Threshold autoregressive model (TAR) estimation with R

Shota Komatsu

December 01, 2019

I would like to consider how to estimate an econometric model based on the Law of One Price.

Suppose there are two markets and that there is one commodity (e.g. apple). If two markets are linked by trade and arbitrage, those two markets will have a common, unique price, provided there is no transportation cost.

Let $p_{i,t}$ and p_j, t be prices of a commodity at time t in market i and j, respectively, and $c_{ij,t}$ be the transportation cost between market i and j at time t. If market i and j are integrated, then

$$|p_{i,t} - p_{j,t}| \leq c_{ij,t}$$

should hold. This is because if

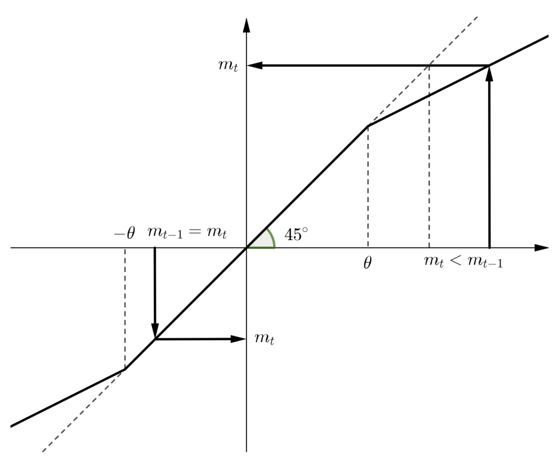
$$|p_{i,t} - p_{j,t}| > c_{ij,t}$$

holds, arbitrageurs have incentives to make arbitrage for profits. But as more and more arbitrages happen, the price spread shrinks so that it equals the transaction cost.

Thus I would like to have a model in which the price adjustment mechanism works only if the abusolute value of the eprice difference in the previous period exceeds the transaction cost. This can be modeled as follows:

$$m_{ij,t} - m_{ij,t-1} = \begin{cases} \varepsilon_{ij,t} & (|m_{ij,t-1}| \leq \theta_{ij,t-1}) \\ \rho m_{ij,t-1} + \varepsilon_{ij,t} & (|m_{ij,t-1}| > \theta_{ij,t-1}) \end{cases},$$

where $m_{ij,t} := p_{ij,t} - p_{ij,t-1}$. A graphical explanation of this model is in the figure below.



There are R libraries that such as `tsDyn`, but the problem is that as far as I check, I cannot estimate the above model with the existing libraries. So it is necessary to write R codes tailored to my own need.

TAR with constant thresholds

First, I write a code that assumes $\theta_t = \theta$ for all t.

Simulating data

We will simulate the data folling the data generating process below:

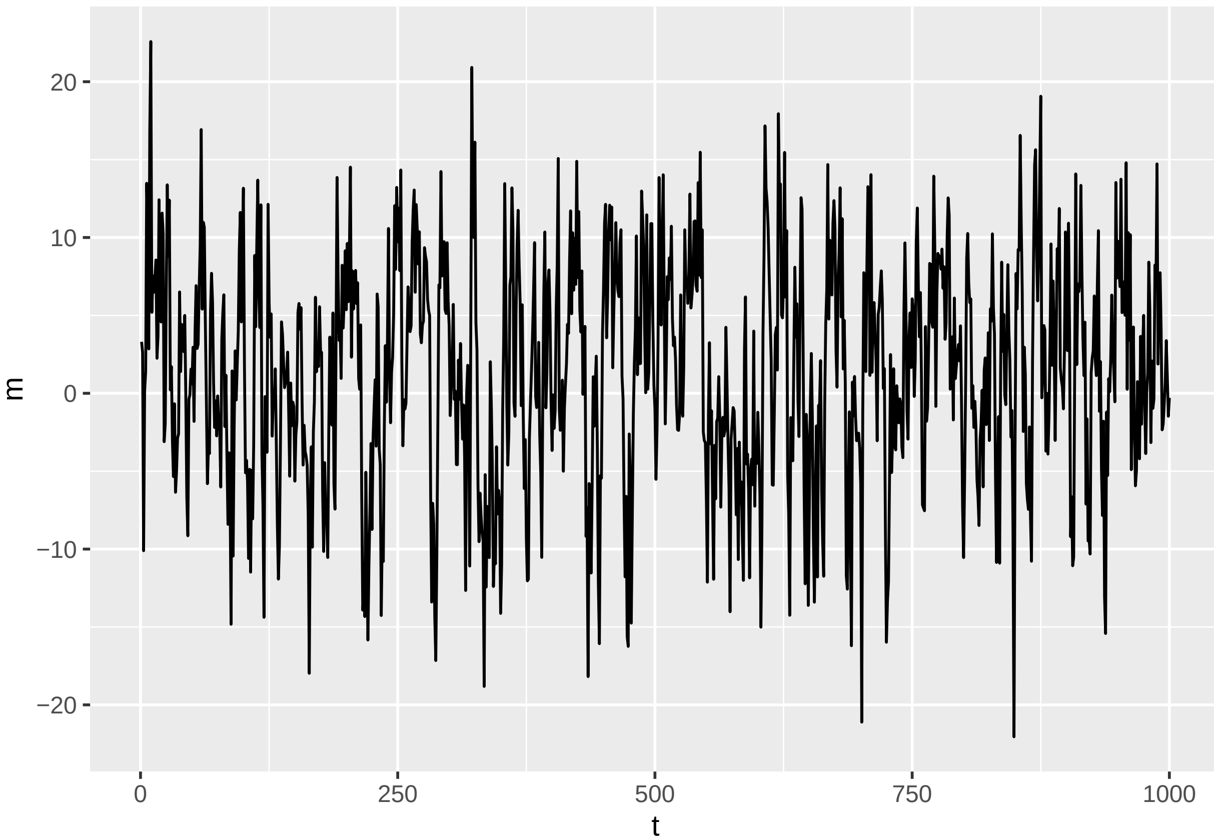
$$\begin{aligned} & m_t - m_{t-1} = \rho m_{t-1} \times 1\{|m_{t-1}| > \theta\} + \varepsilon_t \\ \Leftrightarrow \quad & m_t = (1+\rho) m_{t-1} \times 1\{|m_{t-1}| > \theta\} + \varepsilon_t, \end{aligned}$$

