APPLICATION DE LA LIBRAIRIE "NSE" EN GESTION DES RISQUES FINANCIERS

Keven Bluteau
Université de Neuchâtel & sentometrics
David Ardia

R à Québec 2019

MOTIVATION AND EXAMPLES

NUMERICAL STANDARD ERROR

- NSE is the standard deviation of a simulation result, if the simulation experiment were to be repeated many times.

- Consider the expectation of a scalar function $\mathrm{E}(g(\boldsymbol{X}))$ where $g(\boldsymbol{X})$ is estimated by generating pseudo-random draw x_i $(i = 1,2,\dots,n)$.

- The adequate estimator of the mean would be:

$$\hat{\mu} = \sum_{i=1}^{n} g(x_i)/n$$

- If the x_i $(i = 1,2,\dots,n)$ are $i.i.d.$, the NSE of $\hat{\mu}$ would simply be the standard deviation of $g(\boldsymbol{X})$ divided by $\sqrt{n}$.

NUMERICAL STANDARD ERROR

- Problems arise when the x_i are **not $i.i.d$.**

- Many time-series or simulation method (such as Markov chain Monte Carlo) **exhibit high autocorrelation**.

- As such, the *naive* method would **overstate the precision of the estimate** (that is, understate the NSE).

- One could possibly **repeat the simulation experiment** several time and compute the standard deviation of $\hat{\mu}$**.**

- This however might be impossible or **prohibitive because of computation time**.

EXAMPLE AR(1) PROCESS

- Suppose you have a AR(1) process:

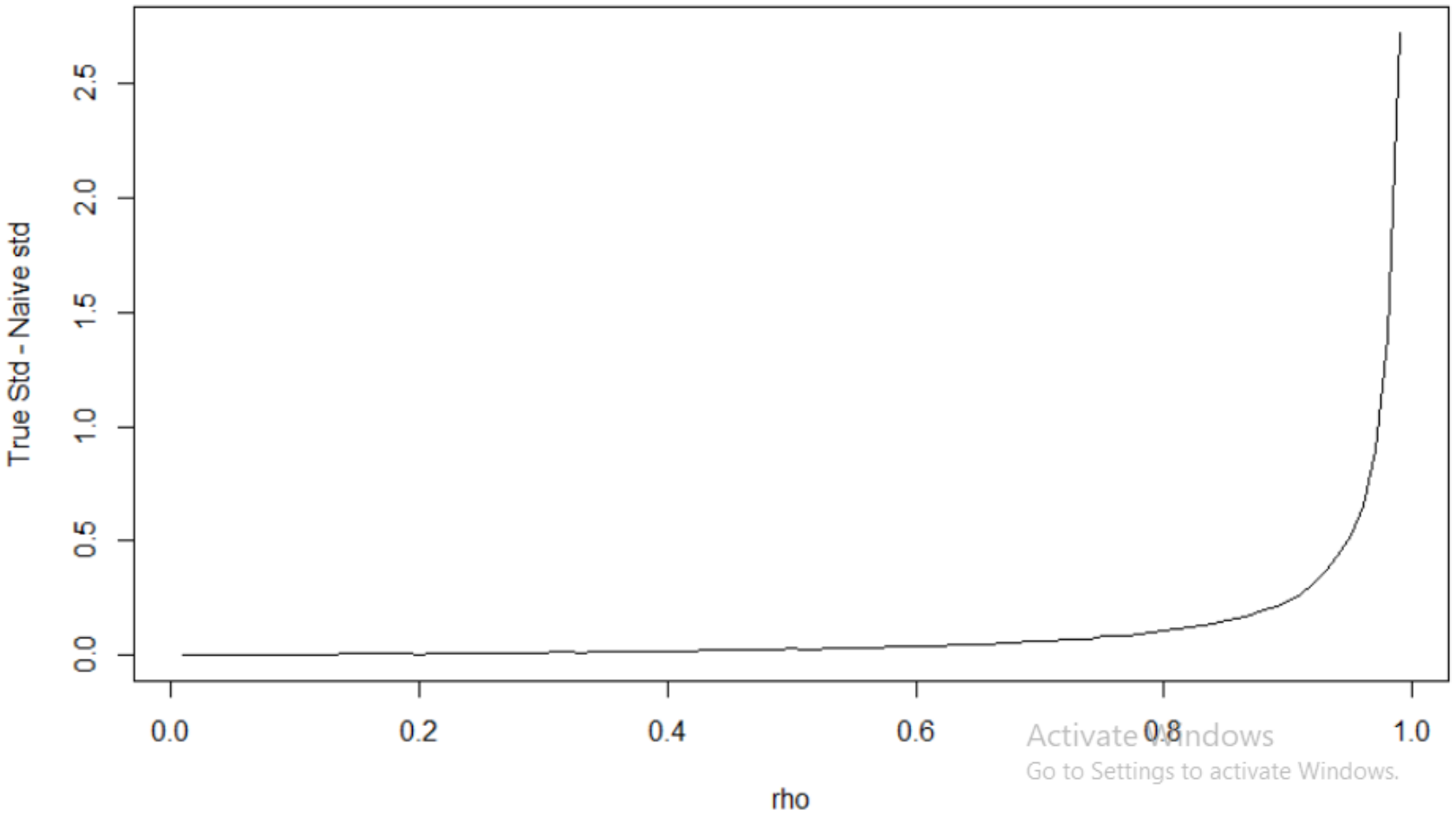
$$y_t = \alpha + \rho y_{t-1} + \epsilon_t \qquad \epsilon_t \sim N(0,1)$$

- You want to estimate μ_y as well as the accuracy of the estimate, that is, σ_{μ_y}.

- Without previous knowledge of the process, we estimate naively σ_{μ_y} by taking the standard deviation of y_t divided by $\sqrt{n}$.

