Usage

In this section, for simplicity, we detail how to use the functions in **DPTM** by explaining the augments and running through an example with simulated data. The package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=DPTM` and can be installed and loaded in the usual manner[3]:

```
R> install.packages("DPTM")
R> library("DPTM")
```

3.1. Estimation: DPTS and DPML

DPTM contains the dynamic panel multiple threshold model with fixed effects, `DPTS`, and the dynamic panel linear model with effects, `DPML`. The function `DPTS` has more arguments than `DPML` because the dynamic panel multiple threshold model with fixed effects can degenerate to the linear regression model if there are no threshold effects. As a result, we introduce `DPTS` first.

There are 25 different augments are shown in Table 1. Except for unuseful `delty0` and print option `display`, others can mainly be divided into three different augment sets. Ranking by importance, they are the I-type augment set, II-type augment set, and III-type augment set, respectively. The I-type augment set means the augments in this set not only determine the specification of the model used but also guarantee the running of `DPTS` and thus is necessary. In other words, the lack of any augment in the I-type augment set will cause an error. The I-type augment set involves the dependent variable `y`, the threshold variable `q`, the length of period `tt`, and the number of individuals `nn`.

The II-type augment set's augments are optional and only influence the model specification researchers decide on in empirical studies. This set involves the number of thresholds `Th`, the lag dependent variable `y_1`, the independent variables with threshold effects `x`, the independent variables without threshold effects `cvs`, the period term `time_trend`, the time fixed effects `time_fix_effects` and the option of whether there are threshold effects in y_1 `Noy`. It is worth noting that `time_trend` and `time_fix_effects` can not be `TRUE` simultaneously,

[3]The development version of the package is located on GitHub (https://github.com/HujieBai/DPTM).

Argument	Type	Description
`y`	I-type	the dependent variable.
`q`	I-type	the threshold variable.
`tt`	I-type	the length of time period.
`nn`	I-type	the number of individuals.
`Th`	II-type	the number of thresholds.
`y1`	II-type	the lag dependent variable.
`x`	II-type	the independent variable.
`NoY`	II-type	the option of threshold effects on the lag dependent variable.
`cvs`	II-type	the set of control variables.
`time_trend`	II-type	the time_trend.
`time_fix_effects`	II-type	the time fixed effects.
`x1`	III-type	the initial values of independent variable.
`Only_b`	III-type	the option of initial equation.
`w`	III-type	the variance ratio.
`var_u`	III-type	the option of variance of error term.
`sro`	III-type	the least ratio of sample in regimes.
`r0x`	III-type	the lower bound of thresholds.
`r1x`	III-type	the upper bound of thresholds.
`restart`	III-type	the option of iterations.
`burnin`	III-type	the length of burn-in.
`ms`	III-type	the length of MCMC chains after burn-in.
`types`	III-type	the type of MCMC used; More details see BayesianTools::runMCMC.
`ADs`	III-type	the options for MCMC; More details see BayesianTools::runMCMC.
`nCR`	III-type	parameter determining the number of cross-over proposals of DREAM MCMC.
`autoburnin`	III-type	a logical flag indicating of the Gelman and Rubin's convergence diagnostic.
`delty0`	other	the option of Δy_{it}.
`display`	other	the option of whether to print the messages of estimated results or not.

Table 1: The description of `DPTS`'s arguments

and **DPTM** allows no threshold effects in the lag of dependent variable y_1 by setting `Noy` as `TRUE`. Table 2 shows some combinations of the above augments containing the I-type augment set. There is only one threshold for simplicity; people should choose these augments according to their specific model.

