PanelMatch Overview

Data Preparation

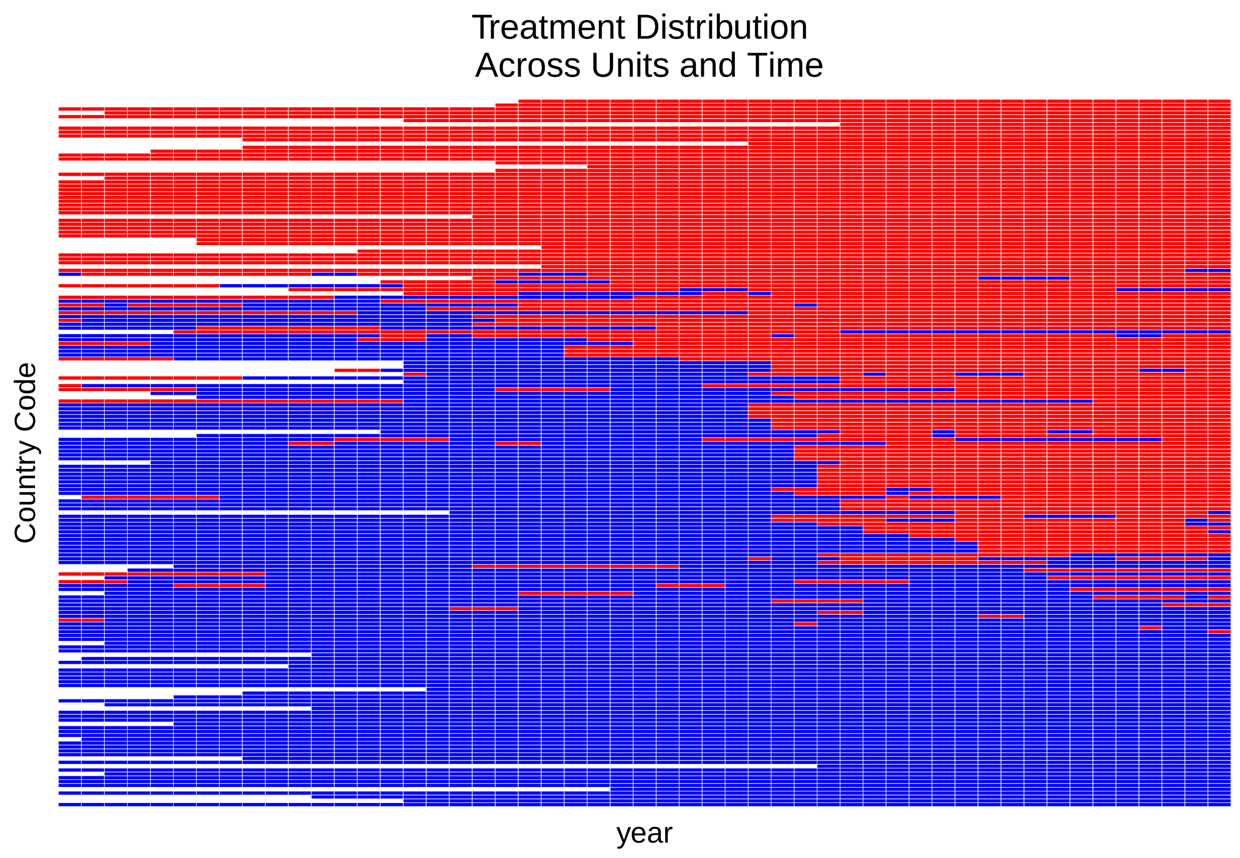
Users should begin by preparing their data with the `PanelData()` function. `PanelData()` conducts a number of error checks on the data, balances the panel, and creates a `PanelData` object which stores the time identifier, unit identifier, treatment, and outcome variables. Storing this metadata simplifies the interface at later stages, so users do not need to repeatedly specify these important variables.

```r
library(PanelMatch)
dem.panel <- PanelData(panel.data = dem,
                       unit.id = "wbcode2",
                       time.id = "year",
                       treatment = "dem",
                       outcome = "y")
```

Treatment Variation Plot

Users can visualize the variation of treatment across space and time. This will help users build an intuition about how comparison of treated and control observations can be made.

```r
dem.panel <- PanelData(panel.data = dem,
               unit.id = "wbcode2",
               time.id = "year",
               treatment = "dem",
               outcome = "y")
DisplayTreatment(panel.data = dem.panel, legend.position = "none",
                 xlab = "year", ylab = "Country Code",
                 hide.x.tick.label = TRUE, hide.y.tick.label = TRUE)
```



While one can create simple plots easily, some additional customization may be desirable. For instance, user-specified labels can help clarify the substantive interpretation of the figures and visual adjustments might be necessary to accommodate larger data sets, as automatically generated labels will become illegible. To this end, the `DisplayTreatment()` function offers a large number of options for adjusting common features of the plot. Additionally, the `DisplayTreatment()` function returns a `ggplot2` object (created using `geom_tile()`), meaning that standard `ggplot2` syntax can be used to further customize any aspect of the figure.

