Package ‘MarkowitzR’

July 2, 2014

Maintainer Steven E. Pav <shabbychef@gmail.com>

Version 0.1402

Date 2014-02-11

License LGPL-3

Title Statistical significance of the Markowitz portfolio

BugReports https://github.com/shabbychef/MarkowitzR/issues

Description a collection of tools for analyzing significance of Markowitz portfolios.

Depends R (>= 3.0.2)

Imports matrixcalc, sandwich, gtools

Suggests SharpeR, testthat, knitr

URL http://www.r-project.org, https://github.com/shabbychef/MarkowitzR

VignetteBuilder knitr

Author Steven E. Pav [aut, cre]

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-12 07:19:08

R topics documented:

`itheta_vcov` *Compute variance covariance of Inverse 'Unified' Second Moment*

Description

Computes the variance covariance matrix of the inverse unified second moment matrix.

Usage

```
itheta_vcov(X,vcov.func=vcov,fit.intercept=TRUE)
```

Arguments

`X`	an $n \times p$ matrix of observed returns.
`vcov.func`	a function which takes an object of class `lm`, and computes a variance-covariance matrix. If equal to the string `"normal"`, we assume multivariate normal returns.
`fit.intercept`	a boolean controlling whether we add a column of ones to the data, or fit the raw uncentered second moment. For now, must be true when assuming normal returns.

Details

Given p-vector x with mean μ and covariance, Σ, let y be x with a one prepended. Then let $\Theta = E\left(yy^{\top}\right)$, the uncentered second moment matrix. The inverse of Θ contains the (negative) Markowitz portfolio and the precision matrix.

Given n contemporaneous observations of p-vectors, stacked as rows in the $n \times p$ matrix X, this function estimates the mean and the asymptotic variance-covariance matrix of Θ^{-1}.

One may use the default method for computing covariance, via the `vcov` function, or via a 'fancy' estimator, like `sandwich:vcovHAC`, `sandwich:vcovHC`, *etc.*

Value

a list containing the following components:

`mu`	a $q = (p+1)(p+2)/2$ vector of 1 + squared maximum Sharpe, the negative Markowitz portfolio, then the vech'd precision matrix of the sample data
`Ohat`	the $q \times q$ estimated variance covariance matrix.
`n`	the number of rows in `X`.
`pp`	the number of assets plus `as.numeric(fit.intercept)`.

Note

By flipping the sign of X, the inverse of Θ contains the *positive* Markowitz portfolio and the precision matrix on X. Performing this transform before passing the data to this function should be considered idiomatic.

A more general form of this function exists as `mp_vcov`.

Replaces similar functionality from SharpeR package, but with modified API.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Pav, S. E. "Asymptotic Distribution of the Markowitz Portfolio." 2013 http://arxiv.org/abs/1312.0557

See Also

theta_vcov, mp_vcov.

Examples

```
X <- matrix(rnorm(1000*3),ncol=3)
# putting in -X is idiomatic:
ism <- itheta_vcov(-X)
iSigmas.n <- itheta_vcov(-X,vcov.func="normal")
iSigmas.n <- itheta_vcov(-X,fit.intercept=FALSE)
# compute the marginal Wald test statistics:
qidx <- 2:ism$pp
wald.stats <- ism$mu[qidx] / sqrt(diag(ism$Ohat[qidx,qidx]))

# make it fat tailed:
X <- matrix(rt(1000*3,df=5),ncol=3)
ism <- itheta_vcov(X)
qidx <- 2:ism$pp
wald.stats <- ism$mu[qidx] / sqrt(diag(ism$Ohat[qidx,qidx]))
## Not run:
if (require(sandwich)) {
 ism <- itheta_vcov(X,vcov.func=vcovHC)
 qidx <- 2:ism$pp
 wald.stats <- ism$mu[qidx] / sqrt(diag(ism$Ohat[qidx,qidx]))
}

## End(Not run)
# add some autocorrelation to X
Xf <- filter(X,c(0.2),"recursive")
colnames(Xf) <- colnames(X)
ism <- itheta_vcov(Xf)
qidx <- 2:ism$pp
wald.stats <- ism$mu[qidx] / sqrt(diag(ism$Ohat[qidx,qidx]))
## Not run:
if (require(sandwich)) {
ism <- itheta_vcov(Xf,vcov.func=vcovHAC)
 qidx <- 2:ism$pp
 wald.stats <- ism$mu[qidx] / sqrt(diag(ism$Ohat[qidx,qidx]))
}

## End(Not run)
```

MarkowitzR *statistics concerning the Markowitz portfolio*

Description

Inference on the Markowitz portfolio.

