Journal of Statistical Software

September 2023, Volume 107, Issue 7. *doi: 10.18637/jss.v107.i07*

Panel Data Visualization in R (panelView) and Stata (panelview)

Hongyu Mou
University of California,
Los Angeles

Licheng Liu
Massachusetts Institute
of Technology

Yiqing Xu
Stanford University

Abstract

We develop an R package **panelView** and a Stata package **panelview** for panel data visualization. They are designed to assist causal analysis with panel data and have three main functionalities: (1) They plot the treatment status and missing values in a panel dataset; (2) they visualize the temporal dynamics of the main variables of interest; and (3) they depict the bivariate relationships between a treatment variable and an outcome variable either by unit or in aggregate. These tools can help researchers better understand their panel datasets before conducting statistical analysis.

Keywords: panel data, visualization, treatment status, missing data, R, Stata.

1. Introduction

In the social and biomedical sciences, observational panel data are commonly used to determine the relationship between variables of interest. This type of data enables researchers to take advantage of both cross-sectional and temporal variations to account for unobserved confounding factors, thus enhancing the credibility of their causal inferences. Popular methods for analyzing panel data include the difference-in-differences (e.g., Card and Krueger 1994), fixed effects and random effects models (e.g., Wooldridge 2001; Hsiao 2003), the synthetic control method (Abadie and Gardeazabal 2003; Abadie, Diamond, and Hainmueller 2010), and marginal structural models (e.g., Robins, Hernán, and Brumback 2000), among others.

However, it is not common practice for researchers to plot their raw data before conducting a panel data analysis. Such visualization is rarely seen in published papers or their accompanying materials. Neglecting to do so can have serious consequences: First, researchers may apply the wrong methods to their data. For example, recent research has demonstrated that two-way fixed effects may not provide causally interpretable results when the treatment effects are heterogeneous and the adoption of treatment occurs gradually (referred to as “staggered adoption”) (e.g., de Chaisemartin and D’Haultfœuille 2020; Goodman-Bacon 2021). Hence, it is essential for researchers to understand whether the treatment starts at the same time for all treated units or is adopted gradually. Second, without visualizing the data, researchers may lack sufficient knowledge about the key sources of variation in their data. This is especially critical when a large number of fixed effects are added to the model, absorbing a significant portion of the variations. Third, researchers may miss anomalies, such as outliers and missing values, in their data, leading to incorrect statistical analysis and invalid inferences.

We believe that one reason why analysts do not often plot their raw panel data prior to analysis is the lack of proper tools available on popular data analytical platforms such as R (R Core Team 2023) and Stata (StataCorp 2021). On the Comprehensive R Archive Network (CRAN; `https://CRAN.R-project.org/`), the most popular packages for panel data visualization are **panelr** (Long 2023) and **ExPanDaR** (Gassen 2020). These packages allow users to plot trends for a single variable, perform several widely used statistical routines, and interactively explore panels using histograms, scatterplots, and bar plots. Moreover, **PanelMatch** (Kim, Rauh, Wang, and Imai 2022) allows users to visualize binary treatment status. On Stata, there are a few ways to visualize longitudinal data, including the official command **xtline** (Cox 2009) and the user-written package **profileplot** (Ender 2014). Both methods enable users to display the temporal dynamics of variables, making them helpful for data exploration. However, they are not specifically designed to assist researchers in understanding patterns of treatment adoption/intensity or exploring relationships between multiple relevant variables.

To fill this gap, we develop **panelView** for R (Mou, Liu, and Xu 2023) and **panelview** for Stata (Mou and Xu 2023). These two packages offer three key capabilities. First, they enable users to plot the treatment status and missing values in their panel data, providing a deeper understanding of the sources of variation in the treatment and whether a particular estimation strategy is feasible. This functionality is similar to `DisplayTreatment` in **PanelMatch**, but our packages offer greater customization options and support both continuous and discrete treatments in addition to binary treatments. Second, they allow users to visualize the temporal dynamics of the variables of interest, using color to indicate treatment status and improving upon the capabilities of **ExPanDaR**, **xtline** and **profileplot**. Third, they can display the bivariate relationships between a treatment variable and an outcome variable, either for individual units or aggregated, providing a basis for any future statistical modeling the analyst may undertake.

panelView is available on CRAN at `https://CRAN.R-project.org/package=panelView` and **panelview** is distributed through the Statistical Software Components (SSC) at `https://ideas.repec.org/c/boc/bocode/s459034.html`. To demonstrate the functionality of both packages, we use three sample datasets: `capacity` (Wang and Xu 2018), `simdata`, and `turnout` (Xu 2017). These datasets will be automatically downloaded once **panelView** or **panelview** is installed. To aid in the reproducibility of the examples discussed in this article, we provide replication codes in both R and Stata.

The structure of the article is as follows: In the next section, we provide details on the installation process and the three datasets used for illustration. In Section 3, we outline the key syntax of both **panelView** and **panelview**. In Sections 4 to 6, we present a detailed discussion of the three main functionalities of the packages. The last section concludes.

2. Installation

Researchers can install the **panelView** package from CRAN via

```
R> install.packages("panelView")
```

Additionally, the latest development version is available on GitHub at https://github.com/xuyiqing/panelView. **panelView** depends on the following packages, which will be installed automatically along with **panelView**: **ggplot2**, **gridExtra**, **grid**, and **dplyr**.

To install **panelview** in Stata, users need to first install several dependencies:

```
net install grc1leg, from(http://www.stata.com/users/vwiggins) replace
net install gr0075, from(http://www.stata-journal.com/software/sj18-4) replace
ssc install labutil, replace
ssc install sencode, replace
```

Then install **panelview** from the SSC archive via

```
ssc install panelview, all replace
```

Three datasets associated with the package, `turnout.dta`, `capacity.dta`, and `simdata.dta`, will be downloaded to the default Stata folder when the `all` option is added.

Data for illustration. We illustrate both packages using three datasets: `capacity`, `simdata`, and `turnout`. The `capacity` data includes 7,186 country-year observations. Wang and Xu (2018) use it to study the impact of democracy (the treatment) on state capacity (the outcome). The dataset consists of 8 variables for each country:

- `country`: Country.
- `ccode`: Country code.
- `year`: Time indicator.
- `Capacity`: A continuous outcome variable indicating state capacity.
- `demo`: A binary indicator for treatment status (1 if democracy, 0 if control).
- `lnpop`: Logged population.
- `lngdp`: Logged gross domestic product (GDP) per capita.
- `polity2`: Continuous Polity score.

The `simdata` data is a simulated dataset that includes 60 observations. The dataset includes four variables:

- `id`: Unit indicator.
- `time`: Time indicator.

- `Y`: A discrete outcome with three levels: $0, 1$ and 2.
- `D`: A binary treatment variable.

