLP(c(dLogCPI, dLogGDP, dSTIR) ~ RRShock + l(..vars, 1:3),
    data = JordaTaylor2025, horizons = 24, shock = "RRShock",
    n_threads = 2)

## 6. Lead operator
fLP(dLogCPI ~ RRShock + l(dLogCPI, 1:12) + f(dSTIR, 1),
    data = JordaTaylor2025, horizons = 24, shock = "RRShock",
    n_threads = 2)
```

`fLPDID` *Local Projections Difference-in-Differences (fixest-like syntax)*

Description

Estimates event-study treatment effects with the LP-DiD estimator of Dube, Girardi, Jordà and Taylor (2025, *Journal of Applied Econometrics*, doi:10.1002/jae.70000): long-difference local projections on clean-control samples with absorbed time fixed effects and cluster-robust (CR1, Stata 'reghdfe' convention) standard errors. The horizon loop runs in parallel C++ (OpenMP). Formula syntax mirrors `fLPPanel`: supports `..macro` expansion and the panel-aware operators `l()`, `f()` and `d()`.

Usage

```
fLPDID(
  formula,
  data,
  panel_id,
  treat,
  post = 10L,
  pre = 5L,
  cluster = NULL,
  nonabsorbing = FALSE,
  L = NULL,
  ccc = 1L,
  pmd = FALSE,
  reweight = FALSE,
  conf = 90,
  n_threads = -1L
)
```

Arguments

formula	A formula `y ~ treat + controls`, e.g. `lshare ~ bank + l(lshare, 1:4) + l(d(lshare), 1:4)`. • **LHS**: the outcome. Transformations are evaluated, so `log(y) ~ treat` estimates the model for `log(y)`. • **RHS**: the treatment *level* column (0/1; `NA` allowed) plus any controls. The term named in `treat` is the treatment; all other terms are controls. • **Macros**: `..name` where `name` is a character vector in the calling environment. • **Operators**: `l(x, k)`, `l(x, k1:k2)`, `f(x, k)`, `d(x)`, `l(d(x), 1:4)`, `l(..vars, 1:4)`. Never cross unit boundaries; time gaps are preserved. • Time fixed effects are always absorbed; an optional `| time` part is accepted but redundant. Unit effects are removed by the long difference.
data	A `data.frame` or `tibble` in long panel form.
panel_id	Length-2 character vector `c(id_col, time_col)`.
treat	Character. The RHS term label of the treatment level.
post, pre	Number of post- and pre-treatment horizons (`event_time = -1` is the normalization unless `pmd`).
cluster	Column name for clustering. Default: the unit id.
nonabsorbing	Set `TRUE` when treatment can switch on and off (entries and reversals). Requires `L`.
L	Stabilization window for the clean-control sets (non-absorbing case).
ccc	Clean-control condition flavor for the non-absorbing case: `0` = none (ANRR-style), `1` = clean past for treated and controls, `2` = controls additionally clean through `t+h`.

pmd	Use the pre-mean-differenced baseline (average of all past outcomes) instead of y[t-1].
reweight	Recover the equally-weighted ATT via inverse implicit weights 1/(1-p_t) (exogenous/absorbing case; with covariates this is an approximation — the regression-adjustment version is not yet implemented in C++).
conf	Confidence level for the reported bands, in percent (default 90).
n_threads	OpenMP threads (<= 0 = all cores minus one).

Value

A tibble of class "fLPDID" with columns event_time, estimate, se, conf_low, conf_high, nobs, nclust, ndrop. Confidence bands use a cluster-t critical value with nclust - 1 degrees of freedom (the reghdfe convention); ndrop counts control columns omitted for collinearity at that horizon.

Examples

```
data("BankingDeregulation")

m <- fLPDID(lshare ~ bank, data = BankingDeregulation,
            panel_id = c("id", "year"), treat = "bank",
            post = 4, pre = 4, n_threads = 2)
m
fPlotLPDID(m)

# Controls through the panel-aware operators and a macro
ctrl <- c("grgsp", "corptax", "unionmem")
fLPDID(lshare ~ bank + l(lshare, 1:4) + l(d(lshare), 1:4) + l(..ctrl, 1:4),
       data = BankingDeregulation, panel_id = c("id", "year"),
       treat = "bank", post = 9, pre = 9, n_threads = 2)
```

---

fLPIV *Local Projections with an External Instrument (LP-IV)*

Description

Estimates 2SLS local projections horizon-by-horizon using the delta-method Newey-West HAC sandwich for the endogenous treatment effect. Matches Cesa-Bianchi's VAR Toolbox LPmodel.m in LP-IV mode when nw_lags_iv is set to the toolbox's nlag_iv.

Usage

```
fLPIV(
  formula,
  instruments,
  data,
  endog,
  horizons = 12L,
```

```
  conf = 90,
  nw_lags_iv = NULL,
  nw_prewhite = FALSE,
  cumulative = TRUE,
  shock_size = c("unit", "sd"),
  n_threads = 0L
)
```

Arguments

formula	A formula `y ~ D + controls`. LHS may be `c(y1, y2, ...)` for multi-outcome LP-IV. RHS lists the endogenous treatment plus all exogenous controls. `l()` / `f()` lag/lead operators and `..macro` expansion behave exactly as in `fLP()`.
instruments	A one-sided formula `~ z + ...` whose RHS is the full external instrument matrix. Use `l(z, 0:6)` to include the contemporaneous instrument plus six lags (over-identified 2SLS).
data	A `data.frame` or `tibble`. Rows are consecutive periods. Treatment, controls and instruments must be observed on one block of rows; outcomes may have `NA`, handled per horizon and outcome as in `fLP` (both stages use the dates whose y_{t+h}, and y_{t-1} when cumulative, is observed).
endog	Character. The single RHS term (of `formula`) that is the endogenous treatment. The remaining RHS terms are the exogenous control matrix.
horizons	Integer or the sequence `0:H`. Default 12.
conf	Numeric in (0, 100), in percent. Scalar or vector of levels. Vectors trigger multi-band output; see `fLP`.
nw_lags_iv	Integer. Bartlett HAC bandwidth for the IV score variance. `> 0`: fixed bandwidth across horizons — this matches Stata's `vce(hac nw <k>)` and the toolbox's LP-IV bands. `= 0`: horizon-varying bandwidth `= h` (just-identified rule). Default `NULL` → use `ncol(Z) - 1`, i.e. the lag order implied by the instrument formula, which reproduces the toolbox default.
nw_prewhite	Logical. If `TRUE`, apply Andrews-Monahan (1992) VAR(1) prewhitening to the regression scores before the Newey-West sum (same bandwidth), shrinking the VAR(1) matrix when its spectral radius exceeds 0.97. Default `FALSE` (plain Newey-West, as in the VAR Toolbox).
cumulative	Logical. `TRUE` regresses $y_{t+h} - y_{t-1}$ on the RHS. Default `TRUE` — the standard long-difference LP-IV convention. When generated lags trim the leading rows, the outcome immediately before the first regressor date is still observed and is used as that date's base, so no observation is lost.
shock_size	Character. Units of the reported impulse response. `"unit"` (default) = response to a one-unit move in the treatment. `"sd"` = response to a one-standard-deviation move, scaling the 2SLS coefficient and its standard error by `sd(D)` over the dates with every regressor observed (one value for all horizons). These correspond to `LPopt.impact = 1` and `LPopt.impact = 0` in Cesa-Bianchi's `LPmodel.m`.
n_threads	Integer. OpenMP threads for the horizon loop. Default 0 (= all cores minus one).

Value

An object of class `c("fLPIV", "fLP")` so all existing `fLP`-family methods (`print`, `tidy.fLP`, `fPlotIRF`) work unchanged. Additional fields: `Fstat_fs` and `rsqr_fs` (both `(H+1) x n_y`: first-stage F-statistic and R² per horizon and outcome, since each outcome's sample has its own first stage), `nobs` (`(H+1) x n_y` observations), plus `endog` and `instrument_vars`. `Fstat_fs` is the textbook exclusion F for $D \sim [1, C, Z]$, with $T_h - k_c - n_z$ residual degrees of freedom. Cesa-Bianchi's `LPmodel.m` reports $n_h - k_z$ instead (it ignores the df consumed by the residualised controls), so the two F values differ by construction. Point estimates, standard errors and bands are unaffected.

Examples

```
data("JordaTaylor2025IV")

fLPIV(Unemployment ~ FFRates + l(Unemployment, 1:3),
      instruments = ~ l(RRShock, 0:3),
      data        = JordaTaylor2025IV,
      endog       = "FFRates",
      horizons    = 6,
      n_threads   = 2)

controls <- c("Unemployment", "Inflation", "FFRates")
fLPIV(Unemployment ~ FFRates + l(..controls, 1:6),
      instruments = ~ l(RRShock, 0:6),
      data        = JordaTaylor2025IV,
      endog       = "FFRates",
      horizons    = 48,
      conf        = c(68, 95),
      nw_lags_iv  = 6,
      cumulative  = TRUE,
      n_threads   = 2)
```

---

fLPPanel *Panel Local Projections (fixest-like syntax)*

Description

Estimates panel impulse responses by local projections, as in the household debt application of Mian, Sufi and Verner (2017, QJE). Formula syntax mirrors **fixest**: RHS variables before `|` are the shock and controls, variables after `|` are absorbed fixed effects. Interactions (e.g. `shock:size`) are computed explicitly from the formula — the estimator uses *exactly* the specification the user writes. Supports `..macro` expansion and panel-aware `l()` / `f()` operators.

Usage

```
fLPPanel(
  formula,
  data,
```

```
  panel_id,
  horizons = 12L,
  shock,
  conf = 90,
  ssc = c("none", "CR1", "IK"),
  small_sample = FALSE,
  cumulative = FALSE,
  p_max = 0L,
  n_threads = 0L,
  cluster = NULL
)
```

Arguments

formula	A formula of the form `y ~ shock + controls | fe1 + fe2`, e.g. `y ~ shock:size + l(y, 1:2) | unit + tt`. • **LHS**: single response variable. • **RHS-main** (before `|`): every term you want in the regression. Interactions with `:` are computed as elementwise products (this is where the heterogeneity term $s_{it} \cdot X_t$ appears — write `shock:size` in the formula). • **RHS-FE** (after `|`): categorical fixed-effect variables (omit `|` for no fixed effects). • **Macros**: `..name` where `name` is a character vector in the calling environment. • **Panel-aware lag/lead**: `l(x, k)`, `l(x, k1:k2)`, `f(x, k)`, `l(..vars, 1:3)`. Never crosses unit boundaries.
data	A `data.frame` or `tibble` in long panel form.
panel_id	Length-2 character vector `c(id_col, time_col)` naming the unit-id column and the time column, respectively.
horizons	Integer. Maximum horizon H (projections run for $h = 0, 1, \ldots, H$) or the full sequence `0:H`. Default 12.
shock	Character (single or vector). The RHS-main term label(s) whose coefficient is the impulse response. Everything else in the RHS-main is treated as a control. Must match the term label exactly (after macro / lag expansion); interactions are labelled with `":"`, e.g. `"shock:size"`.
conf	Numeric in (0, 100), in percent. One or more confidence levels, e.g. `90` or `c(68, 95)`. Bands are rebuilt from the raw SEs, so multi-level fits cost the same as single-level fits.
ssc	Character. Small-sample correction applied to the cluster-robust variance: `"none"` (default) the asymptotic sandwich with normal critical values (`df = Inf`). This is the convention of asymptotic panel LP inference, in both the cross-section and the time dimension, and of Stata's `xtivreg2` without the `small` option. `"CR1"` the correction used by Stata and **fixest**: every one-way component of the meat is multiplied by $G/(G-1) \cdot (N-1)/(N-K)$ with its own number of clusters G, and inference uses t with $\min(G) - 1$ degrees of freedom. K

counts the regressors plus the absorbed fixed effects that are not nested in a clustering dimension. Standard errors are slightly wider than `"none"`; the difference is negligible with many clusters and matters when there are few. Available for every clustering dimension.

`"IK"` the Imbens-Kolesar (2016, REStat) refinement, a bias-reduced (CR2-type) variance with Satterthwaite degrees of freedom. Defined only for time clustering.

Point estimates are identical under all three; only the standard errors and the critical values change.

`small_sample` Deprecated. `TRUE` is equivalent to `ssc = "IK"`; use `ssc` instead.

`cumulative` Logical. If `TRUE`, project $\sum_{k=0}^{h} y_{i,t+k}$ instead of $y_{i,t+h}$. Default `FALSE`.

`p_max` Integer. Maximum number of lags of the regressand and of the shock terms added as controls (at horizon h the effective number of lags is $\min(h, p_{\max})$). Default 0.

`n_threads` Integer. OpenMP threads for the horizon loop. 0 = all cores minus one. Default 0.

`cluster` One-sided formula selecting the clustering dimension: the unit variable, the time variable, or both variables declared in `panel_id`; for example `~unit`, `~tt`, or `~unit + tt`. Two-way clustering uses the Cameron-Gelbach-Miller inclusion-exclusion estimator. The default `NULL` is equivalent to clustering on the time variable. The `ssc = "IK"` refinement is available only with time clustering.

Details

Each horizon h is estimated on its own sample: the rows whose led regressand $y_{i,t+h}$, shock, controls and lags at t are all observed. Missing values in these columns are kept in the data, so a row with a missing regressor can still be the lead target of an earlier row. This is the sample Stata or **fixest** use when the leads are built before the regression. Only rows with a missing unit, time or fixed effect are dropped before estimation.

Value

An object of class `c("fLPPanel", "fLP")` that plugs into `fPlotIRF` and `tidy.fLP`. Element `nobs` holds the number of observations used at each horizon.

Examples