Number	x	cvs	time_trend	time_fix_effects	Noy	model
1	×	×	×	×	×	$y_{it} = \beta_1 y_{it-1} I(q_{it} \leq \gamma_1) + \beta_2 y_{it-1} I(\gamma_1 < q_{it}) + \alpha_i + \varepsilon_{it}$
2	√	×	×	×	×	$y_{it} = (\beta_1 y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it}) I(q_{it} \leq \gamma_1) + (\beta_2 y_{it-1} + \boldsymbol{\theta}_2' \mathbf{x}_{it}) I(\gamma_1 < q_{it}) + \alpha_i + \varepsilon_{it}$
3	√	√	×	×	×	$y_{it} = (\beta_1 y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it}) I(q_{it} \leq \gamma_1) + (\beta_2 y_{it-1} + \boldsymbol{\theta}_2' \mathbf{x}_{it}) I(\gamma_1 < q_{it}) + \boldsymbol{\psi}' \mathbf{z}_{it} + \alpha_i + \varepsilon_{it}$
4	√	√	√	×	×	$y_{it} = (\beta_1 y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it}) I(q_{it} \leq \gamma_1) + (\beta_2 y_{it-1} + \boldsymbol{\theta}_2' \mathbf{x}_{it}) I(\gamma_1 < q_{it}) + \boldsymbol{\psi}' \mathbf{z}_{it} + \alpha_i + c_t + \varepsilon_{it}$
5	√	√	×	√	×	$y_{it} = (\beta_1 y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it}) I(q_{it} \leq \gamma_1) + (\beta_2 y_{it-1} + \boldsymbol{\theta}_2' \mathbf{x}_{it}) I(\gamma_1 < q_{it}) + \boldsymbol{\psi}' \mathbf{z}_{it} + \alpha_i + t + \varepsilon_{it}$
6	√	√	×	×	√	$y_{it} = \beta y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it} I(q_{it} \leq \gamma_1) + \boldsymbol{\theta}_2' \mathbf{x}_{it} I(\gamma_1 < q_{it}) + \boldsymbol{\psi}' \mathbf{z}_{it} + \alpha_i + \varepsilon_{it}$

Table 2: Some different specifications of dynamic panel multiple threshold model

The III-type augment set is related to the assumption of the MLE method and the choice of MCMC, which does not change model specification and is designed for robustness. Specifically, {`x1`, `Only_b`, `w`, `var_u`, `sro`, `r0x`, `r1x`, `restart`} are the augments for the assumptions of MLE discussed in Section 2 while {`burnin`, `ms`, `types`, `ADs`, `nCR`, `autoburnin`} are the setting of MCMC used. Although the model works well by default, unexpected situations are inevitable because of the diversity of data used by researchers (some critical assumptions may be violated; more details can be found in Hsiao *et al.* (2002) and Ramírez-Rondán (2020)), and thus we hope **DPTM** can still work well by changing these augments. Therefore, we next explain these augments in detail.

`x1` is the $\Delta \mathbf{x}_{i1}$ in (5) while if `Only_b` is `TRUE` equal (5) will degenerate to $\Delta y_{i1} = \delta_0 + v_{i1}$. `w` and `var_u` are the value of $\hat{\omega}(\boldsymbol{\gamma})$ in (8) and $\hat{\sigma}_u^2(\boldsymbol{\gamma})$ in (9), respectively[4]. The incorrect setting of the above augments will cause MLE's iteration procedure to fail, so these augments are very important. If users do not know the specific reason for failure, we supply the `restart` to adjust `w` and `var_u` automatically. In addition, `sro` is the regime sample portion η while `r0x` and `r1x` are the lower bound and upper bound of the parameter space of threshold parameters $\boldsymbol{\Gamma}$, respectively. For MCMC, `burnin` and `ms` are the length of burning and the length of MCMC chains after burning, respectively, which have an essential influence on the convergence of the MCMC samples. As for `types`, `ADs`, `nCR` and `autoburnin`, they determine the kind of MCMC, and more details can be found in **BayesianTools** (Hartig *et al.* 2023).

