

# PanelView in STATA: Visualizing Panel Data

The `panelview` package has two main functionalities: (1) it visualizes the treatment and missing-value statuses of each observation in a panel/time-series-cross-sectional (TSCS) dataset; and (2) it plots the variables of interest (either continuous or discrete) in a time-series fashion.

We develop this package in the belief that it is always a good idea to understand your raw data better before conducting statistical analyses.

---

**Date:** November 07, 2021

**Version:** 0.1 ([Github](#))

**Authors:** Hongyu Mou, Yiqing Xu

Please report bugs to [muhongyu@pku.edu.cn](mailto:muhongyu@pku.edu.cn) or [yiqingxu@stanford.edu](mailto:yiqingxu@stanford.edu).

We thank Yifan Huang for the initial coding and Ziyi Liu for the kind suggestion.

---

## Table of Contents

### PanelView in STATA: Visualizing Panel Data

- 0. Installation
  - Installation from GitHub
  - Manual Installation
- 1. Syntax
- 2. Plotting Treatment Conditions
  - 2.1 Two treatment conditions
  - 2.2 Treatment: missing & switch on and off
  - 2.3 Plotting a subset of units
- 3. Ignoring Treatment Conditions
  - 3.1 `ignoretreat` subcommand
  - 3.2 Treatment level = 1 & Plotting treatment
  - 3.3 Treatment level = 1 & Plotting outcome
  - 3.4 Plotting outcome & Continuous treatment / More than two treatment levels
    - 3.4.1 Continuous outcomes
    - 3.4.2 Discrete outcomes
- 4. More Than Two Treatment Conditions
  - 4.1 Treatment level = 3
  - 4.2 Treatment level = 4
  - 4.3 Treatment level  $\geq 5$
  - 4.4 Continuous treatment
- 5. Continuous Outcomes
  - 5.1 Continuous outcomes
  - 5.2 Specify which unit(s) we want to take a look at
  - 5.3 Put each unit into different groups, then plot respectively
- 6. Discrete Outcomes
  - 6.1 Discrete outcomes

- 
- 6.2 Put each unit into different groups, then plot respectively
  - 7. Plotting Any Variable In Panel Dataset
  - 8. Plotting Y And D Time Series In One Graph
    - 8.1 Plot average time series for all units
    - 8.2 Plot by each unit
- 

## 0. Installation

We provide 2 methods for the installation of `panelView`:

### Installation from GitHub

As a pre-requisite, the `github` package needs to be installed:

```
net install github, from("https://haghish.github.io/github/")
```

Then install the `panelView` package with Stata v15 or greater:

```
cap ado uninstall panelView //in-case already installed  
github install xuyiqing/panelview_stata
```

For the `sysuse` command to access example datasets of `panelView`, users should download the datasets (`capacity.dta`, `simdata.dta`, `turnout.dta`) and place them in the directories searched by Stata from `adopath`, as Stata's download commands would not do that automatically.

### Manual Installation

Manual installation takes three simple steps and will load the example datasets without additional work:

1. Download the zip file from: [https://github.com/xuyiqing/panelView\\_stata](https://github.com/xuyiqing/panelView_stata)
2. Unzip the file
3. Type the following commands in your STATA console:

```
cap ado uninstall panelView //in-case already installed  
net install panelView, all replace from(your full local path)
```

---

## 1. Syntax

The general syntax of the package can be summarized as:

```
panelView Y D X [if] [in],           ///  
i(varname) t(varname numeric)       ///
```

```
type(string)           ///
[                   ///
continoustreat       ///
discreteoutcome      ///
bytiming             ///
MYCOLor(string)      ///
PREpost              ///
xlabdist(integer 1)  ///
ylabdist(integer 1)  ///
ignoretreat          ///
bygroup              ///
style(string)         ///
byunit               ///
theme(string)         ///
lwd(string)          ///
*                   ///
]
```

where the subcommand can be:

Subcommand	Description
<code>Y D X</code>	<code>varlist</code> of outcome variable, treatment variable, and covariates, respectively. Including covariates may change the look of the plot due to missing values in these covariates.
<code>if</code> and <code>in</code>	We recommend users to add variable that is not included in the <code>varlist</code> or <code>i()</code> / <code>t()</code> but appears in the <code>if</code> / <code>in</code> subcommand to the <code>varlist</code> following <code>panelView</code> command.
<code>i()</code> and <code>t()</code>	Specify the unit (group) and time indicators.
<code>type()</code>	Use <code>type(treat)</code> to plot treatment assignment using a heatmap. Use <code>type(outcome)</code> to plot an outcome variable---or any variable---in a time series fashion. Use <code>type(bivar)</code> or <code>type(bivariate)</code> to plot the outcome and treatment variables against time in the same graph.
<code>continuoustreat</code>	The treatment variable is presented as a continuous variable in a <code>type(outcome)</code> plot
<code>discreteoutcome</code>	When a variable is discrete, make sure <code>panelView</code> respects its discreteness in <code>type(outcome)</code> plots.
<code>bytiming</code>	Sort units by the timing of first receiving the treatment; if the timing is the same, then by the total number of periods exposed to the treatment.
<code>MYCOLOR()</code>	Change the color schemes; click <a href="#">here</a> for sequential colors (3-9 colors).
<code>PREpost</code>	Distinguish the pre- and post-treatment periods for treated units.
<code>xlabdist()</code> and <code>ylabdist()</code>	Change integer gaps between labels on the x- and y-axes. Default is 1.
<code>ignoretreat</code>	Omit the treatment indicator, that is, any variables after <code>y</code> will be interpreted as covariates.
<code>bygroup</code>	Put each unit into different treatment groups, then plot them separately when <code>type(outcome)</code> is invoked.
<code>style()</code>	Determine the style of the elements in a plot. The first and second entries define the style of the outcome and treatment, respectively. <code>connected</code> or <code>c</code> for connected lines, <code>line</code> or <code>l</code> for lines, <code>bar</code> or <code>b</code> for bars.
<code>byunit</code>	Plot the outcome and treatment variables against time by each unit when <code>type(bivar)</code> is invoked.
<code>theme(bw)</code>	Use the black and white theme (default when specified <code>type(bivar)</code> ).
<code>lwd()</code>	Set the line width in <code>type(bivar)</code> (default is <code>medium</code> ).

## 2. Plotting Treatment Conditions

First, we show how to visualize the dichotomous treatment conditions in a panel dataset. The treatment may switch on and off or have missing values.

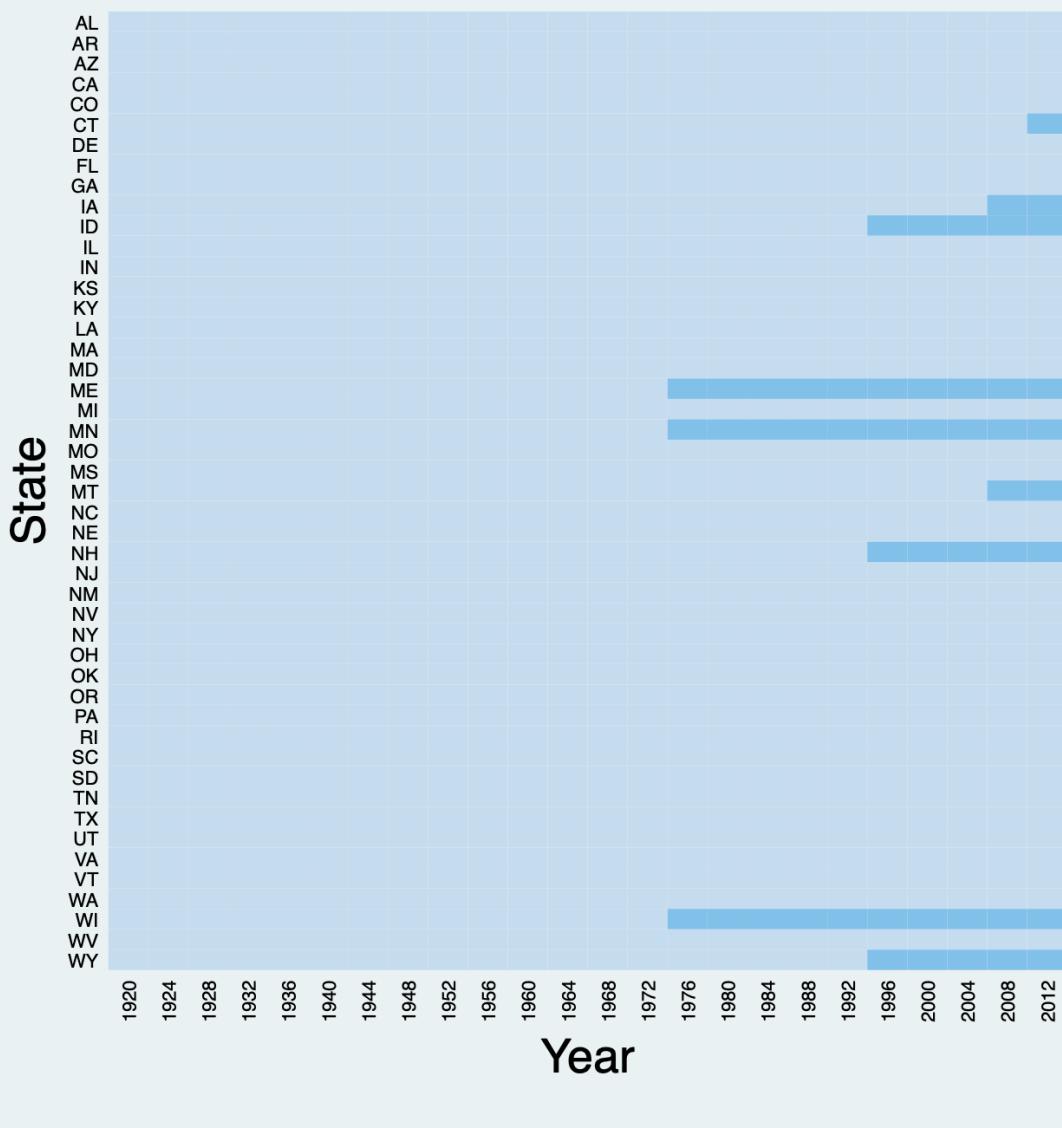
### 2.1 Two treatment conditions

Using the `turnout` dataset (a balanced panel), we show the treatment status of Election Day Registration (EDR) in each state in a given year ([Xu 2017](#)). We can use the `title` option to change the title of the plot and change the titles of x- and y-axes through `xtitle` and `ytitle`, respectively. For DID-type panel data with a dichotomous treatment indicator, we can distinguish the pre- and post-treatment periods for treated units by specifying `prepost`.

In the plot below, `turnout` is the outcome, `policy_edr` is the treatment, `policy_mail_in` and `policy_motor` are covariates.

```
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)
xtitle("Year") ytitle("State") title("Treatment Status")
```

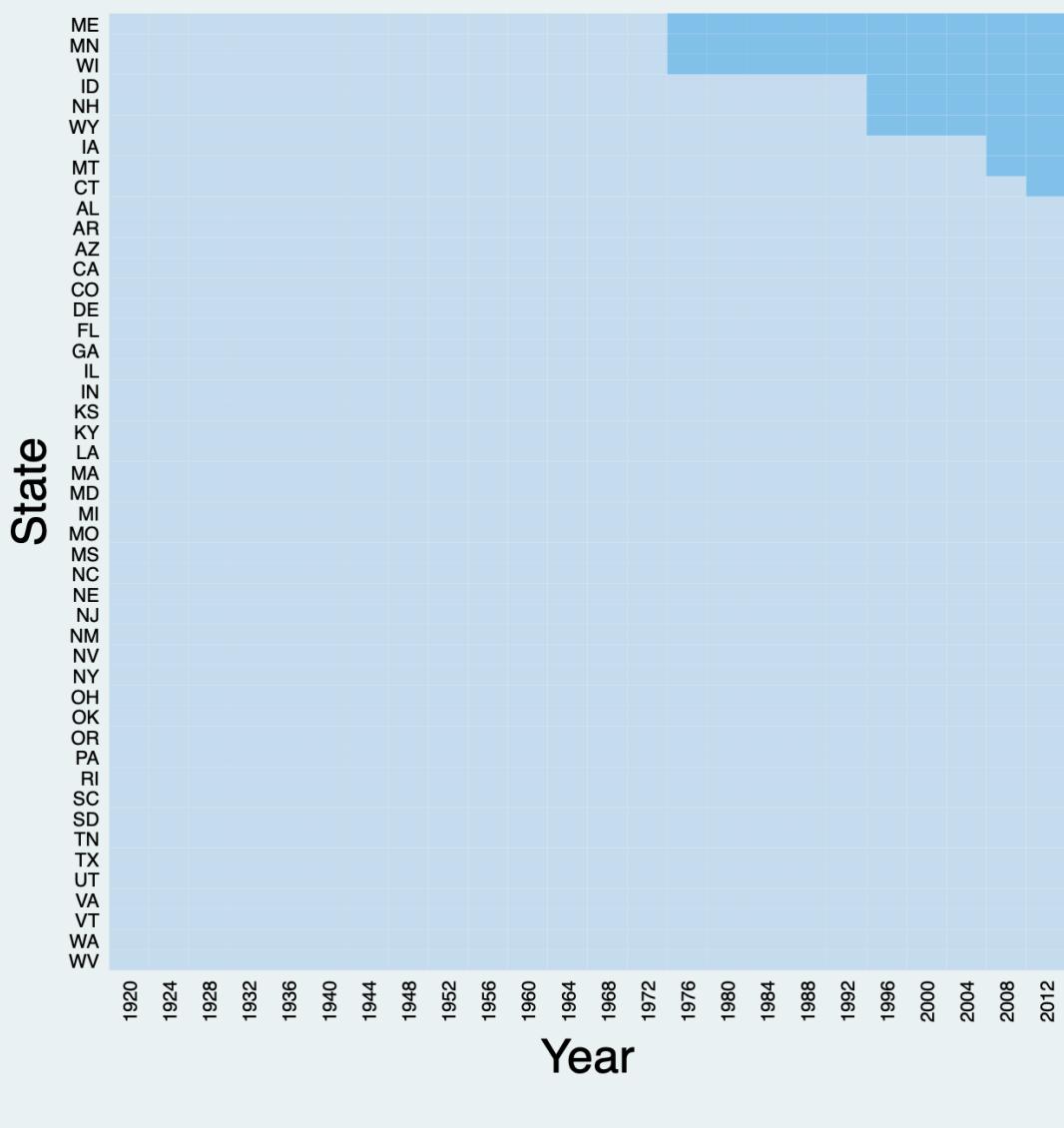
# Treatment Status



We can use the `bytiming` option to sort units by the timing of first receiving the treatment and use `legend` to change labels in the legend:

```
*bytiming  
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)  
xtitle("Year") ytitle("State") title("Treatment Status") bytiming legend(label(1 "No  
EDR") label(2 "EDR"))
```

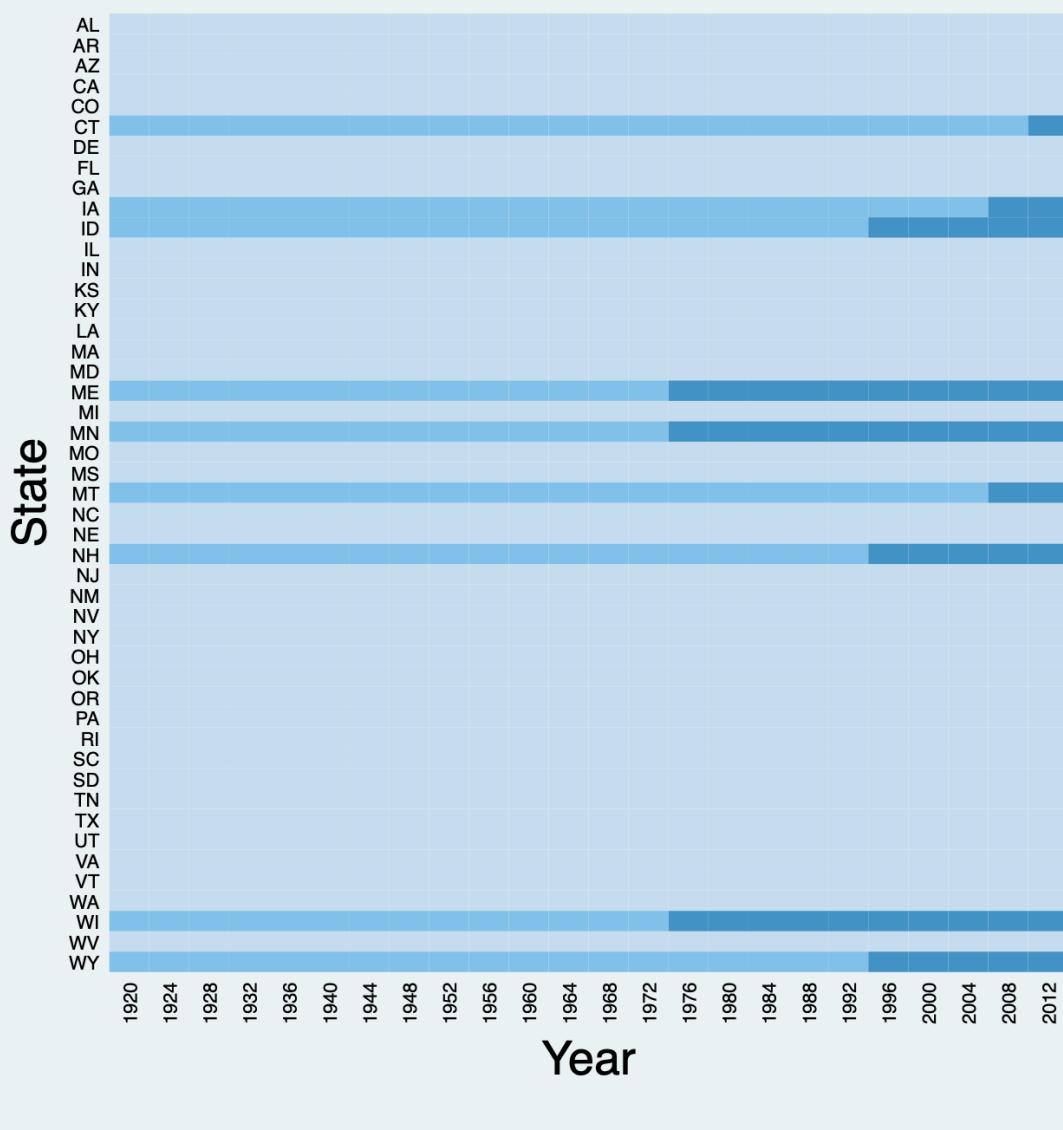
# Treatment Status



Distinguish the pre- and post-treatment periods for treated units by specifying `prepost`:

```
*prepost  
panelview turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)  
xtitle("Year") ytitle("State") title("Treatment Status") prepost
```

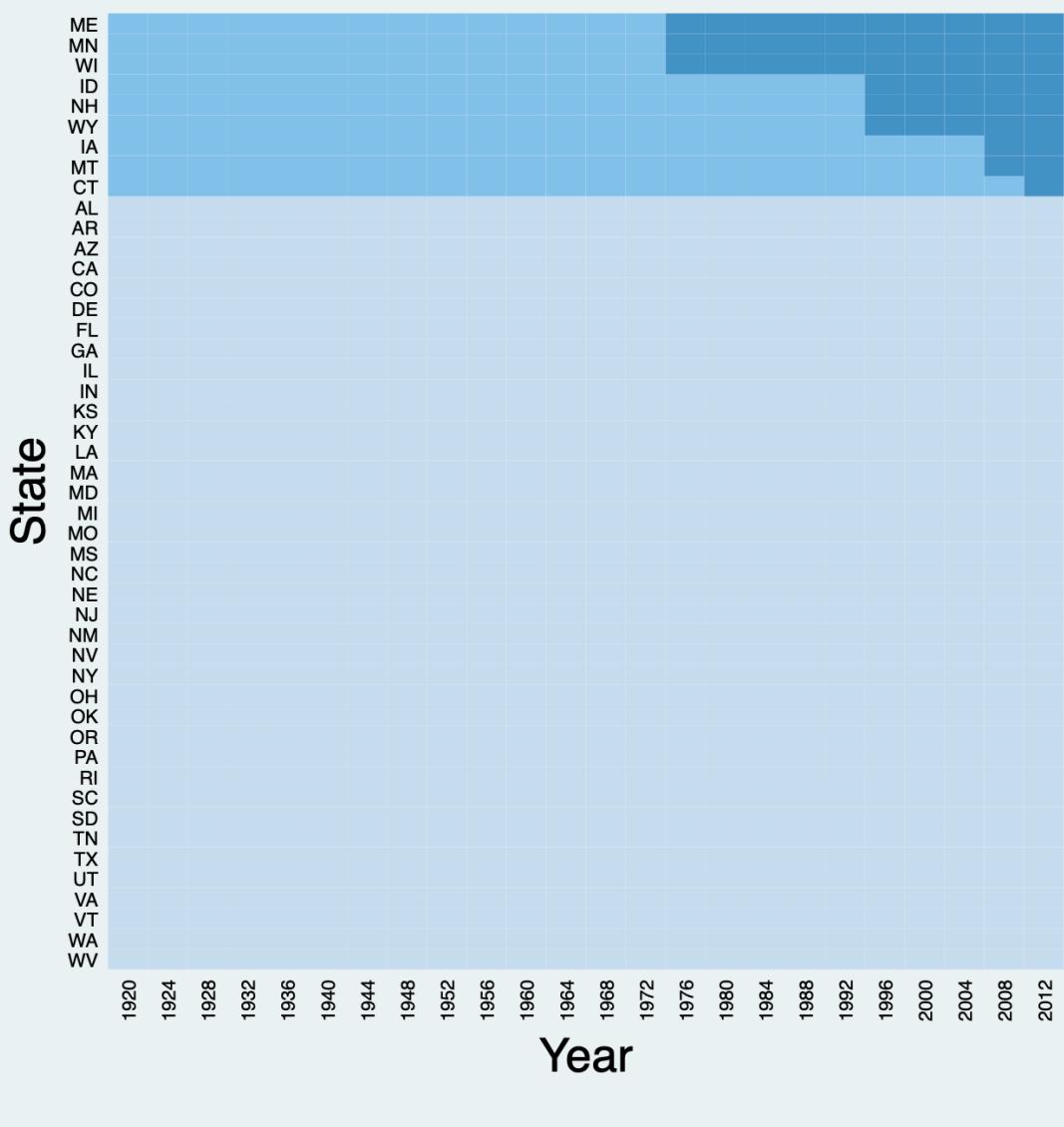
# Treatment Status



Again, sort units by the timing of receiving the treatment:

```
*bytiming
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)
xtitle("Year") ytitle("State") title("Treatment Status") prepost bytiming
```

