

Multivariate GARCH models in R

Kris Boudt

Ghent University

VU Brussel/Amsterdam

R/Finance 2019

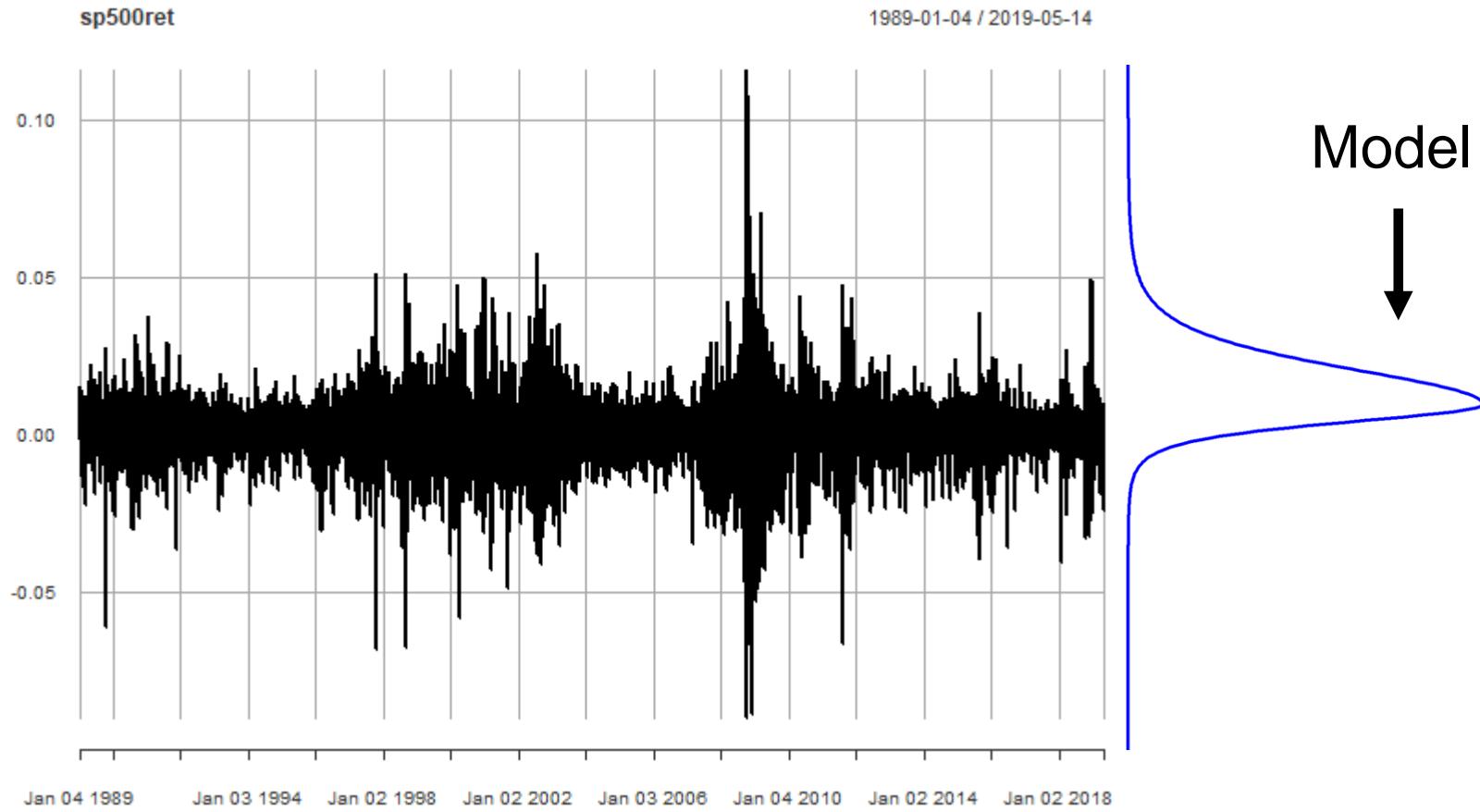
References

Ghalanos, Alexios (2019). *rmgarch*: Multivariate GARCH models. R package version 1.3-6.

Ardia, David, Kris Boudt, and Leopoldo Catania (2019). “Generalized Autoregressive Score Models in R: The GAS Package.” *Journal of Statistical Software* 88 (6), 1-28. doi: 10.18637/jss.v088.i06

Boudt, Kris, Alexios Ghalanos, Scott Payseur, and Eric Zivot. 2019. “Multivariate GARCH models for large-scale applications: A survey.” In H.D. Vinod and C.R. Rao (Ed.) *Handbook of Statistics*, Volume 41, forthcoming. doi: 10.1016/bs.host.2019.01.001