where $m_t := y_t - y_{t-1}$.

```
# Set parameters
# Set seed
set.seed(123)
# Number of observations + 1
n <- 1001
# Draw data from normal distribution
y <- rnorm(n, mean = 0, sd = 10)
z <- rep(0, n)
e <- rnorm(n, mean = 0, sd = 5)
# AR parameter
rho <- -0.5
# Threshold value
theta <- 10
```

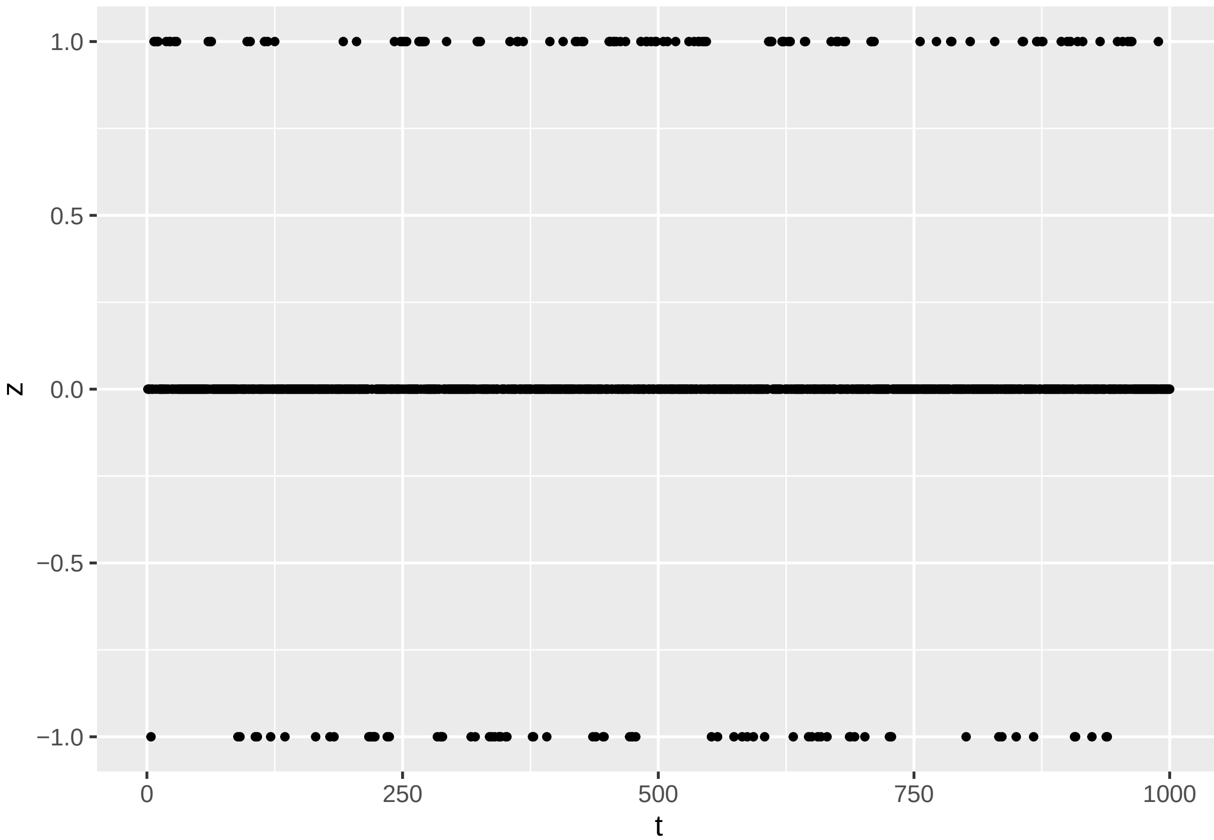
```
# Make dataset
df <- tibble::tibble(y, z, e) %>%
  dplyr::mutate(L.y = dplyr::lag(y, k = 1)) %>%
  dplyr::mutate(m = y - L.y) %>%
  dplyr::mutate(t = 0:(n-1)) %>%
  dplyr::filter(!is.na(m))

for (i in 2:nrow(df)) {
  if (abs(df$m[i-1]) <= theta) {
    df$m[i] <- df$m[i-1] + df$e[i]
    df$z[i] <- 0
  }
  else if (df$m[i-1] < -theta) {
    df$m[i] <- (1 + rho) * df$m[i-1] + df$e[i]
    df$z[i] <- -1
  }
  else {
    df$m[i] <- (1 + rho) * df$m[i-1] + df$e[i]
    df$z[i] <- 1
  }
}
```

```
# Plot the simulated data
ggplot(data = df, aes(x = t, y = m)) +
  geom_line()
```