The `turnout` dataset contains voter turnout information for 47 US states in general elections from 1920 to 2012 and is used to study the effect of Election Day Registration (EDR) on voter turnout (Xu 2017). The data includes six variables:

- `abb`: State identifier.
- `year`: Time indicator.
- `turnout`: A continuous outcome variable indicating voter turnout rate.
- `policy_edr`: A binary treatment variable indicating effective EDR laws.
- `policy_mail_in`: A binary indicator for universal mail-in registration.
- `policy_motor`: A binary indicator for motor voter registration.

3. Summary of syntax

The main function name of the R package **panelView** is `panelview` (with a lowercase "v"). The command name of the Stata package is also `panelview`. Throughout the paper, we will use `panelview` to refer to both the R function and the Stata command. To use `panelview`, the panel data must have a unit indicator and a time (period) indicator and be in long form, meaning each row represents one unit for a given period. Additionally, each observation must be uniquely identified by the unit and time indicators, meaning that there should be no more than one observation for each *unit* × *period* combination.

Table 1 summarizes the syntax of `panelview`. Please note that the variables Y, D, and `X` in Table 1 are just labels and can refer to any variables in a panel dataset. There are four types of plots available in `panelview`: `treat`, `missing`, `outcome`, and `bivariate`.

In a `treat` plot, `panelview` displays the treatment status and missing values of the treatment variable, though missingness may also be influenced by other variables in the formula. One major difference between the R and Stata syntax is that R uses the $\sim$ symbol to distinguish between the left-hand-side (LHS) and right-hand-side (RHS) variables in a formula, while Stata makes this distinction based solely on the order of the variables. As a result, `panelview` (R) visualizes the treatment status of the first RHS variable, while `panelview` (Stata) visualizes the treatment status of the second variable in the variable list, unless the option `ignoreY` is on. This option is automatically activated when there is only one variable in the variable list. Control variables `X` will only affect the missing value status of each observation in the plot. Unlike a `treat` plot, in a `missing` plot, `panelview` only displays the missing value status of a panel dataset with selected variables.

An `outcome` plot visualizes a continuous or discrete variable in a panel dataset over time. When a treatment variable is provided, `panelview` also depicts the temporal dynamics of the variable of interest using different colors based on treatment status, making it easier to visually inspect a potential causal relationship.

Formula	`type`	Display
R function: `panelview()`		
`Y ~ D + X`	`treat`	Treatment status of `D` given complete data in `Y`, `D`, and `X`.
`1 ~ D + X`	`treat`	Treatment status of `D` given complete data in `D` and `X`.
`1 ~ D`	`treat`	Treatment status of `D`.
`Y ~ 1`	`treat`	Not allowed; algorithm will force `type = "missing"`.
`Y ~ D + X`	`missing`	Missingness in `Y`, `D`, or `X`.
`Y ~ 1`	`missing`	Missingness in `Y`.
`1 ~ Y`	`missing`	Not allowed.
`Y ~ D + X`	`outcome`	Time series of `Y` colored in `D` given complete data in `Y`, `D`, and `X`.
`Y ~ 1`	`outcome`	Time series of `Y`.
`Y ~ D + X`	`bivariate`	Relationship of `Y` and `D` given complete data in `Y`, `D`, and `X`.
Stata command: `panelview`		
`Y D X`	`treat`	Treatment status of `D`, conditional on no-missing-data in `Y`, `D`, and `X`.
`D X`	`treat` (with `ignoreY`)	Treatment status of `D`, conditional on no-missing-data in `D` and `X`.
`D`	`treat`	Treatment status of `D`; option `ignoreY` is switched on by default.
`Y D X`	`missing`	Missingness in `Y`, `D`, or `X`.
`Y`	`missing`	Missingness in `Y`.
`Y D X`	`outcome`	Time series of `Y`, colored in `D`, conditional on no-missing-data in `Y`, `D`, and `X`.
`Y`	`outcome`	Time series of `Y`.
`Y D X`	`bivariate`	Relationship of `Y` and `D`, conditional on no-missing-data in `Y`, `D`, and `X`.

Table 1: Overview of syntax.

Finally, a `bivariate` plot displays the bivariate relationship between two variables, `Y` and `D`, which are the LHS variable and the first RHS variable in R, or the first two variables in the variable list in Stata. Just like in previous plots, control variables `X` will only affect the missing value status of each observation in the plot.

In the next three sections, we will explore these types of plots and their features. The figures will primarily be generated using R; however, we will also provide equivalent commands for producing similar figures in Stata.

4. Plotting treatment conditions and missing values

We first load the **panelView** package in R, which ships three datasets, `capacity`, `simdata` and `turnout`.

```
R> library("panelView")
R> data("panelView", package = "panelView")
```

We will use the `turnout` dataset to demonstrate how to visualize the treatment status and missing values in a panel dataset. The following R code will be used to plot the presence of EDR laws across US states:

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "treat",
+    xlab = "Year", ylab = "State")
```

Please note that the `type = "treat"` option is the default setting and can be omitted. The `data` option specifies the data frame being used, while the `index` option identifies the unit (group) and time indicators. The usage of these two options remains consistent across all types of plots.

`panelview` designates the first RHS variable in the formula, `policy_edr`, as the treatment indicator, and takes the second and third variables, `policy_mail_in` and `policy_motor`, as covariates. The outcome variable, `turnout`, is not a major input in a `treat` plot, but it and the covariates may affect the appearance of the plot due to missing values. The x- and y-axis titles can be altered using the `xlab` and `ylab` options, respectively. The resulting figure is displayed in Figure 1(a), where light blue and dark blue cells represent control and treatment conditions, respectively. As seen in Figure 1(a), the treatment follows a pattern of staggered adoption, meaning that once it is implemented, it does not revert back in the observed time frame.

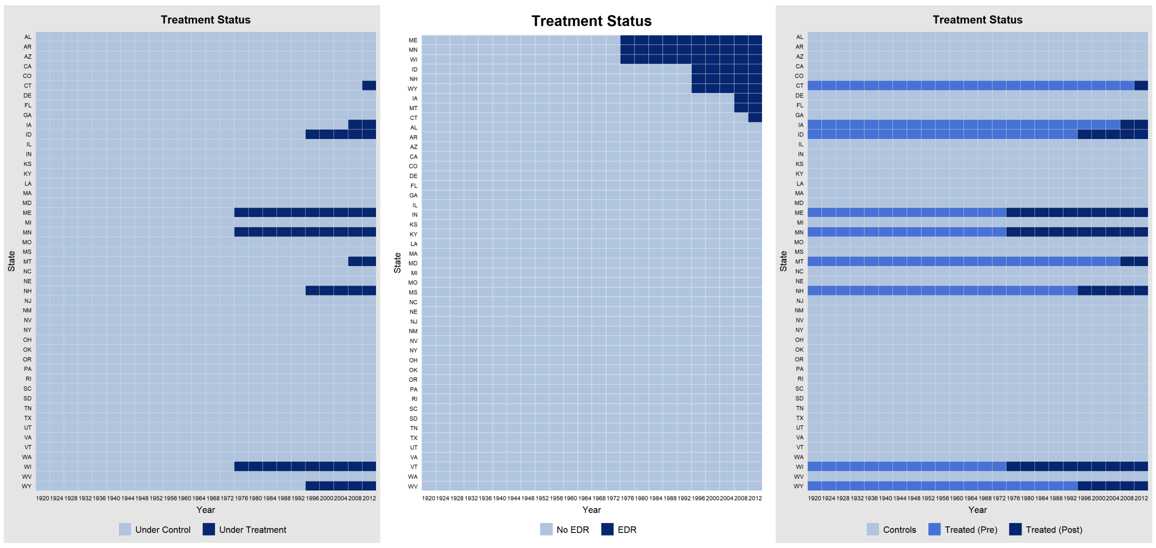
To enhance the visual appeal of the graph and make the data structure easier to understand, the `by.timing` option can be used, which sorts units based on the timing of when they received the treatment (followed by the total number of periods exposed to the treatment), as shown in Figure 1(b). The `background` option and various `cex` options allow the user to set the background color and adjust font sizes for the main title (`cex.main`), axes (`cex.axis`), axis labels (`cex.lab`), and legend (`cex.legend`). The `legend.labs` option allows for label adjustments in the legend, when its length is equal to the number of treatment statuses (in this case, 2):

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), xlab = "Year",
+    ylab = "State", by.timing = TRUE, legend.labs = c("No EDR", "EDR"),
+    background = "white", cex.main = 20, cex.axis= 8, cex.lab = 12,
+    cex.legend = 12)
```

