CAViaRを用いた ボラティリティの予測

東京理科大学　工学部　経営工学科
鈴木 直明

構成

1. はじめに
2. モデル
3. 事例研究
4. 考察
5. まとめ
6. 参考文献

付録

はじめに

- 近年，金融工学では，他の工学の分野で広く使用されていた時系列解析の手法を取り入れた，「金融時系列解析」という分野が注目されている．
  - 時系列解析とは

    時間とともに不規則に変動する現実のデータから本質的な動きを探り，必要な情報を抽出すること[5]．

研究背景

- ボラティリティ・クラスタリングの発見により金融時系列の解析が発展した.
  - ボラティリティクラスタリングとは

    大きな株価の変動があった後は，しばらく大きな変動が続き，小さな株価の変動があった後は小さな変動が続くこと.
- 金融時系列の解析の中で，広く使われているのは，GARCH（generalized autoregressive conditional hetroskedasticity：分散不均一）と呼ばれるモデルである.

研究目的

- GARCHモデルによる予測は分布の仮定に依存している[1].

時間とともに仮定した分布に狂いが生じると予測も間違ってしまうことになる

- **分布の仮定を必要としないモデルを用いてボラティリティの予測を行う.**
  - ボラティリティとは
    株価の収益率がどの程度ばらついたかを表す指標で, 株価の収益率の分散で表現される.

GARCH(p,q)モデル

- GARCHモデルは，標準偏差を求める(1.3)式に，標準偏差の自己回帰が組み込まれているところに特徴がある．（※自己回帰：過去の値を説明変数とした回帰のこと．）
- 株価の t 期の収益率を r_t とすると，GARCHモデルは次の(1)式で表される．

$$r_t = \mu + \varepsilon_t \qquad (1.1)$$

$$\varepsilon_t = \sigma_t u_t \qquad (1.2)$$

$$\sigma_t^2 = \alpha + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_i \sigma_{t-j}^2 \qquad (1.3)$$

ただし

$\mu = E[r_t]$，　ε_t：誤差項　，　σ_t：ボラティリティ　，

u_t：互いに独立で平均　0，分散　1 の分布を仮定する

$\alpha, \alpha_i, \beta_i$：パラメータ

GJRGARCHモデル

- ボラティリティの変動は，収益率の上がった直後よりも，収益率の下がった直後のほうが大きい．
- この事実をモデルに反映させているのが，GJR (Glosten=Jagannathan=Runkle) GARCH モデルである．
- GJRGARCHモデルは(1.3)の条件付分散を次のように表現しなおしたモデルである．

$$\sigma_t^2 = \alpha + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \gamma_i S_{t-i} \varepsilon_{t-i}^2 \sum_{j=1}^{q} \beta_i \sigma_{t-j}^2 \qquad (2)$$

$S_{t\text{-}i}$: もし $\varepsilon_{t-i} < 0$ ならば $S = 1$，その他のとき $S = 0$ となるダミー変数

γ_i : パラメータ

CAViaRモデル

- CAViaR（Conditional Autoregressive Value at Risk：条件付自己回帰VaR）[2]モデルの特徴
  - VaR（Value at risk）という手法をもとにしている
  - 分布の仮定を必要としない
  - 自己回帰モデル
    - 自己回帰モデルとは
      過去の実現値を説明変数とした回帰モデル.

VaR

- ある信頼区間の中で最悪の収益率を分位値を用いて表現する.
  - 分位点とは

    $Q(\theta)$ で表現する：数値を小さい順に並べたときに小さいほうから数えて総度数の θ%に当たる値 .

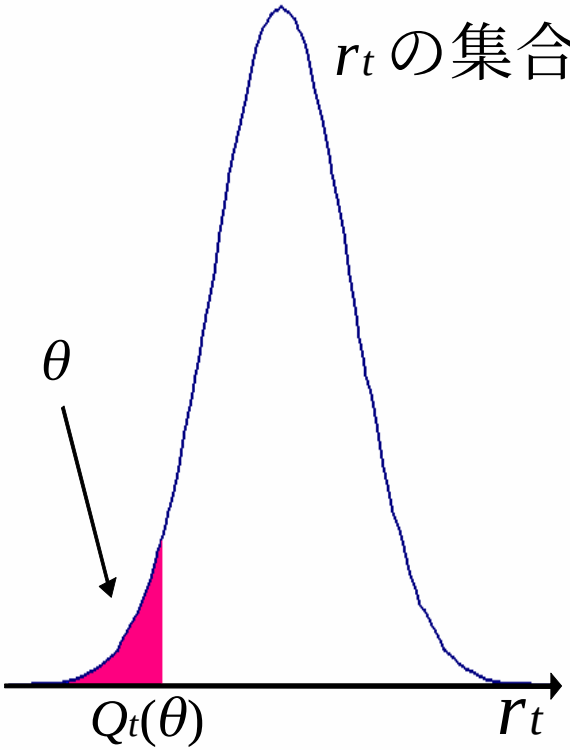

図1：VaRの概念図

AS CAViaR

- **本研究では，Asymmetric Slope CAViaR（以降AS CAViaRと表現）モデルを用いる．**
  - **予測において，予測に用いる過去の収益率がプラスかマイナスかによって結果が異なる．**
  - **予測に用いる過去の収益率がマイナスの方が，予測に大きな影響を与える．**

AS CAViaRのモデル式

- AS CAViaRは次の(3)式で定式化される.

$$Q_t(\theta) = \omega + \alpha Q_{t-1}(\theta) + \beta_1 (\varepsilon_{t-1})^+ + \beta_2 (\varepsilon_{t-1})^- \quad (3)$$

$Q_t(\theta)$ ：θによって与えられる分位点

ω , α , β_1 , β_2：パラメータ

$$(x)^+ = \max(x,0), \quad (x)^- = -\min(x,0)$$

$$\varepsilon_t = r_t - E(r_t \mid I_{t-1}) \quad (4)$$

- (4)式は t 時点の条件付期待値で，過去の情報 を用いて計算される.

