

## Vorlesung "Empirische Bewertung in der Informatik"

Freie Universität Berlin, Institut für Informatik, Arbeitsgruppe Software Engineering  
Prof. Dr. Lutz Prechelt, Stephan Salinger  
Übungsblatt 2            SS 2007            zur Übung 2007-05-07

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

In dieser Aufgabe wollen wir einige grundlegende Funktionen zur Datenmanipulation und Datenanalyse von R kennen lernen.

Bitte lösen Sie jeden Teilschritt so, dass das Ergebnis sich nach Ausführung direkt ablesen lässt, ohne weitere Handarbeit. Das ist in allen Fällen mit wenigen Zeilen R-Code möglich (1-5). Die komplette Lösung ist *erheblich* kürzer als die Aufgabenstellung. Lesen Sie zu jeder Operation in der Dokumentation *mehr* nach, als Sie für die gegebene Aufgabe benötigen, um ein solides Grundwissen von R zu erlangen.

Meist sind die zur Lösung benötigten Funktionen angegeben. Sie müssen diese allerdings nicht unbedingt verwenden; häufig gibt es auch andere Lösungswege.

Es geht in der Aufgabe im wesentlichen um R, nicht um die behandelten Daten; erwarten Sie also keine spektakulären Analyseergebnisse, sondern freuen Sie sich an der Eleganz der Arbeitsweise. Klären Sie Rückfragen zu Vorgehen und Stil über die Mailingliste.

- a) Beschaffen Sie sich die Dateien `jikes.tsv`, `junit.tsv`, `zile.tsv` und `junit200.tsv` von der Vorlesungswebsite (`data.zip`).
  - Es handelt sich um Ausschnitte aus den CVS-Daten der Projekte Jikes (Java-Compiler, [jikes.sourceforge.net](http://jikes.sourceforge.net)), JUnit (Framework für Unit-Tests in Java, [www.junit.org](http://www.junit.org)) und Zile (Emacs-Clone, [zile.sourceforge.net](http://zile.sourceforge.net)).
  - Die Daten in den Dateien sind tabulatorsepariert.
  - Jede Zeile repräsentiert einen Checkin-Vorgang im CVS. Angegeben sind der Dateiname *file*, der Zeitpunkt *tstamp*, der Aufrufer *developer*, die Versionsnummer der Datei *version*, die Anzahl von Zeilen, die gegenüber der Vorversion zugefügt bzw. entfernt wurden (*lines\_add*, *lines\_del*) sowie der Kommentar des Checkin *description*.
  - Die Datei `junit200.tsv` enthält nur die ersten 200 Zeile von `junit.tsv`. Sie sollten zur Zeitersparnis beim Erarbeiten Ihrer Lösung zuerst nur mit dieser Teilmenge arbeiten, bis Sie herausgefunden haben, wie es geht, und erst dann auf die vollen Datensätze übergehen. In der Übung sollten Sie dann jeweils Aussagen über die vollständigen Datensätzen machen können.
- b) Lesen Sie die Datei ein und speichern Sie sie als Datensatz (`data.frame`) in einer Variablen.
  - Benötigte Funktion: `read.table` oder `read.delim`.
  - Sorgen Sie mit einem `as.is`-Argument dafür, dass die Zeitstempel *tstamp*, die Namen der Entwickler *developer* und die Namen der Dateien *file* nicht in einen Faktor verwandelt werden.
    - Lesen Sie *gründlich* über Faktoren nach: `?factor`  
Diese benutzt man sehr häufig. (Einen Faktor zu erzeugen ist z.B. eine bequeme Art, die Anzahl verschiedener Werte in einem Vektor zu ermitteln, ohne gleich deren Häufigkeit zu zählen, wie es `table` tun würde. Außerdem sind Faktoren durch Zahlen dargestellt, was bei der Verarbeitung gegenüber einer Stringdarstellung viel Speicherplatz und Rechenzeit sparen kann.)
    - Fügen Sie deshalb *developer* und *file* nochmals explizit zusätzlich zu Ihrem Datenrahmen als Faktoren *developerf* und *filef* zu (mittels `cbind` plus Zuweisung an `names` oder einfacher `per $`).