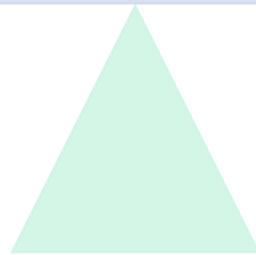
Predictive financial return modelling



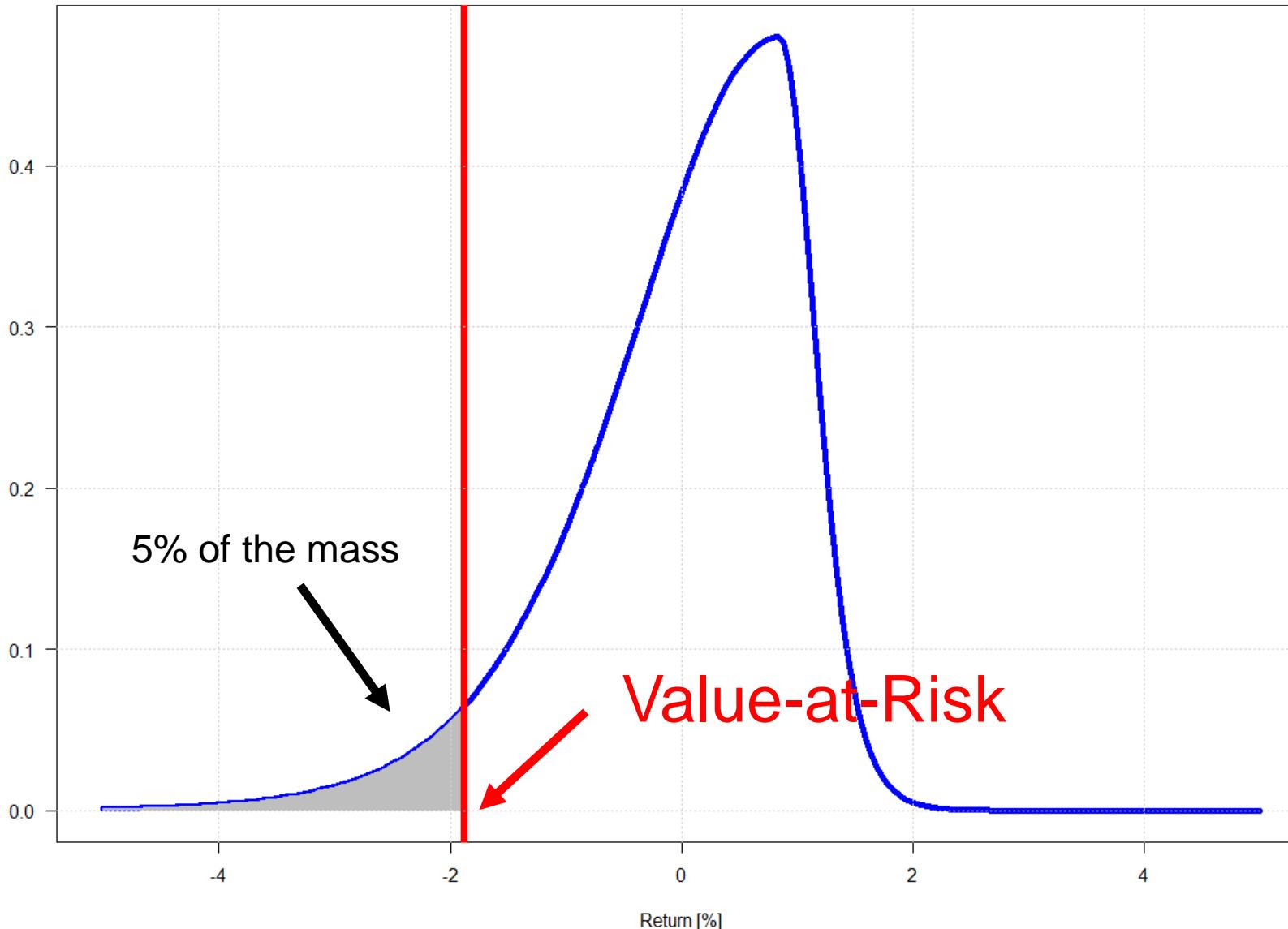
Goal: Optimal financial decision making

Maximizing
 $E[r_t^p | \mathfrak{I}_{t-1}]$

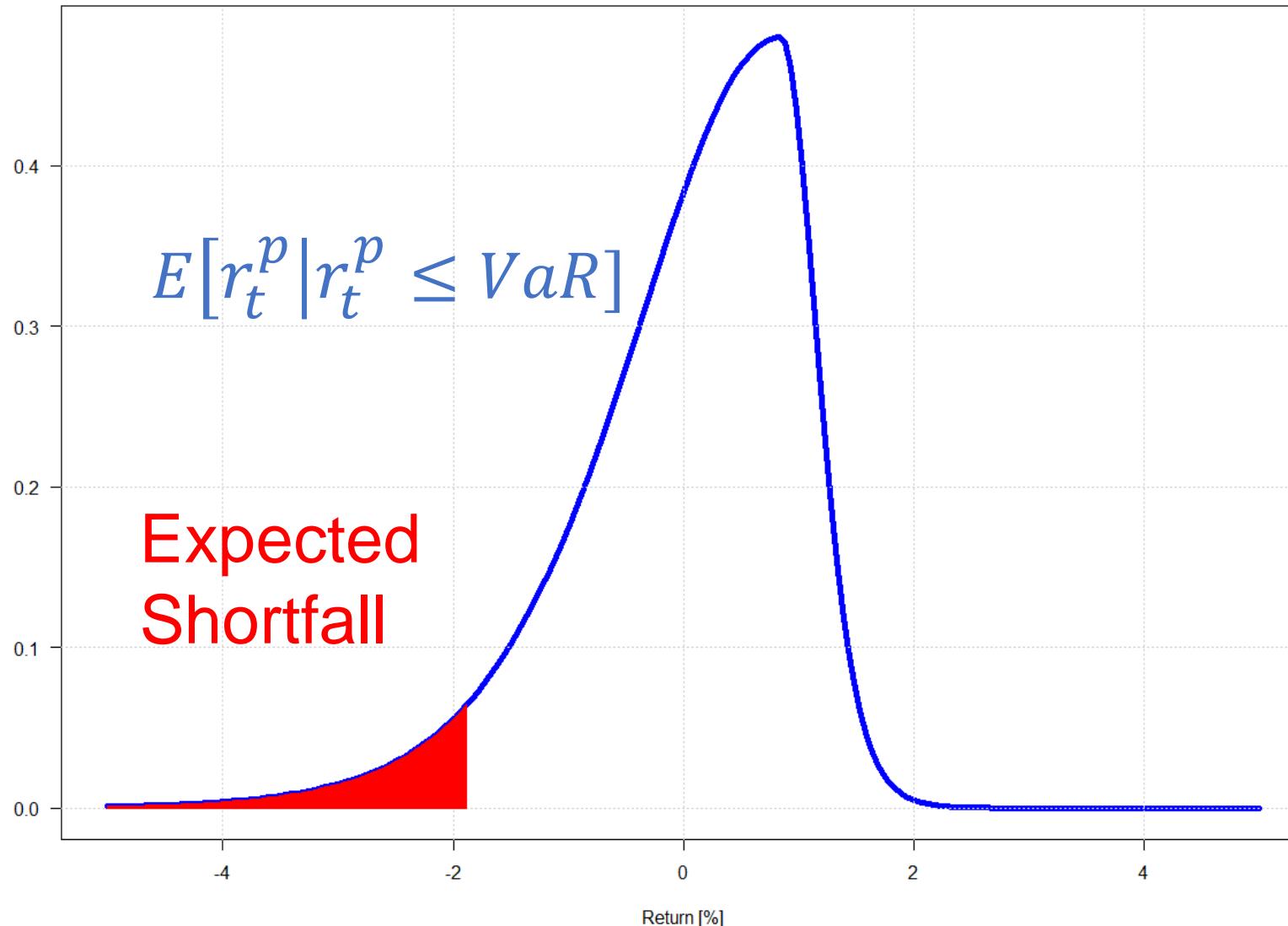
Minimizing
 $Risk[r_t^p | \mathfrak{I}_{t-1}]$



Which risk measure?



Which risk measure?



Multivariate view

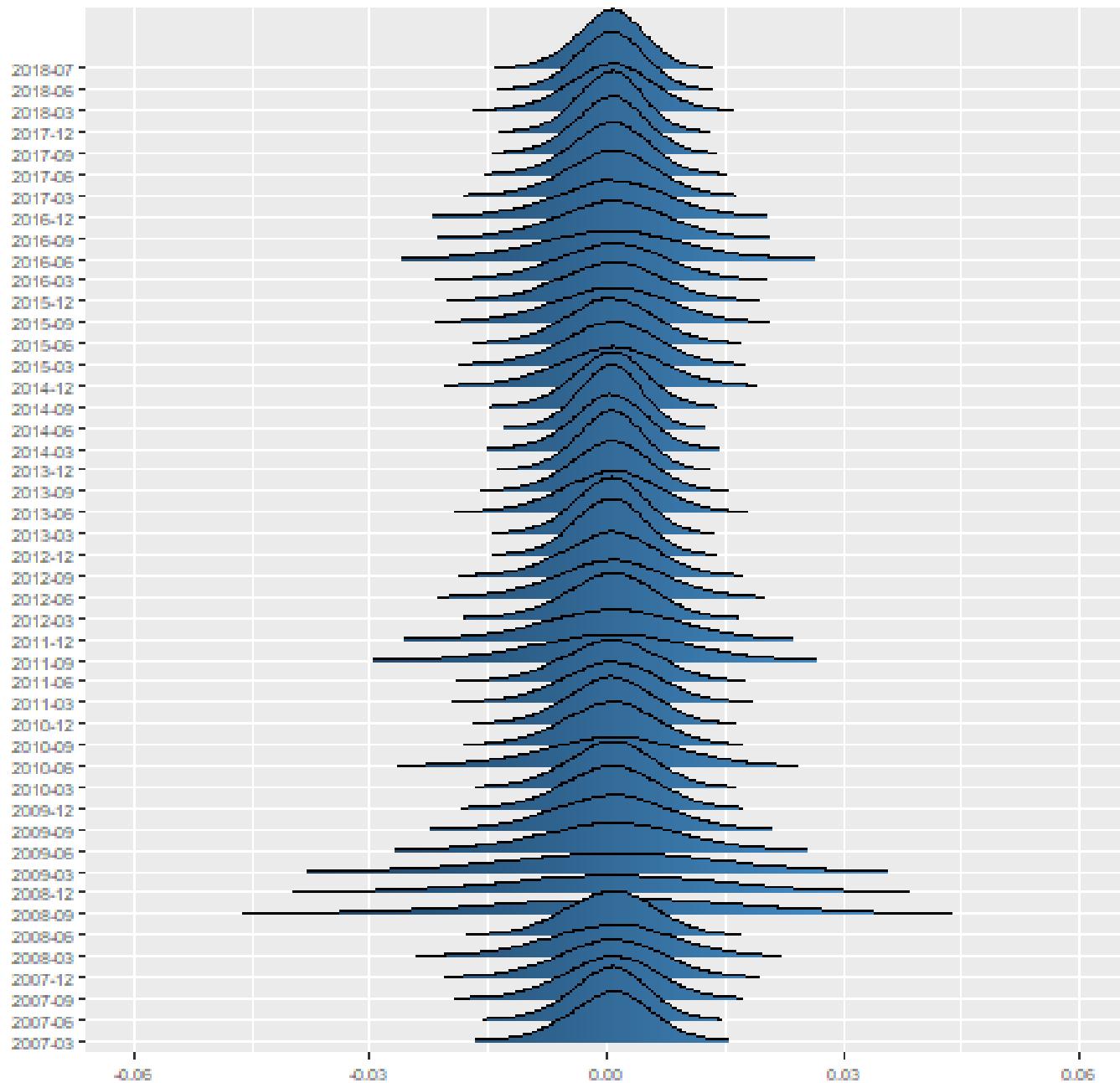
- The return of your investment is the portfolio return affected by the N underlying asset returns:

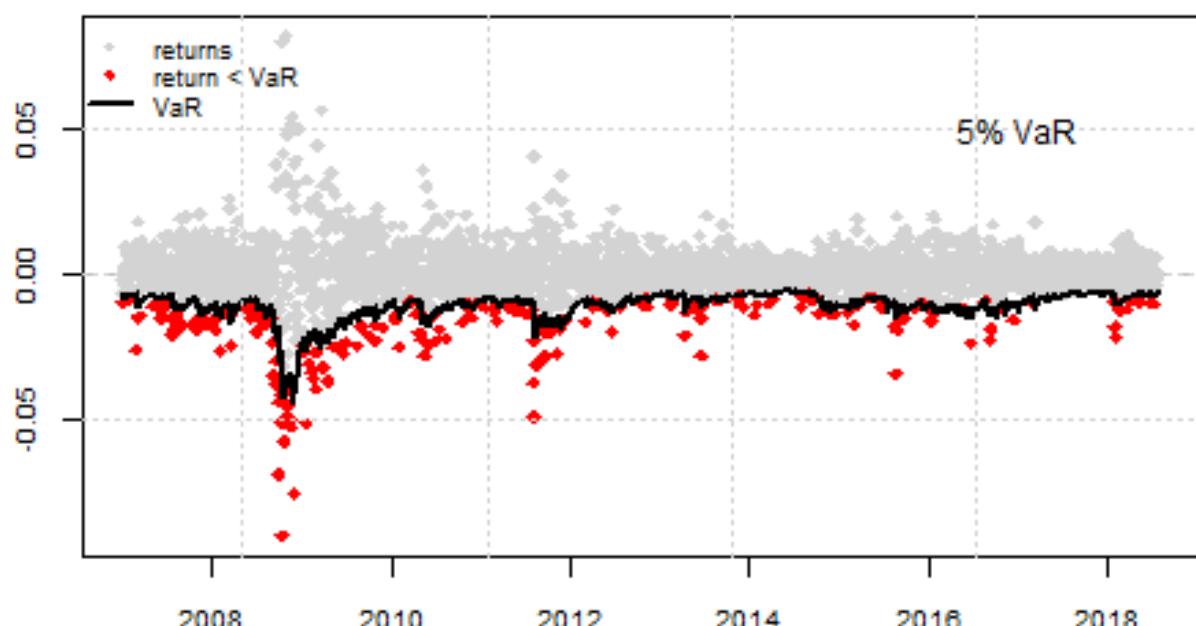
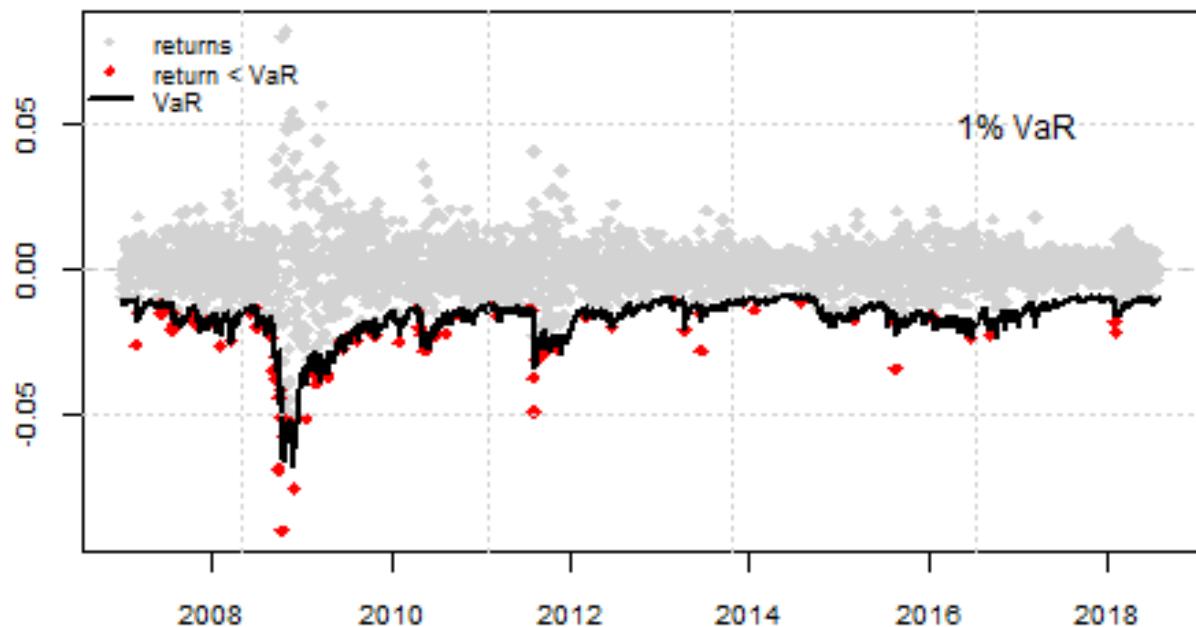
$$r_t^p = \mathbf{w}'_t \mathbf{r}_t$$

with $\mathbf{w}_t = \begin{bmatrix} w_{1t} \\ \vdots \\ w_{Nt} \end{bmatrix}$ and $\mathbf{r}_t = \begin{bmatrix} r_{1t} \\ \vdots \\ r_{Nt} \end{bmatrix}$

- Predict at time $t-1$ the conditional distribution of the N -dimensional return vector \mathbf{r}_t given the information in the returns $\mathbf{r}_1, \dots, \mathbf{r}_{t-1}$ (information set, denoted by \mathfrak{I}_{t-1})

Equal Weighted Portfolio Density (gogarch)





Outline

1. Use of filters to estimate the parameters of the multivariate return distribution
2. MGARCH models for returns
3. Illustration on predicting the distribution of the return of an equally-weighted portfolio of 10 Vanguard funds.
4. Outlook

Parameters of interest: multivariate moments

- [Conditional mean] Where the return is expected to be

$$\boldsymbol{\mu}_t = E[\mathbf{r}_t | \mathfrak{I}_{t-1}]$$

- [Conditional covariance] How large the joint deviations are expected to be

$$\boldsymbol{\Sigma}_t = E[(\mathbf{r}_t - \boldsymbol{\mu}_t)(\mathbf{r}_t - \boldsymbol{\mu}_t)^t | \mathfrak{I}_{t-1}]$$

Multivariate higher order moments

- [Conditional coskewness] How skewed the joint deviations are expected to be

$$\Phi_t = E[(r_t - \mu_t)(r_t - \mu_t)^t \otimes (r_t - \mu_t)^t | \mathfrak{I}_{t-1}]$$

- [Conditional cokurtosis] How fat-tailed the joint deviations are expected to be

$$\Psi_t = E[(r_t - \mu_t)(r_t - \mu_t)^t \otimes (r_t - \mu_t)^t \otimes (r_t - \mu_t)^t | \mathfrak{I}_{t-1}]$$

Once you know the multivariate moments...