分位点回帰最小化法

- CAViaRのパラメータの推定方法
  - (5)式を分位点回帰 (quantile regression) [3]と呼ぶ
  - 最も小さくするものをパラメータとして選ぶ[3]

$$\sum_{t|y_t \geq Q_t(\theta)} \theta \,|\, y_t - Q_t(\theta) \,| + \sum_{t|y_t < Q_t(\theta)} (1-\theta) \,|\, y_t - Q_t(\theta) \,| \quad (5)$$

t ：過去の実測値 *1* 時点から *t-1* 時点まで.

y_t ：実現値の集合である.

分析対象データ

- 内容
  - JASDAQIndex，米国S&Pindexの日次の終値の時系列データ
- 期間
  - 1997年4月1日～2003年8月31日
- 学習用データ
  - 2つのデータとも4月1日から1001営業日分のデータをモデルの学習用データとする．

ヒストリカルボラティリティ

- 1年250営業日として，年率換算したヒストリカルボラティリティの算出方法は，次の(6)式である．

$$\left(\log\left(\frac{r_t}{r_{t-1}}\right)\right)^2 \times (250)^{\frac{1}{2}} \times 100 \quad (\%) \qquad (6)$$

- S-PLUSを用いてヒストリカルボラティリティを出力したものを図2に示す（scriptは付録B参照）．
- 図2は，横軸が営業日，縦軸がボラティリティを表しており，山が大きいほど，その時点で株価の収益率に大きな変化があったといえる．

ヒストリカルボラティリティ出力

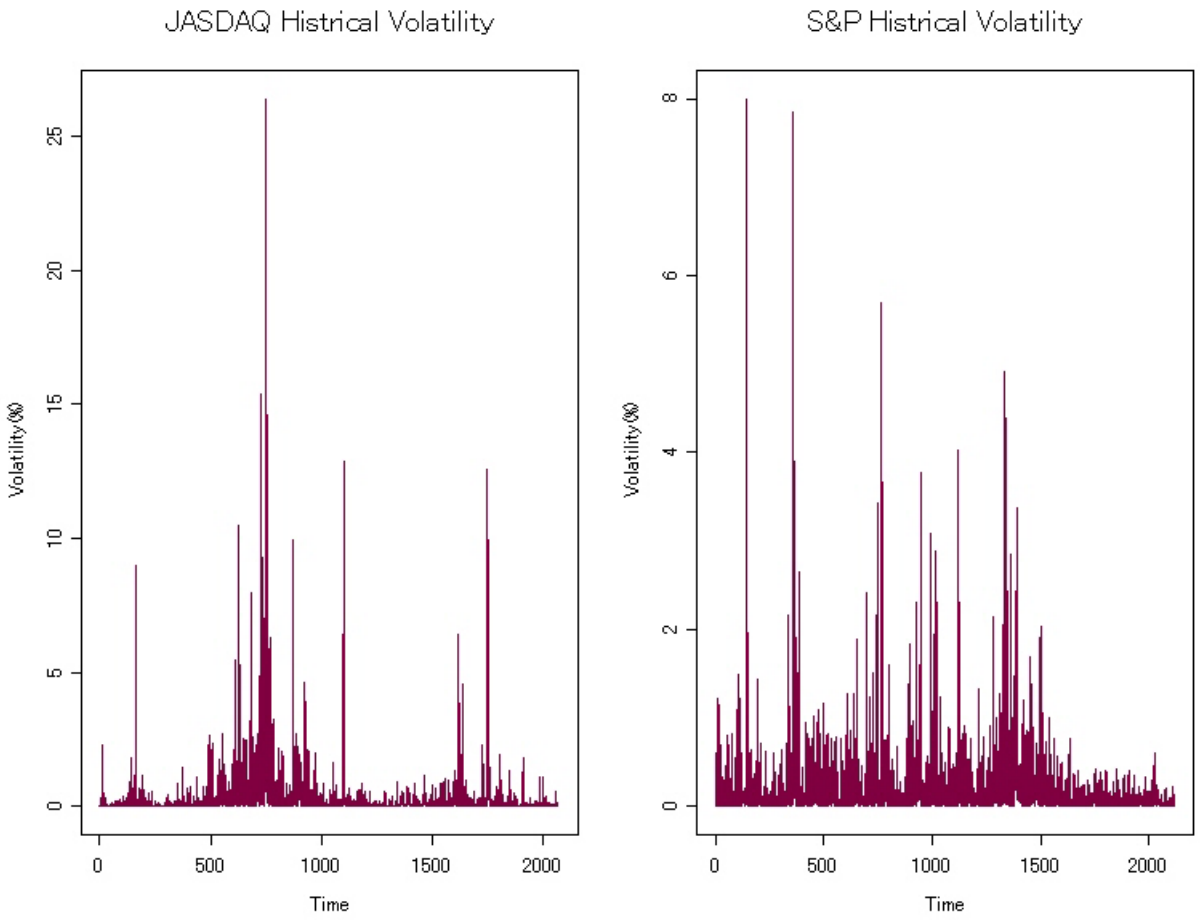

図2：ヒストリカルボラティリティ

GARCH型モデルを用いたボラティリティの予測

- 主に使用した，S-PLUSのfunction
  - GARCHモデルのパラメータの推定

    garch　：　GARCHモデル

    tgarch　：　GJRGARCHモデル
  - GARCHモデルを用いたシミュレーション

    simulate
- scriptは付録C参照

GARCH型による予測の出力

- GARCHモデル，GJRGARCHモデルを用いてボラティリティを予測した結果を図3に示す．
  - 縦軸第1軸はボラティリティの大きさを示す
    - 緑線：ボラティリティの予測値
    - 赤線：ヒストリカルボラティリティ
  - 縦軸第2軸は株価の収益率の実測値を示す
    - 黒線：t 期の収益率から収益率の平均を引いたもの
- ボラティリティの予測値が，前半部分で単調減少しているように見えるが，詳細を観察すると，細かい上下を繰り返している．