Creating and Refining Matched Sets

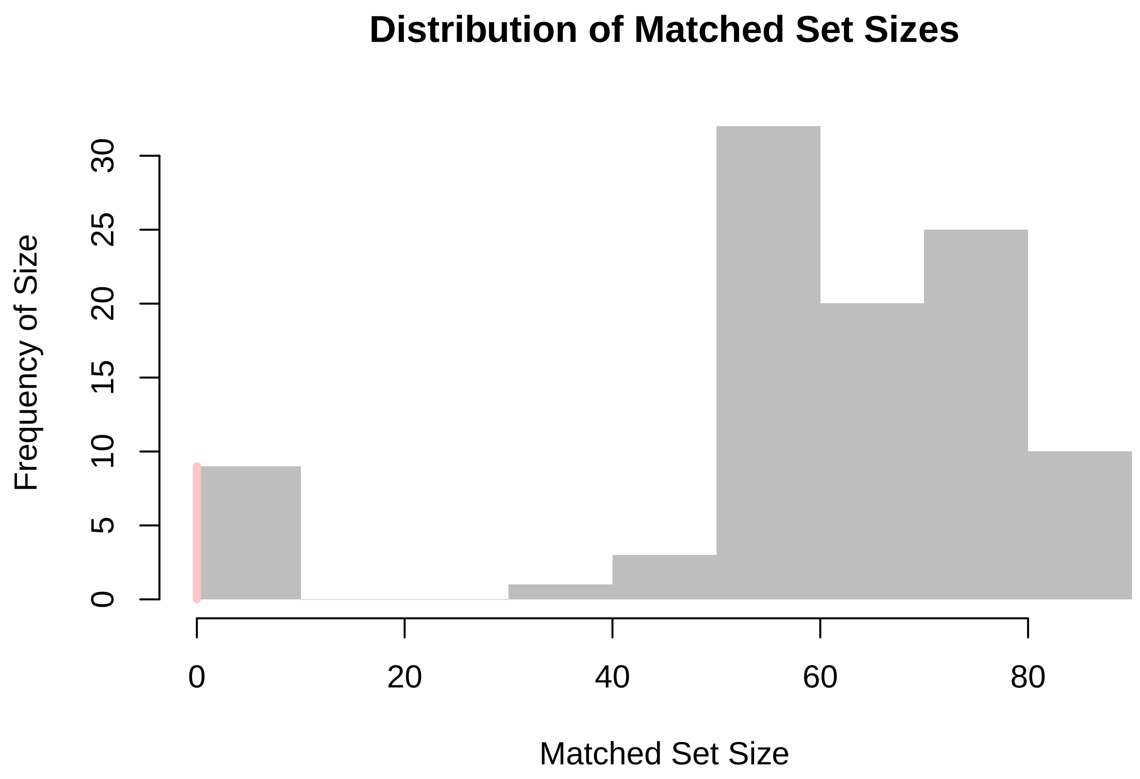
Users can then create and refine matched sets using `PanelMatch()`. There are a large number of parameters that control this process. Please see the function documentation for descriptions.

```
PM.maha <- PanelMatch(panel.data = dem.panel,
                      lag = 4,
                      refinement.method = "mahalanobis",
                      match.missing = FALSE,
                      covs.formula = ~ I(lag(tradewb, 0:4)) +
                                       I(lag(y, 1:4)),
                      size.match = 5,
                      qoi = "att",
                      lead = 0:2,
                      use.diagonal.variance.matrix = TRUE,
                      forbid.treatment.reversal = FALSE)



PM.ps.weight <- PanelMatch(lag = 4,
```

```r
                             refinement.method = "ps.weight",
                             panel.data = dem.panel,
                             match.missing = FALSE,
                             covs.formula = ~ I(lag(tradewb, 0:4)) +
                                              I(lag(y, 1:4)),
                             qoi = "att",
                             lead = 0:2,
                             use.diagonal.variance.matrix = TRUE,
                             forbid.treatment.reversal = FALSE)
```

One can examine the distribution of the sizes of matched sets with the `plot.PanelMatch()` method:

```r
plot(PM.maha)
```

