Anglican Getting Started Guide

Frank Wood, Brooks Paige

July 2015

1 Introduction

This guide and associated set of programming exercises will get you up and programming using the Anglican programming language and system `http://www.robots.ox.ac.uk/~fwood/anglican` in a very short period of time (10 minutes to complete the setup instructions in this guide; 4 hrs. to learn the fundamentals of functional programming, Clojure, and Anglican by going through the included exercises).

If you go through all the materials you will end up comfortable programming in Clojure and Anglican, and, more importantly, have an idea about with how to program in probabilistic programming languages in general.

2 Installing Anglican

Anglican is a probabilistic programming language that compiles to Clojure which subsequently compiles to JVM bytecode. For this reason you need the Java and Clojure ecosystems installed on either your own personal computer 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. **Note that Leiningen version 2.x is required**; the version in GNU-Linux package repositories may be quite a bit out of date.

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

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

# Add ~/bin to path
# Note: Mac OS X users should replace ".bashrc" with ".profile"
echo  'export PATH="$HOME/bin:$PATH"' >> ~/.bashrc

# Run lein
lein
```

2.3 Download the exercises

The exercises themselves can be downloaded from the following url:

`https://bitbucket.org/probprog/mlss2015/get/master.zip`

(make sure your browser doesn't insert weird line-break characters in the url when you click on it; if it does manually copy and past the url)

Download and unzip this file — the resulting directory contains everything you need to learn Anglican.

When you have managed to do all this successfully then, in effect, you will have a Leiningen (Clojure) project sitting locally on your machine. Within it you should try to start a web-based, Anglican-enabled Gorilla REPL: [1]

```
# replace "mlss2015" with the unzipped directory containing
# the exercises
cd mlss2015
lein gorilla :port 8990
```

which will start a web service on port 8990 which allows you to view, edit, and run the different exercises. Open up a web browser, and visit `http://localhost:`

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

8990/worksheet.html. Using the menu on the top right (Fig. 1) you should be able to open and interactively run the exercises, using the "Load a worksheet" command.

Figure 1: Click here to load worksheets.

Start by loading beta-flip/hello-world-worksheet.clj. You can step through (and execute) the cells in the worksheet by pressing shift+enter (also accessible from the menu). This worksheet introduces you to functional programming, Clojure, and Anglican in order. It contains everything required to complete the pedagogical exercises in the following worksheets, which should be completed in the following order

1. gaussian/gaussian-worksheet.clj
2. physics/physics-worksheet.clj
3. traces/poisson-trace-worksheet.clj
4. coordination/coordination-worksheet.clj

3 Optional

3.1 Clojure Programming

Ideally you would be a Clojure programmer, or, at least, a functional programming expert in advance of learning Anglican. It is not necessary however, as these materials are self contained. Familiarizing yourself with Clojure may help your overall experience however. 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.

3.2 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.

3.2.1 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.