501 - Assignment 3

Isaac Qi

2025-11-14

```r
library(tidyverse)
library(zoo) # rollapply
library(kableExtra) # table background
```

Write 2–3 sentences in your report naming the file(s) you used, the time span, and the key variables, and briefly justify your choice. Then load the data with datetimes parsed correctly and prices as numeric. To be specific, end the sample on August 31, 2025.

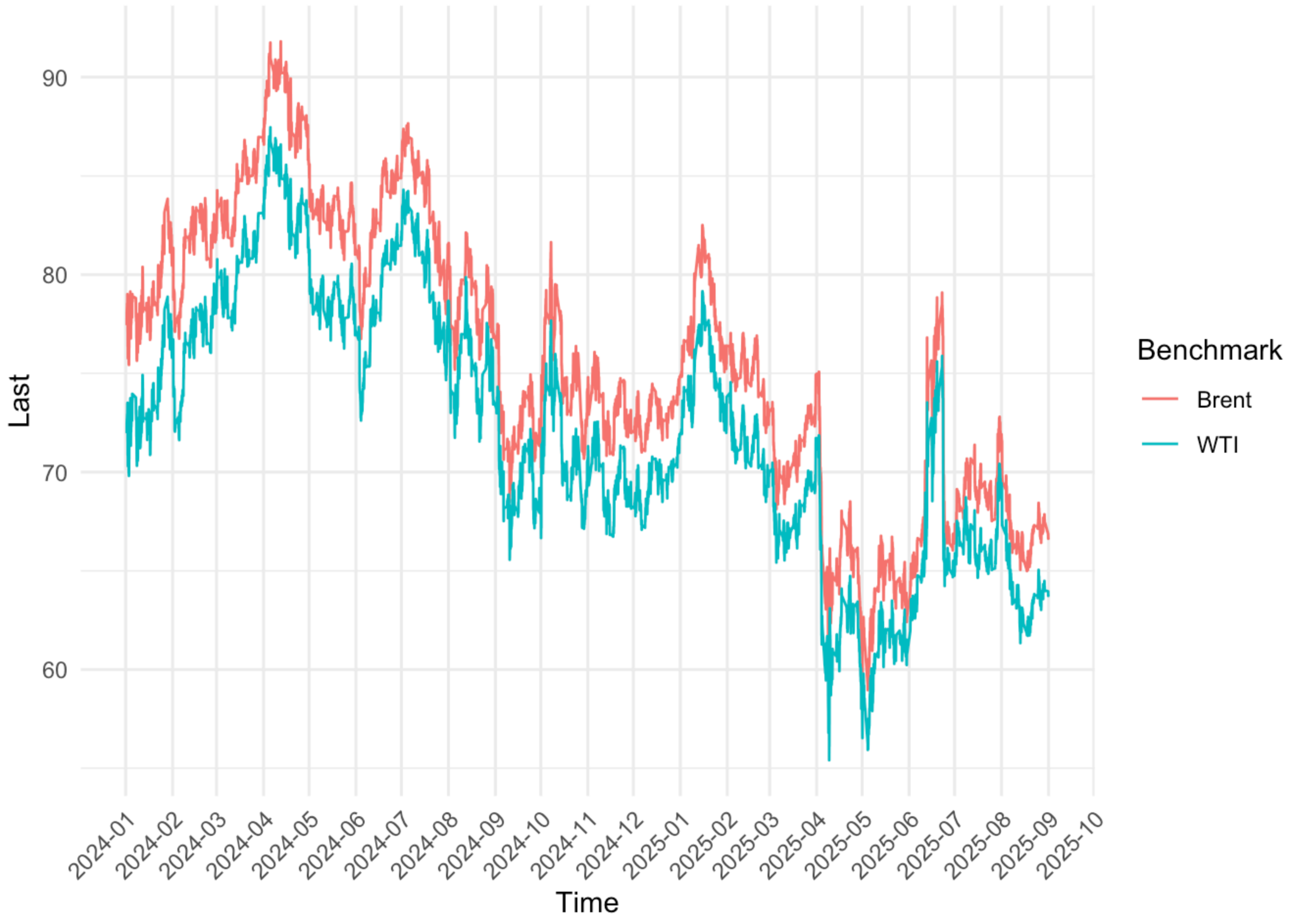
I used 2 csv files housed in the “data” folder: “WTI_Crude.csv” and “Brent_Crude.csv”. The raw data spans 2024-01-01 to 2025-08-31, and variables includes Symbol, Time, Open, High, Low, Last, Change, % Change, Volume, and Open Interest. I focused on Last as the price variable because it represents the most recently traded price at each hourly interval. Compared to Open which includes potential noise from overnight activities, and high / low which represents fluctuating range extremes, Last provides the most realistic basis for constructing the Brent–WTI spread in the absence of bid–ask data.

```r
# relative paths for data import accounting for MAC & PC differences & format date &
price
WTI_raw <- read.csv(file.path("data", "WTI_Crude.csv")) %>%
  mutate(
    Time = ymd_hm(Time),
    across(c(Open, High, Low, Last, Change), ~ as.numeric(.))
    )

Brent_raw <- read.csv(file.path("data", "Brent_Crude.csv")) %>%
  mutate(
    Time = ymd_hm(Time),
    across(c(Open, High, Low, Last, Change), ~ as.numeric(.))
  )

data_raw <- bind_rows(
  WTI_raw  %>% mutate(Benchmark = "WTI"),
  Brent_raw %>% mutate(Benchmark = "Brent")
) %>%
  group_by(Time) %>%
  filter(n_distinct(Benchmark) == 2) %>%   # keep only times where both WTI & Brent e
xists
  ungroup()
```

```
# exploratory plotting
ggplot(data_raw, aes(x = Time, y = Last, colour = Benchmark)) +
  geom_line() +
  scale_x_datetime(date_breaks = "1 month", date_labels = "%Y-%m") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), # tilt 45 degrees for visi
bility
        panel.grid.minor.x = element_blank())   # remove minor vertical lines
```