- **Would that be a good estimate ?**

EXAMPLE AR(1) PROCESS



EXAMPLE MARKOV-SWITCHING PROCESS

- Suppose you have a process:

$$y_t | \, s_t = k \sim N(\alpha_k, 1)$$

- $\alpha_1 = -5$, $\alpha_2 = 5$, and s_t is a Discrete-state variable that evolves according to a first-order Markov chain with transition matrix $\boldsymbol{P}$:

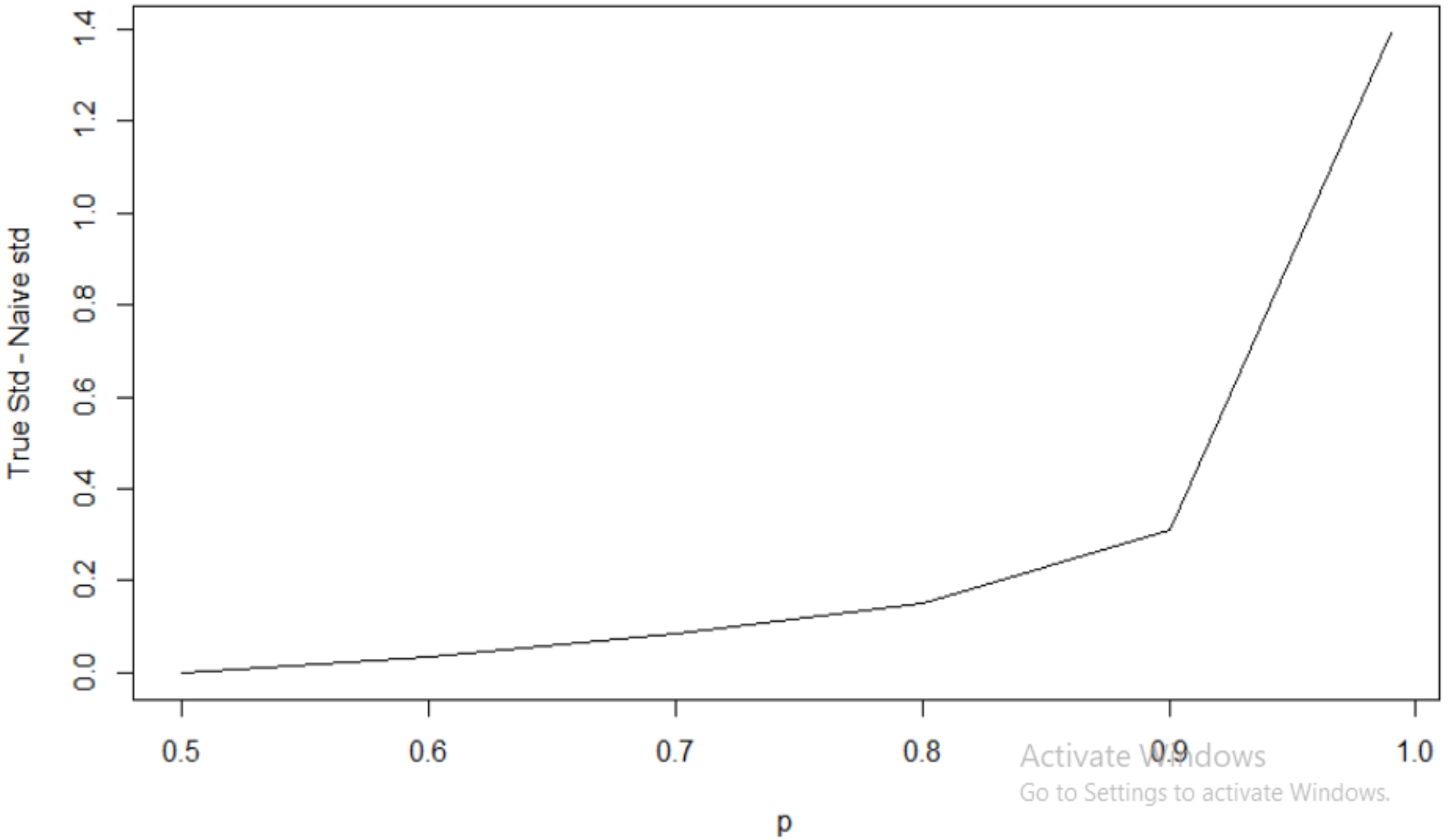
$$\boldsymbol{P} = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}$$

- You want to estimate μ_y as well as the accuracy of the estimate σ_{μ_y}

- Without previous knowledge of the process, we estimate naively σ_{μ_y} by taking the standard deviation of y_t divided by $\sqrt{n}$.

- **Would that be a good estimate ?**

EXAMPLE MARKOV-SWITCHING PROCESS



Activate Windows
Go to Settings to activate Windows.

MCMC ESTIMATION OF VARIANCE MODELS

- Some volatility models like Markov-switching GARCH model are better estimated using simulation—based **Markov Chain Monte Carlo** techniques.

http://keblu.github.io/MSGARCH/

```
library(MSGARCH)

data(SMI)

spec <- CreateSpec(variance.spec = list(model = c("gjrGARCH","gjrGARCH")),
                   distribution.spec = list(distribution = c("sstd","sstd")))

set.seed(1234)
fit <- FitMCMC(spec = spec,
               data = SMI,
               ctr  = list(nthin = 1, nburn = 5000, nmcmc = 10000))
```

MCMC ESTIMATION OF VARIANCE MODELS

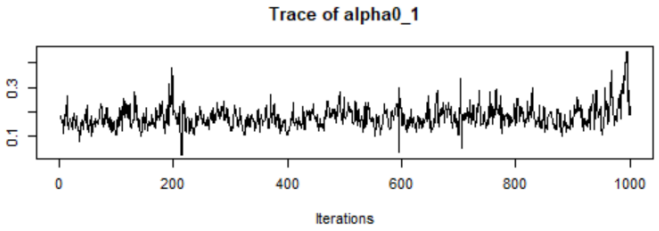
- Using MCMC estimation, we have now **generated multiple sample from the multivariate posterior distribution** of the Markov-Switching GARCH model parameters.

