Package ‘intrinsicFRP’

September 18, 2023

Title Oracle Estimation and Inference for Tradable Factor Risk Premia

Version 1.0.0

Date 2023-09-18

Maintainer Alberto Quaini <alberto91quaini@gmail.com>

Description Tradable factor risk premia are given by the negative factor covariance with the Stochastic Discount Factor projection on returns. This package provides efficient computation of tradable and Oracle tradable factor risk premia estimators and their standard errors; see A. Quaini, F. Trojani and M. Yuan (2023) <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4574683>. Tradable factor risk premia are robust to misspecification or weak identification in asset pricing models, and they are zero for any factor weakly correlated with returns. Their Oracle estimator performs as well as if the weak or useless factors were known in advance. This means it not only consistently removes useless factors and factors weakly correlated with returns but also gives rise to reliable tests of asset pricing models.

License GPL (>= 3)

URL https://github.com/a91quaini/intrinsicFRP

BugReports https://github.com/a91quaini/intrinsicFRP/issues

Encoding UTF-8

RoxygenNote 7.2.3

LinkingTo Rcpp, RcppArmadillo

Imports graphics, Rcpp, stats

Depends R (>= 2.10)

LazyData true

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Alberto Quaini [aut, cre, cph] (<https://orcid.org/0000-0002-1251-0599>)

Repository CRAN

Date/Publication 2023-09-18 12:40:13 UTC

R topics documented:

ChenFang2019BetaRankTest 2
factors 3
FRP 4
HJMisspecificationTest 5
OracleTFRP 6
returns 8
TFRP 9

Index **10**

ChenFang2019BetaRankTest

Compute the Chen Fang 2019 beta rank test

Description

Computes the Chen fang 2019 rank statistic and p-value of the null that the matrix of regression loadings of test asset excess returns on risk factors has reduced rank. If `target_level_kp2006_rank_test` > `0`, it uses the iterative Kleibergen Paap 2006 rank test to estimate the initial rank, with `level` = `target_level_kp2006_rank_test` / `n_factors`. If `target_level_kp2006_rank_test` <= `0`, the initial rank estimator is taken to be the number of singular values above `n_observations^(-1/4)`. It assumes n_factors < n_returns.

Usage

```
ChenFang2019BetaRankTest(
  returns,
  factors,
  n_bootstrap = 500,
  target_level_kp2006_rank_test = 0.05,
  check_arguments = TRUE
)
```

Arguments

`returns` — `n_observations x n_returns`-dimensional matrix of test asset excess returns.

`factors` — `n_observations x n_factors`-dimensional matrix of risk factors.

`n_bootstrap` — numeric integer indicating the number of bootstrap samples used to compute the Chen fang 2019 test. Default is `500`.

`target_level_kp2006_rank_test` — numeric level of the Kleibergen Paap 2006 rank test. If it is strictly grater than zero, then the iterative Kleibergen Paap 2006 rank test at `level` = `target_level_kp2006_rank_test` / `n_factors` is used to compute an initial estimator of the rank of the factor loadings in the Chen Fang 2019 rank test. Otherwise, the initial rank estimator is taken to be the number of singular values above `n_observations^(-1/4)`. Default is `0.05` (as correction for multiple testing).

`check_arguments`
: boolean `TRUE` if you want to check function arguments; `FALSE` otherwise. Default is `TRUE`.

Value

a list containing the Chen fang 2019 rank statistic and the corresponding p-value.

Examples

```
# import package data on 15 risk factors and 42 test asset excess returns
factors = intrinsicFRP::factors[,-1]
returns = intrinsicFRP::returns[,-1]

# compute the model identification test
hj_test = ChenFang2019BetaRankTest(returns, factors)
```

`factors` *Factors - monthly observations from* `01/1970` *to* `12/2021`

Description

Monthly observations from `01/1970` to `12/2021` of the Fama-French 5 factors the momentum factor.

Usage

```
factors
```

Format

`factors`**:**

A data frame with `624` rows and `7` columns:

Date Date in yyyymm format

... Factor observations

Source

https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

FRP *Compute factor risk premia from data*

Description

Computes Fama MachBeth (1973) or misspecification-robust factor risk premia of Kan Robotti Shanken (2013) from data on factors and test asset excess returns. Optionally computes the corresponding heteroskedasticity and autocorrelation robust standard errors using the Newey-West (1994) plug-in procedure to select the number of relevant lags, i.e., `n_lags = 4 * (n_observations/100)^(2/9)`.