Markowitz Portfolio

Suppose x is a p-vector of returns of some assets with expected value μ and covariance Σ. The *Markowitz Portfolio* is the portfolio $w = \Sigma^{-1}\mu$. Scale multiples of this portfolio solve various portfolio optimization problems, among them

$$\mathrm{argmax}_{w : w^{\top}\Sigma w \le R^2} \frac{\mu^{\top} w - r_0}{\sqrt{w^{\top}\Sigma w}}$$

This packages supports various statistical tests around the elements of the Markowitz Portfolio, and its Sharpe ratio, including the possibility of hedging, and scalar conditional heteroskedasticity and conditional expectation.

Legal Mumbo Jumbo

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Pav, S. E. "Asymptotic Distribution of the Markowitz Portfolio." 2013 http://arxiv.org/abs/1312.0557

Britten-Jones, Mark. "The Sampling Error in Estimates of Mean-Variance Efficient Portfolio Weights." The Journal of Finance 54, no. 2 (1999): 655–671. http://www.jstor.org/stable/2697722

Bodnar, Taras and Okhrin, Yarema. "On the Product of Inverse Wishart and Normal Distributions with Applications to Discriminant Analysis and Portfolio Theory." Scandinavian Journal of Statistics 38, no. 2 (2011): 311–331. http://dx.doi.org/10.1111/j.1467-9469.2011.00729.x

Markowitz, Harry. "Portfolio Selection." The Journal of Finance 7, no. 1 (1952): 77–91. http://www.jstor.org/stable/2975974

Brandt, Michael W. "Portfolio Choice Problems." Handbook of Financial Econometrics 1 (2009): 269–336. http://shr.receptidocs.ru/docs/5/4748/conv_1/file1.pdf#page=298

MarkowitzR-NEWS	*News for package 'MarkowitzR':*

Description

News for package 'MarkowitzR':

MarkowitzR Initial Version 0.1402 (2014-02-14)

- first CRAN release.

mp_vcov	*Estimate Markowitz Portfolio*

Description

Estimates the Markowitz Portfolio or Markowitz Coefficient subject to subspace and hedging constraints, and heteroskedasticity.

Usage

```
mp_vcov(X,feat=NULL,vcov.func=vcov,fit.intercept=TRUE,weights=NULL,Jmat=NULL,Gmat=NULL)
```

Arguments

`X`	an $n \times p$ matrix of observed returns.
`feat`	an $n \times f$ matrix of observed features. defaults to none, in which case `fit.intercept` must be `TRUE`. If `fit.intercept` is true, ones will be prepended to the features.
`weights`	an optional n vector of the weights. The returns and features will be multiplied by the weights. Weights should be inverse volatility estimates. Defaults to homoskedasticity.
`Jmat`	an optional $p_j \times p$ matrix of the subspace in which we constraint portfolios. Defaults essentially to the $p \times p$ identity matrix.
`Gmat`	an optional $p_g \times p$ matrix of the subspace to which we constraint portfolios to have zero covariance. The rowspace of `Gmat` must be spanned by the rowspace of `Jmat`. Defaults essentially to the $0 \times p$ empty matrix.
`vcov.func`	a function which takes an object of class `lm`, and computes a variance-covariance matrix. If equal to the string `"normal"`, we assume multivariate normal returns.
`fit.intercept`	a boolean controlling whether we add a column of ones to the data, or fit the raw uncentered second moment. For now, must be true when assuming normal returns.

Details

Suppose that the expectation of p-vector x is linear in the f-vector f, but the covariance of x is stationary and independent of f. The 'Markowitz Coefficient' is the $p \times f$ matrix W such that, conditional on observing f, the portfolio Wf maximizes Sharpe. When f is the constant 1, the Markowitz Coefficient is the traditional Markowitz Portfolio.

Given n observations of the returns and features, given as matrices X, F, this code computes the Markowitz Coefficient along with the variance-covariance matrix of the Coefficient and the precision matrix. One may give optional weights, which are inverse conditional volatility. One may also give optional matrix J, G which define subspace and hedging constraints. Briefly, they constrain the portfolio optimization problem to portfolios in the row space of J and with zero covariance with the rows of G. It must be the case that the rows of J span the rows of G. J defaults to the $p \times p$ identity matrix, and G defaults to a null matrix.

One may use the default method for computing covariance, via the `vcov` function, or via a 'fancy' estimator, like `sandwich:vcovHAC`, `sandwich:vcovHC`, *etc.*

Value

a list containing the following components:

`mu`	Letting $r = f + p + fit.intercept$, this is a $q = (r)(r + 1)/2$ vector...
`Ohat`	The $q \times q$ estimated variance covariance matrix of `mu`.
`W`	The estimated Markowitz coefficient, a $p \times (fit.intercept + f)$ matrix. The first column corresponds to the intercept term if it is fit. Note that for convenience this function performs the sign flip, which is not performed on `mu`.
`What`	The estimated variance covariance matrix of `vech(W)`. Letting $s = p(fit.intercept + f)$, this is a $s \times s$ matrix.
`widxs`	The indices into `mu` giving `W`, and into `Ohat` giving `What`.
`n`	The number of rows in `X`.
`ff`	The number of features plus `as.numeric(fit.intercept)`.
`p`	The number of assets.

