Introduction to the **eventstudies** package in R

Ajay Shah Sargam Jain

January 21, 2019

1 The standard event study in finance

In this section, we look at using the 'eventstudies' package for the purpose of doing the standard event study using daily returns data in financial economics. This is a workhorse application of event studies. The treatment here assumes knowledge of event studies (Corrado, 2011).

To conduct an event study, you must have a list of firms with associated dates, and you must have returns data for these firms. In order to use the package, you have to place your data into two objects, using certain conventions for the dates and certain conventions for the returns data.

The dates must be stored as a simple `data.frame`. To illustrate this, we use the object *SplitDates* in the package which is used for doing examples.

```
> library(eventstudies)
> data(SplitDates)                          # The sample
> str(SplitDates)                           # Just a data frame

'data.frame':        22 obs. of  2 variables:
 $ name: chr  "BHEL" "Bharti.Airtel" "Cipla" "Coal.India" ...
 $ when: Date, format: "2011-10-03" "2009-07-24" ...

> head(SplitDates)

           name       when
1          BHEL 2011-10-03
2 Bharti.Airtel 2009-07-24
3         Cipla 2004-05-11
4    Coal.India 2010-02-16
5      Dr.Reddy 2001-10-10
6     HDFC.Bank 2011-07-14
```

The representation of dates is a data frame with two columns. The first column is the name of the unit of observation which experienced the event. The second column is the event date.

The second thing that is required for doing an event study is data for stock price returns for all the firms. The sample dataset supplied in the package is named *StockPriceReturns*:

```
> data(StockPriceReturns)                 # The sample
> str(StockPriceReturns)                  # A zoo object

'zoo' series from 2010-07-01 to 2013-03-28
  Data: num [1:720, 1:30] 0.528 -1.731 -0.253 -0.317 -1.277 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:30] "Bajaj.Auto" "BHEL" "Bharti.Airtel" "Cipla" ...
  Index:  Date[1:720], format: "2010-07-01" "2010-07-02" "2010-07-05" "2010-07-06" ...

> head(StockPriceReturns,3)               # Time series of dates and returns.

           Bajaj.Auto      BHEL Bharti.Airtel      Cipla Coal.India   Dr.Reddy
2010-07-01  0.5277396 -1.236944    0.51151007 -0.7578608         NA -0.8436534
2010-07-02 -1.7309383 -1.669938    0.09443763  0.4910359         NA -0.3687345
2010-07-05 -0.2530097 -1.282136    0.80850304  0.1335015         NA  1.7035363
                 GAIL   HDFC.Bank Hero.Motocorp Hindalco.Industries
2010-07-01 -0.04282197 -0.39248572   -1.18437633          -1.0434877
2010-07-02 -1.23903972  0.30627094   -0.11129386           0.7662873
2010-07-05  0.84206478 -0.00261373    0.05442581          -1.7501322
           Hindustan.Unilever        HDFC      ICICI         ITC    Infosys
2010-07-01           2.003034 -0.87769291 -2.4188100 -0.61218636 -0.7577362
2010-07-02          -1.144980  0.01543461 -0.1546239 -0.08587377 -1.4009144
2010-07-05          -1.139454  0.80273702  0.1070728 -0.23156582  0.5116970
           Jindal.Steel Larsen.and.Toubro Mahindra.and.Mahindra Maruti.Suzuki
2010-07-01  -1.67202312        -0.6978939             -1.522933    -1.7145301
2010-07-02   0.27655785        -0.3129893             -2.216658     0.6375233
2010-07-05   0.04872503         0.1174858              1.319498    -0.9488549
               NTPC       ONGC Reliance.Industries        SBI
2010-07-01  0.4009026 -1.4217526          -0.9939750 -1.7660793
2010-07-02  1.1437221  0.2516497          -0.7180857  0.1524172
2010-07-05 -1.1187189 -1.4634859          -0.0608543  0.3239422
           Sterlite.Industries Sun.Pharmaceutical        TCS Tata.Motors
2010-07-01          -3.2309122        -1.64167399 -2.6713224  -1.9983796
2010-07-02          -2.3378487        -0.01709499  1.6476625   0.5294303
2010-07-05          -0.2181026        -0.70629234 -0.5259266  -0.2610968
            Tata.Power  Tata.Steel      Wipro
2010-07-01  0.02297530 -2.22809842 -2.5401116
2010-07-02  0.02297002 -0.02105928  2.9033238
2010-07-05 -0.15323325 -0.69745607  0.8894677
```

The *StockPriceReturns* object is thus a *zoo* object which is a time series of daily returns. These are measured in per cent, i.e. a value of +4 is returns of +4%. The zoo object has many columns of returns data, one for each unit of observation which, in this case, is a firm. The column name of the zoo object must match the firm name (i.e. the name of the unit of observation) in the list of events.