- $z_t = 1$ if $m_{t-1} > \theta$.
- $z_t = 0$ if $-\theta \leq m_{t-1} \leq \theta$.
- $z_t = -1$ if $m_{t-1} < -\theta$.

```
# Plot to which regime each observation belongs.
ggplot(data = df, aes(x = t, y = z)) +
  geom_point(size = 1)
```



Estimating the parameters

The estimation of TAR model consists of the following steps:

1. The possible candidates for θ are selected from the m_t in the data in such a way that at least 20% of observations are either within or outside the band formed by the thresholds. This means that to assure regime switching, you exclude the candidates which make almost all m_t inside or outside the band.
2. For each candidate θ, create a variable $z_t := 1\{|m_{t-1}| > \theta\}$.
3. Compute $m_{t-1} \times z_t$.
4. Regress $m_t - m_{t-1}$ on $m_{t-1} \times z_t$. Compute the residual sum of squares ($RSS(\theta)$).
5. Select the model that minimizes $RSS(\theta)$.

The absolute value of ρ measures the adjustment speed. In the literature of market integration with the autoregressive approach, it is conventional to report 'halflife' values to show how fast markets respond to arbitrage opportunities. A halflife value h is defined by

$$h := \frac{\ln(0.5)}{\ln(1+\rho)},$$

which measures how much time is needed for a given shock to return to half its initial value. The smaller h is, the faster the adjustment is.

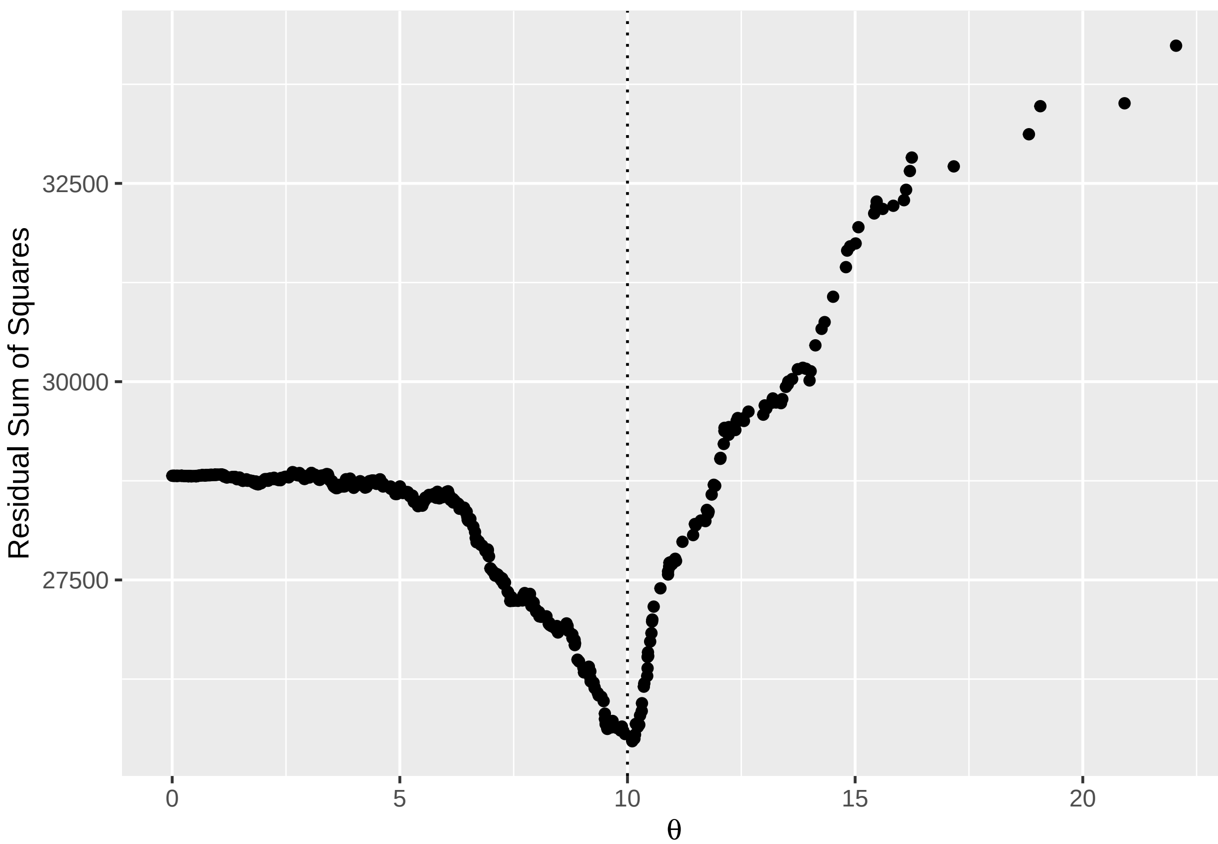
```
# Estimation
result1 <- TAR_const(df$m)
theta1 <- round(result1$theta, digits = 3)
halflife1 <- round(result1$halflife, digits = 3)
```

```r
# Regression table
stargazer::stargazer(
  result1$regression,
  type = "text",
  add.lines = list(c("Threshold Value", theta1), c("Halflife", halflife1))
)
```

```
##
##                           Dependent variable:
##                                   D.y
## i.t                            -0.542***
##                                 (0.029)
## Threshold Value                 10.105
## Halflife                         0.888
## Observations                      999
## R2                               0.263
## Adjusted R2                      0.262
## Residual Std. Error        5.051 (df = 998)
## F Statistic            355.530*** (df = 1; 998)
## Note:                 *p<0.1; **p<0.05; ***p<0.01
```

