MLSS 2015 Probabilistic Programming Practical
Getting Started Guide

Frank Wood and Tom Jin

July 2015

1 Introduction

The probabilistic programming practical will give you hands-on experience with probabilistic programming, particularly the Anglican programming language and system `http://www.robots.ox.ac.uk/~fwood/anglican`.

By the end of the practical you should be comfortable programming in Clojure and Anglican, and, more importantly, familiar with how to program in probabilistic programming languages in general.

The exercises and short-lecture introductions to the exercises will be constructed so as to help you understand how probabilistic programming works in general. And while the amount of time we will have together will be insufficient for you to either implement a probabilistic programming system or implement a custom inference algorithm in an existing system, you should be left with a very clear impression of a) how one would go about implementing such a probabilistic programming language and b) where you would go and what it would take to start extending, for instance, Anglican with both new probabilistic primitives and new inference algorithms.

Prior to arriving at the practical, please be sure to do the following pre-practical preparation (note that while this document is long, the required preparation work should take, realistically, no more than a few minutes to complete):

2 Required Pre-Practical Preparation

Anglican is a language that compiles to Clojure that compiles to the JVM. For this reason you need the Java and Clojure ecosystems installed on either your own personal laptop or on a machine into which you can ssh, and, in the latter case, to which you can open socket (http) connections.

2.1 Java Prerequisites

Clojure depends on having a JVM installed of version ≥ 1.5 (the most recent available version is fine). Windows and Mac OS X users can download Java installers from https://www.java.com/en/download/manual.jsp. Linux users who do not already have Java installed can install from their package managers:

```
# Debian/Ubuntu
sudo apt-get install default-jre

# Fedora
sudo yum install java-1.7.0-openjdk
```

2.2 Install Leiningen

Leiningen is a self-installing automated Clojure project management system. You must install Leiningen from http://leiningen.org/. "lein" (short for Leiningen) is a self installing script as well as the primary means of invoking both Anglican and Clojure read eval print loops (REPL). Fortunately "lein" is trivial to install in *nix environments (see below). Windows users have it just as easy but should refer to the website.

```
# Download lien to ~/bin
mkdir ~/bin
cd ~/bin
wget http://git.io/XyijMQ

# Make executable
chmod a+x ~/bin/lein

# Add ~/bin to path
echo  'export PATH="$HOME/bin:$PATH"' >> ~/.bashrc

# Run lein
lein
```

2.3 Install Git

Git is a powerful version control system. Anglican itself is open source, versioned in Git, and available at https://bitbucket.org/probprog/anglican. The practical will make use of the Anglican user repository https://bitbucket.org/probprog/anglican-user which provides a bare-bones Leiningen Anglican project which will allow you to get up and running very quickly.

A short tutorial on Git is available from https://try.github.io.

2.3.1 Windows

An installer is available for download from http://git-scm.com/download/win.

2.3.2 OS X

If you have Xcode installed then git is already installed, otherwise install the Xcode command line tools which includes git.

```
xcode-select --install
```

2.3.3 Linux

Git is available from your distribution's package manager.

```
# Debian/Ubuntu
sudo apt-get install git

# Fedora
sudo yum install git
```

2.3.4 Post installation (All platforms)

Configure git with your name and email address to sign your commits with.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@domain.com"
```

2.4 Fork and Clone the anglican-user Git Repository

Here you have an option: if you already have a Bibucket account or don't mind creating one, you may optionally use a Bitbucket account to fork the anglican-user repository and, subsequently clone your fork on your local computer.

Alternatively you may simply clone, via https, the public anglican-user repo

```
git clone https://bitbucket.org/probprog/anglican-user.git
```

2.5 You're Done!

If you have managed to do all this successfully then you will should have a Leiningen project called anglican-user sitting locally on your machine. With it you should be able to start a web-based Anglican REPL: [1]

```
cd anglican-user
lein gorilla
```

which will open a browser window which points to a locally hosted url. If it doesn't you should be able to connect to the address printed in the terminal window. Using the menu on the top right you should be able to open and interactively run `template.clj`.

You should be able to start a Clojure command line REPL:

[1] Users installing on a server will instead run `lein gorilla :ip 0.0.0.0` .

```
lein repl
```

and within the REPL you should be able to execute the following commands

```
anglican.core=> (ns mine (:use [anglican core runtime emit]))
mine=> (sample (normal 1 1))
```

which should emit a sample from a $\mathcal{N}(1,1)$ distribution.

3 Optional Pre-Practical Preparation

3.1 IDE Installation

We will be developing Anglican applications in both the web-based REPL and standard software development tools. Clojure is compatible with a number of IDE's and software development toolchains including, for instance, Emacs and Eclipse. We will use LightTable to demonstration IDE-based development. You may use any tool chain you wish, but LightTable is elegantly compatible with Clojure, if spartan and initially slightly frustrating.
`http://lighttable.com/`

3.2 Clojure Programming

Ideally you would be a Clojure programmer, or, at least, a functional programming expert in advance of the practical. Familiarizing yourself with Clojure can only help the overall experience. The Clojure main website `http://clojure.org/` has links to a large number of language learning resources, in particular `http://clojure-doc.org/articles/tutorials/introduction.html`.

3.3 Probabilistic Programming Reading

There are a number of probabilistic programming papers and online resources which you may wish to read in advance of the practical. For the purposes of this practical we specifically recommend reading an introduction to probabilistic programming and the Anglican language [4] (this is an arXiv update to [3] that includes source code in an updated and, as of now, current syntax), a paper on how to implement inference in a probabilistic programming language, in this case probabilistic-C [2], and a paper about how about higher order probabilistic programming languages can be used to code expressive, particularly Bayesian nonparametric, models [1].

Also there are a number of online resources for Anglican in particular and probabilistic programming in general.

3.3.1 Online Resources

- `http://www.robots.ox.ac.uk/~fwood/anglican/`

- `http://dippl.org/`
- `http://probabilistic-programming.org/wiki/Home`

References

[1] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: A language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI2008)*, pages 220–229, 2008.

[2] Brooks Paige and Frank Wood. A compilation target for probabilistic programming languages. In *JMLR; ICML 2014*, pages 1935–1943, 2014.

[3] F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pages 1024–1032, 2014.

[4] F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. *arXiv preprint arXiv:1507.00996*, 2015.