GARCHモデルの出力

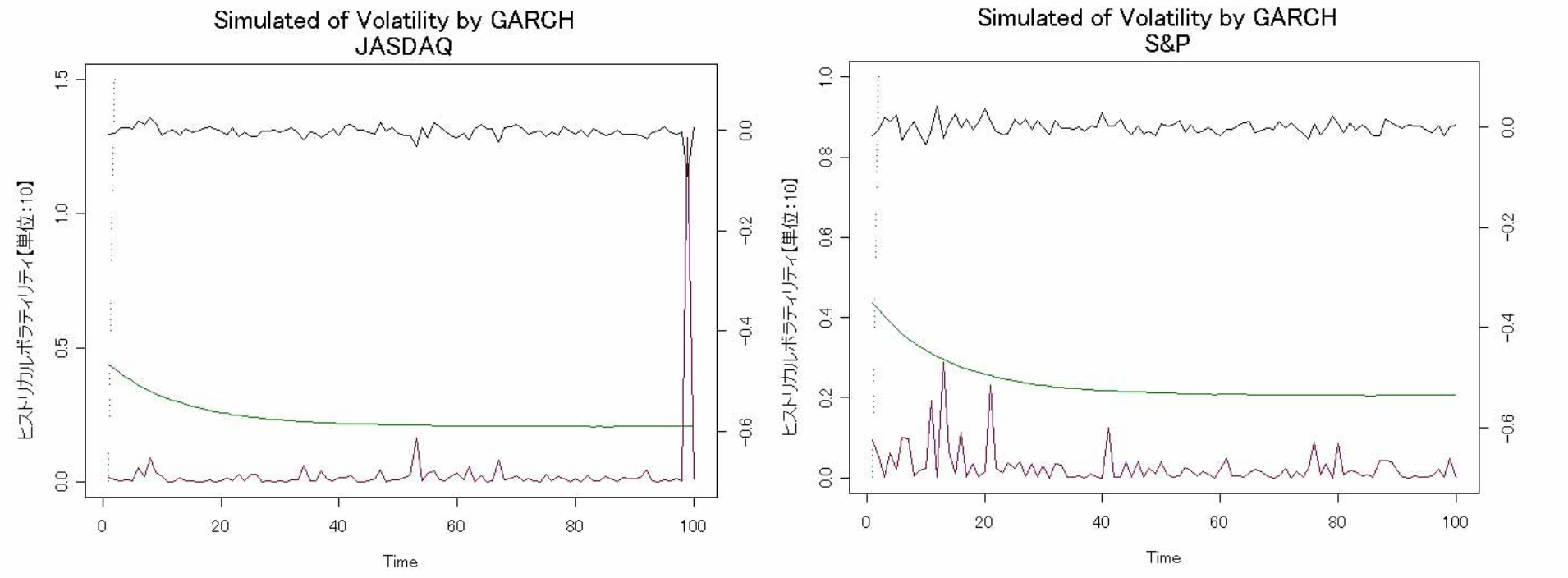

図3.1：GARCHモデルを用いたボラティリティの予測

GJRGARCHモデルの出力

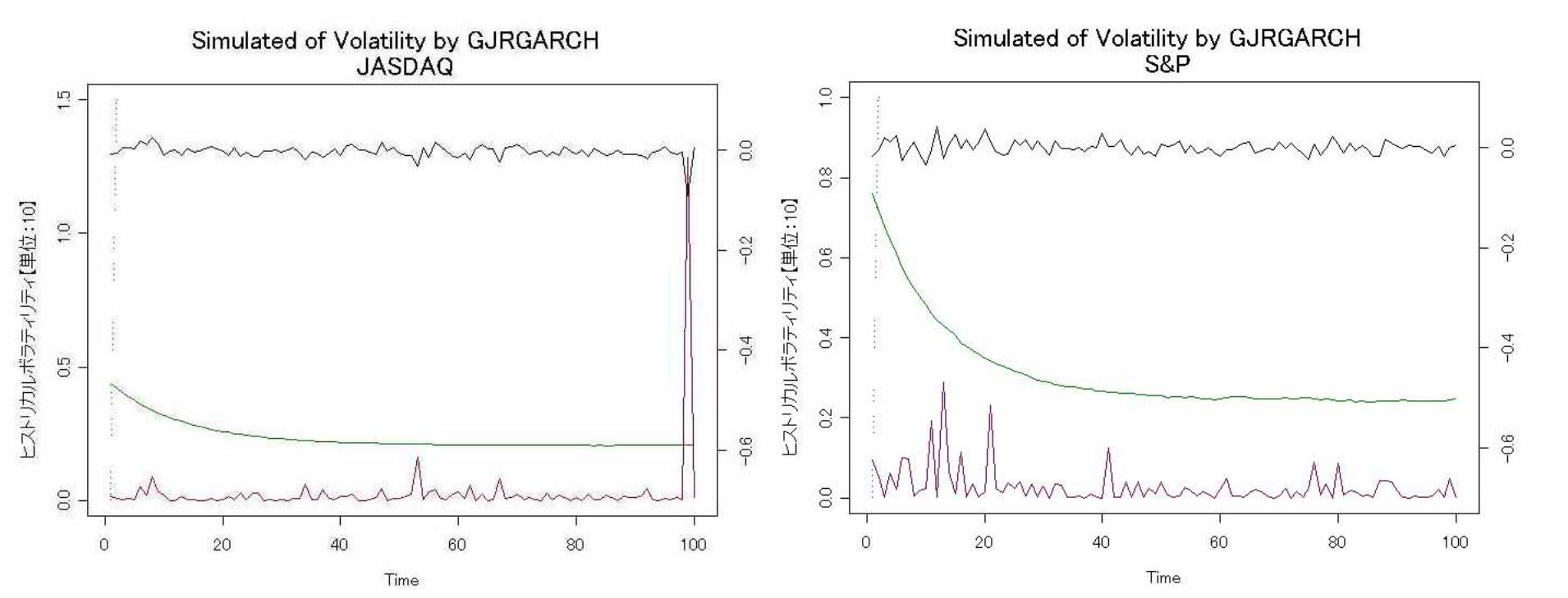

図3.2：GJRGARCHモデルを用いたボラティリティの予測

GARCH型による予測の検証

- 予測精度のよさを寄与率を使って表現する.
- 寄与率は, モデルによって予測されたボラティリティとヒストリカルボラティリティの値の相関係数を計算し, 相関係数の2乗とした.
  - 本研究では寄与率は, ヒストリカルボラティリティを株価の収益率のボラティリティの実測値と定義をした.
- scriptは付録C参照
- モデルの寄与率を計算したところ, 表1のような結果を得た.