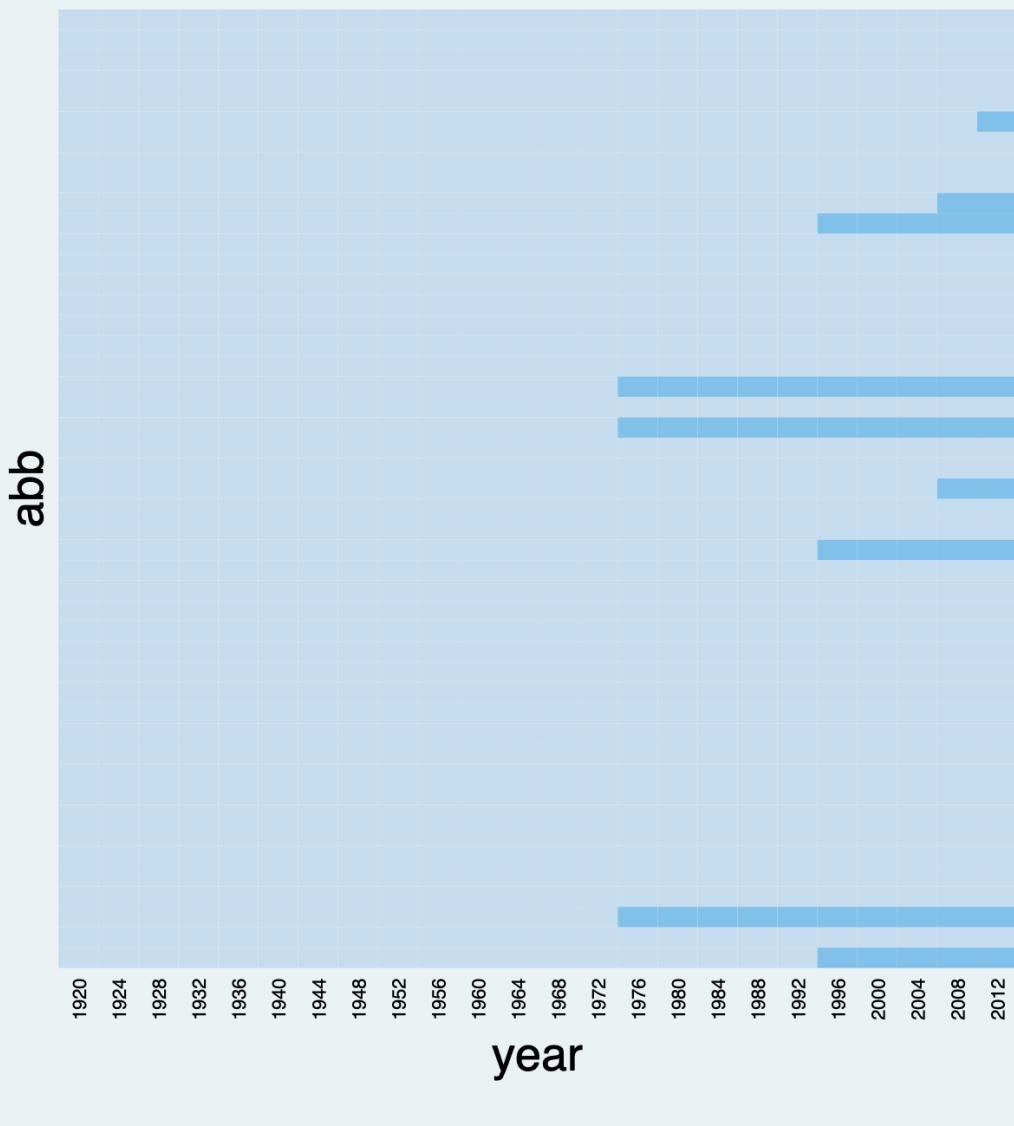
# Treatment Status



Remove the labels on the y-axis by specifying `ylabel("")` or `ylabel(none)`:

```
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)  
title("EDR Reform") ylabel("")
```

# EDR Reform



Change the color schemes for the controls and treated using the `mycolor` option. For example, `PuBu` indicates light purple to blue. Click [here](#) for more sequential colors' choice.

```
*mycolor(PuBu)
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(treat)
xtitle("Year") ytitle("State") title("Treatment Status") mycolor(PuBu) bytiming
```

# Treatment Status

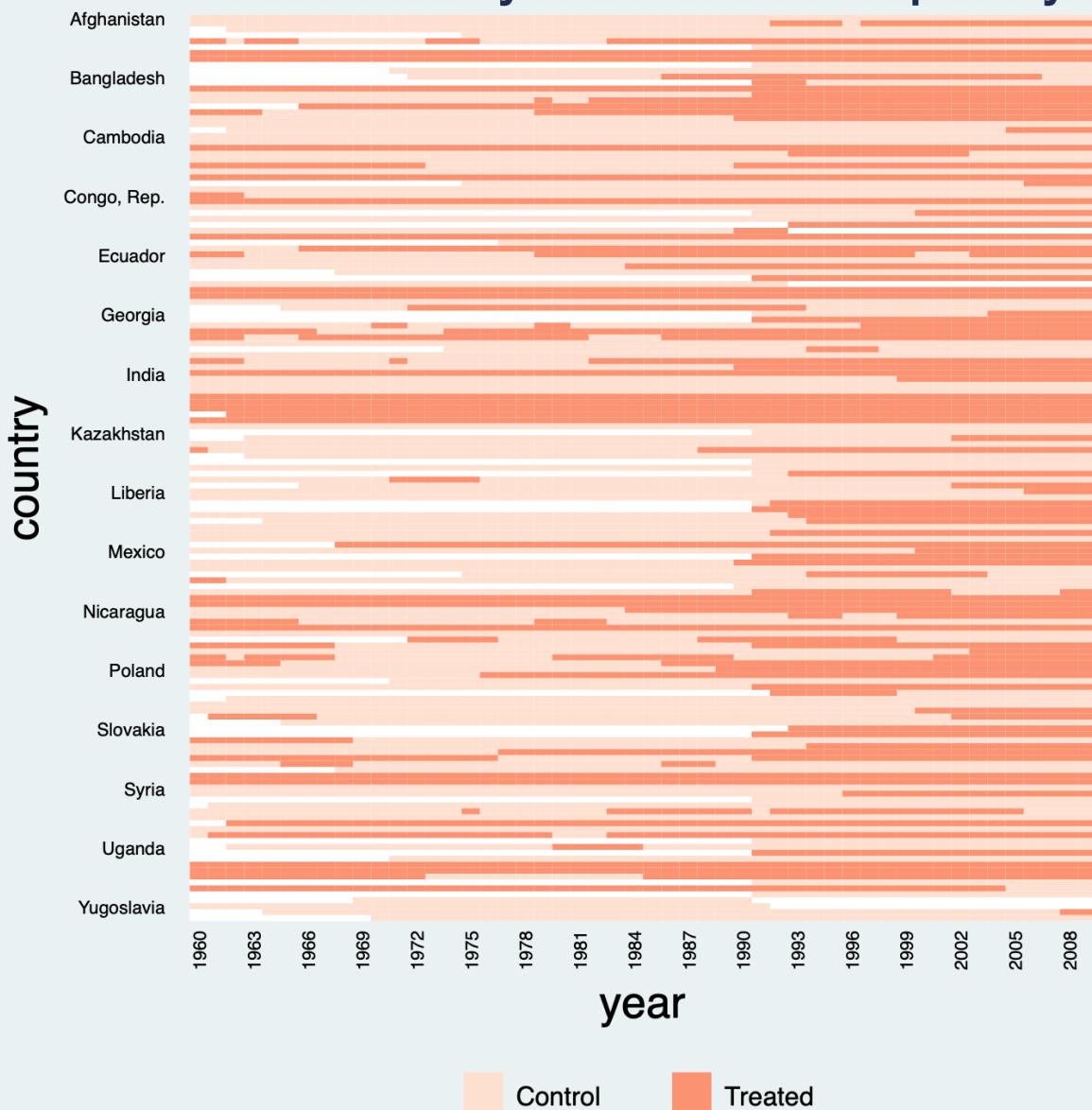


## 2.2 Treatment: missing & switch on and off

For a panel dataset in which the treatment may switch on and off, we no longer differentiate between pre- and post-treatment statuses. To demonstrate how `panelView` can be used in a more general setting, the following plot uses the `capacity` dataset, which is used to investigate the effect of democracy, the treatment, on state capacity, the outcome ([Wang and Xu 2018](#)). `demo` is a binary indicator of regime type. From the figure below, we see quite a few cases of democratic reversals and that there are many missing values (the white area). We use the `xlabdist` and `ylabdist` option to change the gaps between labels on the x- and y-axes:

```
use capacity.dta, clear
panelView lnpop demo lndgp , i(country) t(year) type(treat) mycolor(Reds)
title("Democracy and State Capacity") xlabdist(3) ylabdist(10)
```

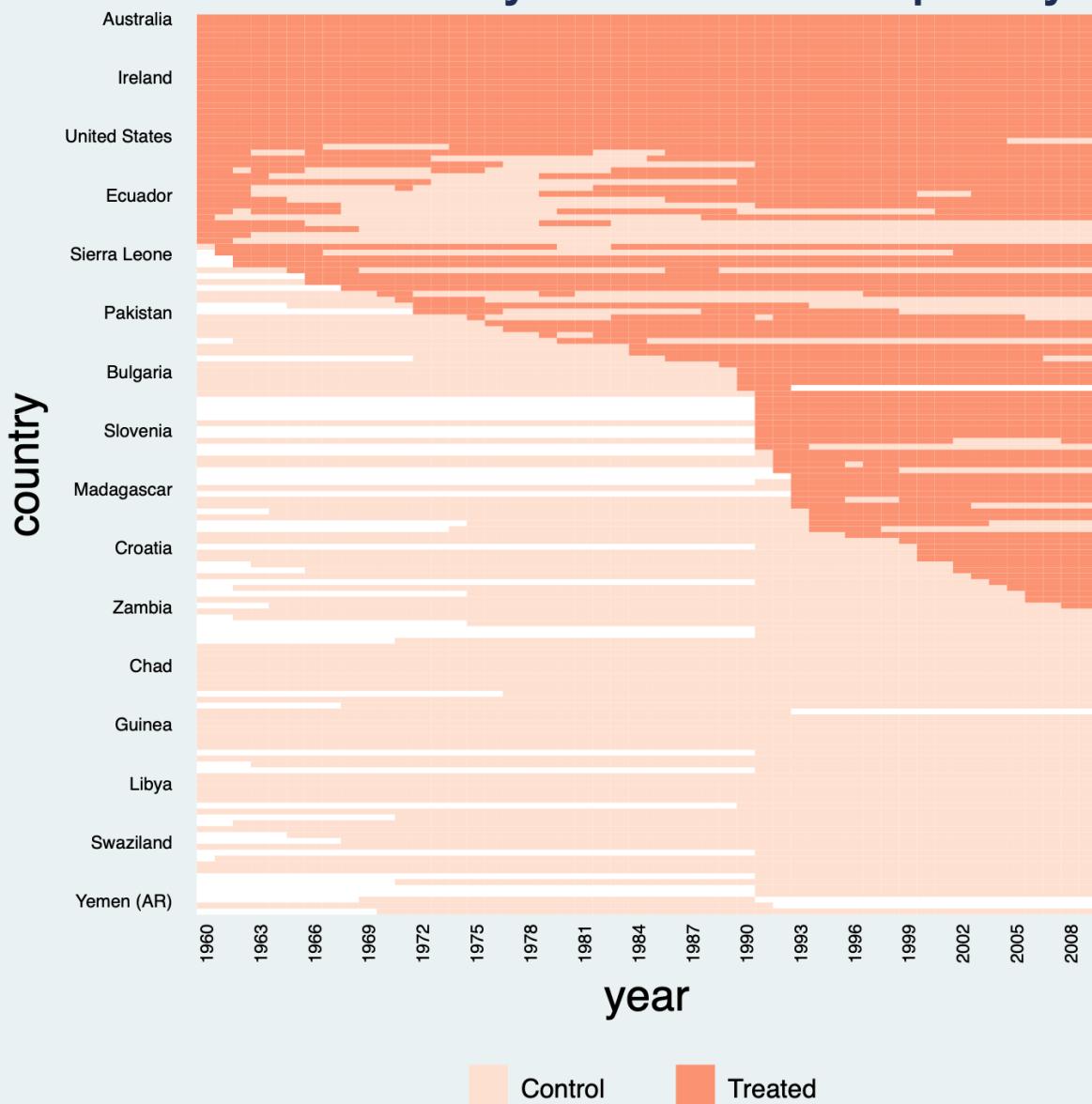
# Democracy and State Capacity



Sorting units based on the first period a unit receives the treatment gives a more appealing visual:

```
*bytiming
panelView lnpop demo lngdp, i(country) t(year) type(treat) mycolor(Reds)
title("Democracy and State Capacity") xlabdist(3) ylabdist(10) bytiming
```

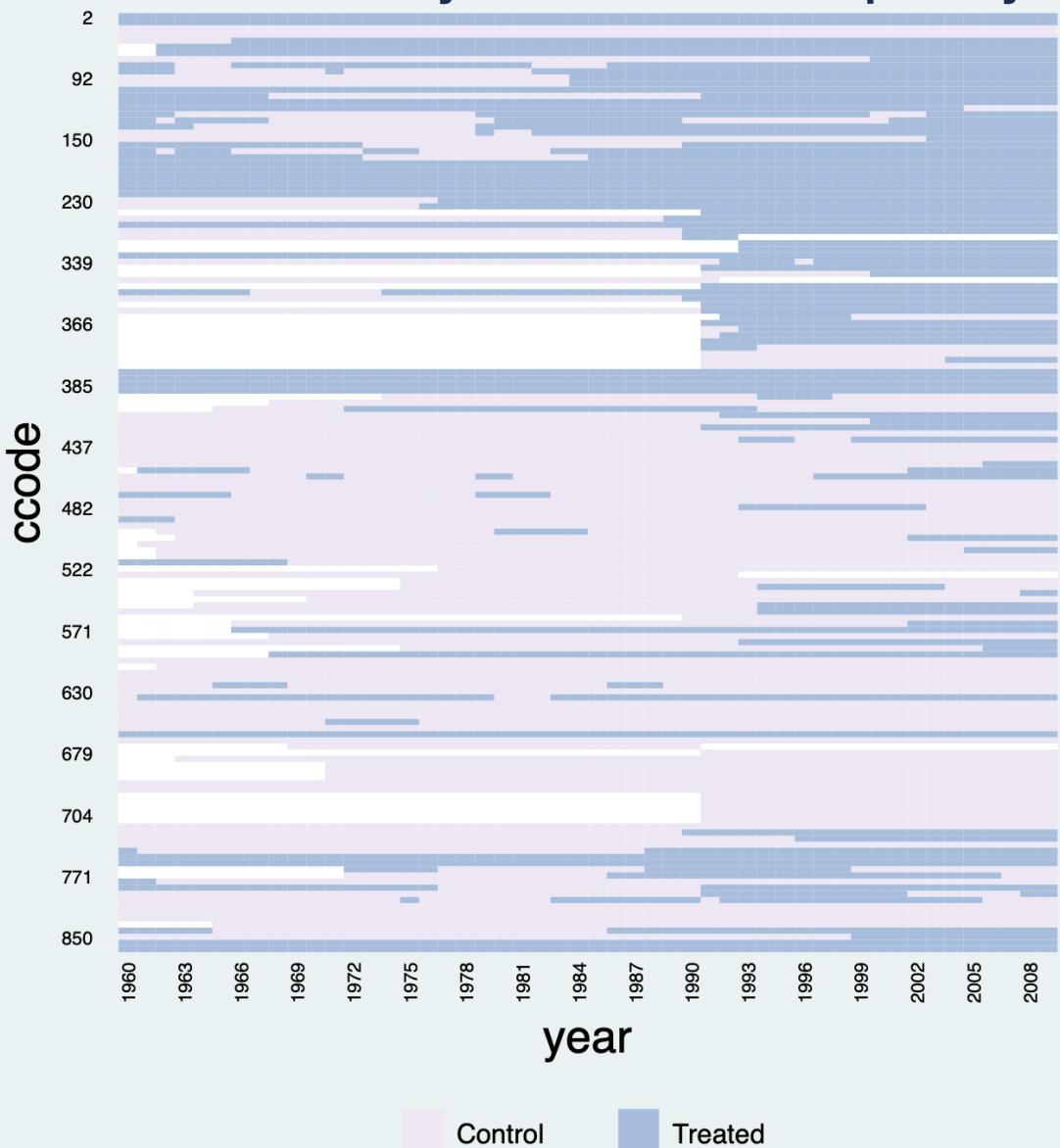
# Democracy and State Capacity



Instead of indicate `country` as units, we use `i(ccode)` to indicate country code as units, which will change the label and sequence in our figure:

```
panelview lnpop demo lnpdgp, i(ccode) t(year) type(treat) mycolor(PuBu) title("Democracy and State Capacity") xlabdist(3) ylabdist(10) //If we set ylabdist(11), the "155" appears at the bottom of ylabel and is hard to remove, different with R package
```

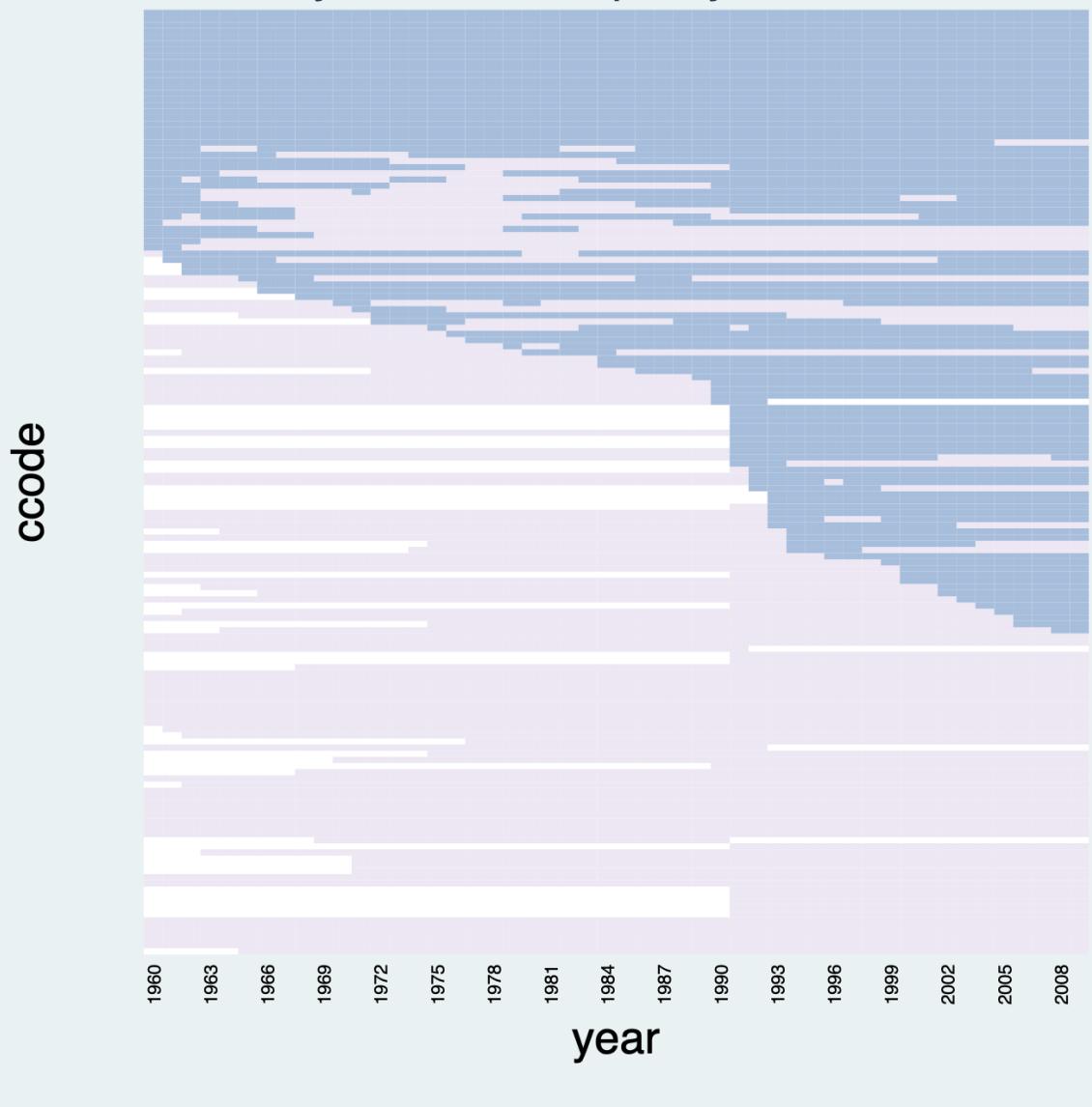
# Democracy and State Capacity



Sort units based on the first period a unit receives the treatment and use `ylabel("none")` to remove the labels on the y-axis:

```
*bytiming
panelView lnpop demo lngdp, i(ccode) t(year) type(treat) mycolor(PuBu) title("Democracy
and State Capacity: Treatment Status", size(medsmall)) bytiming xlabdist(3)
ylabel("none")
```