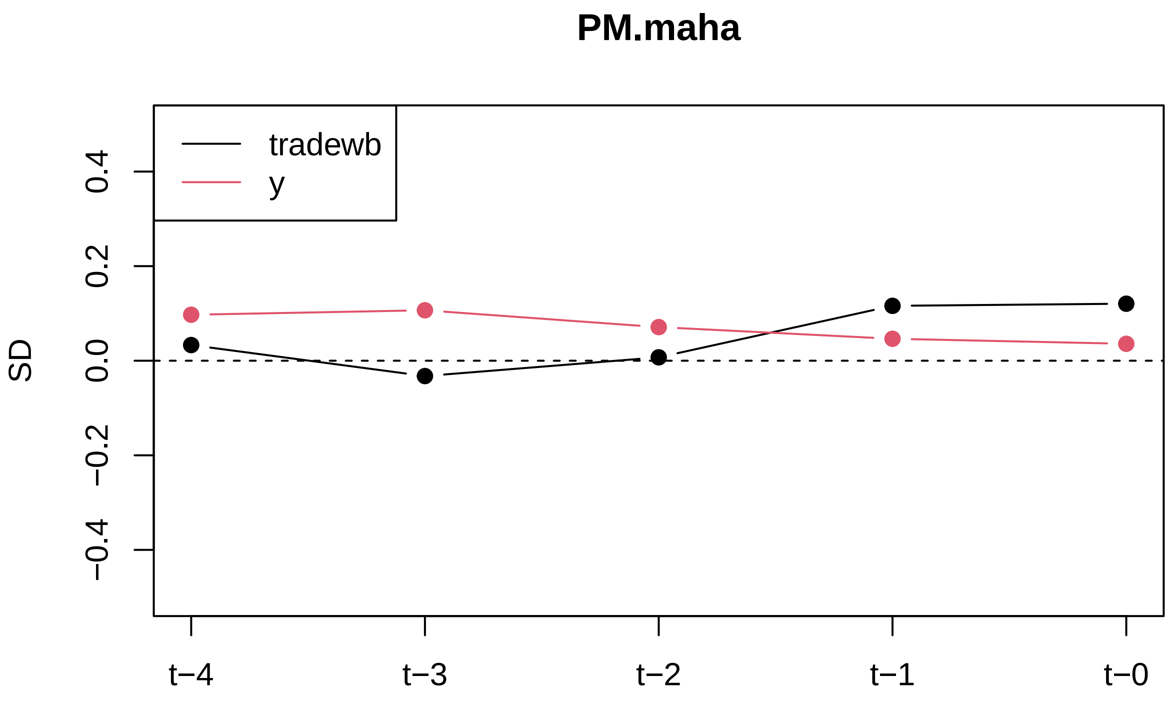


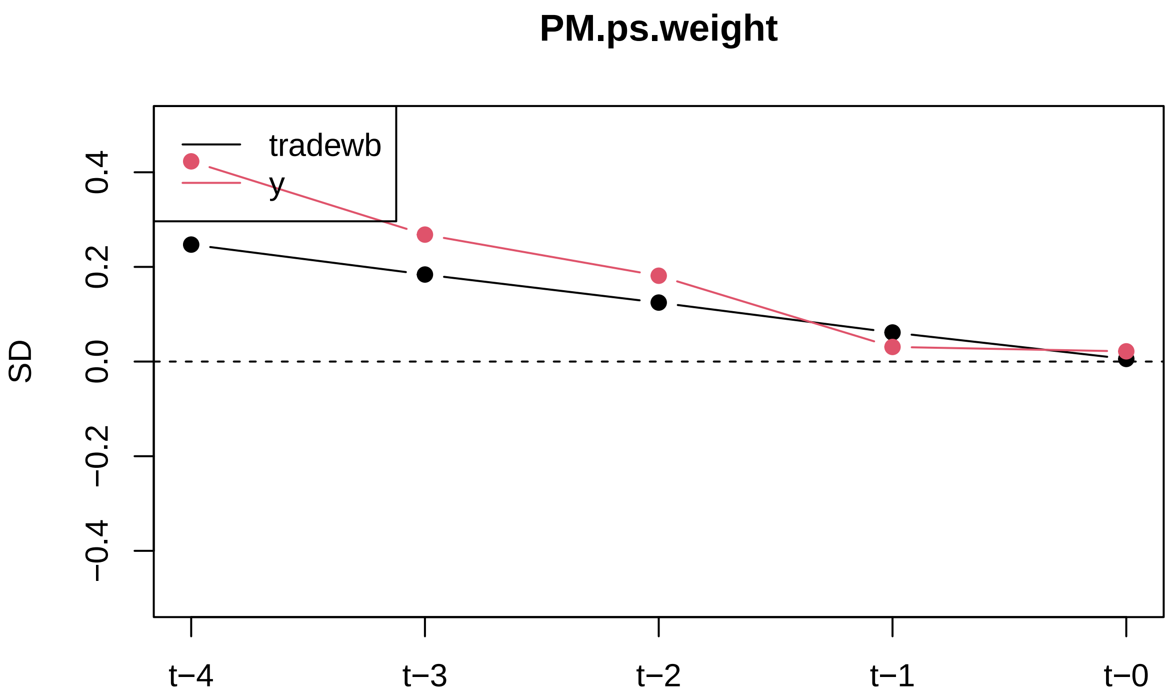
Using the `get_covariate_balance()` function, we can examine the covariate balance measure. Note that the covariate balance measure should be much lower for matched sets after refinement if the configuration used is effective.

```r
covbal <- get_covariate_balance(PM.maha, PM.ps.weight,
                                panel.data = dem.panel,
                                covariates = c("tradewb", "y"),
                                include.unrefined = TRUE)
```