GARCH型モデルの予測の寄与率

表1.1：GARCHモデルを用いた予測の寄与率

	10日先まで予測	20日先まで予測	100日先まで予測
JASDAQ	27.4%	3.9%	0.3%
S&P	8.7%	0.0%	12.1%

表1.2：GJRGARCHモデルを用いた予測の寄与率

	10日先まで予測	20日先まで予測	100日先まで予測
JASDAQ	27.5%	3.5%	0.3%
S&P	8.2%	0.0%	11.6%

寄与率に関する考察

- 寄与率の値を見てみると, GARCHモデルとGJRGARCHモデルの間に, 大きな相違は見つからなかった.
- 今回用いた2つのデータでは2つのモデルの精度に大きな違いはないといえる.
- GARCH型モデルは短期間の予測用である[1].
- JASDAQのデータでは, 短期間の予測の寄与率の方が長期間の予測の寄与率よりも高いことがいえる. よって, 上の事実を証明している.
- S&Pにおいて, 100日先までの予測の寄与率が高くなってしまった理由は分からなかったので, 今後検討する必要がある.

AS CAViaRモデルのパラメータの推定

- 主に使用したS-PLUSのfunction
  - AS CAViaRモデルのパラメータ推定

    caviar : 付録D参照
    - caviarにデータセット入れて実行するとリスト形式の出力を得る.
    - リストには次が含まれる.

      $percentile ： パラメータによって計算された分位点

      $beta ： モデルの各パラメータ

      $theta ： 設定した確率

      $model ： 設定したCAViaRの種類

モデルによって計算された分位点

- 実際の株価とfunctionであるcaviarを用いて得られた95%分位点と5%分位点をあわせて出力したものを図4に示す.
  - 青線:実際の株価の収益率
  - 緑線:95%分位点
  - 赤線:5%分位点
- 95%分位点は実際の株価の収益率の上側に沿って,5%分位点は実際の株価の収益率の下側に沿って推移している様子が分かる.
- scriptは付録E参照.

モデルによって計算された分位点の出力

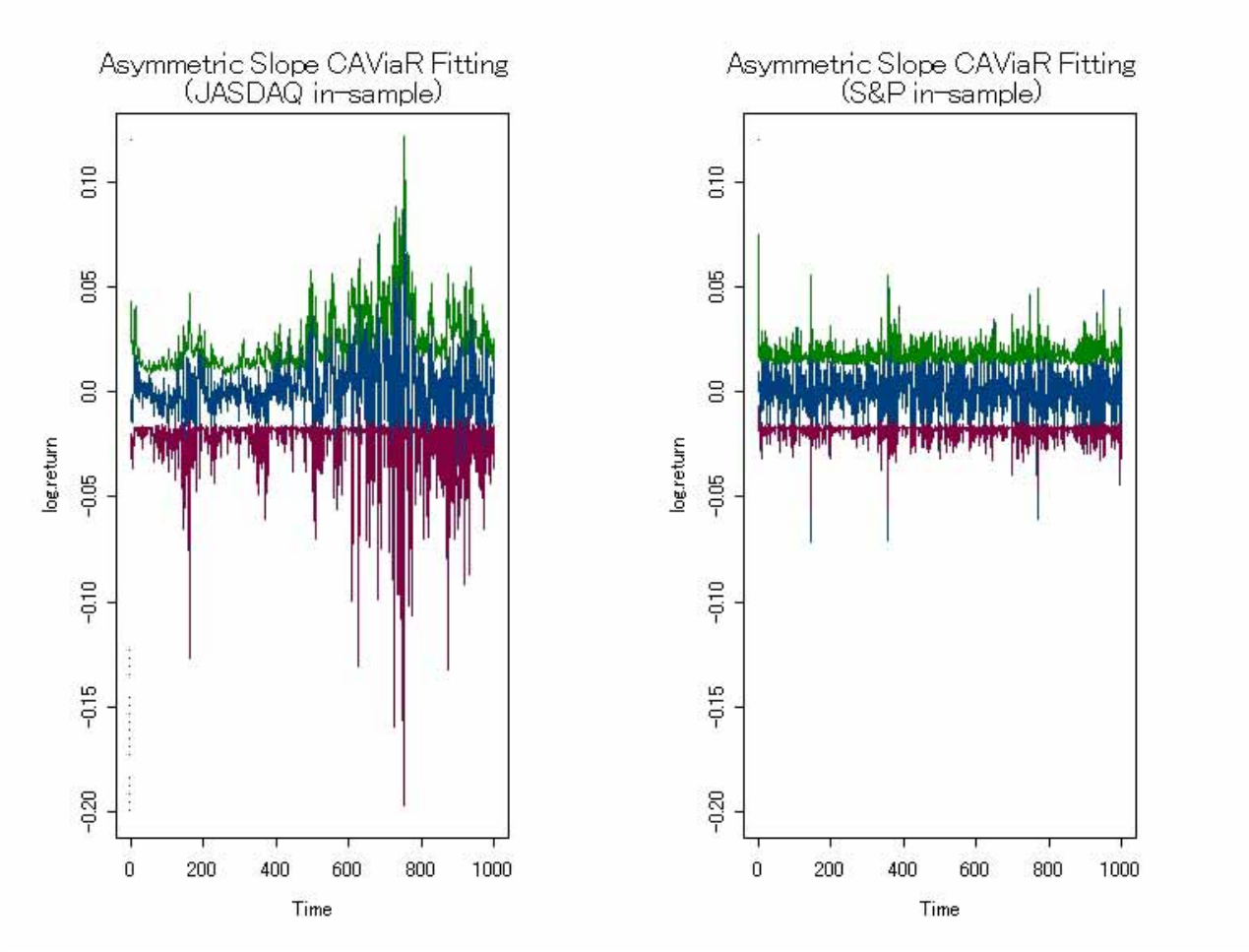

図4：AS CAViaRモデルによって得られた学習データの分位点

AS CAViaRを用いたボラティリティの予測

- functionであるcaviarによって出力された$betaを用いて95%, 5%の分位点を次々と予測する.
- ボラティリティは分位点を用いて次の(7)式で近似できる.

$$\sigma \approx \frac{Q(0.95) - Q(0.05)}{3.25} \qquad (7)$$

- 予測結果を図5に示す.
  - 各軸および線の定義は図3と同じ.
- scriptは付録F参照.

