Performance Evaluation of Computer Systems
Plotting, statistical tests, ...

Vojtěch Horký    Petr Tůma

Department of Distributed and Dependable Systems
Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic

May 10, 2017
Task

Individual tasks

1. Warm-up detection.
2. Central tendency selection.
3. Computation of confidence intervals.
4. Comparison of alternatives.
5. Histograms and density plots.

Implementation

Java and C++ (both), data processing in R.
Source Code & Submission

Harness and examples

http://d3s.mff.cuni.cz/teaching/peva/files/lab04.tar.gz
(Includes compilation and run scripts.)

Submission

By e-mail to horky@d3s.mff.cuni.cz.

Send TAR (or ZIP) with (commented) sources, run scripts, results and a brief README.
A Bit of Theory
Confidence Intervals

Interval such that with chosen probability (confidence level), the true mean lies in interval $C_1 \leq X_n \leq C_2$.

- Higher confidence means larger interval
- Commonly used are 95% and 99%.

If we keep making new sets of observations and constructing confidence intervals from them, 95% will be the ratio of intervals that contain the unknown mean.
Comparing Alternatives via Confidence Intervals

Check whether the CI overlap.

No overlap

We can conclude that there is a difference between systems.

Overlap, means are not within the other interval

We cannot conclude anything.

Overlap, means are within the other interval

We think the systems are the same, but we are not that sure.
Comparing Alternatives via Statistical Test

Null hypothesis \( H_0 \) – means are equal. We seek evidence to reject it.

Alternate hypothesis

Software gives us \textit{p-value}: the smallest significance level that would allow us to reject \( H_0 \).

The smaller the p-value is, the larger the evidence against \( H_0 \).

Typically, we start rejecting if p-value is less than 0.05.
The Tasks
Warm-up Detection

Manually decide what is enough warm-up for

- `std::sort` (see `cpp-sort.cpp`)
  - Use 1,000,000 elements for the sorting
- xalan benchmark from the DaCapo suite
- fop benchmark from the DaCapo suite
Central Tendency Metric

Select suitable central tendency metric for the three measurements. R histograms or density plots can help you with your decision.

R code excerpts

```r
hist(x)
plot(density(x))
```
Confidence Intervals

Compute 95% and 99% confidence intervals for the three measurements.

Compare results of two methods:
  • using Student’s t-distribution
  • using percentile bootstrap

R code excerpts

```R
sample(x, replace=TRUE)
quantile(x, probs=c(0.01, 0.99))
t.test(x, conf.level=0.99)$conf.int[1:2]
```
Comparing Alternatives

Rerun the experiments in different settings:
- `std::sort` with `-O2` and `-O3` optimizations.
- `xalan` with 40 MB smaller heap

Is there significant difference?
- Compare using confidence intervals
- Compare using statistical tests

R code excerpts
```
t.test(x, y)
```
Density Plotting

Plot density graph for std::sort with -01, -02 and -03 optimizations.

Plot everything into one graph (one color each).
Also plot reference normal distribution (use -03 parameters).

R code excerpts

```r
plot(density(x), col="blue")
rainbow(4)
legend("topleft", legend=c("-01", "-02"), col=c("blue", "red"))
rnorm(100, 10, 3)
```
Provided Examples
prepare.sh

1. Downloads DaCapo benchmark.
2. Prepares random data for std::sort benchmark.
3. Compiles std::sort benchmark.

You probably do not need to modify this script at all.
run-dacapo.sh

Provides stable configuration of JVM for running DaCapo benchmarks.

Use the run_it function, giving it the following parameters:

1. Heap size (640m)
2. New generation size (192m)
3. Benchmark name (xalan or fop)
4. Number of iterations (5 or more depending on the warm-up time)
5. Log filename (e.g. xalan.out)
6. Data filename (e.g. xalan.640-192.5.data)

The data can be loaded into R by the following command

```r
x <- read.table("xalan.640-192.5.data")$V1
```
cpp-sort.cpp

Wrapper around std::sort.

You do not need to modify this file at all.

Add file with random numbers and iteration count as parameters (e.g. ./cpp-sort-01 data.1000000 5).
\begin{verbatim}
utils.r

my.printf()
   A convenient wrapper with the expected functionality.

my.pdf.open()
   Redirects graphical output to PDF file.
   (Set LATEX_FONTS environment variable to YES to use \LaTeX fonts in the plots.)

my.pdf.close()
   Closes the last PDF, embedding fonts if needed.

my.min.max()
   Tells minimum and maximum of a vector,
   suitable for \texttt{ylim} or \texttt{xlim} parameters.

my.get.ci.from.bootstrap()
   Compute confidence intervals using quantiles of
   bootstrapped distribution.
\end{verbatim}