The package gracefully handles the three kinds of problems encountered with real world data: (a) a firm where returns is observed but there is no event, (b) a firm with an event where

returns data is lacking and (c) a stream of missing data in the returns data surrounding the event date.

With this in hand, we are ready to run our first event study, using raw returns:

```
> es <- eventstudy(firm.returns = StockPriceReturns,
+                  event.list = SplitDates,
+                  event.window = 5,
+                  type = "None",
+                  to.remap = TRUE,
+                  remap = "cumsum",
+                  inference = TRUE,
+                  inference.strategy = "bootstrap")
```

This runs an event study using events listed in *SplitDates*, and using returns data for the firms in *StockPriceReturns*. An event window of 5 days is analysed.

Event studies with returns data typically do some kind of adjustment of the returns data in order to reduce variance. In order to keep things simple, in this first event study, we are doing no adjustment, which is done by setting `type` to "`None`".

While daily returns data has been supplied, the standard event study deals with cumulated returns. In order to achieve this, we set `to.remap` to `TRUE` and we ask that this remapping be done using "`cumsum`".

Finally, we come to inference strategy. We instruct eventstudy to do inference and ask for "`bootstrap`" inference.

Let us peek and poke at the object 'es' that is returned.

```
> class(es)

[1] "es"

> str(es)

List of 2
 $ result  : num [1:10, 1:3] 0 -0.52 -1.693 -0.783 -3.541 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:3] "2.5%" "Mean" "97.5%"
 $ outcomes: chr [1:22] "wdatamissing" "wrongspan" "wrongspan" "wrongspan" ...
 - attr(*, "CAR")='zoo' series from -4 to 5
  Data: num [1:10, 1:6] 0 -1.37 -3.1 -1.68 -1.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:6] "6" "9" "16" "20" ...
  Index:  int [1:10] -4 -3 -2 -1 0 1 2 3 4 5
 - attr(*, "event.window")= num 5
 - attr(*, "inference")= logi TRUE
 - attr(*, "inference.strategy")= chr "bootstrap"
```

```
> par(mai=c(.8,.8,.2,.2), cex=.7)
> plot(es)
```

