Análise de Volatilidade Estocástica e Retornos Intradiários — PETR4 2021

Guia Educativo do Código R

Gustavo Vitor da Silva

Dados: MetaTrader 5 | Frequência: 1 minuto | Período: Jan–Jun 2021

49.611 observações	**1 minuto** frequência	**3 modelos** stochvol	**6 meses** jan–jun 2021

■ Índice do Documento

1 Introdução e Contexto

Este documento é um guia educativo que explica, linha a linha, cada bloco de código R contido no arquivo **analise_volatilidade_petr4_Q1.qmd**. O objetivo é tornar acessível a metodologia utilizada para analisar a volatilidade estocástica e os retornos intradiários da ação **PETR4 (Petrobrás)** ao longo do primeiro semestre de 2021.

O que é PETR4?

PETR4 é o ticker (código de negociação) das ações preferenciais da Petrobrás na B3 (Bolsa brasileira). É um dos ativos mais líquidos do mercado brasileiro e tem grande peso no índice Ibovespa. Em fevereiro de 2021, a ação sofreu uma queda abrupta motivada pelo anúncio da troca do presidente da empresa pelo governo federal — evento que serve como laboratório natural para estudar regimes de volatilidade.

O que são dados de 1 minuto?

Os dados utilizados são de **frequência intradiária de 1 minuto**, ou seja, para cada minuto do pregão existe um registro com: timestamp (data e hora), preço de fechamento do minuto (Close.1min) e retorno percentual daquele minuto (Ret.1min). Isso resulta em aproximadamente 390 observações por dia (das 10h às 16h55, após os filtros aplicados) e cerca de 49.611 observações no período total analisado.

Estrutura do Estudo

A análise é dividida em três **regimes temporais** que permitem comparar o comportamento estatístico antes, durante e após o choque de mercado:

- **Segmento A — Pré-queda:** observações 1 a 10.000 (mercado estável)
- **Segmento B — Pós-queda:** observações 11.600 a 22.600 (inclui o choque)
- **Segmento C — Recuperação:** observações 33.600 a 43.600 (semanas após)

O estudo utiliza métodos tanto **clássicos** (volatilidade realizada, ACF, ADF) quanto **bayesianos** (modelo de volatilidade estocástica via MCMC) para obter uma visão completa do comportamento da série.

2 Bloco 1 — Carregamento de Pacotes (library)

O primeiro bloco carrega todos os pacotes necessários para a análise. Em R, um pacote é uma coleção de funções prontas que estendem as capacidades básicas da linguagem. Sem carregar os pacotes, as funções específicas não estariam disponíveis.

```
library(ggplot2) # Gráficos elegantes
library(zoo) # Séries temporais irregulares
library(xts) # Extensible Time Series
library(forecast) # Modelos de previsão (ARIMA, ETS)
library(lubridate) # Manipulação de datas e horas
library(PerformanceAnalytics) # Métricas financeiras
library(quantmod) # Download e análise de dados financeiros
library(tseries) # Testes de séries temporais (ADF, KPSS)
library(FinTS) # Teste ARCH
library(stochvol) # Modelos de volatilidade estocástica (MCMC)
library(rugarch) # Família GARCH
library(moments) # Assimetria e curtose
library(coda) # Diagnóstico de cadeias MCMC
library(car) # Teste de Levene (homogeneidade de variância)
library(boot) # Bootstrap
library(wavelets) # Análise de ondaletas (wavelets)
```

Pacote	Finalidade Principal	Funções Usadas
ggplot2	Visualizações gráficas declarativas	ggplot(), geom_*(), theme_minimal()
xts / zoo	Séries temporais indexadas por data/hora	as.xts(), rollapply()
lubridate	Datas: parse, extração (hour, minute, date)	ymd_hms(), hour(), minute(), date()
stochvol	Volatilidade estocástica via MCMC Bayesiano	svsample(), para(), latent()
moments	Estatísticas de forma da distribuição	skewness(), kurtosis()
boot	Reamostragem bootstrap para IC de métricas	boot(), boot.ci()
car	Teste de Levene para variância	leveneTest()
coda	Diagnóstico de convergência MCMC	gelman.diag(), traceplot()

rugarch	Modelos GARCH (alternativa ao stochvol)	ugarchspec(), ugarchfit()

3 Bloco 2 — Leitura e Pré-processamento dos Dados

Este bloco realiza a **ingestão dos dados brutos** e aplica uma série de filtros para garantir que a série temporal seja limpa, representativa e adequada para análise estatística.