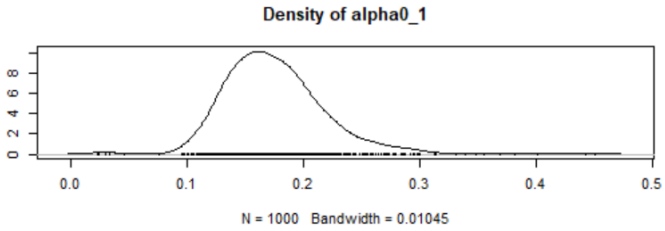
```
head(round(fit$par, 4), 10)
```

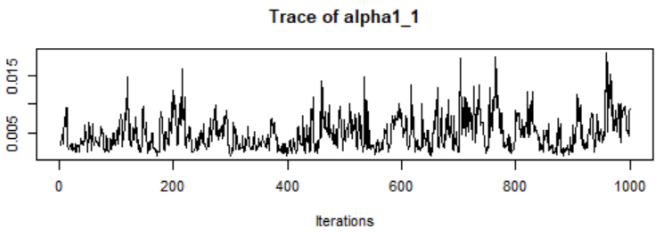
```
Markov Chain Monte Carlo (MCMC) output:
Start = 1
End = 10
Thinning interval = 1
      alpha0_1 alpha1_1 alpha2_1 beta_1  nu_1  xi_1 alpha0_2 alpha1_2 alpha2_2 beta_2   nu_2  xi_2 P_1_1 P_2_1
 [1,]    0.179    0.003    0.247  0.669 4.258 0.796    0.077    0.007    0.113  0.892 24.572 0.821 0.985 0.017
 [2,]    0.172    0.003    0.254  0.675 4.312 0.830    0.081    0.006    0.138  0.883 29.774 0.846 0.987 0.018
 [3,]    0.174    0.003    0.196  0.675 3.993 0.821    0.072    0.008    0.148  0.878 32.542 0.856 0.984 0.017
 [4,]    0.164    0.004    0.275  0.654 4.667 0.844    0.051    0.004    0.115  0.913 24.486 0.866 0.985 0.018
 [5,]    0.156    0.005    0.249  0.656 4.823 0.795    0.041    0.006    0.090  0.926 22.374 0.854 0.981 0.016
 [6,]    0.146    0.004    0.265  0.654 4.399 0.856    0.057    0.005    0.129  0.905 28.234 0.799 0.985 0.017
 [7,]    0.109    0.003    0.229  0.722 5.593 0.825    0.048    0.007    0.100  0.917 40.312 0.864 0.980 0.015
 [8,]    0.148    0.004    0.135  0.707 4.552 0.872    0.033    0.010    0.107  0.912 35.473 0.849 0.972 0.016
 [9,]    0.187    0.008    0.097  0.702 4.260 0.821    0.017    0.010    0.110  0.925 32.378 0.791 0.958 0.011
[10,]    0.140    0.007    0.207  0.700 4.371 0.826    0.024    0.005    0.093  0.932 22.809 0.865 0.977 0.012
```