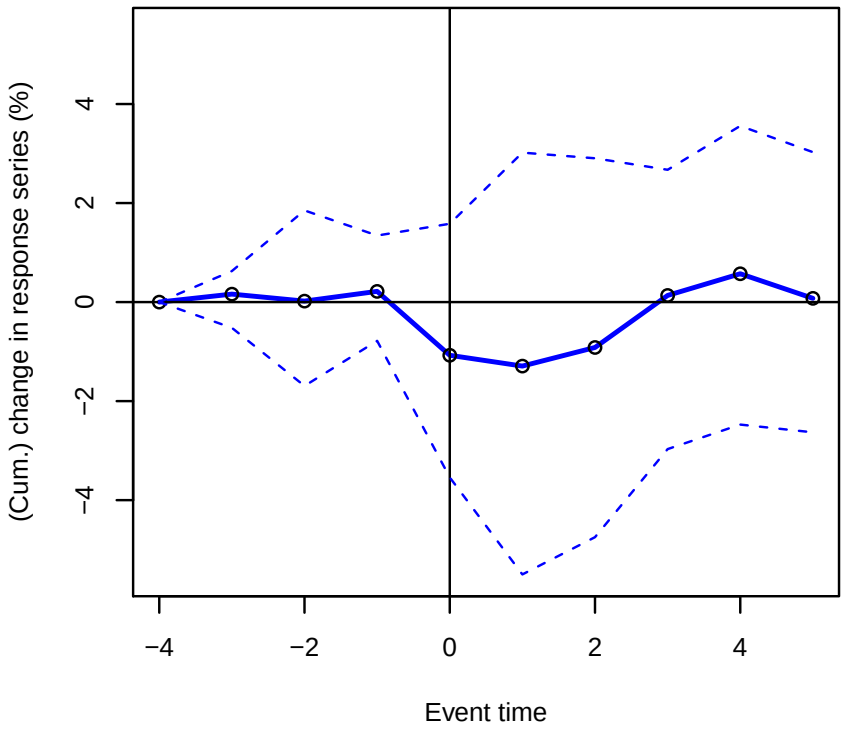

Figure 1: Plot method applied to es object

```
 - attr(*, "remap")= chr "cumsum"
 - attr(*, "class")= chr "es"
```

The object returned by eventstudy is of `class` 'es'. It is a list with two components. Three of these are just a record of the way `eventstudy()` was run: the inference procedure adopted ("`bootstrap`" inference in this case), the window width (5 in this case) and the method used for mapping the data ("`cumsum`"). The two new things are '`outcomes`' and '`result`'.

The vector '`outcomes`' shows the disposition of each event in the events table. There are 22 rows in *SplitDates*, hence there will be 22 elements in the vector '`outcomes`'. In this vector, "`success`" denotes a successful use of the event. When an event cannot be used properly, various error codes are supplied. E.g. "`unitmissing`" is reported when the events table shows an event for a unit of observation where returns data is not observed.

Plot and print methods for the class 'es' are supplied. The standard plot is illustrated in Figure 1. In this case, we see the 95% confidence interval is above 0 and below 0 and in no case can the null of no-effect, compared with the starting date (5 days before the stock split date), be rejected.

In this first example, raw stock market returns was utilised in the event study. It is important to emphasise that the event study is a statistically valid tool even under these circumstances.

Averaging across multiple events isolates the event-related fluctuations. However, there is a loss of statistical efficiency that comes from fluctuations of stock prices that can have nothing to do with firm level news. In order to increase efficiency, we resort to adjustment of the returns data.

The standard methodology in the literature is to use a market model. This estimates a time-series regression $r_{jt} = \alpha_j + \beta_j r_{Mt} + \epsilon_{jt}$ where r_{jt} is returns for firm j on date t, and r_{Mt} is returns on the market index on date t. The market index captures market-wide fluctuations, which have nothing to do with firm-specific factors. The event study is then conducted with the cumulated ϵ_{jt} time series. This yields improved statistical efficiency as $\text{Var}(\epsilon_j) < \text{Var}(r_j)$.

This is invoked by setting `type` to "`marketModel`":

```
> data(OtherReturns)
> es.mm <- eventstudy(firm.returns = StockPriceReturns,
+                     event.list = SplitDates,
+                     event.window = 5,
+                     type = "marketModel",
+                     to.remap = TRUE,
+                     remap = "cumsum",
+                     inference = TRUE,
+                     inference.strategy = "bootstrap",
+                     model.args = list(market.returns=OtherReturns$NiftyIndex)
+                     )
```

In addition to setting `type` to "`marketModel`", we are now required to supply data for the market index, r_{Mt}. In the above example, this is the data object '`NiftyIndex`' supplied from the *OtherReturns* data object in the package. This is just a zoo vector with daily returns of the stock market index.

A comparison of the range of the y axis in Figure 1 versus that seen in Figure 2 shows the substantial improvement in statistical efficiency that was obtained by market model adjustment.

We close our treatment of the standard finance event study with one step forward on further reducing $\text{Var}(\epsilon)$: by doing an 'augmented market model' regression with more than one explanatory variable. The augmented market model uses regressions like:

$$r_{jt} = \alpha_j + \beta_1, jr_{M1,t} + \beta_2, jr_{M2,t}\epsilon_{jt}$$

where in addition to the market index $r_{M1,t}$, there is an additional explanatory variable $r_{M2,t}$. One natural candidate is the returns on the exchange rate, but there are many other candidates.