Check if RSS is minimized around the true parameter.

```r
plot(result1$rssplot + ggplot2::geom_vline(xintercept = theta, linetype = "dotted"))
```



Compare the true and estimated parameters.

```r
# Compare the true and estimated parameters
comparison <-
```

```
  data.frame(
    parameter = c("rho", "theta"),
    true = c(rho, theta),
    estimate = c(as.numeric(result1$regression$coefficients), theta1)
  )
comparison
```

```
##   parameter true   estimate
## 1       rho -0.5 -0.5420088
## 2     theta 10.0 10.1050000
```

TAR with time-varying thresholds

Following Van Campenhout (2007), TAR model allows transaction costs to be time-varying as

$$\theta_{ij,t} = \theta_{ij,0} + (\theta_{ij,T} - \theta_{ij,0}) \frac{t}{T-1}, \quad (t = 0, 1, \cdots, T-1)$$

where T is the total number of observations. Since the direct measurement of transaction costs is unavailable, $\theta_{ij,0}$ and $\theta_{ij,T}$ are determined through a grid search over possible candidates for those thresholds.

Estimating the parameters

The estimation of TAR model consists of the following steps:

1. The possible candidates for θ are selected from the m_t in the data in such a way that at least 20% of observations are either within or outside the band formed by the thresholds. This means that to assure regime switching, you exclude the candidates which make almost all m_t inside or outside the band.
2. Make all possible pairs of θ_0 and θ_T from the threshold candidates.
3. For each pair of threthold candidate θ_0 and θ_T, compute θ_t for each t.
4. Create a variable $z_t := 1\{|m_{t-1}| > \theta_t\}$.
5. Compute $m_{t-1} \times z_t$.
6. Regress $m_t - m_{t-1}$ on $m_{t-1} \times z_t$. Compute the residual sum of squares ($RSS(\theta_0, \theta_T)$).
7. Select the model that minimizes $RSS(\theta_0, \theta_T)$.

It is possible that there are multiple minimizer aroud the global minimum. In that case, the code yields the arithmetic average of them. You should check whether the averaged parameters are concentrated arould the global minimum by plotting RSS.

Simulate data

We will simulate the data folling the data generating process below:

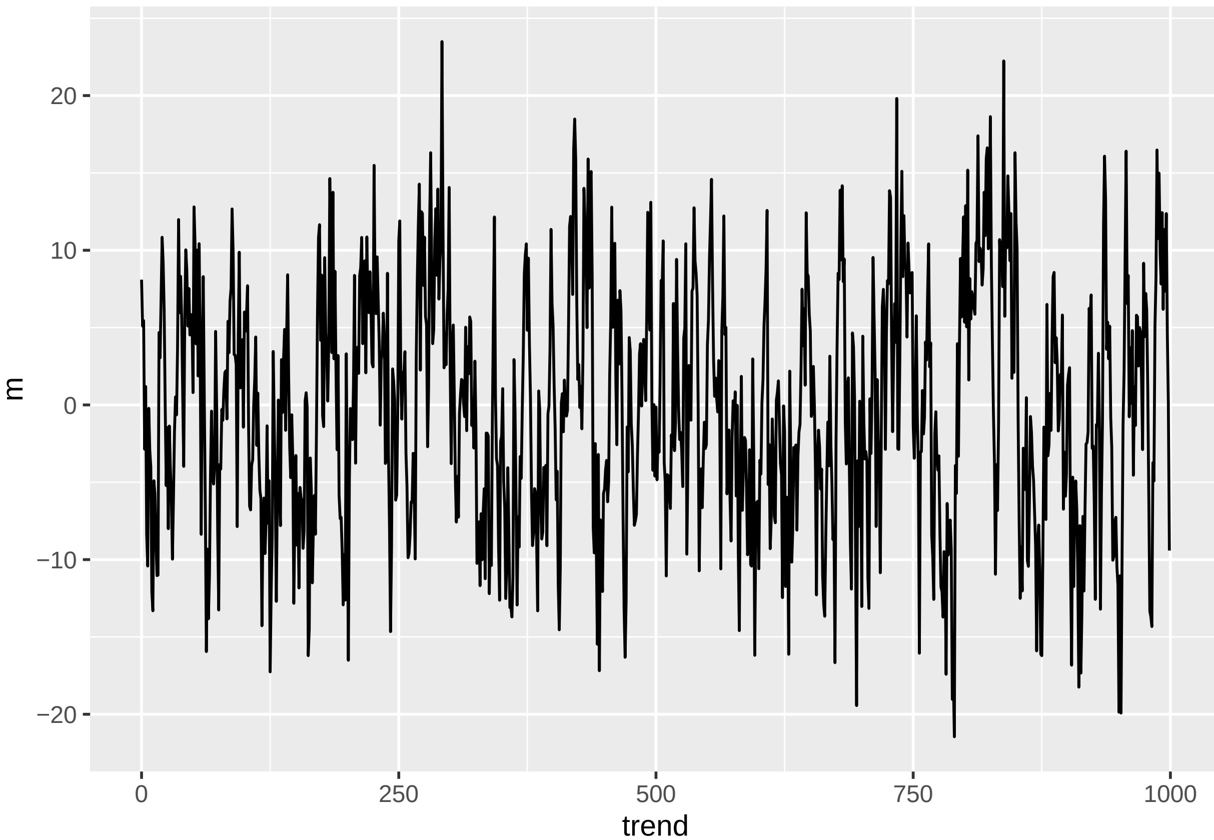
$$\begin{aligned} & m_t - m_{t-1} = \rho m_{t-1} \times 1\{|m_{t-1}| > \theta_t\} + \varepsilon_t \\ \Leftrightarrow \quad & m_t = (1+\rho) m_{t-1} \times 1\{|m_{t-1}| > \theta_t\} + \varepsilon_t, \end{aligned}$$

where $m_t := y_t - y_{t-1}$, $\theta_{ij,t} = \theta_{ij,0} + (\theta_{ij,T} - \theta_{ij,0}) \frac{t}{T-1}, \quad (t = 0, 1, \cdots, T-1)$.

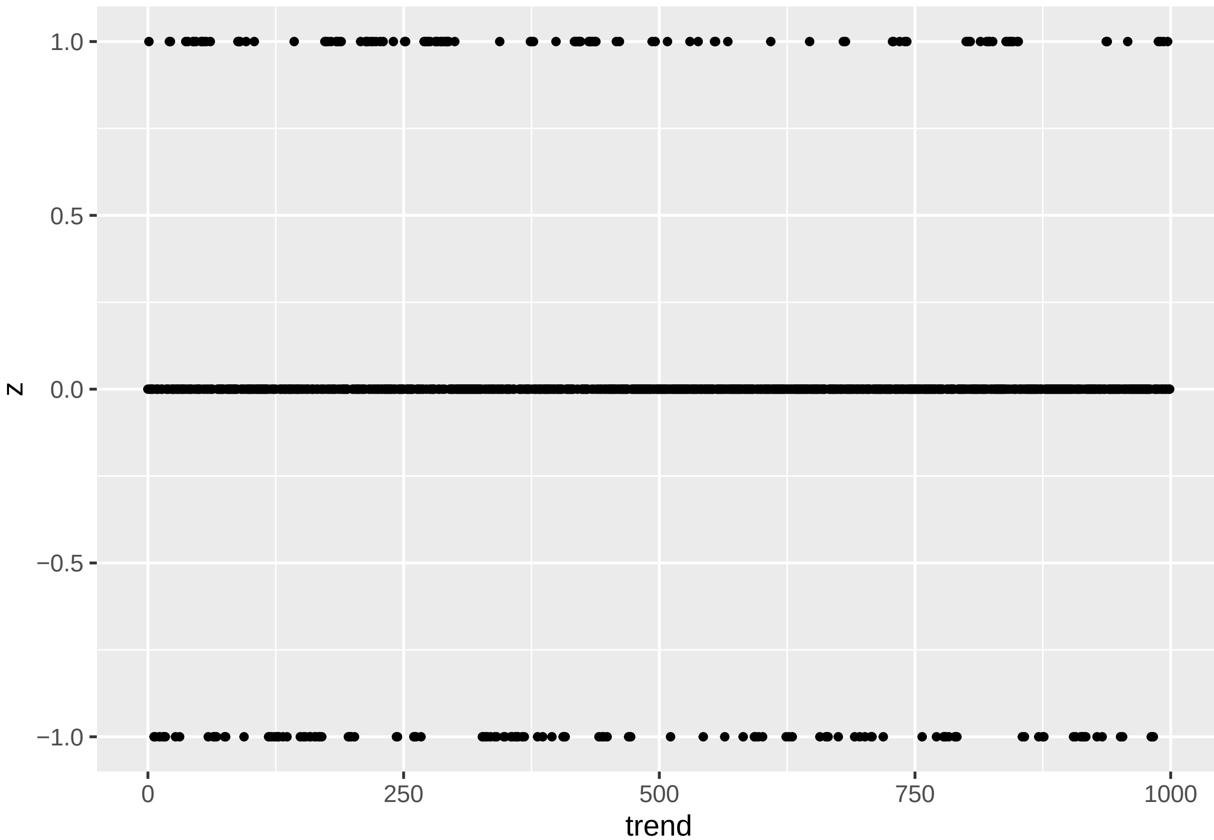
```r
# Set parameters
# Set seed
set.seed(1)
# total number of observations + 1
n <- 1001
# Draw data from normal distribution
y <- rnorm(n, mean = 0, sd = 10)
z <- rep(0, n)
e <- rnorm(n, mean = 0, sd = 5)
# AR parameter
rho <- -0.5
# Threshold value
theta_first <- 7
theta_last <- 12
```

```r
df <- tibble::tibble(y, z, e) %>%
  dplyr::mutate(L.y = dplyr::lag(y, k = 1)) %>%
  dplyr::mutate(m = y - L.y) %>%
  dplyr::mutate(t = 0:(n-1))%>%
  dplyr::filter(!is.na(m))
# Total number of observations
T <- nrow(df)
# Create a variable for time-varying threshold
df <- df %>%
  dplyr::mutate(trend = 0:(T - 1)) %>%
  dplyr::mutate(theta_trend = theta_first + (theta_last - theta_first) * (trend / (T - 1)))

for (i in 2:nrow(df)) {
  if (abs(df$m[i-1]) <= df$theta_trend[i]) {
    df$m[i] <- df$m[i-1] + df$e[i]
    df$z[i] <- 0
  }
  else if (df$m[i-1] < -df$theta_trend[i]) {
    df$m[i] <- (1 + rho) * df$m[i-1] + df$e[i]
    df$z[i] <- -1
  }
  else {
    df$m[i] <- (1 + rho) * df$m[i-1] + df$e[i]
    df$z[i] <- 1
  }
}
```

```r
# Plot the simulated data
ggplot(data = df, aes(x = trend, y = m)) +
  geom_line()
```




```r
# Regime switching
ggplot(data = df, aes(x = trend, y = z)) +
  geom_point(size = 1)
```