Then, we give an example of `DPTS` by running the model 3 in Table 2 with simulated data, $y_{it} = (\beta_1 y_{it-1} + \boldsymbol{\theta}_1' \mathbf{x}_{it}) I(q_{it} \leq \gamma_1) + (\beta_2 y_{it-1} + \boldsymbol{\theta}_2' \mathbf{x}_{it}) I(\gamma_1 < q_{it}) + \boldsymbol{\psi}' \mathbf{z}_{it} + \alpha_i + \varepsilon_{it}$. The simulated data we used has been packed into **DPTM**.

```
R> data("data", package = "DPTM")
R> y <- data$data_test$y
R> q <- data$data_test$q
R> x <- as.matrix(data$data_test$x)
R> z <- as.matrix(data$data_test$z)
R> tt <- data$data_test$tt
R> nn <- data$data_test$nn
```

We need to set a random seed to guarantee the reproducibility by `set.seed`, and we set `Th` as 1 to make `DPTS` run a dynamic panel single threshold model.

[4]`w` must be greater than $\frac{T}{T-1}$ and `var_u` must be greater than 0.

```
R> set.seed(2024)
R> m1 <- DPTS(y=y,q=q,x=x,cvs = z,tt=tt,nn=nn,Th=1,ms = 1000,burnin = 1000)

This is a dynamic panel threshold model with fixed effects!
---------------------------------------------------
Time Fixed Effects: FALSE!
---------------------------------------------------
Time Shifts: FALSE!
---------------------------------------------------
The number of threshold is 1!
---------------------------------------------------
The estimates of thresholds:
        Th_1
       0.217
Their confidence intervals are :
    Lower Upper
90% 0.192 0.255
95% 0.179 0.296
99% 0.168 0.298
---------------------------------------------------
The coefs are:
            Estimate Std. Error Z-value   Pr(>|z|) Significance
y_regime_1  "-0.683" "0.05"     "-13.568" "0"      "***"
y_regime_2  "-0.246" "0.06"     "-4.096"  "0"      "***"
x1_regime_1 "2.104"  "0.553"    "3.808"   "0"      "***"
x1_regime_2 "0.665"  "0.641"    "1.037"   "0.3"    ""
Control_1   "2.203"  "0.179"    "12.312"  "0"      "***"
---------------------------------------------------
The Gelman and Rubin Convergence Diagnostic is
Potential scale reduction factors:
        Point est.  Upper C.I.
[1,]       1.05       1.15
If the Upper C.I. are not close to 1, please set a longer burn-in or ms!
If there is any Inf or NaN, please set a longer burn-in or ms, or set nCR
as 1 !
```

Ths and Ths_IC are estimate and confidence intervals for parameters.

```
R> print(m1$Ths)
R> print(m1$Ths_IC)

        Th_1
       0.217
    Lower Upper
90% 0.192 0.255
95% 0.179 0.296
99% 0.168 0.298
```

We can find that the estimated threshold is 0.217 and has fallen into its confidence intervals, which have also been shown. DPTS also prints the estimate, standard error, Z-value, p-value, and significance of coefficients. More details can be found in the value Coefs of DPTS.

```
R> print(print(m1$Coefs))
```

```
             Estimate Std. Error Z-value   Pr(>|z|) Significance
y_regime_1  "-0.683" "0.05"     "-13.568" "0"      "***"
y_regime_2  "-0.246" "0.06"     "-4.096"  "0"      "***"
x1_regime_1 "2.104"  "0.553"    "3.808"   "0"      "***"
x1_regime_2 "0.665"  "0.641"    "1.037"   "0.3"    ""
Control_1   "2.203"  "0.179"    "12.312"  "0"      "***"
```