```
data("DemocracyGrowth")

# Response of log GDP to a democratisation event, unit and year fixed
# effects, clustered on countries.
fLPPanel(lgdp ~ dem | id + year,
         data = DemocracyGrowth, panel_id = c("id", "year"),
         shock = "dem", horizons = 4, cluster = ~id, n_threads = 2)

# The Stata / fixest finite-sample correction
fLPPanel(lgdp ~ dem | id + year,
         data = DemocracyGrowth, panel_id = c("id", "year"),
```

```
          shock = "dem", horizons = 12, cluster = ~id,
          ssc = "CR1", n_threads = 2)

# Cumulative response with lags of the outcome as controls
fLPPanel(lgdp ~ dem + l(lgdp, 1:2) | id + year,
          data = DemocracyGrowth, panel_id = c("id", "year"),
          shock = "dem", horizons = 12, cumulative = TRUE,
          cluster = ~id + year, n_threads = 2)
```

`fMaxIRF` *Long-Run Maximum-Response Impulse Responses*

Description

Selects the shock direction that maximises the response of a selected variable at the last supplied horizon, subject to a zero first shock coordinate and unit norm. The sign is chosen so the selected variable's last-horizon response is non-negative.

Usage

```
fMaxIRF(wold, S, var_idx)
```

Arguments

`wold`	N x N x (horizon+1) Wold impulse response array, including horizon zero in the first slice, as returned by `fWoldIRF`.
`S`	N x N lower triangular Cholesky factor of the VAR residual covariance matrix. At least two variables are required.
`var_idx`	Index (1-based) of the variable whose response is maximised.

Value

An N x (horizon+1) matrix of structural impulse responses.

See Also

`fBootstrapMax`, `fUhligIRF`

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 12)

# Shock with the largest long-run effect on variable 1
irf <- fMaxIRF(wold, t(chol(var_result$sigma)), var_idx = 1)
dim(irf)
```

fMBBVAR *Moving Block Bootstrap for VAR Residuals and Instruments*

Description

Computes a moving-block bootstrap sample for the reduced form errors of a VAR(p) model. Optionally bootstraps a matrix of instruments M.

Usage

```
fMBBVAR(eps, lags, BlockSize, M = NULL)
```

Arguments

eps	(T-p) x N matrix of VAR residuals
lags	Integer lag order p of the VAR model
BlockSize	Integer block size for moving block bootstrap. If 0 or negative, automatically computed as round(5.03 * T^0.25)
M	Optional (T-p) x K matrix of instruments to bootstrap (default = NULL)

Details

The moving block bootstrap preserves the temporal dependence structure in the residuals. The default block size follows the recommendation from "Proxy SVARs: Asymptotic Theory, Bootstrap Inference, and the Effects of Income Tax Changes in the United States".

Both residuals and instruments are centered to preserve their sample properties. For instruments, only non-zero values are centered (to handle sparse instruments), and the centering keeps the sample mean of the non-zero values, as in Kaenzig (2021).

Value

A list containing:

- eps_boot: (T-p) x N matrix of bootstrapped residuals (centered)
- M_boot: (T-p) x K matrix of bootstrapped instruments (centered), or empty matrix if M not provided

Examples

```
set.seed(123)
eps    <- matrix(rnorm(200), 100, 2)
result <- fMBBVAR(eps, lags = 2, BlockSize = 0)
dim(result$eps_boot)

# Bootstrap the residuals and an instrument jointly
M      <- matrix(rnorm(100), 100, 1)
result <- fMBBVAR(eps, lags = 2, BlockSize = 0, M)
```

```
dim(result$M_boot)
```

---

fMSW *Weak-IV Robust Impulse Response Inference (Montiel-Olea, Stock, Watson 2021)*

Description

Computes both plug-in (delta method) and Anderson-Rubin weak-IV robust confidence sets for IV-identified structural IRFs following Montiel-Olea, Stock and Watson (2021).

Usage

```
fMSW(
  var_result,
  Z,
  finaldata,
  adjustu,
  hor = 48L,
  nvar = 1L,
  scale = 1,
  confidence = 0.9,
  NWlags = 0L
)
```

Arguments

var_result	List from fVAR.
Z	Instrument matrix (proxy sample, no missing values).
finaldata	T x N matrix of original endogenous variables (full sample).
adjustu	Integer vector c(start, end) (1-based) selecting the proxy-sample rows of the VAR residuals.
hor	Forecast horizon. Default 48.
nvar	1-based index of the normalization variable. Default 1.
scale	Scale of the structural shock. Default 1.
confidence	Confidence level (0-1). Default 0.9.
NWlags	Number of Newey-West lags for HAC correction: 0 = White heteroskedasticity-consistent only (no autocorrelation), k > 0 = Newey-West with k lags. Must be >= 0. Default 0.

Value

A list with elements:

IRF	n x (hor+1) plug-in IRF point estimates.
IRFstderr	n x (hor+1) delta-method standard errors.
Dmlbound	n x (hor+1) delta-method lower confidence bounds.
Dmubound	n x (hor+1) delta-method upper confidence bounds.
MSWlbound	n x (hor+1) Anderson-Rubin lower bounds.
MSWubound	n x (hor+1) Anderson-Rubin upper bounds.
Waldstat	Wald statistic for instrument relevance (HAC-robust F-stat).
Fstat	Same as Waldstat (reported for compatibility).

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)

u    <- var_result$residuals
Z    <- matrix(u[, 1] + rnorm(nrow(u), sd = 0.5), ncol = 1)
rows <- c(1, nrow(u))

msw <- fMSW(var_result, Z, y, rows, hor = 12, nvar = 1, NWlags = 2)
msw$Fstat
dim(msw$MSWlbound)
```

---

fNLLP *Nonlinear and State-Dependent Local Projections*

Description

Estimates local projections in which the shock has two coefficients per horizon: one per regime (smooth state dependence), one per sign (asymmetry), or a linear and a cubic term (size nonlinearity). The formula syntax is that of fLP: the RHS is estimated as written, a constant is added internally, and `l()`, `f()` and `..macro` expand the same way. Both coefficients come from one regression, together with their Newey-West covariance, so their difference has a valid standard error.

Usage

```
fNLLP(
  formula,
  data,
  horizons = 12,
  shock,
  type = c("state", "sign", "cubic"),
```

```
  state = NULL,
  common = NULL,
  regimes = c("high", "low"),
  conf = 90,
  nw_lags = NULL,
  nw_offset = 1,
  nw_prewhite = FALSE,
  balanced = FALSE,
  cumulative = FALSE,
  store_full = FALSE,
  n_threads = 0
)
```

Arguments

formula	A formula of the form `y ~ shock + controls`. • **LHS**: one variable, or `c(y1, y2)` for multi-equation. LHS variables are *never* automatically placed on the RHS. • **RHS**: exactly what you want in the regression. No automatic lag augmentation. Use `l()` / `f()` for that. • **Macros**: write `..name` where `name` is a character vector in the calling environment. • **Lag operator**: `l(var, k)` or `l(var, k1:k2)`. • **Lead operator**: `f(var, k)` or `f(var, k1:k2)`. • Macros inside lag/lead: `l(..controls, 1:3)`.
data	A `data.frame` or `tibble`. Must contain all variables referenced in the formula (before lag/lead expansion). Rows are consecutive periods. The regressors must be observed on one block of rows. Outcomes may have `NA`: at each horizon h, each outcome uses the dates t whose regressors and y_{t+h} are observed (and y_{t-1} when `cumulative = TRUE`), as Stata does with `F.h y`. So y_{t+h} is used after the regressors end, and each outcome of `c(y1, y2)` has its own sample. An outcome must be observed on one block of rows; an interior gap is an error.
horizons	Integer. Maximum horizon H, or the complete sequence `0:H`; projections run for $h = 0, 1, \ldots, H$. Default: 12.
shock	Character. **Required.** Name of the shock variable; it must appear on the RHS.
type	How the shock enters the regression: `"state"` (default) every RHS term x and the intercept enter as Fx and $(1-F)x$, with F the column named by `state`. Components are named by `regimes`. `"sign"` the shock enters as $\max(s, 0)$ and $\min(s, 0)$. Components `"positive"` and `"negative"`, each per unit of the shock. `"cubic"` the shock enters as s and s^3. Components `"linear"` and `"cubic"`; use `predict()` for the response to a shock of a given size.
state	Character. For `type = "state"`: name of the column of `data` holding the regime weight F, in $[0, 1]$. It must not also appear on the RHS.
common	Character vector of RHS terms that keep a single coefficient across regimes (e.g. a trend). Only for `type = "state"`; the shock cannot be common.

`regimes` Two distinct labels for the components of `type = "state"`: the first for weight F, the second for $1 - F$. Default `c("high", "low")`.

`conf` Numeric in (0, 100), in percent. One or more confidence levels for bands, e.g. `90` or `c(68, 95)`. When more than one is supplied, `irfs_upper` and `irfs_lower` become named lists of matrices (one per level, keyed by the level as a string) and the plot functions (`fPlotIRF()` for LP fits, `fPlotNLLP()` for nonlinear fits) draw one ribbon per level. Bands are always rebuilt in R from `irfs_se`, so multi-level fits cost the same as single-level fits. Default: 90.

`nw_lags` Integer or `NULL`. Base Newey-West bandwidth. At horizon h the effective bandwidth is `max(nw_lags + h + nw_offset, 0)` (Bartlett kernel), capped at the horizon's observations minus one. Defaults to `floor(T^(1/3))`, with T the number of dates with every regressor observed.

`nw_offset` Integer. Shift applied to the NW bandwidth rule. Default `1L`, the Miranda-Agrippino & Ricco convention (bandwidth grows as $h + 1$). Set to `0L` to reproduce the classic Jordà (2005) rule, bandwidth = h (LP residuals at horizon h are MA(h) by construction). `fLP`'s `h` is 0-indexed (h=0 is the impact horizon), which makes this the exact equivalent of Cesa-Bianchi's VAR Toolbox (`LPmodel.m`: `OLSmodel(Y,X,0,hh-1)` with 1-indexed `hh`, so `hh-1 == h`) — needed to match that toolbox's SEs exactly, e.g. in the Jordà-Taylor (2025) replication.

`nw_prewhite` Logical. If `TRUE`, apply Andrews-Monahan (1992) VAR(1) prewhitening to the regression scores before the Newey-West sum (same bandwidth), shrinking the VAR(1) matrix when its spectral radius exceeds 0.97. Default `FALSE` (plain Newey-West, as in the VAR Toolbox).

`balanced` Logical. If `TRUE`, the regressor dates are the same at every horizon and the outcome is read h rows later, as in the Tenreyro-Thwaites (2016) code; `data` must then run `horizons` rows past the last regressor date, and the regressors must have no interior gaps. Default `FALSE`: as in `fLP()`, each horizon and outcome uses the dates whose regressors and y_{t+h} are observed, so the sample shrinks with the horizon once the outcome ends.

`cumulative` Logical. If `TRUE`, the LHS at horizon h is $y_{t+h} - y_{t-1}$ (cumulative response from $t-1$ to $t+h$). Default `FALSE`. When generated lags trim the leading rows, the outcome immediately before the first regressor date is still observed and is used as that date's long-difference base, so no observation is lost — matching `endo_lag1` in Cesa-Bianchi's `LPmodel.m`.

`store_full` Logical. If `TRUE`, also return the full coefficient and standard-error matrices of every horizon (`betas`, `ses`). Default `FALSE`.

`n_threads` Integer. Number of OpenMP threads for the horizon loop. `0` = all cores minus one. Default `0`.

Value

An object of class `"fNLLP"`, a list with

`irfs`, `irfs_se` Arrays `(H+1) x n_y x 2` [horizon, outcome, component] of the two shock coefficients and their Newey-West standard errors.

`irfs_upper`, `irfs_lower` Normal bands of the same shape; named lists of arrays (one per level) when `conf` has several levels.

`irfs_cov` `(H+1) x n_y` covariance of the two coefficients.

`diff`, `diff_se` `(H+1) x n_y` first minus second component and its standard error; `NaN` for `type = "cubic"`.

`nobs` `(H+1) x n_y` observations per horizon and outcome.

`betas`, `ses` With `store_full = TRUE`: one matrix per horizon, rows named by `design_vars`.

plus the settings (`type`, `components`, `design_vars`, `shock`, `state`, `common`, `horizons`, `conf`, `nw_lags`, `nw_offset`, `nw_prewhite`, `balanced`, `cumulative`, `formula`, ...). `print`, `coef`, `predict` and `tidy.fNLLP` methods are available.

References

Tenreyro, S., & Thwaites, G. (2016). Pushing on a string: US monetary policy is less powerful in recessions. *American Economic Journal: Macroeconomics*, 8(4), 43–74.

See Also

`fLP`, `fPlotNLLP`, `tidy.fNLLP`, `predict.fNLLP`, `TenreyroThwaites2016`

Examples