It takes time to estimate the parameters because we have to compute RSS for all possble pairs of thresholds. If $T = 1000$, the number of pairs is $(1000 * 0.8)^2 = 640,000$. To speed up computation, it is better to write the code using Rcpp.

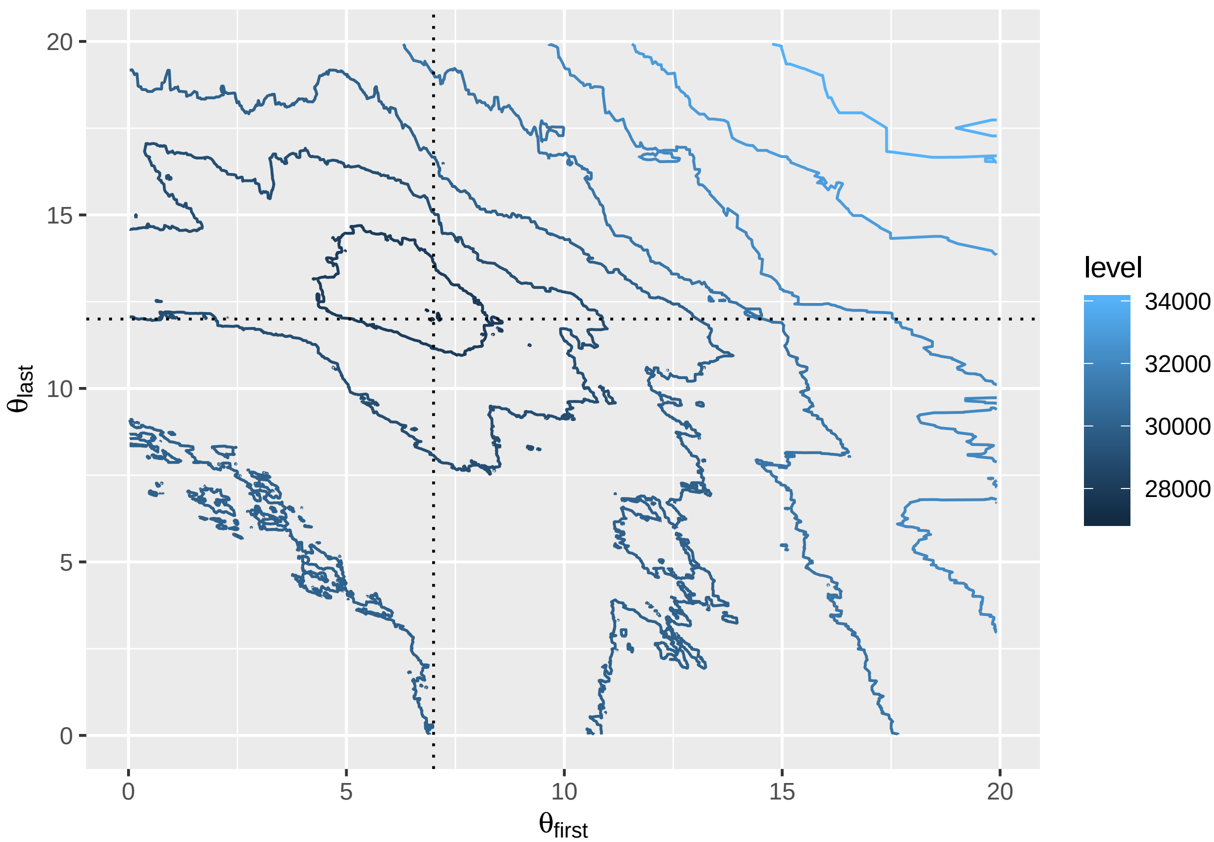
```
# Estimate TAR with time-varying thresholds.
result <- TAR_threshold_varying(df$m)
```

```
# Estimated regression function
reg <- result$regression
# Robust standard error
RobustSE <- as.vector(lmtest::coeftest(reg, vcov = sandwich::vcovHC(reg, "HC1"))[, 2])
# Threshold value
theta_first_estimate <- round(mean(result$theta_first), digits = 3)
theta_last_estimate <- round(mean(result$theta_last), digits = 3)
# Halflife value
halflife <- round(result$halflife, digits = 3)
# Output by stargazer
stargazer::stargazer(
  result$regression,
  se = list(c(RobustSE)),
  type = "text",
  add.lines = list(
    c("Threshold Value (first)", theta_first_estimate),
    c("Threshold value (last)", theta_last_estimate),
    c("Halflife", halflife)
  )
)
```

```
##
##                               Dependent variable:
##                                      D.y
## i.t                                -0.470***
##                                     (0.026)
## Threshold Value (first)              6.967
## Threshold value (last)              11.986
## Halflife                             1.091
## Observations                          999
## R2                                   0.242
## Adjusted R2                          0.241
## Residual Std. Error            5.197 (df = 998)
## F Statistic               317.967*** (df = 1; 998)
## Note:                    *p<0.1; **p<0.05; ***p<0.01
```