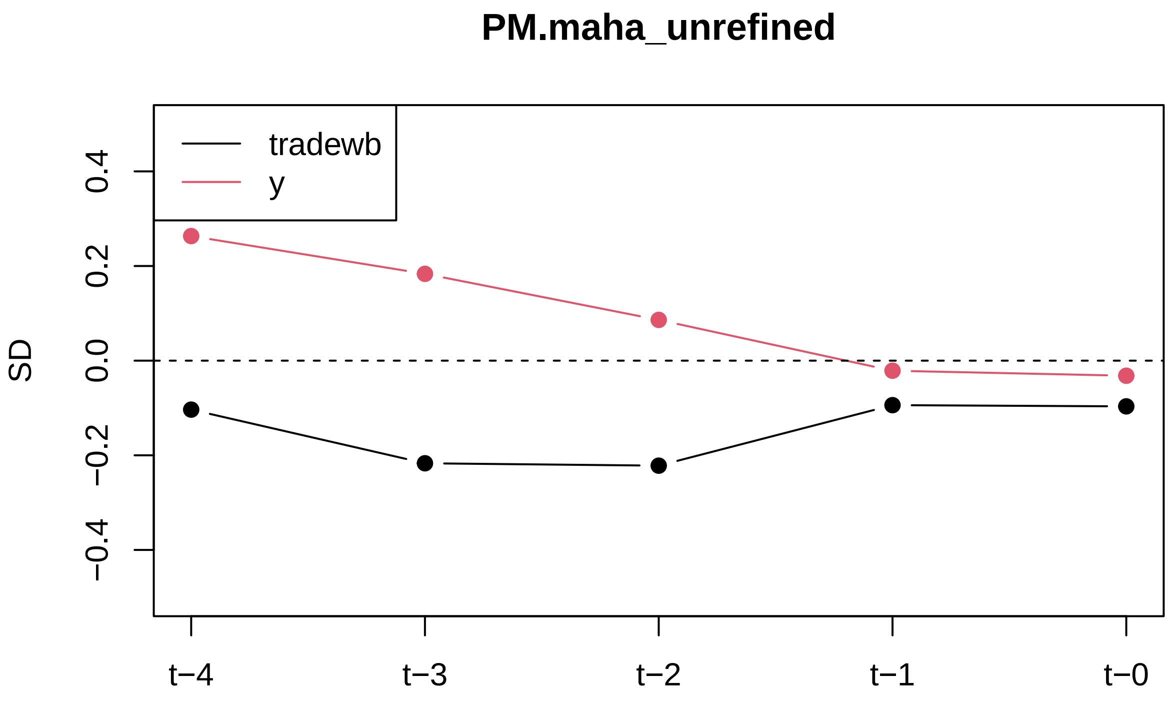
We can examine and plot these results. We can see that the Mahalanobis distance based matching refinement method generally performs better than the propensity score weighting method. We can also visualize the pre-refinement balance measures to see how much refinement improved covariate balance.

```r
summary(covbal)
#> $PM.maha
#>     tradewb_unrefined y_unrefined      tradewb          y
#> t_4       -0.10344989  0.26326816  0.033144720 0.09725542
#> t_3       -0.21683623  0.18343654 -0.032415036 0.10669214
#> t_2       -0.22188279  0.08612944  0.007162189 0.07108454
#> t_1       -0.09402417 -0.02126611  0.116077872 0.04641091
#> t_0       -0.09657564 -0.03184226  0.120572693 0.03583203
#>
#> $PM.ps.weight
#>     tradewb_unrefined y_unrefined    tradewb          y
#> t_4       -0.10344989  0.26326816 0.24713734 0.42302119
#> t_3       -0.21683623  0.18343654 0.18390406 0.26826780
#> t_2       -0.22188279  0.08612944 0.12460414 0.18135719
#> t_1       -0.09402417 -0.02126611 0.06142984 0.03086512
#> t_0       -0.09657564 -0.03184226 0.00554273 0.02157618
plot(covbal, type = "panel",
     include.unrefined.panel = FALSE, ylim = c(-.5, .5))
```





