Champion-Model.R

gavin-kunish

2025-08-18

```
# Champion Model Plots

# ======= Libraries =======
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(vars)
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

```
library(forecast)
library(zoo)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(scales)

# ======= Load & basic transforms (assumes monthly dates on the 1st) =======
# setwd("~/Desktop")
data <- read.csv("combined_time_series.csv")
data$Date <- as.Date(data$Date)

# Core transforms used earlier
log_CPI <- log(data$CPI)
diff_log_cpi <- diff(log_CPI)
final_CPI <- diff_log_cpi

lambda_ppi <- BoxCox.lambda(data$PPI)
PPI_bc <- BoxCox(data$PPI, lambda = lambda_ppi)
final_PPI <- diff(PPI_bc)

housing_bc <- BoxCox(data$HousingIndex, lambda = "auto")
diff_housing <- diff(housing_bc)
final_housing <- diff(diff_housing)  # as used previously

final_Unemp <- diff(data$Unemployment)
final_FedFund <- diff(data$FedFunds)

# ======= Align as zoo and merge (keep common intersection) =======
z_CPI   <- zoo(final_CPI,          order.by = data$Date[-1])
z_PPI   <- zoo(final_PPI,          order.by = data$Date[-1])
z_House <- zoo(final_housing,      order.by = data$Date[-(1:2)])
z_Unemp <- zoo(final_Unemp,        order.by = data$Date[-1])
z_Fund  <- zoo(final_FedFund,      order.by = data$Date[-1])

Z <- merge(CPI = z_CPI,
           PPI = z_PPI,
           FedFund = z_Fund,
           Housing = z_House,
           all = FALSE)

covid_step_all <- zoo(
  as.integer(data$Date >= as.Date("2020-12-01") & data$Date <= as.Date("2022-05-01")),
  order.by = data$Date
)
Z2 <- merge(Z, COVID = covid_step_all, all = FALSE)

# Helper to convert zoo -> monthly ts (preserves calendar start)
zoo_to_ts <- function(z) {
  idx <- index(z)
  start <- c(as.integer(format(min(idx), "%Y")), as.integer(format(min(idx), "%m")))
  ts(coredata(z), start = start, frequency = 12)
}

Y_full <- zoo_to_ts(Z2[, c("CPI","PPI","FedFund","Housing")])
X_full <- zoo_to_ts(Z2[, "COVID", drop = FALSE])

vars_in <- c("PPI","FedFund","Housing")
y_model <- Y_full[, c("CPI", vars_in)]
x_model <- X_full

# ======= Force test = Jan-Dec 2024, train = everything strictly before Jan 2024 =======
y_dates <- as.Date(as.yearmon(time(y_model)))

test_start <- as.Date("2024-01-01")
test_end   <- as.Date("2024-12-01")  # monthly timestamp convention

train <- which(y_dates <  test_start)
test  <- which(y_dates >= test_start & y_dates <= test_end)

if (length(test) != 12) {
  stop(paste0("Expected 12 test months in 2024, but found ", length(test),
              ". Check your input dates; y_dates range is ", min(y_dates), " to ", max(y_dates),
              "."))
}

# Confirm split dates
last_train_date <- max(y_dates[train])  # should be 2023-12-01
message("Train ends at: ", last_train_date)
```

```
## Train ends at: 2023-12-01
```

```
message("Test spans: ", min(y_dates[test]), " to ", max(y_dates[test]))
```

```
## Test spans: 2024-01-01 to 2024-12-01
```

```
# ======= Fit VAR (choose p as before or reselect) =======
p <- 6  # keep your chosen lag
var_mod <- VAR(y_model[train, ], p = p, type = "const", exogen = x_model[train, , drop = FALSE], season = 12)

# ======= Forecast exactly over 2024 =======
h <- length(test)
fc <- predict(var_mod, n.ahead = h, dumvar = x_model[test, , drop = FALSE], ci = 0.95)

# Build tс objects for CPI diffing forecasts and actuals aligned to 2024
cpi_fc  <- ts(fc$fcst$CPI[,"fcst"],  start = time(y_model)[test][1], frequency = 12)
cpi_act <-                 y_model[test, "CPI"]

# ======= Invert back to level =======
# Grab last observed log(CPI) right before 2024 (i.e., Dec 2023)
last_log_cpi_train <- log(data$CPI[match(last_train_date, data$Date)])
if (is.na(last_log_cpi_train)) {
  # Fallback if dates are not exact day matches
  closest_idx <- which.min(abs(as.numeric(data$Date - last_train_date)))
  last_log_cpi_train <- log(data$CPI[closest_idx])
}

inverted_fcast_cpi  <- exp(cumsum(cpi_fc) + last_log_cpi_train)
inverted_actual_cpi <- exp(cumsum(cpi_act) + last_log_cpi_train)

inverted_fcast_cpi
```

```
##  [1] 308.1562 309.3945 310.7020 311.4589 312.3027 313.0580 313.3015 313.8215
## [9] 314.5926 316.8843 314.5416 314.0634
```

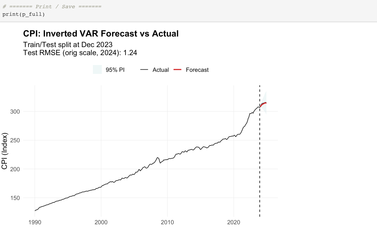
```
inverted_actual_cpi
```