An extensive literature has worked out the unique problems of econometrics that need to be addressed in doing augmented market models. The package uses the synthesis of this literature as presented in Patnaik and Shah (2010).[1]

[1]The source code for augmented market models in the package is derived from the source code written for Patnaik and Shah (2010).

```
> par(mai=c(.8,.8,.2,.2), cex=.7)
> plot(es.mm)
```

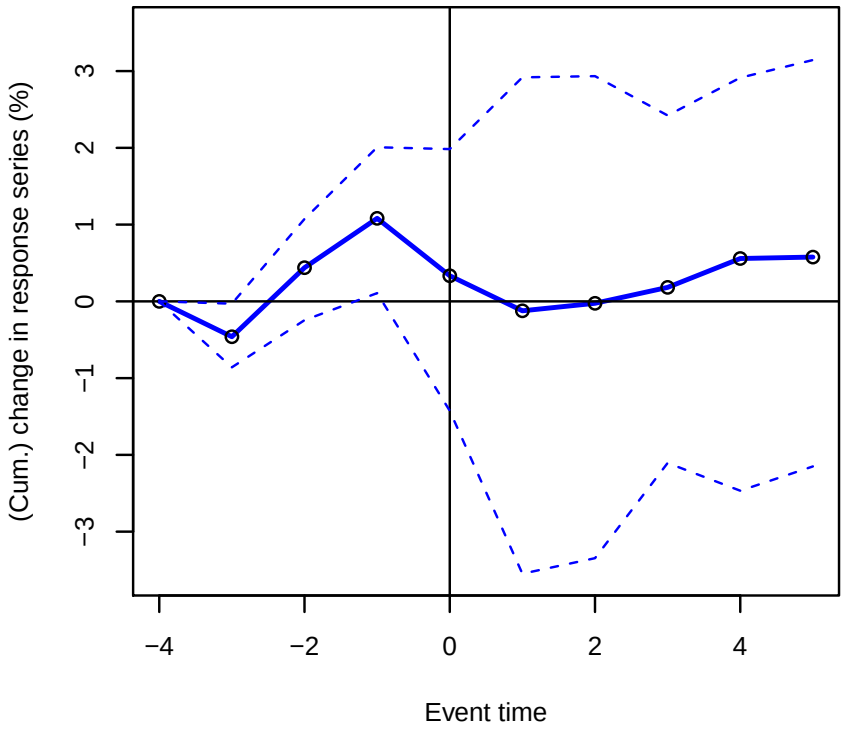

Figure 2: Adjustment using the market model

To repeat the stock splits event study using augmented market models, we use the incantation:

```
> es.amm <- eventstudy(firm.returns = StockPriceReturns,
+                      event.list = SplitDates,
+                      event.window = 5,
+                      type = "lmAMM",
+                      to.remap = TRUE,
+                      remap = "cumsum",
+                      inference = TRUE,
+                      inference.strategy = "bootstrap",
+                      model.args = list(
+                          market.returns=OtherReturns$NiftyIndex,
+                          others=OtherReturns$USDINR,
+                          market.returns.purge=TRUE
+                          )
+                      )
```

Here the additional regressor on the augmented market model is the returns on the exchange rate, which is the slot '`USDINR`' in *OtherReturns*. The full capabilities for doing augmented market models from Patnaik and Shah (2010) are available. These are documented elsewhere. For the present moment, we will use the feature `market.returns.purge` without explaining it.

Let us look at the gains in statistical efficiency across the three variants of the event study. We will use the width of the confidence interval at date 0 as a measure of efficiency.

```
> tmp <- rbind(es$result[5, ],
+              es.mm$result[5, ],
+              es.amm$result[5, ]
+              )[, c(1, 3)]
> rownames(tmp) <- c("None", "MM", "AMM")
> print(tmp["MM", ] - tmp["None", ])
```

```
     2.5%     97.5%
2.1155558 0.4024186
```

```
> print(tmp["AMM", ] - tmp["None", ])
```

```
      2.5%      97.5%
 0.6692943 -0.3898813
```

This shows a sharp reduction in the width of the bootstrap 95% confidence interval from "`None`" to MM adjustment. Over and above this, a small gain is obtained when going from MM adjustment to AMM adjustment.