## Democracy and State Capacity: Treatment Status

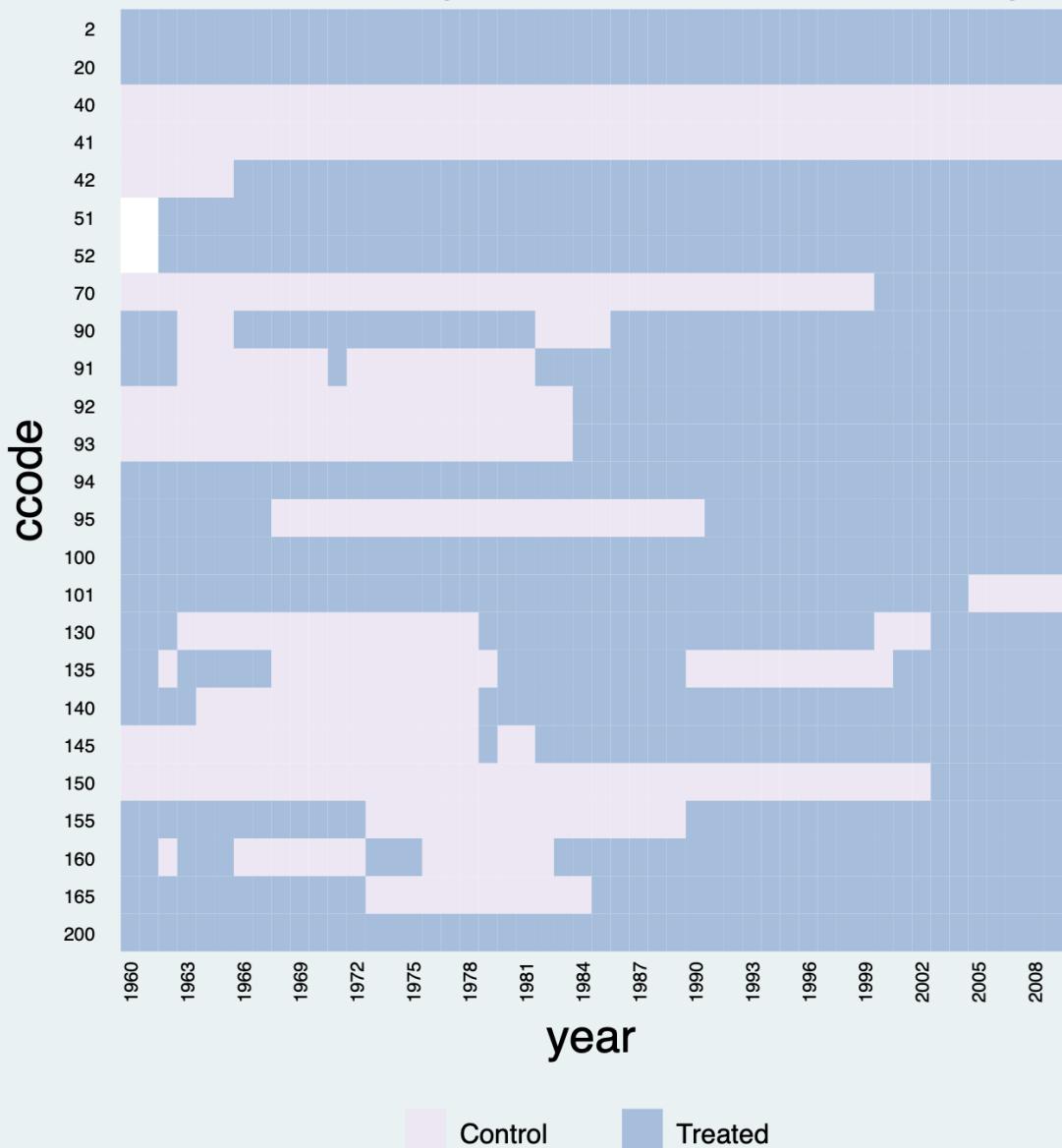


### 2.3 Plotting a subset of units

Sometimes a dataset has many units and we only want to take a peak of a subset of the units. `panelView` allows users to specify the units to be shown by the `if` subcommand. Note that if any variable not included in the `varlist` or `i()` / `t()` following `panelView` appears in the `if` or `in` command, we recommend researchers to add such variable into the `varlist` following `panelView`. In the following figure, we plot the treatment statuses of the first 25 units:

```
use capacity.dta, clear
egen ccodeid = group(ccode)
panelView lnpop demo lngdp ccodeid if ccodeid >= 1 & ccodeid <= 26, i(ccode) t(year)
type(treat) mycolor(PuBu) title("Democracy and State Capacity") xlabdist(3)
```

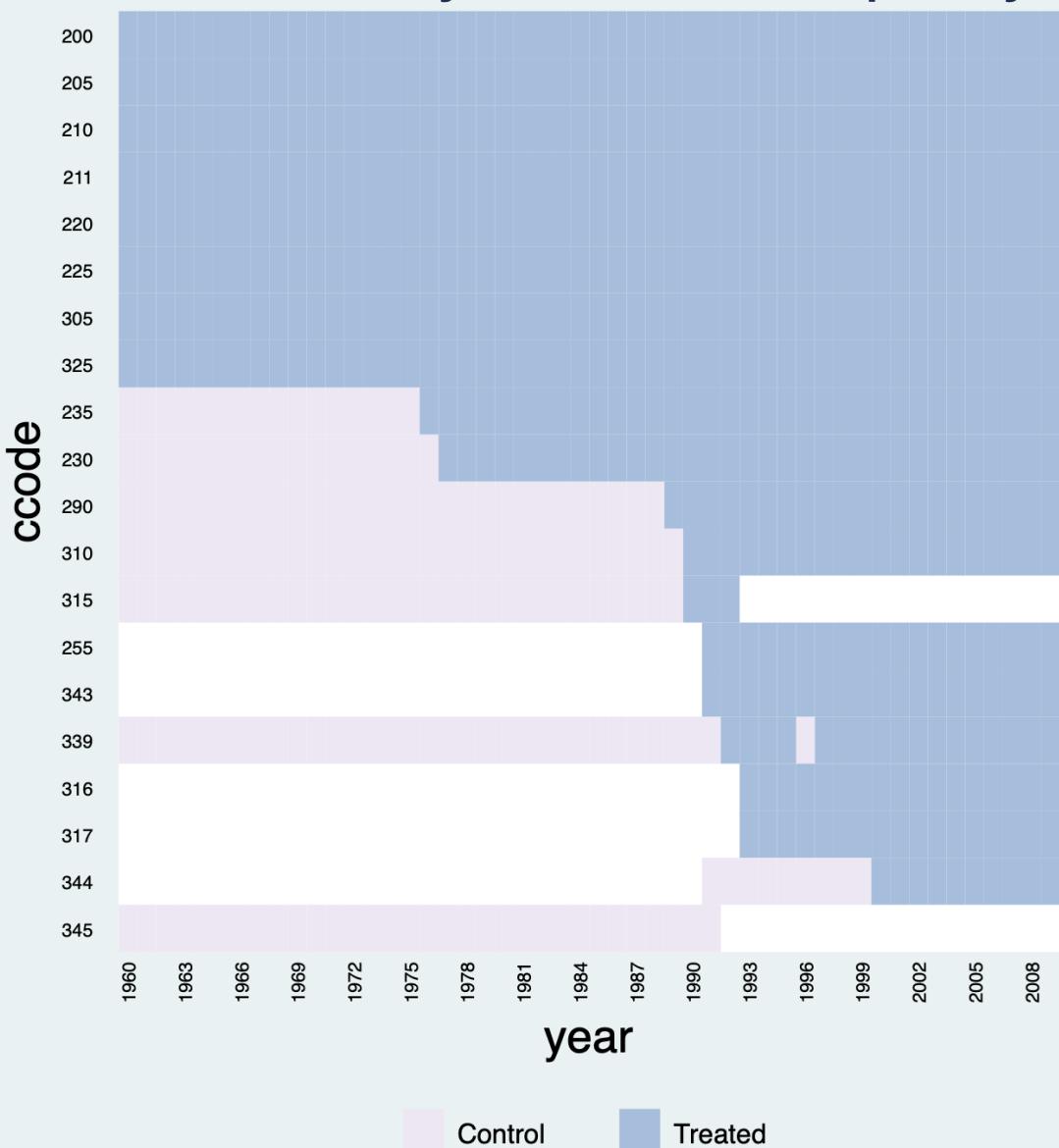
# Democracy and State Capacity



Sort units based on the first period a unit receives the treatment:

```
*bytiming
panelview lnpop demo lngdp ccodeid if ccodeid >= 26 & ccodeid <= 51, i(ccode) t(year)
type(treat) mycolor(PuBu) title("Democracy and State Capacity") xlabdist(3) bytiming
```

# Democracy and State Capacity



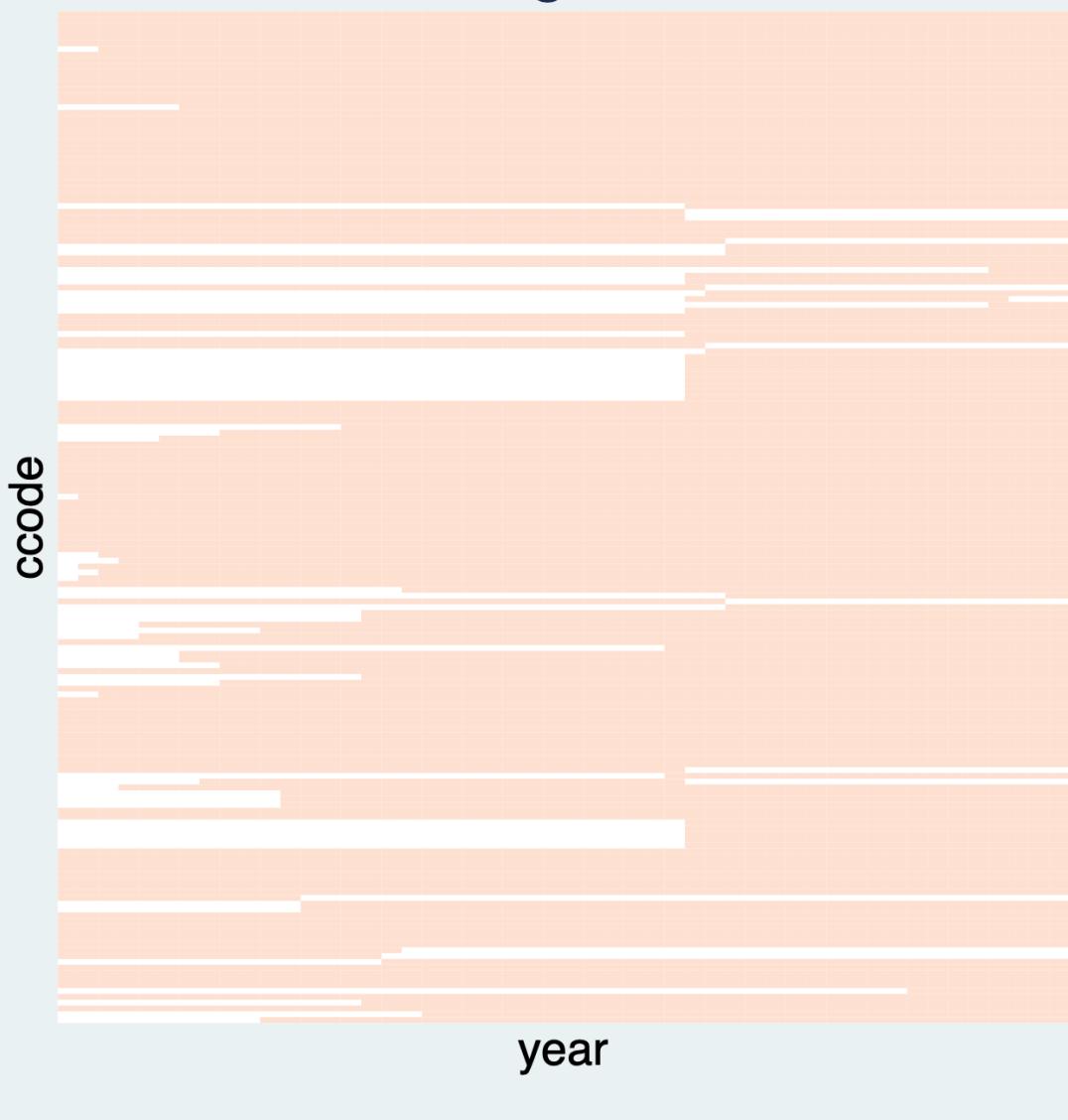
## 3. Ignoring Treatment Conditions

### 3.1 `ignoretreat` subcommand

Omit the treatment variable in a `type(treat)` plot, in which case, the plot will show missing (the white area) and non-missing values only.

```
use capacity.dta, clear
panelView demo, i(ccode) t(year) type(treat) mycolor(Reds) title("Missing Values")
xlabel(none) ylabel(none) ignoretreat
```

# Missing Values

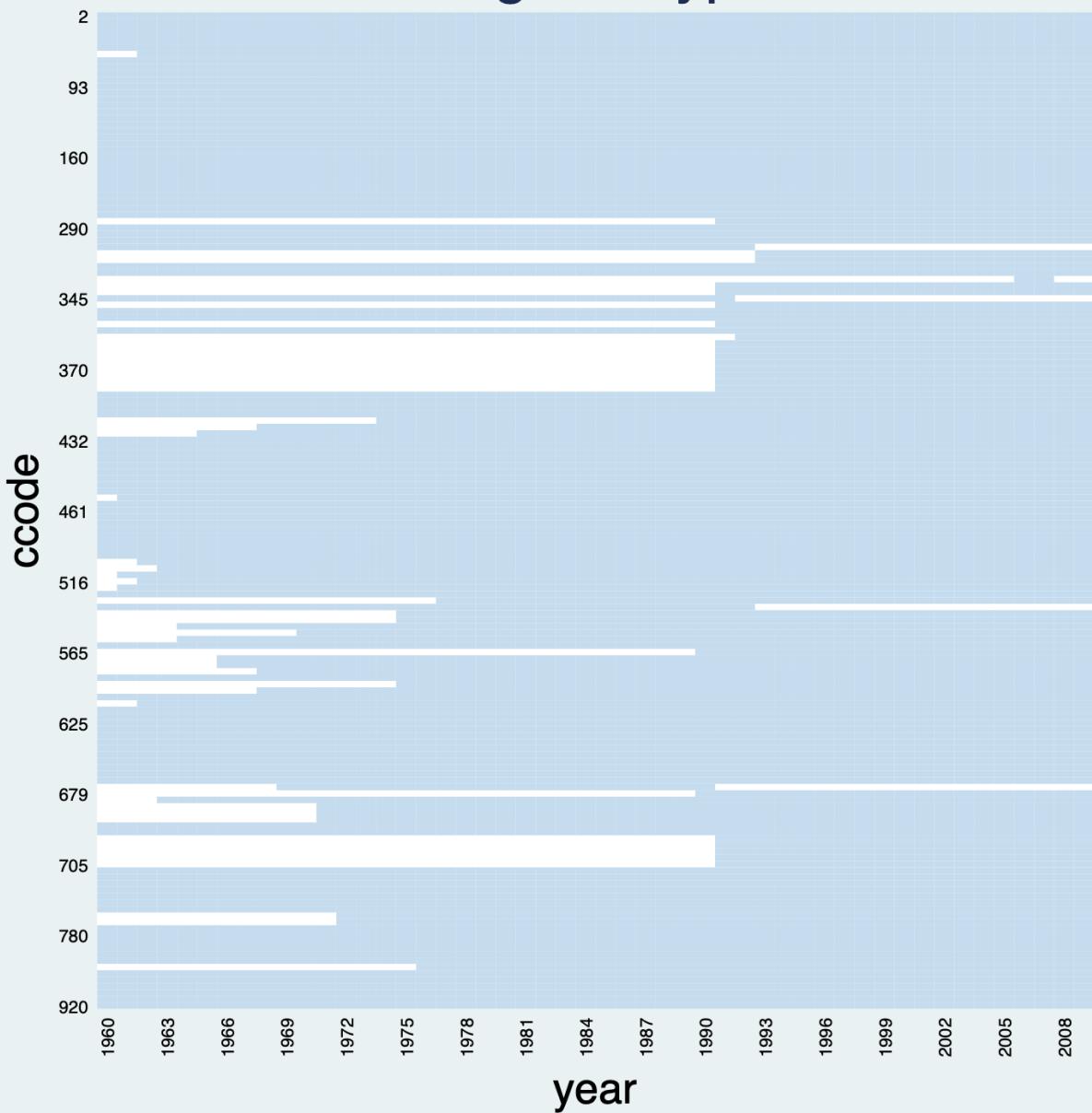


## 3.2 Treatment level = 1 & Plotting treatment

If the treatment indicator has only 1 level, then treatment status will not be shown on the `type(treat)` plot, which is the same as `ignoretreat`:

```
use capacity.dta, clear
gen demo2 = 0
panelView Capacity demo2 lngdp, i(ccode) t(year) type(treat) title("Regime Type")
xlabdist(3) ylabdist(11) legend(off) // type(treat) & number of treatment level = 1:
same as ignoretreat
```

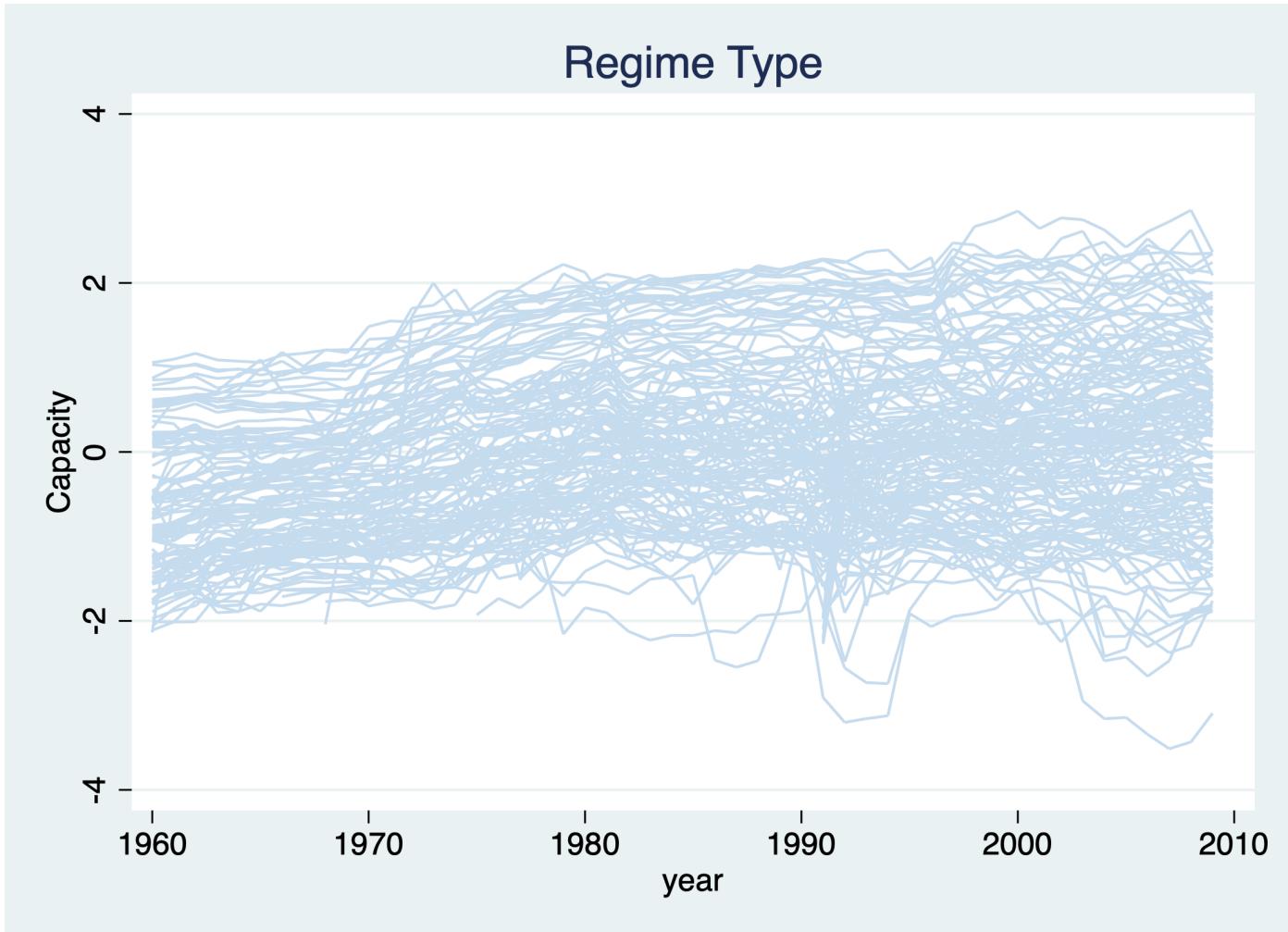
# Regime Type



## 3.3 Treatment level = 1 & Plotting outcome

If the treatment indicator has only 1 level, then treatment status will not be shown on the `type(outcome)` plot, which is the same as `ignoretreat`:

```
use capacity.dta, clear
gen demo2 = 0
panelView Capacity demo2 lngdp, i(ccode) t(year) type(outcome) title("Regime Type")
legend(off) // type(outcome) & number of treatment level = 1: same as ignoretreat
```



## 3.4 Plotting outcome & Continuous treatment / More than two treatment levels

If the treatment indicator has more than 2 treatment levels or is a continuous variable, then treatment status will not be shown on the `type(outcome)` plot. In other words, `type(outcome)` combined with `continuoustreat` or `> 2` treatment levels is the same as `ignoretreat`.