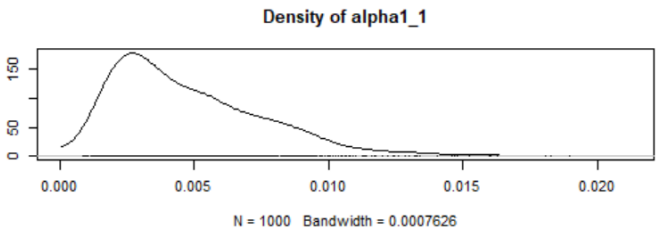
MCMC ESTIMATION OF VARIANCE MODELS

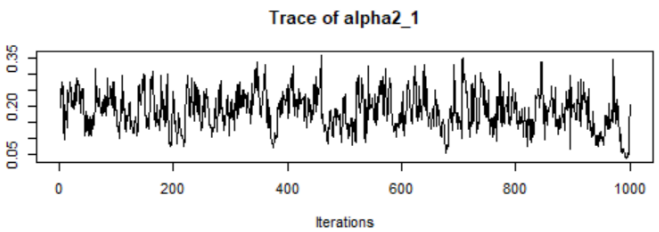
```
plot(fit$par)
```

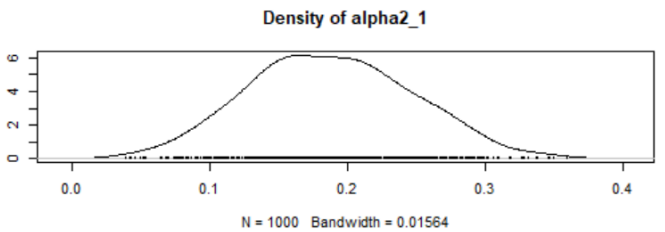


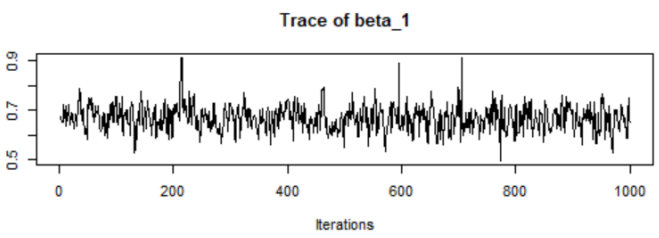


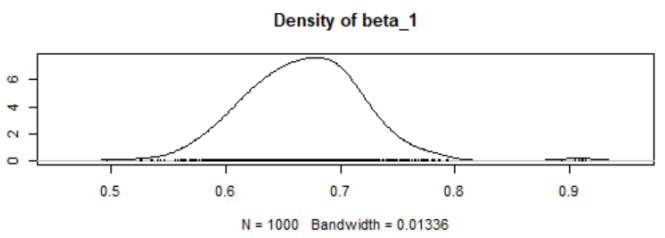












VALUE-AT-RISK ESTIMATION

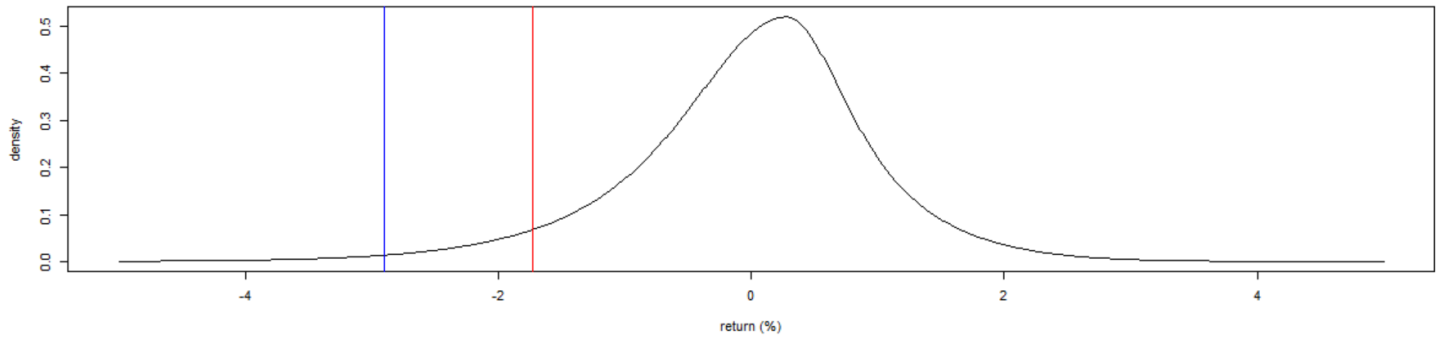
- Now we are interested in computing the **Value-at-Risk** a time $T+1$ (after the end of the training sample). This is useful for **financial risk management**.

- The Value-at-Risk at time $T+1$ is just a **percentile of the** $T+1$ **predicted distribution of the returns** given the information available at time T.

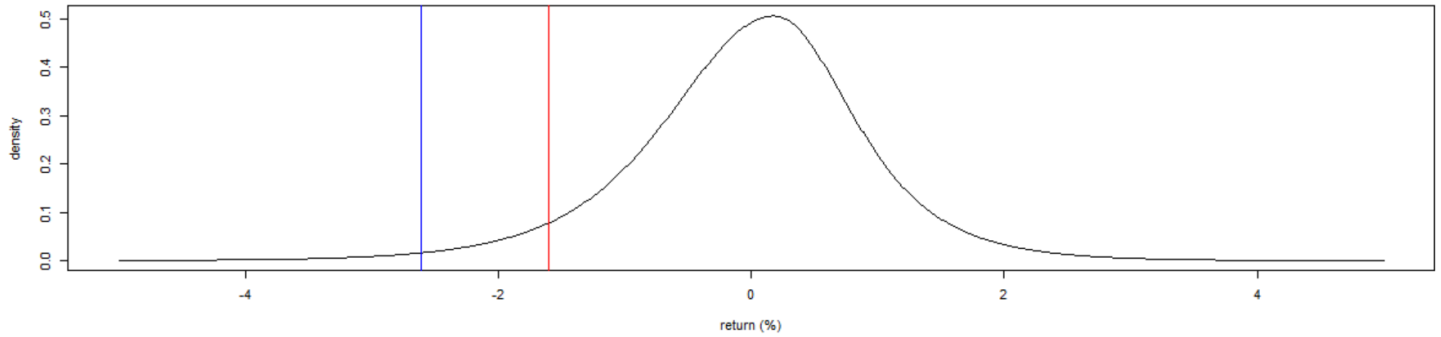
```
x <- seq(-5, 5,length.out = 1000)
i <- 1
density_pred <- PredPdf(spec,par = fit$par[i,],data = SMI, x = x)

VaR <- Risk(spec,par = fit$par[i,],
                 data = SMI,alpha = c(0.05,0.01),
                 nahead = 1,
                 do.es = FALSE)$VaR

plot(x = x, y = as.vector(density_pred), xlab = "return (%)", ylab =
"density", type = "l")
abline(v = VaR[,1], col = "red")
abline(v = VaR[,2], col = "blue")
```