- ... you also know the portfolio moments

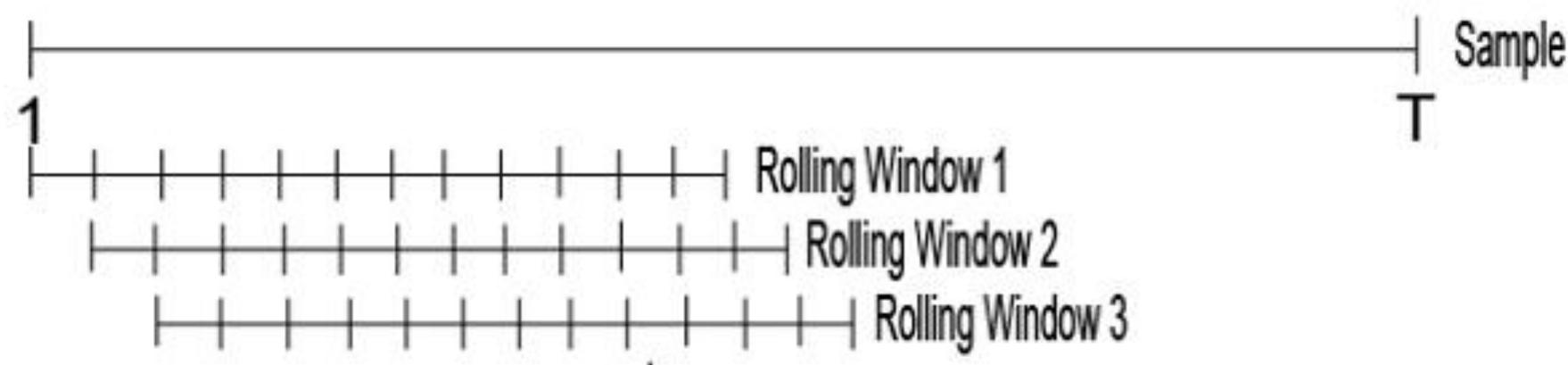
$$\mu_t^p = \mathbb{E}[r_t^p | \mathfrak{I}_{t-1}] = \mathbf{w}_t^t \boldsymbol{\mu}_t$$

$$\sigma_{p,t}^2 = \mathbb{E}[(r_t^p - \mu_t^p)^2 | \mathfrak{I}_{t-1}] = \mathbf{w}_t^t \boldsymbol{\Sigma}_t \mathbf{w}_t$$

$$\phi_t^p = \mathbb{E}[(r_t^p - \mu_t^p)^3 | \mathfrak{I}_{t-1}] = \mathbf{w}_t^t \boldsymbol{\Phi}_t (\mathbf{w}_t \otimes \mathbf{w}_t)$$

$$\psi_t^p = \mathbb{E}[(r_t^p - \mu_t^p)^4 | \mathfrak{I}_{t-1}] = \mathbf{w}_t^t \boldsymbol{\Psi}_t (\mathbf{w}_t \otimes \mathbf{w}_t \otimes \mathbf{w}_t)$$

Filtering by doing rolling window estimation



Use the return in each window to compute an unconditional estimate for the mean, covariance, coskewness and cokurtosis *for that window*.

Comoment estimators

	Traditional sample estimator	Shrinkage estimators	Structured estimator
	<i>Deal with the curse of dimensionality</i>		
PerformanceAnalytics functions	mean() cov() M3.MM() M4.MM()	M2.shrink() M3.shrink() M4.shrink()	M2.struct() M3.struct() M4.struct()
	(with arguments to finetune the estimation)		
Outliers?	Achieve robustness by applying PerformanceAnalytics::clean.boudt() to the return data prior to estimating the comoments.		
Disadvantage?	Ignores the position within the window: Descriptive of what has been But not forward looking about what will be.		

Exponentially Weighted Moving Average (EWMA)

$$\widehat{\Sigma}_t = (1 - \lambda_{\Sigma})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t + \lambda_{\Sigma}\widehat{\Sigma}_{t-1}$$

$$\widehat{\Phi}_t = (1 - \lambda_{\Phi})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t \otimes (\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t + \lambda_{\Phi}\widehat{\Phi}_{t-1}$$

$$\widehat{\Psi}_t = (1 - \lambda_{\Psi})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})(\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t \otimes (\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t \otimes (\mathbf{r}_{t-1} - \widehat{\mu}_{t-1})^t + \lambda_{\Psi}\widehat{\Psi}_{t-1}$$

- The scalar parameters λ_{Σ} , λ_{Φ} and λ_{Ψ} determine the weight on the past prediction.
- Use a grid search to find the values of λ_{Σ} , λ_{Φ} and λ_{Ψ} that are optimal with respect to your preferred statistical or economic criterion of choice.
- Implementation in PerformanceAnalytics: `M2.ewma()`, `M3.ewma()` and `M4.ewma()` taking as input the de-meanned returns.

From EWMA filters to MGARCH models

EWMA filters	→	MGARCH models	rmgarch package GAS package
<ul style="list-style-type: none"> Comoment estimates $\hat{\Sigma}_t$, $\hat{\Phi}_t$ and $\hat{\Psi}_t$ as measurable function of past returns. No assumptions about the true data generating process. 		<ul style="list-style-type: none"> Specify assumptions about the data generating process of r_t. There is a structural parameter θ. Those model assumptions lead to a parametric function for the multivariate moments: $(\mu_t, \Sigma_t, \Phi_t, \Psi_t) = h_\theta(r_1, \dots, r_{t-1})$ Fit the model to get an estimated θ After estimating θ, you get a parametric filter $(\hat{\mu}_t, \hat{\Sigma}_t, \hat{\Phi}_t, \hat{\Psi}_t) = h_{\hat{\theta}}(r_1, \dots, r_{t-1})$ 	dccspec() , gogarchspec() , MultiGASSpec() dccfit() , gogarchfit() , MultiGASfit() gogarchfilter() , dccfilter() , GAS:::GASFilter_multi()

Outline

1. Use of filters to estimate the parameters of the multivariate return distribution
2. MGARCH models for returns
3. Illustration on predicting the distribution of the return of an equally-weighted portfolio of 10 Vanguard funds.
4. Outlook

MGARCH models come in many flavors

- They have the following equations in common:

$$(1) \quad \mathbf{r}_t \mid \mathfrak{I}_{t-1} = \boldsymbol{\mu}_t + \boldsymbol{\varepsilon}_t$$

$$(2) \quad \boldsymbol{\varepsilon}_t = \boldsymbol{\Sigma}_t^{1/2} \mathbf{z}_t$$

$$(3) \quad \mathbf{z}_t \text{ iid } (0, \mathbf{I}_N)$$

- Under those equations:

$$Cov(\mathbf{r}_t \mid \mathfrak{I}_{t-1}) = \boldsymbol{\Sigma}_t^{1/2} Cov(\mathbf{z}_t \mid \mathfrak{I}_{t-1}) (\boldsymbol{\Sigma}_t^{1/2})' = \boldsymbol{\Sigma}_t$$

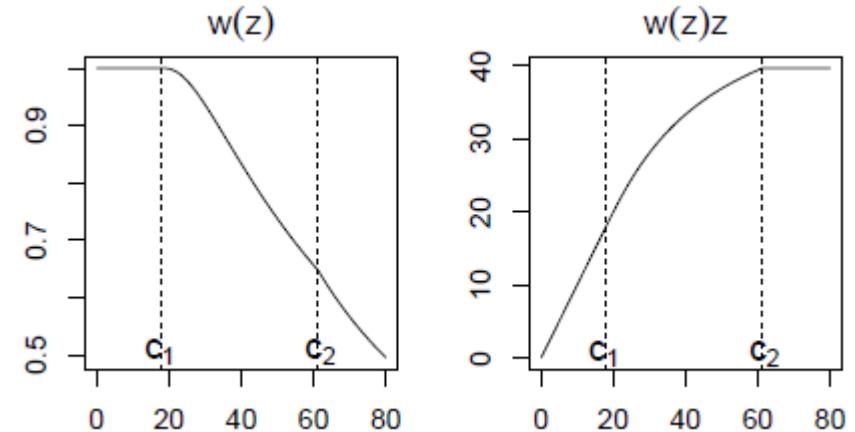
Flavor 1: BEKK

$$\boldsymbol{\Sigma}_t = \mathbf{C}'\mathbf{C} + \sum_{i=1}^p \mathbf{A}'_i \boldsymbol{\varepsilon}_{t-i} \boldsymbol{\varepsilon}_{t-i}' \mathbf{A}_i + \sum_{j=1}^q \mathbf{B}'_j \boldsymbol{\Sigma}_{t-j} \mathbf{B}_j$$

- EWMA with an intercept and parameter matrices
- Can be stationary (under conditions on the parameter matrices)
- Assumes unbounded, quadratic reaction to past de-means.