AS CAViaRモデルの予測の出力

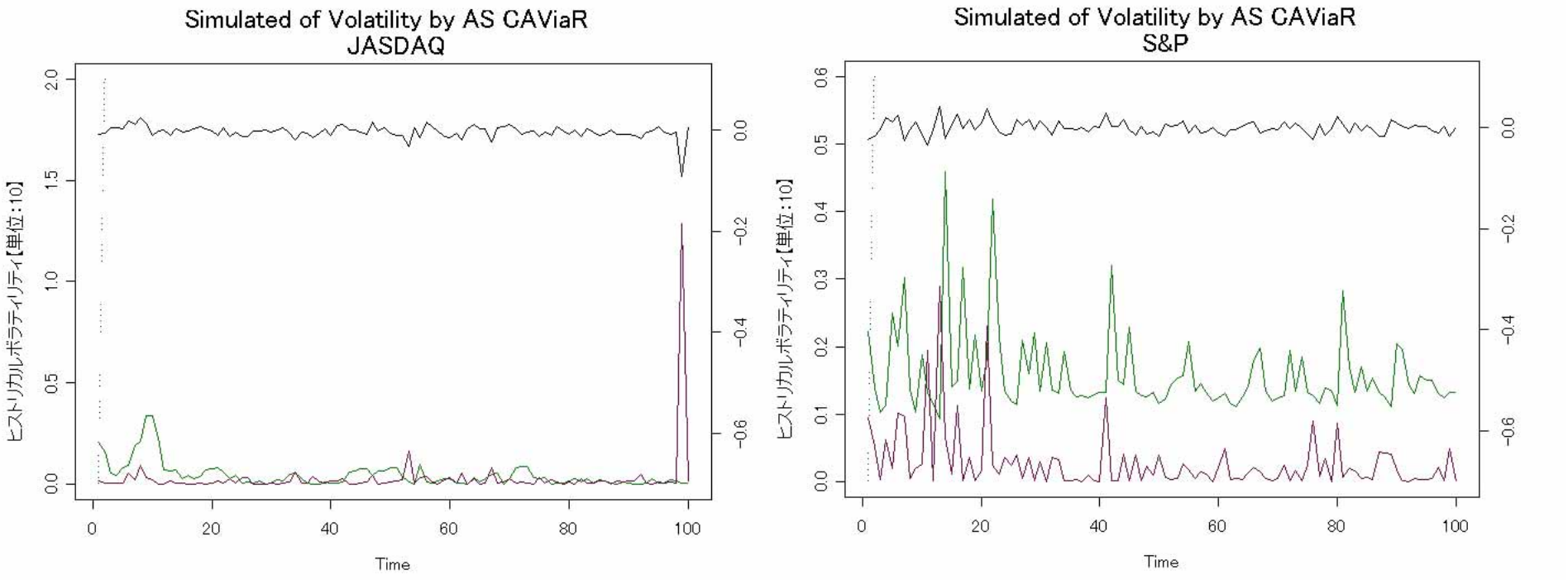

図5：AS CAViaRモデルを用いたボラティリティの予測

AS CAViaRモデルによる予測の検証

- 寄与率を計算した結果を表2に示す.

表2：CAViaRモデルを用いた予測の寄与率

	10日先まで予測	20日先まで予測	100日先まで予測
JASDAQ	8.5%	19.1%	0.2%
S&P	31.8%	1.2%	0.0%

- 2つのデータともに短期間の予測の寄与率は高いが長期間の予測の寄与率は0に近い.
- AS CAViaRモデルを用いた予測はGARCH型モデルによる予測と同様に短期期間用である.
- 黄色の部分は, GARCH型の予測よりも寄与率が高かった部分である

考察

- AS CAViaRモデルもGARCH型モデルも短期間予測用のモデルである.
  - 実務においてボラティリティの予測を行う際は，頻繁に新しいデータを取り込み，予測を更新していく
  - よって短期間の予測ができれば問題ない.
- AS CAViaRモデルの方がGARCH型モデルよりも，高い精度でボラティリティの予測が行える.
  - 用いたデータは少ないが，JASDAQがボラティリティの高い市場を，S&Pがボラティリティの低い市場を代表されるといえる

まとめ

- 本研究では，S-PLUSを用いてボラティリティの予測を行った．S-PLUSを用いたことによる利点を挙げる．
  - データセットさえ作っておけば，scriptを少し書き換えることで予測を実行することができる．
  - 金融関係のfunctionが充実しており，収益率の計算からGARCHモデルのあてはめまで容易に行うことができる．
  - 計算した結果をきれいに出力できる．
- S-PLUSにおける今後の課題
  - for文を用いてプログラムを組んでしまったので，データが大きくなったときにかなりの時間を要することが予想される．
  - S-PLUSはベクトル演算が早いのが特徴なので，ベクトルを中心に記述されたプログラムを組むのが課題．

参考文献

[1] James W. Taylor (2005), Generating Volatility Forecasts from Value at Risk Estimates, MANAGEMENT SCIENCE, 51: 712-725

[2] Engle, R. F. and S. Manganelli (2004), CAViaR: Conditional autoregressive value at risk by regression quantiles, J. Bus. Econm. Statist, 22: 367-381

[3] Roger Koenker and Gilbert Bassett, Jr (1978), Regression Quantiles, Econometrica, Vol 46, No.1: 33-50