Note

Should also modify to include the theta estimates.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Pav, S. E. "Asymptotic Distribution of the Markowitz Portfolio." 2013 http://arxiv.org/abs/1312.0557

See Also

`theta_vcov`, `itheta_vcov`

Examples

```
set.seed(1001)
X <- matrix(rnorm(1000*3),ncol=3)
ism <- mp_vcov(X,fit.intercept=TRUE)
walds <- ism$W / sqrt(diag(ism$What))
print(t(walds))
# subspace constraint
Jmat <- matrix(rnorm(6),ncol=3)
ism <- mp_vcov(X,fit.intercept=TRUE,Jmat=Jmat)
walds <- ism$W / sqrt(diag(ism$What))
print(t(walds))
# hedging constraint
Gmat <- matrix(1,nrow=1,ncol=3)
ism <- mp_vcov(X,fit.intercept=TRUE,Gmat=Gmat)
walds <- ism$W / sqrt(diag(ism$What))

# now conditional expectation:

# generate data with given W, Sigma
Xgen <- function(W,Sigma,Feat) {
 Btrue <- Sigma %*% W
 Xmean <- Feat %*% t(Btrue)
 Shalf <- chol(Sigma)
 X <- Xmean + matrix(rnorm(prod(dim(Xmean))),ncol=dim(Xmean)[2]) %*% Shalf
}

n.feat <- 2
n.ret <- 8
n.obs <- 10000
set.seed(101)
Feat <- matrix(rnorm(n.obs * n.feat),ncol=n.feat)
Wtrue <- 10 * matrix(rnorm(n.feat * n.ret),ncol=n.feat)
Sigma <- cov(matrix(rnorm(100*n.ret),ncol=n.ret))
Sigma <- Sigma + diag(seq(from=1,to=3,length.out=n.ret))
X <- Xgen(Wtrue,Sigma,Feat)
ism <- mp_vcov(X,feat=Feat,fit.intercept=TRUE)
Wcomp <- cbind(0,Wtrue)
errs <- ism$W - Wcomp
dim(errs) <- c(length(errs),1)
Zerr <- solve(t(chol(ism$What)),errs)
```

---

theta_vcov — *Compute variance covariance of 'Unified' Second Moment*

Description

Computes the variance covariance matrix of sample mean and second moment.

Usage

```
theta_vcov(X,vcov.func=vcov,fit.intercept=TRUE)
```

Arguments

X — an $n \times p$ matrix of observed returns.

vcov.func — a function which takes an object of class lm, and computes a variance-covariance matrix. If equal to the string "normal", we assume multivariate normal returns.

fit.intercept — a boolean controlling whether we add a column of ones to the data, or fit the raw uncentered second moment. For now, must be true when assuming normal returns.

Details

Given p-vector x, the 'unified' sample is the $(p+1)(p+2)/2$ vector of 1, x, and vech$(xx^\top)$ stacked on top of each other. Given n contemporaneous observations of p-vectors, stacked as rows in the $n \times p$ matrix X, this function computes the mean and the variance-covariance matrix of the 'unified' sample.

One may use the default method for computing covariance, via the vcov function, or via a 'fancy' estimator, like sandwich:vcovHAC, sandwich:vcovHC, *etc*.

Value

a list containing the following components:

mu — a $q = (p+1)(p+2)/2$ vector of 1, then the mean, then the vech'd second moment of the sample data.

Ohat — the $q \times q$ estimated variance covariance matrix. When fit.intercept is true, the left column and top row are all zeros.

n — the number of rows in X.

pp — the number of assets plus as.numeric(fit.intercept).

Note

Replaces similar functionality from SharpeR package, but with modified API.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Pav, S. E. "Asymptotic Distribution of the Markowitz Portfolio." 2013 http://arxiv.org/abs/1312.0557

See Also

itheta_vcov.

Examples

```
X <- matrix(rnorm(1000*3),ncol=3)
Sigmas <- theta_vcov(X)
Sigmas.n <- theta_vcov(X,vcov.func="normal")
Sigmas.n <- theta_vcov(X,fit.intercept=FALSE)

# make it fat tailed:
X <- matrix(rt(1000*3,df=5),ncol=3)
Sigmas <- theta_vcov(X)
## Not run:
if (require(sandwich)) {
 Sigmas <- theta_vcov(X,vcov.func=vcovHC)
}

## End(Not run)
# add some autocorrelation to X
Xf <- filter(X,c(0.2),"recursive")
colnames(Xf) <- colnames(X)
Sigmas <- theta_vcov(Xf)
## Not run:
if (require(sandwich)) {
Sigmas <- theta_vcov(Xf,vcov.func=vcovHAC)
}

## End(Not run)
```

Index