Data analysis by regular expressions and the analysis of event sequences

Colin Goodall

AT&T Labs

Text string matching and manipulation by regular expressions is pervasive in contemporary computer languages, ranging from shell programming, to SQL, to java, to python (Lutz, 1996, 2001) and perl (Wall et al, 1991, 1996, 2000). Indeed, the advanced use of regular expression in perl is one of its strongest features. UNIX utilities for regular expressions include sed, awk, egrep, and tr. In statistical software, with its emphasis on numerical data, regular expression analysis of string data has a lesser role; for example grep appeared in S (Becker et al , 1988) in rudimentary form.

Regular expression engines (Friedl, 1997, 2002) are among the most powerful - most optimized - most used of modern computational tools. This paper considers harnessing regular expressions to the cause of data analysis, and, specifically, to the analysis of event sequences.

An event sequence is an usually-time related set of events of differing types in time order. In telecommunications, this might be a sequence of telephone calls to different terminating locations with various durations at different times of day. In customer care, events may include billing, payments, change in service, and inbound and outbound customer contact. Additionally, an event sequence might include events for elapsed times between successive events, or, we might introduce markers into the sequence at specific times, such as 1st day of a month, or start of a bill cycle.

We represent each type of event by a distinct character, or possibly subsequence of characters, after binning continuous variables to categorical form. The most convenient set of characters are the 95 printable ASCII characters, but the full set of single-byte characters is available. Further, the multi-byte characters with UCS, UTF-8, Unicode encoding provide a potentially huge set, though limited by the implementation and interface to the regular expression engine (which may be single-byte). The use of character classes, such as *, [aeiou], [A-G], or \pAlnum, in regular expressions allow projection of a rich to a simple representation of types.

Some of the simplest operations using regular expression to analyze event sequences are:

m/^ c[^ A-Z]*c/	match two type c events, with no intervening type A-Z events
tr/0-9/./	change each digit to a period
tr/a-z//d	delete each lower case character

Typical uses are to simplify sequences and to count matches.

We describe a system for data analysis by regular expressions, implemented in Splus/R and perl/UNIX, with the following components:

- Data encoding. The multivariate data contain one observation for each event. These data are coded to categorical form as necessary. Each event is a concatentation of categories; a table maps each combination of categories for selected variables to an event type.

- Sequence creation. Each event includes a variable for time, and one or more key fields. Event type is represented by one or more characters. The sequence for each key field or fields comprises the event types in temporal order. Sequences are output to a text file with one line for each sequence.

- Regular expression applications include translation, substitution, and matching/counting.
- Summarization tools include such utilities as run-length encoding and common-subsequence detection.
- Sequence visualization is both graphical and text-based.

The system provides for rapid sequence creation and use of multiple match criteria in quick succession. The set of sequences may be very much smaller than the table of events.

We conclude with a discussion of classes of data analysis problems amenable to coding as regular expressions, and other problems that are not, and consider the generalization of use of regular expressions to use of parsers.

References

Becker, R.A., Chambers, J.M, and Wilks, A.R. (1988). *The New S Language*. London, Chapman and Hall.

Friedl, J.E.F. (1997, 2002). *Mastering Regular Expressions*. Sebastopol, CA, O'Reilly and Associates.

Lutz, M. (1996, 2001). *Programming Python*. Sebastopol, CA, O'Reilly and Associates.

Wall, L., Christiansen, T. and Orwant, J. (1991, 1996, 2000). *Programming Perl*. Sebastopol, CA, O'Reilly and Associates.