3.1 Detecção do Sistema Operacional

```
os <- Sys.info()["sysname"]

if(os == "Windows") {

dt.intra <- read.csv("D:/Code/R_studio/Petr4_ana/...", sep=";", dec=",")

} else if(os == "Linux") {

dt.intra <- read.csv("~/Documents/Coding/Statistics_in_R/...", sep=";", dec=",")

}
```

Como funciona: `Sys.info()["sysname"]` retorna o nome do sistema operacional ("Windows", "Linux" ou "Darwin" para macOS). Isso permite que o mesmo código funcione em máquinas diferentes, apontando para o arquivo CSV correto conforme o caminho do sistema. O CSV usa ponto-e-vírgula como separador e vírgula como decimal (padrão brasileiro), daí os argumentos `sep=";"` e `dec=","`.

3.2 Pipeline de Limpeza com dplyr

```
dt1 <- as_tibble(dt.intra) %>%

mutate(Period = ymd_hms(X)) %>% # converte string p/ datetime

select(-X) %>% # remove coluna original

filter(!(hour(Period)==10 & minute(Period)<20)) %>% # remove abertura

filter(hour(Period) < 17) %>% # remove após 17h

filter(!(hour(Period)==16 & minute(Period)>54)) %>% # remove >16:54

filter(!(date(Period)=="2021-02-17")) %>% # remove feriado

arrange(Period) # ordena cronologicamente
```

O operador **%>%** (pipe) do pacote dplyr encadeia operações, passando o resultado de uma função como primeiro argumento da próxima. É equivalente a funções aninhadas, mas muito mais legível.

Filtro	Motivo Estatístico	Observações Removidas
10:00–10:19 (abertura)	Leilão de abertura gera spreads artificialmente grandes, distorcendo retornos	~19 obs/dia
Após 16:55	Leilão de fechamento com liquidez reduzida e preços distorcidos	~5 obs/dia
17/02/2021 (feriado)	Carnaval — pregão fechado, dados ausentes ou incorretos	~390 obs

3.3 Conversão para Série Temporal xts

```
ret.1min <- as.xts(dt1$Ret.1min, order.by = dt1$Period)
```

O objeto **xts** (eXtensible Time Series) armazena os dados junto com seus timestamps, facilitando operações como janelas deslizantes, subsets por data e plots automáticos com eixo temporal correto. A função `as.xts(vetor, order.by=datas)` cria esse objeto especial.

■ **Dica conceitual:** Em finanças de alta frequência, remover os primeiros minutos do pregão é prática padrão. Os preços de abertura refletem acúmulo de ordens noturnas e comportamento do leilão, não a dinâmica contínua de mercado.

4 Bloco 3 — Observação Inicial dos Dados

```
plot(dt1$Close.1min, main="PETR4 Retornos 1min", col="black")

boxplot(as.double(ret.1min), main="Boxplot dos Retornos 1min")
```

Antes de qualquer modelagem, é essencial **visualizar os dados brutos**. Dois gráficos complementares são gerados:

- **Série temporal dos preços:** permite identificar visualmente tendências, quebras estruturais (como a queda de fevereiro) e a presença de clusters de volatilidade.
- **Boxplot dos retornos:** resume a distribuição estatística — mediana, quartis e outliers. Em retornos financeiros, esperam-se outliers extremos (*fat tails*), que no boxplot aparecem como pontos distantes das hastes.

Como funciona o Boxplot: A caixa central vai do 1º quartil (Q1, 25%) ao 3º quartil (Q3, 75%). A linha interna é a mediana (Q2, 50%). As hastes se estendem até Q1–1.5×IQR e Q3+1.5×IQR. Pontos além das hastes são outliers — em retornos de 1 minuto, representam movimentos extremos como o choque de fevereiro de 2021.

5 Bloco 4 — Teste de Estacionariedade (ADF)

```
tseries::adf.test(ret.1min)
```

O que é estacionariedade?

Uma série temporal é **estacionária** quando suas propriedades estatísticas (média, variância, autocovariância) não mudam ao longo do tempo. A maioria dos métodos de análise de séries temporais exige estacionariedade como premissa.

