An Introduction to Events

Will Lowe
MZES, University of Mannheim

January 7, 2012

1 Introduction

The `events` package takes political event data in the form generated by KEDS (Schrodt et al., 1994; Gerner et al., 1994). For this vignette we use the Reuters-derived event chronology from the collapse of Yugoslavia, focusing on Serbian and Bosnian interactions in the period in 1991 and 1995. The events in this event data are coded according to the WEIS event scheme (McClelland, 1978). In the following sections we perform a typical set of data manipulations; we load and clean a set of event data, restrict it to actors and period of interest, apply a scale to the raw events, aggregate to make a time series and plot the results. The package does not current contain function for the analysis of event data because once the data is finally in a regular time series format, other packages can be used to analyse it. The package provides the link between raw output from an event data extraction system such as KEDS/TABARI and a set of regularly spaced time series.

1.1 Data Loading and Cleaning

A version of the Balkans data is built into the package. Here we load and summarise it.

```
> data("balkans.weis")
> summary(balkans.weis)
```

```
72953 events involving 295 sources and 296 targets
from Sun Apr 02, 1989 to Thu Jul 31, 2003
```

An event data set can be constructed from text file event data output using the `read_keds` function. And event data set is essentially a data frame with column names `date`, `source`, `target`, and `code`.

```
> head(balkans.weis)
```

```
        date source target code            desc
1 1989-04-02    YUG    KSV  224          (RIOT)
2 1989-04-04    YUG ETHALB  212 (ARREST PERSON)
3 1989-04-07    ALB ETHALB  224          (RIOT)
4 1989-04-08 ETHALB    KSV  123   (INVESTIGATE)
5 1989-04-10    PRK    YUG  032         (VISIT)
6 1989-04-10    YUG    PRK  033       (RECEIVE)
```

Subsequent columns of event label, shown above, and matching phrase from the original text, not shown above, are optional.

Duplicated stories are a common type of information extraction error. We can prefilter the events by removing all instances of the same pair of actors experiencing the same event on the same date using the on-a-day filter

```
> dd1 <- one_a_day(balkans.weis)
```

This can also be applied as part of the `read_keds` function.

1.2 Actor Filtering

In the next step we filter out actors whose interactions are not of interest. A complete list of actors is given by `actors` function

```
> head(actors(dd1))

[1] "AFG"    "AFR"    "ALB"    "ALBMED" "ALG"    "AMN"
```

The functions `sources` and `targets` list actor codes in the corresponding roles, and `codes` lists all the codes that are used.

We will focus on actors identified in the data as Serbia 'SER' and the Serbian military 'SERMIL', and Bosnia 'BOS' and the Bosnian military 'BOSMIL'

```
> dd2 <- filter_actors(dd1, fun=spotter("SER", "SERMIL", "BOS", "BOSMIL"))
```

The `filter_actors` function takes two arguments, an event data set and a filter function, and returns a filtered event data set. The filter may be any function that returns TRUE for things that are of interest and FALSE otherwise. Here we have used a convenience function `spotter`, which creates a function that returns TRUE for any exact matches of its arguments.

The function takes an optional `which` argument which can be used to specify that the filtering should apply to 'source', 'target' or 'both', which is the default.

We would like to treat the Serbian and Bosnian actors identified in the previous step as equivalent and refer to then for convenience as 'ser' and 'bos' respectively. We do this by aggregating actor codes:

```
> actor.agg <- list(ser=c("SER", "SERMIL"), bos=c("BOS", "BOSMIL"))
> dd3 <- map_actors(dd2, fun=actor.agg)
```

Here we specify the mapping from new to old actor codes as a list and pass it to the mapping function. We could also have written a function that for any object returned its new name, in the same style as the filter function in the previous section. For example

```
actor.aggregator <- function(oldname){
    newname <- NA
    if (oldname %in% c("SER", "SERMIL")) newname <- "ser"
    if (oldname %in% c("BOS", "BOSMIL")) newname <- "bos"
    return(newname)
}
```

would work, but it's rather longwinded.