This exploratory figure confirms a few of my understandings. Brent typically trades at a premium compared to WTI given its proximity to transport infrastructure, while WTI is more so affected by U.S. specific constraints (i.e. storage, pipeline, etc). This structural difference forms the basis of a spread trading strategy. When the Brent–WTI spread widens beyond its usual range, a trader expecting convergence would short Brent and long WTI; when the spread narrows unusually, they would long Brent and short WTI. This strategy hinges on the spread's tendency to revert to a long run equilibrium, also known as mean reversion.

Choose your start date after plotting so the period includes obvious trading opportunities. Report your chosen window and why.

Visual inspection shows that from January 2024 to March 2024, the spread between WTI and Brent appears stable which reflects a lack of trading opportunities. Beginning in April 2024, there appears to be many convergences and divergences. Since we want a baseline for the mean difference between WTI and Brent, I will begin my chosen window one month earlier than April 1st, 2024, with my chosen window starting on March 1st, 2024 and ending on August 31, 2025.

```r
data_dated <- data_raw %>%
  filter(
    Time > "2024-03-01 00:00:00"
  )
```

RQ1. When is WTI vs Brent “out of line” at the hourly horizon?

Define a state variable that flags when the relative price of Brent vs WTI looks unusually high or low, and explain why this construction is suitable for mean reversion.

```r
# need to have both Brent and WTI in one row in order to calculate spread
data_wide <- data_dated %>%
  select(Time, Benchmark, Last) %>%
  pivot_wider(
    names_from = Benchmark,
    values_from = Last,
    names_glue = "{Benchmark}_Last"
  ) %>%
  arrange(Time)
```

I’ve opted to do a hedge adjusted spread. Although Brent and WTI move closely together, they are influenced by different regional and global factors that causes their volatilities to differ. Using a price spread of simply (Brent - WTI) assumes that the two share a 1-1 hedging relationship. However, Brent tends to be more volatile because it is more influenced by global pricing and geopolitical shocks. To adjust for this difference, I run a regression regressing Brent’s last price on WTI’s last price. The slope coefficient beta1 represents how much Brent typically changes for a one-unit change in WTI, and therefore how many barrels of WTI are needed to hedge one barrel of Brent.

I chose the hedge-adjusted spread instead of a price ratio because ratios are nonlinear and can generate exaggerated or misleading signals when the denominator moves quickly. The hedge-adjusted spread is a linear and economically interpretable measure that more accurately reflects the true movement between Brent and WTI.

```r
# regress Brent on WTI
hedge_model <- lm(Brent_Last ~ WTI_Last, data = data_wide)

# extract beta1 or the 2nd coefficient
beta_hat <- coef(hedge_model)[2]
beta_hat
```

```
## WTI_Last
## 1.058812
```

