

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.

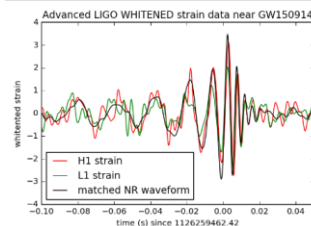
- The detection of gravitational wave was reported in *Physical Review Letters* in Feb. 2016, together with [ipython notebook](#) with analysis code and data.



```
In [9]: # We need to suppress the high frequencies with some bandpassing:
bb, ab = butter(4, [20.*2./fs, 300.*2./fs], btype='band')
strain_H1_whitenbp = filtfilt(bb, ab, strain_H1_whiten)
strain_L1_whitenbp = filtfilt(bb, ab, strain_L1_whiten)
NR_H1_whitenbp = filtfilt(bb, ab, NR_H1_whiten)

# plot the data after whitening:
# first, shift L1 by 7 ms, and invert. See the GW150914 detection paper.
strain_L1_shift = -np.roll(strain_L1_whitenbp, int(0.007*fs))

plt.figure()
plt.plot(time-tevent, strain_H1_whitenbp, 'r', label='H1 strain')
plt.plot(time-tevent, strain_L1_shift, 'g', label='L1 strain')
plt.plot(hitime+0.002, NR_H1_whitenbp, 'k', label='matched NR waveform')
plt.xlim([-0.1, 0.05])
plt.ylim([-4, 4])
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('whitened strain')
plt.legend(loc='lower left')
plt.title('Advanced LIGO WHITENED strain data near GW150914')
plt.savefig('GW150914_strain_whitened.png')
```



R/Sweave to Python/Pweave

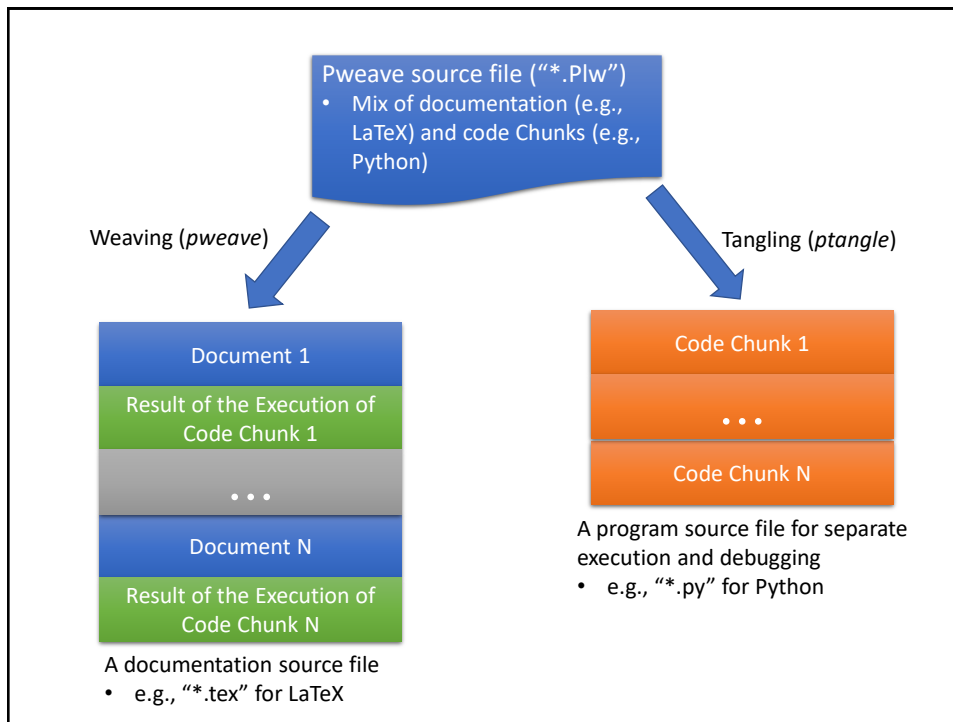
- Until recently, R was the language of choice for statistical processing and data analysis.
 - Still, R has the largest code base for a wide variety of statistical and graphical techniques.
- Like *ipython* (now *jupyter*), R provides a nice tool called *Sweave* (now replaced by *knitr*) to weave documentation 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).

```
### 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$N, df$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 <- dply(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(ymin = 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_continuous(limits=limits)
  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_shape_manual(values=c(1, 2, 3, 4, 5, 6, 7))
  p <- p + geom_errorbar(limits, width=0.1) + scale_colour_discrete("Line Rate[nGb/s]")
  rf_N1.plots[[.i]] <- p
}
```