- Entfernen Sie die Spalte *description* aus dem Datensatz (mittels Zuweisen von `NULL` oder Funktion `subset`).
  - Der entstandene Datensatz sollte nun 8 Variablen aufweisen. Verschaffen Sie sich einen ersten Überblick.
    - Oder haben Sie diesen schon, weil Sie zu allererst einen ausführlichen Blick in die Textdatei geworfen haben? Gut!
    - Benötigte Funktionen: `names`, `nrow`, `str`, `summary`
  - Definieren Sie sich nun eine eigene Funktion `myread.csvdata`, welche einen Pfad zu einer Datei mit solchen tabseparierten CVS-Daten übergeben bekommt und einen Datensatz mit den oben beschriebenen Eigenschaften zurück gibt.
    - Lesen Sie über Funktionen nach: `help("function")`
- c) Beantworten Sie Fragen zu den beteiligten Entwicklern:
- Wie viele unterschiedliche Entwickler haben überhaupt Änderungen (einzelne Speicherungen in das CVS) vorgenommen? (Funktionen: `levels`, `length`)
  - Wie viele Änderungen wurden jeweils von den 5 eifrigsten Entwicklern vorgenommen? (Funktionen: `sort`, `table`)
  - Wie viel Prozent der Dateien hat jeder einzelne Entwickler zumindest einmal bearbeitet?  
Hier gibt es zwei Ansätze: (1) Sie zählen für jeden Entwickler die Anzahl der Dateien, die bei ihm vorkommen (`tapply`, `levels`, `length`, `factor`, `sort`) oder (2) Sie machen eine Häufigkeitstabelle (Entwickler x Datei) und zählen die Nicht-Null-Einträge für jeden Entwickler (`table`, `apply`, `length`, `levels`, `sum`, `sort`).  
Methode 1 ist geringfügig kürzer hinzuschreiben und läuft für große Datenmengen mit sehr viel weniger Speicheraufwand durch; eine Häufigkeitstabelle (Methode 2) erlaubt dafür aber auch noch sehr viele andere Abfragen.
    - Was fällt Ihnen am Ergebnis bzgl. der JUnit-Daten auf?
- d) Machen Sie sich klar, dass der vorherige und auch alle folgenden Analyseschritte mehrmals durchzuführen sind, wobei sich die Analysen ausschließlich im benutzten Datensatz unterscheiden.
- Sie sollten deshalb die endgültigen Versionen aller Kommandos, die Sie im Verlauf der Analyse entwickelt haben, in einer Textdatei speichern, damit Sie die Analyse bei Bedarf leicht wiederholen können.
  - Notieren Sie dort zusätzlich auch Zweck/Thema, Autor, Datum, R-Version und das Verzeichnis, in dem die Daten zu finden sind, sowie außerdem die Ergebnisse der einzelnen Analyseschritte der Aufgabe. Das Kommentarzeichen von R ist ein `#`.
  - Sie können eine solche Datei vollautomatisch mittels des Kommandos `source` komplett wieder ausführen. Starten Sie eine neue R-Sitzung („Workspace speichern?": Nein) und probieren Sie dies aus.
  - Was ist anders als bei Ausführung einzeln von Hand?  
Lesen Sie darüber in „Introduction to R" (`help.start`) im Unterabschnitt „R commands" von Abschnitt „Introduction and preliminaries".  
Zur Ausgabe von Erläuterungen und skalaren Ergebnissen benutzen Sie `cat`, für komplexere Ergebnisse `print`.
  - Es empfiehlt sich ferner, alle Schritte in eine Funktion `myanalyze.csvdata` zu schreiben, die mit dem Datensatz parametrisiert ist.
  - Lesen Sie zusätzlich über `with`, `attach`, `detach` nach, womit Sie sich die Schreibweise der Analysen erheblich vereinfachen können.
- e) Beantworten Sie nun Fragen zu den im CVS verwalteten Dateien (und machen Sie sich klar, dass viele davon analog zu den obigen sind):
- Wie viele unterschiedliche Dateien wurden geändert (befinden sich also laut Ihres Datensatzes im CVS)?
  - Welches sind die 5 am häufigsten gespeicherten Dateien? (`table`, `sort`)

- Wie viele Dateien wurden genau einmal, zweimal, dreimal, fünfmal bzw. zehnmal geändert? (`table`, `order` und geschickte Indizierung)
- Wie viele Dateien gibt es die von genau einem, genau zwei, u.s.w. bis genau zehn Entwicklern geändert wurden? (`tapply`, `levels`, `length`, `factor`, `table`) Versuchen Sie dasselbe vergleichsweise mit `sapply` statt `tapply`.
- Wir wollen nun die Dateien getrennt nach Dateityp untersuchen.
  - Dazu erweitern wir unseren Datensatz um eine Spalte `filetypef`, die nur den Dateityp enthält. Diesen können wir aus dem Feld `file` extrahieren. (`sub("^.*\\.([a-zA-Z0-9]*)$", "\\1", ...)`)
  - Wie viele Dateien gibt es von jedem Typ? (`sort`, `table`)
  - Für jeden Dateityp: wie viele Male wurden Dateien dieses Dateityps im arithmetischen Mittel geändert? (`sapply`, `levels`, `mean`, `table`)
  - Schaffen Sie es, dass in derselben Tabelle über und unter dem Mittelwert das Minimum und Maximum jedes Typs und zusätzlich in der ersten Zeile die Anzahl der unterschiedlichen Dateien des Typs angegeben ist?  
(Hinweis: Die Funktion im `sapply` darf auch einen Vektor liefern. Diesmal wäre das Problem mit `tapply` nicht so elegant zu lösen.)
  - Schaffen Sie es, dass die Tabelle am Rand auch passend beschriftet ist? (Zuweisung an `row.names`)
  - Und dass die Tabelle nach den Mittelwerten sortiert ist? (`order`)
- f) Gibt es für die Aktivitäten an den Dateien eine 80:20-Regel (Auf die am häufigsten geänderten 20% der Dateien entfallen 80% der Änderungen)? (`length`, `levels`, `sort`, `quantile`, `cumsum`, `max`)
- g) So, diese Aufgaben waren jetzt zwar ganz gut geeignet, um R kennen zu lernen, stellen aber nicht gerade die beste Analyse des Datensatzes dar. Überlegen Sie sich deshalb jetzt selber mindestens zwei *interessante* Fragen, die man aus den Daten beantworten kann und drücken Sie die Antwort mit R aus.

Senden Sie diese Textdatei **als Attachment bis Montag, den 07.05.2006, 09:00 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>_ue2.R` haben.

Die Email muss folgenden Betreff (Topic) haben:

[UE Empir07-2][<Familienname.Vorname>]