As for ms and burnin, we set them as 1000 here, and the upper limit of the Gelman and Rubin Convergence Diagnostic by MCMC_Convergence_Diagnostic is 1.15, which is very close to 1 and thus guarantee the convergence of MCMC samples. We also supply the trace plot of MCMC chains by plot(m1$MCMC) for robustness. **DPTM** allows users to check the convergence of their MCMC samples by the above two methods.

```
R> print(m1$MCMC_Convergence_Diagnostic)
R> plot(m1$MCMC)
```

```
The Gelman and Rubin Convergence Diagnostic is
Potential scale reduction factors:
        Point est.  Upper C.I.
[1,]       1.05       1.15
If the Upper C.I. are not close to 1, please set a longer burn-in or ms!
If there is any Inf or NaN, please set a longer burn-in or ms, or set nCR
as 1 !
```

In addition, **DPTM** also supplies the DMPL for the dynamic panel linear model with fixed effects to compare the comparison with the threshold model in an empirical study. The augments of DMPL are contained in DPTS; thus, there is no more explanation. The example of $y_{it} = \beta y_{it-1} + \boldsymbol{\psi}'\mathbf{z}_{it} + \alpha_i + \varepsilon_{it}$ is

```
R> m0 <- DPML(y=y,x=z,tt=tt,nn=nn)
```

```
This is a dynamic panel linear model with fixed effects
----------------------------------------------------
Time Fixed Effects:  FALSE!
----------------------------------------------------
Time Shifts:  FALSE!
----------------------------------------------------
The coefs are:
    Estimate Std. Error Z-value  Pr(>|z|) Significance
y1  "-0.466" "0.056"    "-8.342" "0"      "***"
x_2 "2.301"  "0.228"    "10.093" "0"      "***"
```

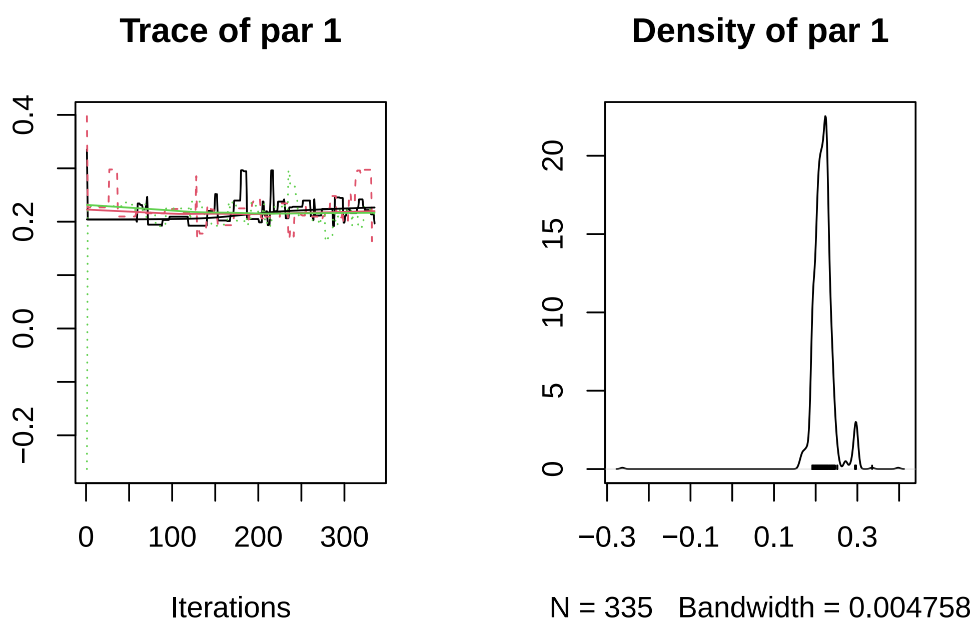

Figure 1. The trace plot of MCMC chains of `m1`

3.2. Inference: `Threshold_Test`

In empirical analysis, if we want to estimate a threshold model, we must first confirm that the threshold effects exist (i.e., there is a nonlinear relationship rather than a linearity). We mentioned the test of threshold effects in (13). Similarly, once we confirm the existence of the threshold effects, we need to verify the number of thresholds so that we can use the specific dynamic panel multiple threshold model, which tests the number of thresholds in (14).