1.3 Temporal Restriction

We will focus on the period between January 1991 and December 1995

```
> dd4 <- filter_time(dd3, start="1991-01-01", end="1995-12-30")
```

The optional `start` and `end` parameters may be anything that can be converted into a `Date` object.

The new data set is considerably smaller than before

```
> summary(dd4)

853 events involving 2 sources and 2 targets
from Fri Mar 15, 1991 to Tue Dec 26, 1995
```

1.4 Scaling

Scales are mappings from event codes to real numbers. You can create your own event code by constructing a headerless csv file with event codes in the first column and numbers in the second column, and reading it in with the `make_scale` command. This is a thin wrapper around the `read.csv` function.

Here we will use the extended Goldstein scale bundled with the package (Goldstein, 1992)[1]. This maps WEIS event codes onto a number representing level of conflict or cooperation.

```
> data("weis.goldstein.scale")
> summary(weis.goldstein.scale)
```

```
Scale name: goldstein
Unrecognized event codes: NA
109 event codes assigned scores between -10 and 10
```

When we apply the scale to an event data set a column is added with the same name as the scale

```
> dd5 <- add_eventscale(dd4, weis.goldstein.scale)
> head(dd5)
```

```
           date source target code             desc goldstein
897  1991-03-15    ser    ser  094       (CALL FOR)      -0.1
1425 1991-07-04    ser    ser  212  (ARREST PERSON)      -4.4
2223 1991-09-19    ser    ser  223 (MIL ENGAGEMENT)     -10.0
2224 1991-09-19    ser    ser  223 (MIL ENGAGEMENT)     -10.0
2341 1991-09-25    ser    ser  081 (MAKE AGREEMENT)       6.5
2342 1991-09-25    ser    ser  081 (MAKE AGREEMENT)       6.5
```

1.4.1 Score Aggregation

The final step is to aggregate quantities of interest into a regular time series for each directed pair of actors. Here we construct a typical dyad set using the summed scored event counts per week:

```
> dyads <- make_dyads(dd5, scale="goldstein", unit="week", monday=TRUE,
+                 fun=sum, missing.data=0)
```

We are asserting here that weekly counts should start on a monday, that they should be summed rather than e.g. averaged, and that weeks with no events observed should be given score zero. Note that this is only an example; these are not necessarily sensible setting for actual applications.

Alternative aggregation units are 'day', 'month', 'quarter', and 'year'. The fun parameter should be any function that will transform a numerical vector into a scalar.

The output of `make_dyads` is a list of directed dyad time series. All combinations of actors are constructed, so it is a good idea to filter and aggregate actors before calling the function. The naming scheme for the dyads is concatenation with a period: `dyads$ser.bos` is the temporally aggregated sequence of summed scores with the 'ser' actor as source and 'bos' the target, `dyads$bos.ser` is the reverse direction, and `dyads$ser.ser` is the activities internal to the 'ser' actor.

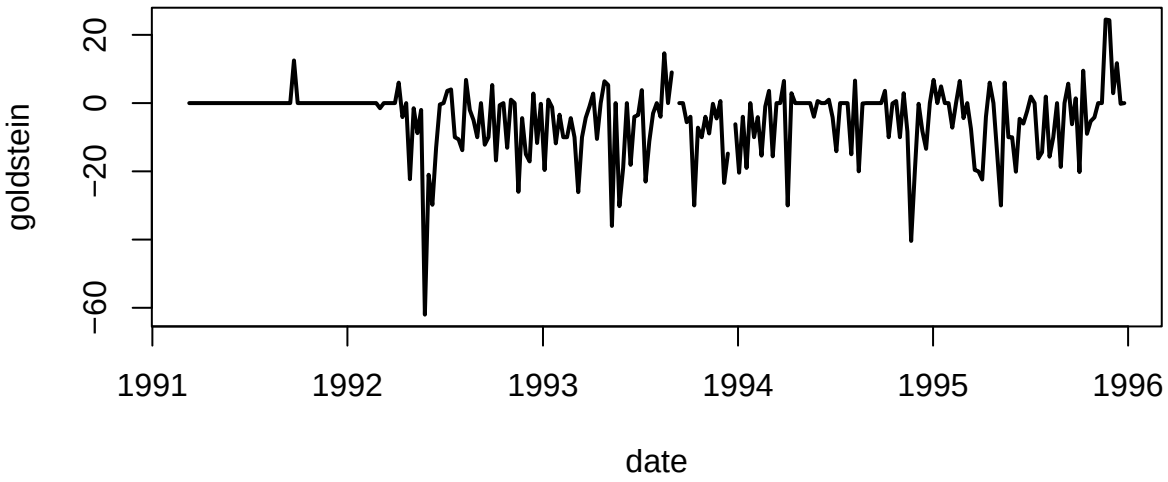
```
> tail(dyads$ser.bos)
```

```
          date goldstein  n
246 1995-11-20      24.5 11
247 1995-11-27      24.3  5
248 1995-12-04       2.9  2
249 1995-12-11      11.7 10
250 1995-12-18      -0.2  1
251 1995-12-25       0.0  0
```