Teste ADF — Augmented Dickey-Fuller

O teste ADF verifica a presença de **raiz unitária** — uma característica de séries não-estacionárias. A hipótese nula (H0) é que a série possui raiz unitária (não é estacionária). Um p-valor pequeno (< 0,05) indica rejeição de H0, confirmando estacionariedade.

Hipótese	Definição	Resultado Esperado
H0 (nula)	Série tem raiz unitária (não-estacionária)	Rejeitada (p < 0.01)
H1 (alternativa)	Série é estacionária	Confirmada
Interpretação	Retornos de 1 min são estacionários	Esperado em finanças

■ **Por que retornos são estacionários mas preços não são?** Preços de ações seguem um processo de passeio aleatório (raiz unitária), enquanto os *retornos* (variação percentual) são tipicamente estacionários — daí a prática padrão de trabalhar com retornos em vez de preços nas análises estatísticas.

6 Bloco 5 — Análise de Tendências Pré e Pós Queda

```
idx_min <- which.min(dt1$Ret.1min) # índice do menor retorno (maior queda)
dados_xts <- xts(dt1[, c("Close.1min", "Ret.1min")], order.by = dt1$Period)
p2 = idx_min + 10000 # janela de 10.000 min após a queda
```

`which.min()` retorna o índice da observação com o menor valor em um vetor. Aqui identifica o momento exato da maior queda dos retornos — o episódio de fevereiro de 2021. Esse índice serve como âncora para definir os segmentos pré e pós-choque.

6.1 Log-Retornos e Remoção de NAs

```
dados_xts$LogRet.1min <- log(1 + dados_xts$Ret.1min)
dados_xts <- na.omit(dados_xts)
```

O **log-retorno** é calculado como ln(1 + r), onde r é o retorno simples. Log-retornos têm duas vantagens: (1) são aditivos ao longo do tempo — a soma de log-retornos diários é igual ao log-retorno do período; (2) são aproximadamente simétricos para retornos pequenos, o que melhora propriedades estatísticas. `na.omit()` remove qualquer linha com valores ausentes.

6.2 Comparação Visual dos Segmentos

```
plot(volhist30min[1:10000], type="l",
main="Volatilidade (Janela 30 min) Pré-queda")
plot(volhist30min[idx_min:p2], type="l",
main="Volatilidade (Janela 30 min) Pós-queda")
```

Dois gráficos de linha mostram a volatilidade histórica nos períodos pré e pós-queda. A comparação visual é o primeiro passo para identificar **mudanças de regime** — transições entre estados distintos de volatilidade que os modelos formais depois quantificam.

7 Bloco 6 — Volatilidade Histórica e ACF/PACF

7.1 Função de Volatilidade Histórica

```
calc_volatilidade_historica <- function(retornos, janela = 21) {
vol_hist <- rollapply(retornos,
width = janela,
FUN = function(x) sd(x, na.rm = TRUE),
by.column = TRUE,
align = "right")
return(vol_hist)
}

dados_xts$Vol_Hist_30min <- calc_volatilidade_historica(
dados_xts$LogRet.1min, janela = 30)

minutos_ano <- 252 * 390 # dias úteis x minutos por pregão
dados_xts$Vol_Anual <- dados_xts$Vol_Hist_30min * sqrt(minutos_ano)
```

Esta função implementa a **volatilidade histórica com janela deslizante**:

- **rollapply():** aplica uma função (`sd`) a cada janela de `width` observações consecutivas, deslizando um passo por vez. Com `align="right"`, o resultado é atribuído ao último ponto da janela.
- **janela = 30:** usa os 30 minutos anteriores para estimar a volatilidade instantânea. Janelas curtas reagem rápido mas são ruidosas; janelas longas são mais suaves.
- **Anualização:** multiplicar por √(252×390) converte a volatilidade por minuto para a escala anual, comparável à volatilidade implícita de opções (expressa em %/ano). 252 é o número convencional de dias úteis e 390 é o número de minutos no pregão.

