Freie Universität Berlin, Software Engineering Research Group
Prof. Dr. Lutz Prechelt, Franz Zieris

Practice sheet 4 **Practice R, part 3; Survey: Topic** due on 2017-05-14

Preface

The first half of this practice sheet is the third (and last) part that deals with analyzing the CVS data. The analyses here are a bit more sophisticated, but still not too hard.

The second half marks the beginning of the main part of the tutorial: Conducting an entire study from A to Z. You will work in small groups on this.

Task 4-1: More Advanced Data Analyses

Answer the following questions using short R programs *for each* of the three projects JUnit, Jikes, and Zile:

a) Counting files:

1. How many different files were changed, i.e. how many files' names are mentioned in the CVS log excerpts? (`length`, `levels`)
2. Which are top 5 of the most often changed files and how often were they changed? (`table`, `sort`)
3. How many files were changed exactly once, twice, three times, five times, and ten times? (`table`, `factor` and its `levels` argument for "counting zero occurrences")
4. How many files were changed by exactly one, two, ..., nine, or ten developers? (`tapply`, `levels`, `length`, `factor`, `table`)

   Try to achieve the same outcome using `sapply` instead of `tapply`.

b) Create an overview of the changes per file type. First, extend your data frame by the factor `filetypef` that contains only the file type. You can extract the filetype from the `file` column. (function `sub`, with regular expressions[1])

Include that logic in your `myread.cvsdata` function.

Now, write a function that creates a tabular overview. Perform the following steps:

1. Per type: How often were files of that type changed? (`sort`, `table`)
2. Per type: What is the mean number of changes for files of that type? Display this as a table with one row, to which we will add more rows in the next steps. (`sapply`, `levels`, `mean`, `table`)
3. Add three more rows: number of different files of that type, minimum number of changes for files of that type, maximum number of changes for files of that type.
4. Display the table with proper row labels. (`rownames`)
5. Sort the table columns (i.e. the file types) by the mean number of changes. (`order`, plus indexing)

c) Does the 80:20 rule apply, i.e. do the most-often changed 20% of the files account for 80% of all changes? Or (as it is unlikely to be *exactly* 80% for any project): What fraction of the changes belongs to the top-20% files?

> Hint for **c)**:
>
> Use `sort` and `table` for a frequency table; `quantile` for the top-20%-cutoff; indexing for selection; `cumsum` and `max` for cumulative changes of the top-20%-files; `nrow` for percentage

As always, include your new and amended implementations in your submission.

[1] A helpful regex might be `"^.*\\.([a-zA-Z0-9]*)$"` to deal with file names that contain a file type. But beware that some files don't have any file ending.

Example outputs for `junit20.tsv`

Task 4-1 a), Counting files.

```
# 1. Count changed files
changedfiles.count(junit20)

## [1] 2

# 2. Top-5 changed files
changedfiles.top(junit20)

##
## /cvsroot/junit/junit/README.html   /cvsroot/junit/junit/build.xml
##                               11                                 9
##                             <NA>                              <NA>
##
##                             <NA>
##

# 3. Files changed n times (n=1,2,3,5,10)
changedfiles.ntimes(junit20)

##
##  1  2  3  5 10
##  0  0  0  0  0

# 4. Files touched by n developers (n=1..10)
changedfiles.touchedbyn(junit20)

##
##  1  2  3  4  5  6  7  8  9 10
##  1  1  0  0  0  0  0  0  0  0
```

Task 4-1 b), Changes per type.

```
# 1. Changes per file type
filetypes.changes.total(junit20)

##
## html  xml
##   11    9

# Meaning: There were eleven changes pertaining to HTML files,
# and nine for XML.
```

```
# 2. Table w/ mean number of changes
filetypes.changes.mean(junit20)

## html  xml
##   11    9

# Meaning: HTML files were changed eleven times on average,
# nine times for XML.
```

```
# 3.-5. Sorted table w/ addtional values
filetype.table(junit20)

##        xml html
## mean     9   11
## number   1    1
## min      9   11
## max      9   11

# Meaning: Overall, one HTML file was changed; each touched HTML file was
# changed at least eleven times, and at most eleven times.
```

Task 4-1 c), Twenty-eighty rule

```
twenty.eighty.rule(junit20)

## [1] 55

# Meaning: 55% of the changes in junit20 pertain to the most often changed
# 20% of the files (20:80-rule not fulfilled)
```

Task 4-2: Survey: Choose a topic

The remainder of this course's tutorial will consist of a small project. You will conduct a **survey** in small groups of 4 or 5. We compiled the survey groups in the tutorial on 2017-05-08. Tasks that are to be worked on by your whole survey group are marked with the icon .