```
data("TenreyroThwaites2016")
tt <- TenreyroThwaites2016
tt$GDP   <- 100 * tt$logGDP
tt$trend <- seq_len(nrow(tt))
# Expansion weight: logistic transform of standardised trailing
# four-quarter GDP growth
z <- as.numeric(stats::filter(c(NA, diff(tt$GDP)), rep(1 / 4, 4), sides = 1))
tt$Expansion <- plogis(3 * (z - mean(z, na.rm = TRUE)) / sd(z, na.rm = TRUE))
tt <- tt[tt$Date >= as.Date("1968-10-01"), ]

fit <- fNLLP(GDP ~ RRShock + l(GDP, 1) + l(FFR, 1) + trend,
             data = tt, horizons = 12, shock = "RRShock",
             type = "state", state = "Expansion", common = "trend",
             regimes = c("Expansion", "Recession"), balanced = TRUE,
             n_threads = 2)
fit

# Asymmetry between contractionary and expansionary shocks
fit_sign <- fNLLP(GDP ~ RRShock + l(GDP, 1) + l(FFR, 1) + trend,
                  data = tt, horizons = 12, shock = "RRShock",
                  type = "sign", balanced = TRUE, n_threads = 2)
fit_sign
```

fOLS *Ordinary Least Squares Regression*

Description

Ordinary Least Squares Regression

Usage

```
fOLS(y, X, c = 1L, lag = 0L)
```

Arguments

y	Dependent variable matrix (T x 1)
X	Independent variables matrix (T x N)
c	Integer indicator for intercept (1 if included, 0 otherwise)
lag	Number of lags for the robust F-statistic. Zero uses White's heteroskedasticity-robust covariance; a positive value uses Newey-West HAC covariance with that lag order.

Value

A list containing:

- beta: Coefficient estimates
- fitted: Fitted values
- err: Residuals
- r2: R-squared
- r2adj: Adjusted R-squared
- F: F-statistic for overall significance
- Frobust: White or Newey-West robust F-statistic
- fitted_partial: Fitted values excluding intercept

Examples

```
set.seed(123)
X   <- matrix(rnorm(200), ncol = 2)
y   <- X %*% c(1, -0.5) + rnorm(100)
fit <- fOLS(y, X, c = 1)
names(fit)
```

`fPlotHistDec` *Plot Historical Decomposition*

Description

Plots the historical decomposition for multiple variables, showing the contribution of each structural shock as individual stacked areas, plus a total line overlay. Mirrors the MATLAB Bloom replication figure with facets per variable.

Usage

```
fPlotHistDec(
  histdec_list,
  shock,
  shockname,
  shocknames = NULL,
  dates = NULL,
  p = 0,
  return_data = FALSE,
  facet_ncol = 2,
  facet_scales = c("free", "free_y", "free_x", "fixed"),
  shock_fill = "#407EC9",
  other_fills = "#E87722",
  total_color = "#1a1a1a",
  total_linewidth = 0.8,
  area_alpha = 1
)
```

Arguments

`histdec_list`	Named list of (T-p) x N matrices, one per variable to plot. Each element is the `$histdec` output of `fHistDec()` called for that variable. Names are used as facet labels.
`shock`	Integer (1-indexed) selecting the primary shock of interest.
`shockname`	Character string label for the selected shock (legend entry).
`shocknames`	Optional character vector of length N with names for *all* shocks. When supplied the non-selected shocks are shown as individually named stacked areas instead of a single "Residual" bar. Default `NULL` (falls back to "Residual").
`dates`	Optional vector of dates (Date, numeric, or integer). Can be the full T-length vector (trimmed internally using `p`) or the already-trimmed (T-p)-length vector. Default `NULL` uses an integer index.
`p`	Integer lag order of the VAR. Used only when `dates` has length T to trim to `dates[(p+1):T]`. Default `0`.
`return_data`	Logical. If `TRUE` returns the long-format tibble instead of the plot. Default `FALSE`.

facet_ncol	Number of columns in the facet grid. Default 2.
facet_scales	Scale option for facet_wrap: one of "free" (default), "free_y", "free_x", or "fixed".
shock_fill	Fill color for the primary shock area (default: "#407EC9").
other_fills	Character vector of fill colors for the remaining shocks. Recycled if shorter than N - 1. Default "#E87722" (single color used for the "Residual" bar or cycled across named other shocks).
total_color	Color for the total line overlay (default: "#1a1a1a").
total_linewidth	Line width for the total line (default: 0.8).
area_alpha	Transparency for the stacked areas (default: 1).

Details

When shocknames is NULL all non-selected shocks are summed into a single "Residual" bar (original behaviour, suitable for the Bloom replication). When shocknames is supplied each shock gets its own named stacked area — useful for the BQ case where the second shock has a meaningful label (e.g. "Non-Technology").

Value

A ggplot object or a tibble (if return_data = TRUE) with columns date, variable, series, value.

See Also

fHistDec

Examples

```
set.seed(123)
y   <- matrix(rnorm(400), ncol = 2,
              dimnames = list(NULL, c("Output", "Prices")))
fit <- fVAR(y, p = 2, c = 1)

# One (T-p) x N contribution matrix per series, Cholesky-identified
K  <- t(chol(fit$sigma))
hd <- lapply(seq_len(ncol(y)),
             function(i) fHistDec(y, fit, K, series = i)$histdec)
names(hd) <- colnames(y)

fPlotHistDec(
    histdec_list = hd,
    shock        = 1,
    shockname    = "Supply",
    shocknames   = c("Supply", "Demand"),
    dates        = seq_len(nrow(y) - fit$p),
    p            = fit$p
)
```

fPlotIRF *Plot Impulse Responses*

Description

Draws impulse responses with nested confidence bands, one panel per variable. Several results can be passed at once; each gets its own colour, which makes it easy to compare identification schemes, bootstrap variants or specifications in one figure.

Usage

```
fPlotIRF(
  ...,
  bands = NULL,
  varnames = NULL,
  shocknames = NULL,
  cumulate = NULL,
  conf = NULL,
  shock = NULL,
  variables = NULL,
  scale = 1,
  colors = NULL,
  ribbon_fill = "#407EC9",
  line_color = "#910048",
  ribbon_alpha = NULL,
  zero_line_color = "#707372",
  facet_scales = "free",
  facet_ncol = NULL,
  return_data = FALSE
)
```

Arguments

`...`	One or more results. Argument names become the legend labels, e.g. `fPlotIRF(Standard = a, Corrected = b)`.
`bands`	Bootstrap result supplying the confidence bands for a point estimate, e.g. `fPlotIRF(point_irf, bands = boot_chol)`. Only meaningful with a single result: when several are compared, each one carries its own bands, so build them with `fIRF` first.
`varnames, shocknames`	Labels for the variables and shocks, applied to every result given. They are passed to `fIRF` when a result still has to be converted, and rename a result that `fIRF()` already built, so one set of labels can serve several results.
`cumulate, conf`	Passed to `fIRF` when a result still has to be converted. They change how the table is built, so they are refused for a result that `fIRF()` already built.
`shock`	Shocks to plot, by position or name (within each result). `NULL` plots every shock. With one shock per result the panels are indexed by variable; with several, by shock and variable.

variables	Variables to plot (names). NULL plots all.
scale	Scalar or vector named by variable, multiplying the point estimate and bands.
colors	Colours for several results (one per result). Defaults to the tidyMacro palette.
ribbon_fill, line_color	Band and line colours for a single result. ribbon_fill may also give one colour per band, outer first or named by band, e.g. c("Anderson-Rubin" = "red", "Delta method" = "blue").
ribbon_alpha	Band transparency: one value per band, outer first. A single value sets the outer band; inner bands get more opaque. The default is lighter when several results overlap.
zero_line_color	Colour of the dashed zero line.
facet_scales	Passed to ggplot2::facet_wrap().
facet_ncol	Number of panel columns. Defaults to a square-ish grid for one shock and one column per variable for several shocks.
return_data	If TRUE, return the fIRF tibble instead of the plot.

Details

Every argument in ... is converted with fIRF, and the fIRF() arguments below can be given directly, so plotting one result never needs a nested call: fPlotIRF(point, bands = boot, varnames = vn). Comparing several results is the one case that still needs fIRF(), because each result carries its own bands.

Value

A ggplot object, or an fIRF tibble when return_data = TRUE.

See Also

fIRF

Examples

```
set.seed(123)
y     <- matrix(rnorm(400), ncol = 2, dimnames = list(NULL, c("y1", "y2")))
fit   <- fVAR(y, p = 2, c = 1)
wold  <- fWoldIRF(fit, horizon = 12)
point <- fCholeskyIRF(wold, t(chol(fit$sigma)))
boot  <- fBootstrapChol(y, fit, nboot = 50, horizon = 12, n_threads = 2)