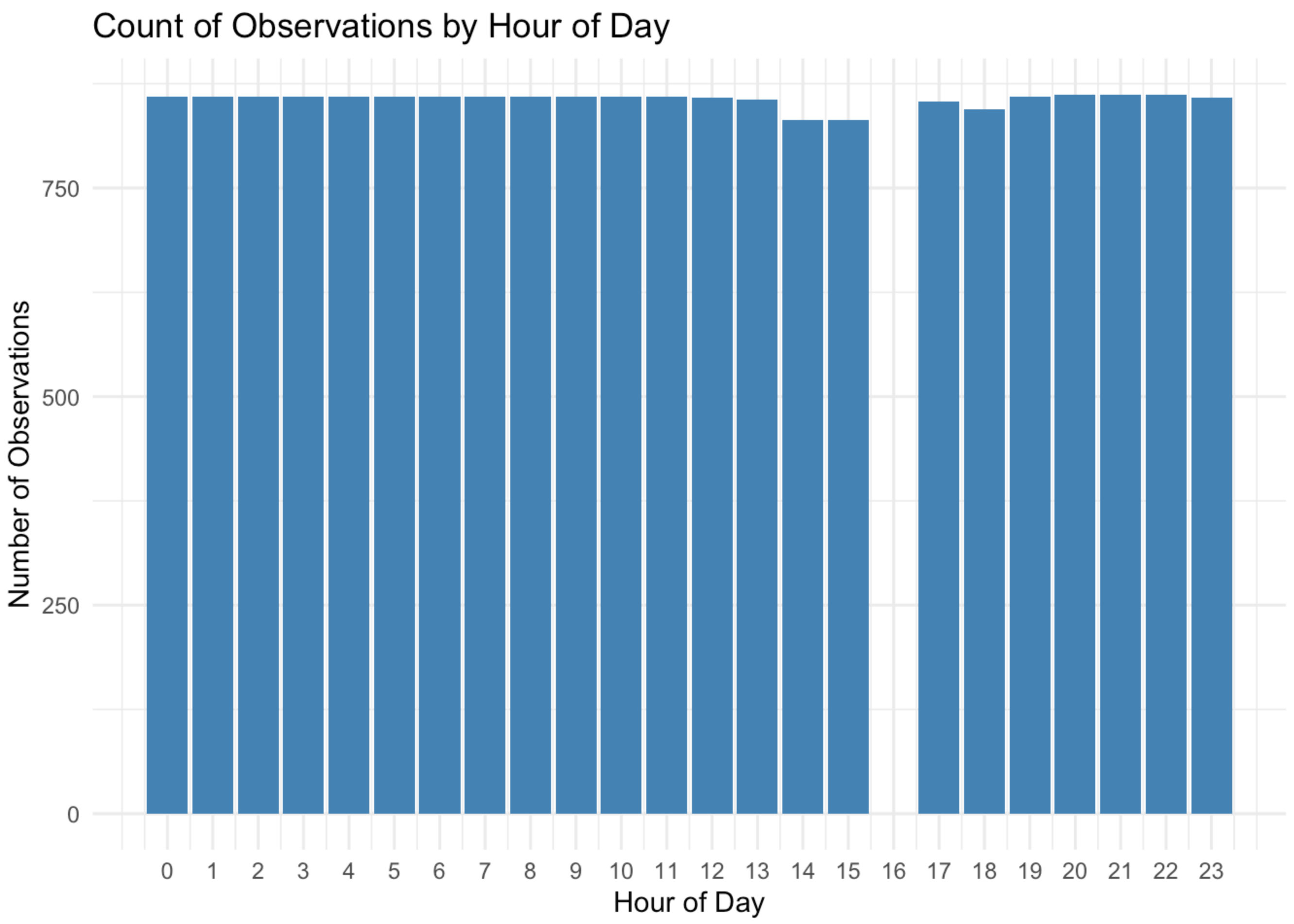
```
summary(hedge_model)
```

```
##
## Call:
## lm(formula = Brent_Last ~ WTI_Last, data = data_wide)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.96168 -0.39912  0.04846  0.45593  1.21913
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.7322728  0.0672961  -10.88   <2e-16 ***
## WTI_Last     1.0588120  0.0009329 1135.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5987 on 8850 degrees of freedom
## Multiple R-squared:  0.9932, Adjusted R-squared:  0.9932
## F-statistic: 1.288e+06 on 1 and 8850 DF,  p-value: < 2.2e-16
```

Based on this data, beta 1 hat is approximately 1.06, so 1.06 barrels of WTI is needed to hedge one barrel of Brent. To calculate a state variable that senses a divergence, I use a one month (720 hours) rolling window to detect extremes. A very short window reacts quickly to news or structural noise which will produce a mean value that conforms to such extremes. In contrast, a month of hourly data will be able to capture cyclical events that influences Brent and WTI pricing, such as multiple inventory releases or announcements. This will be a more reliable window for estimating the spread equilibrium.