7.2 ACF e PACF — Estrutura de Autocorrelação

```
par(mfrow = c(1, 2)) # 2 gráficos lado a lado
acf(log_ret_vec[1:10000], main="ACF - Log-Retornos Pré queda", na.action=na.pass)
pacf(log_ret_vec[1:10000], main="PACF - Log-Retornos Pré queda", na.action=na.pass)

acf(log_ret_vec[p1:p2], main="ACF - Log-Retornos Pós queda")
pacf(log_ret_vec[p1:p2], main="PACF - Log-Retornos Pós queda")
```

Função	Definição	Interpretação em Retornos 1min
ACF(k)	Correlação entre r(t) e r(t-k) para todo lag k	Mede dependência linear direta e indireta em cada defasagem
PACF(k)	Correlação entre r(t) e r(t-k) removendo efeitos intermediários	Isola a contribuição única de cada lag, útil para identificar ordem AR(p)

Barras azuis	Bandas de confiança 95% (~±1.96/√n)	Barras fora da banda indicam autocorrelação estatisticamente significativa

■ **O que esperar:** Log-retornos financeiros tipicamente têm pouca autocorrelação linear (mercado eficiente), mas os seus *quadrados* têm alta autocorrelação — evidência de clustering de volatilidade (períodos voláteis tendem a ser seguidos de períodos voláteis).

8 Bloco 7 — Modelo de Volatilidade Estocástica (svsample)

Este é o **núcleo analítico** do estudo. O modelo de Volatilidade Estocástica (SV) trata a volatilidade como uma variável latente (não observável) que evolui ao longo do tempo segundo um processo estocástico próprio.

8.1 Formulação Matemática do Modelo SV

O modelo tem duas equações simultâneas:

- **Equação de observação:** $r(t) = \exp(h(t)/2) \cdot \varepsilon(t)$ onde $\varepsilon(t) \sim N(0,1)$
- **Equação de estado:** $h(t) = \mu + \phi \cdot (h(t-1) - \mu) + \sigma \cdot \eta(t)$ onde $\eta(t) \sim N(0,1)$

Aqui **h(t)** é o log-volatilidade latente (não observado), que segue um processo AR(1) em torno do nível de longo prazo μ. Os três parâmetros estruturais do modelo são:

Parâmetro	Nome	Interpretação	Valor Típico (PETR4)
μ (mu)	Nível médio de log-volatilidade	Log-volatilidade de equilíbrio; valores menos negativos = maior vol média	−13 a −14
ϕ (phi)	Persistência (AR)	Quão rapidamente choques de volatilidade se dissipam. $\phi \to 1$: muito persistente	0.97–0.99
σ (sigma)	Vol da volatilidade	Amplitude das flutuações da própria volatilidade latente	0.11–0.16

8.2 Estimação por MCMC — svsample()

```
sv_fit_raw <- svsample(
ret_vec, # vetor de retornos
draws = 5000, # número de amostras MCMC
burnin = 1000 # descarta as primeiras 1000 (warm-up)
)
```

MCMC (Markov Chain Monte Carlo) é um método computacional para amostrar de distribuições posteriores complexas no contexto bayesiano. Em vez de resolver analiticamente a distribuição dos parâmetros, o algoritmo gera uma cadeia de amostras que, após convergência, representa fielmente a distribuição posterior.

- **draws = 5000:** tamanho total da cadeia após o período de aquecimento
- **burnin = 1000:** as primeiras 1000 iterações são descartadas — a cadeia ainda não convergiu para a distribuição estacionária
- **3 modelos ajustados:** sv_fit (pré), sv_fit2 (pós-queda), sv_fit3 (recuperação), permitindo comparação bayesiana dos regimes

■■ **Custo computacional:** O MCMC em dados de alta frequência (10.000 observações) pode levar de vários minutos a horas. Por isso o código implementa um sistema de cache (.rds files) explicado no próximo bloco.

9 Bloco 8 — Sistema de Cache e Compressão (reduce_and_save)

Um dos blocos mais complexos do código implementa um **sistema inteligente de cache** para evitar reprocessamento desnecessário dos modelos MCMC, que são computacionalmente caros.

9.1 Lógica de Cache

```
flag_file <- "compressed.txt"

if (file.exists(flag_file)) {
# Arquivo já existe: carrega resultados salvos
sv_fit <- readRDS("sv_fit.rds")
sv_fit2 <- readRDS("sv_fit2.rds")
sv_fit3 <- readRDS("sv_fit3.rds")
} else {
# Arquivo não existe: roda MCMC e salva
sv_fit_raw <- svsample(ret_vec, draws=5000, burnin=1000)
sv_fit <- reduce_and_save(sv_fit_raw, "sv_fit.rds", max_draws=2000)
file.create(flag_file) # marca que já foi processado
}
```

`file.exists()` verifica se um arquivo existe no disco. Na primeira execução, o arquivo de flag não existe e o código roda o MCMC completo. Nas execuções seguintes, simplesmente carrega os resultados salvos com `readRDS()`, economizando horas de processamento.