With **this practice sheet**, we are in the first step of the research process, where you

- decide on the topic of interest,
- formulate the research question and hypotheses, and
- select a target population.

In the following phases, i.e. the **next practice sheets**, you will

- think about how to operationalize the research question and the hypotheses,
- design, test, and revise the survey instrument, i.e. the *questionnaire*,
- administer the survey,
- collect, validate, and analyze the data
- answer the research question, and
- present the results.

The last page of this practice sheet contains a timeline.

Individually:

a) Read the overview on the next page to get an idea of what lies ahead.

In your survey group:

b) Discuss possible topics of interest and agree on one you would like to investigate in your study.

c) Familiarize yourself with that topic. Only then can you conduct a good survey. Find out whether there are already comparable studies to the one you're planning.

d) Most interesting topics are complex and multi-faceted. It is impossible to cover them completely in one single study and you need to focus your research interest. Write down your considerations. Also write down which aspects of the topic you decided to drop, and why.

e) Decide on a specific goal for your study and write down your research question. Form some rough ideas regarding the survey's type and execution, e.g. group of participants to target, type of questionnaire, means of distributing the questionnaire (should be web-based), ...

f) Create a wiki-page in the KVV for your survey project. During the next weeks, you will collect the results of the individual steps there.

Choose a page name using to the following scheme:

```
<ProjectName> - <Surname 1> {, <Surname n>}*
```

Make sure the page contains at least the following information:

- Complete names and e-mail addresses of all of your team members.
- Topic area of the survey (overall goal).
- Research question (clearly formulated in one to two sentences).
- Rough description of the group of participants to be addressed.

Each group will be asked to present their survey topic in the tutorial on 2017-05-15. We will discuss all topics to make sure they are all suitable for the survey research method and propose amendments if need be.

Some advice

Topic areas

Areas for potential topics have already been presented in the tutorial on 2017-05-08, including:

- Consequences of the change to the Bachelor/Master system for the studies of computer science (lecture's organization).
- Effect of having to work while studying on study duration and success.
- Preconceptions of computer science/computer scientists.
- Usage of software (OSS, e-mail, data safety, ...)
- Software engineering: theory and reality of different activities or problem areas.

You can also pick your topic from a different informatics-related area.

Project timeline

CW 19 Choose a topic (sheet 4)

You determine the concrete topic and focus of your survey (incl. research question(s)).

CW 20 Learn about surveys, design the questionnaire (sheet 5)

You design an catalogue of questions (suitable to answer the research question), develop the questionnaire, and implement it as a web form.

CW 21/22 Survey (sheets 6 & 7).

You will peer-review each other's high-level survey design and later more detailed issues regarding the questionnaire itself (layout, formulation, etc.).

CW 23 Pilot testing and recruiting (sheet 8)

You run a pilot test and give helpful feedback. In the pilot test, three to five suitable participants complete the questionnaire under observation (but without receiving help). Its strengths and weaknesses are documented. After receiving the results, you finalize your questionnaire.

You look for and choose forums in which to present your survey and ask for participation. Possible sources for participants are university lectures (via the lecturers), relevant mailing lists, and possibly others. You formulate and review a suitable recruitment letter.

CW 24–26 Conduct the survey; prepare interim report (sheets 9 & 10)

You send off your recruitment letter; the survey starts. Duration: 2 weeks. Before the end of the first week you prepare an interim report, regarding issues like: How many questionnaires have been completed? How complete? Who are the participants?

CW 26–28 Evaluate the survey's results, present the results, and compile a final report (sheets 12 & 13).

You compile the answers from all questionnaires in machine-readable form. You evaluate them: characterization of the respondent's population; global overview of the results; analysis of the correlations between answers and respondent/domain/etc. You summarize the most interesting results on slides to present them within 15 minutes.

Total effort

This course is worth 5 credit points, each of which being equivalent to 30 hours of work. The total effort (including lecture, tutorial, and exam) therefore sums up to 10 hours per week over 15 weeks.

Per week, you (as a single person) need to invest about 5 hours of work for the tutorial in addition to being present. As there is a lot to do in relatively short time:

- split the work sensibly within your group,
- work as a team, i.e. help each other, and
- start early enough (the coming practice sheets will be available early on).