

```

if (is.null(df)) {
  stop("Execution halted because the data file could not be loaded.")
}

# 3. Convert Date, define commodities/models
df$Date <- as.Date(df$Date, format = "%Y-%m-%d")
commodities <- c(
  "Red Bird's Eye Chili",
  "Regular Grade Rice",
  "Packaged Cooking Oil",
  "Broiler Chicken Meat",
  "Broiler Chicken Eggs"
)
garch_models_to_test <- c("sGARCH", "gjrGARCH", "eGARCH")

## Initialize data frames for summaries and plots ##
model_selection_summary <- data.frame()
final_parameter_summary <- data.frame()
all_volatilities_df <- data.frame(Date = df$Date)
best_fits_storage <- list() # To store the best model for each commodity

# 4. Loop over each commodity
for (comm in commodities) {

```

```

cat("\n=====
=====\n")

cat("Processing Commodity:", comm, "\n")

cat("=====
=====\n\n")

# 4a. Compute log-returns
ret_name <- paste0(gsub(" ", "_", comm), "_ret")
df[[ret_name]] <- c(NA, diff(log(df[[comm]])))
ret_series <- na.omit(df[[ret_name]])

fitted_models_list <- list()
aic_values <- c()
bic_values <- c()

# 4b. Loop over GARCH models
for (model_name in garch_models_to_test) {
  cat("----- Fitting", model_name, "-----\n")
  spec <- ugarchspec(
    variance.model = list(model = model_name, garchOrder = c(1, 1)),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE),
    distribution.model = "std"
  )
  fit <- try(ugarchfit(spec = spec, data = ret_series, solver = "hybrid"), silent =
TRUE)

  if (class(fit) != "try-error") {

```

```

    fitted_models_list[[model_name]] <- fit
    aic_values[model_name] <- infocriteria(fit)["Akaike", ]
    bic_values[model_name] <- infocriteria(fit)["Bayes", ]
    cat("Fit successful for", model_name, ".\n\n")
  } else {
    aic_values[model_name] <- NA
    bic_values[model_name] <- NA
    cat("Model", model_name, "failed to converge.\n\n")
  }
}

# 4c. Determine best model and populate summaries
if (length(fitted_models_list) > 0) {
  best_model_name <- names(which.min(bic_values))
  cat(">>> Best model for", comm, "based on BIC is:", best_model_name,
"<<<\n")

  # Store the best fit for later use in VaR calculation
  best_fits_storage[[comm]] <- fitted_models_list[[best_model_name]]

  # Add row to Model Selection table
  selection_row <- data.frame(
    Commodity = comm,
    sGARCH_AIC = aic_values["sGARCH"], sGARCH_BIC =
bic_values["sGARCH"],
    gjrGARCH_AIC = aic_values["gjrGARCH"], gjrGARCH_BIC =
bic_values["gjrGARCH"],
    eGARCH_AIC = aic_values["eGARCH"], eGARCH_BIC =
bic_values["eGARCH"],

```

```

    Selected = best_model_name
  )

  model_selection_summary <- rbind(model_selection_summary,
selection_row)

  best_model_fit <- fitted_models_list[[best_model_name]]
  cfs <- coef(best_model_fit)

  # Handle different parameter names for asymmetric models
  alpha_val <- ifelse("alpha1" %in% names(cfs), cfs["alpha1"], NA)
  gamma_val <- ifelse("gamma1" %in% names(cfs), cfs["gamma1"], NA) #
For GJR/eGARCH

  # Add row to Parameter Estimation table
  parameter_row <- data.frame(
    Commodity = comm, mu = cfs["mu"], AR1 = cfs["ar1"],
    omega = cfs["omega"], alpha1 = alpha_val, beta1 = cfs["beta1"],
    gamma1 = gamma_val, # Leverage effect term
    shape = cfs["shape"]
  )

  final_parameter_summary <- rbind(final_parameter_summary,
parameter_row)

  # Add volatility data to the plot data frame
  cond_vol <- sigma(best_model_fit)
  dates_for_vol <- df$Date[!is.na(df[[ret_name]])]
  temp_vol_df <- data.frame(Date = dates_for_vol, Vol =
as.numeric(cond_vol))
  names(temp_vol_df)[2] <- gsub("[^a-zA-Z0-9_]", "_", comm)