9.2 Função reduce_and_save — Thinning de MCMC

O objeto MCMC completo pode ter centenas de MB. A função reduz seu tamanho mantendo a qualidade estatística:

```
reduce_and_save <- function(obj, file_out, max_draws=2000,
thin_method="systematic") {
thin_rows <- function(mat, max_rows) {
# thinning sistemático: seleciona linhas igualmente espaçadas
idx <- unique(round(seq(1, nrow(mat), length.out=max_rows)))
return(mat[idx, , drop=FALSE])
}
# aplica redução a cada componente da lista do stochvol
for (nm in names(obj)) {
comp <- obj[[nm]]
if (is.matrix(comp)) obj[[nm]] <- thin_rows(comp, max_draws)
```

```
  # arrays 3D: reduz primeira dimensão (draws)
  # vetores grandes: resume com mean/sd/quantis
  }
  saveRDS(obj, file=file_out, compress="xz") # comprime com xz
  }
```

Tipo de Dado	Estratégia de Redução	Justificativa
Matriz (draws × parâmetros)	Thinning sistemático: seleciona a cada k=n/max_draws linhas	Preserva estrutura da distribuição posterior sem viés
Array 3D (draws × T × vars)	Reduz 1ª dimensão com mesmo índice	Mantém correspondência entre draws dos parâmetros e estados latentes
Vetor numérico longo	Amostra aleatória + resumo (mean, sd, q025, q975)	Não é necessário guardar todos os valores brutos
compress='xz'	Compressão máxima ao salvar .rds	Reduz tamanho do arquivo em disco significativamente

10 Bloco 9 — Variância Realizada, Bipower e Identificação de Saltos

Este bloco implementa métricas não-paramétricas para quantificar e decompor a volatilidade realizada — calculadas diretamente dos retornos observados, sem suposições sobre o modelo.

10.1 Variância Realizada (RV)

```
calc_RV <- function(r) {
rv <- sum((r)^2, na.rm = TRUE)
return(rv)
}
```

A **Variância Realizada** é a soma dos quadrados dos retornos intradiários. É um estimador não-paramétrico da variância integrada do processo de preços. Quanto maior a frequência dos dados (1 min vs. 5 min), mais precisa a estimativa, até o limite onde o ruído de microestrutura começa a dominar.

10.2 Variação Bipower (BV)

```
calc_BV <- function(r) {
bv_raw <- sum(abs(r[-1]) * abs(r[-length(r)]), na.rm=TRUE) # |r_t| * |r_{t-1}|
bv <- (pi/2)^(-1) * bv_raw # constante de correção
return(bv)
}
```

A **Variação Bipower** (Barndorff-Nielsen & Shephard, 2004) multiplica retornos absolutos adjacentes. Sua propriedade chave: é **robusta a saltos**. Enquanto RV captura tanto a variância contínua quanto os saltos, BV captura apenas a variância contínua. O fator $\pi/2$ é a constante de normalização que torna BV um estimador não-viesado.

10.3 Identificação de Saltos (Jump)

```
calc_jump_share <- function(r) {
rv <- calc_RV(r)
bv <- calc_BV(r)
jump <- max(rv - bv, 0) # componente de salto (não-negativo)
JumpShare <- jump / rv # proporção da variância explicada por saltos
return(list(RV=rv, BV=bv, Jump=jump, JumpShare=JumpShare))
}
```

Métrica	Fórmula	Interpretação
RV	$\Sigma\, r^2(t)$	Variância total realizada — inclui difusão contínua + saltos

BV	(π/2)■¹ · Σ|r(t)|·|r(t-1)|	Variância contínua apenas — robusta a saltos discretos
Jump	max(RV – BV, 0)	Componente de salto: variância atribuída a movimentos abruptos
JumpShare	Jump / RV	Fração da variância total explicada por saltos (0 a 1)
VolAnn ≈	√(RV/n) · √(252×390)	Volatilidade anualizada aproximada derivada do RV

■ **Resultados obtidos no estudo:** JumpShare ≈ 61–62% em todos os segmentos, indicando que mais da metade da variância realizada é explicada por movimentos abruptos (saltos). Isso é elevado, mas típico em ativos de alta frequência com eventos de notícias.