2 Event study using intra-day data

The 'eventstudies' package is designed to conduct standard event study analysis using not only daily financial returns data but also the intra-day returns data. In this section, we aim

to assess the impact of key interest rate cuts by Reserve Bank of India (RBI) on value of the Indian banks that have the highest weight in the Bank Nifty Index. To conduct the intra-day event study, the package provides a sample data frame, *RateCuts*, which provides associated dates and time stamps of the event analysed for each corresponding bank. In addition to the dataframe with the event list, the package includes *AggregateReturns*, a *zoo* object for intra-day stock price returns of all included banks.

```
> data(AggregateReturns) # Data used
> data(RateCuts)
> data(IndexReturns)
```

The abnormal returns are estimated using *marketModel* on stock price returns provided in *AggregateReturns*. The regressor used in the market model is the intra-day returns on CNX NIFTY 50 index. The intra-day returns data for CNX NIFTY 50 is provided in *IndexReturns*. Further descriptions of the data sets used in this exercise are provided in the package.

```
>                                          # IntraDay eventstudy
>                                          # For 35 minutes pre and post event
>
> intraday.es <- eventstudy(firm.returns = AggregateReturns,
+                           event.list = RateCuts,
+                           event.window = 35,
+                           type = "marketModel",
+                           to.remap = TRUE,
+                           remap = "cumsum",
+                           inference = TRUE,
+                           inference.strategy = "bootstrap",
+                           model.args = list(market.returns=IndexReturns)
+                           )
```

The plot to study the intra-day CARs is illustrated in Figure 3. The abnormal returns are averaged across multiple events for multiple banks to isolate event-related fluctuations and thus, obtain statistically significant results from event study analysis. The 95% confidence interval is above 0, implying a significant impact of the RBI announcements on stock price returns of these banks.

```
> par(mai=c(.8,.8,.2,.2), cex=.7)
> plot(intraday.es)
```

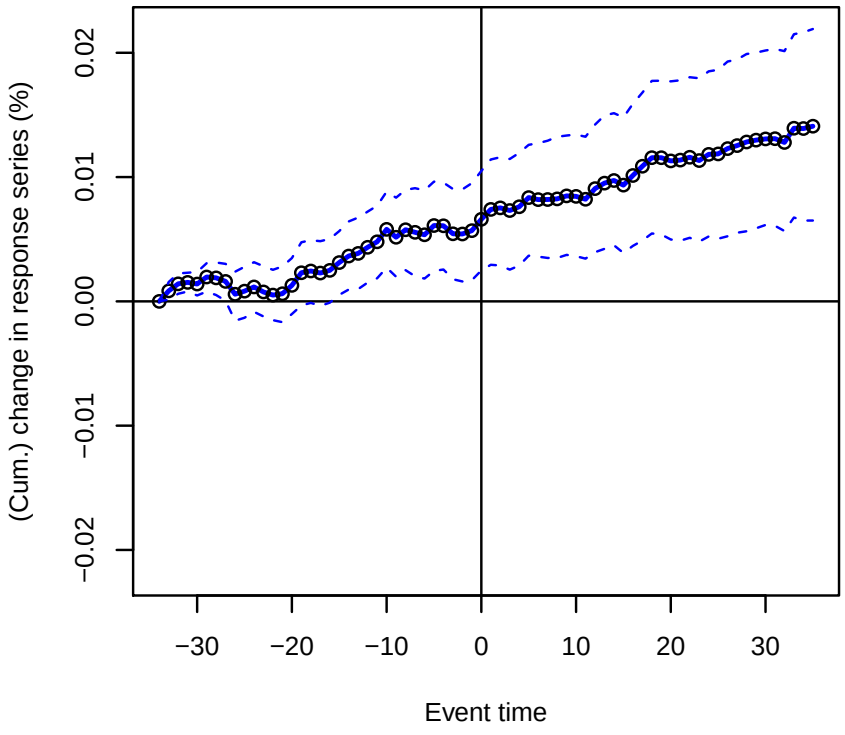

Figure 3: Intraday event study plot

References

Corrado CJ (2011). "Event studies: A methodology review." *Accounting and Finance*, **51**, 207–234.

Patnaik I, Shah A (2010). "Does the currency regime shape unhedged currency exposure?" *Journal of International Money and Finance*, **29**(5), 760–769.