```

```

    all_volatilities_df <- merge(all_volatilities_df, temp_vol_df, by = "Date",
all.x = TRUE)

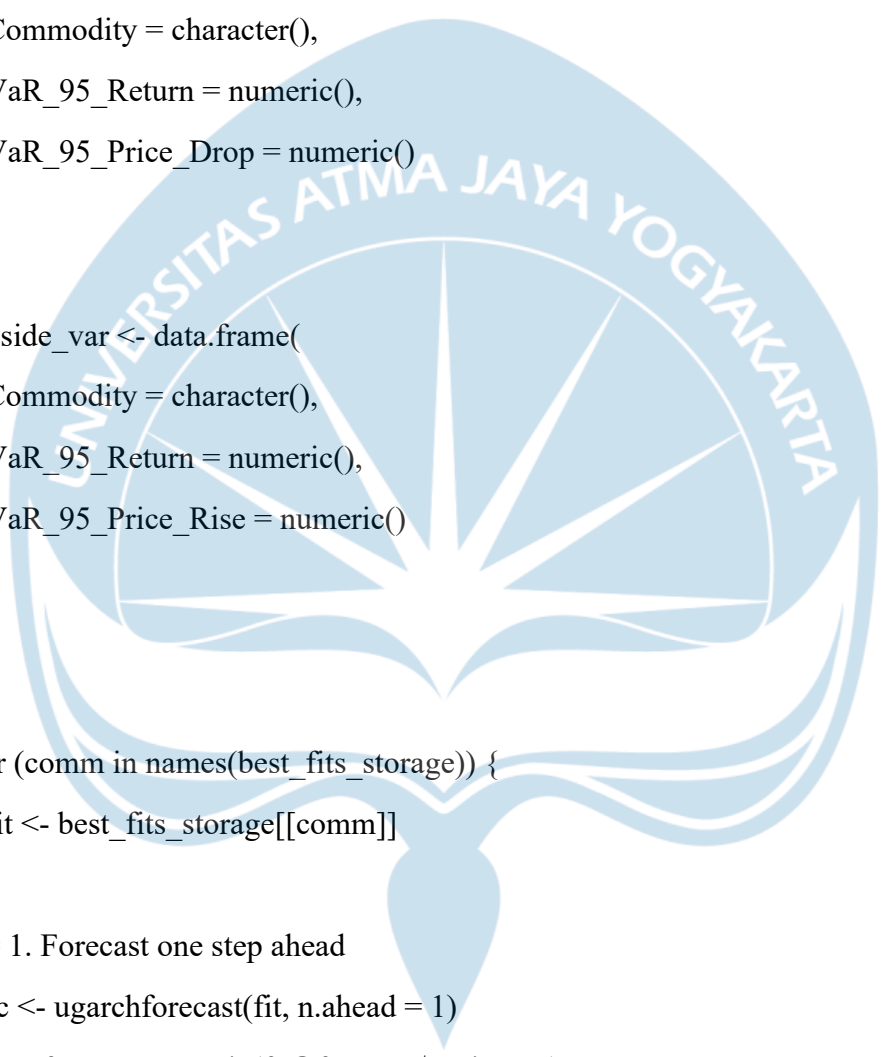
} else {
    cat(">>> Could not determine the best model for", comm, ".\n")
}
}

# 5. Print the final summary tables
cat("\n\n=====
\n")
cat("          TABLE 1: MODEL SELECTION SUMMARY\n")
cat("=====
\n\n")
# Use kable for cleaner table printing
kable(model_selection_summary, caption = "Model Selection Summary based
on Information Criteria")

cat("\n\n=====
\n")
cat("          TABLE 2: PARAMETER ESTIMATION SUMMARY\n")
cat("=====
\n\n")
# Use kable for cleaner table printing
kable(final_parameter_summary, caption = "Parameter Estimates of Best
Fitting GARCH Models", digits = 4)

# 6. Calculate 95% Value at Risk (VaR)