Solution: BIP-BEKK

$$\boldsymbol{\Sigma}_t = \mathbf{C}'\mathbf{C} + \sum_{i=1}^p w(\boldsymbol{\varepsilon}'_{t-i}\boldsymbol{\Sigma}_{t-i}^{-1}\boldsymbol{\varepsilon}_{t-i})\mathbf{A}'_i\boldsymbol{\varepsilon}_{t-i}\boldsymbol{\varepsilon}'_{t-i}\mathbf{A}_i + \sum_{j=1}^q \mathbf{B}'_j\boldsymbol{\Sigma}_{t-j}\mathbf{B}_j$$



- BIP stands for Bounded Innovation Propagation
- Uses a *conditional* outlier detection rule
- There is now an upper bound on the impact of single observations.
- Uses only daily close prices.

Solution: realized BEKK

$$\boldsymbol{\Sigma}_t = \boldsymbol{C}'\boldsymbol{C} + \sum_{i=1}^p \boldsymbol{A}_i' \boldsymbol{V}_{t-i} \boldsymbol{A}_i + \sum_{j=1}^q \boldsymbol{B}_j' \boldsymbol{\Sigma}_{t-j} \boldsymbol{B}_j$$

- \boldsymbol{V}_{t-i} is a realized covariance matrix computed from intraday data (e.g. using the `rCov()`, `rTSCov()`, `rRTSCov()` functions in the package `highfrequency`)

Flavor 2: GO-GARCH factor models

$$(4 - N \text{ factors}) \quad \varepsilon_t = A f_t$$

$$(5 - \text{orthogonal}) \quad \text{cov}(f_t | \mathfrak{I}_{t-1}) = V_t = \text{diag}\{\nu_{1t}^2, \nu_{2t}^2, \dots, \nu_{Nt}^2\}$$



$$\Sigma_t = A V_t A'$$

- Estimate μ_t and A . Compute factors using $f_t = A^{-1}(r_t - \mu_t)$. Specify a GARCH model for the variance of each factor ν_{it}^2 .
- If factors are assumed to be independent, you obtain the Chicago model with straightforward expressions for the coskewness and cokurtosis matrix:

$$\Phi_t = A \Phi_{f,t} (A' \otimes A')$$

$$\Psi_t = A \Psi_{f,t} (A' \otimes A' \otimes A')$$

where the factor coskewness and cokurtosis are sparse.

Functionality in the rmgarch package

```
1 library(rmgarch)
2 data("dji30ret")
3 cl<-makeCluster(4)
4 spec<-gogarchspec(mean.model=list(model="AR"), lag=1,
5 variance.model=list(model="eGARCH", variance.targeting=TRUE),
6 distribution.model="manig", ica="fastica")
7 model<-gogarchfit(spec, data=dji30ret[,1:4], cluster=cl)
8 stopCluster(cl)
```

Snippet 1: GO-GARCH Example

Functions/Methods	Description	Input Classes
gogarchspec	model specification	NA
gogarchfit	model estimation	1
gogarchforecast	1- to n-ahead forecasts	2
gogarchsim	simulation	2,3
gogarchfilter	1-ahead ahead filtering	2,3
convolution	calculates the weight density by FFT	2,3,4,5,6
fitted	conditional mean equation fitted/forecasted values	2,3
residuals	conditional mean equation residuals	2,3
coef	coefficients of model	2,3
show	summary of output	2,3,4,5,6
nisurface	news impact surface	2,3
gportmoments	geometric portfolio moments	2,3,4,5,6
rccor	conditional correlations	2,3,4,5,6
rcov	conditional covariance	2,3,4,5,6
sigma	conditional margin volatilities	2,3,4,5,6
reoskew	conditional coskewness	2,3,4,5,6
rcokurt	conditional cokurtosis	2,3,4,5,6
betacovar	conditional covariance beta to a benchmark	2,3
betacoskew	conditional coskewness beta to a benchmark	2,3
betacokurt	conditional cokurtosis beta to a benchmark	2,3
dfft	FFT density function (interpolated)	7
pfft	FFT distribution function (interpolated)	7
qfft	FFT quantile function (interpolated)	7
nportmoments	first 4 conditional portfolio moments from FFT interpolated density using quadrature integration	7
gogarchroll	rolling estimation/forecasting	1

Note: The Table provides a list of the methods and functions for working with GO-GARCH models based on the ICA transformation in the **rmgarch** package. The Input Classes are as follows: 1=goGARCHspec, 2=goGARCHfit, 3=goGARCHfilter, 4=goGARCHforecast, 5=goGARCHsim, 6=goGARCHroll, 7=goGARCHfft, NA=not a method.

Flavor 3: DCC models

- Model the conditional covariance matrix as the product of conditional volatilities and correlation, with a univariate GARCH model for the variance and a multivariate model for the correlation.

$$\begin{pmatrix} \sigma_{1t}^2 & \sigma_{12t} & \cdots & \sigma_{N1t} \\ \sigma_{21t} & \sigma_{2t}^2 & \cdots & \sigma_{2Nt} \\ \vdots & \ddots & \ddots & \vdots \\ \sigma_{N1t} & \sigma_{N1t} & \cdots & \sigma_{Nt}^2 \end{pmatrix} = \begin{pmatrix} \sigma_{1t} & 0 & \cdots & 0 \\ 0 & \sigma_{2t} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{Nt} \end{pmatrix} \begin{pmatrix} 1 & \rho_{12t} & \cdots & \rho_{N1t} \\ \rho_{21t} & 1 & \cdots & \rho_{2Nt} \\ \vdots & \ddots & \ddots & \vdots \\ \rho_{N1t} & \rho_{N2t} & \cdots & 1 \end{pmatrix} \begin{pmatrix} \sigma_{1t} & 0 & \cdots & 0 \\ 0 & \sigma_{2t} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{Nt} \end{pmatrix}$$

$$\boldsymbol{\Sigma}_t = \mathbf{D}_t \mathbf{R}_t \mathbf{D}_t$$

Flavor 3: DCC models

- Step 1: Estimate the garch models and compute standardized returns

$$\mathbf{z}_t = \mathbf{D}_t^{-1}(\mathbf{r}_t - \boldsymbol{\mu}_t)$$

- Step 2: Estimate the GARCH covariance of the standardized returns

$$\mathbf{Q}_t = (1 - a - b)\bar{\mathbf{Q}} + a \mathbf{z}_{t-1} \mathbf{z}'_{t-1} + b\mathbf{Q}_{t-1}$$

- Step 3: Scale \mathbf{Q}_t into a correlation matrix

$$\mathbf{R}_t = \text{diag}(\mathbf{Q}_t)^{-1/2} \mathbf{Q}_t \text{diag}(\mathbf{Q}_t)^{-1/2}$$

Functionality in the rmgarch package

```
1 library(rmgarch)
2 data("dji30ret")
3 cl<-makeCluster(4)
4 uspec = multispec(list(
5 ugarchspec(mean.model=list(armaOrder=c(1,0)),variance.model=
  list(model="eGARCH")),
6 ugarchspec(mean.model=list(armaOrder=c(1,1)),variance.model=
  list(model="sGARCH")),
7 ugarchspec(mean.model=list(armaOrder=c(2,0)),variance.model=
  list(model="gjrGARCH")),
8 ugarchspec(mean.model=list(armaOrder=c(1,0)),variance.model=
  list(model="csGARCH"))
))
9 spec<-dccspec(uspec, model = "DCC", distribution="mvnrm")
10 model<-dccfit(spec, data=dji30ret[,1:4])
11 stopCluster(cl)
```