DPTM packs the two different test into the function `Threshold_Test`, and users can change `Th` to choose the specific test. Specifically, if `Th` is, 0 `Threshold_Test` is the threshold existence test while when setting `Th` as a positive integer `Threshold_Test` is the test of whether the number of thresholds is `Th` or not. `Threshold_Test` involves all augments in Table 1, and, because of the bootstrap process, `Threshold_Test` has an extra augment `bt` to set the number of bootstraps.

Firstly, we test whether there are threshold effects in the simulated data. Similarly, we set a random seed to guarantee the reproducibility by `set.seed`. To save time, we set `bt` as 10.

```
R> set.seed(2024)
R> F0 <- Threshold_Test(y=y,x=x,q=q,cvs=z,tt=tt,nn=nn,Th=0,ms = 1000,burnin=1000,
bt=10)

Test for the number of Thresholds
It is noted that when under H0 the number of Thresholds is 0,
this test is the so called threshold existence test.
---------------------------------------------------
H0: There are  0  thresholds
```

```
H1: There are  1  thresholds
---------------------------------------------------
Bootstrap:
Parallel:  FALSE
 1 / 10 No
 2 / 10 No
 3 / 10 No
 4 / 10 No
 5 / 10 No
 6 / 10 No
 7 / 10 No
 8 / 10 No
 9 / 10 No
10 / 10 No
P-value =  0
```

By default, `Threshold_Test` will report whether $F_1^\star$ ($F_s^\star$) in a single bootstrap exceeds F_1 (F_s) or not so users can know whether the null hypothesis should be rejected or not, as soon as possible [5]. Users can also run `Threshold_Test` in parallel by setting `parallel` as `TRUE` to save time, but it will not print the single bootstrap. And we set `bt` as 100.

```
R> set.seed(2024)
R> F0 <- Threshold_Test(y=y,x=x,q=q,cvs=z,tt=tt,nn=nn,Th=0,ms = 1000,burnin=1000,
bt=100,parallel=TRUE)
```

```
Test for the number of Thresholds
It is noted that when under H0 the number of Thresholds is 0, this
test is the so called threshold existence test.
 ---------------------------------------------------
 H0: There are  0  threshold
 H1: There are  1  threshold
 ---------------------------------------------------
 Bootstrap:
 Parallel:  TRUE
  |====================================================================| 100%
P-value =  0
```

The p-value is shown in the value `ps` of `Threshold_Test`, which is 0 and thus rejects the null hypothesis of no threshold effects.

```
R> print(F0$ps)

0.00
```

[5] As shown in Algorithm B, p-value is decided by the frequency of $F_1^\star(B)$ ($F_s^\star(B)$) in bootstrap exceeds F_1 (F_s). So, `Threshold_Test` can allow users to know the results of tests even if the tests are still running. For example, when the number of bootstraps is 100, and once the number of $F_1^\star$ ($F_s^\star$) in a single bootstrap exceeds F_1 (F_s) is more than 5, there will be no reason that the $p-$value will be less than 0.05.

Next, we need to make sure the number of thresholds. By setting `Th` as 1, `Threshold_Test` can test whether or not the number of thresholds is one.

```
R> set.seed(2024)
R> Fs_1 <- Threshold_Test(y=y,x=x,q=q,cvs=z,tt=tt,nn=nn,Th=1,ms = 1000,
burnin=1000, bt=100,parallel=TRUE,display = FALSE)
R> print(Fs_1$ps)
```

```
0.40
```

Here, we cancel the print of estimated results by setting `display` as `FALSE`. The p-value is 0.4; thus, we can not reject the null hypothesis of the single threshold. The result corresponds to the simulated data based on the dynamic panel single threshold model. The above test can continue to ensure the specific threshold number by setting bigger `Th`, but we stop testing it for simplicity.