### 3.4.1 Continuous outcomes

With a continuous treatment variable (e.g. `polity2`), the treatment status will not be shown on the `type(outcome)` plot. We also indicate `theme(bw)` for black and white color style.

```
use capacity.dta, clear
* Continuous Outcome: Capacity; Continuoustreat: polity2
panelView Capacity polity2 lngdp, i(ccode) t(year) type(outcome) continuoustreat
title("Measuring Stata Capacity") legend(off) theme(bw)
```

## Measuring Stata Capacity



Same as the following two commands:

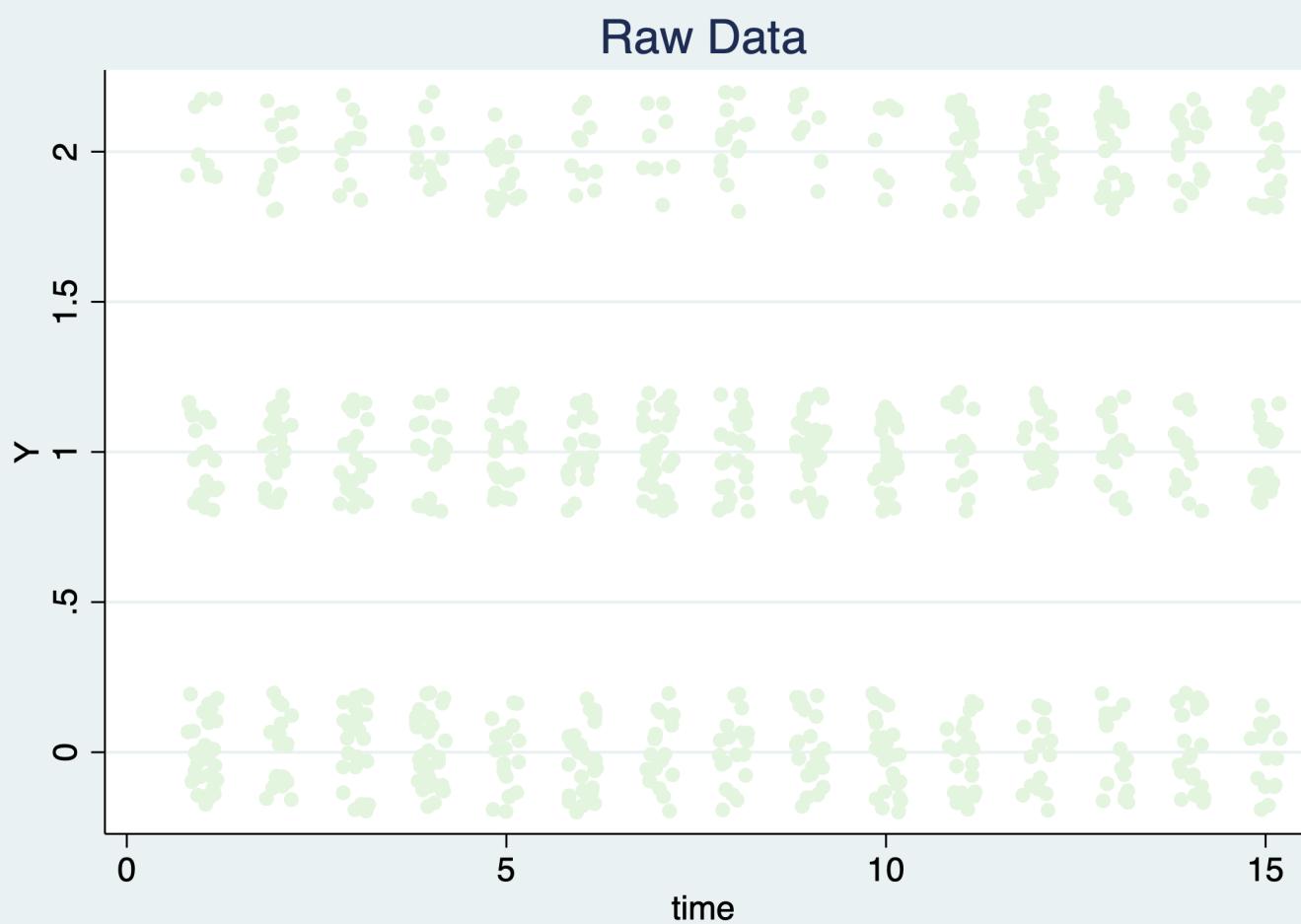
```
use capacity.dta, clear
panelView Capacity demo lngdp, i(ccode) t(year) type(outcome) title("Measuring Stata
Capacity") ignoretreat legend(off)
```

```
* Treatment indicator has more than 2 treatment levels
* Continuous Outcome: Capacity
use capacity.dta, clear
gen demo2 = 0
replace demo2 = -1 if polity2 < -0.5
replace demo2 = 1 if polity2 > 0.5
tab demo2, m
panelView Capacity demo2 lngdp, i(ccode) t(year) type(outcome) title("Measuring Stata
Capacity") legend(off) // number of treatment level = 3
```

### 3.4.2 Discrete outcomes

When the number of treatment levels is more than two, the treatment status will not be shown on the `type(outcome)` plot:

```
use simdata.dta, replace  
replace D = 2 if time < 5  
tab D, m  
panelView Y D, type(outcome) i(id) t(time) mycolor(Greens) discreteoutcome title("Raw  
Data") // number of treatment level = 3
```



Same as the following two commands:

```
use simdata.dta, replace  
panelView Y D, type(outcome) i(id) t(time) mycolor(Greens) discreteoutcome title("Raw  
Data") ignoretreat
```

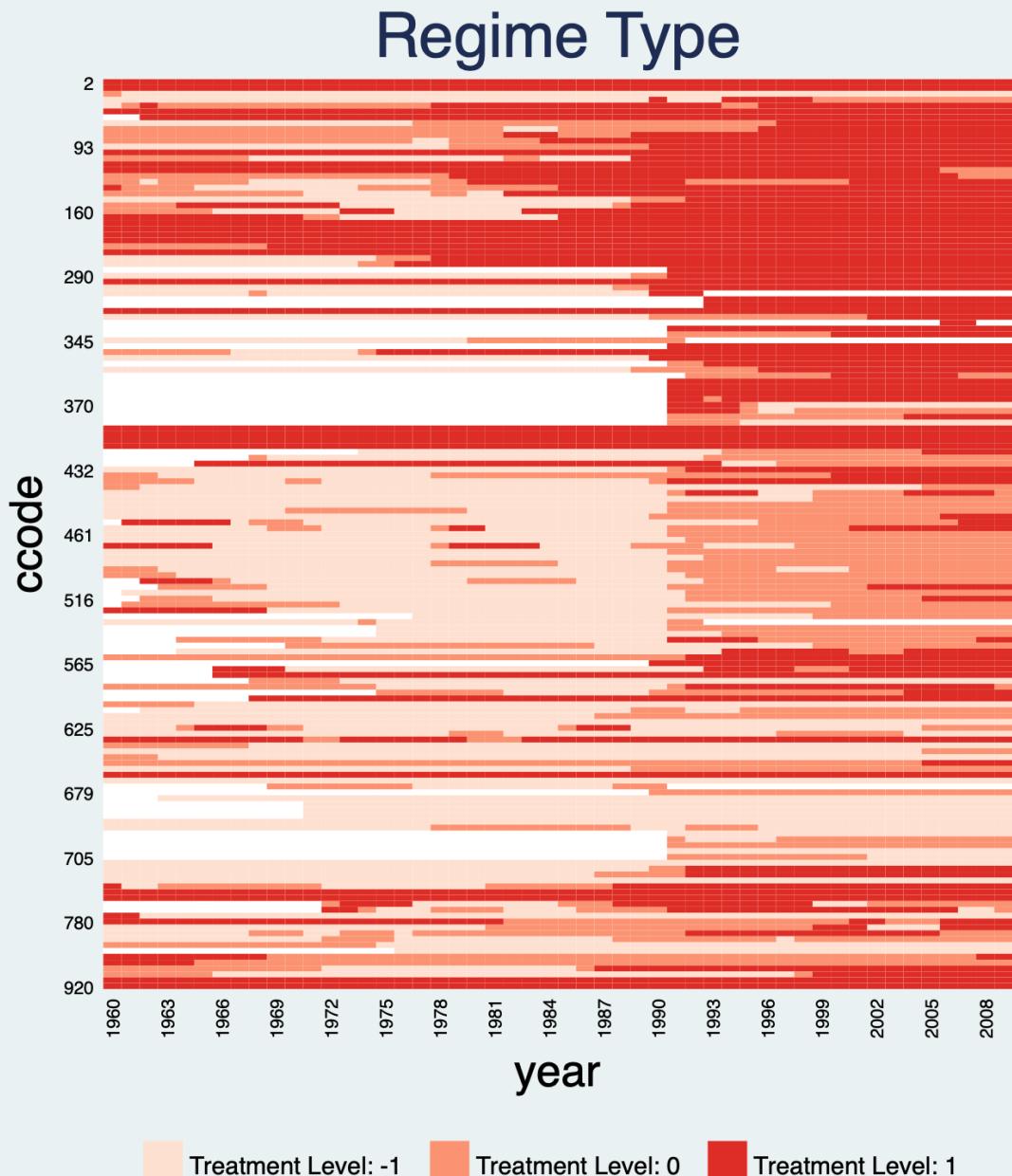
```
use simdata.dta, replace  
range x 0 1  
panelView Y x, type(outcome) i(id) t(time) discreteoutcome title("Raw Data")  
continuoustreat theme(bw) // continuous treatment & black and white theme
```

## 4. More Than Two Treatment Conditions

### 4.1 Treatment level = 3

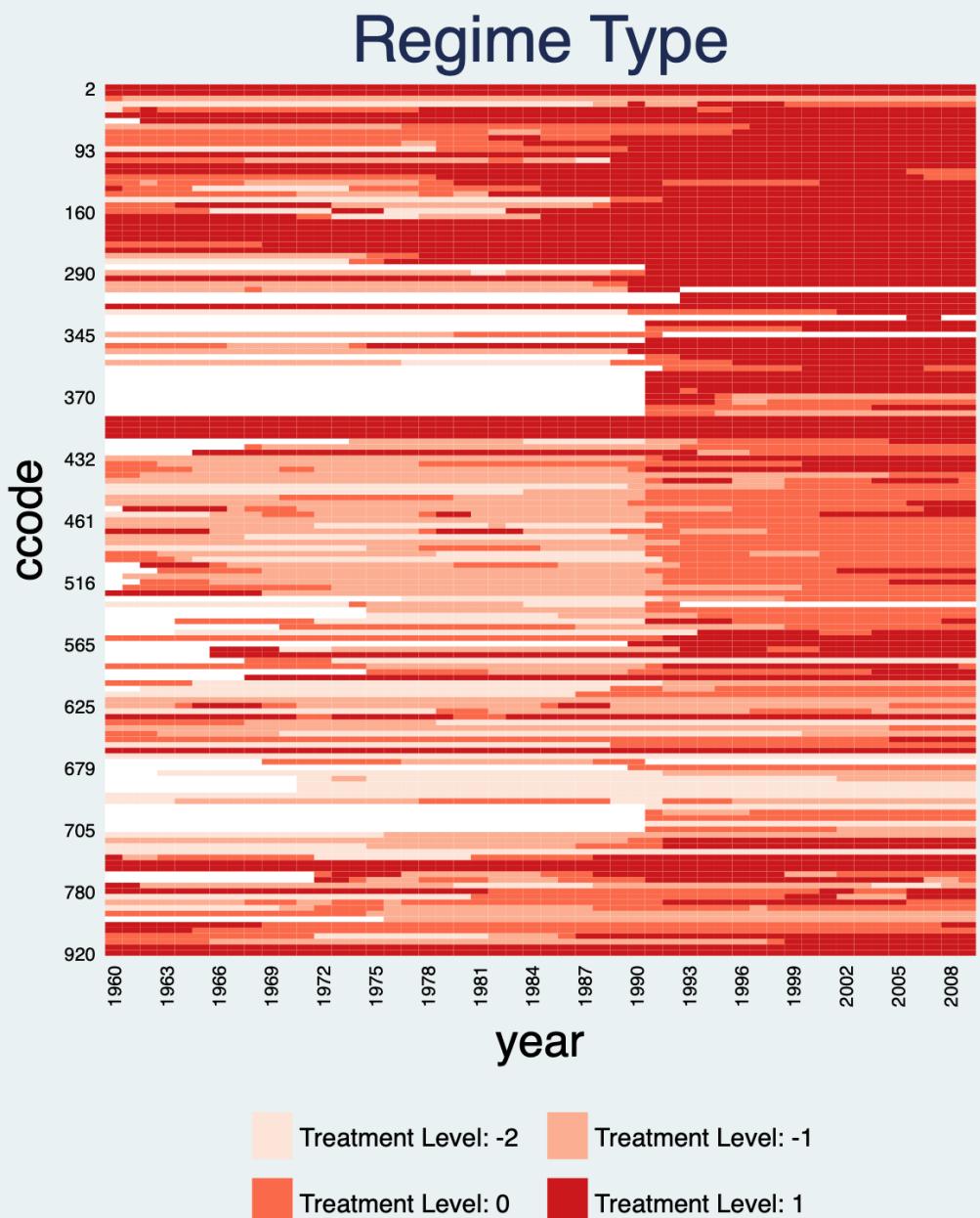
**panelView** supports panel data with more than 2 treatment levels. For example, we create a measure of regime type with three treatment levels:

```
use capacity.dta, clear
gen demo2 = 0
replace demo2 = -1 if polity2 < -0.5
replace demo2 = 1 if polity2 > 0.5
panelView Capacity demo2 lngdp, i(ccode) t(year) type(treat) title("Regime Type")
xlabdist(3) ylabdist(11) mycolor(Reds) // type(treat) & number of treatment level = 3
```



## 4.2 Treatment level = 4

```
use capacity.dta, clear
gen demo2 = 0
replace demo2 = -2 if polity2 < -0.7
replace demo2 = -1 if polity2 < -0.5 & polity2 > -0.7
replace demo2 = 1 if polity2 > 0.5
panelView Capacity demo2 lndgp, i(ccode) t(year) type(treat) title("Regime Type")
xlabdist(3) ylabdist(11) mycolor(Reds) // number of treatment level = 4
```

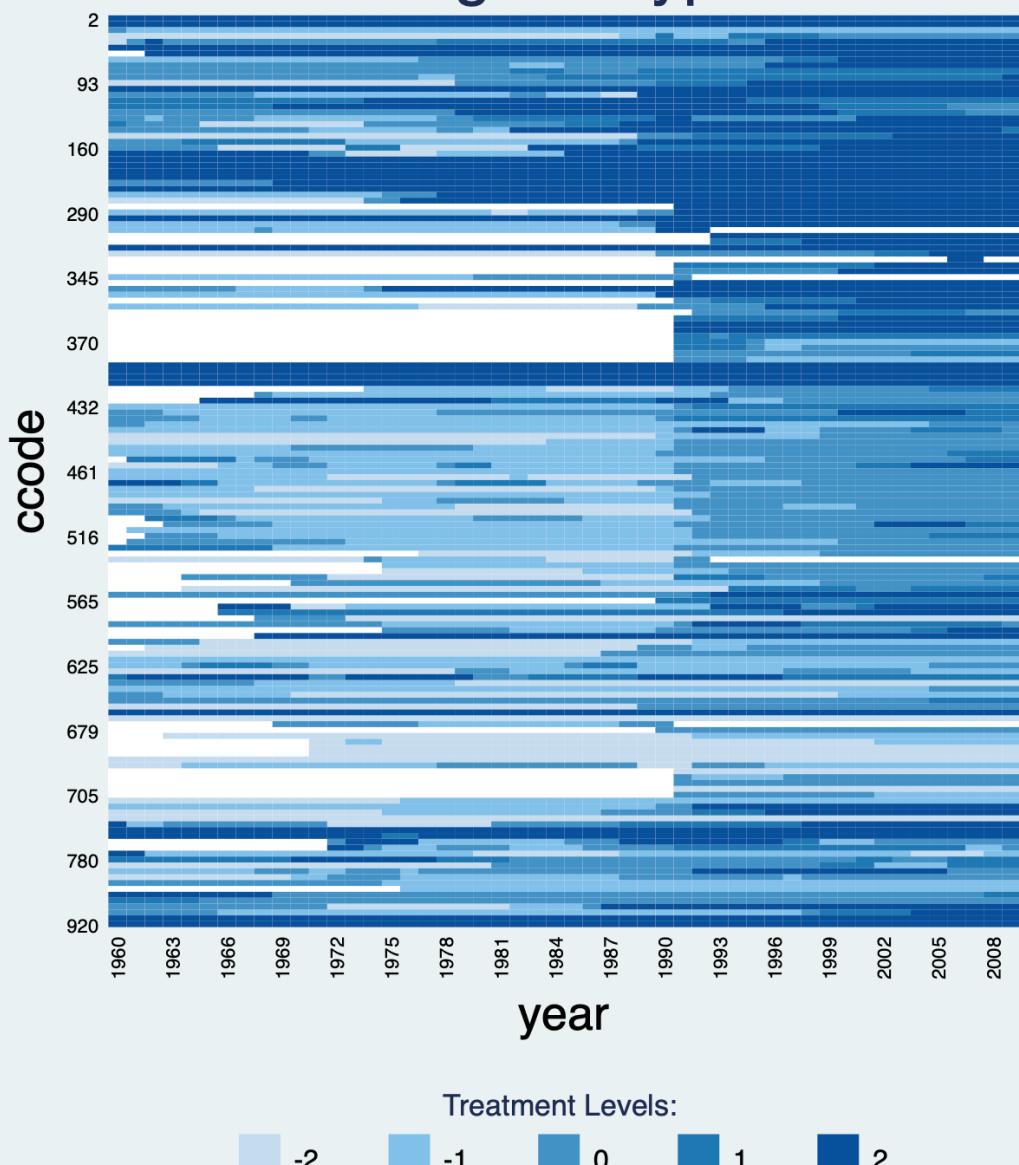


## 4.3 Treatment level >= 5

If the number of treatment levels is greater than 5, then the treatment indicator will be regarded as a continuous variable.

```
use capacity.dta, clear
gen demo2 = 0
replace demo2 = -2 if polity2 < -0.7
replace demo2 = -1 if polity2 < -0.5 & polity2 > -0.7
replace demo2 = 1 if polity2 > 0.5 & polity2 < 0.7
replace demo2 = 2 if polity2 > 0.7
tab demo2, m
panelView Capacity demo2 lngdp, i(ccode) t(year) type(treat) title("Regime Type")
xlabdist(3) ylabdist(11) continuoustreat
```

# Regime Type

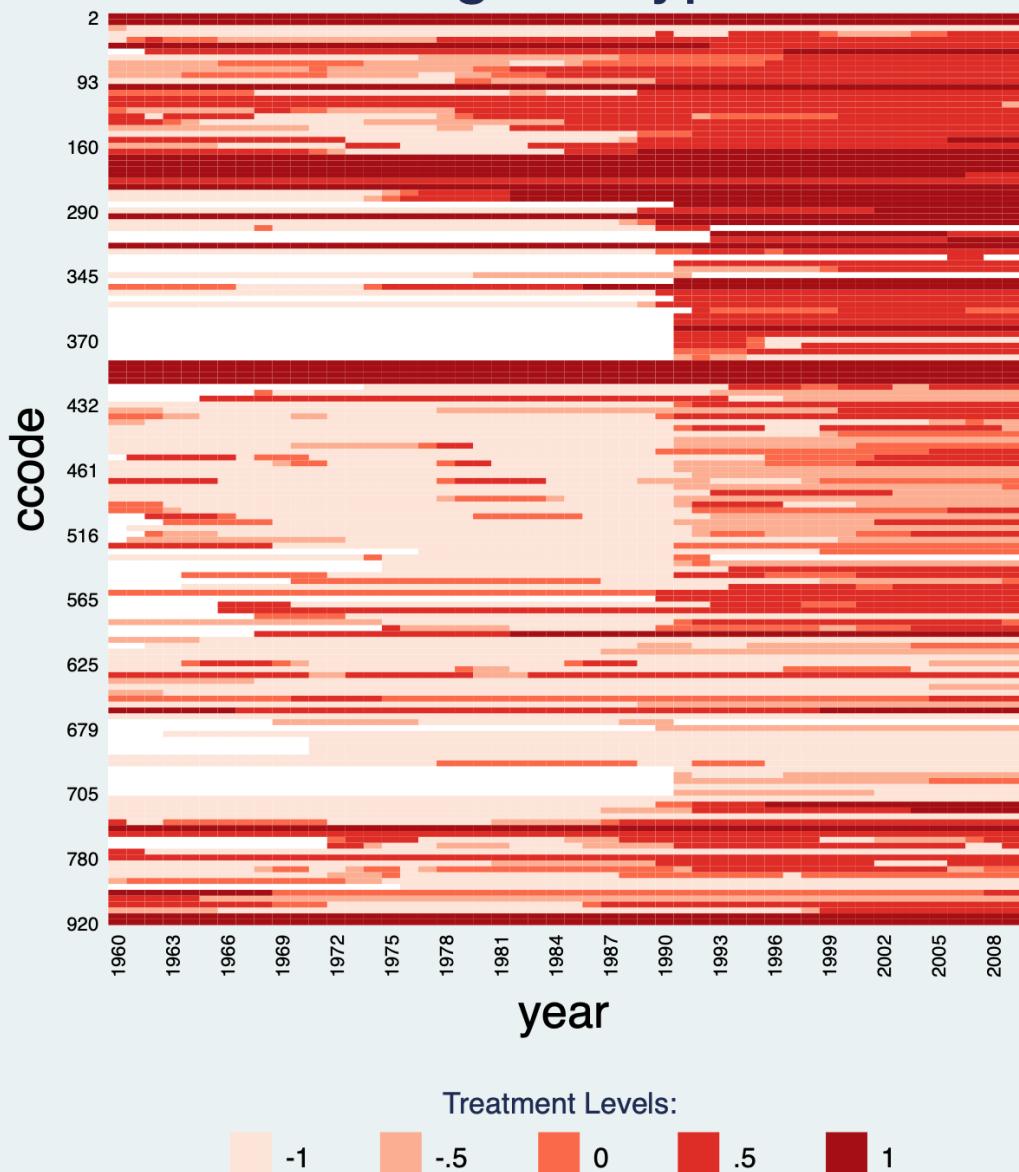


## 4.4 Continuous treatment

Plot the continuous treatment variable by `continuoustreat`.

```
use capacity.dta, clear
panelView lngdp polity2, i(ccode) t(year) type(treat) continuoustreat mycolor(Reds)
title("Regime Type") xlabdist(3) ylabdist(11)
```