# A point estimate and its bands, in one call
fPlotIRF(point, bands = boot, varnames = colnames(y), shock = 1)

# Comparing results: each one carries its own bands, so build them first
corrected <- fBootstrapCholCorrected(y, fit, nboot1 = 50, nboot2 = 50,
                                     horizon = 12, n_threads = 2)
fPlotIRF(Standard    = fIRF(point, bands = boot),
```

```
  `Corrected` = fIRF(point, bands = corrected),
  varnames = colnames(y), shock = 1)
```

`fPlotLPDID` *Plot LP-DiD event-study estimates*

Description

Pass a single `fLPDID` result, or several named results directly — one facet per result: `fPlotLPDID(`Only FEs` = a, `With controls` = b)`.

Usage

```
fPlotLPDID(
  ...,
  title = NULL,
  xlab = "Event time",
  ylab = "Estimate",
  ncol = 2L,
  scales = "free_y",
  ci = c("ribbon", "bars")
)
```

Arguments

`...`	One or more `fLPDID` results (ideally named).
`title`, `xlab`, `ylab`	Plot labels.
`ncol`	Number of facet columns when several results are given.
`scales`	Facet scales: `"fixed"`, `"free"`, `"free_x"` or `"free_y"` (default).
`ci`	How to draw the confidence intervals: `"ribbon"` (shaded band, DGJT Figure-4 style; default) or `"bars"` (point estimates with capped error bars, coefplot / Figure-3 style).

Details

Horizons that could not be estimated (non-finite estimates) are dropped from the plot with a message, so no ggplot warnings are emitted.

Value

A ggplot object.

Examples

```
data("BankingDeregulation")

m <- fLPDID(lshare ~ bank, data = BankingDeregulation,
            panel_id = c("id", "year"), treat = "bank",
            post = 4, pre = 4, n_threads = 2)
fPlotLPDID(m)

# Several fits side by side, named by their arguments
m95 <- fLPDID(lshare ~ bank, data = BankingDeregulation,
              panel_id = c("id", "year"), treat = "bank",
              post = 4, pre = 4, conf = 95, n_threads = 2)
fPlotLPDID(`90%` = m, `95%` = m95, ci = "bars")
```

`fPlotNLLP` *Plot Nonlinear Local Projection Responses*

Description

Plots the two shock coefficients of one or more `fNLLP` fits, one colour per component, or their difference, with one panel per outcome. Fits with several confidence levels get one ribbon per level.

Usage

```
fPlotNLLP(
  ...,
  variables = NULL,
  what = c("components", "difference"),
  scale = 1,
  return_data = FALSE,
  colors = c("#910048", "#407EC9"),
  ribbon_alpha = 0.2,
  zero_line_color = "#707372",
  facet_scales = "free_y",
  facet_ncol = NULL
)
```

Arguments

`...`	One or more objects returned by `fNLLP`, with the same `type`, components and `conf`, and distinct outcomes.
`variables`	Optional character vector of outcomes to plot. Defaults to all outcomes of all fits.
`what`	`"components"` (default) plots both coefficients; `"difference"` plots the first minus the second with its band. Not available for `type = "cubic"`.
`scale`	Numeric scalar applied to responses and bands before plotting. Default 1.

return_data	Logical. If TRUE, return the plotting data instead of a ggplot object.
colors	Colours of the components, in order.
ribbon_alpha	Transparency of each confidence band.
zero_line_color	Color for the horizontal zero line.
facet_scales	Facet scales option: "free_y", "free", "free_x", or "fixed".
facet_ncol	Number of facet columns. Defaults to an automatic layout.

Value

A ggplot object, or with return_data = TRUE a tibble with variable, component, horizon, point, lower, upper (and conf for several levels).

See Also

fNLLP, tidy.fNLLP

Examples

```
data("TenreyroThwaites2016")
tt <- TenreyroThwaites2016
tt <- tt[tt$Date >= as.Date("1968-10-01"), ]
fit <- fNLLP(FFR ~ RRShock + l(FFR, 1), data = tt, horizons = 8,
             shock = "RRShock", type = "sign", balanced = TRUE,
             n_threads = 2)
fPlotNLLP(fit)
fPlotNLLP(fit, what = "difference")
```

---

fPlotVarDec *Plot Forecast Error Variance Decomposition*

Description

Creates area plots showing the forecast error variance decomposition (FEVD) for multiple variables across horizons.

Usage

```
fPlotVarDec(fevd, varnames, shocknames, return_df = FALSE)
```

Arguments

fevd	A numeric array containing forecast error variance shares. Either a 3-dimensional array (N x N x horizon) for multiple shocks, or a 2-dimensional matrix (N x horizon) for a single shock.
varnames	Character vector of variable names. Length should equal N (the number of variables).
shocknames	Character vector of shock names. For 3-dimensional fevd, length should equal N. For 2-dimensional fevd, should be a single name.
return_df	Logical. If TRUE, returns a data frame instead of a plot. Default is FALSE.

Details

The function creates stacked area charts showing how forecast error variance is decomposed across different shocks over the forecast horizon. Each variable is displayed in a separate facet panel.

For 2-dimensional input (single shock), the function automatically calculates and displays "Other Shocks" as the complement to ensure the decomposition sums to 1.

Value

If return_df = FALSE, returns a ggplot object showing the FEVD for each variable in separate facets. If return_df = TRUE, returns a data frame in long format with columns: horizon, variable, shock, and value.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 20)
chol_irf   <- fCholeskyIRF(wold, t(chol(var_result$sigma)))
fevd       <- fFEVDChol(chol_irf)$fevd

fPlotVarDec(fevd, varnames = c("y1", "y2"),
            shocknames = c("Shock 1", "Shock 2"))
```

---

fPolyConvolve *Polynomial Convolution of Two Matrix Lag Polynomials*

Description

Computes $C(L) = A(L) \cdot B(L)$ truncated to nlags terms. Both polynomials are stored as 3-D arrays where the third dimension indexes lag order (slice 1 = lag 0, slice 2 = lag 1, etc.).

Usage

```
fPolyConvolve(A, B, nlags)
```

Arguments

A	3-D array of dimension $d_1 \times d_2 \times A_3$ (e.g. Wold IRF coefficients from fWoldIRF).
B	3-D array of dimension $d_2 \times K \times B_3$ (e.g. $\Psi(L)$ coefficients).
nlags	Integer. Number of output slices to return (must be $\leq A_3 + B_3 - 1$).

Value

A 3-D array of dimension $d_1 \times K \times$ nlags.

See Also

fWoldIRF, fCholeskyIRF

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 12)

# Multiplying by the identity polynomial returns the Wold coefficients
I     <- array(diag(2), c(2, 2, 1))
out   <- fPolyConvolve(wold, I, nlags = 13)
all.equal(out, wold)
```

---

fSetTidyMacroTheme *Update ggplot2 Theme Defaults to tidyMacro Style*

Description

Sets the tidyMacro theme and color scales as the default for all subsequent ggplot2 plots in the current R session. This is convenient when creating multiple plots with consistent styling.

Usage

```
fSetTidyMacroTheme(base_size = 14)
```

Arguments

base_size	Numeric. Base font size for the theme. Default is 14.

Details

This function modifies the default ggplot2 theme and discrete color/fill scales for the current session. All plots created after calling this function will automatically use the tidyMacro styling unless explicitly overridden.

To revert to ggplot2 defaults, use ggplot2::theme_set(ggplot2::theme_gray()) and remove the scale defaults with ggplot2::update_geom_defaults().

Value

Invisibly returns NULL. Called for side effects.

See Also

fThemeTidyMacro, scale_color_tidyMacro, theme_set

Examples

```
library(ggplot2)
library(tidyMacro)

# Set tidyMacro as default theme for the session
fSetTidyMacroTheme()

# All subsequent plots will use tidyMacro styling
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point()

# Revert to ggplot2 defaults
theme_set(theme_gray())
```

---

fSignRestr *Sign, Narrative and Sign-Plus-IV Identification of a Structural VAR*

Description

Estimates a VAR, then identifies it by sign restrictions, optionally combined with Antolin-Diaz and Rubio-Ramirez narrative restrictions and with an external instrument that pins down the first structural impact column. Inference is Bayesian throughout: reduced-form parameters are drawn from their flat-prior Normal-inverse-Wishart posterior and each draw is paired with a Haar-uniform rotation.

Usage

```
fSignRestr(
  y,
  p,
  c = 1,
  sign,
  nsteps = 40,
  ndraws = 500,
  sr_hor = 1,
  sr_rot = 500,
  max_post_draws = 1000,
  conf = 90,
  inference = 1,
```

```
  narrative = NULL,
  dates = NULL,
  instrument = NULL,
  narr_weight_mc = 0,
  resid_from_draw = FALSE,
  store_draws = TRUE,
  exog = NULL,
  varnames = NULL,
  n_threads = 0,
  seed = 42,
  verbose = FALSE
)

## S3 method for class 'fSignRestr'
print(x, ...)
```

Arguments

y	A T x N numeric matrix (or data frame) of endogenous variables.
p	Integer lag order.
c	Integer intercept indicator: 1 to include an intercept (default), 0 to exclude it.
sign	An N x ds matrix of sign restrictions: +1 the response must be non-negative, -1 non-positive, 0 unrestricted. ds must equal N when no instrument is used, and N - k with a k-column instrument (k = ncol(instrument$Z)), since shocks 1:k are the instrumented ones.
nsteps	Integer number of horizons to report, impact included (default 40).
ndraws	Integer number of accepted draws (default 500).
sr_hor	Integer. Restrictions are imposed at horizons 0, ..., sr_hor - 1 (default 1, impact only).
sr_rot	Integer maximum rotations attempted per parameter draw (default 500).
max_post_draws	Integer maximum parameter draws per accepted draw (default 1000).
conf	Numeric coverage of the credible bands, in percent (default 90; Uhlig (2005) and Antolin-Diaz and Rubio-Ramirez (2018) report 68, so pass conf = 68 to match them). A vector requests several levels at once, e.g. c(68, 90): the first is reported in IRinf/IRsup and every level is returned in bands. Extra levels do not require storing the draws.
inference	Integer: 1 (default) draws reduced-form parameters from the posterior; 0 holds them at OLS, leaving only rotation uncertainty.
narrative	Optional list of narrative restrictions with components sign (a list of shock, period, sign) and dom (a list of shock, period, var). Periods may be integer rows of y or, when dates is supplied, date strings or Date values.
dates	Optional vector of length nrow(y) — character labels or Date values — used to resolve narrative periods given as dates rather than as row numbers.

`instrument` Optional list with component `Z`, a T x k instrument matrix aligned with the rows of `y`, missing values allowed. Selects the `sign+iv` scheme. With `k = 1` the single shock is identified as in Gertler-Karadi/Mertens-Ravn; with `k > 1` the k shocks are jointly identified, following the block generalisation in `VARiriv_B.m` (Cesa-Bianchi & Sokol 2022). Either way the first stage is estimated once at OLS and not re-estimated across draws, as in the VAR Toolbox, so the bands do not carry the instrument's own sampling uncertainty. The identified columns are then completed against each draw's covariance: the first keeps its direction but is rescaled, and with `k > 1` the later columns are Gram-Schmidt orthogonalised, which can change their direction as well as their length.

`narr_weight_mc` Integer. Positive values switch on the ADRR importance reweighting, using this many Monte Carlo replications per draw. `0` (default) uses plain rejection sampling, as the VAR Toolbox does. The weights correct for the narrative restrictions only. They target Algorithm 1 of Antolin-Diaz and Rubio-Ramirez (2018) only with `sr_rot = 1`, since keeping the first admissible of several rotations reweights the parameter draws; with `resid_from_draw = TRUE` when `inference = 1`, so the narrative shocks use each draw's own coefficients; and up to Monte Carlo error in the weights.

`resid_from_draw` Logical. `FALSE` (default) evaluates narrative restrictions on OLS residuals, matching the VAR Toolbox; `TRUE` uses the residuals implied by each parameter draw.

`store_draws` Logical; keep the full IRF and FEVD draw distributions (default TRUE).

`exog` Optional T x M matrix of exogenous regressors.

`varnames` Optional character vector of length N.

`n_threads` Integer; 0 (default) uses all cores minus one.

`seed` Integer base seed (default 42).

`verbose` Logical; print progress diagnostics (default FALSE).

`x` An object of class `"fSignRestr"`.

`...` Ignored.

Details

Impulse responses are indexed `[variable, shock, horizon]` and FEVD shares are on the `[0, 1]` scale.

A slot that exhausts `max_post_draws` parameter draws without an accepted rotation is counted in `n_failed` rather than silently dropped. When at most three columns are free, a linear-infeasibility certificate can prove that no rotation satisfies the impact signs of a shock; such a parameter draw is skipped without spending `sr_rot` rotations and counted in `n_ruled_out`. With `inference = 0` every draw shares the OLS covariance, so a certified specification stops with an error before any draw is made.

With a k-column instrument, the first column of `sign` corresponds to structural shock k + 1: shocks 1:k are the instrumented ones and are not sign-restricted.

Value

An object of class `"fSignRestr"`: a list with medians and credible bands for the IRFs (`IRmed`, `IRinf`, `IRsup`) and the FEVD (`VDmed`, `VDinf`, `VDsup`), each an N x N x nsteps array indexed `[variable, shock, horizon]`; the accepted impact matrices `Ball` and coefficient draws `beta_all`; the median impact matrix `Bmed` and the Fry-Pagan draw `Bfp` with its `IRfp`, `VDfp` and index `fp_index`; the importance `weights`; `accept_rate`, `ndraws_tried`, `n_tried`, `n_param_draws`, `n_ruled_out` and `n_failed`; `n_errors` and `first_error` for numerical failures; `bands`, one list of `IRinf`, `IRsup`, `VDinf` and `VDsup` per level in `conf`; when `store_draws = TRUE`, the draw distributions `IRall` and `VDall`, each an `N x N x nsteps x ndraws` array; the OLS fit `var`; and `varnames`, `ident`, `p`, `c`, `nsteps`, `conf`, `iv` and `narrative_active`.

Methods (by generic)

- `print(fSignRestr)`: Print a compact summary of the identified model.

References

Uhlig, H. (2005). *Journal of Monetary Economics*, 52(2), 381–419.

Antolin-Diaz, J., & Rubio-Ramirez, J. F. (2018). *American Economic Review*, 108(10), 2802–2829.

See Also

`fPlotIRF`, `fHDShock`

Examples