VALUE-AT-RISK ESTIMATION

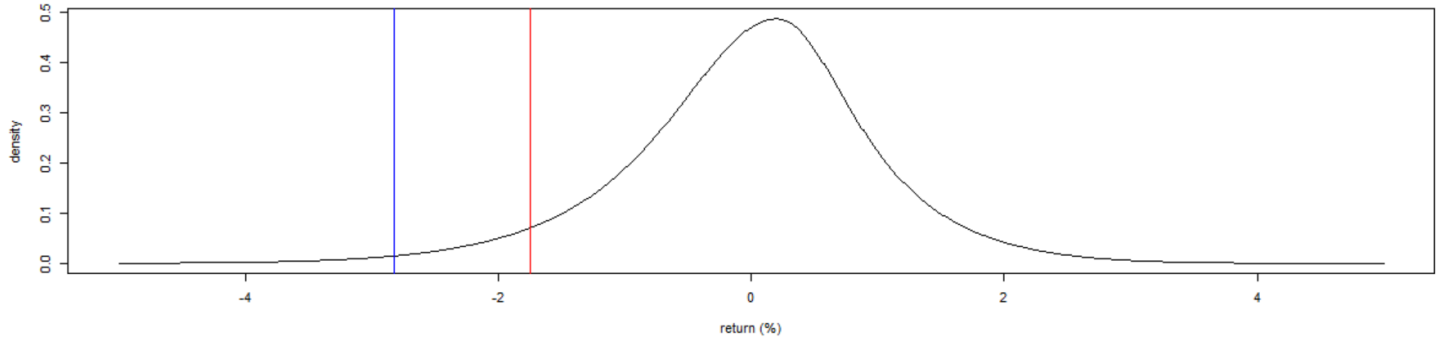
$i = 1$



$i = 300$



$i = 700$



VALUE-AT-RISK ESTIMATION

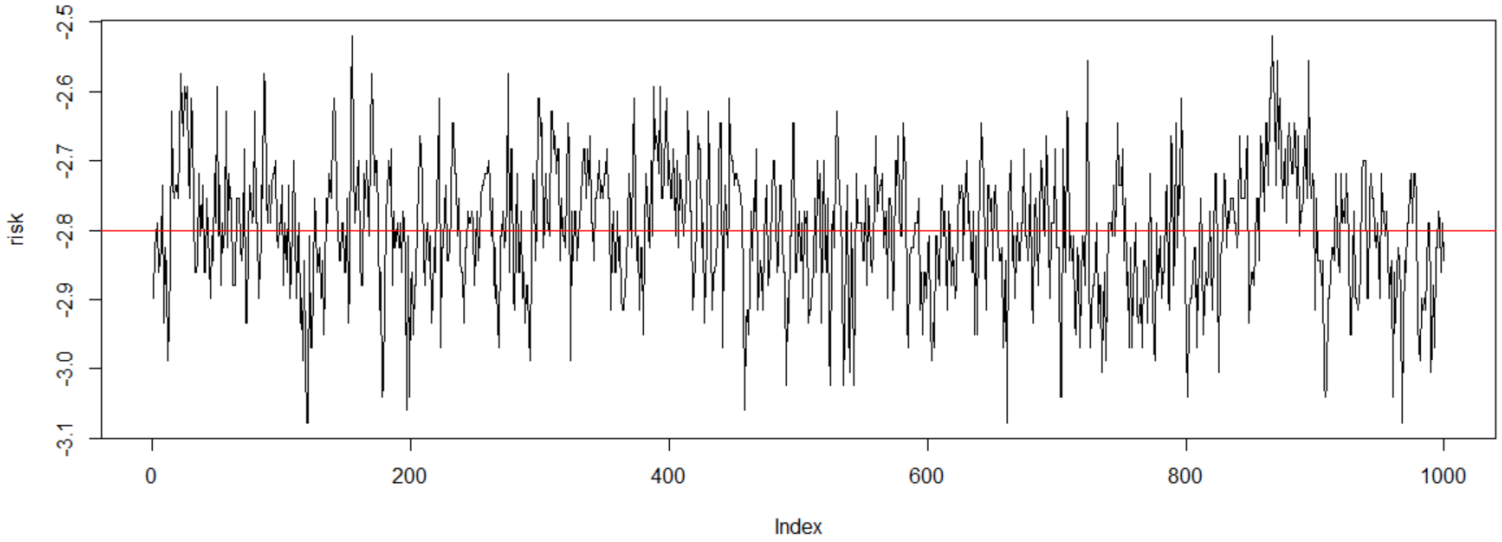
- We can thus get an **estimate of the Value-at-Risk for each of the sampled parameters** from the multivariate posterior distribution.

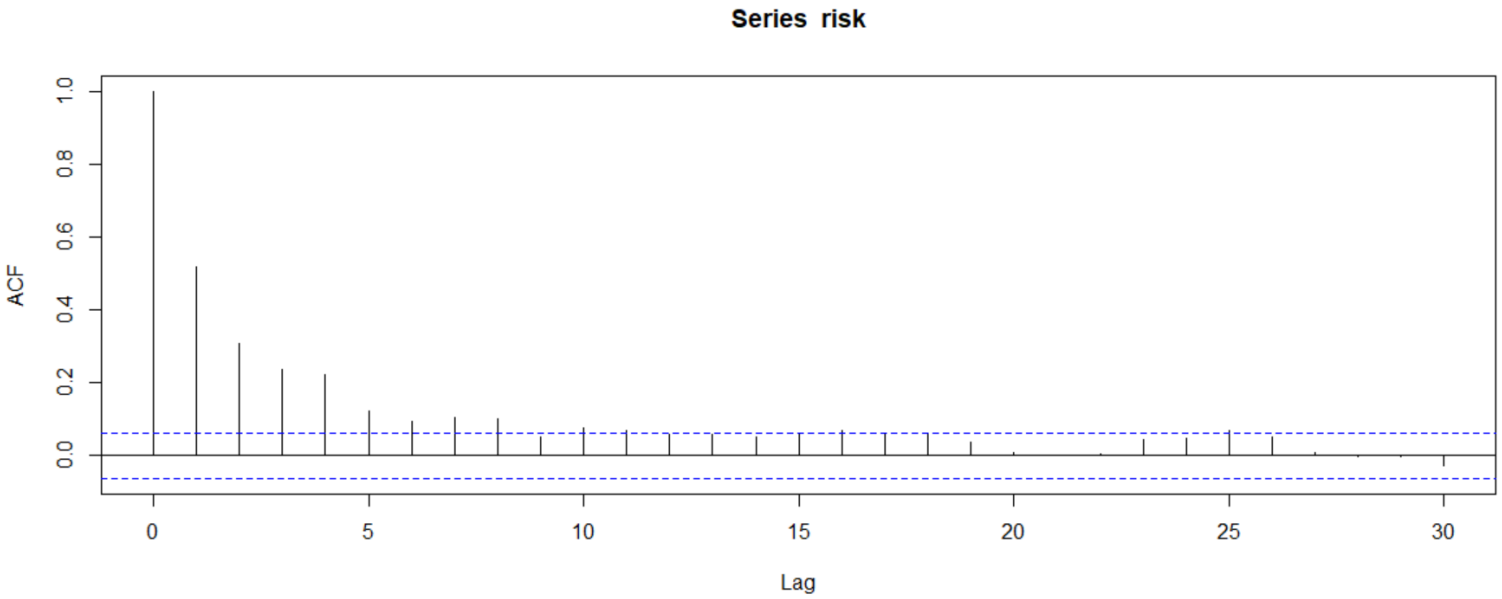
- We could compute the sample average get **the mean Value-at-Risk**. We could also compute it's **standard deviation to get the NSE, or the accuracy of that estimate**.