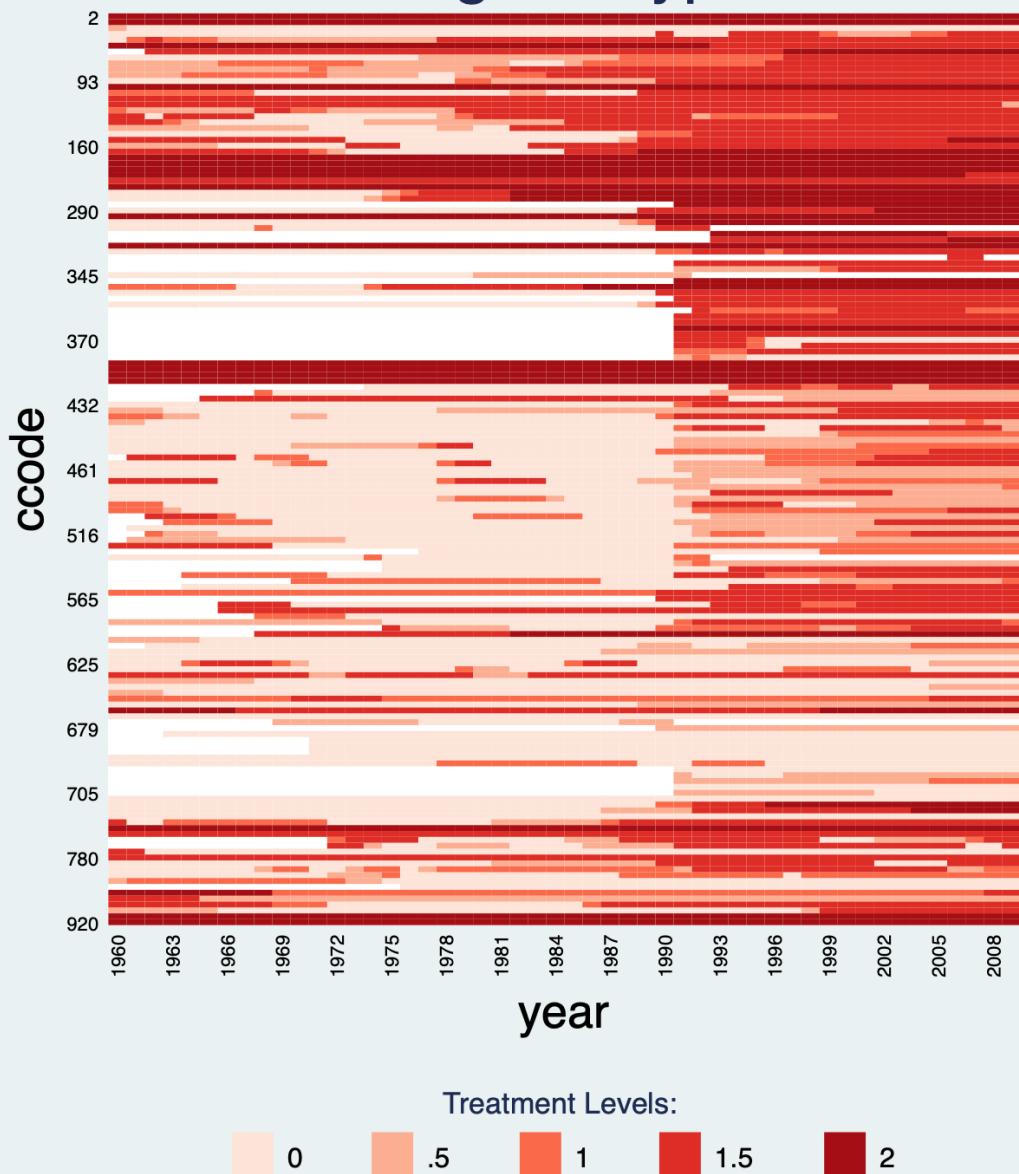
# Regime Type



If we change the level of the continuous treatment variable, the legend will modify correspondingly:

```
use capacity.dta, clear
replace polity2 = polity2 + 1
panelView lngdp polity2, i(ccode) t(year) type(treat) continuoustreat mycolor(Reds)
title("Regime Type") xlabdist(3) ylabdist(11)
```

# Regime Type



## 5. Continuous Outcomes

The second functionality of **panelView** is to show the raw outcome variable of a panel dataset in a time-series fashion. The syntax is very similar except that we need to specify `type(outcome)`. Different colors represent different treatment conditions.

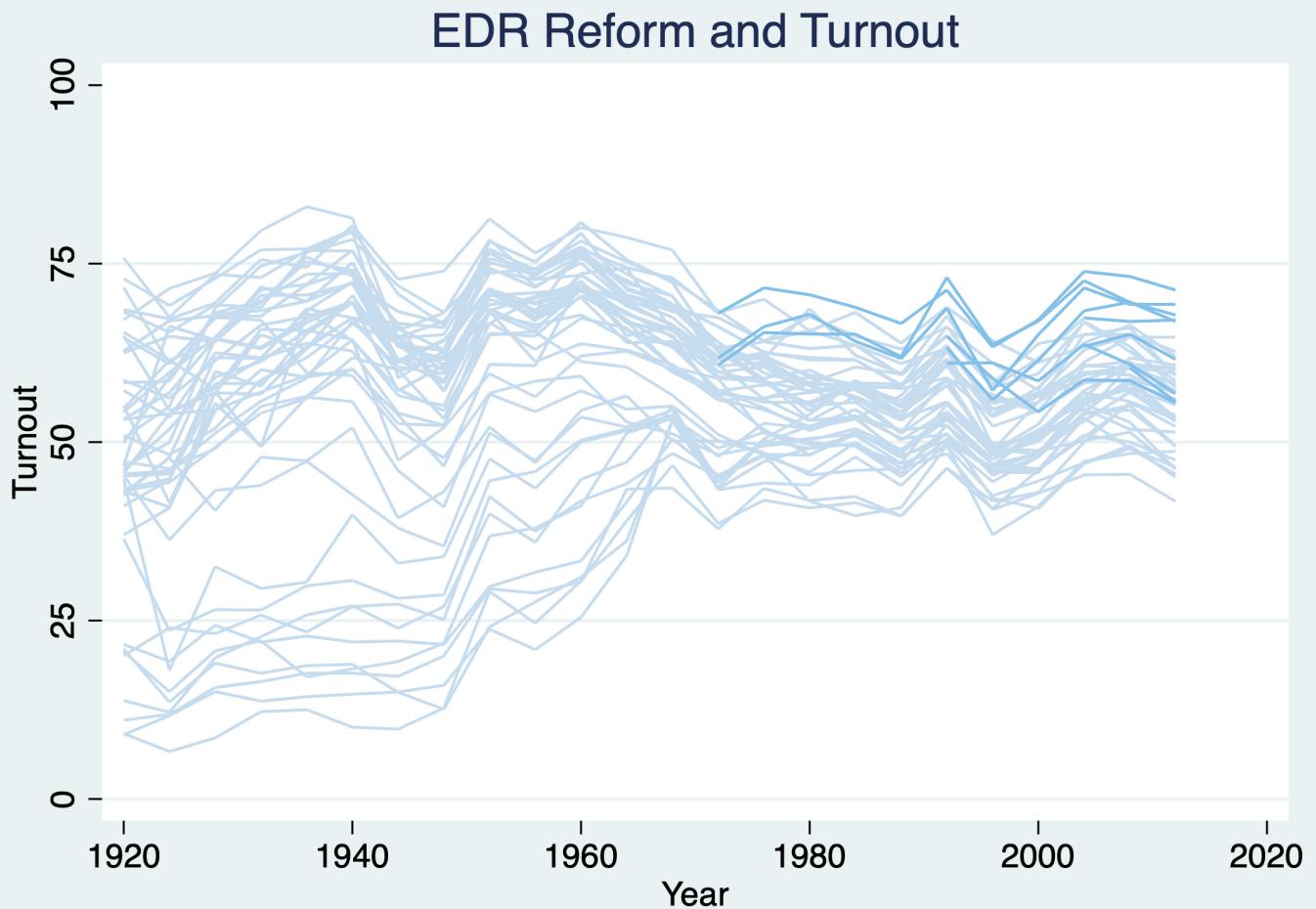
### 5.1 Continuous outcomes

Note that we paint the period right before when the treatment begin as treated period. Different with `type(treat)`, `type(outcome)` does not need `xlabdist` and `ylabeldist`. If needed, we should use `xlabel` and `ylabel` instead.

```

* Continuous outcome: turnout: 0-100; Discrete Treatment: policy_edr: 0/1
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(outcome)
xtitle("Year") ytitle("Turnout") title("EDR Reform and Turnout") ylabel(0 (25) 100)

```



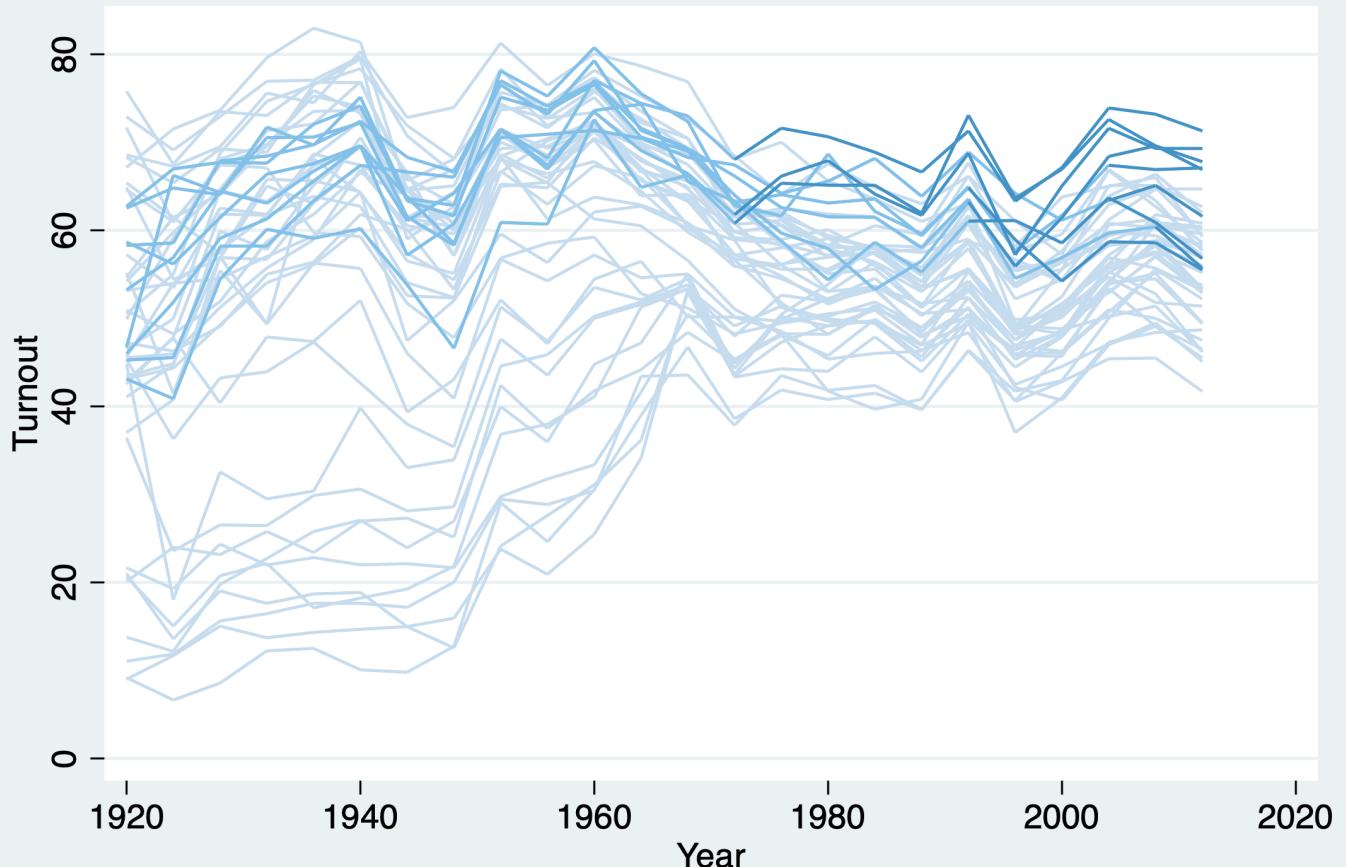
Distinguish the pre- and post-treatment periods for treated units:

```

*prepost
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(outcome)
xtitle("Year") ytitle("Turnout") title("EDR Reform and Turnout") prepost

```

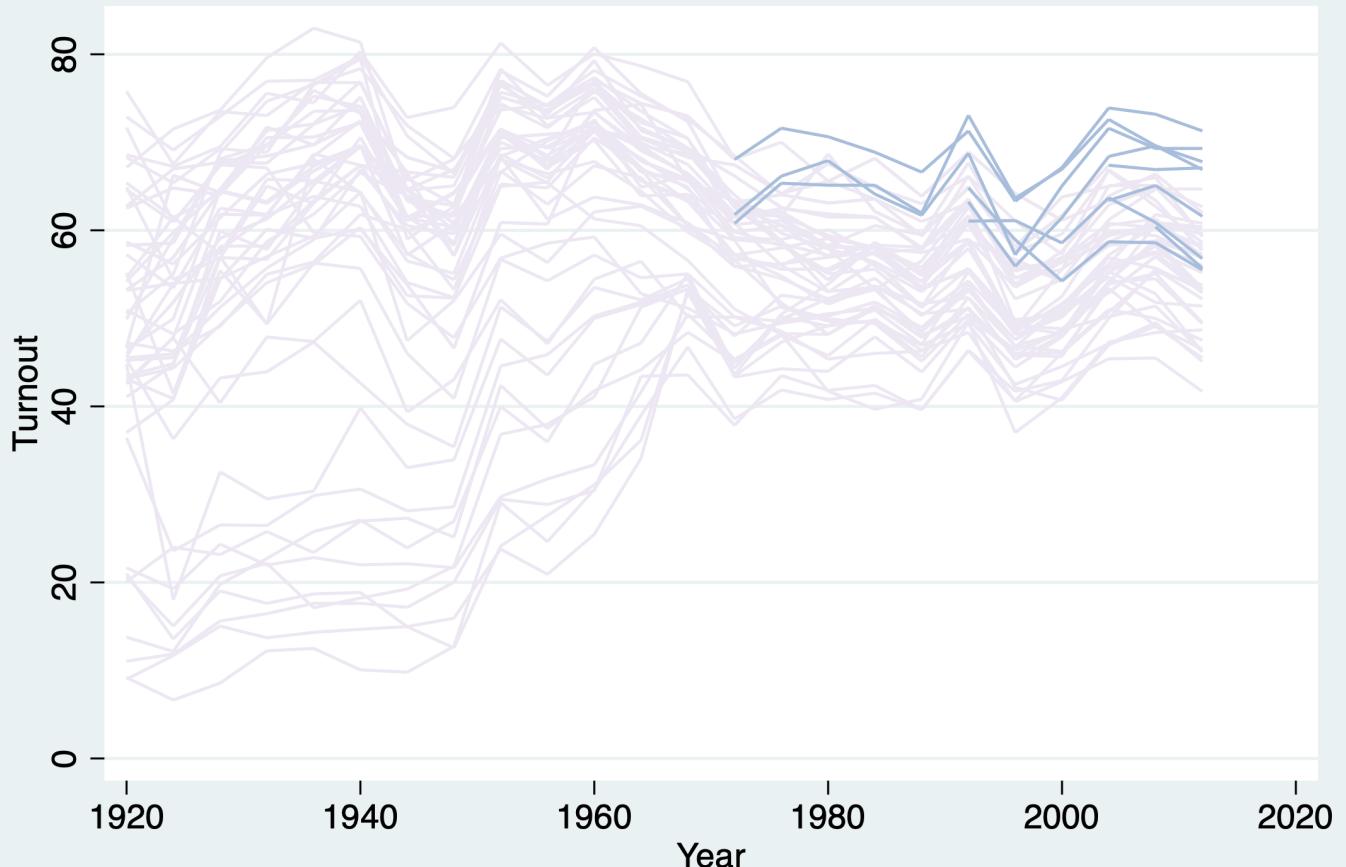
## EDR Reform and Turnout



Apply the light purple to blue theme by specifying `mycolor(PuBu)`:

```
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(outcome)
xtitle("Year") ytitle("Turnout") title("EDR Reform and Turnout") mycolor(PuBu)
```

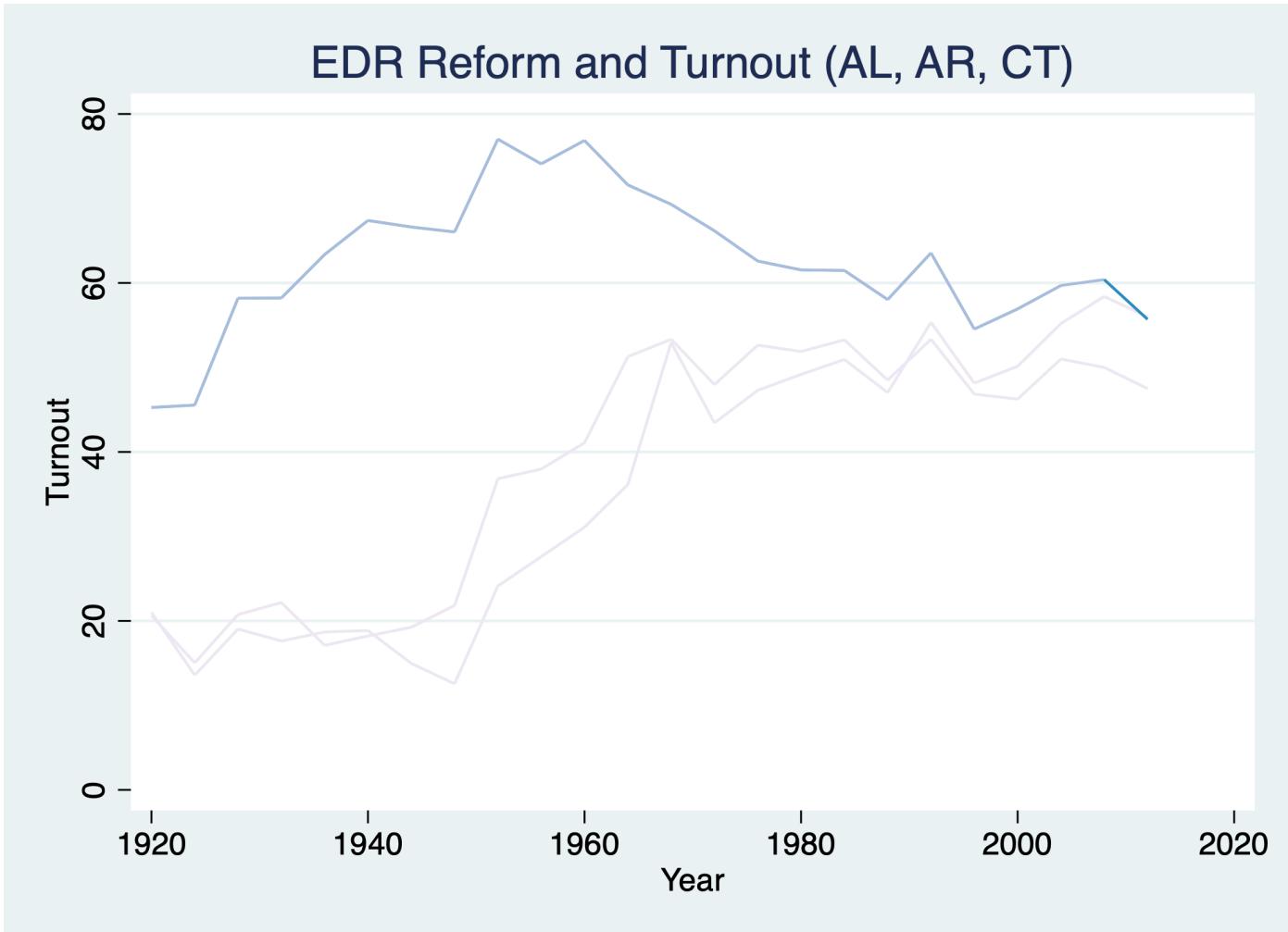
## EDR Reform and Turnout



### 5.2 Specify which unit(s) we want to take a look at

We can specify which unit(s) we want to take a look at:

```
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor if abb == 1|abb == 2|abb == 6,
i(abb) t(year) type(outcome) xtitle("Year") ytitle("Turnout") title("EDR Reform and
Turnout (AL, AR, CT)") mycolor(PuBu) prepost
```

