

Vorlesung "Empirische Bewertung in der Informatik"

Freie Universität Berlin, Institut für Informatik, Arbeitsgruppe Software Engineering
Prof. Dr. Lutz Prechelt, Stephan Salinger
Übungsblatt 3 SS 2007 zur Übung 2007-05-14

Aufgabe 3-1: (Elementare Datenanalyse in R erlernen, Teil 2)

In dieser Aufgabe wollen wir einige weitere Funktionen zur Datenanalyse von R kennen lernen, insbesondere solche zur Behandlung von Zeitangaben und zur grafischen Darstellung.

Wir arbeiten weiter mit den Datensätzen `jikes.tsv`, `junit.tsv`, `zile.tsv` aus dem letzten Übungsblatt und dessen Teilmenge `junit200.tsv`. Auch hier ist jeder Schritt mit wenigen Zeilen R-Code möglich.

- a) Der Zeitstempel `tstamp` in Ihren eingelesenen Datensätzen (z.B. `junit`) liegt bisher als String vor. Um ihn für Operationen in R nutzbar zu machen, muss er in ein `POSIXct`-Objekt verwandelt werden.
 - o Lesen Sie `?DateTimeClasses` und den Artikel über *Date-Time Classes* von *Brian D. Ripley and Kurt Hornik* in den *R-News Volume 1/2, June 2001* (http://cran.r-project.org/doc/Rnews/Rnews_2001-2.pdf)
 - o Fügen Sie die entsprechend gewandelte Variable als `tstamp2` dem Datensatz zu. (Das geht am einfachsten mit `junit$tstamp2 = ...`)
 - o Sehen sie sich die Elemente von `tstamp2` an und machen Sie sich klar, wie es zu der vorliegenden Darstellung der Objekte am Bildschirm kommt. Verwenden Sie hierzu `mode` und `class`.
 - o Benötigte Funktionen: `as.POSIXct`, `strptime`
- b) Fügen Sie den Zeitstempel nun außerdem noch als `tstamp3` im Unix-'time()'-Format (also als Anzahl Sekunden seit 1.1.1970) zu. Das ist oft am praktischsten, wenn man mit Zeiten rechnen möchte.
 - o Benötigte Funktion: `as.numeric`
 - o Hintergrund: Vergleichen Sie die Ergebnisse von `diff(junit200$tstamp2[1:2])` und `diff(junt200$tstamp3[1:2])`
 - o Verwenden Sie `print.default`, `class` und `?difftime`, um den Unterschied zu verstehen.
- c) Untersuchen Sie die zeitlichen Schwankungen der Aktivitäten der Entwickler im Tages- und Wochenverlauf.
 - o Extrahieren Sie den Wochentag aus `tstamp2` und fügen Sie ihn ihrem `data.frame` als Variable `wday` hinzu.
 - o Extrahieren Sie die Stunde aus `tstamp2` und fügen Sie den Wert ihrem `data.frame` als Variable `hour` hinzu.
 - o Benötigte Funktionen: `as.POSIXlt` (Sie extrahieren jeweils ein Element aus dem Resultat)
 - o Betrachten Sie nun eine Aufstellung der Gesamtanzahl von Aktivitäten zu jeder Stunde (unabhängig vom Datum) und an jedem Wochentag (unabhängig vom Datum).
 - Wo liegen die Minima und Spitzen? Wie viel kleiner oder größer sind sie als der Mittelwert? Was schließen Sie aus dem Verlauf?
 - o Benötigte Funktionen: `as.vector`, `summary`, `table`

- d) Veranschaulichen Sie sich die eben berechneten stunden- und wochentageweisen Aktivitäten
- o Benötigte Funktion: `plot`
- e) `plot` ist (wie z.B. auch `print`, das implizit immer auf das Ergebnis jedes Kommandos angewendet wird) eine objektorientierte Funktion. Es gibt separate Versionen von `plot` für viele verschiedene Arten von Objekten.
- o Verschaffen Sie sich mittels `methods(plot)` einen Überblick und lesen Sie die Dokumentation von drei Ihnen interessant erscheinenden `plot`-Funktionen nach.
 - o Wir haben eben zur Wochentage-Darstellung implizit `plot.table` benutzt. Es ginge ähnlich aber auch mit `plot.factor`. Probieren Sie dies aus.
 - o Benötigte Funktionen: `factor`, `plot`
 - o Weisen Sie für die Wochentage auch richtige Kurznamen zu, indem Sie das `labels`-Argument von `factor` verwenden.
- f) Verschaffen Sie sich einen Überblick über die Verteilung der Anzahl der in einem Schritt zugefügten bzw. gelöschten Zeilen (Variablen `lines_add` und `lines_del`) pro Entwickler
- o Boxplots: Verwenden Sie `bwplot` mit einer Formel der Art `developer~log(lines_add+1,2)`. Benutzen Sie auch die Argumente `varwidth` und `box.ratio`, um die Darstellung zu verbessern.
 - o Gibt es Entwickler mit auffällig häufig besonders vielen zugefügten bzw. gelöschten Zeilen oder besonders hoher oder geringer Streuung der Größen?
 - o Vergleichen Sie die Dicke der Boxen mit der Veranschaulichung der Jobanzahlen durch `plot(table(lpq$developer))`
 - o Benötigte Funktionen: `library(lattice)`, `bwplot`, `log`, `plot`, `table`, `?panel.bwplot`
- g) Wiederholen Sie die gleiche Analyse mit einer Aufteilung der Daten nach Dateityp anstatt nach Entwickler.
- o Benötigte Funktionen: `bwplot`, `factor`, `log`
- h) Vollziehen Sie die Ergebnisse für Entwickler an einem Dichteplot nach. Dieser stellt die Häufigkeitsverteilung durch eine Kurve dar.
- o Die Formel lautet hier z.B. ungefähr `log(pages+1,2)|developer`; verwenden Sie außerdem das Argument `width=1`. Lesen Sie darüber nach und probieren Sie die Wirkung aus.
 - o Benötigte Funktionen: `densityplot`, `log`
- i) Lesen Sie nun zumindest die Abschnitte 1, 2.1 und 3.2.2 im Artikel `MocFieHer02` („Two Case Studies of Open Source Software Development: Apache and Mozilla“, zu finden auf der Webseite der Veranstaltung) um einen Eindruck über die dort gemachten Untersuchungen zu bekommen.
- o Machen Sie sich klar, was die Werte auf der x-Achse in Fig. 1 im Abschnitt 3.2.2 bedeuten.
 - o Führen Sie mit unseren Daten eine Analyse analog zu der in Abschnitt 3.2.2 in Fig. 1 gemachten durch. Fertigen Sie also entsprechende Bilder für unsere Datensätze an und interpretieren Sie die Ergebnisse.
 - o Verwenden Sie dabei eine lineare Skala auf der x-Achse.
 - o Jedes Ihrer drei Bilder muss 3 Kurven enthalten.

- o Benötigte Funktionen: `cumsum`, `length`, `lines`, `plot`, `sum`, `tapply`
- Senden Sie Ihre Ergebnisse wieder **als Attachment bis Montag, den 14.05.2007, 9:00 Uhr, an salinger[klam*mer*af*fe]inf.fu-berlin.de** und bringen Sie einen Ausdruck in die nächste Übung mit.

Das Attachment *muss* den Namen `<Familienname.Vorname>_ue3.R` haben.

Die Email muss folgenden Betreff (Topic) haben:

[UE Empir07-3][`<Familienname.Vorname>`]