```
set.seed(123)
y <- matrix(rnorm(400), ncol = 2)

# Shock 1 raises both variables on impact; shock 2 moves them apart
sign <- matrix(c(1, 1, -1, 1), 2, 2)

# Few draws here to keep the example quick; use 500 or more in practice.
sr <- fSignRestr(y, p = 1, sign = sign, nsteps = 12, ndraws = 50,
                 n_threads = 2)
dim(sr$IRmed)
fPlotIRF(sr, shock = 1)
```

---

fSpectralFEVD *Spectral Forecast Error Variance Decomposition*

Description

Computes the fraction of spectral variance attributable to a single structural shock within a given frequency band.

Usage

```
fSpectralFEVD(D, irf_s, band, J, fourier = TRUE)
```

Arguments

D	Cholesky IRF cube of dimension $N \times N \times H$ (output of fCholeskyIRF).
irf_s	Matrix of dimension $N \times H$: IRF of each variable to the structural shock of interest (unit-variance normalisation).
band	Numeric vector of length 2: c(low_period, high_period) in the same time units as the data (e.g. months). Example: c(18, 96) for the business-cycle band.
J	Integer. Number of points on the frequency grid.
fourier	Logical. If TRUE (default), uses Fourier frequencies $(\pi/J, 2\pi/J, \ldots, \pi)$ and sums over those falling inside band. If FALSE, places J points uniformly across the band via linspace and averages over all of them.

Value

Numeric vector of length N: share of spectral variance explained by the shock in the specified band.

See Also

fCholeskyIRF, fPolyConvolve

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 40)
D          <- fCholeskyIRF(wold, t(chol(var_result$sigma)))

# Share of business-cycle variance (6 to 32 periods) due to shock 1
fSpectralFEVD(D, D[, 1, ], band = c(6, 32), J = 256)
```

---

fSVARSteps *Print SVAR Identification Steps*

Description

Prints a numbered guide to the proxy-SVAR identification and invertibility-testing procedure to the R console.

Usage

```
fSVARSteps()
```

Details

Step 1: Estimate the VAR(p) and extract the residuals.

Step 2: Regress the instrument on the current residuals plus f leads; save the fitted value.

Step 3: F-test whether the lead coefficients are jointly zero. Rejection implies the shock is non-invertible.

Step 4: If invertible, apply the standard Stock-Watson IV procedure to obtain IRFs and unit-variance shocks.

Step 5: If non-invertible, test recoverability via the Ljung-Box test on the fitted instrument series.

Step 6: If recoverable (null of no autocorrelation not rejected), repeat the IV procedure adding residual leads to the first-stage regression.

Step 7: If recoverability is rejected, SVAR analysis with this instrument is not feasible.

Value

Called for its side-effect of printing to the console. Returns `NULL` invisibly.

Examples

```
fSVARSteps()
```

---

fTestInvertibility *Invertibility Test for External Instrument SVAR*

Description

Tests whether the structural shock is invertible (fundamental) by regressing the cleaned instrument on current and r future VAR residuals and applying an F-test for the joint significance of the future residual coefficients (Null = invertible).

Usage

```
fTestInvertibility(z, eps, r)
```

Arguments

`z`	Numeric vector. Cleaned instrument of length $T - p$.
`eps`	$(T - p) \times N$ matrix of VAR residuals.
`r`	Integer. Number of future residual lags to include in the test.

Details

The test regresses the cleaned instrument z_t on current and r future VAR residuals:

$$z_t = c + \delta_0 \hat{\varepsilon}_t + \delta_1 \hat{\varepsilon}_{t+1} + \cdots + \delta_r \hat{\varepsilon}_{t+r} + u_t.$$

The F-test checks $\delta_1 = \cdots = \delta_r = 0$. Rejection implies the shock is non-invertible and the generalised SVAR-IV procedure of Forni, Gambetti & Ricco (2022) should be applied. The returned `delta`, `fit`, and `vt` are needed as inputs to `fBootstrapIVRecover`.

Value

A list with:

F_stat F statistic for the joint significance of future residuals.

pval p-value. Small values reject invertibility.

instr_relev Correlation between fitted and actual instrument.

delta Coefficient vector from regressing z on $[1, \varepsilon_t, \varepsilon_{t+1}, \ldots, \varepsilon_{t+r}]$. Length $N(r+1)+1$ (intercept first).

fit Fitted instrument values $\hat{\delta}(F)\hat{\varepsilon}_t$. Length $T - p - r$.

vt Noise residuals $z - \hat{z}$. Same length as `fit`.

References

Forni, M., Gambetti, L., & Ricco, G. (2022). Recovering the structural shock from non-invertible structural VARs. *Review of Economics and Statistics*.

See Also

`fTestRecoverability`, `fBootstrapIVRecover`

Examples

```
set.seed(123)
X   <- matrix(rnorm(600), ncol = 3)
fit <- fVAR(X, p = 2, c = 1)

# An instrument correlated with the first structural shock
u   <- fit$residuals
z   <- u[, 1] + rnorm(nrow(u), sd = 0.5)
inv <- fTestInvertibility(z, u, r = 3)
cat(sprintf("F = %.3f  p = %.4f\n", inv$F_stat, inv$pval))
```

`fTestRecoverability` *Recoverability Test for External Instrument SVAR*

Description

Tests whether a non-invertible structural shock is recoverable by applying a Ljung-Box autocorrelation test to the fitted instrument from the invertibility regression (Null = recoverable, i.e. no autocorrelation).

Usage

```
fTestRecoverability(fit, lags = 2L)
```

Arguments

`fit`	Numeric vector of fitted instrument values returned by `fTestInvertibility` (`$fit`).
`lags`	Integer. Number of lags for the Ljung-Box test. Default `2L`.

Details

Implements the recoverability test of Forni, Gambetti & Ricco (2022). After rejecting invertibility via `fTestInvertibility`, the fitted instrument $\hat{z}_t = \hat{\delta}(F)\hat{\varepsilon}_t$ is tested for autocorrelation. Under recoverability the fitted series has no serial correlation (it is a white noise projection); under non-recoverability it is autocorrelated. Failure to reject the null means absolute IRFs, historical decompositions, and spectral FEVDs can all be recovered via the generalised SVAR-IV procedure.

Value

A list with:

statistic Ljung-Box Q statistic.

pval p-value. Large values fail to reject recoverability.

lags Number of lags used.

References

Forni, M., Gambetti, L., & Ricco, G. (2022). Recovering the structural shock from non-invertible structural VARs. *Review of Economics and Statistics*.

See Also

`fTestInvertibility`, `fBootstrapIVRecover`

Examples

```
set.seed(123)
X   <- matrix(rnorm(600), ncol = 3)
fit <- fVAR(X, p = 2, c = 1)
u   <- fit$residuals
z   <- u[, 1] + rnorm(nrow(u), sd = 0.5)

inv <- fTestInvertibility(z, u, r = 3)
rec <- fTestRecoverability(inv$fit, lags = 2)
cat(sprintf("Ljung-Box p-value (lag %d): %.4f\n", rec$lags, rec$pval))
```

---

fThemeTidyMacro *Custom ggplot2 Theme for tidyMacro Package*

Description

A customized ggplot2 theme based on `theme_bw()` with Palatino font family, adjusted grid lines, and centered plot elements. Designed for professional publications and presentations of VAR model results.

Usage

```
fThemeTidyMacro(base_size = 14)
```

Arguments

`base_size`	Numeric. Base font size in points. Default is 14. All other text sizes are scaled relative to this value. The plot title will be 1.5 times the base size.

Value

A ggplot2 theme object that can be added to ggplot2 plots.

The theme inherits from `theme_bw()` and modifies specific elements to create a consistent visual style suitable for economic time series and impulse response function plots.

Note

The Palatino font family must be available on your system for proper rendering. If Palatino is not available, ggplot2 will fall back to the default font family. On most systems, you can check available fonts using `extrafont::fonts()` after loading the extrafont package.

See Also

`theme_bw`, `theme`, `scale_color_tidyMacro`, `fSetTidyMacroTheme`

Examples

```
library(ggplot2)

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(title = "Miles per gallon vs weight") +
  fThemeTidyMacro()

ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point(size = 3) +
  labs(x = "Weight (1000 lbs)", y = "Miles per gallon") +
  scale_color_tidyMacro() +
  fThemeTidyMacro(base_size = 16)
```

---

fUhligIRF *Uhlig Maximum-Share Impulse Responses*

Description

Computes impulse responses using the direction returned by `fUhligMaxShare`. The sign is chosen so the first variable's last-horizon response is non-negative.

Usage

```
fUhligIRF(wold, S, idx)
```

Arguments

`wold`	N x N x (horizon+1) Wold impulse response array, including horizon zero in the first slice, as returned by `fWoldIRF`.
`S`	N x N lower triangular Cholesky factor of the VAR residual covariance matrix. At least two variables are required.
`idx`	Index (1-based) of the variable whose forecast error variance contribution is maximised.

Value

An N x (horizon+1) matrix of structural impulse responses.