Before computing the hedge-adjusted spread, there are hourly no trade windows that needs to be documented. We first visualize to identify any hours that see no trading activity.

```
data_raw %>%
  mutate(Hour = hour(Time)) %>%
  count(Hour) %>%
  ggplot(aes(x = Hour, y = n)) +
  geom_col(fill = "steelblue") +
  scale_x_continuous(breaks = 0:23) +
  labs(
    title = "Count of Observations by Hour of Day",
    x = "Hour of Day",
    y = "Number of Observations"
  ) +
  theme_minimal()
```



The histogram reveals that the raw data already excludes no-trade windows, which are periods where futures markets are closed for maintenance and on Saturdays. The data only contains hours where both WTI and Brent are present, as confirmed by my filtering that requires both benchmarks to be present at each timestamp. The remaining observation counts are stable.

This pre-filtered structure has important implications for the analysis. First, the hedge-adjusted spread calculations use only synchronized, actual market prices rather than stale prices. Second, the proposed 720 hour rolling window therefore represents approximately 5-6 weeks of calendar time rather than 30 days, since each week contributes approximately 137 hours rather than 168 (subtracting Saturdays and 1 hour a day for maintenance). Knowing this, we adjust our 720 hour window to 560 hours instead.

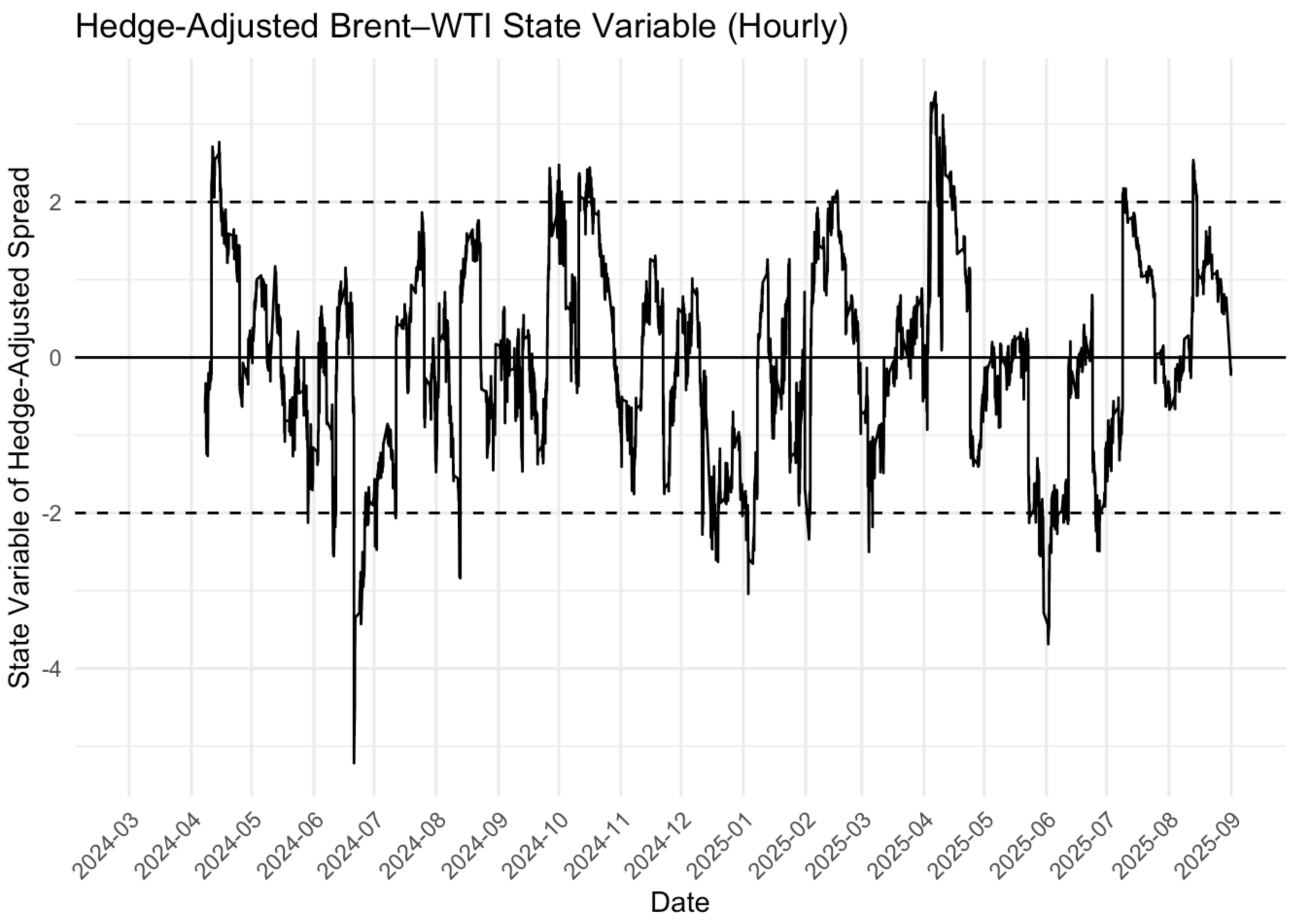
```r
window_hours <- 560 # one month window

state_var <- data_wide %>%
  mutate(
    HAS = Brent_Last - beta_hat * WTI_Last # hedge adjusted spread
  ) %>%
  mutate(
    HAS_Mean = rollapply(HAS, width = window_hours, FUN = mean, fill = NA, align = "r
ight"), # FUN = function
    HAS_SD = rollapply(HAS, width = window_hours, FUN = sd, fill = NA, align = "righ
t")
  ) %>%
  mutate(
    state_var = (HAS - HAS_Mean) / HAS_SD
  )
```