Usage

```
FRP(
  returns,
  factors,
  misspecification_robust = TRUE,
  include_standard_errors = FALSE,
  check_arguments = TRUE
)
```

Arguments

`returns` `n_observations x n_returns`-dimensional matrix of test asset excess returns.

`factors` `n_observations x n_factors`-dimensional matrix of factors.

`misspecification_robust`
boolean `TRUE` for the "misspecification-robust" Kar Robotti Shanken GLS approach using the inverse covariance matrix of returns; `FALSE` for standard Fama-Mac-Beth risk premia. Default is `TRUE`.

`include_standard_errors`
boolean `TRUE` if you want to compute the factor risk premia HAC standard errors; `FALSE` otherwise. Default is `FALSE`.

`check_arguments`
boolean `TRUE` if you want to check function arguments; `FALSE` otherwise. Default is `TRUE`.

Value

a list containing `n_factors`-dimensional vector of factor risk premia in `"risk_premia"`; if `include_standard_errors=TRU` then it further includes `n_factors`-dimensional vector of factor risk premia standard errors in `"standard_errors"`.

Examples

```
# import package data on 15 risk factors and 42 test asset excess returns
factors = intrinsicFRP::factors[,-1]
returns = intrinsicFRP::returns[,-1]
```

```
# compute KRS factor risk premia and their standard errors
frp = FRP(returns, factors, include_standard_errors = TRUE)
```

HJMisspecificationTest

Compute factor risk premia

Description

Computes the Hansen-Jagannatan misspecification statistic and p-value of an asset pricing model from test asset excess returns and risk factors.

Usage

```
HJMisspecificationTest(returns, factors, check_arguments = TRUE)
```

Arguments

`returns`	`n_observations x n_returns`-dimensional matrix of test asset excess returns.
`factors`	`n_observations x n_factors`-dimensional matrix of risk factors.
`check_arguments`	boolean `TRUE` if you want to check function arguments; `FALSE` otherwise. Default is `TRUE`.

Value

a list containing the HJ test statistic and the corresponding p-value.

Examples

```
# import package data on 15 risk factors and 42 test asset excess returns
factors = intrinsicFRP::factors[,-1]
returns = intrinsicFRP::returns[,-1]

# compute the HJ model misspecification test
hj_test = HJMisspecificationTest(returns, factors)
```

OracleTFRP *Compute optimal adaptive tradable factor risk premia*

Description

Computes optimal adaptive tradable factor risk premia for various penalty parameter values from data on factors and test asset excess returns. Tuning is performed via Generalized Cross Validation (GCV), Cross Validation (CV) or Rolling Validation (RV). Oracle weights can be based on the correlation between factors and returns, on the regression coefficients of returns on factors or on the first-step tradable risk premia estimator. Optionally computes the corresponding heteroskedasticity and autocorrelation robust standard errors using the Newey-West (1994) plug-in procedure to select the number of relevant lags, i.e., `n_lags = 4 * (n_observations/100)^(2/9)`.

Usage

```
OracleTFRP(
  returns,
  factors,
  penalty_parameters,
  weighting_type = "c",
  tuning_type = "g",
  relaxed = FALSE,
  include_standard_errors = FALSE,
  one_stddev_rule = FALSE,
  gcv_scaling_n_assets = FALSE,
  gcv_identification_check = FALSE,
  target_level_kp2006_rank_test = 0.05,
  n_folds = 5,
  n_train_observations = 120,
  n_test_observations = 12,
  roll_shift = 12,
  plot_score = TRUE,
  check_arguments = TRUE
)
```

Arguments

`returns` `n_observations x n_returns`-dimensional matrix of test asset excess returns.

`factors` `n_observations x n_factors`-dimensional matrix of factors.

`penalty_parameters` `n_parameters`-dimensional vector of penalty parameter values from smallest to largest.

`weighting_type` character specifying the type of adaptive weights: based on the correlation between factors and returns `'c'`; based on the regression coefficients of returns on factors `'b'`; based on the first-step tradable risk premia estimator `'a'`; otherwise a vector of ones (any other character). Default is `'c'`.

tuning_type
: character indicating the parameter tuning type: 'g' for generalized cross validation; 'r' for rolling validation. Default is 'g'.

relaxed
: boolean TRUE if you want to compute a post-selection unpenalized tradable factor risk premia to remove the bias due to shrinkage; FALSEotherwise. Default isFALSE‘.

include_standard_errors
: boolean TRUE if you want to compute the adaptive tradable factor risk premia HAC standard errors; FALSE otherwise. Default is FALSE.