If the treatment is dichotomous and never reverts, we can distinguish the pre-treatment and post-treatment periods for treated units by setting `pre.post = TURE`, as shown in Figure 1(c). This figure can be produced by:

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), xlab = "Year",
+    ylab = "State", pre.post = TRUE)
```

Researchers can use the `main` option to change the title of the plot, the `axis.lab.gap` option to adjust the spacing between labels on the x- and y-axes, and `axis.lab = "time"` (or `"unit"`) to remove the labels on the y-axis (or x-axis). Setting `axis.lab = "off"` will remove labels on both axes. By default, `axis.lab = "both"`. Additionally, instead of using a formula, the treatment indicator can be named directly, which will produce a similar result as Figure 1(c):



(a) Baseline. (b) Add `by.timing = TRUE`. (c) Add `pre.post = TRUE`.

Figure 1: Treatment Status of Election Day Registration (EDR) for the `turnout` data.

```
R> panelview(D = "policy_edr", data = turnout, index = c("abb", "year"),
+    xlab = "Year", ylab = "State", pre.post = TRUE, main = "EDR Reform",
+    axis.lab = "time")
```

If the time variable is unevenly distributed, the `leave.gap = TRUE` option can be used to display the gaps in time using white bars, as demonstrated in Figure 2. If this option is not used, `panelview` will skip over the time gaps and issue a warning message saying "Time is not evenly distributed (possibly due to missing data)."

```
R> turnout2 <- turnout[!(turnout$year=="1924" | turnout$year=="1928" |
+    turnout$year == "1940"), ]
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout2, index = c("abb", "year"), type = "treat",
+    leave.gap = TRUE)
```

Below, we also include a script that reproduces all the examples described above using the Stata version of the package. The command structure and option names are comparable to those in the R version.

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

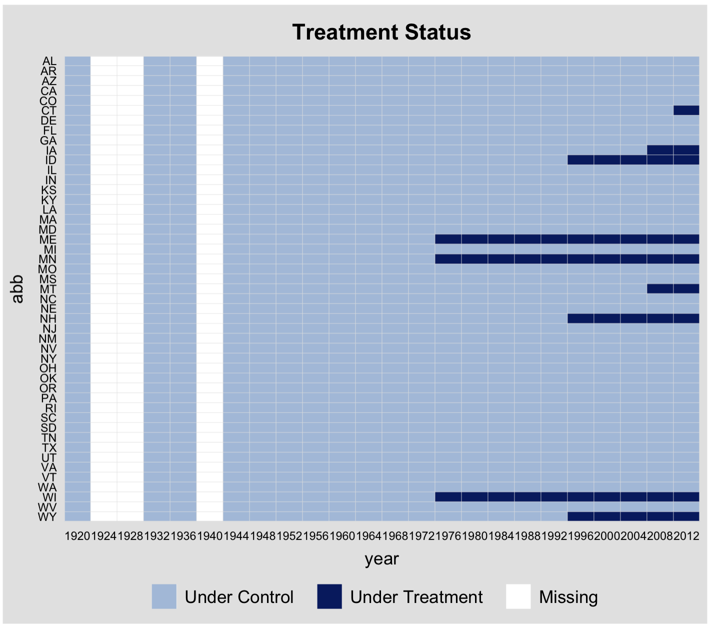


Figure 2: Leave gaps as white bars when time is not evenly distributed.

```
          i(abb) t(year) type(treat) xtitle("Year") ytitle("State") ///
          title("Treatment Status") prepost
panelview policy_edr, ///
          i(abb) t(year) type(treat) xtitle("Year") ytitle("State") ///
          title("EDR Reform") prepost ylabel(none)
drop if year==1924 | year==1928 | year==1940
panelview turnout policy_edr policy_mail_in policy_motor, ///
          i(abb) t(year) type(treat) leavegap
```

In general, the R and Stata versions produce similar visualizations. One difference is that the R version generates a **ggplot2** object that can be easily customized. For instance, if the statement "`+ ggtitle("")`" is added to a **panelView** object in R, the title of the plot will be removed. With Stata, users can customize graphs using Stata's graph editor.

4.1. Treatment reversals

In panel data with treatment reversals, meaning the treatment may switch from on to off after being implemented, we can no longer differentiate between pre- and post-treatment statuses. To demonstrate `panelview` in a panel setting with treatment reversals, we will use the `capacity` dataset. Figure 3 displays the first 100 units of the `capacity` dataset, where there are many treatment reversals and some missing values. Users can tabulate the number and percentage of missing values using the `report.missing` option.