```



```

alpha_level <- 0.95

# Initialize data frames for VaR results
downside_var <- data.frame(
  Commodity = character(),
  VaR_95_Return = numeric(),
  VaR_95_Price_Drop = numeric()
)

upside_var <- data.frame(
  Commodity = character(),
  VaR_95_Return = numeric(),
  VaR_95_Price_Rise = numeric()
)

for (comm in names(best_fits_storage)) {
  fit <- best_fits_storage[[comm]]

  # 1. Forecast one step ahead
  fc <- ugarchforecast(fit, n.ahead = 1)
  mu_fc <- as.numeric(fc@forecast$seriesFor)
  sd_fc <- as.numeric(fc@forecast$sigmaFor)
  shape <- coef(fit)["shape"]

  # Get the last observed price to convert VaR from return to price
  last_price <- tail(df[[comm]], 1)

```

```

# --- Calculate 95% Downside VaR (Maximum Loss) ---
q_down <- qdist("std", p = (1 - alpha_level), shape = shape)
var_ret_down <- mu_fc + sd_fc * q_down
var_price_drop <- last_price * (exp(var_ret_down) - 1)

downside_var <- rbind(downside_var, data.frame(
  Commodity = comm,
  VaR_95_Return = var_ret_down,
  VaR_95_Price_Drop = var_price_drop
))

# --- Calculate 95% Upside VaR (Maximum Gain) ---
q_up <- qdist("std", p = alpha_level, shape = shape)
var_ret_up <- mu_fc + sd_fc * q_up
var_price_rise <- last_price * (exp(var_ret_up) - 1)

upside_var <- rbind(upside_var, data.frame(
  Commodity = comm,
  VaR_95_Return = var_ret_up,
  VaR_95_Price_Rise = var_price_rise
))
}

# 7. Print VaR Tables
cat("\n\n\n=====
=====\\n")

cat("      TABLE 3: 95% DOWNSIDE VAR (MAXIMUM PRICE
DROP)\\n")

```

```

cat("=====\n\n")

kable(downside_var, caption = "One-Day 95% Downside Value at Risk
Forecast", digits = 4)

cat("\n\n=====\n\n")

cat("      TABLE 4: 95% UPSIDE VAR (MAXIMUM PRICE RISE)\n")

cat("=====\n\n")

kable(upside_var, caption = "One-Day 95% Upside Value at Risk Forecast",
digits = 4)

# 8. Create the combined volatility plot

cat("\n\n=====\n\n")

cat("      PLOT 1: CONDITIONAL VOLATILITY ( $\sigma_t$ )\n")

cat("=====\n\n")

# Set a graphical parameter to ensure the plot is generated
par(mar = c(5, 5, 2, 2)) # Adjust margins if needed

ylimits <- range(all_volatilities_df[, -1], na.rm = TRUE)
colors <- rainbow(length(commodities))
plot_col_names <- gsub("[^a-zA-Z0-9_]", "_", commodities)

plot(all_volatilities_df$Date, all_volatilities_df[[plot_col_names[1]]],

```

```

    type = 'l', col = colors[1], ylim = ylims,

    xlab = "Date", ylab = expression(paste("Conditional Volatility (", sigma[t],
    "))), main = "Conditional Volatility of Staple Food Commodities"
  )

  for (i in 2:length(commodities)) {
    lines(all_volatilities_df$Date, all_volatilities_df[[plot_col_names[i]]], type =
    'l', col = colors[i])
  }

  legend("topright", legend = commodities, col = colors, lty = 1, cex = 0.8, bg =
  "white")

```

R studio Results

```

# Complete R script: Fit, Compare, Plot Volatility, and Calculate VaR
>
> # 0. Clean session
> rm(list = ls())

> graphics.off()

> # 1. Install & load required packages
> pkgs <- c("readxl", "rugarch", "knitr")

> to_install <- pkgs[!pkgs %in% installed.packages()[, "Package"]]

> if (length(to_install)) install.packages(to_install, dependencies = TRUE)