Snippet 2: DCC Example

Functions/Methods	Description	Input Classes
dccspec	model specification for the univariate GARCH models, conditional mean, joint dynamics and distribution	NA
dccfit	model estimation	1
dccforecast	1- to n-ahead forecasts	2
dccsim	simulation	1,2
dccfilter	1-ahead ahead filtering	2,3
fitted	conditional mean equation fitted/forecasted values	2,3,4,5,6
residuals	conditional mean equation residuals	2,3
coef	coefficients of model	2,3
show	summary of output	2,3,4,5,6
nisurface	news impact surface	2,3
rcor	conditional correlations	2,3,4,5,6
rcov	conditional covariance	2,3,4,5,6
sigma	conditional margin volatilities	2,3,4,5,6
dccroll	rolling estimation/forecasting	1

Note: The Table provides a list of the methods and functions for working with DCC models in the **rmgarch** package. The Input Classes are as follows: 1=DCCspec, 2=DCCfit, 3=DCCfilter, 4=DCCforecast, 5=DCCsim, 6=DCCroll, NA=not a method. Distributions allowed as Multivariate Normal, Laplace and Student (QML), with a number of different options for the first stage GARCH model dynamics, and the asymmetric or symmetric DCC, or the flexible DCC model of Billio et al. (2006) for the joint dynamics.

Flavor 4: GAS models

- You stack all the time-varying parameters into a vector

$$\boldsymbol{\theta}_t = (\mu_{1t}, \dots, \mu_{Nt}, \sigma_{1t}, \dots, \sigma_{Nt}, \rho_{21t}, \dots, \rho_{(N-1)Nt}, \dots)'$$

- Transform using a link function $\Lambda()$ into an unconstrained parameter

$$\tilde{\boldsymbol{\theta}}_t = \Lambda(\boldsymbol{\theta}_t)$$

- Compute the marginal impact of $\tilde{\boldsymbol{\theta}}_t$ on the log-density of the last observation

$$s_t = \frac{\partial \log f(\mathbf{r}_t | \boldsymbol{\theta}_t)}{\partial \tilde{\boldsymbol{\theta}}_t}$$

- Apply update equation:

$$\tilde{\boldsymbol{\theta}}_t = \kappa + \mathbf{A} s_{t-1} + \mathbf{B} \tilde{\boldsymbol{\theta}}_{t-1}$$

$$\boldsymbol{\theta}_t = \Lambda^{-1}(\tilde{\boldsymbol{\theta}}_t)$$

This is the score. If it is positive, the increasing $\tilde{\boldsymbol{\theta}}_t$ improves the fit. If it is negative, then decreasing $\tilde{\boldsymbol{\theta}}_t$ improves the fit.

Functionality in the GAS package

```
1 library("GAS")
2 data("dji30ret", package = "GAS")
3 mGASSpec <- MultiGASSpec(Dist = "mvt",
4 ScalingType = "Identity",
5 GASPar = list(scale = TRUE, correlation = TRUE))
6 model <- MultiGASFit(data = dji30ret[, 1:4],
7 GASSpec = mGASSpec)
```

Snippet 3: GAS Example

Functions/Methods	Description	Input Classes
MultiGASSpec	model specification	NA
MultiGASFit	model estimation	1
getFilteredParameters	1-ahead ahead filtering	1
ConfidenceBands	onfidence bands for the filtered parameters	1
getMoments	Extract conditional moments	1, 3, 5
getForecast	Extract parameter forecast	3, 5
LogScore	Extract log scores	3, 5
MultiGASFor	1- to h -ahead forecasts	2
MultiGASSim	simulation	2
residuals	conditional mean equation residuals	2
coef	coefficients of model	2
show	summary of output	3,4,5
summary	summary of output	2
MultiGASRoll	rolling estimation/forecasting	1

Note: The Table provides a list of the methods and functions for working with multivariate GAS models in the **GAS** package. The Input Classes are as follows: 1= mGASSpec, 2= mGASFit, 3= mGASFor, 4=mGASSim, 5=mGASRoll.

Outline

1. Use of filters to estimate the parameters of the multivariate return distribution
2. MGARCH models for returns
3. Illustration on predicting the distribution of the return of an equally-weighted portfolio of 10 Vanguard funds.
4. Outlook

```
symbols <-
c("VFINX", "VEURX", "VPACX", "VEIEX", "VGSIX", "VGENX", "VGPMX", "VBLTX", "VMNFX", "VWESX")
# Vanguard 500 Index Investor
# Vanguard European Stock Index Investor
# Vanguard Pacific Stock Index Investor
# Vanguard Emerging Mkts Stock Index Investor
# Vanguard Real Estate Index Investor
# Vanguard Energy Fund Investor Shares
# Vanguard Precious Metals and Mining Investor
# Vanguard Long-Term Bond Index Investor
# Vanguard Market Neutral Investor
# Vanguard Long-Term Investment-Grade Investor

# Get the fund close prices from Yahoo!Finance
library(quantmod)
dat <- do.call(cbind, lapply(1:length(symbols), function(i){
  tmp<-getSymbols(symbols[i], from="2000-01-01",to="2018-07-30",auto.assign = FALSE)
  tmp <-adjustOHLC(tmp, symbol.name = symbols[i])
  tmp <- Cl(tmp)
  colnames(tmp) <- symbols[i]
  return(tmp)
}))
```

```

# Compute the log-returns
ret <- do.call(cbind, lapply(1:ncol(dat), function(i){
  tmp <- quantmod:::allReturns(dat[,i], type = "log") [,1]
  colnames(tmp) <- symbols[i]
  return(tmp)
} ))
ret = na.omit(ret)

# Split-sample analysis (see the book chapter for rolling analysis)
N <- ncol(ret)
train_sample <- "/2006-12-31"
test_sample <- "2007/2018"

# GO-GARCH model specification, estimation on trained sample, prediction on full sample
library(rmgarch)
spec.gogarch <- gogarchspec(mean.model=list(model="constant"),
  variance.model = list(model="csGARCH", garchOrder=c(1,1), variance.targeting=TRUE),
  distribution.model="manig", ica="fastica")
fit.gogarch <- gogarchfit(spec.gogarch, data=ret[train_sample], gfun="tanh")
filter.gogarch <- gogarchfilter(fit.gogarch, ret, n.old = nrow(ret[train_sample]))

```

```

# Compute density and VaRs of portfolio return for the out-of-sample period
start <- which(index(ret)==as.Date("2006-12-29"))+1
end <- nrow(ret)
set.seed(1234)
cf <- convolution(filter.gogarch, weights = rep(1/N, N), trace=1, use.ff = FALSE,
support.method = c("adaptive"))
varMatrix.gogarch <- matrix(NA, ncol=2, nrow=end-start+1)
colnames(varMatrix.gogarch) <- c("VaR[1%]", "VaR[5%]")
cdraws <- 50000
density.gogarch <- matrix(NA, ncol=cdraws, nrow=end-start+1)
for(i in start:end) {
  f <- qfft(cf, index = i)
  varMatrix.gogarch[i-start+1,1] <- f(0.01)
  varMatrix.gogarch[i-start+1,2] <- f(0.05)
  density.gogarch[i-start+1,] <- f(runif(cdraws))
}
varMatrix.gogarch <- xts(varMatrix.gogarch, index(ret[test_sample]))
density.gogarch <- xts(density.gogarch, index(ret[test_sample]))