```
R> panelview(Capacity ~ demo + lnpop + lngdp, data = capacity,
+    index = c("ccode", "year"), axis.lab.gap = c(2,0),
+    main = "Democracy and State Capacity", show.id = 1:100,
+    axis.lab = "time", report.missing = TRUE)
```

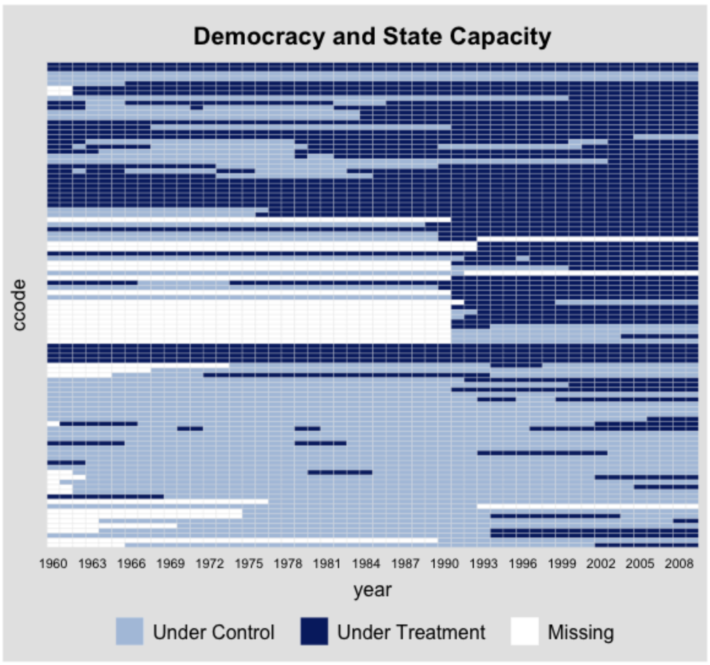


Figure 3: Democracy and State Capacity for the `capacity` data.

```
         # Missing % Missing
Capacity         0       0.0
demo             0       0.0
lnpop          118       1.6
lngdp          505       7.0
```

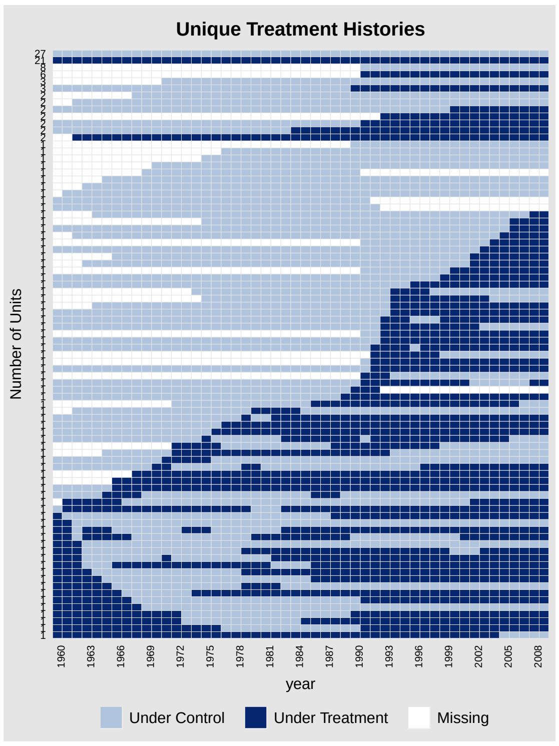
`panelview` provides the option of specifying the units to be displayed using either `show.id` (units in alphabetical order) or `id` (original unit IDs as recorded in the "unit" variable). These options are particularly useful when the dataset contains a large number of units (e.g., more than 250). A similar functionality can be achieved in the Stata package using an `if` clause:

```
sysuse capacity.dta, clear
egen ccodeid = group(ccode)
panelview Capacity demo lnpop lngdp ccodeid if ccodeid >= 1 & ccodeid <= 100, ///
          i(ccode) t(year) type(treat) ylabel(none) ///
          title("Democracy and State Capacity") xlabdist(3)
```

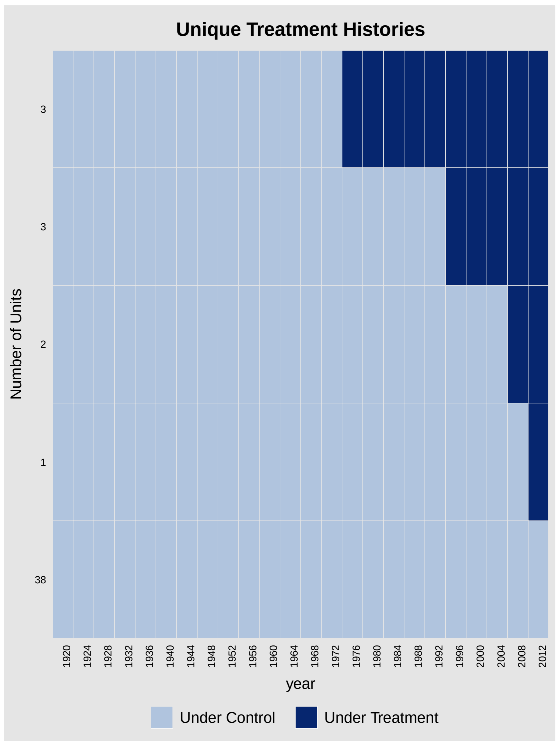
4.2. High-dimensional data

`panelview` supports the visualization of high-dimensional data, i.e., data with a significant number of units. For data with binary treatments, users have the option to plot unique treatment histories by setting `collapse.history = TRUE`.

```
R> panelview(Capacity ~ demo + lnpop + lngdp, data = capacity,
+    index = c("ccode", "year"), axis.lab.gap = c(2,0), axis.lab.angle = 90,
+     collapse.history = TRUE)
```



(a) `capacity` data.



(b) `turnout` data.

Figure 4: Unique treatment histories.

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), by.timing = TRUE,
+    collapse.history = TRUE)
```

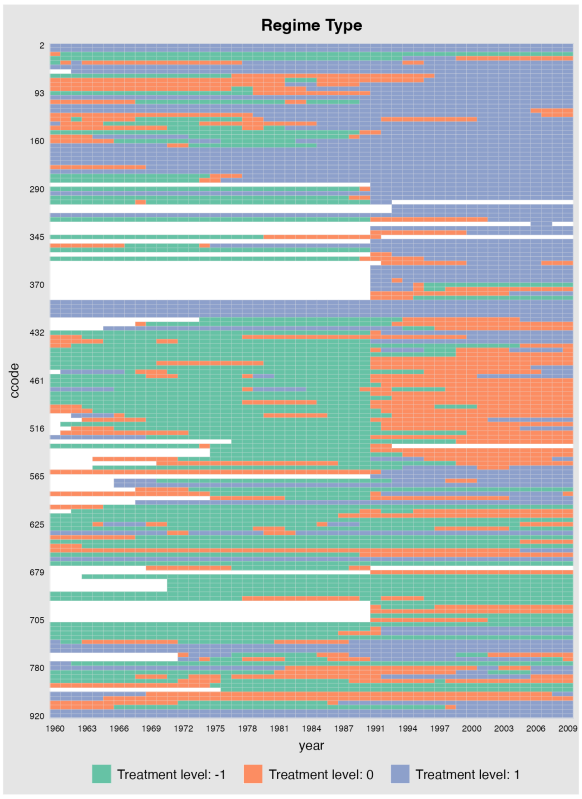
A cohort is defined by a unique individual treatment history. If there are more than 500 units, `collapse.history` is automatically enabled. In Figure 4, the rows are arranged based on the cohort size if `by.timing` is set to `FALSE`, otherwise, the rows are sorted by the timing of the treatment. The values on the y-axis indicate the cohort size for each treatment history. We thank an anonymous reviewer for suggesting this functionality.