Snippets of R Source Code
and Sweave File for LaTeX

```
\subsection{Hybrid PON}
%%
%% tables for dedicated access
%%
<<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"
)
tabledf <- xtable(df, caption="Performance measures of FTTP traffic
digits(tabledf)[2:9] <- c(0, 1, rep(-4, 6))
print(tabledf,
  tabular.environment="longtable", caption.placement="top",
  include.rownames=FALSE, floating=FALSE, NA.string="NA")
@
```



Weaving Example: Automatic Table Generation

The following Python code chunk can automatically generate a long table over multiple pages from a pandas dataframe¹:

```
<<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))
@
```

Weaving & LaTeXing

The following Python code chunk can automatically generate a long table over multiple pages from a pandas dataframe¹:

```
<<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))
@
```

	File	Module	Name	Unnamed: 19
0	Fifo1-st=0.01-#0.sca	_runattrs_	st	0.010000
1	Fifo1-st=0.01-#0.sca	FifoNet.fifo	queueingTime:mean	0.000262
2	Fifo1-st=0.01-#0.sca	FifoNet.fifo	queueingTime:max	0.031311
3	Fifo1-st=0.01-#0.sca	FifoNet.fifo	busy:timeavg	0.049941
4	Fifo1-st=0.01-#0.sca	FifoNet.fifo	qlen:timeavg	0.001308
5	Fifo1-st=0.01-#0.sca	FifoNet.fifo	qlen:max	4.000000
6	Fifo1-st=0.01-#0.sca	FifoNet.sink	lifetime:mean	0.010262

Continued on next page

¹Note that a space is inserted between 'r' and 's' 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

```
// This file is part of an OMNeT++/OMNEST simulation example.
// Copyright (C) 1992-2015 Andrea Varga
// This file is distributed WITHOUT ANY WARRANTY. See the file
// "license" for details on this and other legal matters.
//
// Simple queueing network: generator + FIFO + sink.
//
network FifoNet
{
  submodules:
  {
    gen: Source {
      parameters:
        @display("p=85,100");
    }
    fifo: Fifo {
      parameters:
        @display("p=205,100");
    }
    sink: Sink {
      parameters:
        @display("p=325,100");
    }
  }
  connections:
    gen.out --> fifo.in;
    fifo.out --> sink.in;
}
```

```
commit 857ae37cd233914fd7271584afc4be10bcf75a61
Author: Kyeong Soo (Joseph) Kim <kyeongsso.kim@gmail.com>
Date: Mon Feb 27 08:59:31 2017 +0000

    Add ini file.

commit f1e7fad0265068d906efd02026e774076c00297
Author: Kyeong Soo (Joseph) Kim <kyeongsso.kim@gmail.com>
Date: Mon Feb 27 08:56:07 2017 +0000

    Remove README.rst; only the markdown version of README

commit 8765336f9e2f5543fea8c4f37a0cf894da7f4c8e
Author: Kyeong Soo (Joseph) Kim <kyeongsso.kim@gmail.com>
Date: Sun Oct 2 17:32:02 2016 +0000

    Change simulation time.
```

Listing 1: 'FifoNet.ned' 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.

```
In [1]: df
Out[1]:
```

	date	variable	value
0	2000-01-03	A	0.469112
1	2000-01-04	A	-0.282863
2	2000-01-05	A	-1.509059
3	2000-01-03	B	-1.135632
4	2000-01-04	B	1.212112
5	2000-01-05	B	-0.173215
6	2000-01-03	C	0.119209
7	2000-01-04	C	-1.044236
8	2000-01-05	C	-0.861849
9	2000-01-03	D	-2.104569
10	2000-01-04	D	-0.494929
11	2000-01-05	D	1.071804



```
In [3]: df.pivot(index='date', columns='variable', values='value')
Out[3]:
```

variable	A	B	C	D
date				
2000-01-03	0.469112	-1.135632	0.119209	-2.104569
2000-01-04	-0.282863	1.212112	-1.044236	-0.494929
2000-01-05	-1.509059	-0.173215	-0.861849	1.071804

Demo: OMNeT++ FIFO Simulation