See Also

`fBootstrapUhlig`

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 12)

# Shock explaining the largest forecast-error-variance share of variable 1
irf <- fUhligIRF(wold, t(chol(var_result$sigma)), idx = 1)
dim(irf)
```

`fUhligMaxShare` *Uhlig Maximum-Share Shock Direction*

Description

Finds the unit-length shock direction with a zero first coordinate that maximises the selected variable's forecast error variance contribution, summed over the supplied horizons. For H slices, horizon h receives weight H-h, with h starting at zero. The sign of the direction is not normalised; `fUhligIRF` applies the IRF sign convention.

Usage

```
fUhligMaxShare(wold, S, idx)
```

Arguments

`wold`	N x N x (horizon+1) Wold impulse response array, including horizon zero in the first slice, as returned by `fWoldIRF`.
`S`	N x N lower triangular Cholesky factor of the VAR residual covariance matrix. At least two variables are required.
`idx`	Index (1-based) of the variable whose forecast error variance contribution is maximised.

Value

An N x 1 matrix containing the shock direction, with first element zero.

Examples

```
set.seed(123)
y          <- matrix(rnorm(400), ncol = 2)
var_result <- fVAR(y, p = 2, c = 1)
wold       <- fWoldIRF(var_result, horizon = 12)

q <- fUhligMaxShare(wold, t(chol(var_result$sigma)), idx = 1)
sum(q^2)   # a unit-length rotation vector
```

fVAR *Vector Autoregression (VAR) Model Estimation*

Description

Estimates a Vector Autoregression model with optional exogenous variables using ordinary least squares (OLS). Optimised for maximum speed and memory efficiency via RcppArmadillo.

Usage

```
fVAR(y, p, c, exog = NULL)
```

Arguments

y	A numeric matrix of time series data with T rows (time periods) and N columns (endogenous variables).
p	An integer specifying the number of lags to include in the VAR.
c	An integer indicator for including a constant term (1 = include intercept, 0 = no intercept).
exog	An optional T x M matrix of exogenous variables. If provided, these variables enter contemporaneously (not lagged). Default is NULL.

Details

The model is

$$Y_t = c + A_1 Y_{t-1} + \cdots + A_p Y_{t-p} + B X_t + e_t$$

where X_t are contemporaneous exogenous variables. Without exogenous variables the model reduces to a standard VAR(p). Estimation uses `arma::solve` for numerical stability.

Value

A list with elements: beta — coefficient matrix of dimensions (Np + c + M) x N, where the first row is the intercept (if c = 1), followed by N*p lag-coefficient rows, then M exogenous-coefficient rows; residuals — (T-p) x N matrix of OLS residuals; sigma — N x N residual covariance matrix with degrees-of-freedom correction: denominator is (T - p) - c - N*p - n_exog, matching Bloom (2009) and Kaenzig (2021); p — lag order (echoed from input); c — intercept indicator (echoed from input); n_exog — number of exogenous variables (0 if none provided).

See Also

fWoldIRF, fBootstrapChol, fAICBIC

Examples

```
set.seed(123)
y      <- matrix(rnorm(400), ncol = 2)
result <- fVAR(y, p = 2, c = 1)
result$beta

# With one exogenous regressor
exog   <- matrix(rnorm(200), ncol = 1)
result <- fVAR(y, p = 2, c = 1, exog = exog)
dim(result$residuals)
```

---

fVARX *Vector Autoregression with Exogenous Variables (VARX) Model Estimation*

Description

Estimates a VARX(p) model where both endogenous and exogenous variables are lagged using ordinary least squares (OLS). Optimised for maximum speed and memory efficiency.

Usage

```
fVARX(y, ex, p, c)
```

Arguments

y	A numeric matrix of endogenous time series data with T rows (time periods) and N columns (variables).
ex	A numeric matrix of exogenous time series data with T rows (time periods) and M columns (variables).
p	An integer specifying the number of lags for both endogenous and exogenous variables.
c	An integer indicator for including a constant term (1 = include intercept, 0 = no intercept).

Details

The VARX(p) model is:

$$Y_t = c + A_1 Y_{t-1} + \cdots + A_p Y_{t-p} + B_0 X_t + B_1 X_{t-1} + \cdots + B_p X_{t-p} + e_t$$

Both endogenous and exogenous variables are lagged p periods. Exogenous variables also enter contemporaneously at time t.

Value

A list with two elements: beta — coefficient matrix of dimensions (Np + M + Mp + c) x N; residuals — (T-p) x N residual matrix.

Examples

```
set.seed(123)
y  <- matrix(rnorm(400), ncol = 2)
ex <- matrix(rnorm(200), ncol = 1)
result <- fVARX(y, ex, p = 2, c = 1)
dim(result$beta)
```

`fWoldIRF` *Compute Wold Impulse Response Functions for VAR Model*

Description

Compute Wold Impulse Response Functions for VAR Model

Usage

```
fWoldIRF(fVAR, horizon)
```

Arguments

`fVAR`	A list containing VAR estimation results with elements: • beta: Coefficient matrix • c: Integer indicator for intercept (1 if intercept, 0 otherwise) • p: Integer lag order • n_exog: Number of exogenous variables (optional)
`horizon`	Integer, the IRF horizon (number of periods ahead)

Details

This function computes the Wold (reduced-form) impulse response functions for a VAR(p) model with optional exogenous variables by calculating powers of the companion matrix. The Wold IRF shows the effect of a one-unit shock to each variable on all variables in the system over time, without imposing any identifying restrictions.

Exogenous variables do not contribute to the dynamic propagation of shocks and are therefore excluded from the companion matrix representation.

The IRFs are computed as:

$$\Phi_h = A^h[1:N, 1:N]$$

where A is the companion matrix and h is the horizon.

Value

A 3D array (cube) of Wold impulse response functions with dimensions N x N x (horizon+1), where irfwold[,,h] contains the response at horizon h

Examples

```
set.seed(123)
y   <- matrix(rnorm(400), ncol = 2)
VAR <- fVAR(y, p = 2, c = 1)
irf <- fWoldIRF(VAR, horizon = 10)
dim(irf)   # 2 x 2 x 11
```

`Gali1999` *Galí (1999) Data*

Description

Quarterly US data on labour productivity and hours used to replicate Galí (1999). Both series enter the VAR in first differences (growth rates).

Usage

```
Gali1999
```

Format

A data frame with 183 rows and 3 columns:

Date Quarter end date (`Date`), 1948 Q1 – 1994 Q4

Productivity Growth rate of labour productivity (GDP per hour, all businesses), in percent

Hours Growth rate of total employee hours in non-agricultural establishments, in percent

Details

Data sourced from Fernald's quarterly TFP series (https://www.johnfernald.net/TFP). Productivity is defined as output per hour for all businesses; hours are total employee hours in non-agricultural establishments.

The identifying assumption (Blanchard-Quah long-run restriction) is that only technology shocks can have a permanent effect on the level of labour productivity.

References

Galí, J. (1999). Technology, employment, and the business cycle: Do technology shocks explain aggregate fluctuations? *American Economic Review*, 89(1), 249–271.

Examples

```
data("Gali1999")
head(Gali1999)
y <- Gali1999 |> dplyr::select(Productivity, Hours) |> as.matrix()
```

GK2015 *Gertler & Karadi (2015) Data, Forni-Gambetti-Ricco Sample*

Description

Monthly US data and the Gertler-Karadi high-frequency monetary policy surprise, restricted to the January 1990 - June 2012 window over which the instrument is observed. This is the sample used by Forni, Gambetti and Ricco (2024) to test invertibility and recoverability.

Usage

```
GK2015
```

Format

A data frame with 270 rows and 5 columns:

Date Month start date (`Date`), 1990-01 - 2012-06

BY1 One-year Treasury rate, in percent

CPI_Inflation Monthly CPI inflation, in percent

IP_Growth Monthly industrial production growth, in percent

instr Three-month-ahead fed funds futures surprise (FF4) in a 30-minute window around FOMC announcements, in percentage points

Details

Distinct from `GK2015_2`, which starts in 1979m7, keeps the price and quantity series in log levels, adds the excess bond premium, and leaves `FF4` as `NA` before 1990. Use this one for the invertibility and recoverability tests, which need a balanced sample of macro data and instrument; use `GK2015_2` for the proxy-SVAR, which estimates the reduced form on the longer sample.

The 49 zero values in `instr` are genuine - months without an FOMC announcement surprise - not missing observations.

References

Gertler, M., & Karadi, P. (2015). Monetary Policy Surprises, Credit Costs, and Economic Activity. *American Economic Journal: Macroeconomics*, 7(1), 44-76. doi:10.1257/mac.20130329

Forni, M., Gambetti, L., & Ricco, G. (2024). External Instrument SVAR Analysis for Noninvertible Shocks. *Journal of Applied Econometrics*, 39(7), 1173-1193. doi:10.1002/jae.3072

See Also

`fTestInvertibility`, `fTestRecoverability`, `GK2015_2`

Examples

```
data("GK2015")

X     <- GK2015 |> dplyr::select(BY1, CPI_Inflation, IP_Growth) |> as.matrix()
instr <- GK2015$instr
```

GK2015_2 *Gertler & Karadi (2015) Data with External Instrument*

Description

Monthly US data and the high-frequency monetary policy surprise used to replicate the proxy-SVAR of Gertler and Karadi (2015), covering July 1979 - June 2012.

Usage

```
GK2015_2
```

Format

A data frame with 396 rows and 6 columns:

Date Month start date (`Date`), 1979-07 - 2012-06

Consumer Price Index Log CPI

Industrial Production Log industrial production

1-year T-Bill One-year Treasury rate, in percent

EBP Excess bond premium of Gilchrist and Zakrajsek (2012), in percentage points

FF4 Three-month-ahead fed funds futures surprise in a 30-minute window around FOMC announcements, cumulated over the month, in percentage points. `NA` before January 1990, where the futures series does not exist.

Details

This is a different object from `GK2015`, which holds a four-variable system on the shorter 1990m1 sample and carries no instrument. `GK2015_2` adds the `FF4` proxy and starts in 1979m7, so the VAR is estimated on the full sample while the first stage uses only the 270 months where the instrument is observed.

The columns are stored in source order. `fSignRestr` instruments the residual of the *first* endogenous variable, so the policy rate must be reordered to the front before estimation - see the example. Note that 49 of the 270 observed `FF4` values are genuine zeros (months with no FOMC announcement surprise), not missing data.

Data taken from the replication files of the VAR Toolbox of Ambrogio Cesa-Bianchi (https://github.com/ambropo/VAR-Toolbox), where missing instrument values are flagged with the sentinel `123456789`; they are stored here as `NA`.

References

Gertler, M., & Karadi, P. (2015). Monetary Policy Surprises, Credit Costs, and Economic Activity. *American Economic Journal: Macroeconomics*, 7(1), 44-76. doi:10.1257/mac.20130329

Gilchrist, S., & Zakrajsek, E. (2012). Credit Spreads and Business Cycle Fluctuations. *American Economic Review*, 102(4), 1692-1720. doi:10.1257/aer.102.4.1692

See Also

fSignRestr, GK2015

Examples

```
data("GK2015_2")

# Policy rate first: the estimator instruments the first variable.
endo <- GK2015_2 |>
  dplyr::select(`1-year T-Bill`, `Consumer Price Index`,
                `Industrial Production`, EBP) |>
  as.matrix()

iv <- GK2015_2 |> dplyr::select(FF4) |> as.matrix()

# Shock 1 is instrument-identified, so `sign` has k - 1 columns.
SIGN <- matrix(0, 4, 3)
SIGN[, 1] <- c(0, -1, -1, 0)

fit <- fSignRestr(endo, p = 12, c = 1, sign = SIGN, nsteps = 48,
                  ndraws = 100, sr_hor = 3, instrument = list(Z = iv),
                  seed = 42, n_threads = 2)
```

JordaTaylor2025 *Jorda and Taylor (2025) Quarterly Local Projection Data*

Description

Quarterly US macroeconomic data and a Romer-Romer monetary policy shock from the local projection replication files of Jorda and Taylor.

Usage