[4] **刈屋武明, 矢島美寛, 田中勝人, 竹内啓**(2003)**, 金融時系列の統計. 岩波書店**

[5] **廣松毅, 浪花貞夫**(1990)**, 経済時系列分析. 朝倉書店**

[6] R.A.**ベッカー**・J.M.**チェンバース**・A.R.**ウィルクス**(1991)**,** S**言語 II . 共立出版株式会社**

[7] Eric Zivot and Jiahui Wang(2001), Modeling Finacial Time Series with S-PLUS. Insightful

[8] Yahoo!**ファイナンス, 株価・投信時系列データ** http://table.yahoo.co.jp/t **, 最終閲覧日**9**月**30**日**

[9] JASDAQ, JASDAQ INDEX **算出要領** http://www.jasdaq.co.jp/data/JASDAQ_INDEX_Summary170204.pdf **, 最終閲覧日**10**月**1**日**

A．データセット

- 今回用いた，データセットであるJASDAQおよびS．Pは，日付を示すposition，始値を示すstart，高値を示すhigh，安値を示すlow，終値を示すendで構成されたデータフレームである．

B. ヒストリカルボラティリティの算出及び図の出力

```
## JASDAQ
tmp.J = ts(JASDAQ[,5])                          # データをセット
HV.J = vector(mode = "double",length = length(tmp.J) - 1) # 代入用のベクトルを準備
for(i in 1:length(tmp.J) - 1){
  a = tmp.J[i+1]
  b = tmp.J[i]
  c = (log(a/b))^2
  HV.J[i] = c * sqrt(250) *100
}                                  # ヒストリカルボラティリティを計算し、順次代入
HV.J = ts(HV.J)                           # 値を時系列化
## S&P
tmp.S = ts(S.P[,5])
HV.S = vector(mode = "double",length = length(tmp.J) - 1)
for(i in 1:length(tmp.S) - 1){
  a = tmp.S[i+1]
  b = tmp.S[i]
  c = (log(a/b))^2
  HV.S[i] = c * sqrt(250) *100
}
HV.S = ts(HV.S)
## plot
par(mfrow=c(1,2))
tsplot(HV.J, col=3)
title(main="JASDAQ Histrical Volatility", xlab="Time", ylab="Volatility(%)")
tsplot(HV.S, col=3)
title(main="S&P Histrical Volatility", xlab="Time", ylab="Volatility(%)")
```

C. GARCHモデル型によるボラティリティの予測

```
#なお，このscriptはデータセットを変えるだけで様々なデータに適用可能.
#TGARCHモデルを適用するには、garchをtgarchに変更することで実現できる.

## データを用意
tmp = JASDAQ[,5]
data1.J = ts(tmp[1:1001])   # 学習用データ
data2.J = ts(tmp)       # 全データ
data3.J = ts(getReturns(data1.J))    # 学習用データの収益率（標本数：1000）
data4.J = ts(getReturns(data2.J))    # 全データ収益率
eps.J = ts(getReturns(data2.J) - mean(getReturns(data2.J)[1:1000]))
             # 評価用データの偏差
# 学習用データでGARCHモデルの当てはめ
# ~garch を ~tgarch に変更することによってGJRGARCHに変更可能
data.g.J = garch(data3.J~1,~garch(1,1), trace=F, leverage=T)
## ヒストリカルボラティリティの計算
data.s.J = ts(JASDAQ[,5])
HV.1.J = 1:length(data.s.J - 1)
for(i in 1:length(data.s.J - 1)){
  a = data.s.J[i+1]
  b = data.s.J[i]
  c = (log(a/b))^2
  HV.1.J[i] = c * sqrt(250) *100
}
```

```
## 以下、ボラティリティの推定　（参照 -> [5] P255）
sigma.start.J = as.numeric(data.g.J$sigma.t[1000])                    # 学習用データの最後の標準偏差
eps.start.J = as.numeric(data.g.J$residuals[1000])                    # 学習用データの最後の残差
eps.start.J = matrix(eps.start.J, 1, 1000)                                    # 以下シミュレーションのパスを作
成
error.J = rbind(eps.start.J, matrix(rnorm(100*1000), 100))
set.seed(4989)
data.t.pred.J = simulate(data.g.J, n=100, n.rep=1000, sigma.start=sigma.start.J, etat=error.J)$sigma.t
            # シミュレーションをする
vol.mean.J = rowMeans(data.t.pred.J)                          # 作成されたパスの平均を取ってボラティリティの平方根とする
vol.stdev.J = rowStdevs(data.t.pred.J)
sigma.2.J = (vol.mean)^2 * sqrt(250) * 100                    # 1年250営業日として，ボラティリティを年率で表現する
## 作図
HV.data.J = ts(HV.1.J[1001:1100])                                                     # 評価期間のヒスト
リカルボラティリティを抽出
par(mar=rep(6,4))
                        # 余白を設定する
tsplot(sigma.2.J, col=4, range(0,1.5))
tslines(HV.data.J/10, col=3,axes=F)
par(new=TRUE)
                                  # 新しい高水準であると宣言
tsplot(eps.J[1001:1100],axes=F,range(-0.7,0.1))                       # 別の縦軸でplot
axis(side=4)
title(main="Simulated of Volatility by GARCH ¥n JASDAQ", xlab="Time", ylab="ヒストリカルボラティリティ【単位：10】")
            # 緑線が予測値，赤線が実測値
## 寄与率
corr.100.J = cor(HV.data.J,sigma.2.J)                                                 # 100日先までの予
測の寄与率
r.2.100.J = (corr.100.J)^2
r.2.100.J
corr.20.J = cor(HV.data.J[1:20],sigma.2.J[1:20])                      # 20日先までの予測の寄与率
r.2.20.J = (corr.20.J)^2
r.2.20.J
corr.10.J = cor(HV.data.J[1:10],sigma.2.J[1:10])                      # 10日先までの予測の寄与率
r.2.10.J = (corr.10.J)^2
r.2.10.J
```