[1] These codes are taken from http://web.ku.edu/ keds/data.dir/KEDS.WEIS.Codes.txt

The directed dyad can be treated like a regular time series:

```
> with(dyads$ser.bos, plot(goldstein ~ date, type="l", lwd=2))
```



There are a few gaps in this series. This is because the scale does not cover all the events that occur in the event data. We can investigate this further with

```
> scale_coverage(weis.goldstein.scale, dd5)

Scale goldstein does not cover codes:
[1] "130" "199" "204"
```

1.4.2 Count Aggregation

If `scale` is NULL a sequence then directed dyadic event count streams are created instead of scaled scores. This will generate an event count for each distinct event code and each temporal unit. Sometimes it is helpful to aggregate code before constructing these count streams. Here we aggregate them into four categories: verbal and material cooperation, and verbal and material conflict

```
> evts <- codes(dd4)
> event.agg <- list(
+     coop.verb=grep("02.|03.|04.|05.|08.|09.|10.", evts, value=TRUE),
+     coop.mat=grep("01.|06.|07.", evts, value=TRUE),
+     conf.verb=grep("11.|12.|13.|14.|15.|16.|17.", evts, value=TRUE),
+     conf.mat=grep("18.|19.|20.|21.|22.", evts, value=TRUE)
+ )
> dc1 <- map_codes(dd4, fun=event.agg)
```

Like the other aggregation function, `map_codes` function in the final line takes a list or a function to map old event codes to new ones. We start by using the `codes` function to list all the event codes that are used in the data. WEIS is a two level scheme that by convention indicates the upper level code category in first two digits and subcategory in remaining digits. Here, we use `grep` to identify all the codes in "01", "06", and "07" at any level and assign them to a new material cooperation category `mat.coop`.

```
> dyad.counts <- make_dyads(dc1, scale=NULL, unit="week", monday=TRUE,
+                 fun=sum, missing.data=0)
> tail(dyad.counts$ser.bos)

          date conf.mat conf.verb coop.mat coop.verb
246 1995-11-20        1         1        2         7
247 1995-11-27        0         0        0         5
248 1995-12-04        0         0        0         2
249 1995-12-11        0         2        0         8
250 1995-12-18        0         0        0         1
251 1995-12-25        0         0        0         0
```

References

Gerner, D. J., Schrodt, P. A., Francisco, R. A., and Weddle, J. L. (1994). The analysis of political events using machine coded data. *International Studies Quarterly*, 38(1):91–119.

Goldstein, J. S. (1992). A conflict-cooperation scale for WEIS events data. *Journal of Conflict Resolution*, 36(2):369–385.

McClelland, C. (1978). World Event/Interaction Survey (WEIS) Project, 1966–1978. http://dx.doi.org/10.3886/ICPSR05211.v3.

Schrodt, P. A., Davis, S., and Weddle, J. (1994). KEDS – A program for machine coding of events data. *Social Science Computer Review*, 12(3):561–554.