```

```
> library(readxl)

> library(rugarch)

> library(knitr)

> # 2. Read your data (UPDATE THIS PATH)
> df <- NULL

> tryCatch({
+   # Use the path to your actual data file
+   df <- read_excel("~/Downloads/Big Data Final Commodiites.xlsx")
+ }, error = function(e) .... [TRUNCATED]

> if (is.null(df)) {
+   stop("Execution halted because the data file could not be loaded.")
+ }

> # 3. Convert Date, define commodities/models
> df$Date <- as.Date(df$Date, format = "%Y-%m-%d")

> commodities <- c(
+   "Red Bird's Eye Chili",
+   "Regular Grade Rice",
+   "Packaged Cooking Oil",
+   "Broiler Chicken Meat",
+   "Broiler Chicken ..." ... [TRUNCATED]
```

```

> garch_models_to_test <- c("sGARCH", "gjrGARCH", "eGARCH")

> ## Initialize data frames for summaries and plots ##
> model_selection_summary <- data.frame()

> final_parameter_summary <- data.frame()

> all_volatilities_df <- data.frame(Date = df$Date)

> best_fits_storage <- list() # To store the best model for each commodity

> # 4. Loop over each commodity
> for (comm in commodities) {
+
+
+ cat("\n=====
=====\\n")
+ cat("P ..." ... [TRUNCATED]
=====
=
Processing Commodity: Red Bird's Eye Chili
=====
=

----- Fitting sGARCH -----
Fit successful for sGARCH .

----- Fitting gjrGARCH -----

```

Fit successful for gjrGARCH .

----- Fitting eGARCH -----

Fit successful for eGARCH .

>>> Best model for Red Bird's Eye Chili based on BIC is: eGARCH <<<

=====

=

Processing Commodity: Regular Grade Rice

=====

=

----- Fitting sGARCH -----

Fit successful for sGARCH .

----- Fitting gjrGARCH -----

Fit successful for gjrGARCH .

----- Fitting eGARCH -----

Fit successful for eGARCH .

>>> Best model for Regular Grade Rice based on BIC is: sGARCH <<<

=====

=

Processing Commodity: Packaged Cooking Oil

=====

=

----- Fitting sGARCH -----

Fit successful for sGARCH .

----- Fitting gjrGARCH -----

Fit successful for gjrGARCH .

----- Fitting eGARCH -----

Fit successful for eGARCH .

>>> Best model for Packaged Cooking Oil based on BIC is: eGARCH <<<

=====

=

Processing Commodity: Broiler Chicken Meat

=====

=

----- Fitting sGARCH -----

Fit successful for sGARCH .

----- Fitting gjrGARCH -----

Fit successful for gjrGARCH .

----- Fitting eGARCH -----

Fit successful for eGARCH .

>>> Best model for Broiler Chicken Meat based on BIC is: eGARCH <<<

```

=====
=
Processing Commodity: Broiler Chicken Eggs
=====
=

---- Fitting sGARCH ----
Fit successful for sGARCH .

---- Fitting gjrGARCH ----
Fit successful for gjrGARCH .

---- Fitting eGARCH ----
Fit successful for eGARCH .

>>> Best model for Broiler Chicken Eggs based on BIC is: eGARCH <<<

> # 5. Print the final summary tables
>
cat("\n\n=====
=====\\n")

=====
=

> cat("          TABLE 1: MODEL SELECTION SUMMARY\\n")

```

TABLE 1: MODEL SELECTION SUMMARY

```
>
cat("=====
=====\n\n")
=====
=
```

```
> # Use kable for cleaner table printing
> kable(model_selection_summary, caption = "Model Selection Summary
based on Information Criteria")
```

Table: Model Selection Summary based on Information Criteria

	Commodity	sGARCH_AIC	sGARCH_BIC	gjrGARCH_AIC	gjrGARCH_BIC	eGARCH_AIC	eGARCH_BIC	Selected
sGARCH	Red Bird's Eye Chili	-4.666363	-4.645179	-4.665035	-	4.640320	-4.670270	-4.645556 eGARCH
sGARCH1	Regular Grade Rice	-10.605044	-10.583860	-10.607820	-	10.583106	-10.568676	-10.543961 sGARCH
sGARCH2	Packaged Cooking Oil	-6.759824	-6.738640	-6.759450	-	6.734735	-6.765667	-6.740952 eGARCH
sGARCH3	Broiler Chicken Meat	-7.710831	-7.689647	-7.710070	-	7.685356	-7.725903	-7.701188 eGARCH
sGARCH4	Broiler Chicken Eggs	-7.940629	-7.919445	-7.939436	-	7.914721	-7.954613	-7.929898 eGARCH

```

>
cat("\n\n=====
=====\\n")

=====

> cat("
TABLE 2: PARAMETER ESTIMATION SUMMARY\\n")
TABLE 2: PARAMETER ESTIMATION SUMMARY

>
cat("=====
=====\\n\\n")

=====

> # Use kable for cleaner table printing
> kable(final_parameter_summary, caption = "Parameter Estimates of Best
Fitting GARCH Models", digits = 4)