11 Bloco 10 — Testes Estatísticos Formais

Após as análises exploratórias, testes formais verificam se as diferenças observadas entre segmentos são **estatisticamente significativas** ou poderiam ter ocorrido por acaso.

11.1 Teste Permutacional para Diferença em RV

```
perm_test_RV_diff <- function(r1, r2, R=2000) {
obs_diff <- calc_RV(r2) - calc_RV(r1) # diferença observada
pooled <- c(r1, r2) # pool dos dois grupos
perm_diffs <- numeric(R)
set.seed(123)
for(i in seq_len(R)) {
perm <- sample(pooled) # embaralha aleatoriamente
perm_diffs[i] <- calc_RV(perm[(n1+1):(n1+n2)]) - calc_RV(perm[1:n1])
}
p_value <- mean(abs(perm_diffs) >= abs(obs_diff)) # p-valor bilateral
}
```

O **teste permutacional** é um método não-paramétrico robusto: em vez de supor uma distribuição teórica, ele constrói a distribuição nula empiricamente embaralhando os dados. Se a diferença observada for muito maior que a esperada por acaso, o p-valor será pequeno, indicando diferença significativa.

- **set.seed(123):** garante reprodutibilidade dos resultados
- **R=2000:** número de permutações — mais é melhor mas mais lento
- **Hipótese nula:** os dois segmentos têm a mesma RV (mesma população)

11.2 Teste de Levene (Homogeneidade de Variâncias)

```
combined <- data.frame(
ret = c(rets_A, rets_B, rets_C),
seg = factor(c(rep("A",...), rep("B",...), rep("C",...)))
)
levene_res <- car::leveneTest(ret ~ seg, data=combined)
```

O **Teste de Levene** verifica se os grupos têm variâncias iguais (homocedasticidade). Em finanças, variâncias diferentes entre períodos confirmam mudanças de regime de volatilidade. É mais robusto que o Teste de Bartlett quando as distribuições não são normais — o que é o caso dos retornos financeiros (distribuições com caudas pesadas).

11.3 Teste Kolmogorov-Smirnov (KS)

```
ks_AB <- ks.test(as.numeric(rets[segments$A]), as.numeric(rets[segments$B]))
ks_AC <- ks.test(as.numeric(rets[segments$A]), as.numeric(rets[segments$C]))
```

O **Teste KS** compara as distribuições acumuladas de dois grupos, verificando se provêm da mesma distribuição. Enquanto Levene foca na variância, KS captura diferenças em qualquer parte da distribuição — localização, variância, assimetria e caudas.

Teste	Hipótese Nula	Captura	Premissas
Permutacional RV	RV igual entre grupos	Diferença no nível de variância total	Nenhuma (não-param.)
Levene	Variâncias iguais	Heterogeneidade de variância entre grupos	Nenhuma (não-param.)
KS	Mesma distribuição	Diferença em qualquer aspecto da distribuição	Variáveis contínuas
Bootstrap RV	IC para RV individual	Incerteza do estimador de variância	i.i.d. dentro do bloco

12 Bloco 11 — Comparação Bayesiana das Distribuições Posteriores

Com os três modelos MCMC estimados, este bloco realiza a **comparação bayesiana formal** dos regimes, calculando probabilidades posteriores de diferenças.

12.1 Extração das Amostras Posteriores

```
get_para_mat <- function(svobj) {
pm <- as.matrix(para(svobj, chain="concatenated"))
return(pm) # linhas=draws, colunas=mu,phi,sigma
}
get_latent_mat <- function(svobj) {
lm <- as.matrix(latent(svobj, chain="concatenated"))
return(lm) # linhas=draws, colunas=h_1,...,h_T (volatilidade latente)
}

para1 <- get_para_mat(sv_fit) # pré-queda
para2 <- get_para_mat(sv_fit2) # pós-queda
para3 <- get_para_mat(sv_fit3) # recuperação
```

`para()` extrai a matriz de parâmetros (μ, ϕ, σ) de todos os draws MCMC. `latent()` extrai os estados latentes h(t) — a trajetória da log-volatilidade não observada. Cada linha é um draw MCMC: uma amostra completa e coerente de todos os parâmetros.