In the Stata package, we sort each treatment history by treatment timing:

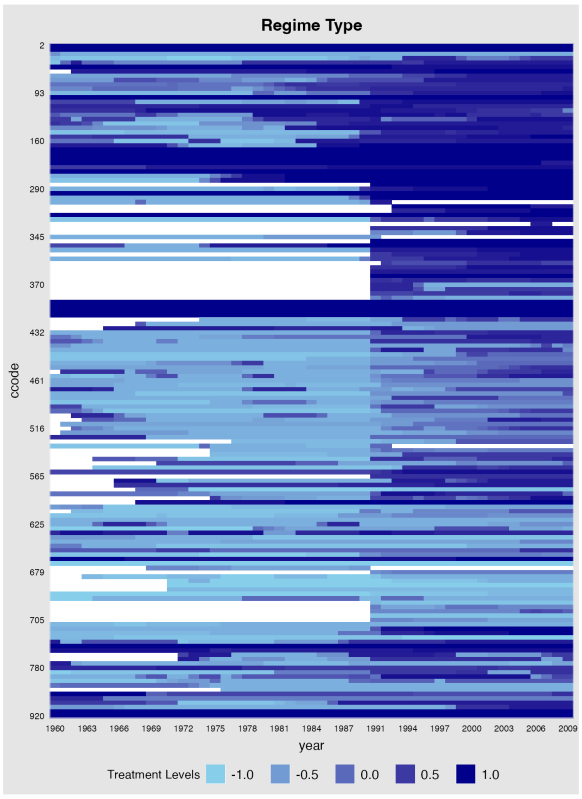
```
sysuse capacity.dta, clear
panelview Capacity demo lnpop lngdp, i(ccode) t(year) type(treat) ///
    xlabdist(3) collapsehistory title("Unique Treatment Histories")

sysuse turnout, clear
panelview turnout policy_edr policy_mail_in policy_motor, i(abb) t(year) ///
      type(treat) collapsehistory title("Unique Treatment Histories")
```

When working with high-dimensional data, we recommend users first gain an understanding of the data's dimensionality and tabulate the unique values of the treatment variable, and



(a) Three treatment levels.



(b) Continuous treatment.

Figure 5: More than two treatment levels: `capacity` data.

then visualize distinct treatment histories with the help of `panelview`, after which users can visualize the dynamics of other variables of interest.

4.3. More than two treatment conditions

`panelview` supports panel data with more than two treatment levels. For example, a measure of regime type with three treatment levels can be created in the `capacity` dataset by assigning `demo2` the values of -1, 0, and 1 based on the Polity score being less than -0.5, between -0.5 and 0.5, and greater than 0.5, respectively. The treatment status of `demo2` is visualized in Figure 5(a).

```
R> panelview(Capacity ~ demo2 + lngdp, data = capacity,
+    index = c("ccode", "year"), axis.lab.gap = c(2, 10),
+    main = "Regime Type")
```

When the number of treatment levels exceeds five, the treatment indicator will be considered a continuous variable. To demonstrate this difference, we use the continuous Polity score, `polity2`, and remove the grid lines in Figure 5(b) by setting `gridOff = TRUE`:

```
R> panelview(Capacity ~ polity2 + lngdp, data = capacity,
+    index = c("ccode", "year"), axis.lab.gap = c(2, 10),
+    main = "Regime Type", gridOff = TRUE)
```

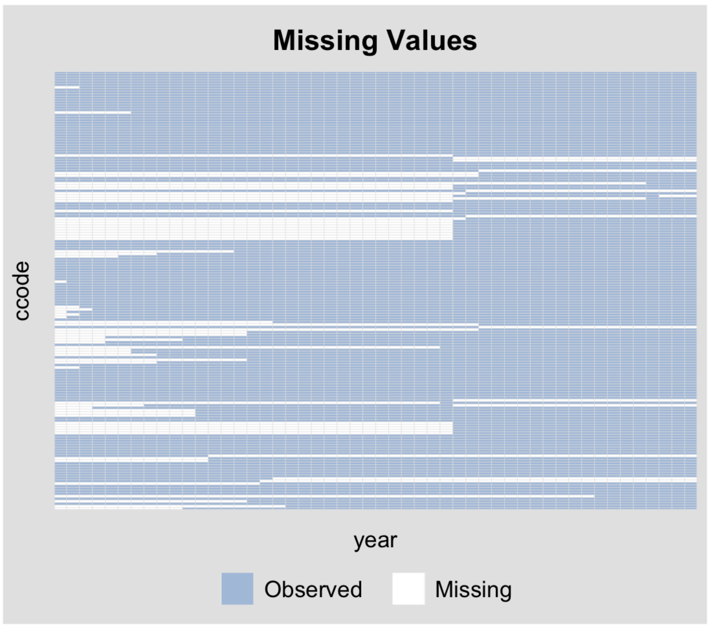


Figure 6: Plotting missing values: `capacity` data.

We can replicate Figure 5(a) in Stata using the following code:

```
panelview Capacity demo2 lngdp, i(ccode) t(year) ///
          type(treat) title("Regime Type") xlabdist(3) ylabdist(10)
```

Similar to the R package, the Stata version of `panelview` will designate the treatment indicator as continuous and then divide the continuous variable into five levels evenly. The resulting plot will be the same as Figure 5(b).

```
panelview Capacity polity2 lngdp, i(ccode) t(year) type(treat) ///
          title("Regime Type") xlabdist(3) ylabdist(10)
```

4.4. Showing missing values only

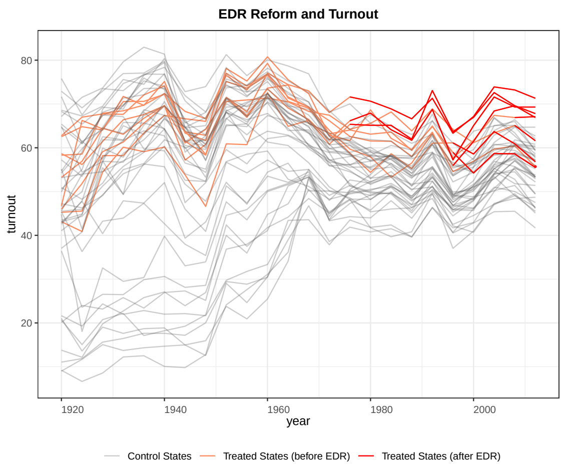
A `missing` plot is similar to a `treat` plot, but only displays information about missing values. To create a `missing` plot, the option `type = "missing"` can be specified (as shown in Figure 6):