RQ1 Deliverables

One figure that shows your chosen relative measure, a baseline reference, and bands that indicate unusual levels.

```r
ggplot(state_var, aes(x = Time, y = state_var)) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = "solid") + # baseline reference = 0
  geom_hline(yintercept =  2, linetype = "dashed") + # unusual levels as +/- 2 SD
  geom_hline(yintercept = -2, linetype = "dashed") +
  labs(
    title = "Hedge-Adjusted Brent–WTI State Variable (Hourly)",
    x = "Date",
    y = "State Variable of Hedge-Adjusted Spread"
  ) +
  scale_x_datetime(date_breaks = "1 month", date_labels = "%Y-%m") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), # tilt 45 degrees for visi
bility
        panel.grid.minor.x = element_blank())   # remove minor vertical lines
```



A short paragraph that explains your choices and the mean reversion intuition.

The Brent–WTI spread exhibits mean reversion due to the fundamental arbitrage relationship between these benchmarks, where divergences from this relationship creates profitable trade-spread opportunities. When the state variable exceeds +2 standard deviations (Brent unusually higher relative to WTI), traders can short Brent and long WTI, profiting as the spread reverses to the mean. Conversely, when the state variable falls below -2 standard deviations, the opposite trade becomes attractive. The figure shows several clear instances where the spread reaches these extreme levels before reverting toward zero, validating the mean reversion hypothesis is present between WTI and Brent.

RQ2. Can a simple rule on your state variable make money after costs?

```
# Using CME Group Fee Schedule, member pricing = 0.80 for both Brent and WTI futures
contracts + a 0.70 facilitation fee, totaling $1.50 per side of the contract

# every contract (WTI and Brent) is 1000 barrels, so transaction cost per contract wo
uld be $1.50/1000
transaction_cost <- 1.50/1000*2
```