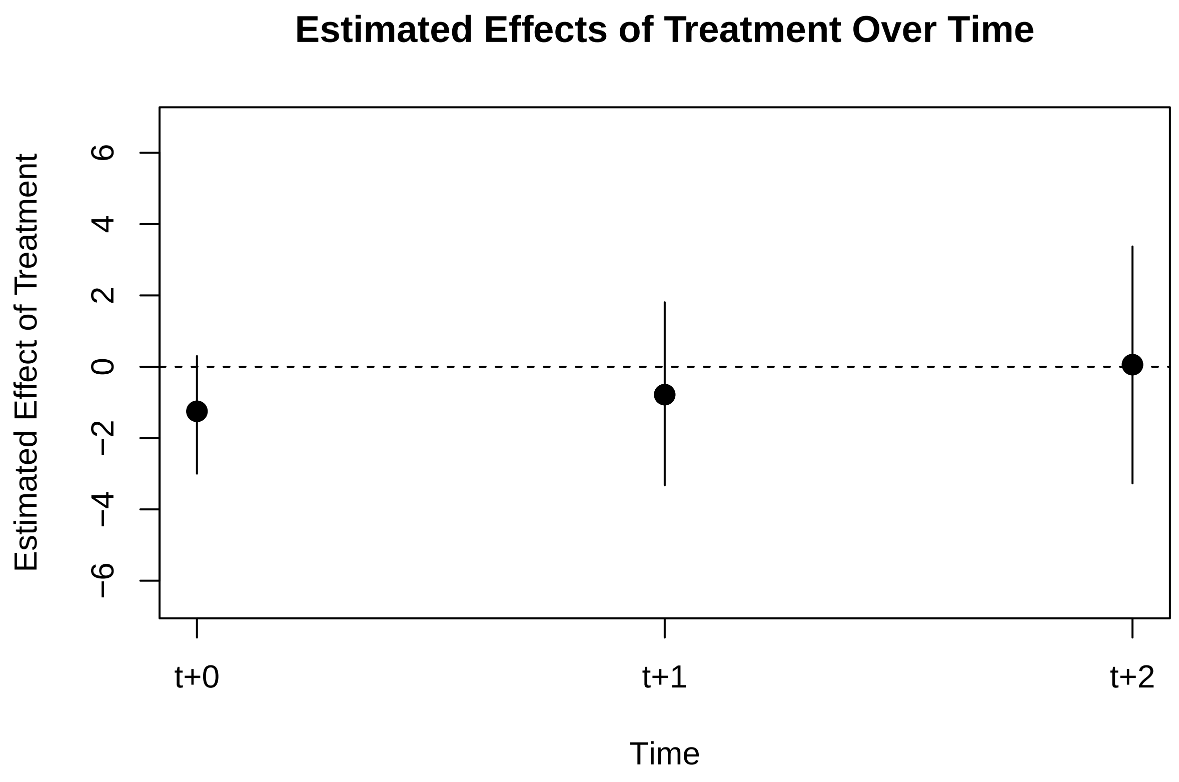
```r
# Since specifications are identical except
# for refinement method, just look at the first result.
plot(get_unrefined_balance(covbal)[1],
     include.unrefined.panel = FALSE, ylim = c(-.5, .5))
```



Getting Estimates and Standard Errors

Once proper matched sets are attained by `PanelMatch()`, users can estimate the causal quantity of interest such as the average treatment effect using `PanelEstimate()`. Users can estimate the contemporaneous effect as well as long-term effects. In this example, we illustrate the use of `PanelEstimate()` to estimate the average treatment effect on treated units (att) at time `t` on the outcomes from time `t+0` to `t+4`.

```
PE.results <- PanelEstimate(sets = PM.maha,
                            panel.data = dem.panel,
                            se.method = "bootstrap")

plot(PE.results)
```


Estimated Effects of Treatment Over Time
Estimated Effect of Treatment
t+0
t+1
t+2
Time