```
JordaTaylor2025
```

Format

A data frame with 224 rows and 8 columns:

Date Quarter as a character string, 1960Q1 through 2015Q4

RRShock Romer-Romer monetary policy shock; missing outside its sample

logCPI Log Consumer Price Index, not multiplied by 100

dLogCPI Annualised quarterly log CPI growth, in percent

STIR Short-term interest rate, in percent

dSTIR Quarterly change in the short-term rate, in percentage points

lGDP Log real GDP, not multiplied by 100

dLogGDP Annualised quarterly log real GDP growth, in percent

See Also

fLP, JordaTaylor2025IV

Examples

```
data("JordaTaylor2025")
head(JordaTaylor2025)
```

JordaTaylor2025IV *Jorda and Taylor (2025) Monthly Instrumental Variable Data*

Description

Monthly US data for local projections with external monetary policy instruments, from the local projection replication files of Jorda and Taylor.

Usage

```
JordaTaylor2025IV
```

Format

A data frame with 517 rows and 6 columns:

Date Month start as a UTC POSIXct, 1965-12 through 2008-12

Unemployment Unemployment rate, in percent

Inflation PCE price inflation series from the replication data

FFRates Federal funds rate, in percent

RRShock Updated Romer-Romer monetary policy shock

KuttnerShock Kuttner monetary policy surprise instrument

Details

Missing instrument observations are stored as NA.

See Also

fLPIV, JordaTaylor2025

Examples

```
data("JordaTaylor2025IV")
head(JordaTaylor2025IV)
```

`Kaenzig2021` *Kaenzig (2021) Data*

Description

Monthly data on global oil markets and US macroeconomic aggregates used to replicate the OPEC production cut shock identification of Kaenzig (2021). Includes a high-frequency external instrument based on oil futures price changes around OPEC announcements.

Usage

```
Kaenzig2021
```

Format

A data frame with rows and 8 columns:

Date Month start date (`Date`)

Oil_Price Log real oil price, in percent

World_Oil_Prod Log world oil production, in percent

World_Oil_Inven Log world oil inventories, in percent

World_IP Log world industrial production, in percent

US_IP Log US industrial production, in percent

US_CPI Log US Consumer Price Index, in percent

iv_kanzig_final High-frequency external instrument: oil futures price change in a narrow window around OPEC announcements. Contains `NA` outside announcement dates.

Details

The structural shock is identified using the external instrument `iv_kanzig_final` via the proxy-SVAR / IV approach. The instrument isolates exogenous variation in oil supply caused by OPEC production decisions.

References

Kaenzig, D. A. (2021). The macroeconomic effects of oil supply news: Evidence from OPEC announcements. *American Economic Review*, 111(4), 1092–1125.

Examples

```
data("Kaenzig2021")
head(Kaenzig2021)
y <- Kaenzig2021 |>
  dplyr::select(Oil_Price, World_Oil_Prod, World_Oil_Inven,
                World_IP, US_IP, US_CPI) |>
  as.matrix()
```

MianSufiVerner2017 *Mian, Sufi and Verner (2017) Data*

Description

Annual country panel from the replication files of Mian, Sufi and Verner (2017), restricted to the GDP and debt series behind Table II and the local projections of Figure 2 of the paper. Used with `fLPPanel`.

Usage

```
MianSufiVerner2017
```

Format

A tibble with 2,254 rows (30 countries, 1940-2015) and 10 columns:

country Country name, as in the replication files

year Year

lnGDP Log real GDP, constant prices, local currency units

lnGDP_d3 Three-year growth of real GDP, $100\times$ (`lnGDP` in t minus `lnGDP` in $t-3$)

HHD_L1GDP Nominal household debt in t over nominal GDP in $t-1$, in percent

NFD_L1GDP Nominal nonfinancial firm debt in t over nominal GDP in $t-1$, in percent

L1D3HHD_GDP Change in the household debt to GDP ratio from $t-4$ to $t-1$, in percentage points

L1D3NFD_GDP Change in the nonfinancial firm debt to GDP ratio from $t-4$ to $t-1$, in percentage points

mainsmp1 1 for the rows of the paper's main regression sample (1962-2012), `NA` otherwise

c_mainsmp 1 on every row. After a merge with `MianSufiVerner2018` it marks the rows that come from this file, the `if c_mainsmp == 1` sample of the 2018 update

Details

Missing observations are stored as `NA`. Table II of the paper regresses `lnGDP_d3` led $k=-1,\dots,5$ years on `L1D3HHD_GDP` and `L1D3NFD_GDP` with country fixed effects and standard errors clustered on country and year. Figure 2 uses `lnGDP`, `HHD_L1GDP` and `NFD_L1GDP` with five lags. Built by `inst/MianVerner2018/make_MianSufiVerner.R` from `masterdata_annual.dta`.

References

Mian, A., Sufi, A., & Verner, E. (2017). Household Debt and Business Cycles Worldwide. *The Quarterly Journal of Economics*, 132(4), 1755-1817. doi:10.1093/qje/qjx017

See Also

fLPPanel, MianSufiVerner2018

Examples

```
data("MianSufiVerner2017")
d <- MianSufiVerner2017

# Table II: the regressand is growth ending last year, so horizon h is
# column k = h - 1 of the table.
key <- paste(d$country, d$year)
d$L1yd3 <- d$lnGDP_d3[match(paste(d$country, d$year - 1), key)]

fit <- fLPPanel(L1yd3 ~ L1D3HHD_GDP + L1D3NFD_GDP | country, data = d,
                panel_id = c("country", "year"),
                shock = c("L1D3HHD_GDP", "L1D3NFD_GDP"), horizons = 6,
                cluster = ~country + year, n_threads = 2)
round(fit$irfs, 3)
```

MianSufiVerner2018 *Mian, Sufi and Verner (2018 Update) Global Debt Data*

Description

Annual extract of the IMF Global Debt Database (vintage of 30 April 2018) that Mian, Sufi and Verner merged with MianSufiVerner2017 to re-estimate Table II of their 2017 paper on a much larger set of countries.

Usage

```
MianSufiVerner2018
```

Format

A tibble with 12,730 rows (190 countries, 1950-2016) and 5 columns:

country Country name, as in the IMF Global Debt Database

year Year

rgdp Real GDP from the World Development Indicators, billions of constant local currency units

hh_ls Household debt, loans and debt securities, percent of GDP

nfc_ls Nonfinancial corporate debt, loans and debt securities, percent of GDP

Details

Missing observations are stored as NA. To merge with MianSufiVerner2017, rename "Hong Kong" to "Hong Kong SAR" and "Korea, Rep." to "Korea, Republic of" there, as the authors' do file does. Built by inst/MianVerner2018/make_MianSufiVerner.R from GlobalDebtDatabase_v043018merge.dta.

References

Mian, A., Sufi, A., & Verner, E. (2017). Household Debt and Business Cycles Worldwide. *The Quarterly Journal of Economics*, 132(4), 1755-1817. doi:10.1093/qje/qjx017

Mbaye, S., Moreno Badia, M., & Chae, K. (2018). Global Debt Database: Methodology and Sources. *IMF Working Paper* 18/111.

See Also

fLPPanel, MianSufiVerner2017

Examples

```
data("MianSufiVerner2018")
head(MianSufiVerner2018[!is.na(MianSufiVerner2018$hh_ls), ])
```

---

predict.fNLLP *Responses to Shocks of Given Sizes from a Nonlinear Local Projection*

Description

Combines the two shock coefficients of an fNLLP fit into the response to a shock of size d: $\beta^+ \max(d,0) + \beta^- \min(d,0)$ for type = "sign", $\beta_1 d + \beta_3 d^3$ for type = "cubic", and $d(F\beta_1 + (1-F)\beta_2)$ at state value F for type = "state". Standard errors use the covariance of the two coefficients.

Usage

```
## S3 method for class 'fNLLP'
predict(object, size = 1, state = NULL, conf = NULL, ...)
```

Arguments

object	An object returned by fNLLP.
size	Numeric vector of shock sizes. Default 1.
state	Numeric vector of regime weights in $[0,1]$. Required for type = "state", not allowed otherwise; every combination with size is evaluated.
conf	Confidence levels in percent; defaults to those of the fit.
...	Not used.

Value

A data frame with `horizon`, `lhs`, `shock`, `size`, `state` (for `type = "state"`), `estimate`, `se`, and `lower`/`upper` (or `lower_<level>`/`upper_<level>` for several levels).

See Also

`fNLLP`

Examples

```
data("TenreyroThwaites2016")
tt <- TenreyroThwaites2016
tt <- tt[tt$Date >= as.Date("1968-10-01"), ]
fit <- fNLLP(FFR ~ RRShock + l(FFR, 1), data = tt, horizons = 8,
             shock = "RRShock", type = "cubic", balanced = TRUE,
             n_threads = 2)
# A small and a large shock
head(predict(fit, size = c(0.25, 2)))
```

---

scale_color_tidyMacro *tidyMacro Color Scale for ggplot2*

Description

Apply the tidyMacro color palette to the color aesthetic in ggplot2 plots. This scale is particularly useful for line plots, impulse response functions, and other visualizations where multiple series need distinct colors.

Usage

```
scale_color_tidyMacro(
  palette = tidyMacro_colors,
  discrete = TRUE,
  reverse = FALSE,
  ...
)

scale_colour_tidyMacro(
  palette = tidyMacro_colors,
  discrete = TRUE,
  reverse = FALSE,
  ...
)
```

Arguments

palette	Character vector of hex colors. Default is tidyMacro_colors.
discrete	Logical. If TRUE (default), creates a discrete scale.
reverse	Logical. If TRUE, reverses the order of colors. Default is FALSE.
...	Additional arguments passed to discrete_scale.

Details

This function creates a discrete color scale using the tidyMacro palette. If the number of groups exceeds the 15 available colors, the palette will be recycled with a warning.

Value

A ggplot2 color scale that can be added to a plot.

See Also

scale_fill_tidyMacro, tidyMacro_colors, fThemeTidyMacro

Examples

```
library(ggplot2)

# Use the tidyMacro theme for every plot in the session
old <- ggplot2::theme_get()
fSetTidyMacroTheme()
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) + geom_point()

ggplot2::theme_set(old)   # restore
```

---

scale_fill_tidyMacro *tidyMacro Fill Scale for ggplot2*

Description

Apply the tidyMacro color palette to the fill aesthetic in ggplot2 plots. This scale is particularly useful for bar charts, area plots, and confidence bands in impulse response visualizations.

Usage

```
scale_fill_tidyMacro(
  palette = tidyMacro_colors,
  discrete = TRUE,
  reverse = FALSE,
  ...
)
```

Arguments

palette	Character vector of hex colors. Default is tidyMacro_colors.
discrete	Logical. If TRUE (default), creates a discrete scale.
reverse	Logical. If TRUE, reverses the order of colors. Default is FALSE.
...	Additional arguments passed to discrete_scale.

Details

This function creates a discrete fill scale using the tidyMacro palette. If the number of groups exceeds the 15 available colors, the palette will be recycled with a warning.

Value

A ggplot2 fill scale that can be added to a plot.

See Also

scale_color_tidyMacro, tidyMacro_colors, fThemeTidyMacro

Examples

```
library(ggplot2)

ggplot(mtcars, aes(x = factor(cyl), fill = factor(gear))) +
  geom_bar(position = "stack") +
  labs(x = "Cylinders", y = "Count", fill = "Gears") +
  scale_fill_tidyMacro() +
  fThemeTidyMacro()

ggplot(mtcars, aes(x = factor(cyl), y = mpg, fill = factor(gear))) +
  geom_col(position = "dodge") +
  scale_fill_tidyMacro(reverse = TRUE) +
  fThemeTidyMacro()
```

TenreyroThwaites2016 *Tenreyro and Thwaites (2016) Data*

Description

Quarterly US data from the replication workbook of Tenreyro and Thwaites (2016), used to estimate state-dependent and nonlinear responses to monetary policy shocks with fNLLP. Series are in their original units.

Usage