According to the CME Group Fee Schedule, transaction costs includes an exchange fee and a facilitation fee on each trade. The combined cost is 1.50 per contract per side, so 3.00 in total for 1000 barrels. This calculation does not include broker commissions nor bid–ask spreads.

```
# Enter when the state variable breaches an upper or lower band. Exit when it returns
toward the center. Use a one bar signal delay to avoid look ahead.

# signal function that tracks when state_var goes beyond or below customizable thresh
olds
generate_signals <- function(data, entry_threshold, exit_threshold) {
  data %>%
    arrange(Time) %>%
    mutate(
      raw_signal = case_when(
        state_var > entry_threshold ~ -1,  # spread wide: short Brent, long WTI
        state_var < -entry_threshold ~ 1,  # spread narrow: long Brent, short WTI
        abs(state_var) <= exit_threshold ~ 0, # abs of the state variable <= exit_thr
eshold centered around 0
        TRUE ~ NA_real_ # hold previous value
      )
    ) %>%
    mutate(raw_signal = na.locf(raw_signal, na.rm = FALSE)) %>% # locf = last observa
tion carried forward
    mutate(raw_signal = replace_na(raw_signal, 0)) %>% # before any signals appeared
at all, replace with 0
    mutate(
      position = lag(raw_signal, 1, default = 0)  # one-bar delay - assume hourly tra
ding intervals, always places one bar after the signal fires
    )
}
```

```r
# backtest = "evaluating the effectiveness of a trading strategy by running it agains
t historical data" - Buttignol, n.d.

# backtest function with trade log bookkeeping
backtest <- function(data, entry_threshold, exit_threshold = 0) {
  data_with_signals <- generate_signals(data, entry_threshold, exit_threshold) # firs
t run signal function to add raw signals
  data_with_pnl <- data_with_signals %>% # calculate pnl = profit and loss using posi
tion
    mutate(
      position_change = position - lag(position, default = 0),
      spread_change = HAS - lag(HAS),
      gross_pnl_bar = position * spread_change,
      trade_costs_bar = abs(position_change) * transaction_cost,
      net_pnl_bar = gross_pnl_bar - trade_costs_bar
    )

# Record a trade log with timestamps, direction, entry and exit state values, and per
trade PnL.

  trades <- data_with_pnl %>%
    filter(position_change != 0) %>% # filter only for when trades actually happened
    mutate(
      trade_type = case_when(
        position_change > 0 & position == 1 ~ "Entry_Long",
        position_change < 0 & position == -1 ~ "Entry_Short",
        position == 0 ~ "Exit",
        TRUE ~ "Partial"
      )
    )

  # only entry + exit pairs
  trade_log <- trades %>%
    filter(trade_type %in% c("Entry_Long", "Entry_Short", "Exit")) %>%
    mutate(
      trade_id = cumsum(trade_type %in% c("Entry_Long", "Entry_Short"))
    ) %>%
    group_by(trade_id) %>%
    filter(n() == 2) %>%  # round-trip trades only
    summarise(
      entry_time = first(Time),
      exit_time = last(Time),
      direction = first(position),
      entry_state = first(state_var),
      exit_state = last(state_var),
      entry_spread = first(HAS),
      exit_spread = last(HAS),
      holding_hours = as.numeric(difftime(exit_time, entry_time, units = "hours")),
      gross_trade_pnl = if_else(
        first(position) == 1,
        exit_spread - entry_spread, # long spread
        entry_spread - exit_spread # short spread
      ),
      trade_costs = 2 * transaction_cost,
      net_trade_pnl = gross_trade_pnl - trade_costs,
      .groups = "drop"
    )

  # PnL based ONLY on closed trades
  gross_pnl_closed <- if (nrow(trade_log) > 0) {
    sum(trade_log$gross_trade_pnl, na.rm = TRUE)
  } else {
    0
  }

  net_pnl_closed <- if (nrow(trade_log) > 0) {
    sum(trade_log$net_trade_pnl, na.rm = TRUE)
  } else {
    0
  }

  summary_stats <- list(
    entry_threshold = entry_threshold,
    exit_threshold = exit_threshold,
    trade_count = nrow(trade_log),
    win_rate = if (nrow(trade_log) > 0) mean(trade_log$net_trade_pnl > 0) else NA_rea
l_,
    avg_holding_hours = if (nrow(trade_log) > 0) mean(trade_log$holding_hours) else N
A_real_,
    gross_pnl_per_barrel = gross_pnl_closed,
    net_pnl_per_barrel = net_pnl_closed,
    trade_log = trade_log,
    full_data = data_with_pnl
  )

  return(summary_stats)
}
```

RQ2 Deliverables

A compact table for a few threshold choices that reports: trade count, win rate, average holding time in hours, gross PnL per barrel, and net PnL after costs.

```
# test multiple threshold combinations - pseudo-randomly selected
thresholds_grid <- expand.grid(
  entry = c(1.5, 2.0, 2.5),
  exit  = c(0, 0.5)
)

results <- vector("list", nrow(thresholds_grid))

# function backtesting the entry/exit combos
for (i in seq_len(nrow(thresholds_grid))) {
  results[[i]] <- backtest(
    state_var,
    entry_threshold = thresholds_grid$entry[i],
    exit_threshold  = thresholds_grid$exit[i]
  )
}

summary_table <- map_df(results, function(x) {
  tibble(
    `Entry Threshold` = x$entry_threshold,
    `Exit Threshold` = x$exit_threshold,
    `Trade Count` = x$trade_count,
    `Win Rate` = ifelse(is.na(x$win_rate), "NA", paste0(round(x$win_rate * 100, 1),
"%")),
    `Avg Holding (hrs)` = ifelse(is.na(x$avg_holding_hours), NA, round(x$avg_holding_
hours, 0)),
    `Gross PnL/barrel` = paste0("$", round(x$gross_pnl_per_barrel, 3)),
    `Net PnL/barrel` = paste0("$", round(x$net_pnl_per_barrel, 3))
  )
})

knitr::kable(
  summary_table,
  caption = "Trading Strategy Performance Across Different Thresholds",
  align = c('c', 'c', 'c', 'c', 'c', 'r', 'r'),
  digits = 2
)
```

Trading Strategy Performance Across Different Thresholds

Entry Threshold	Exit Threshold	Trade Count	Win Rate	Avg Holding (hrs)	Gross PnL/barrel	Net PnL/barrel
1.5	0.0	0	NA	NA	$0	$0
2.0	0.0	0	NA	NA	$0	$0
2.5	0.0	0	NA	NA	$0	$0
1.5	0.5	21	85.7%	248	$8.574	$8.448