- **Would that be an adequate estimate ?**

```
par(mfrow = c(2,1))
risk <- rep(NA, nrow(fit$par))

for(i in 1:nrow(fit$par)){
  risk[i] <- Risk(spec,par = fit$par[i,],
                  data = SMI,
                  alpha = 0.01,
                  nahead = 1,
                  do.es = FALSE)$VaR
}

plot(risk, type = "l")
abline(h = mean(risk), col = "red")
acf(risk)
```

VALUE-AT-RISK ESTIMATION





ROBUST ESTIMATORS

http://dx.doi.org/10.2139/ssrn.2741587

DE GRUYTER Journal of Time Series Econometrics. 2018; 20170011

David Ardia[1] / Keven Bluteau[2,4] / Lennart F. Hoogerheide[3]

Methods for Computing Numerical Standard Errors: Review and Application to Value-at-Risk Estimation

[1] Institute of Financial Analysis, University of Neuchâtel, Neuchâtel, Switzerland; Department of Finance, Insurance and Real Estate, Laval University, Québec City, Canada; University of Neuchâtel, Rue A.-L. Breguet 2, CH-2000 Neuchâtel, Switzerland, E-mail: david.ardia@unine.ch
[2] Institute of Financial Analysis, University of Neuchâtel, Neuchâtel, Switzerland, E-mail: keven.bluteau@unine.ch
[3] Department of Econometrics and Tinbergen Institute, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, E-mail: l.f.hoogerheide@vu.nl
[4] Vrije Universiteit Brussel, Solvay Business School, Brussel, Belgium, E-mail: keven.bluteau@unine.ch

Abstract:
Numerical standard error (NSE) is an estimate of the standard deviation of a simulation result if the simulation experiment were to be repeated many times. We review standard methods for computing NSE and perform a Monte Carlo experiments to compare their performance in the case of high/extreme autocorrelation. In particular, we propose an application to risk management where we assess the precision of the value-at-risk measure when the underlying risk model is estimated by simulation-based methods. Overall, heteroscedasticity and autocorrelation estimators with prewhitening perform best in the presence of large/extreme autocorrelation.

Keywords: bootstrap, GARCH, HAC kernel, numerical standard error (NSE), Monte Carlo, Markov chain Monte Carlo (MCMC), spectral density, value-at-risk, Welch

DOI: 10.1515/jtse-2017-0011

https://CRAN.R-project.org/package=nse

nse: Computation of Numerical Standard Errors in R

David Ardia[1] **and Keven Bluteau**[1]

1 Institute of Financial Analysis - University of Neuchâtel

`nse: Numerical Standard Errors Computation in R`

Collection of functions designed to calculate numerical standard error (NSE) of univariate time series as described in Ardia et al. (2018) <doi:10.2139/ssrn.2741587> and Ardia and Bluteau (2017) <doi:10.21105/joss.00172>.

Version:	1.19
Imports:	Rcpp (≥ 0.12.0), coda, mcmc, mcmcse, np, sandwich
LinkingTo:	Rcpp
Suggests:	testthat
Published:	2018-09-13
Author:	David Ardia [aut], Keven Bluteau [aut, cre]
Maintainer:	Keven Bluteau <Keven.Bluteau at unine.ch>
BugReports:	https://github.com/keblu/nse/issues
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]
Copyright:	see file COPYRIGHTS
URL:	https://github.com/keblu/nse
NeedsCompilation:	yes
Citation:	nse citation info
Materials:	NEWS
In views:	Econometrics
CRAN checks:	nse results

GEYER ESTIMATORS

Bm	Batch means	G92
BmO	Overlapping bach means	G92
IsPo	Initial sequence, nonnegative constraint	G92
IsDe	Initial sequence, nonnegative, nonincreasing constraints	G92
IsCo	Initial sequence, nonnegative, nonincreasing,convex constraints	G92
IsPoBm	IsPo applied to batch means	G92
IsDeBm	IsDe applied to batch means	G92
IsCoBm	IsCo applied to batch means	G92

```
nse.geyer(x, type = c("iseq", "bm", "obm", "iseq.bm"), nbatch = 30,
  iseq.type = c("pos", "dec", "con"))
```

SPECTRAL DENSITY AT ZERO ESTIMATORS

SdAr	Parametric spectral density at zero, AR(q) fit	HW91
SdGl	Parametric spectral density at zero, GLM fit	HW91
SdBa	Nonparametric spectral density at zero, Bartlett kernel	FJ10
SdPa	Nonparametric spectral density at zero, Parzen kernel	FJ10
SdQs	Nonparametric spectral density at zero, Quadratic Spectral kernel	FJ10
SdBaWe	SdBa with Welch's smoothing	W67/PC06
SdPaWe	SdPa with Welch's smoothing	W67/PC06
SdQsWe	SdQs with Welch's smoothing	W67/PC06

```
nse.spec0(x, type = c("ar", "glm", "daniell", "modified.daniell",
  "tukey-hanning", "parzen", "triweight", "bartlett-priestley",
  "triangular", "qs"), lag.prewhite = 0, welch = FALSE,
  steep = FALSE)
```