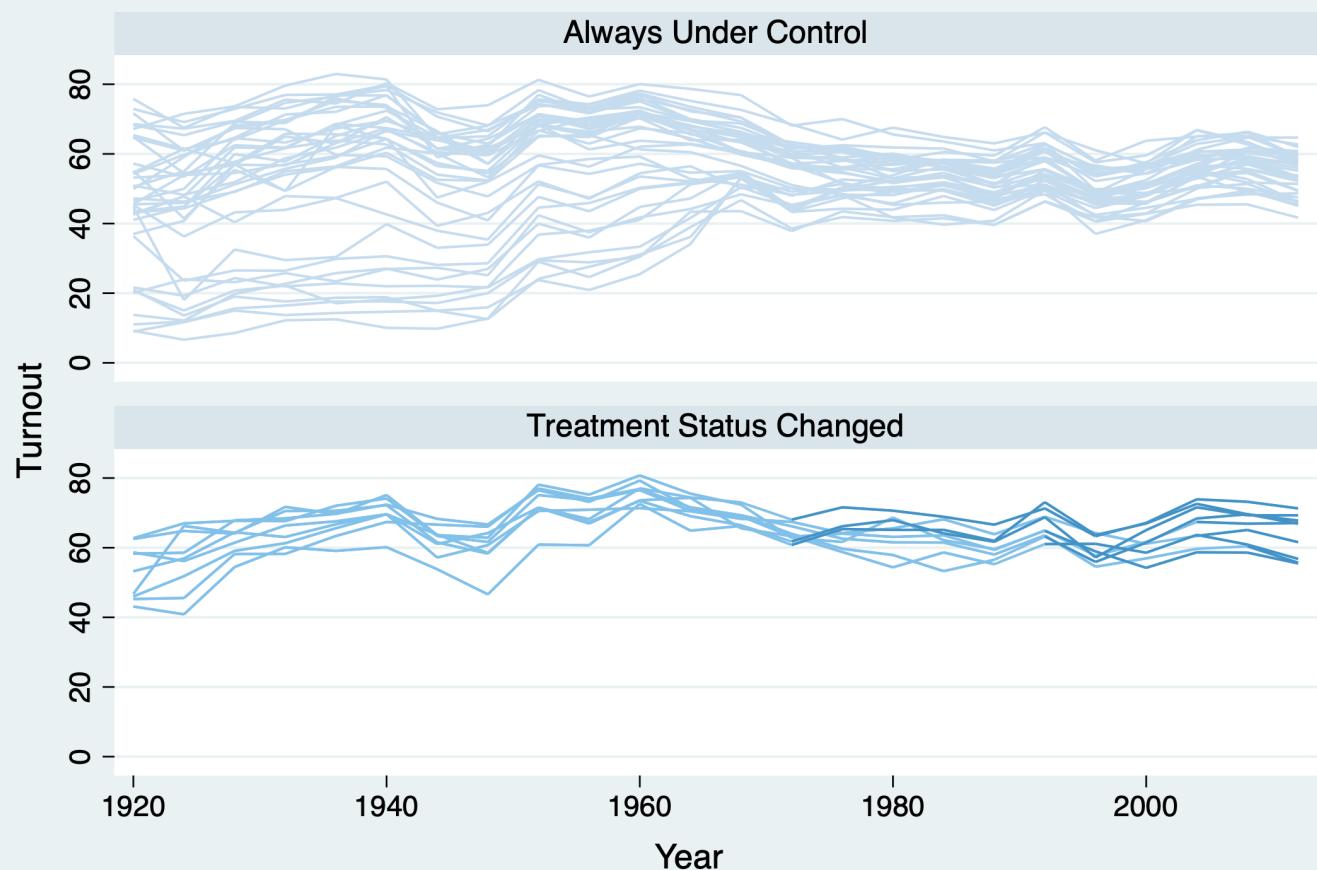


### 5.3 Put each unit into different groups, then plot respectively

To better understand the data, sometimes we want to plot the outcome based on whether the treatment status has changed during the observed time period. We can simply add options `prepost` and `bygroup`. The algorithm will analyze the data and automatically put each unit into different groups, e.g. (1) units always being treated, (2) units always under control, (3) units whose treatment status has changed.

```
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) type(outcome)
xtitle("Year") ytitle("Turnout") by(), title("EDR Reform and Turnout")) bygroup prepost
xlabel(1920 (20) 2000)
```

## EDR Reform and Turnout



## 6. Discrete Outcomes

We can accommodate discrete variables by setting `discreteoutcome`. Below is an example using the `simdata` dataset, in which the outcome variable takes three values: 0, 1, and 2.

### 6.1 Discrete outcomes

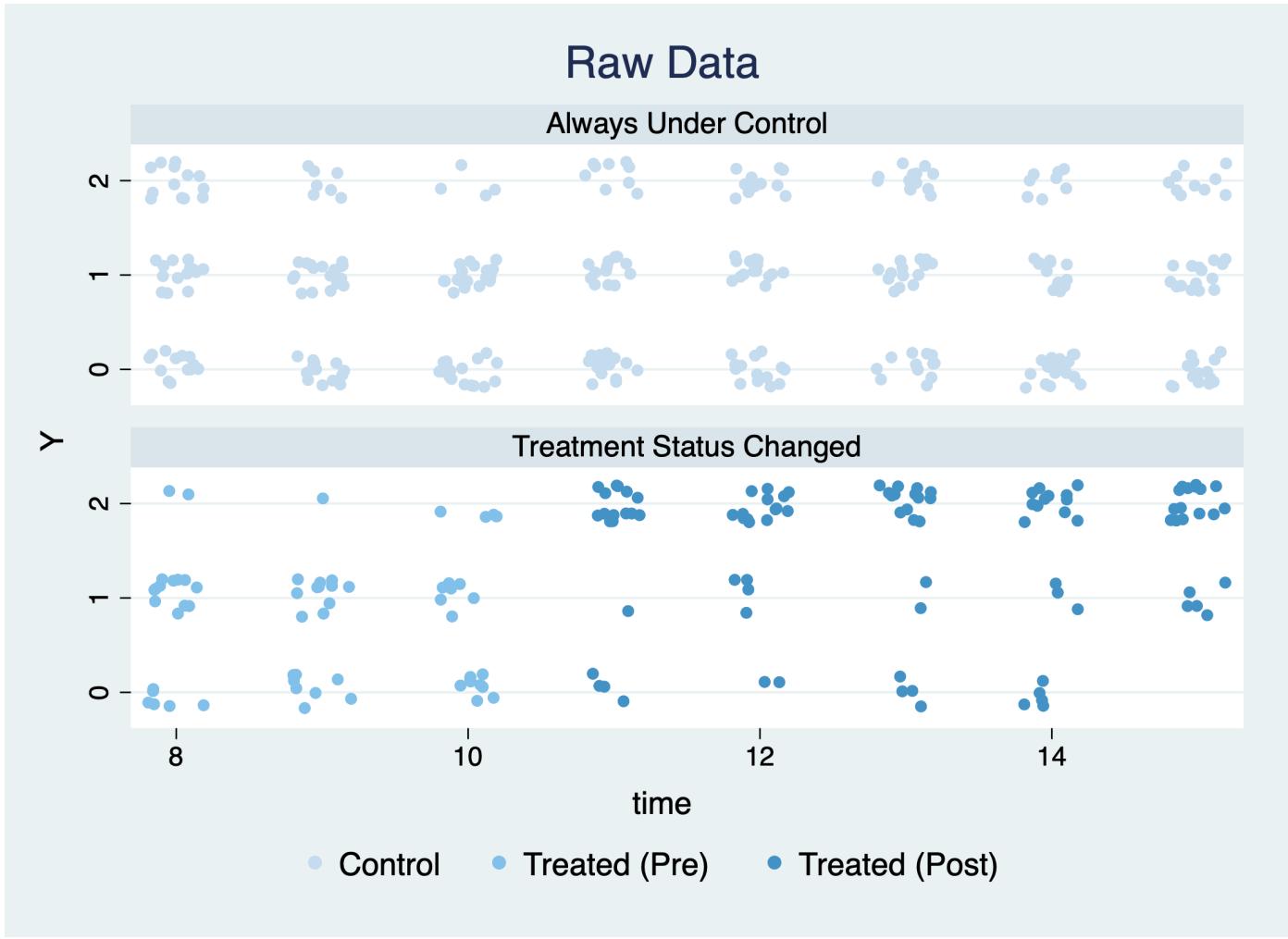
```
use simdata.dta, replace
panelView Y D if time >= 8 & time <= 15, type(outcome) i(id) t(time) mycolor(Reds)
discreteoutcome title("Raw Data") xlabel(8 (2) 15) ylabel(0 (1) 2)
```



## 6.2 Put each unit into different groups, then plot respectively

We split the sample based on changes in treatment status:

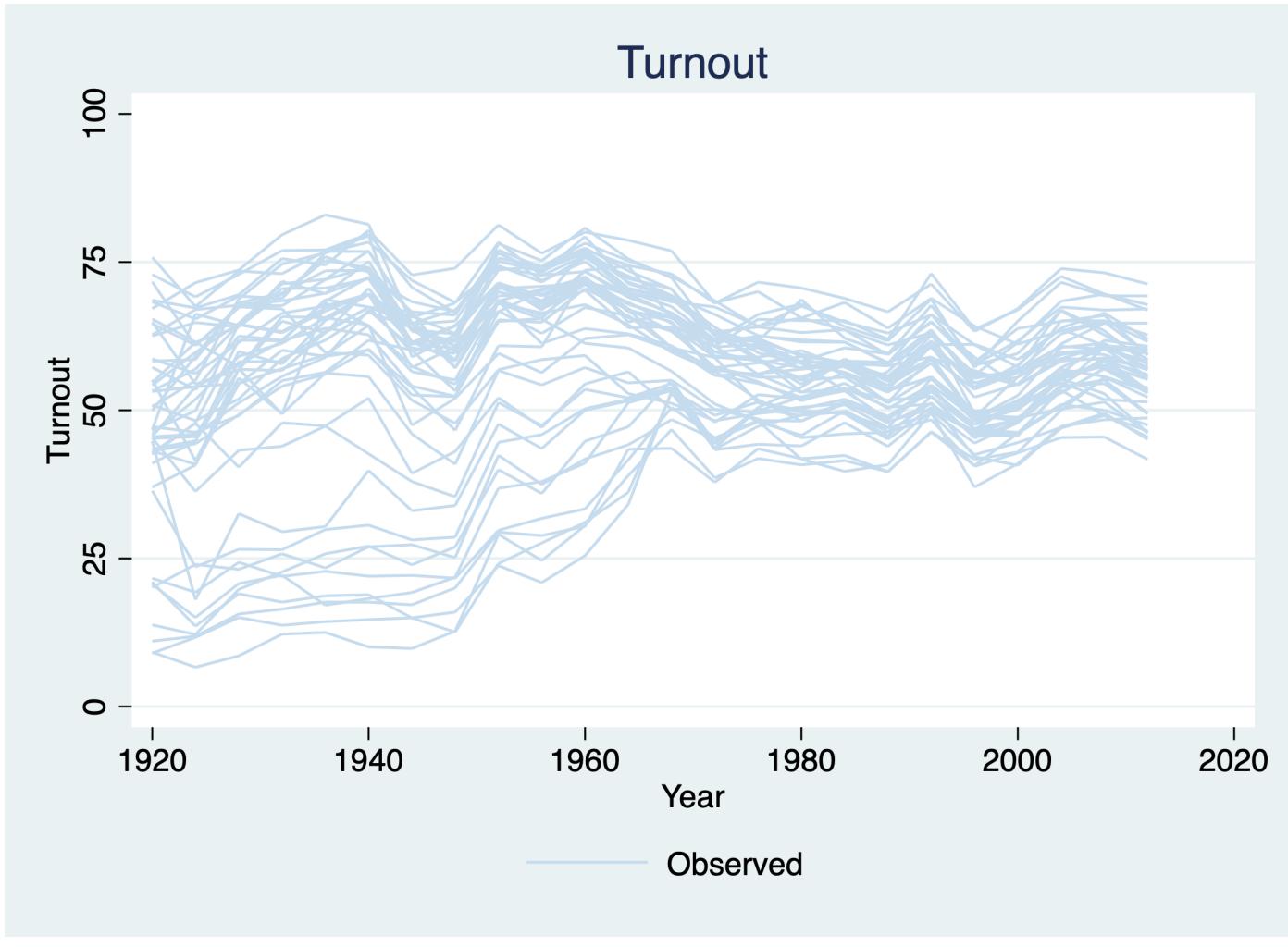
```
use simdata.dta, replace
panelView Y D if time >= 8 & time <= 15, type(outcome) i(id) t(time) discreteoutcome
by(,title("Raw Data")) xlabel(8 (2) 15) ylabel(0 (1) 2) bygroup prepost
```



## 7. Plotting Any Variable In Panel Dataset

Plot an outcome variable (or any variable) in a panel dataset by `type(outcome)` and `ignoretreat`:

```
use turnout.dta, clear
panelView turnout, i(abb) t(year) type(outcome) xtitle("Year") ytitle("Turnout")
title("Turnout") ylabel(0 (25) 100) ignoretreat
```



## 8. Plotting Y And D Time Series In One Graph

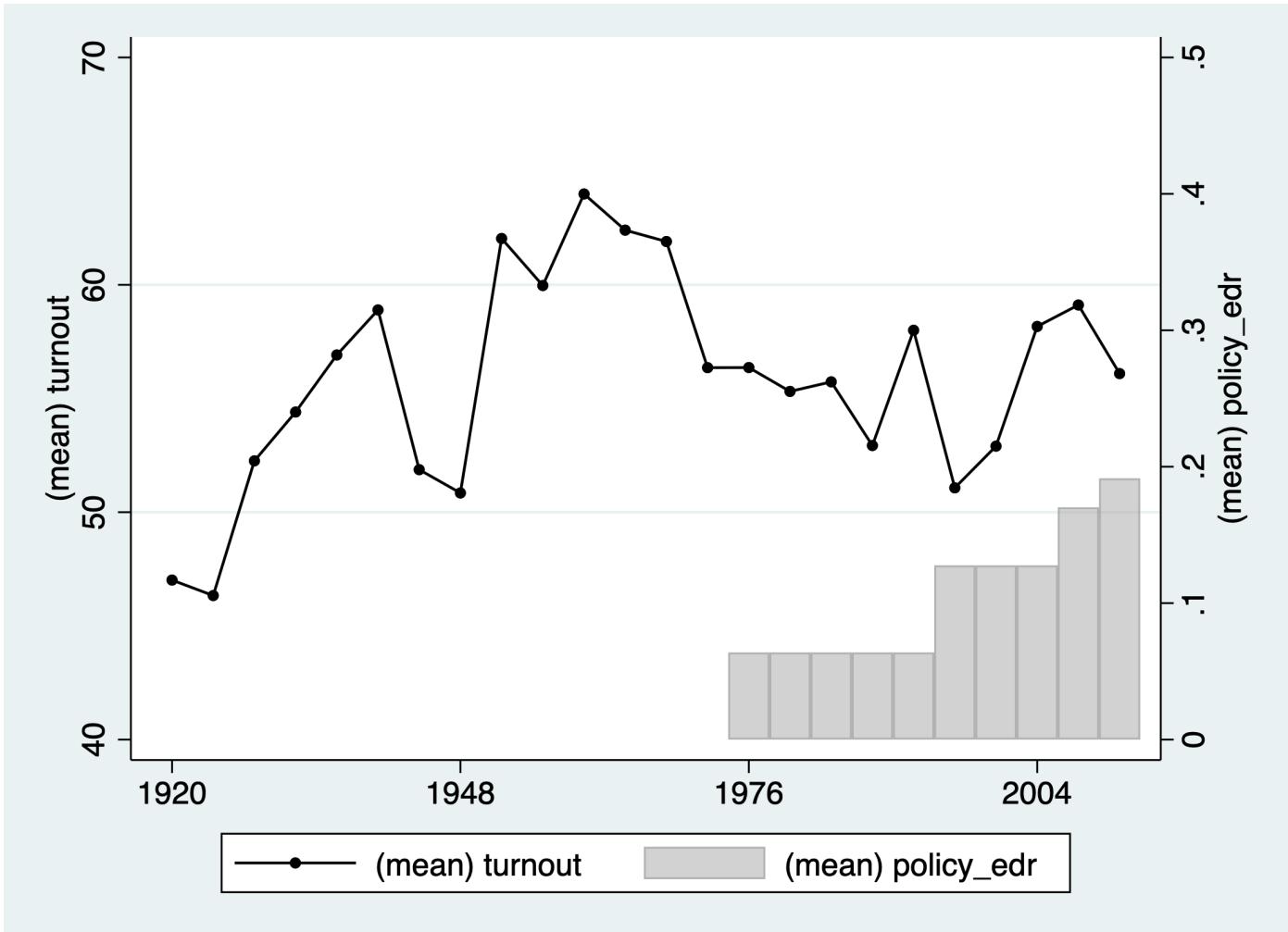
Visualize time series of the outcome and treatment in one figure by specifying `type(bivar)` or `type(bivariate)`. For continuous variable, we use line plot as default; for discrete variable, we use bar plot. To plot connected lines (`connected` or `c`), lines (`line` or `l`), or bars (`bar` or `b`) rather than the default, please add `style( , )`, where the first element defines the outcome style, and the second defines the treatment style.

### 8.1 Plot average time series for all units

This section plots mean D and Y against time in the same graph.

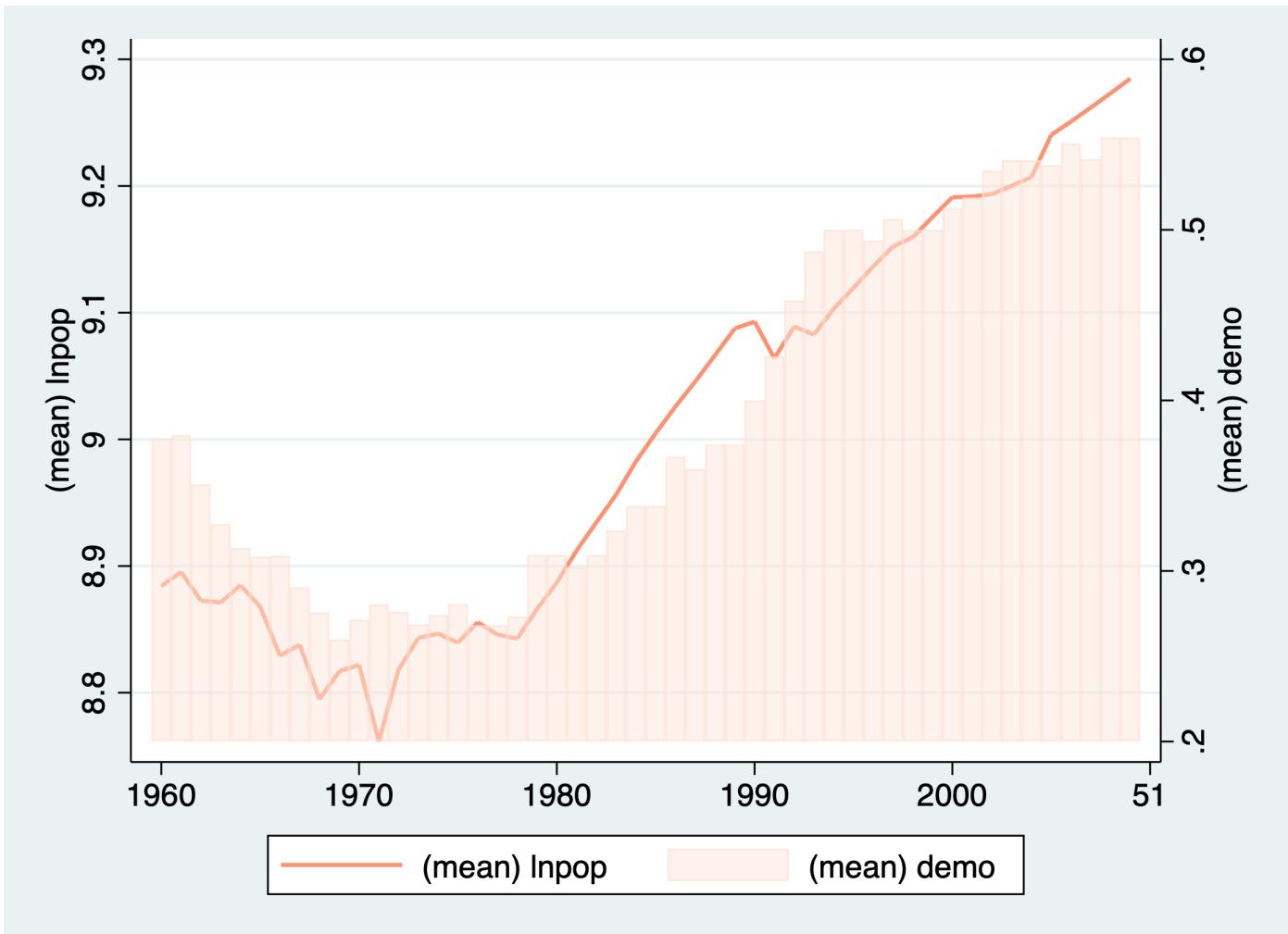
With continuous outcome and discrete treatment, here are two examples. In the former one, `style(c,b)` means connected scatter instead of default line plot for the outcome and bar plot for the treatment. If any connected line, we can specify the symbol size by `msize()`:

```
***** 1. Y: continuous; D: discrete *****
use turnout.dta, clear
*label the first and second y axes
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) xlabdist(7)
type(bivariate) ylabel(40 (10) 70) xlabel(0 (0.1) 0.5, axis(2)) msize(*0.5) style(c b)
```