```
##  [1] 308.417 310.326 312.332 313.548 314.069 314.175 314.540 314.796 315.301
## [10] 315.664 315.493 315.605
```

```
lower_cpi_test <- as.numeric(fc$fcst$CPI[, "lower"])
upper_cpi_test <- as.numeric(fc$fcst$CPI[, "upper"])

inverted_lower_cpi <- exp(cumsum(lower_cpi_test) + last_log_cpi_train)
inverted_upper_cpi <- exp(cumsum(upper_cpi_test) + last_log_cpi_train)

# ======= Plotting data frames =======
fc_df <- tibble(
  Date = y_dates[test],
  Forecast = as.numeric(inverted_fcast_cpi),
  Lower = as.numeric(inverted_lower_cpi),
  Upper = as.numeric(inverted_upper_cpi)
)

actual_df <- tibble(
  Date = data$Date,
  Actual = data$CPI
)

rmse_inv <- sqrt(mean((inverted_fcast_cpi - inverted_actual_cpi)^2, na.rm = TRUE))
message("Test RMSE (original scale, 2024): ", round(rmse_inv, 3))
```

```
## Test RMSE (original scale, 2024): 1.239
```

```
# ======= Tech-ready ggplot objects =======
base_theme <- theme_minimal(base_size = 13) +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_line(linewidth = 0.25),
    panel.grid.major.y = element_line(linewidth = 0.25),
    plot.title = element_text(face = "bold"),
    plot.subtitle = element_text(margin = margin(b = 8)),
    axis.title.x = element_blank(),
    legend.position = "top",
    legend.title = element_blank()
  )

# 1) Full history with clear split at Dec 2023 (kept)
p_full <- actual_df %>%
  ggplot(aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_vline(xintercept = as.numeric(last_train_date), linetype = "dashed") +
  geom_ribbon(data = fc_df, aes(x = Date, ymin = Lower, ymax = Upper, fill = "95% PI"), alpha = 0.15, inherit.aes
= FALSE) +
  geom_line(data = fc_df, aes(y = Forecast, color = "Forecast"), linewidth = 0.9) +
  scale_x_date(labels = label_date("'%Y")) +
  scale_color_manual(values = c("Actual" = "#1f1f1f", "Forecast" = "#d62728")) +
  scale_fill_manual(values = c("95% PI" = "#9ecae1")) +
  labs(title = "CPI: Inverted VAR Forecast vs Actual",
       subtitle = paste0("Train/Test split at ", format(last_train_date, "%b %Y"),
                         "\nTest RMSE (orig scale, 2024): ", round(rmse_inv, 2)),
       y = "CPI (Index)") +
  base_theme

# 2) Zoomed 2024 view WITHOUT the split line
zoom_start <- as.Date("2023-01-01")
zoom_end   <- as.Date("2024-12-31")

p_zoom <- actual_df %>%
  filter(Date >= zoom_start & Date <= zoom_end) %>%
  ggplot(aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_ribbon(data = fc_df, aes(x = Date, ymin = Lower, ymax = Upper, fill = "95% PI"), alpha = 0.15, inherit.aes
= FALSE) +
  geom_line(data = fc_df, aes(y = Forecast, color = "Forecast"), linewidth = 0.9) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y", limits = c(zoom_start, zoom_end)) +
  scale_color_manual(values = c("Actual" = "#1f1f1f", "Forecast" = "#d62728")) +
  scale_fill_manual(values = c("95% PI" = "#9ecae1")) +
  labs(title = "CPI: 2024 Forecast Horizon",
       subtitle = "Full Jan–Dec 2024 test set",
       y = "CPI (Index)") +
  base_theme +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# 3) Residuals (training period)
train_dates_aligned <- y_dates[train]
res_vec <- as.numeric(resid(var_mod)[, "CPI"])                  # length = length(train) - p
res_dates <- tail(train_dates_aligned, length(res_vec))
resid_df <- tibble(Date = res_dates, Resid = res_vec)

p_resid <- ggplot(resid_df, aes(Date, Resid)) +
  geom_hline(yintercept = 0, linewidth = 0.3, color = "#666666") +
  geom_line(color = "#3182bd") +
  labs(title = "VAR(CPI) Residuals (Training)", y = "Residual") +
  base_theme

# ======= Print / Save =======
print(p_full)
```




```
print(p_zoom)
```