```

```

# Plot of time-varying density functions
library(ggplot2)
library(ggridges)

ggplot(plotdata.gogarch, aes(x = y, y = Q, fill = ..x..)) +
  geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01) +
  labs(title = 'Equal Weighted Portfolio Density (gogarch)') + guides(fill=FALSE) +
  theme(plot.title = element_text(color="#666666", face="bold", size=11, hjust=0),
        axis.text = element_text(size = 7),
        axis.title.x=element_blank(),axis.title.y=element_blank()) +
  xlim(-0.06, 0.06)

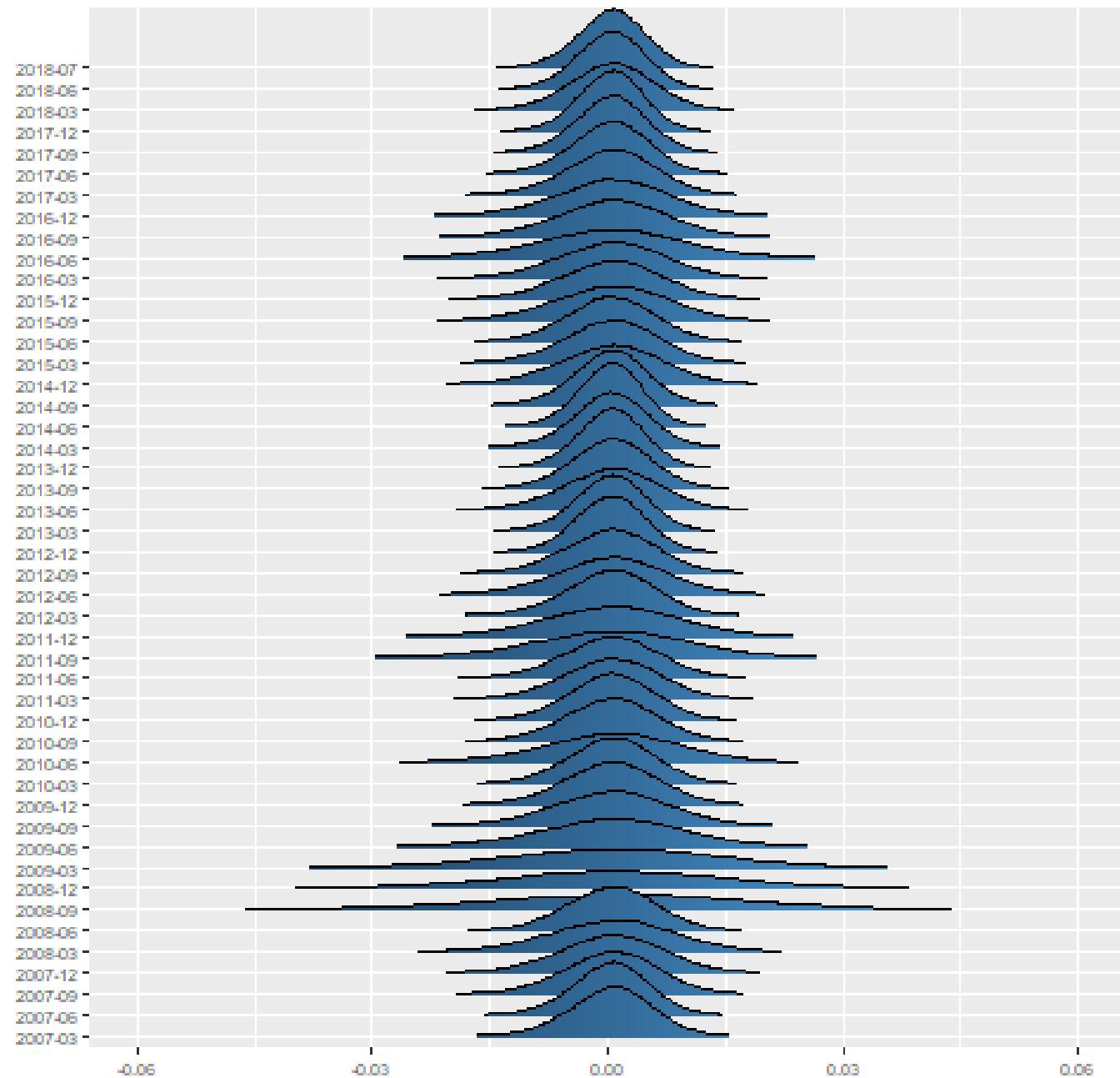
# Plot of VaR exceedances

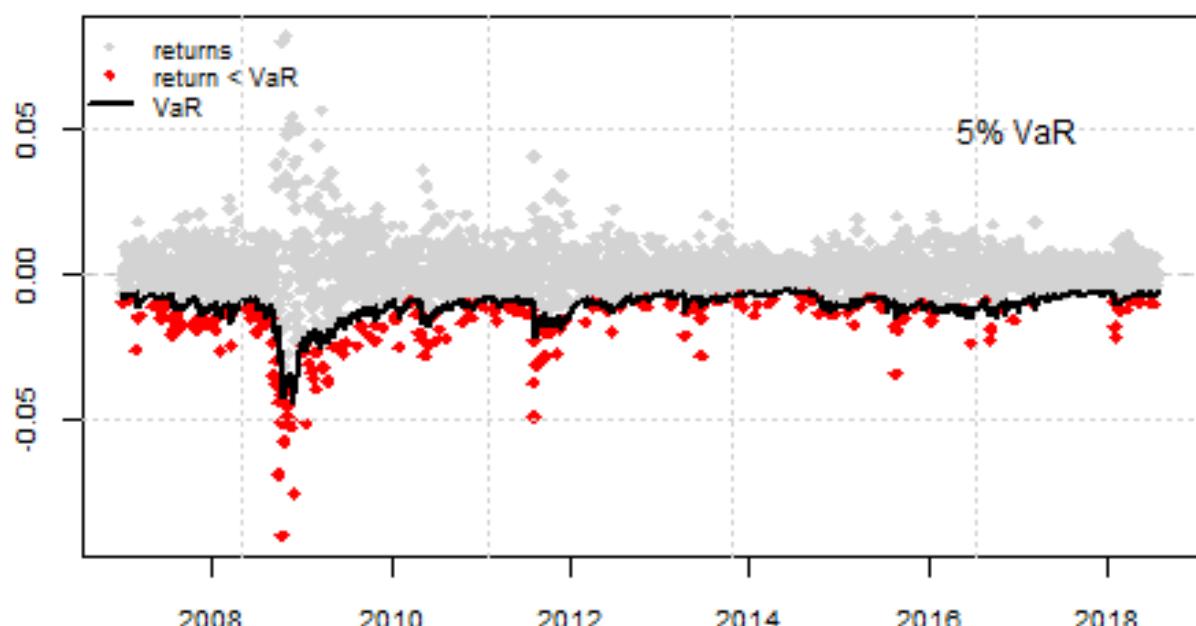
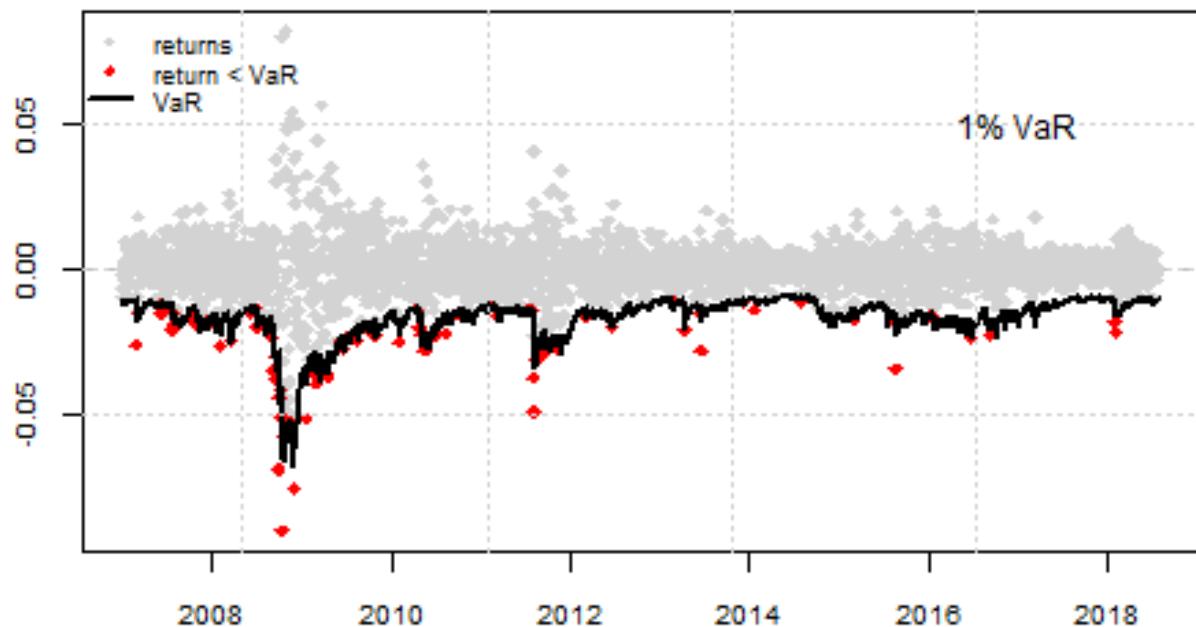
ewreturns <- xts(rowMeans(ret), index(ret))

par(mfrow=c(2,1),mar=c(2,3,1,2))
VaRplot(0.01, ewreturns[test_sample], varMatrix.gogarch[,1] )
text(x=as.Date("2017-01-01"),y=0.05,labels="1% VaR")
VaRplot(0.05, ewreturns[test_sample], varMatrix.gogarch[,2] )
text(x=as.Date("2017-01-01"),y=0.05,labels="5% VaR")