D. function : caviar

```
###QRSum 最小化モデル
##Asymmetric Slope
QRSum.ASCAViaR = function(y,eps.p,eps.m,theta){

  Time <- Set()
  Slope <- Set(1:4)

  length.time <- length(y)

  i <- Element(set = Slope)
  t <- Element(set = Time)

  y <- Parameter(list(1:length.time,y), index = t)
  eps.p <- Parameter(list(1:length.time,eps.p),index = t)
  eps.m <- Parameter(list(1:length.time,eps.m),index = t)

  beta <- Variable(index = i)
  Q <- Variable(index = t)

  obj = Objective(type="minimize")

  Q[t+1,t<length.time] == beta[1] + beta[2] * Q[t] + beta[3] * eps.p[t] + beta[4] * eps.m[t]

  obj ~ Sum(ife(y[t] >= Q[t],theta * (y[t] - Q[t]), (theta - 1) * (y[t] - Q[t])),t)
}
```

```
##Symmetric Absolute Slope
QRSum.SASCAViaR = function(y,eps,theta){

          Time <- Set()
          Slope <- Set(1:3)

          length.time <- length(y)
          eps <- abs(eps)

          i <- Element(set = Slope)
          t <- Element(set = Time)

          y <- Parameter(list(1:length.time,y), index = t)
          abseps <- Parameter(list(1:length.time,eps),index = t)

          beta <- Variable(index = i)
          Q <- Variable(index = t)

          obj = Objective(type="minimize")

          Q[t+1,t<length.time] == beta[1] + beta[2] * Q[t] + beta[3] * abseps[t]

          obj ~ Sum(ife(y[t] >= Q[t],theta * (y[t] - Q[t]), (theta - 1) * (y[t] -
Q[t])),t)
}
```

```
##Adaptive CAViaR
QRSum.AdptCAViaR = function(y,eps,theta){

          Time <- Set()
          Slope <- Set(1:1)

          length.time <- length(y)

          i <- Element(set = Slope)
          t <- Element(set = Time)

          y <- Parameter(list(1:length.time,y), index = t)
          eps <- Parameter(list(1:length.time,eps),index = t)

          beta <- Variable(index = i)
          Q <- Variable(index = t)

          obj = Objective(type="minimize")

          Q[t+1,t<length.time] == Q[t] + beta[1] * (theta - ife(eps[t] <= Q[t],1,0))

          obj ~ Sum(ife(y[t] >= Q[t],theta * (y[t] - Q[t]), (theta - 1) * (y[t] -
Q[t])),t)
}
```

付録

```
###CAViaRモデル
caviar = function(x,model="Asymmetric",prob=0.05,graph=T,scaling="on",ep=1e-3){
          #モジュール呼び出し
          module(nuopt)

          #元データ編集
          eps = diff(log(x))

          #パラメータ推定開始

          if(model == "Asymmetric"){
                    nuopt.options(maxitn = 200,method="auto",scaling=scaling,eps=ep)

                    #epsilon^+とepsilon^-を計算
                    eps.p = ifelse(eps>0,eps,0)
                    eps.m = ifelse(eps<0,abs(eps),0)
                    #最適化
                    result.problem = System(model=QRSum.ASCAViaR,eps,eps.p,eps.m,prob)
                    result.solution = solve(result.problem)

          }else if(model == "Symmetric"){
                    nuopt.options(maxitn = 200,method="auto",scaling=scaling,eps=ep)
```

```
                    #最適化
                    result.problem =
System(model=QRSum.SASCAViaR,eps,eps,prob)
                    result.solution = solve(result.problem)

          }else if(model == "Adaptive"){
                    nuopt.options(maxitn =
150,method="auto",scaling=scaling,eps=ep)
                    #最適化
                    result.problem =
System(model=QRSum.AdptCAViaR,eps,eps,prob)
                    result.solution = solve(result.problem)

          }else{
                    print("モデル名、違うから。。。")
          }

          Q = result.solution$variable$Q$current
          beta = result.solution$variable$beta$current
          rslt = list(percentile = Q, beta = beta, theta =
prob,model=model)

          if(graph){
                    tsplot(eps)
                    lines(Q,col=6)
          }

          rslt
}
```

E. CAViaR

```
##JASDAQ当てはめ
data.J = JASDAQ[,5]    #データの準備
asyca.05.J = caviar(data.J[1:1001], prob=0.05) #確率5%のAS CAViaRを学習用データによって推定
asyca.95.J = caviar(data.J[1:1001], prob=0.95) #確率95%のAS CAViaRを学習用データによって推定
quant.asyca.05.J = ts(asyca.05.J$percentile[2:1000])   #確率5%の分位点を時系列化
quant.asyca.95.J = ts(asyca.95.J$percentile[2:1000])#確率95%の分位点を時系列化
eps.J = ts(getReturns(data.J) - mean(getReturns(data.J)[1:1000]))  #実測値の偏差
##S&P当てはめ
data.S = S.P[,5]
asyca.05.S = caviar(data.S[1:1001], prob=0.05)
asyca.95.S = caviar(data.S[1:1001], prob=0.95)
quant.asyca.05.S = ts(asyca.05.S$percentile[2:1000])
quant.asyca.95.S = ts(asyca.95.S$percentile[2:1000])
eps.S = ts(getReturns(data.S) - mean(getReturns(data.S)[1:1000]))
##plot
par(mfrow=c(1,2))#2つのグラフを一度に表示
tsplot(eps.J[2:1000], col=2, range(-0.2,0.12))
tslines(quant.asyca.95.J, col=4)
tslines(quant.asyca.05.J, col=3)
title(main="Asymmetric Slope CAViaR Fitting ¥n (JASDAQ in-sample)",xlab="Time",ylab="log.return")
tsplot(eps.S[2:1000], col=2, range(-0.2,0.12))
tslines(quant.asyca.95.S, col=4)
tslines(quant.asyca.05.S, col=3)
title(main="Asymmetric Slope CAViaR Fitting ¥n (S&P in-sample)", xlab="Time", ylab="log.return")
```