Entry Threshold	Exit Threshold	Trade Count	Win Rate	Avg Holding (hrs)	Gross PnL/barrel	Net PnL/barrel
2.0	0.5	17	88.2%	280	$8.976	$8.874
2.5	0.5	10	100%	319	$6.986	$6.926

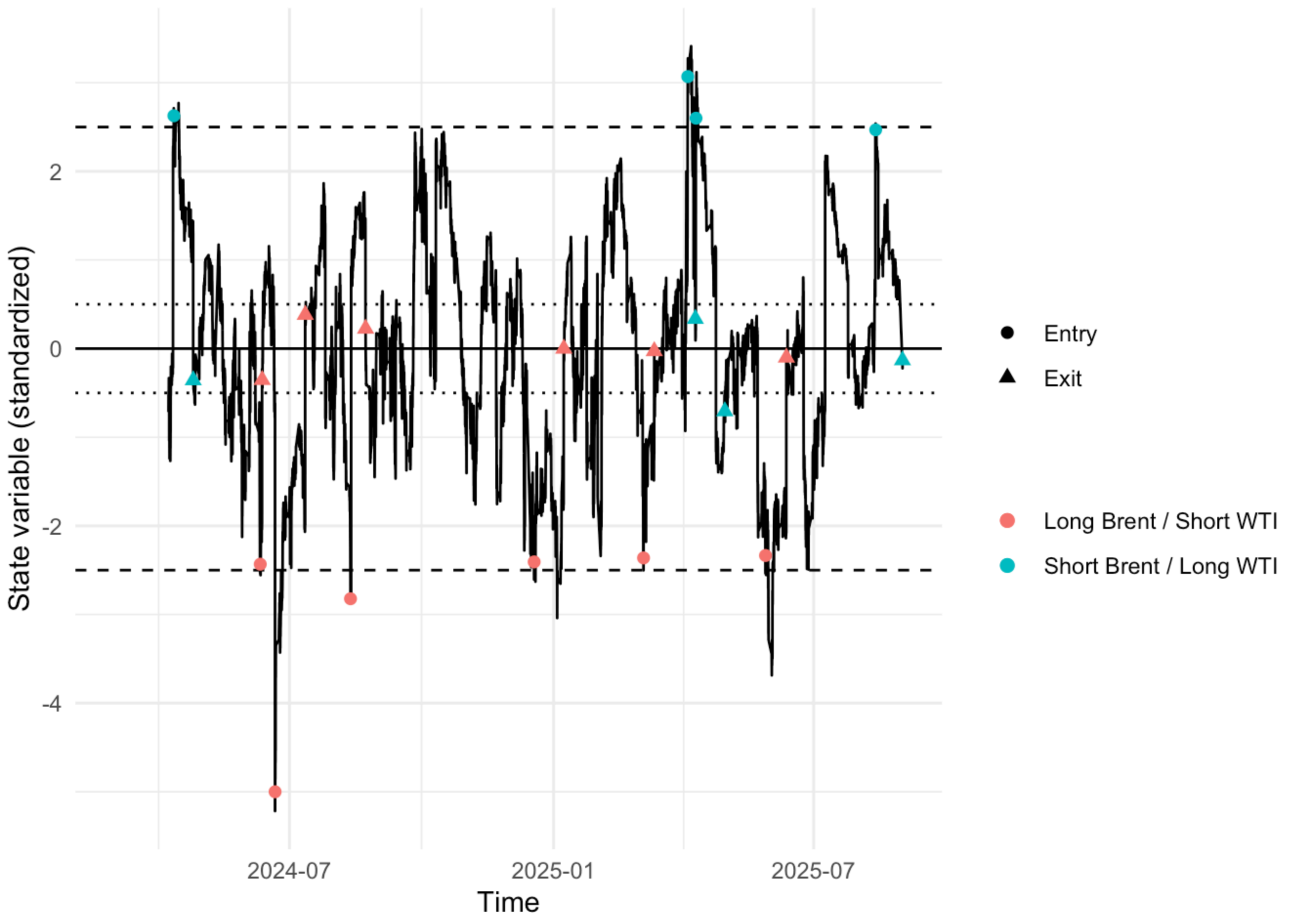
One figure that places entry and exit markers on your state variable so the logic is auditable.

```r
# best win rate
chosen_entry <- 2.5
chosen_exit  <- 0.5

chosen_result <- backtest(
  state_var,
  entry_threshold = chosen_entry,
  exit_threshold = chosen_exit
)

entry_exit_points <- chosen_result$trade_log %>%
  select(entry_time, exit_time, entry_state, exit_state, direction) %>%
  pivot_longer(
    cols = c(entry_time, exit_time),
    names_to = "event",
    values_to = "Time"
  ) %>%
  mutate(
    state_plot = if_else(event == "entry_time", entry_state, exit_state),
    event_type = if_else(event == "entry_time", "Entry", "Exit"),
    direction_label = if_else(
      direction == 1,
      "Long Brent / Short WTI",
      "Short Brent / Long WTI"
    )
  )

ggplot(chosen_result$full_data, aes(x = Time, y = state_var)) +
  geom_line() +
  geom_hline(yintercept = 0) +
  geom_hline(yintercept = c(-chosen_entry, chosen_entry), linetype = "dashed") +
  geom_hline(yintercept = c(-chosen_exit,  chosen_exit),  linetype = "dotted") +
  geom_point(
    data = filter(entry_exit_points, event_type == "Entry"),
    aes(x = Time, y = state_plot, colour = direction_label, shape = event_type),
    size = 2
  ) +
  geom_point(
    data = filter(entry_exit_points, event_type == "Exit"),
    aes(x = Time, y = state_plot, colour = direction_label, shape = event_type),
    size = 2
  ) +
  labs(
    x = "Time",
    y = "State variable (standardized)",
    colour = "",
    shape  = ""
  ) +
  theme_minimal()
```