12.2 Probabilidades Posteriores de Diferença

```
param_diff <- function(matA, matB, parname) {
diffs <- matB[, parname] - matA[, parname] # diferença draw-a-draw
data.frame(
mean = mean(diffs),
sd = sd(diffs),
q2.5 = quantile(diffs, 0.025),
q97.5 = quantile(diffs, 0.975),
prob_gt0 = mean(diffs > 0) # P(param_B > param_A | dados)
)
}
```

Esta é a essência da **inferência bayesiana comparativa**: para cada par de draws correspondentes dos dois modelos, calcula a diferença dos parâmetros. A proporção de draws onde a diferença é positiva é diretamente a probabilidade posterior P(parâmetro pós > pré | dados) — sem necessidade de p-valores ou testes assintóticos.

12.3 Volatilidade Anualizada Posterior

```
per_draw_ann_vol <- function(latmat) {
# Para cada draw: média das variâncias latentes, depois raiz e anualiza
mean_var <- rowMeans(exp(latmat)) # exp(h_t) = variância latente
ann_vol <- sqrt(mean_var) * sqrt(minutos_ano)
return(ann_vol) # vetor com 1 valor por draw MCMC
}
ann1 <- per_draw_ann_vol(latent1s) # pré
ann2 <- per_draw_ann_vol(latent2s) # pós
ann3 <- per_draw_ann_vol(latent3s) # recov
```

A volatilidade latente h(t) está na escala de log-variância. Para obter a volatilidade em escala original: exp(h(t)/2) = desvio-padrão, ou exp(h(t)) = variância. A média sobre todos os tempos t dá a volatilidade média do período, que multiplicada por √(minutos_ano) converte para escala anual.

Resultados Numéricos Obtidos:

Regime	μ (média posterior)	ϕ (média posterior)	σ (média posterior)	Vol Anualizada
Pré-queda	−13.808	0.969	0.162	≈ 35.2%
Pós-queda	−13.336	0.988	0.114	≈ 46.0%
Recuperação	≈ −14.4	≈ 0.98	≈ 0.11	≈ 26.1%

■ **P(vol_pós > vol_pré | dados)** ≈ **1.00** — aumento de volatilidade é praticamente certo após o choque. **P(vol_recov > vol_pré | dados)** ≈ **0.00** — a volatilidade na recuperação é significativamente menor até que o período pré-queda.

13 Bloco 12 — Visualizações Comparativas Finais

O bloco final cria visualizações sofisticadas para comunicar os resultados da comparação bayesiana de forma clara e informativa.

13.1 Gráfico de Violin + Boxplot

```
p_clean_fix <- ggplot(df_for_plot, aes(x=regime, y=ann_vol, fill=regime)) +
geom_violin(width=0.9, trim=FALSE, alpha=0.35) + # densidade posterior
geom_boxplot(width=0.12, outlier.shape=NA) + # IQR + mediana
stat_summary(fun=median, geom="point", shape=23, fill="white") + # mediana
stat_summary(fun=mean, geom="point", shape=21, fill="red") + # média
geom_errorbar(aes(ymin=q025, ymax=q975)) + # IC 95%
geom_text(aes(label=sprintf("mean=%.3f\n95%%CI=[%.3f,%.3f]", ...)))
```

Camada ggplot2	Função	O que representa
geom_violin()	Estima densidade usando KDE	Forma completa da distribuição posterior de vol anualizada por regime
geom_boxplot()	Box e hastes IQR	Resumo robusto: Q1, mediana, Q3 e whiskers (±1.5 IQR)
stat_summary(median)	Losango branco	Mediana — ponto central da distribuição (menos sensível a outliers)
stat_summary(mean)	Círculo vermelho	Média aritmética — pode ser puxada por caudas da distribuição
geom_errorbar()	Barras verticais	Intervalo de credibilidade 95% bayesiano [Q2.5%, Q97.5%]
geom_text()	Anotações de texto	Valores numéricos de mean e IC95 para cada regime

13.2 Densidade Posterior dos Regimes

```
p <- ggplot(df_plot, aes(x=ann, fill=regime)) +
geom_density(alpha=0.4) + # densidades sobrepostas, transparentes
labs(title="Posterior densities of per-segment annualized volatility")
```

`geom_density()` aplica estimação de densidade por kernel (KDE) sobre os draws MCMC de volatilidade anualizada, criando curvas suaves que representam a distribuição posterior de cada regime. A sobreposição com `alpha=0.4` (transparência) permite ver onde as distribuições se separam ou se sobrepõem — separação perfeita entre 'pós' e 'recov' confirma diferença altamente significativa.