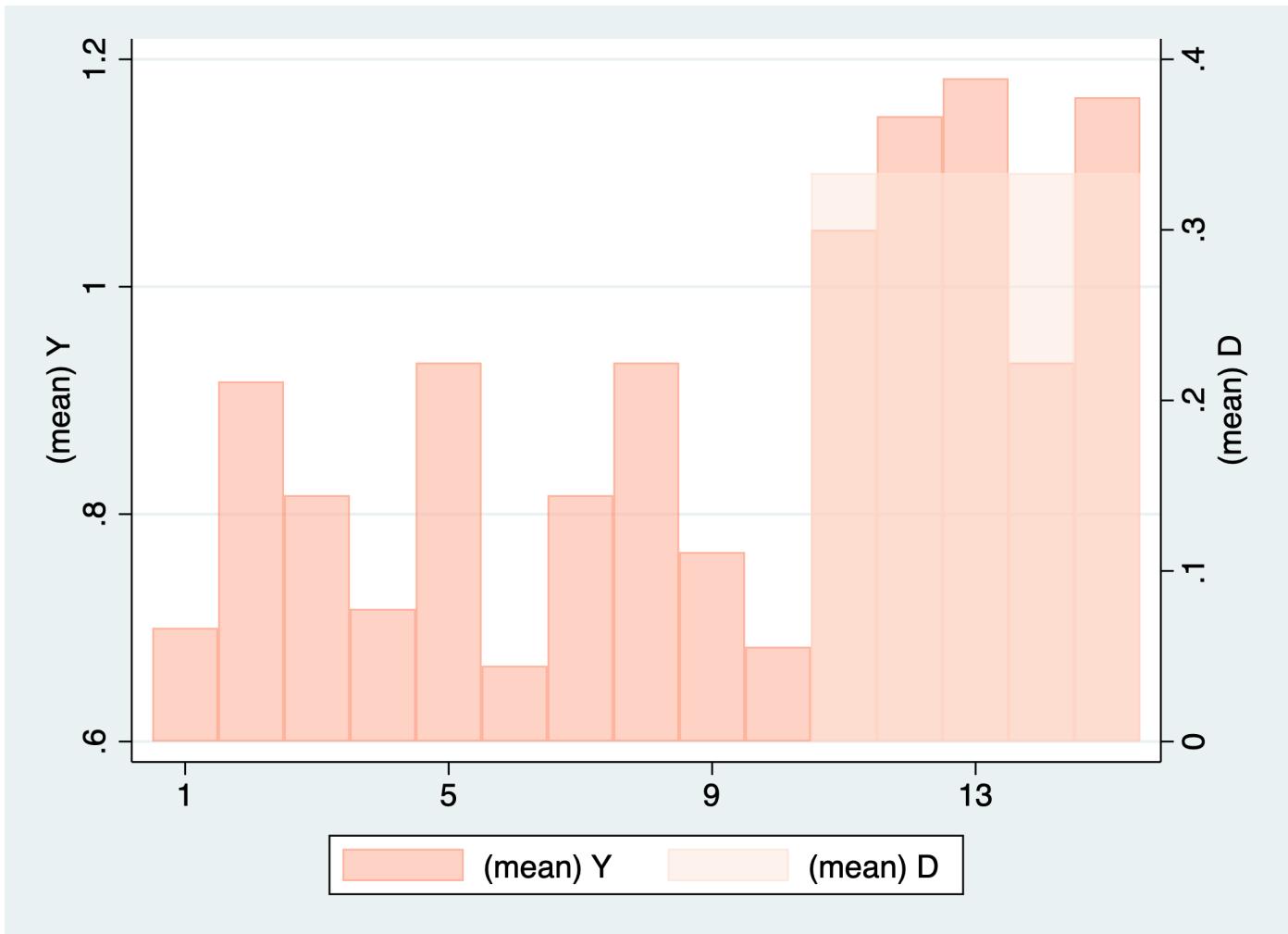
If not apply the default black and white theme, set option `mycolor()`. Besides, `lwd(medthick)` is to change the line width from the default `medium` to `medthick`:

```
use capacity.dta, clear
panelview lnpop demo, i(country) t(year) xlabdist(10) type(bivar) mycolor(Reds)
lwd(medthick)
```



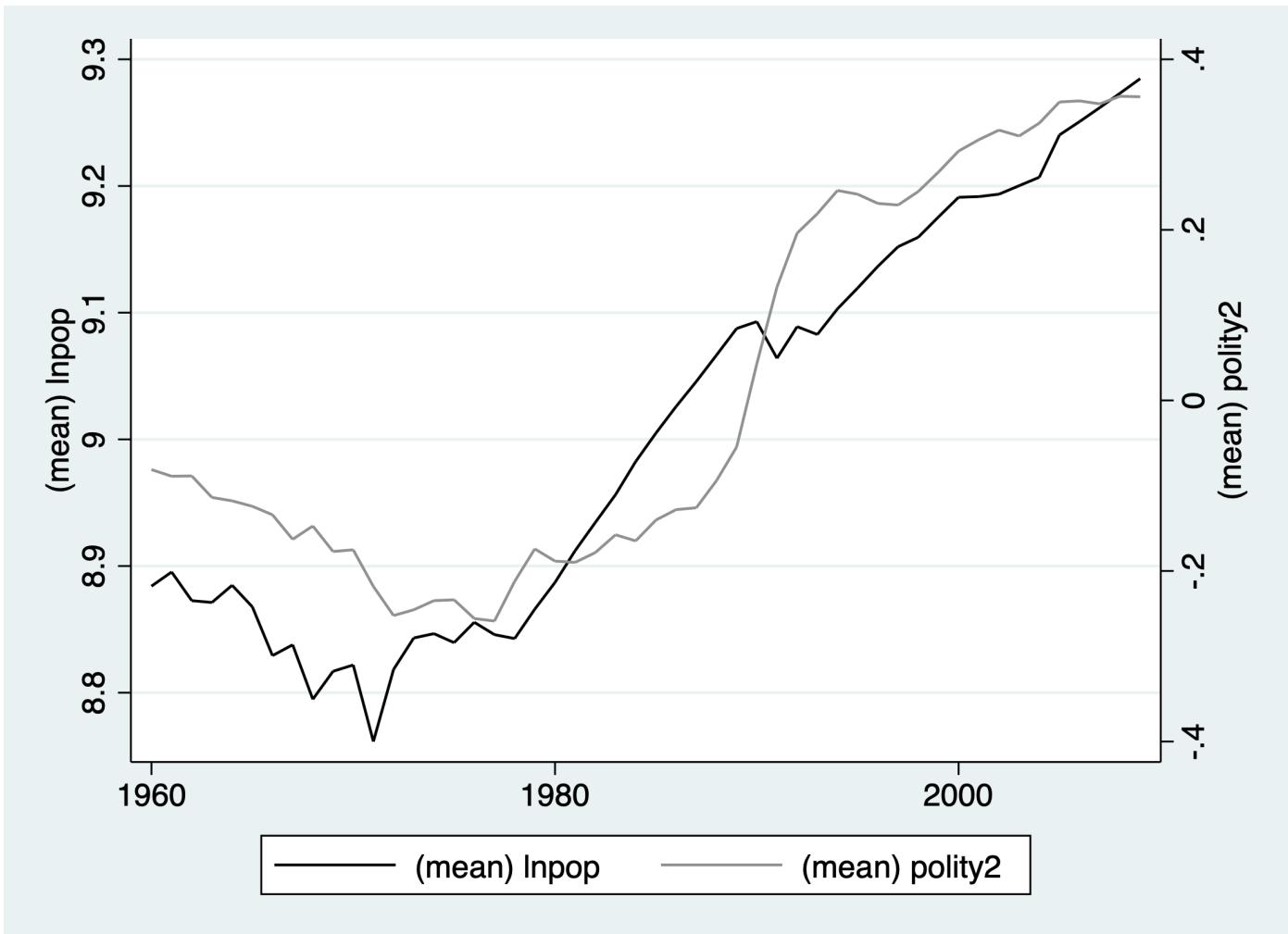
If the outcome is discrete, we can plot outcome and treatment against time in the same figure adding `discreteoutcome`:

```
***** 2. Y: discrete; D: discrete *****/
use simdata.dta, replace
panelView Y D,i(id) t(time) discreteoutcome xlabdist(4) type(bivar) mycolor(Reds)
```



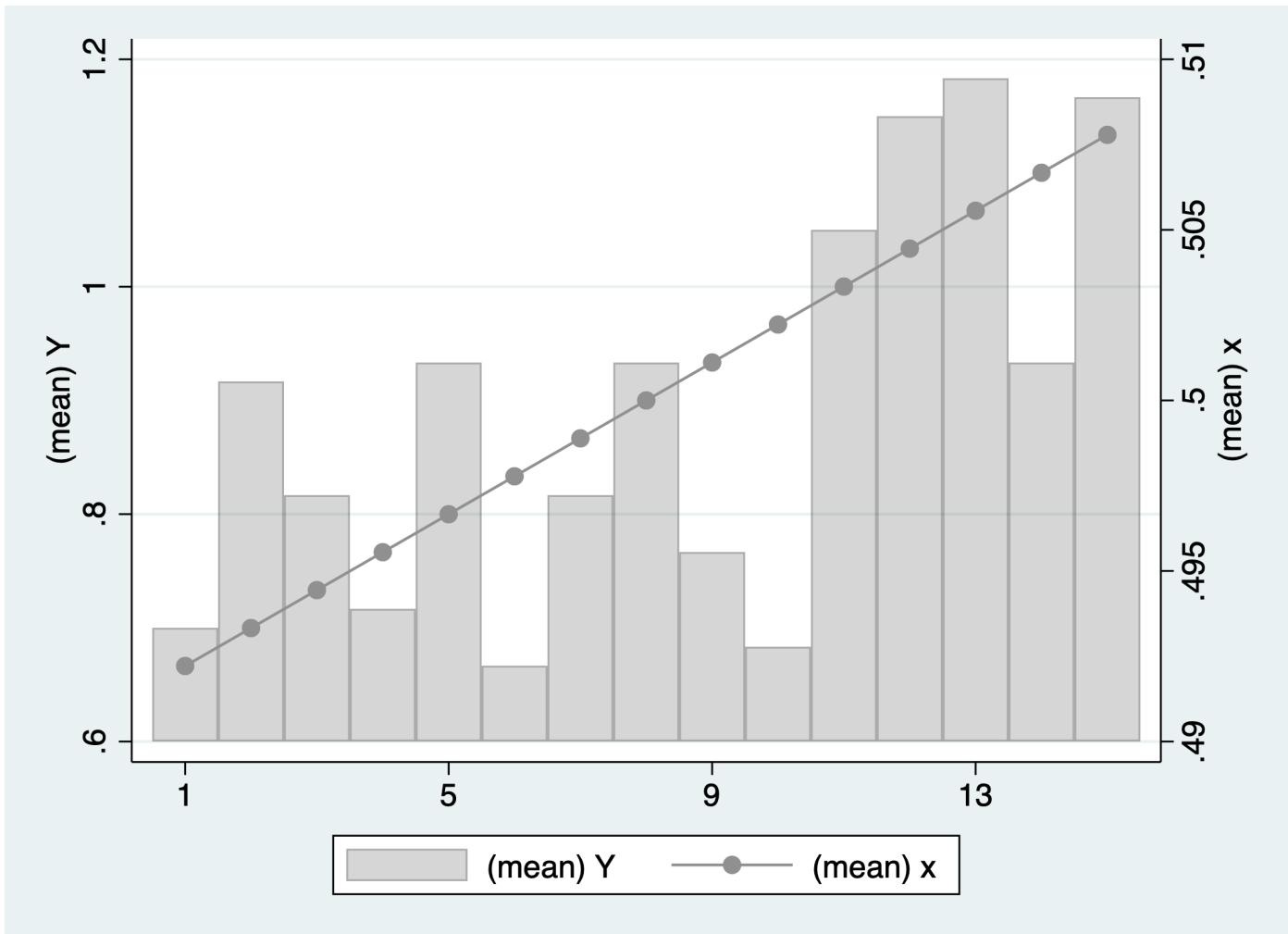
When treatment variable is continuous, we need to add the subcommands of `continuoustreat`:

```
***** 3. Y: continuous; D: continuous *****
use capacity.dta, clear
panelview lnpop polity2, i(country) t(year) continuoustreat xlabdist(20) type(bivar)
```



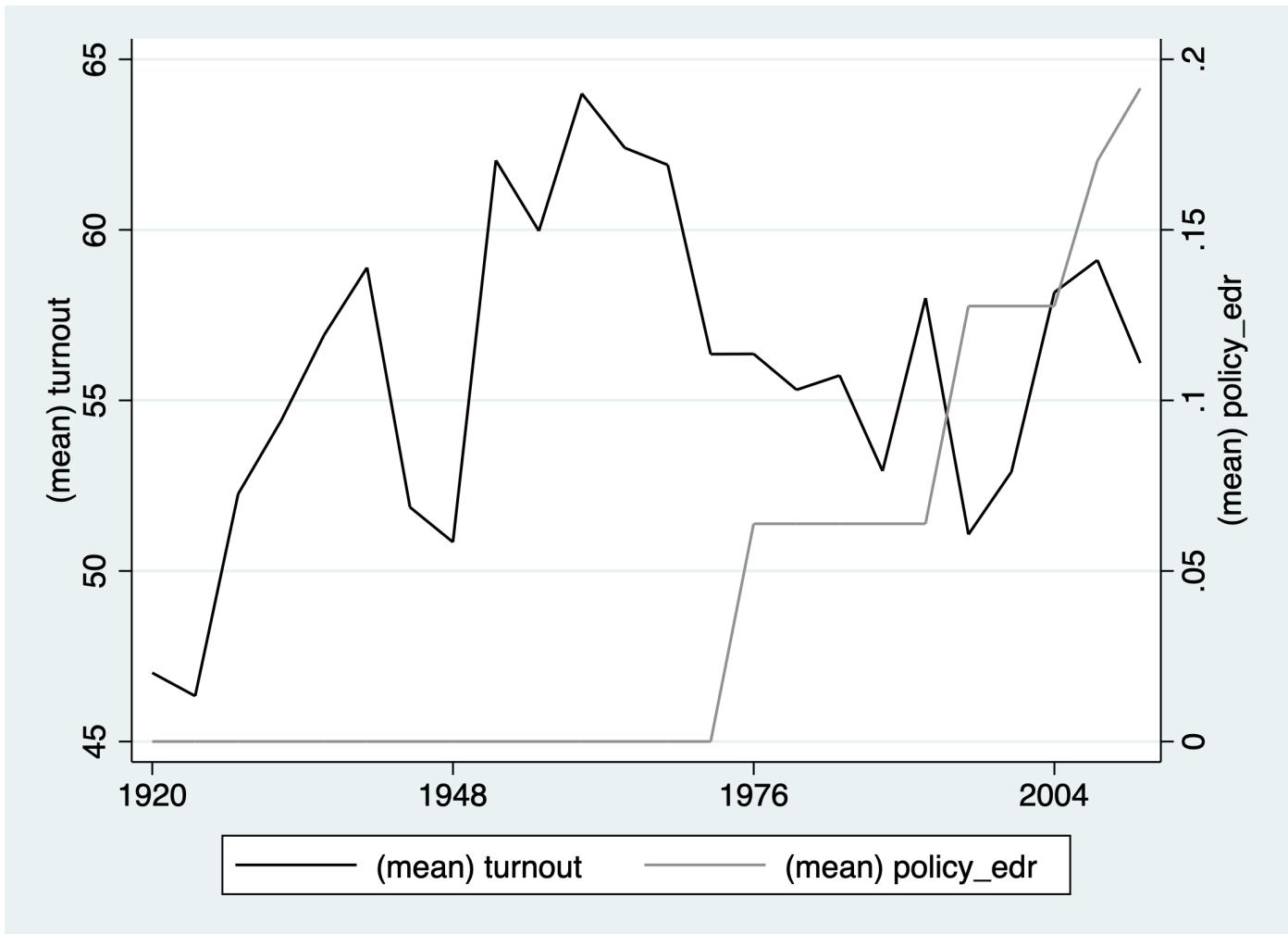
In the last situation, we plot discrete outcome and continuous treatment with options `continuoustreat` and `discreteoutcome`:

```
***** 4. Y: discrete; D: continuous *****
use simdata.dta, replace
range x 0 1
panelView Y x, i(id) t(time) continuoustreat discreteoutcome xlabdist(4) type(bivar)
style(b c)
```

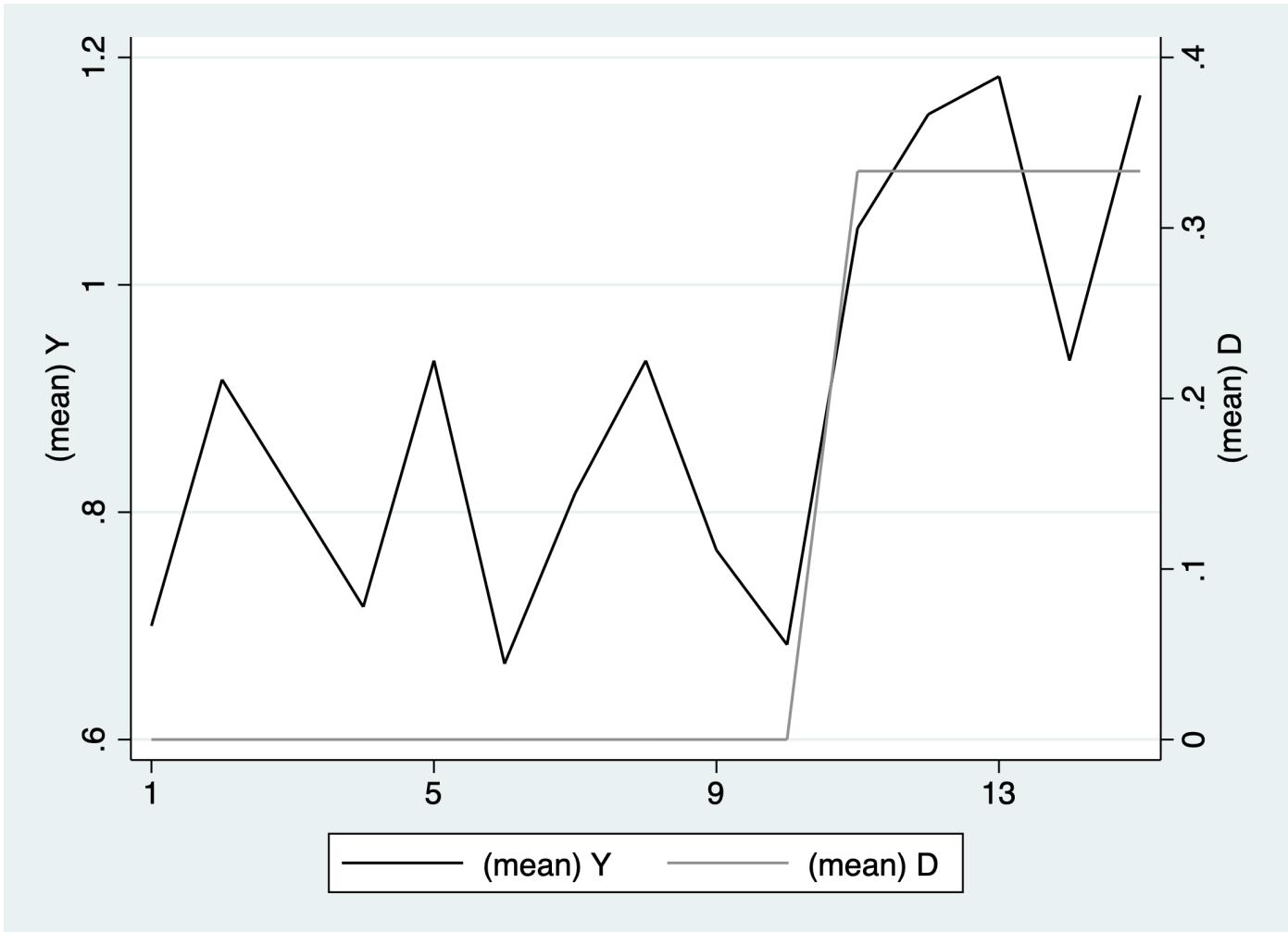


We can add `style(l,l)` or `style(line)` to plot lines instead of bars for treatment:

```
***** Line the discrete treatment *****/
* Y: continuous; D: discrete
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) xlabdist(7)
style(line) type(bivar)
```



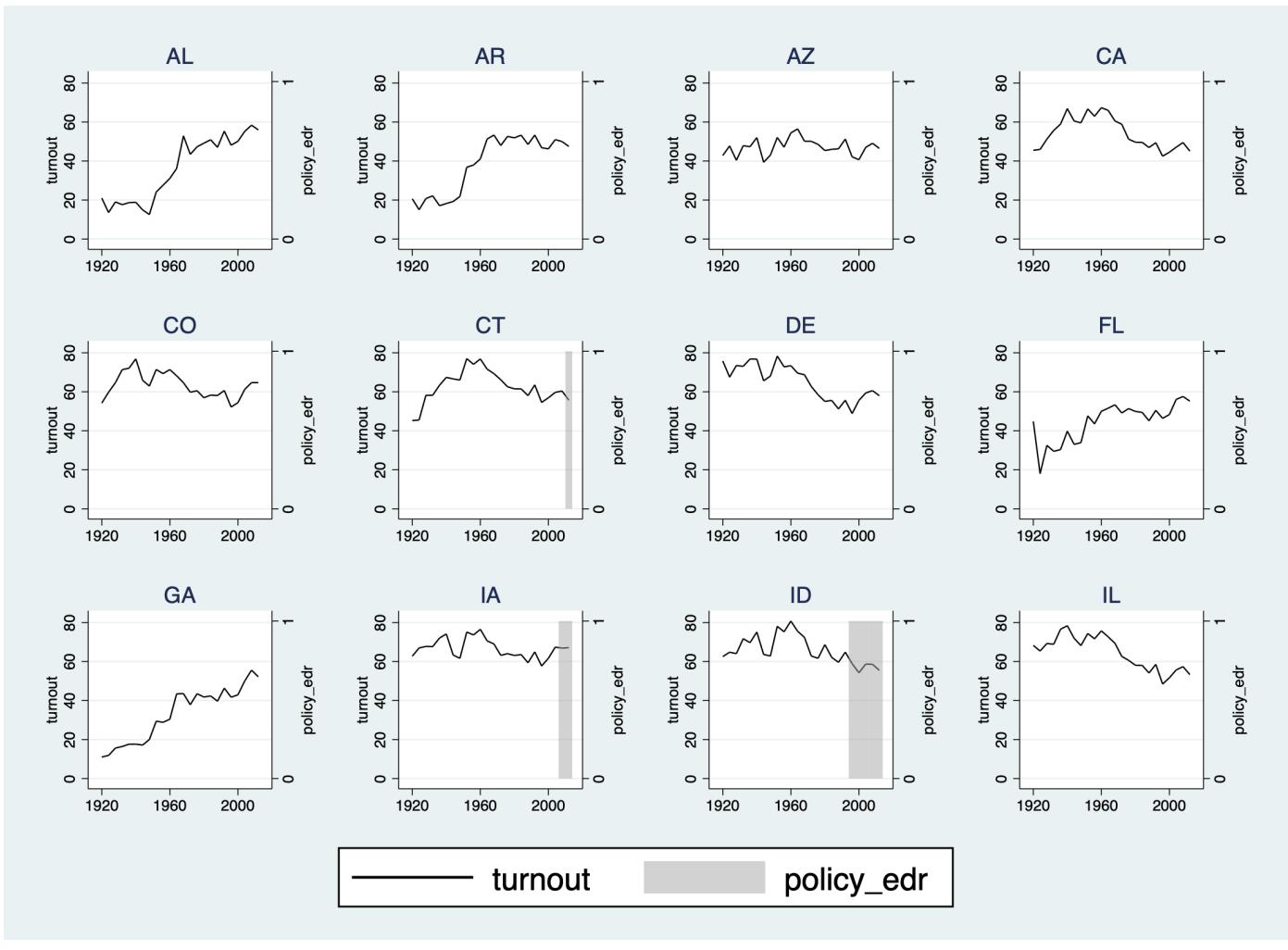
```
*Y: discrete; D: discrete  
use simdata.dta, replace  
panelView Y D,i(id) t(time) discreteoutcome xlabdist(4) style(line) type(bivar)
```