HAC TYPE ESTIMATORS

`Nw`	Newey West	NW87
`NwPr`	`Nw` with prewhitening	NW94
`AnBa`	Andrews bandwidth, Bartlett kernel	A91
`AnBaPr`	`AnBa` with prewhitening	A91+AM92
`AnPa`	Andrews bandwidth, Parzen kernel	A91
`AnPaPr`	`AnPa` with prewhitening	A91+AM92
`AnQs`	Andrews bandwidth, Quadratic Spectral kernel	A91
`AnQsPr`	`AnQs` with prewhitening	A91+AM92
`HiBa`	Hirukawa bandwidth, Bartlett kernel	H10
`HiBaPr`	`HiBa` with prewhitening	H10+AM92
`HiPa`	Hirukawa bandwidth, Parzen kernel	H10
`HiPaPr`	`HiPa` with prewhitening	H10+AM92

```
nse.nw(x, lag.prewhite = 0)
```

```
nse.andrews(x, type = c("bartlett", "parzen", "tukey", "qs", "trunc"),
  lag.prewhite = 0, approx = c("AR(1)", "ARMA(1,1)"))
```

```
nse.hiruk(x, type = c("bartlett", "parzen"), lag.prewhite = 0)
```

BOOTSTRAP ESTIMATORS

`BsSt`	Stationary bootstrap, automatic block–length	PR94+PW04
`BsCi`	Circular bootstrap, automatic block–length	PR92+PW04
`BsStFx`	Stationary bootstrap, ad–hoc block–length	PR94
`BsCiFx`	Circular bootstrap, ad–hoc block–length	PR92

```
nse.boot(x, nb, type = c("stationary", "circular"), b = NULL,
  lag.prewhite = 0)
```

SIMULATION STUDY

SETUP

- AR(1) Case:
  - 1,000 simulation of length 100 and 1,000
  - Two cases: $\rho = 0.9$ and $\rho = 0.99$
  - Closed form solution for the true NSE for the mean is known

- MS Case:
  - 1,000 simulations of length 100 and 1,000
  - Two cases: $p = 0.9$ and $p = 0.99$
  - True NSE estimated by taking the standard deviation of the sample average over 10,000 simulations

- Value-at-Risk Case:
  - 1,000 estimations with MCMC chain of 100 or 1,000
  - True NSE estimated by taking the standard deviation of the sample average Value-at-Risk (at the 5% level) over 10,000 estimations

- For each of the 33 estimators, we compute the RMSE and the bias.

RESULTS

AR(1) RESULTS

	RMSE (×10)				Bias (×10)			
	$\rho = 0.90$		$\rho = 0.99$		$\rho = 0.90$		$\rho = 0.99$	
Method	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$
Nv	7.50	2.43	57.57	28.06	-7.49	-2.43	-57.56	-28.06
Bm	6.02	0.59	54.95	19.34	-5.97	-0.48	-54.91	-19.18
BmO	4.48	0.64	51.50	19.90	-4.22	-0.55	-51.34	-19.76
IsPo	3.78	0.55	47.17	12.24	-2.95	-0.13	-46.54	-9.65
IsDe	3.81	0.52	47.17	12.32	-3.00	-0.16	-46.54	-9.82
IsCo	3.87	0.50	47.44	12.59	-3.17	-0.20	-46.84	-10.35
IsPoBm	3.83	0.72	47.17	12.37	-2.98	-0.16	-46.55	-9.80
IsDeBm	3.84	0.71	47.18	12.45	-3.02	-0.16	-46.56	-9.90
IsCoBm	3.87	0.70	47.28	12.42	-3.08	-0.16	-46.68	-9.96
SdAr	3.67	0.46	44.80	11.38	-2.38	-0.11	-43.56	-7.40
SdGl	5.80	0.39	55.19	22.05	-5.75	-0.26	-55.16	-22.00
SdBa	4.49	1.26	51.14	14.74	-3.30	-0.54	-50.84	-11.34
SdPa	4.41	1.27	50.78	14.52	-3.02	-0.48	-50.46	-10.51
SdQs	4.59	1.36	51.43	15.30	-3.47	-0.73	-51.14	-12.09
SdBaWe	6.65	0.69	57.71	21.48	-6.61	-0.39	-57.71	-21.40
SdPaWe	6.55	0.67	57.62	21.20	-6.52	-0.28	-57.61	-21.11
SdQsWe	6.75	0.78	57.83	21.80	-6.72	-0.54	-57.82	-21.72
Nw	4.83	0.78	52.25	20.91	-4.64	-0.73	-52.13	-20.81
NwPw	3.70	0.46	43.21	11.37	-2.08	-0.10	-40.95	-6.46
AnBa	4.45	0.63	49.53	14.32	-4.01	-0.47	-49.19	-12.92
AnBaPw	3.74	0.46	43.40	11.23	-2.08	-0.09	-41.28	-6.31
AnPa	4.45	0.57	51.26	14.37	-3.96	-0.36	-51.01	-12.83
AnPaPw	3.74	0.46	43.28	11.24	-2.06	-0.10	-41.07	-6.40
AnQs	4.32	0.56	50.72	13.95	-3.76	-0.36	-50.43	-12.20
AnQsPw	3.77	0.46	43.35	11.27	-2.07	-0.09	-41.20	-6.34
HiBa	4.82	0.65	50.17	15.44	-4.52	-0.55	-49.87	-14.58
HiBaPw	3.73	0.47	43.28	11.31	-2.04	-0.11	-41.06	-6.43
HiPa	4.92	0.98	49.80	15.95	-4.34	-0.64	-49.40	-14.18
HiPaPw	3.82	0.47	43.32	11.41	-2.10	-0.11	-40.98	-6.55
BsSt	4.81	0.70	50.91	16.34	-4.49	-0.55	-50.68	-15.81
BsCi	4.66	0.62	50.86	17.21	-4.35	-0.49	-50.63	-16.87
BsStFx	4.85	0.69	50.90	17.92	-4.54	-0.58	-50.67	-17.61
BsCiFx	4.51	0.66	51.00	19.98	-4.20	-0.58	-50.79	-19.84