F. CAViaRモデルによるボラティリティの予測

```
##predict quantile
epsp.J = ifelse(eps.J > 0, eps.J, 0)      #(3)式右辺第3項のepsilon-plusを計算する
epsm.J = ifelse(eps.J < 0, abs(eps.J), 0) #(3)式右辺第3項のepsilon-minusを計算する
epsp.S = ifelse(eps.S > 0, eps.S, 0)
epsm.S = ifelse(eps.S < 0, abs(eps.S), 0)
pred.quant.05.J = vector(mode = "double",length=1100) #ベクトルを用意する
pred.quant.95.J = vector(mode = "double",length=1100)
pred.quant.05.S = vector(mode = "double",length=1100)
pred.quant.95.S = vector(mode = "double",length=1100)
#分位点を先に予測した係数によって計算する
for(i in 1:100){
   if(i == 1){
      pred.quant.05.J[i+1000] = asyca.05.J$beta[1] + asyca.05.J$beta[2] * asyca.05.J$percentile[1000]
+ asyca.05.J$beta[3] * epsp.J[i+999] + asyca.05.J$beta[4] * epsm.J[i+999]
      pred.quant.95.J[i+1000] = asyca.95.J$beta[1] + asyca.95.J$beta[2] * asyca.95.J$percentile[1000]
+ asyca.95.J$beta[3] * epsp.J[i+999] + asyca.05.J$beta[4] * epsm.J[i+999]
      pred.quant.05.S[i+1000] = asyca.05.S$beta[1] + asyca.05.S$beta[2] * asyca.05.S$percentile[1000]
+ asyca.05.S$beta[3] * epsp.S[i+999] + asyca.05.S$beta[4] * epsm.S[i+999]
      pred.quant.95.S[i+1000] = asyca.95.S$beta[1] + asyca.95.S$beta[2] * asyca.95.S$percentile[1000]
+ asyca.95.S$beta[3] * epsp.S[i+999] + asyca.05.S$beta[4] * epsm.S[i+999]
   }
   else{
      pred.quant.05.J[i+1000] = asyca.05.J$beta[1] + asyca.05.J$beta[2] * pred.quant.05.J[i+999]
+ asyca.05.J$beta[3] * epsp.J[i+999] + asyca.05.J$beta[4] * epsm.J[i+999]
      pred.quant.95.J[i+1000] = asyca.95.J$beta[1] + asyca.95.J$beta[2] * pred.quant.95.J[i+999]
+ asyca.95.J$beta[3] * epsp.J[i+999] + asyca.05.J$beta[4] * epsm.J[i+999]
      pred.quant.05.S[i+1000] = asyca.05.S$beta[1] + asyca.05.S$beta[2] * pred.quant.05.S[i+999]
+ asyca.05.S$beta[3] * epsp.S[i+999] + asyca.05.S$beta[4] * epsm.S[i+999]
      pred.quant.95.S[i+1000] = asyca.95.S$beta[1] + asyca.95.S$beta[2] * pred.quant.95.S[i+999]
+ asyca.95.S$beta[3] * epsp.S[i+999] + asyca.05.S$beta[4] * epsm.S[i+999]
   }
}
```

```
pred.quant.05.J
pred.quant.05.J = pred.quant.05.J[1001:1100]      #5%の分位点の予測値
pred.quant.95.J = pred.quant.95.J[1001:1100]      #95%の分位点の予測値
pred.quant.05.S = pred.quant.05.S[1001:1100]
pred.quant.95.S = pred.quant.95.S[1001:1100]
##predict volatility
pred.vol.J = ts ((pred.quant.95.J - pred.quant.05.J)/3.25)^2 * sqrt(250) *100
          #(7)式を用いてボラティリティを予測する
pred.vol.S = ts ((pred.quant.95.S - pred.quant.05.S)/3.25)^2 * sqrt(250) *100
##predict volatility plot
par(mfrow=c(1,1))
par(mar=rep(6,4))
tsplot(pred.vol.J, range(0,2),col=4)
tslines(HV.J[1001:1100]/10,col=3)
par(new=TRUE)
tsplot(eps.J[1001:1100],axes=F,range(-0.7,0.1))
axis(side=4)
title(main="Simulated of Volatility by AS CAViaR ¥n JASDAQ", xlab="Time",
ylab="ヒストリカルボラティリティ【単位：10】")
par(mar=rep(6,4))
tsplot(pred.vol.S, range(0,0.6),col=4)
tslines(HV.S[1001:1100]/10,col=3)
par(new=TRUE)
tsplot(eps.S[1001:1100],axes=F,range(-0.7,0.1))
axis(side=4)
title(main="Simulated of Volatility by AS CAViaR ¥n S&P", xlab="Time",
ylab="ヒストリカルボラティリティ【単位：10】")
```

```
##寄与率
pred.corr.J.100 = cor(HV.J[1001:1100],pred.vol.J[1:100])    # 100日先までの予測の寄与
率
pred.r.2.J.100 = (pred.corr.J.100)^2
pred.r.2.J.100
pred.corr.J.20 = cor(HV.J[1001:1020],pred.vol.J[1:20])      # 20日先までの予測の寄与率
pred.r.2.J.20 = (pred.corr.J.20)^2
pred.r.2.J.20
pred.corr.J.10 = cor(HV.J[1001:1010],pred.vol.J[1:10])      # 10日先までの予測の寄与率
pred.r.2.J.10 = (pred.corr.J.10)^2
pred.r.2.J.10
pred.corr.S.100 = cor(HV.S[1001:1100],pred.vol.S[1:100])    # 100日先までの予測の寄与
率
pred.r.2.S.100 = (pred.corr.S.100)^2
pred.r.2.S.100
pred.corr.S.20 = cor(HV.S[1001:1020],pred.vol.S[1:20])      # 20日先までの予測の寄与率
pred.r.2.S.20 = (pred.corr.S.20)^2
pred.r.2.S.20
pred.corr.S.10 = cor(HV.S[1001:1010],pred.vol.S[1:10])      # 10日先までの予測の寄与率
pred.r.2.S.10 = (pred.corr.S.10)^2
pred.r.2.S.10
```