one_stddev_rule
: boolean TRUE for picking the most parsimonious model whose score is not higher than one standard error above the score of the best model; FALSE for picking the best model. Default is FALSE.

gcv_scaling_n_assets
: (only relevant for tuning_type ='g') boolean TRUE for sqrt(n_assets) scaling (sqrt(n_assets) / n_observations); FALSE otherwise (1 / n_observations). Default is FALSE.

gcv_identification_check
: (only relevant for tuning_type ='g') boolean TRUE for a loose check for model identification; FALSE otherwise. Default is FALSE.

target_level_kp2006_rank_test
: (only relevant for tuning_type ='g' and if gcv_identification_check is TRUE) numeric level of the Kleibergen Paap 2006 rank test. If it is strictly grater than zero, then the iterative Kleibergen Paap 2006 rank test at level = target_level_kp2006_rank_test / n_factors is used to compute an initial estimator of the rank of the factor loadings in the Chen Fang 2019 rank test. Otherwise, the initial rank estimator is taken to be the number of singular values above n_observations^(-1/4). Default is 0.05 (as correction for multiple testing).

n_folds
: (only relevant for tuning_type ='c') integer number of k-fold for cross validation. Default is 5.

n_train_observations
: (only relevant for tuning_type ='r') number of observations in the rolling training set. Default is 120.

n_test_observations
: (only relevant for tuning_type ='r') number of observations in the test set. Default is 12.

roll_shift
: (only relevant for tuning_type ='r') number of observation shift when moving from the rolling window to the next one. Default is 12.

plot_score
: boolean TRUE for plotting the model score; FALSE otherwise. Default is TRUE.

check_arguments
: boolean TRUE if you want to check function arguments; FALSE otherwise. Default is TRUE.

Value

a list containing the n_factors-dimensional vector of adaptive tradable factor risk premia in "risk_premia"; the optimal penalty parameter value in "penalty_parameter"; the model score for each penalty parameter value in "model_score"; if include_standard_errors=TRUE, then it further includes n_factors-dimensional vector of tradable factor risk premia standard errors in "standard_errors".

Examples

```
# import package data on 15 risk factors and 42 test asset excess returns
factors = intrinsicFRP::factors[,-1]
returns = intrinsicFRP::returns[,-1]

penalty_parameters = seq(0., 1., length.out = 100)

# compute optimal adaptive tradable factor risk premia and their standard errors
oracle_tfrp = OracleTFRP(
returns,
factors,
penalty_parameters,
include_standard_errors = TRUE
)
```

---

returns — *Test Asset Excess Returns - monthly observations from* `01/1970` *to* `12/2021`

Description

Monthly excess returns on the 25 Size/Book-to-Market double sorted portfolios and the 17 industry portfolios from `01/1970` to `12/2021`.

Usage

```
returns
```

Format

`returns`**:**

A data frame with 624 rows and 43 columns:

Date Date in yyyymm format

... Asset excess returns

Source

https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

TFRP *Compute tradable factor risk premia*

Description

Computes tradable factor risk premia from data on factors and test asset excess returns. Optionally computes the corresponding heteroskedasticity and autocorrelation robust standard errors using the Newey-West (1994) plug-in procedure to select the number of relevant lags, i.e., `n_lags = 4 * (n_observations/100)^(2/9)`.

Usage

```
TFRP(returns, factors, include_standard_errors = FALSE, check_arguments = TRUE)
```

Arguments

`returns` n_observations x n_returns-dimensional matrix of test asset excess returns.

`factors` n_observations x n_factors-dimensional matrix of factors.

`include_standard_errors` boolean TRUE if you want to compute the tradable factor risk premia HAC standard errors; FALSE otherwise. Default is FALSE.

`check_arguments` boolean TRUE if you want to check function arguments; FALSE otherwise. Default is TRUE.

Value

a list containing `n_factors`-dimensional vector of tradable factor risk premia in `"risk_premia"`; if `include_standard_errors=TRUE`, then it further includes `n_factors`-dimensional vector of tradable factor risk premia standard errors in `"standard_errors"`.

Examples

```
# import package data on 15 risk factors and 42 test asset excess returns
factors = intrinsicFRP::factors[,-1]
returns = intrinsicFRP::returns[,-1]

# compute tradable factor risk premia and their standard errors
tfrp = TFRP(returns, factors, include_standard_errors = TRUE)
```

Index

∗ **datasets**
    factors, 3
    returns, 8

ChenFang2019BetaRankTest, 2

factors, 3
FRP, 4

HJMisspecificationTest, 5

OracleTFRP, 6

returns, 8

TFRP, 9