MARKOV-SWITCHING RESULTS

	RMSE (×10)				Bias (×10)			
	$p = 0.90$		$p = 0.99$		$p = 0.90$		$p = 0.99$	
Method	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$
Nv	9.95	3.12	53.26	13.95	-9.95	-3.12	-53.21	-13.95
Bm	6.74	0.56	50.69	7.41	-6.71	-0.26	-50.49	-7.39
BmO	3.99	0.53	48.75	7.78	-3.64	-0.35	-47.75	-7.76
IsPo	3.63	0.52	47.79	3.80	-2.36	-0.02	-45.26	-2.43
IsDe	3.58	0.48	47.80	3.78	-2.41	-0.06	-45.26	-2.60
IsCo	3.61	0.44	47.82	3.88	-2.57	-0.10	-45.31	-2.85
IsPoBm	3.80	1.12	47.79	3.97	-2.57	-0.09	-45.28	-2.56
IsDeBm	3.84	1.11	47.80	4.00	-2.63	-0.11	-45.28	-2.66
IsCoBm	3.79	1.10	47.79	3.99	-2.65	-0.11	-45.28	-2.70
SdAr	3.14	0.40	47.01	2.94	-1.57	-0.08	-43.19	-1.63
SdGl	6.07	0.53	50.90	9.15	-6.04	0.41	-50.74	-9.14
SdBa	6.39	2.05	49.06	6.26	-2.93	-0.83	-47.95	-3.22
SdPa	6.67	2.04	49.04	6.47	-2.51	-0.77	-47.84	-2.88
SdQs	6.78	2.15	49.29	6.81	-3.62	-1.08	-48.26	-3.97
SdBaWe	7.30	0.91	53.85	8.32	-7.11	-0.30	-53.80	-8.15
SdPaWe	7.00	0.92	53.70	8.03	-6.78	-0.15	-53.64	-7.85
SdQsWe	7.63	1.00	54.00	8.62	-7.45	-0.56	-53.95	-8.46
Nw	4.85	0.64	48.40	8.53	-4.65	-0.57	-47.81	-8.52
NwPw	3.03	0.39	46.58	2.97	-1.34	-0.07	-41.70	-1.71
AnBa	4.50	0.59	48.43	5.65	-4.07	-0.46	-46.47	-5.50
AnBaPw	3.04	0.40	46.60	2.87	-1.49	-0.07	-41.73	-1.48
AnPa	4.30	0.49	48.22	4.90	-3.59	-0.31	-46.08	-4.64
AnPaPw	2.99	0.41	46.62	2.88	-1.46	-0.08	-41.80	-1.48
AnQs	4.08	0.48	48.09	5.10	-3.36	-0.33	-45.80	-4.91
AnQsPw	3.04	0.41	46.63	2.87	-1.45	-0.08	-41.82	-1.48
HiBa	5.11	0.63	48.61	6.00	-4.88	-0.54	-47.12	-5.88
HiBaPw	3.00	0.39	46.61	2.96	-1.33	-0.07	-41.74	-1.58
HiPa	5.85	1.16	48.79	4.95	-4.73	-0.58	-47.23	-3.93
HiPaPw	3.18	0.39	46.65	2.93	-1.49	-0.07	-41.88	-1.64
BsSt	5.42	0.73	49.34	5.80	-5.04	-0.54	-48.01	-5.60
BsCi	4.84	0.62	48.92	6.10	-4.44	-0.47	-47.52	-6.01
BsStFx	5.55	0.72	49.40	6.61	-5.04	-0.49	-48.09	-6.53
BsCiFx	4.44	0.59	48.74	7.86	-3.92	-0.42	-47.49	-7.85

VALUE AT RISK RESULTS

	RMSE ($\times 10^2$)		Bias ($\times 10^2$)	
	$n = 100$	$n = 1{,}000$	$n = 100$	$n = 1{,}000$
Nv	6.88	7.54	-6.88	-7.54
Bm	6.84	7.38	-6.84	-7.38
BmO	6.80	7.40	-6.80	-7.39
IsPo	6.74	7.17	-6.74	-7.16
IsDe	6.74	7.18	-6.74	-7.17
IsCo	6.74	7.19	-6.74	-7.17
IsPoBm	6.74	7.18	-6.74	-7.17
IsDeBm	6.74	7.18	-6.74	-7.17
IsCoBm	6.74	7.18	-6.74	-7.17
SdAr	6.70	7.11	-6.69	-7.09
SdGl	6.85	7.43	-6.85	-7.43
SdBa	6.81	7.29	-6.81	-7.28
SdPa	6.80	7.29	-6.80	-7.28
SdQs	6.81	7.31	-6.81	-7.30
SdBaWe	6.88	7.46	-6.88	-7.46
SdPaWe	6.88	7.46	-6.88	-7.46
SdQsWe	6.88	7.47	-6.88	-7.47
Nw	6.81	7.41	-6.81	-7.41
NwPw	6.64	7.08	-6.63	-7.06
AnBa	6.78	7.25	-6.77	-7.24
AnBaPw	6.64	7.08	-6.63	-7.05
AnPa	6.80	7.28	-6.80	-7.27
AnPaPw	6.64	7.08	-6.63	-7.06
AnQs	6.79	7.27	-6.79	-7.27
AnQsPw	6.64	7.08	-6.63	-7.05
HiBa	6.77	7.25	-6.77	-7.24
HiBaPw	6.64	7.07	-6.63	-7.04
HiPa	6.78	7.26	-6.77	-7.25
HiPaPw	6.63	7.07	-6.62	-7.05
BsSt	6.80	7.31	-6.79	-7.31
BsCi	6.79	7.34	-6.79	-7.33
BsStFx	6.80	7.35	-6.79	-7.35
BsCiFx	6.80	7.39	-6.79	-7.39

THANKS !

QUESTIONS ?

Slides available at
http://keblu.github.io/MSGARCH/