13.3 Gráfico de Preços com Segmentos Destacados

```
p_idx <- ggplot(plot_df_idx, aes(x=index, y=price)) +
geom_rect(data=segments_df_idx,
```

```
  aes(xmin=xmin, xmax=xmax, ymin=-Inf, ymax=Inf, fill=label),
  alpha=0.28) + # regiões coloridas semi-transparentes
  geom_line(color="black", size=0.35) + # série de preços
  geom_vline(aes(xintercept=xmin), linetype="dashed") + # marcadores de início
  scale_fill_manual(values=fill_colors) # cores personalizadas
```

Este gráfico é uma **visualização de contexto**: mostra a série de preços da PETR4 com os três segmentos analíticos destacados por cores. `geom_rect()` cria retângulos do fundo ao topo do gráfico (ymin=-Inf, ymax=Inf) para cada segmento, funcionando como faixas coloridas de fundo. Isso facilita a conexão entre os resultados estatísticos e os eventos de mercado.

14 Resumo dos Resultados e Conclusões

14.1 Síntese das Análises

O estudo empregou uma abordagem multi-método para caracterizar o comportamento da volatilidade da PETR4 ao redor do choque de fevereiro de 2021:

ADF — Estacionariedade	Confirmada nos retornos de 1 minuto; série adequada para modelagem direta
ACF/PACF	Pouca autocorrelação linear nos retornos; alta autocorrelação nos quadrados (clustering de volatilidade)
Volatilidade Histórica 30min	Spike pronunciado no período pós-queda, visível graficamente
Variância Realizada (RV)	RV pós (≈0.021) é 1.8× maior que RV pré (≈0.012); recov (≈0.006) abaixo do pré
JumpShare ≈ 61–62%	Alta participação de saltos em todos os regimes; característica de alta frequência
Testes estatísticos	Perm. test, Levene e KS todos confirmam diferenças significativas entre segmentos
Modelo SV — μ	Aumenta no pós-queda (nível médio de vol mais alto), cai na recuperação abaixo do pré
Modelo SV — ϕ	Aumenta de 0.97 $\rightarrow$ 0.99 no pós (maior persistência), mantém-se alto na recov (0.98)
Modelo SV — σ	Diminui de 0.16 $\rightarrow$ 0.12 no pós $\rightarrow$ 0.11 na recov (choques menos erráticos)
P(vol_pós > vol_pré)	≈ 1.00 — aumento de volatilidade é certeza estatística do ponto de vista bayesiano
P(vol_recov > vol_pré)	≈ 0.00 — recuperação com vol significativamente menor que o pré-choque

14.2 Conclusão Geral

O episódio de queda da PETR4 em fevereiro de 2021 gerou um **choque temporário porém marcante de volatilidade**. O modelo de volatilidade estocástica capta nuances que a volatilidade realizada simples não revela: enquanto o nível médio aumentou e a persistência se elevou (choques duram mais), a volatilidade da própria volatilidade diminuiu — sugerindo que o mercado entrou em um regime de alta volatilidade, mas de forma mais ordenada. A recuperação, por sua vez, levou a níveis de volatilidade inferiores até mesmo ao período pré-choque, consistente com a resolução da incerteza sobre o novo cenário da empresa.

14.3 Ferramentas Utilizadas — Guia de Referência Rápida

Conceito	Função R	Onde Estudar Mais
Séries temporais xts	as.xts(), rollapply(), index()	Pacote xts — CRAN

Manipulação de dados	mutate(), filter(), select(), %>%	R for Data Science (Hadley Wickham)
Volatilidade histórica	sd() em rollapply()	Hull — Options, Futures (Cap. 19–20)
Teste ADF	tseries::adf.test()	Enders — Applied Econometric TS
ACF/PACF	acf(), pacf()	Box-Jenkins — Time Series Analysis
Volatilidade Estocástica MCMC	stochvol::svsample()	Kastner (2016) — JSS
Variância Realizada	sum(r^2)	Andersen et al. (2001) — JFQA
Bipower Variation	(pi/2)^(-1)*sum(|r_t|*|r_{t-1}|)	Barndorff-Nielsen & Shephard (2004)
Teste Permutacional	Implementado manualmente via sample()	Good — Permutation Tests
Violinplot	ggplot2::geom_violin()	ggplot2 documentation