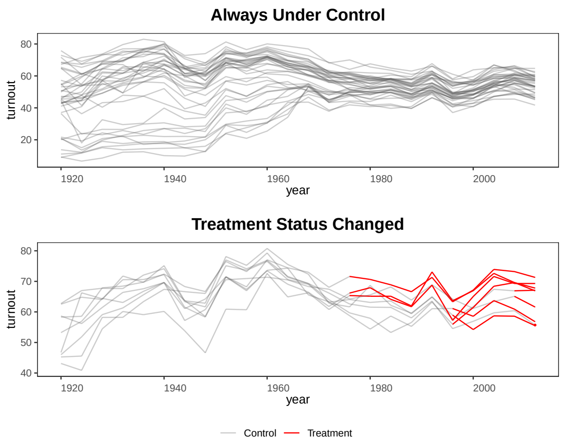
```
R> library("dplyr")
R> capacity %>% panelview(Capacity ~ 1, index = c("ccode", "year"),
+    axis.lab = "off", type = "missing")
```

Alternatively, users can directly specify a variable name without supplying a formula:

```
R> capacity %>% panelview(Y = "Capacity", index = c("ccode", "year"),
+    axis.lab.gap = c(2, 10), type = "missing")
```

The Stata code to generate a plot similar to Figure 6 is:





(a) Baseline. (b) Different treatment groups.

Figure 7: Visualizing a continuous outcome variable: `turnout` data.

```
panelview Capacity, i(ccode) t(year) type(missing) ///
          title("Missing Values") xlabel(none) ylabel(none)
```

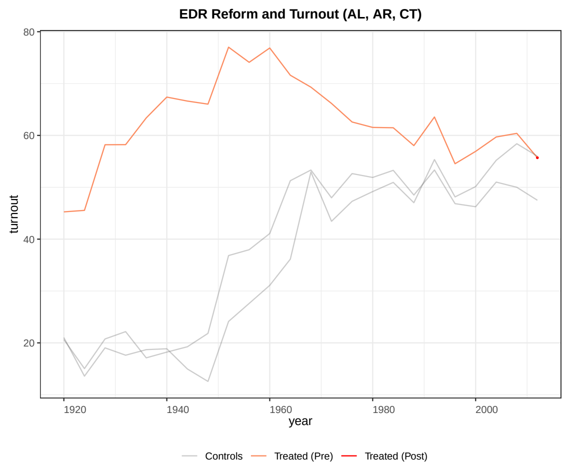
5. Visualizing the temporal dynamics of an outcome variable

Plotting the raw data of an outcome variable over time, along with information about treatment status, can provide insight into the sign and magnitude of the treatment effect and its variability. This can be achieved in `panelview` by setting `type = "outcome"`. The resulting plot will have time on the x-axis and levels of the outcome variable on the y-axis. If the outcome variable is continuous, `panelview` will connect the observations for each unit with a line in temporal order. Different colors represent different treatment conditions for a continuous outcome variable, while dots are used for a discrete outcome variable. Please note that the remaining variables in the formula play no role in the plot, except in cases where they are only partially observed and therefore impact the plot due to missing data.

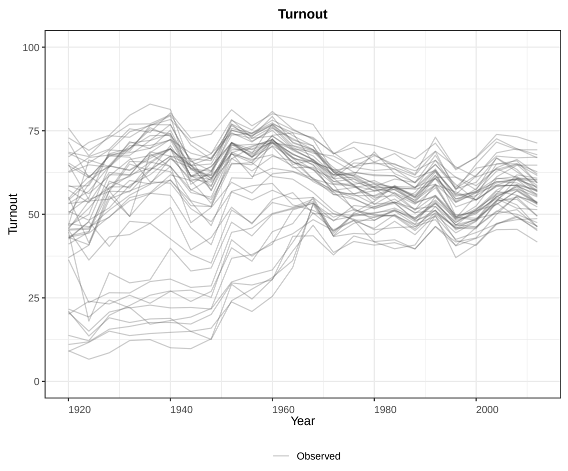
5.1. Continuous outcome

The following code generates Figure 7(a). The plot shows the control states, which never adopted EDR, in gray and the treated states in orange (pre-treatment periods) and red (post-treatment periods). The use of color in the plot allows for a preliminary evaluation of the plausibility of treatment effects.

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "outcome",
+    main = "EDR Reform and Turnout", legend.labs = c("Control States",
+      "Treated States (before EDR)", "Treated States (after EDR)"),
+    theme.bw = TRUE)
```



(a) Units specified.



(b) Ignore treatment status.

Figure 8: Plotting a continuous outcome variable (cont.): `turnout` data.

Analysts can customize the axis ranges by specifying `xlim` and `ylim`. The colors of observations under different treatment conditions can be adjusted using the `color` option. In addition, a black-and-white theme can be applied using `theme.bw = TRUE`.

The following script creates Figure 7(b) where the outcome variable is grouped based on the treatment status change. The plot shows that most of the states are always in control, while only a few states switched to and from the treatment status. The font sizes of the title and subtitle can be adjusted using the `cex.main` and `cex.main.sub` options. To arrange the subfigures in a row instead of a column, use the `by.group.side = TRUE` option instead of `by.group = TRUE`.

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "outcome",
+    main = "", cex.main = 20, cex.main.sub = 15, by.group = TRUE)
```

Analysts can specify the specific unit(s) to be displayed, e.g, Figure 8(a):

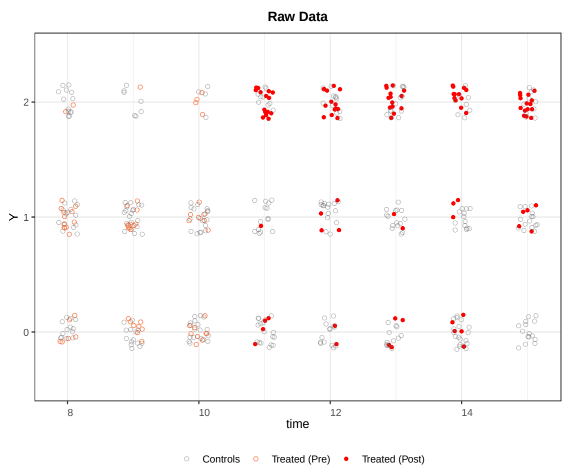
```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "outcome",
+    main = "EDR Reform and Turnout (AL, AR, CT)", id = c("AL", "AR", "CT"))
```

In the absence of a treatment indicator, `panelview` will simply plot the temporal dynamics of the outcome variable without showing treatment status. This is demonstrated in Figure 8(b).