A short paragraph that names the threshold you would choose and why.

I chose 2.5 entry and 0.5 exit largely due to the matrix that I randomly chose to test threshold pairs. I tested 2.0 and 0.5 as well which was my hypothesized best performer. Arguably, 1.5 and 0.5 is the best performer as it gave the highest return overall, but I chose 2.5 and 0.5 for the 100% win rate which I found interesting.

RQ3 How sensitive are your results to modelling choices? (Path B)

```
# choose a split time
split_time <- "2024-12-31 11:59:59"

state_var <- state_var %>%
  arrange(Time)

train_data <- state_var %>%
  filter(Time <= split_time)

test_data <- state_var %>%
  filter(Time > split_time)
```

```r
# reuse thresholds from rq2 "thresholds_grid"
train_results <- vector("list", nrow(thresholds_grid))

for (i in seq_len(nrow(thresholds_grid))) {
  train_results[[i]] <- backtest(
    train_data,
    entry_threshold = thresholds_grid$entry[i],
    exit_threshold  = thresholds_grid$exit[i]
  )
}

train_table <- thresholds_grid %>%
  mutate(
    net_pnl_per_barrel = purrr::map_dbl(train_results, "net_pnl_per_barrel")
  )

knitr::kable(
  train_table,
  caption = "Training Performance",
  digits = 2
)
```

Training Performance

entry	exit	net_pnl_per_barrel
1.5	0.0	0.00
2.0	0.0	0.00
2.5	0.0	0.00
1.5	0.5	4.57
2.0	0.5	3.91
2.5	0.5	3.42

```r
train_result <- backtest(
  train_data,
  entry_threshold = 1.5,
  exit_threshold  = 0.5
)

test_result <- backtest(
  test_data,
  entry_threshold = 1.5,
  exit_threshold  = 0.5
)
```

RQ3 Deliverables

One small comparison table that shows net PnL after costs, trade count, and win rate for your preferred rule under the baseline and under the robustness path.

```r
rq3_table <- tibble(
  Model = c(
    "Training Segment (Manual Rule)",
    "Test Segment (Manual Rule)"
  ),
  Trade_Count = c(
    train_result$trade_count,
    test_result$trade_count
  ),
  Win_Rate = c(
    round(train_result$win_rate * 100, 1),
    round(test_result$win_rate * 100, 1)
  ),
  Net_PnL = c(
    round(train_result$net_pnl_per_barrel, 3),
    round(test_result$net_pnl_per_barrel, 3)
  )
)

knitr::kable(
  rq3_table,
  caption = "RQ3: Train–Test Robustness",
  digits = 2
)
```

RQ3: Train–Test Robustness

Model	Trade_Count	Win_Rate	Net_PnL
Training Segment (Manual Rule)	11	90.9	4.57
Test Segment (Manual Rule)	10	90.0	4.36

A short paragraph that states whether your conclusion survives the change and why.

My main conclusion survives the train–test split. Unfortunatley, I needed to base my decision on PNL and the training data which is why I chose 1.5 / 0.5 rather than my 100% win rate output. The results are scarily stable - trade count, profit, and win rate are nearly identical. I would say that in real applications, the profits should be much lower after accounting for bid-ask spreads and platform fees. Regardless, very interesting assignment and particularly interesting initial view of statistical trading.