```

Table: Parameter Estimates of Best Fitting GARCH Models

	Commodity	mu	AR1	omega	alpha1	beta1	gamma1	shape
	:--:	:-----:	:-----:	:-----:	:-----:	:-----:	:-----:	:-----:
mu	Red Bird's Eye Chili	-3e-04	0.5562	-1.5419	-0.0153	0.7720	0.7271	
		2.5241						

```
|mu1 |Regular Grade Rice | 0e+00| 0.0071| 0.0000| 0.7236| 0.2754| NA|
2.3689|
|mu2 |Packaged Cooking Oil | -1e-04| 0.4540| -1.3069| -0.0079| 0.8518| 0.3820|
2.4775|
|mu3 |Broiler Chicken Meat | 0e+00| 0.5907| -1.3030| 0.0171| 0.8654| 0.5500|
2.4104|
|mu4 |Broiler Chicken Eggs | -6e-04| 0.5445| -1.5470| 0.0444| 0.8405| 0.7102|
2.3093|
```

```
> # 6. Calculate 95% Value at Risk (VaR)
```

```
> alpha_level <- 0.95
```

```
> # Initialize data frames for VaR results
```

```
> downside_var <- data.frame(
+ Commodity = character(),
+ VaR_95_Return = numeric(),
+ VaR_95_Price_ .... [TRUNCATED]
```

```
> upside_var <- data.frame(
+ Commodity = character(),
+ VaR_95_Return = numeric(),
+ VaR_95_Price_Rise = numeric()
+ )
```

```
> for (comm in names(best_fits_storage)) {
+ fit <- best_fits_storage[[comm]]
+
+ # 1. Forecast one step ahead
+ fc <- ugarchforecast(fit, n. .... [TRUNCATED]
```

```
> # 7. Print VaR Tables
```

```
>
```

```
cat("\n\n\n=====  
=====\n")
```

```
=====  
=
```

```
> cat("    TABLE 3: 95% DOWNSIDE VAR (MAXIMUM PRICE  
DROP)\n")
```

```
    TABLE 3: 95% DOWNSIDE VAR (MAXIMUM PRICE DROP)
```

```
>
```

```
cat("=====  
=====\n\n")
```

```
=====  
=
```

```
> kable(downside_var, caption = "One-Day 95% Downside Value at Risk  
Forecast", digits = 4)
```

Table: One-Day 95% Downside Value at Risk Forecast

Commodity	VaR_95_Return	VaR_95_Price_Drop
:-----	-----:	-----:

$$| \begin{smallmatrix} : \\ : \end{smallmatrix} \text{---} | \text{---} | \begin{smallmatrix} : \\ : \end{smallmatrix} \text{---} | \begin{smallmatrix} : \\ : \end{smallmatrix} \text{---} | \begin{smallmatrix} : \\ : \end{smallmatrix} \text{---} |$$

Regular Grade Rice	0.0199	240.3534
--------------------	--------	----------

Broiler Chicken Meat	0.0065	222.4663
----------------------	--------	----------

> # 8. Create the combined volatility plot

```
cat("\n\n\n=====  
=====\n")
```

```
> cat("PLOT 1: CONDITIONAL VOLATILITY ( $\sigma_t$ )\n")
```

```
cat("=====  
====\n\n")
```

```
> # Set a graphical parameter to ensure the plot is generated
```

```

> par(mar = c(5, 5, 2, 2)) # Adjust margins if needed

> ylimits <- range(all_volatilities_df[, -1], na.rm = TRUE)

> colors <- rainbow(length(commodities))

> plot_col_names <- gsub("[^a-zA-Z0-9_]", "_", commodities)

> plot(all_volatilities_df$Date, all_volatilities_df[[plot_col_names[1]]],
+   type = 'l', col = colors[1], ylim = ylimits,
+   xlab = "Date", y .... [TRUNCATED]

> for (i in 2:length(commodities)) {
+   lines(all_volatilities_df$Date, all_volatilities_df[[plot_col_names[i]]], type
+   = 'l', col = colors[i])
+ }

> legend("topright", legend = commodities, col = colors, lty = 1, cex = 0.8, bg =
+   "white")

```