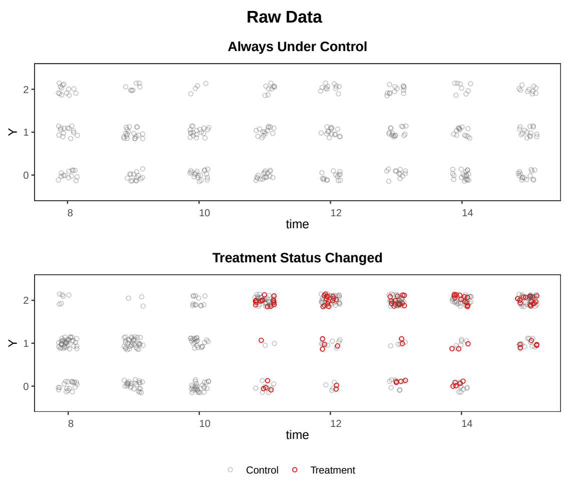
```
R> panelview(turnout ~ 1, data = turnout, index = c("abb", "year"),
+    type = "outcome", main = "Turnout", ylim = c(0, 100),
+    xlab = "Year", ylab = "Turnout")
```

If the treatment indicator has more than two levels or is continuous, `panelview` will not take into account the treatment status and produce a plot similar to Figure 8(b), which only visualizes the temporal dynamics of the outcome variable.

The Stata commands to produce figures similar to Figures 7 and 8 are as follows:



(a) Baseline.



(b) Different treatment groups.

Figure 9: Discrete outcome for the `simdata` data.

```
sysuse 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") prepost graphregion(fcolor(white))
panelview turnout policy_edr policy_mail_in policy_motor, ///
          i(abb) t(year) type(outcome) xtitle("Year") ytitle("Turnout") ///
          bygroup xlabel(1920 (20) 2000)
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)") prepost
panelview turnout, i(abb) t(year) type(outcome) ///
          title("Turnout") ylabel(0 (25) 100)
```

5.2. Discrete outcome

Users can also specify that a variable of interest is discrete by using the option `outcome.type = "discrete"`. When this option is used, `panelview` will not connect the observations with lines and will instead use jittering to better visualize the data. As an example, Figure 9(a) shows the temporal dynamics of a discrete variable `Y` which takes on three values (0, 1, 2) in the `simdata` dataset. By using the option `by.group = TRUE`, the resulting plot, shown in Figure 9(b), groups the observations based on whether the treatment status has changed over time.

```
R> panelview(Y ~ D, data = simdata, index = c("id", "time"),
+    outcome.type = "discrete", type = "outcome", xlim = c(8, 15))
R> panelview(Y ~ D, data = simdata, index = c("id", "time"), by.group = TRUE,
+    outcome.type = "discrete", type = "outcome", xlim = c(8, 15))
```

The following commands produce similar figures in Stata:

```
sysuse simdata.dta, clear
panelview Y D if time >= 8 & time <= 15, ///
          type(outcome) i(id) t(time) discreteoutcome ///
          title("Raw Data") xlabel(8 (2) 15) ylabel(0 (1) 2)
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
```

6. Visualizing bivariate relationships

With the option `type = "bivariate"` (or `"bivar"` for short), `panelview` can be used to explore the relationship between two variables in a panel dataset. If the outcome variable is continuous, a line plot is generated by default (`"line"` or `"l"`). If it is discrete, a bar plot is generated by default (`"bar"` or `"b"`). The connected line plot (`"connected"` or `"c"`) is also available as an option. The style of the plot can be changed using the `style` option, for example, `style = c("c", "b")` generates a connected line plot for the outcome and a bar plot for the treatment. In this context, the LHS variable is referred to as the outcome variable and the first RHS variable is referred to as the treatment variable. However, these labels do not imply causality. The remaining variables only affect the missing value status of the data being visualized.

6.1. Visualizing bivariate relationships in aggregate

In Figure 10(a), `panelview` plots the average turnout against the proportion of states enacting EDR laws using the `turnout` data. By default, the plot is in aggregate (`by.unit = FALSE`), meaning that the treatment and outcome variables are visualized as overall means.

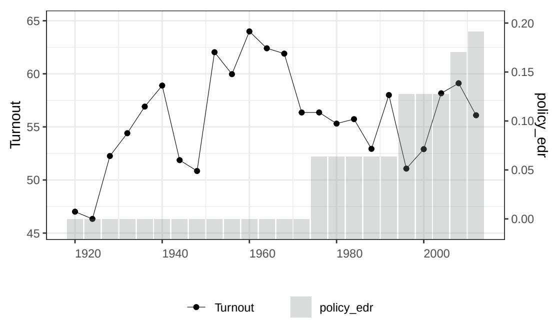
```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "bivariate",
+    style = c("c", "b"), ylab = "Turnout")
```

Figure 10(b) displays a bivariate plot with a continuous outcome and a continuous treatment. The y-ranges of both time series can be adjusted using the `ylim` option. The first vector in `ylim = list(c( , ), c( , ))` specifies the range for the outcome, while the second one specifies the range for the treatment.

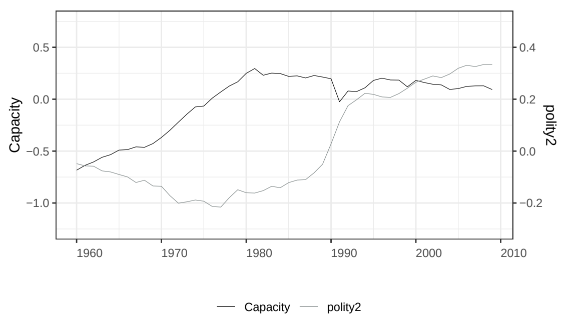
```
R> panelview(Capacity ~ polity2, data = capacity,
+    index = c("country", "year"), ylim = list(c(-1, 0.5), c(-0.2, 0.4)),
+    type = "bivar")
```

The following Stata codes replicate the plots in Figure 10.

```
sysuse turnout.dta, clear
panelview turnout policy_edr, i(abb) t(year) xlabdist(7) type(bivariate) ///
          msize(*0.5) style(c b) ytitle("turnout") ///
          ytitle("policy_edr", axis(2)) ///
          legend(label(1 "turnout") label(2 "policy_edr")) ///
          ylabel(45 (5) 65) ylabel(0 (0.05) 0.2, axis(2))
```



(a) Continuous `Y`, discrete `D`.



(b) Continuous `Y`, continuous `D`.