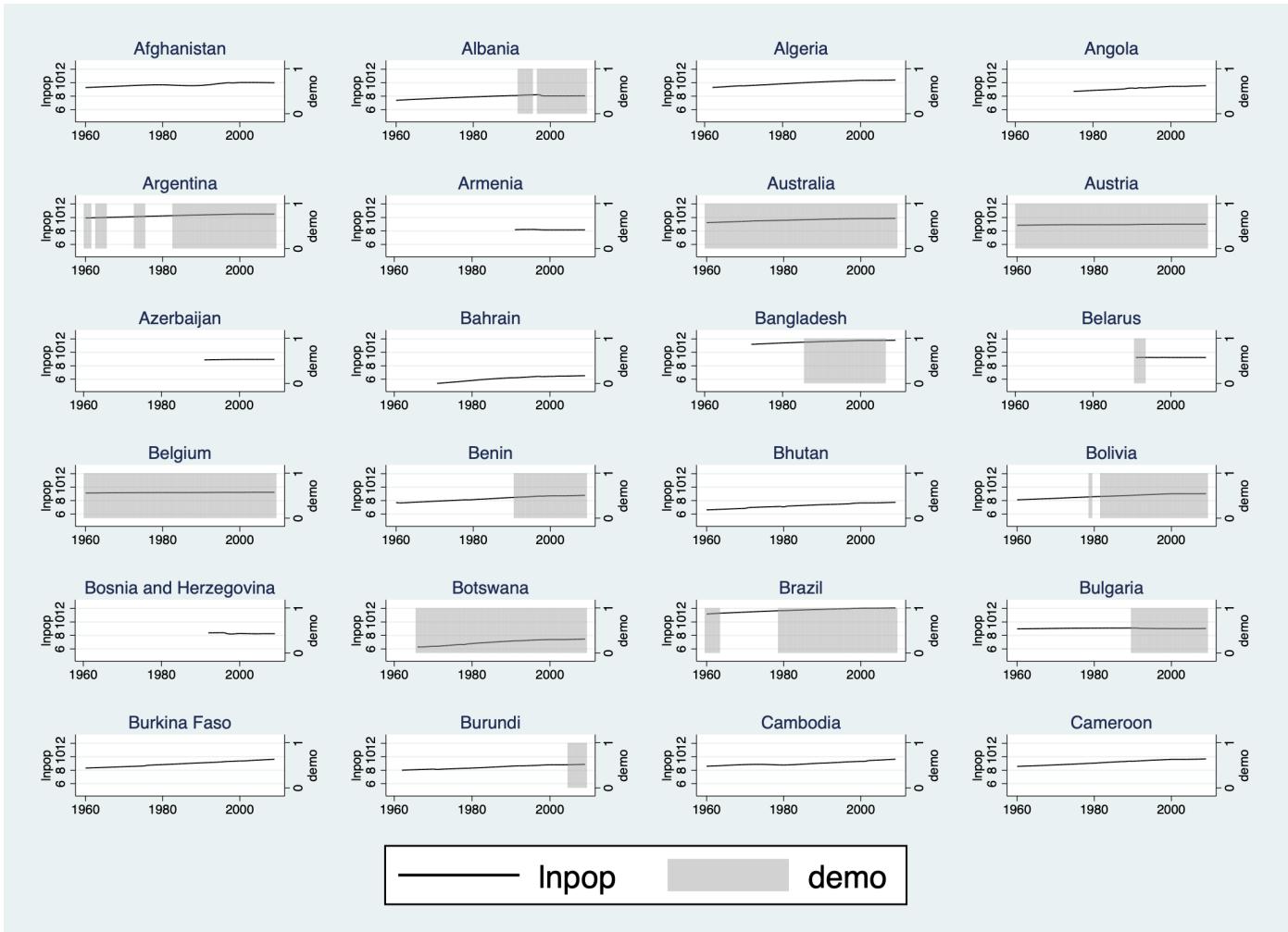
## 8.2 Plot by each unit

We plot D and Y against time by each unit by option `byunit`. Below are two examples with continuous outcome and discrete treatment variable. We arrange four subgraphs in one row:

```
***** 1. Y: continuous; D: discrete *****
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor if abb >= 1 & abb <= 12,
i(abb) t(year) xlabdist(10) type(bivar) byunit
```

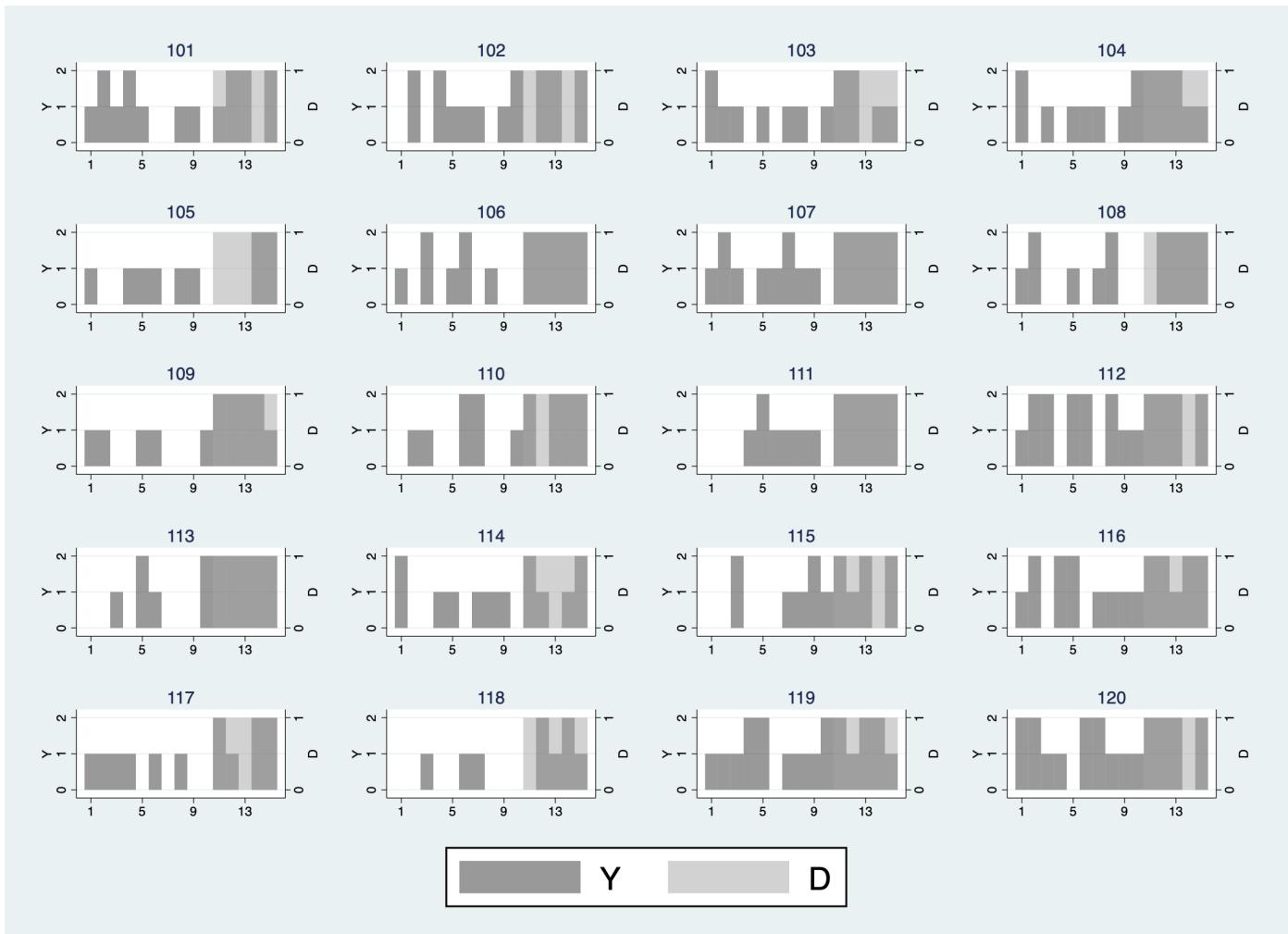


```
use capacity.dta, clear
panelView lnpop demo if country >= 1 & country <= 24, i(country) t(year) xlabdist(20)
type(bivar) byunit
```



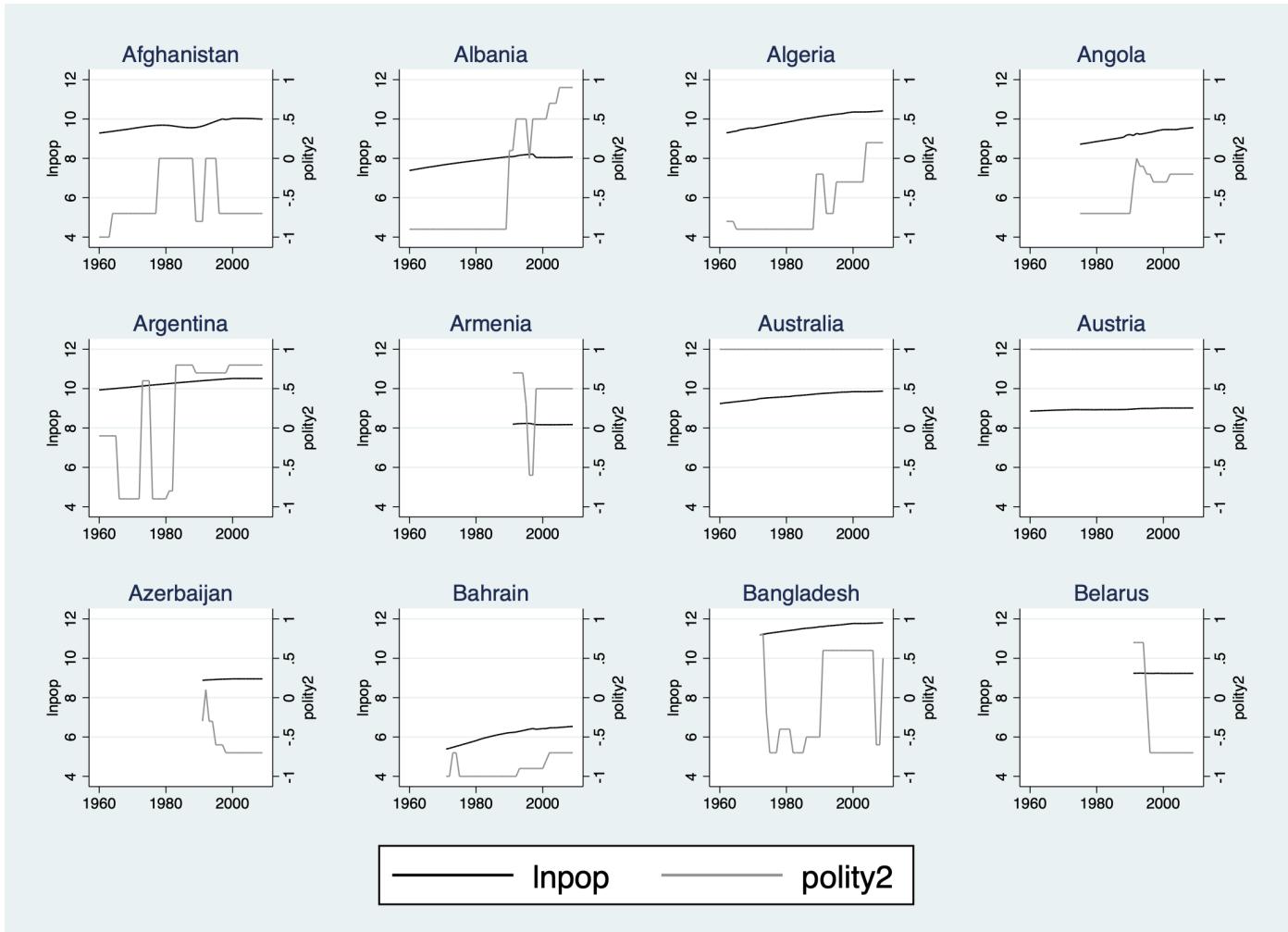
With discrete outcome and treatment:

```
***** 2. Y: discrete; D: discrete *****
use simdata.dta, replace
panelView Y D if id >= 101 & id <= 120,i(id) t(time) discreteoutcome xlabdist(4)
type(bivar) byunit
```



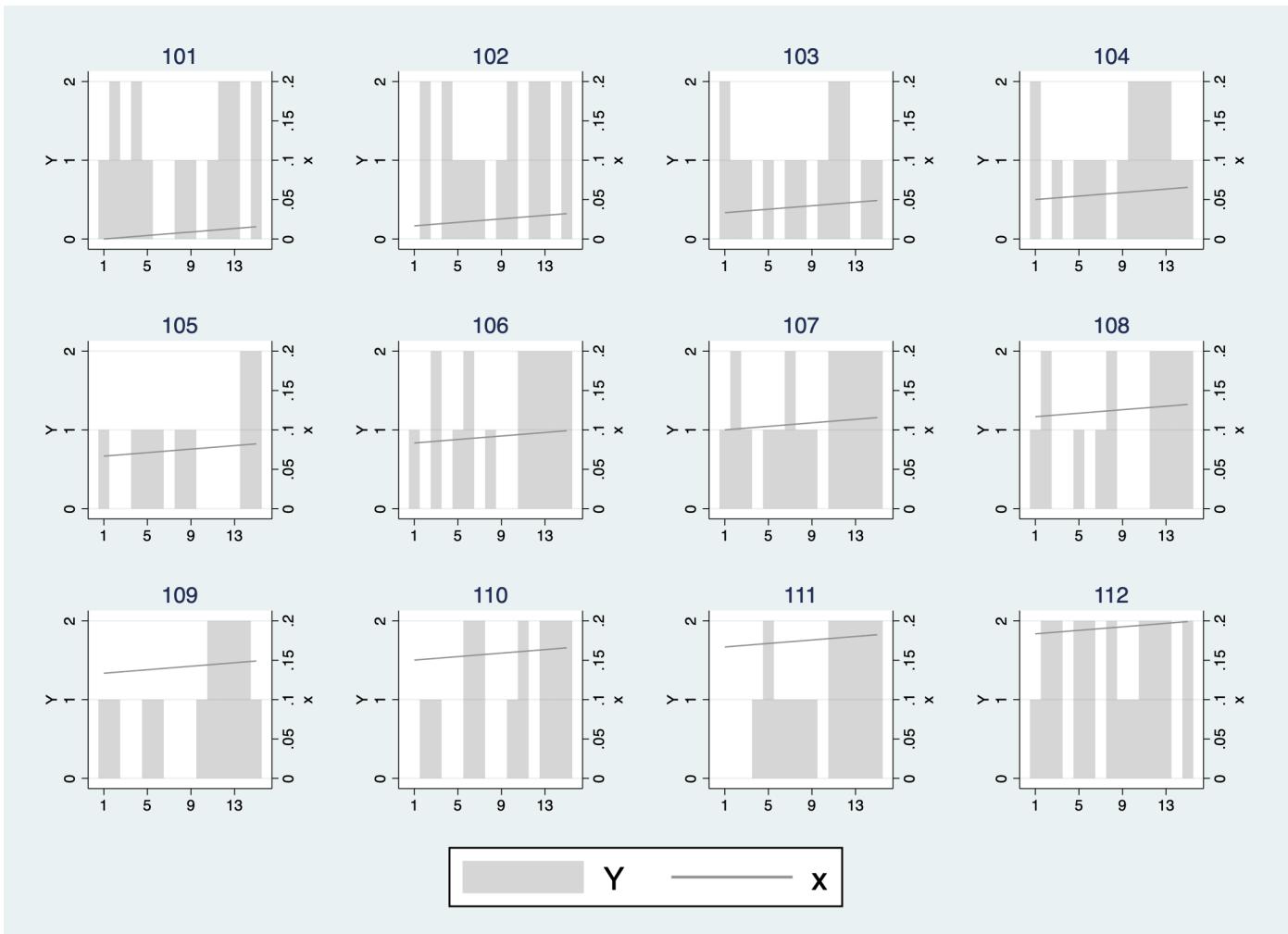
With continuous outcome and treatment:

```
***** 3. Y: continuous; D: continuous *****
use capacity.dta, clear
panelView lnpop polity2 if country >= 1 & country <= 12, i(country) t(year)
continuoustreat xlabdist(20) type(bivar) byunit
```



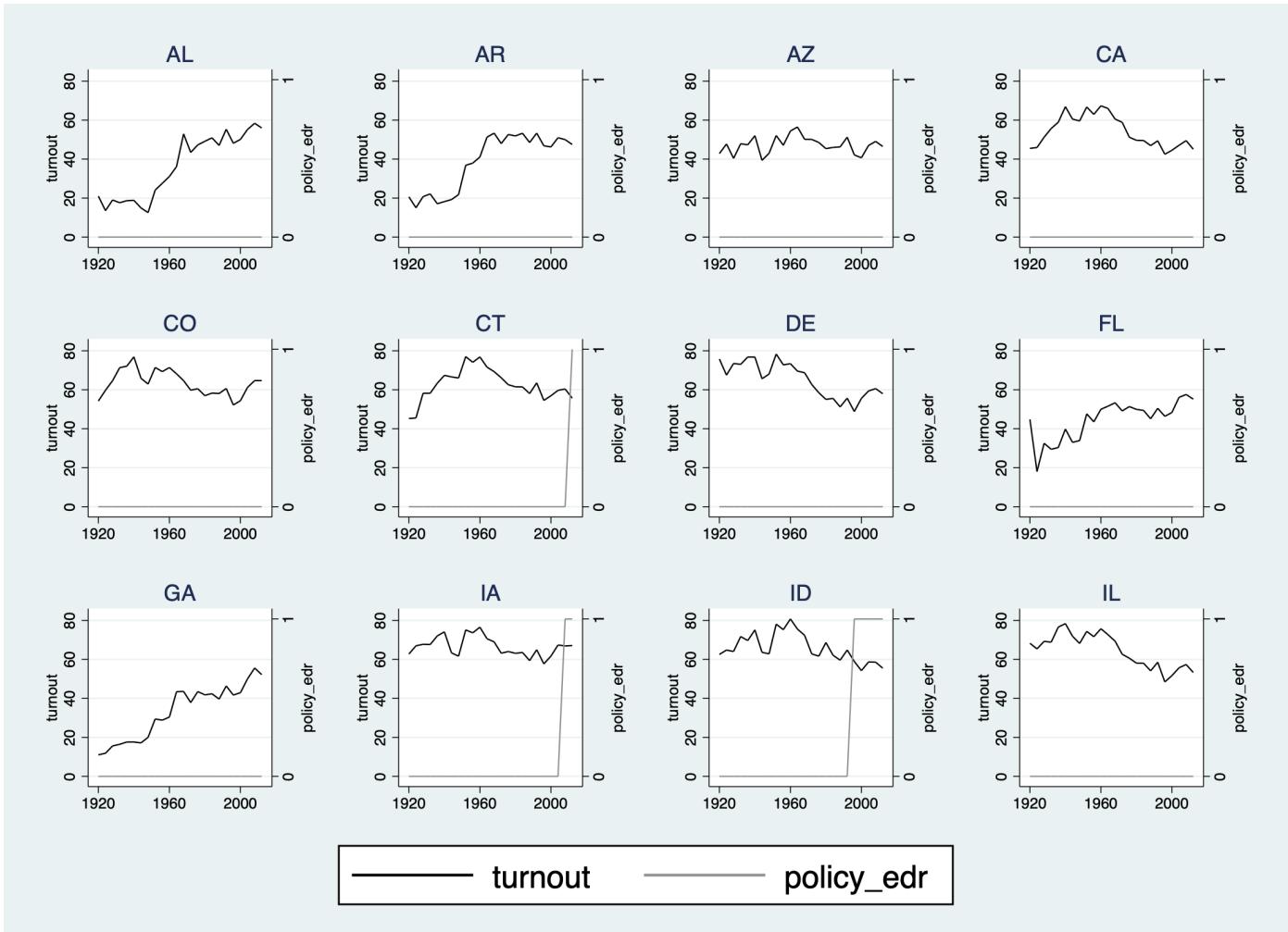
With discrete outcome and continuous treatment:

```
***** 4. Y: discrete; D: continuous *****/
use simdata.dta, replace
range x 0 1
panelView Y x if id >= 101 & id <= 112, i(id) t(time) continuoustreat discreteoutcome
xlabdist(4) type(bivar) byunit
```



To better visualize a discrete treatment whose value is sometimes zero, add `style(line)` to invoke line plots instead of bar plots:

```
***** Line the discrete treatment *****/
* Y: continuous; D: discrete
use turnout.dta, clear
panelView turnout policy_edr policy_mail_in policy_motor if abb >= 1 & abb <= 12,
i(abb) t(year) xlabdist(10) style(line) type(bivar) byunit
```



```
*Y: discrete; D: discrete
use simdata.dta, replace
panelView Y D if id >= 101 & id <= 120,i(id) t(time) discreteoutcome xlabdist(4)
style(l) type(bivar) byunit
```