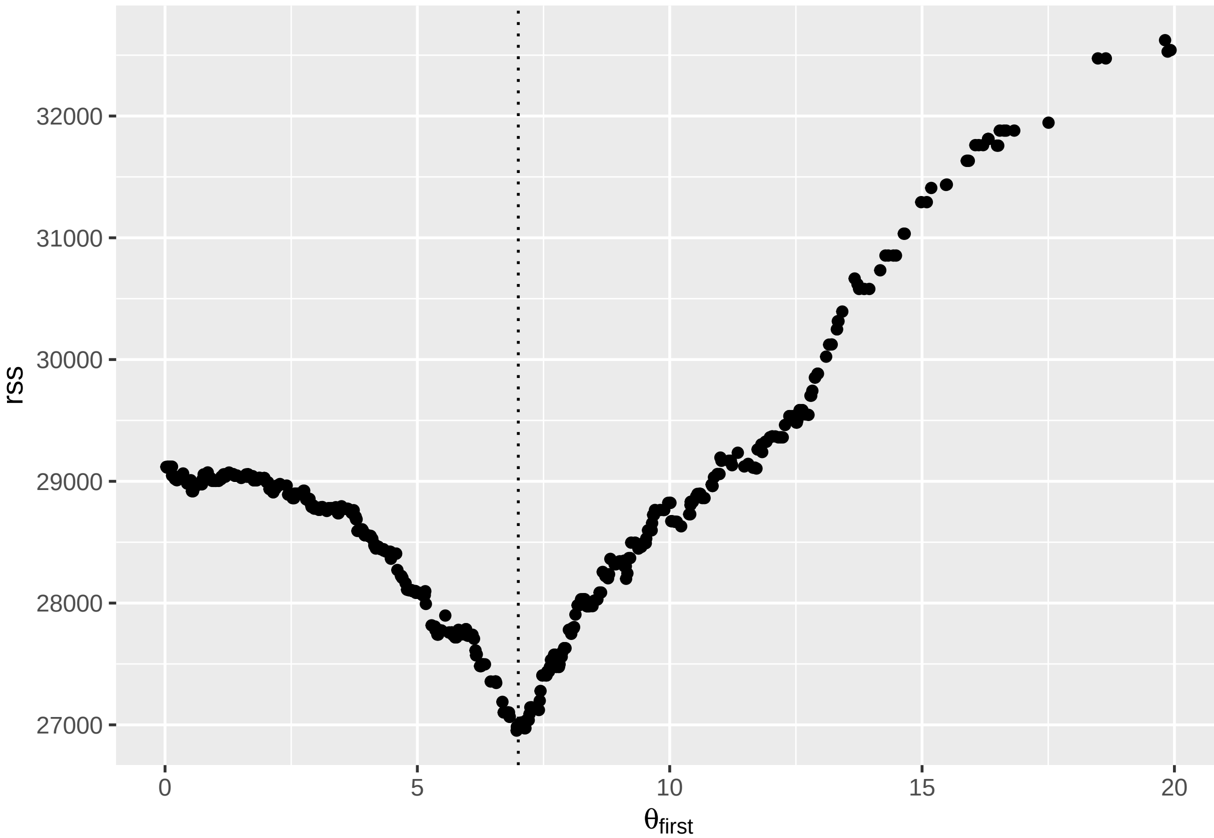
Check if RSS is minimized around the true parameter.

```r
# Contour plot
g_contour <- result$plot_contour +
  geom_vline(xintercept = theta_first, linetype = "dotted") +
  geom_hline(yintercept = theta_last, linetype = "dotted") +
  xlab(latex2exp::TeX("$\\theta_{first}$")) +
  ylab(latex2exp::TeX("$\\theta_{last}$")) +
  scale_linetype_discrete(name = "Residual sum of squares")
plot(g_contour)
```