Figure 10: Bivariate relationships between two variables.

```
sysuse capacity.dta, clear
panelview Capacity polity2, i(country) t(year) type(bivar) xlabdist(20)
```

6.2. Visualizing bivariate relationship by unit

Figure 11(a) illustrates the bivariate relationship between the continuous outcome `turnout` and the discrete treatment `policy_edr` by unit. Each subfigure represents a state, and the line plot and bar plot show the `turnout` outcome in that state and the status of EDR laws, respectively.

```
R> panelview(turnout ~ policy_edr + policy_mail_in + policy_motor,
+    data = turnout, index = c("abb", "year"), type = "bivar",
+    by.unit = TRUE, show.id = 10:21, ylab = "Turnout")
```

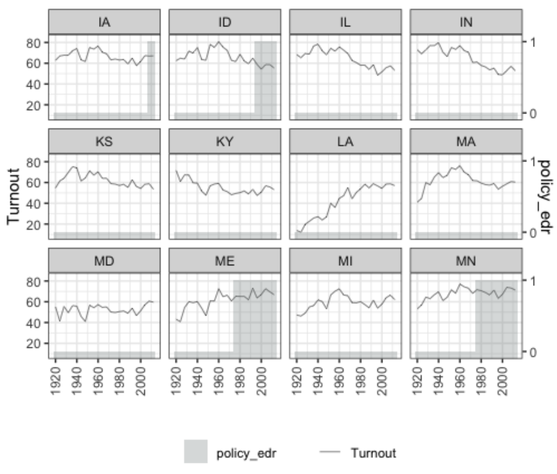
Figure 11(b) shows an example when both variables are continuous.

```
R> panelview(Capacity ~ polity2, data = capacity,
+    index = c("country", "year"), type = "bivar", by.unit = TRUE,
+    show.id = c(20:25, 40:45))
```

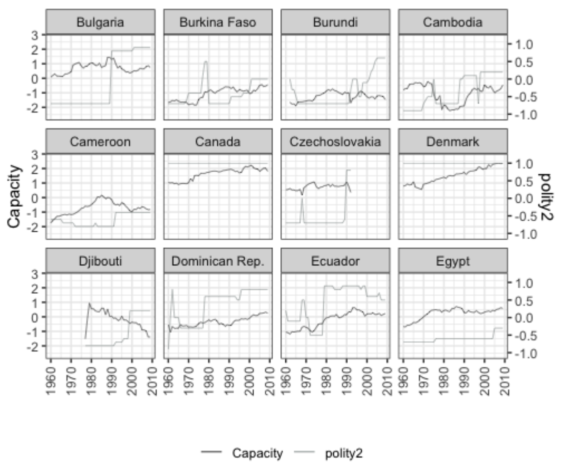
The following Stata code yields similar results as in Figure 11:

```
sysuse turnout.dta, clear
panelview turnout policy_edr policy_mail_in policy_motor ///
          if abb >= 10 & abb <= 21, ///
          i(abb) t(year) xlabdist(10) type(bivar) byunit

sysuse capacity.dta, clear
panelview Capacity polity2 ///
          if country >= 20 & country <= 25 | country >= 40 & country <= 45, ///
          i(country) t(year) xlabdist(20) ///
          type(bivar) byunit
```



(a) Continuous Y, discrete D.



(b) Continuous Y, continuous D.

Figure 11: Bivariate relationships between two variables: by each unit.

7. Conclusion

To facilitate causal inference with panel data, we develop the **panelView** package in R and the **panelview** package in Stata. These tools provide a simple and practical solution for visualizing panel data, allowing users to plot treatment status and missing values, display the temporal dynamics of key outcomes, and explore bivariate relationships between a treatment and an outcome. These visualization techniques promote transparency and are critical steps towards achieving credible causal inference with panel data. More information on both packages, including online manuals with more examples, can be found at https://yiqingxu.org/packages/panelview/.

Acknowledgments

We thank Yiqin Fu, Xincheng Lan, Ziyi Liu, and two anonymous reviewers for their helpful comments.

References

Abadie A, Diamond A, Hainmueller J (2010). "Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program." *Journal of the American Statistical Association*, **105**(490), 493–505. doi:10.1198/jasa.2009.ap08746.

Abadie A, Gardeazabal J (2003). "The Economic Costs of Conflict: A Case Study of the Basque Country." *The American Economic Review*, **93**(1), 113–132. doi:10.1257/000282803321455188.

Card D, Krueger AB (1994). "Minimum Wages and Employment: A Case Study of the Fast-

Food Industry in New Jersey and Pennsylvania." *The American Economic Review*, **84**(4), 772–793. doi:10.3386/w4509.

Cox NJ (2009). "Speaking Stata: Paired, Parallel, or Profile Plots for Changes, Correlations, and Other Comparisons." *The Stata Journal*, **9**(4), 621–639. doi:10.1177/1536867x0900900408.

de Chaisemartin C, D'Haultfœuille X (2020). "Two-Way Fixed Effects Estimators with Heterogeneous Treatment Effects." *The American Economic Review*, **110**(9), 2964–2996. doi:10.1257/aer.20181169.

Ender P (2014). "Profile Analysis." *2014 Stata conference*, Stata Users Group. URL https://ideas.repec.org/p/boc/scon14/1.html.

Gassen J (2020). ***ExPanDaR**: Explore Your Data Interactively.* R package version 0.5.3, URL https://CRAN.R-project.org/package=ExPanDaR.

Goodman-Bacon A (2021). "Difference-in-Differences with Variation in Treatment Timing." *Journal of Econometrics*. doi:10.3386/w25018.

Hsiao C (2003). *Analysis of Panel Data*. Cambridge University Press.

Kim IS, Rauh A, Wang E, Imai K (2022). ***PanelMatch**: Matching Methods for Causal Inference with Time-Series Cross-Sectional Data.* R package version 2.0.1, URL https://CRAN.R-project.org/package=PanelMatch.

Long JA (2023). ***panelr**: Regression Models and Utilities for Repeated Measures and Panel Data.* R package version 0.7.8, URL https://CRAN.R-project.org/package=panelr.

Mou H, Liu L, Xu Y (2023). ***panelView**: Visualizing Panel Data.* R package version 1.1.17, URL https://CRAN.R-project.org/package=panelView.

Mou H, Xu Y (2023). "**panelview**: Stata Module to Visualize Panel Data." Statistical Software Components, Boston College Department of Economics. URL https://EconPapers.repec.org/RePEc:boc:bocode:s459034.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna. URL https://www.R-project.org/.

Robins JM, Hernán MA, Brumback B (2000). "Marginal Structural Models and Causal Inference in Epidemiology." *Epidemiology*, **11**(5), 550–560. doi:10.1097/00001648-200009000-00011.

StataCorp (2021). *Stata Statistical Software: Release 17*. StataCorp LLC, College Station.

Wang EH, Xu Y (2018). "Awakening Leviathan: The Effect of Democracy on State Capacity." *Research & Politics*, **5**(2), 2053168018772398. doi:10.1177/2053168018772398.

Wooldridge JM (2001). *Econometric Analysis of Cross Section and Panel Data*. The MIT Press.

Xu Y (2017). "Generalized Synthetic Control Method: Causal Inference with Interactive Fixed Effects Models." *Political Analysis*, **25**(1), 57–76. doi:10.1017/pan.2016.2.

Affiliation:

Yiqing Xu
Department of Political Science
Stanford University
Stanford, CA 94305, United States of America
E-mail: yiqingxu@stanford.edu
URL: https://yiqingxu.org/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
September 2023, Volume 107, Issue 7 *Submitted:* 2022-04-04
doi:10.18637/jss.v107.i07 *Accepted:* 2023-03-16