```

Equal Weighted Portfolio Density (gogarch)

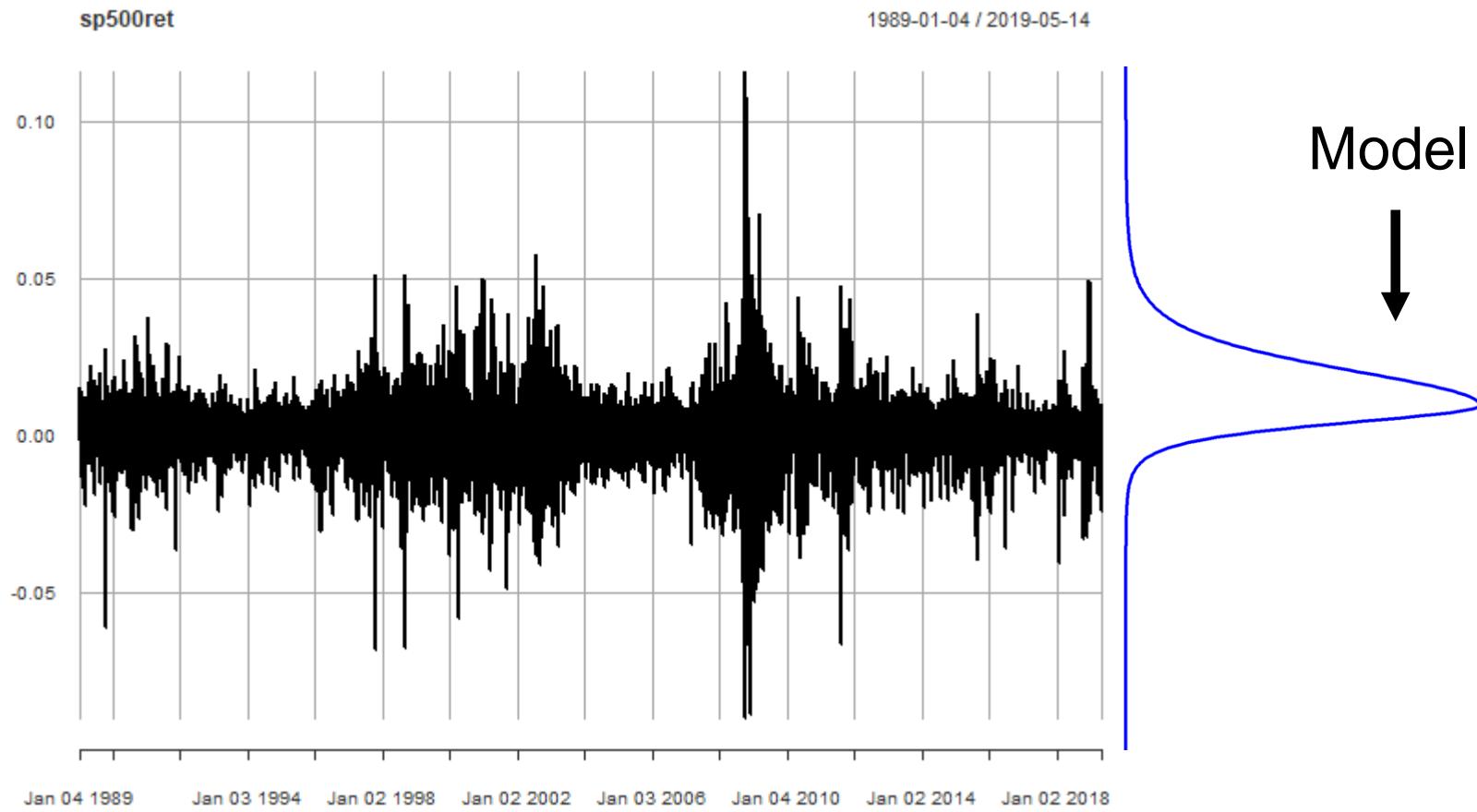




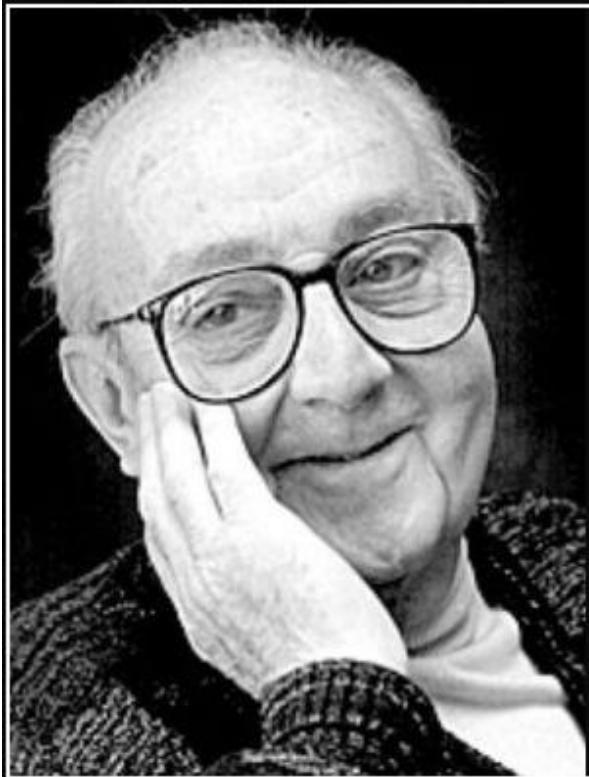
Outline

1. Use of filters to estimate the parameters of the multivariate return distribution
2. MGARCH models for returns
3. Illustration on predicting the distribution of the return of an equally-weighted portfolio of 10 Vanguard funds.
4. Outlook

Predictive financial return modelling



MGARCH models versus filters: Is there a difference?



All models are wrong, but some are useful.

— George E. P. Box —

AZ QUOTES

9 Conclusion

Challenging to capture all empirical properties in the return data → flexible models

Flexible models are not always reliable in practice

Call for more flexible and reliable models in high dimensions, in R.

The joint analysis of more than ten financial return series is a challenging empirical task. It needs to take into account the non-normality of the return series, their time-varying volatility and correlation, and avoid the curse of dimensionality while preserving sufficient flexibility in order to avoid severe model misspecification. This chapter has provided an overview of recent advances in the field, and, when available, their implementation in **R** packages.

We end the overview chapter on feasible MGARCH models with a call for more research on feasible MGARCH models. Our field experience is that most of the large financial institutions are not willing to put a flexible MGARCH model into production. They still base their risk models on the EWMA model or some modification of it, since the simple EWMA model tends to be more reliable than the flexible ones in practice. An important research direction is thus to put reliability as a primary objective when developing the next generation of flexible MGARCH models. We believe that publishing the code using open source software like **R** will help in achieving this objective of reliability, since open sourcing allows code to be more actively vetted by a large community whose feedback helps in developing better models.