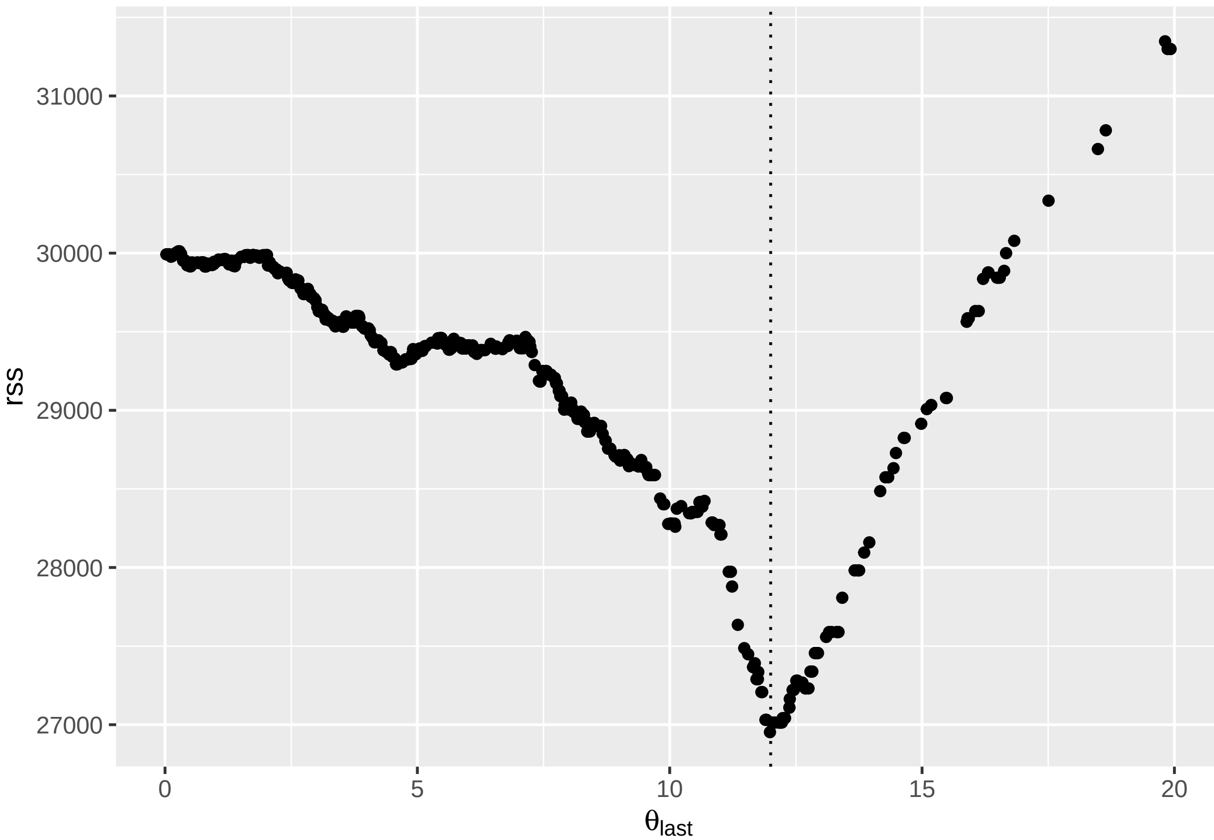
Check if θ_{first} minimizes RSS given the estimated θ_{last}.

```r
g_theta_first <- result$plot_theta_first +
  geom_vline(xintercept = theta_first, linetype = "dotted") +
  xlab(latex2exp::TeX("$\\theta_{first}$"))
plot(g_theta_first)
```



Check if θ_{last} minimizes RSS given the estimated θ_{first}.

```r
g_theta_last <- result$plot_theta_last +
  geom_vline(xintercept = theta_last, linetype = "dotted") +
  xlab(latex2exp::TeX("$\\theta_{last}$"))
plot(g_theta_last)
```




```r
# Compare the true and estimated parameters
comparison <-
  data.frame(
    parameter = c("rho", "theta_first", "theta_last"),
    true = c(rho, theta_first, theta_last),
    estimate = c(as.numeric(result$regression$coefficients), theta_first_estimate, theta_last_estimate)
  )
comparison
```

```
##     parameter true   estimate
## 1         rho -0.5 -0.4700982
## 2 theta_first  7.0  6.9670000
## 3  theta_last 12.0 11.9860000
```

References

Van Campenhout, Bjorn. 2007. "Modeling Trends in Food Market Integration: Method and an Application to Tanzanian Maize Markets." *Food Policy* 32 (1): 112–27.