```
TenreyroThwaites2016
```

Format

A tibble with 254 rows and 6 columns:

Date Quarter start, 1947-01-01 through 2010-04-01

logGDP Log real GDP

Inflation PCE inflation at an annual rate, as a fraction (from 1947Q2)

FFR Federal funds rate, in percent (from 1954Q3)

RRShock Romer-Romer monetary policy shock, the residual of the linear reaction function summed within each quarter (1969Q1-2002Q4)

RRShockNL The same shock from the state-dependent reaction function of Tenreyro and Thwaites (1969Q1-2002Q4)

Details

Missing observations are stored as `NA`. The paper's text and the workbook use the PCE deflator, although its figure notes say "GDP deflator".

References

Tenreyro, S., & Thwaites, G. (2016). Pushing on a String: US Monetary Policy Is Less Powerful in Recessions. *American Economic Journal: Macroeconomics*, 8(4), 43-74. doi:10.1257/mac.20150016

Romer, C. D., & Romer, D. H. (2004). A New Measure of Monetary Shocks: Derivation and Implications. *American Economic Review*, 94(4), 1055-1084. doi:10.1257/0002828042002651

See Also

`fNLLP`

Examples

```
data("TenreyroThwaites2016")
head(TenreyroThwaites2016[!is.na(TenreyroThwaites2016$RRShock), ])
```

`tidy.fLP` *Tidy Local Projection Estimates*

Description

Tidy Local Projection Estimates

Usage

```
tidy.fLP(x, ...)
```

Arguments

x	An object returned by fLP or fLPIV.
...	Reserved for additional arguments.

Value

A data frame with horizon, response variable, shock, estimate, standard error, and confidence bounds. Multiple confidence levels use columns named lower_<level> and upper_<level>.

Examples

```
data("JordaTaylor2025")

m <- fLP(dLogCPI ~ RRShock + l(dLogCPI, 1:3),
         data = JordaTaylor2025, horizons = 6, shock = "RRShock",
         n_threads = 2)
head(tidy.fLP(m))
```

---

tidy.fLPPanel *Tidy Panel Local Projection Estimates*

Description

Tidy Panel Local Projection Estimates

Usage

```
tidy.fLPPanel(x, ...)
```

Arguments

x	An object returned by fLPPanel.
...	Reserved for additional arguments.

Value

A data frame with horizon, shock, estimate, standard error, degrees of freedom, p-value, number of observations, and confidence bounds. Multiple confidence levels use columns named lower_<level> and upper_<level>.

Examples

```
data("DemocracyGrowth")

m <- fLPPanel(lgdp ~ dem | id + year,
              data = DemocracyGrowth, panel_id = c("id", "year"),
              shock = "dem", horizons = 4, cluster = ~id, n_threads = 2)
head(tidy.fLPPanel(m))
```

`tidy.fNLLP` *Tidy Nonlinear Local Projection Estimates*

Description

Tidy Nonlinear Local Projection Estimates

Usage

```
tidy.fNLLP(x, difference = TRUE, ...)
```

Arguments

`x`	An object returned by `fNLLP`.
`difference`	Logical. Add rows for the difference between the two components (`component = "difference"`). Ignored for `type = "cubic"`. Default `TRUE`.
`...`	Reserved for additional arguments.

Value

A data frame with `horizon`, `lhs`, `shock`, `component` (a factor), `estimate`, `se`, and normal confidence bounds `lower`, `upper`. Multiple confidence levels use columns named `lower_<level>` and `upper_<level>`.

See Also

`fNLLP`, `fPlotNLLP`

Examples

```
data("TenreyroThwaites2016")
tt <- TenreyroThwaites2016
tt <- tt[tt$Date >= as.Date("1968-10-01"), ]
fit <- fNLLP(FFR ~ RRShock + l(FFR, 1), data = tt, horizons = 8,
             shock = "RRShock", type = "sign", balanced = TRUE,
             n_threads = 2)
head(tidy.fNLLP(fit))
```

`tidyMacro_colors` *tidyMacro Color Palette*

Description

Official color palette for the tidyMacro package, designed for VAR model visualizations including impulse response functions, forecast error variance decompositions, and other time series plots.

Usage

```
tidyMacro_colors
```

Format

A character vector of 15 hex colour codes.

Details

The palette consists of 15 distinct colors optimized for clarity and professional presentation:

- Crimson (#910048)
- Light Gray (#E5E5E5)
- Blue (#407EC9)
- Orange (#FF8200)
- Gray (#707372)
- Cyan (#00B0B9)
- Navy (#001E60)
- Lime (#78BE20)
- Purple (#8031A7)
- Olive (#658D1B)
- Gold (#F2A900)
- Dark Orange (#E35205)
- Red (#DA291C)
- Black (#231F20)
- Light Blue (#009CDE)

Value

A character vector of hex colour codes, usable wherever **ggplot2** takes colours.

See Also

`scale_color_tidyMacro`, `scale_fill_tidyMacro`, `fThemeTidyMacro`

Examples

```
# The first three colours are the ones the package's own plots use.
tidyMacro_colors[1:3]

# Use with ggplot2
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point() +
  scale_color_manual(values = tidyMacro_colors) +
  fThemeTidyMacro()
```

Uhlig2005 *Uhlig (2005) Data*

Description

Monthly US data used to replicate the agnostic sign-restriction VAR of Uhlig (2005), covering January 1965 - December 2003.

Usage

```
Uhlig2005
```

Format

A data frame with 468 rows and 7 columns:

Date Month start date (`Date`), 1965-01 - 2003-12

Real GDP Log real GDP, interpolated to monthly frequency

GDP Deflator Log GDP deflator, interpolated to monthly frequency

Commodity Price Idx. Log commodity price index

Total Reserves Log total reserves

Non-Borrowed Reserves Log non-borrowed reserves

Fed. Funds Rate Federal funds rate, in percent

Details

The five log-level series are stored unscaled. Multiply them by 100 before estimation so that impulse responses read in percent, as in the original replication code; the funds rate is already in percent and must be left alone.

Identification is by sign restrictions only: a contractionary monetary policy shock raises the funds rate and lowers the GDP deflator, commodity prices and non-borrowed reserves for six months. Real GDP and total reserves are deliberately left unrestricted - the response of output is the object of the exercise, so it is not assumed.

Data taken from the replication files of the VAR Toolbox of Ambrogio Cesa-Bianchi (https://github.com/ambropo/VAR-Toolbox).

References

Uhlig, H. (2005). What are the effects of monetary policy on output? Results from an agnostic identification procedure. *Journal of Monetary Economics*, 52(2), 381-419. doi:10.1016/j.jmoneco.2004.05.007

See Also

`fSignRestr` for the estimator, `ADRR2018` for the same variables over a longer sample.

Examples

```
data("Uhlig2005")

y <- Uhlig2005 |>
  dplyr::mutate(dplyr::across(-c(Date, `Fed. Funds Rate`), \(x) 100 * x)) |>
  dplyr::select(-Date) |>
  as.matrix()

SIGN <- matrix(0, 6, 6)
SIGN[, 1] <- c(0, -1, -1, 0, -1, 1)

fit <- fSignRestr(y, p = 12, c = 1, sign = SIGN, nsteps = 60,
                  ndraws = 100, sr_hor = 6, seed = 42, n_threads = 2)
```

Index

∗ **datasets**
ACR2019, 5
ADRR2018, 6
BankingDeregulation, 8
BeaudryPortier2014, 9
Bloom2009, 10
CBS2022, 11
DemocracyGrowth, 12
Gali1999, 91
GK2015, 92
GK2015_2, 93
JordaTaylor2025, 94
JordaTaylor2025IV, 95
Kaenzig2021, 96
MianSufiVerner2017, 97
MianSufiVerner2018, 98
TenreyroThwaites2016, 102
Uhlig2005, 107

ACR2019, 5
ADRR2018, *5*, 6, *6*, *108*

BankingDeregulation, 8
BeaudryPortier2014, 9
Bloom2009, 10

CBS2022, 11

DemocracyGrowth, 12
discrete_scale, *101, 102*

fAICBIC, 13, *88*
fBootstrapBQ, 14, *49*
fBootstrapChol, 15, *31*, *49*, *88*
fBootstrapCholCorrected, 17, *49*
fBootstrapHDIV, 18
fBootstrapHetero, 20, *49*
fBootstrapIVInvertible, 22, *26*
fBootstrapIVMBB, 23, *49*
fBootstrapIVRecover, *23*, 25, *83, 84*
fBootstrapMax, 27, *49*, *60*
fBootstrapMaxCorrected *(*fBootstrapMax*)*, 27
fBootstrapUhlig, 29, *49*, *86*
fBootstrapUhligCorrected *(*fBootstrapUhlig*)*, 29
fBootstrapVAR, 30
fBQIRF, 31, *49*
fCheckRestrictions, 33
fCheckStability, 34
fCholeskyIRF, 35, *38*, *47*, *49*, *76*, *81*
fCompanionMatrix, 36
fFEVDChol, 37
fFEVDIV, 38, *38*
fGenerateQ, *33*, 39
fGenerateVARData, 40
fGetBands, *23*, *26*, 41
fGetShock, 42
fHDIV, 43
fHDShock, 44, *80*
fHeteroIRF, 45
fHistDec, 46, *69*
fIRF, 47, *70, 71*
fLagMakerMatrix, 50
fLP, 51, *56*, *63*, *66*, *95*, *104*
fLPDID, 53
fLPIV, 55, *95*, *104*
fLPPanel, *53*, 57, *97–99*, *104*
fMaxIRF, *27*, 60
fMBBVAR, 61
fMSW, *49*, 62
fNLLP, 63, *73, 74*, *99, 100*, *102, 103*, *105*
fOLS, 67
fPlotHistDec, *44*, *47*, 68
fPlotIRF, *47*, *49*, *59*, 70, *80*
fPlotLPDID, 72
fPlotNLLP, *66*, 73, *105*
fPlotVarDec, *38*, 74
fPolyConvolve, 75, *81*
fSetTidyMacroTheme, 76, *85*

fSignRestr, *5–7*, *11*, *33*, *44*, 77, *93*, *94*, *108*
fSpectralFEVD, 80
fSVARSteps, 81
fTestInvertibility, 82, *84*, *92*
fTestRecoverability, *83*, 84, *92*
fThemeTidyMacro, *77*, 85, *101*, *102*, *106*
fUhligIRF, *29*, *60*, 86, *87*
fUhligMaxShare, *86*, 87
fVAR, *13*, *28*, *30*, *31*, *39*, *47*, 88
fVARX, 89
fWoldIRF, *38*, *39*, *47*, *60*, *76*, *86–88*, 90

Gali1999, 91
GK2015, 92, *93*, *94*
GK2015_2, *92*, 93

JordaTaylor2025, 94, *95*
JordaTaylor2025IV, 95, *95*

Kaenzig2021, 96

MianSufiVerner2017, 97, *98*, *99*
MianSufiVerner2018, *97*, 98, *98*

predict.fNLLP, *66*, 99
print.fSignRestr *(*fSignRestr*)*, 77

scale_color_tidyMacro, *77*, *85*, 100, *102*, *106*
scale_colour_tidyMacro *(*scale_color_tidyMacro*)*, 100
scale_fill_tidyMacro, 101, *101*, *106*

TenreyroThwaites2016, *66*, 102
theme, *85*
theme_bw, *85*
theme_set, *77*
tidy.fLP, *59*, 103
tidy.fLPPanel, 104
tidy.fNLLP, *66*, *74*, 105
tidyMacro *(*tidyMacro-package*)*, 4
tidyMacro-package, 4
tidyMacro_colors, *101*, *102*, 106

Uhlig2005, *5–7*, 107