Reproducible Research for OMNeT++ Based on Python and Pweave

Kyeong Soo (Joseph) Kim
Department of Electrical and Electronic Engineering
Xi’an Jiaotong-Liverpool University

07 September 2017
Outline

• Reproducible Research

• Python and Pweave

• Reproducible Research for OMNeT++

• Example: OMNeT++ FIFO Simulation
Reproducible Research
Reproducible Research

• Reproducible research is a key to any scientific method and ensures repeating an experiment and the results of its analysis in any place with any person.

• A study can be truly reproducible when it satisfies at least the following three criteria:
  • All experimental methods are fully reported.
  • All data and files used for the analysis are (publicly) available.
  • The process of analyzing raw data is well reported and preserved.

• Reproducible research is to ensure
  • Same data + Same script = Same results
Why Do We Need Reproducible Research: Two Examples

• LIGO - Gravitational Wave Detection

• Schön scandal - Molecular Computing
LIGO - Gravitational Wave Detection

• The **Laser Interferometer Gravitational-Wave Observatory (LIGO)** is a large-scale physics experiment and observatory to detect cosmic gravitational waves.

• The detection of gravitational wave was reported in *Physical Review Letters* in Feb. 2016, together with [ipython notebook](https://github.com/ipython/ipython-notebook) with analysis code and data.
No records found for his ground-breaking experimental results, including lab notebook, experimental samples and data, hard disk drives.

During the investigation, he kept repeating “I clearly observed them in the Lab but ...”
Python and Pweave
R/Sweave to Python/Pweave

• Until recently, R was the language of choice for statistical processing and data analysis.
  • Still, R has the largest code base for a wide variety of statistical and graphical techniques.

• Like *ipython* (now *jupyter*), R provides a nice tool called *Sweave* (now replaced by *knitr*) to weave documentation and the results of the execution of R code chunks into one source file for integrated documentation.

• Python — one of the most popular languages in scientific computing, including artificial intelligence & machine learning — recently takes over R in statistical processing and data analysis as well.
  • Thanks to *pandas* implementing DataFrame object similar to R and *Pweave*, python can replace R for most statistical and data analysis tasks, while retaining its many advantages over R (i.e., fully-featured programming language with easy syntax and higher speed).
Snippets of R Source Code and Sweave File for LaTeX

```r
### customize
.old <- theme_set(theme_bw())
.pt_size <- 3.5

### generate summary plots for reference architecture with N=1
.rf_N1.data <- paste(.rf_N1.wd, paste(.rf_N1.base, "data", sep=".", sep="/"))
.df <- read.csv(.rf_N1.data, header=TRUE)
### .df <- .df[order(.df$N, .df$n, .df$dr, .df$br, .df$repetition), ] # order data frame
.df <- sort_df(.df, vars=c("N", "n", "dr", "br", "repetition")) # sort data frame
.rf_N1.df <- dplyr(.df, c(.n), .(dr)), function(df) {return(GetMeansAndCiWidths(df))}
.rf_N1.plots <- list()
for (.i in 1:7) {
  .df <- subset(.rf_N1.df, select = c(1, 2, (.i*2+1):((.i+1)*2)))
  names(.df)[3:4] <- c("mean", "ci.width")
  .limits <- aes(ymen = mean - ci.width, ymax = mean +ci.width)
  .p <- ggplot(data=.df, aes(group=dr, colour= factor(dr), x=n, y=mean)) + geom_line() + scale_y
  .p <- .p + xlab("Number of Users per ONU (n)") + ylab(.labels.measure.[i])
  \#
  .p <- .p + geom_point(aes(group=dr, colour= factor(dr), x=n, y=mean), size=.pt_size)
  .p <- .p + geom_point(aes(group=dr, shape= factor(dr), x=n, y=mean), size=.pt_size) + scale
  .p <- .p + geom_errorbar(.limits, width=0.1) + scale_colour_discrete("Line Rate\n[Gb/s]")
  .rf_N1.plots[[.i]] <- .p

\subsection{Hybrid PON}
\\
\\
<<echo=F,results=tex>>=
.df <- subset(.hp.df, select=c(1:8))
names(.df)[3:8] <- c("dly.mean", "dly.ci.width", "thr.mean", "thr.ci.width", "trf.mean", "trf.ci.width")
.tablendf <- xtable(.df, caption="Performance measures of FTPP traffic
digits(.tablendf)[2:9] <- c(0, 1, rep(-4, 6))
print(.tablendf,
    tabular.environment="longtable", caption.placement="top",
        include.rownames=FALSE, floating=FALSE, NA.string="NA")
@
Pweave source file ("*.Plw")
• Mix of documentation (e.g., LaTeX) and code Chunks (e.g., Python)

Weaving (pweave)

Tangling (ptangle)

Document 1
Result of the Execution of Code Chunk 1

• A program source file for separate execution and debugging
  • e.g., "*.py" for Python

Document N
Result of the Execution of Code Chunk N

A documentation source file
• e.g., "*.tex" for LaTeX
The following Python code chunk can automatically generate a long table over multiple pages from a pandas dataframe:

```python
<<echo=False, results='raw'>> =
import numpy as np
df = fifo_df.filter(regex="^(!|(r|R)un).*" )  # exclude columns starting with run/Run
print(df.to_latex(longtable=True))
```

Note that a space is inserted between `=' and `:' to prevent Pweave from weaving the code; it seems that there is no way to escape Pweave chunk code markers.
Reproducible Research for OMNeT++
How to Deal with Simulation Input Files

- Include them in the document.
  - OK for small simulations

- Use a snapshot of the whole configurations.
  - e.g., git commit hashes

Listing 1: ‘FifoNet.net’ for FIFO sample model.
How to Guarantee Match Between Input Files and Output Data

• Online generation of results
  • Include simulation execution code within a document
    • Refer to the provided sample Pweave file.
    • OK for smaller simulations, but not for larger simulations.

• Use a snapshot of the whole configurations and data
  • e.g., git commit hashes
  • Version controlling output data together with source code and input configuration files, however, may greatly increase the size of a repository.
How to Present and Analyze Output Data

• Unstacking of stacked DataFrame
  • Use `pivot` function (see the example shown here).

• Aggregated processing of measurement data over independent variables
  • Use `pivot_table` function.
  • Useful for the calculation of mean and confidence intervals over multiple iterations.

• Online calculation of confidence intervals
  • Confidence intervals (CIs) can be calculated by assigning a custom function for CI to `aggfunc` parameter of `pivot_table` function.
  • Now pandas support error bars in its own plot functions.
Example: OMNeT++ FIFO Simulation