IGGPIPE INSTALLATION

Ted Toal <twtoal@ucdavis.edu>

Table of Contents

This describes how to install all elements needed to run the IGGPIPE software.

1. If not done already, download IGGPIPE files

1. Browse to https://github.com/BradyLab/IGGPIPE

2. At bottom of right column on screen, click "Download ZIP" and choose a place to put it on your computer.

3. Unzip the zip file on your computer.

4. Rename the unzipped folder from "IGGPIPE-master" to just "IGGPIPE".

5. If you are a GitHub user and wish to use the IGGPIPE GitHub repository directly, instead of the above steps, you can fork the IGGPIPE repository and work on your own copy. The *master* branch is where the releases reside, each tagged with a version number. You can create a branch whose contents are the same as a tagged version with the command:

   ```
   git checkout -b my_V1.0_branch v1.0"
   ```

   You can then do the installation process using those files.

   I will try to monitor for pull requests if you make useful changes.

2. Requirements for installing IGGPIPE

1. **64-bit Mac OSX or Linux System**

   This describes installation on a 64-bit Mac OSX system. For 64-bit Linux, we have tested it and tried to include installation differences here. It will not work on 32-bit Windows systems. It should work on 64-bit Windows systems but is untested.

2. **The computer must have enough memory and speed**

Before starting the installation, consider whether the computer on which the software is to be installed has enough memory for the problem at hand. I have been running IGGPIPE using 16GB of RAM, and I'd recommend no less than that amount. For larger genomes or more genomes, you will need more. My computer has a 2.4 GHz processor, which has been adequate. You should monitor memory usage while the pipeline is running, and if it becomes excessive for your computer, you know you will need more memory to run those genomes.

3. **Work from the command line**

   Most work here is done from the command line, by opening the Terminal application. You should be familiar with some basic command line commands such as *cd*, *ls*, *cp*, *rm*, *mv*, *sudo*, *less* or *more*, *head* and *tail*, *man*, and *mkdir*. If you do not know how to use the command line or don't know these basic commands, take time to learn a bit. A suggested tutorial (on the long side, but spend as little or as much time as you need to feel you can get started here) from UC Davis is at:

   http://korflab.ucdavis.edu/Unix_and_Perl/current.html

4. **Work in the IGGPIPE main directory unless otherwise instructed**

   While working from the command line to install IGGPIPE, most of the time you will be in the IGGPIPE main directory, unless instructed otherwise. If you unzipped the IGGPIPE zip file in your Documents folder, you would change into the IGGPIPE directory with this command:

   ```
   cd ~/Documents/IGGPIPE
   ```

5. **Know how to use a plain text editor and have one available**

   You must have a plain text editor you know how to use. If nothing else, the simple editor *nano*, which comes with Mac and Linux systems, will work. On a Mac you can also use the *TextEdit* program will work (use Plain Text format). For Mac, the free open-source TextWrangler program is strongly recommended, and its $50 more capable version called BBEdit is totally worth the money for anyone who regularly edits text files. TextWrangler is available from the Apple App Store (Applications, App Store) or from:

   http://www.barebones.com/products/textwrangler

6. **The following programs/applications will be installed if they aren't already:**

   a. Xcode (V6.4 works) (Apple App Store; *make* and *g++*)

   b. Jellyfish V2 or later (V2.2.3 works) (http://www.genome.umd.edu/jellyfish.html)

   c. Perl V5 or later (V5.16.0 works) (https://www.perl.org/get.html)

   d. R V3 or later (V3.2.1 works) (https://www.r-project.org)

   e. Primer3 (V2.3.6 works) (http://sourceforge.net/projects/primer3)

   f. NCBI e-PCR (V2.3.12 works) (http://www.ncbi.nlm.nih.gov/tools/epcr)

   g. MUSCLE (V3.8.31 works) (http://www.drive5.com/muscle/downloads.htm)

7. **Two of the above will require compilation and build on your computer using make:**

   a. Jellyfish

   b. e-PCR (only requires compile on Mac OSX)

8. **One C++ program provided with IGGPIPE will be compiled and built: findMers**

3. Install IGGPIPE

If you want a Quick Start with minimal reading, just look for the command lines below and execute them, with occasional skimming of the text just above/below the commands.

1. **Copy allPathParameters.template file to new file allPathParameters.ours**

   A variety of applications must be installed. Before installing them, you should prepare a text file for editing. There is a text file in the IGGPIPE main directory that contains tool path settings for running IGGPIPE: allPathParameters.template.

   The file provides settings of paths where applications have been installed, and related settings. To make your own version of this file containing your own application paths, copy the file to a new filename, replacing ".template" with ".ours":

   ```
   cp allPathParameters.template allPathParameters.ours
   ```

2. **Open the allPathParameters.ours file in a plain text editor**

   Open the new allPathParameters.ours file created above in your plain text editor for editing. If you are in a hurry, you don’t need to read anything in the file, but can simply search for "@@", which are comment lines marking items that may need to be changed. Each "@@" comment says whether it must be changed, might need to be changed, or probably will never need to be changed, etc. For example, if your text editor is nano, you might use this command line to open your editor to edit the template file:

   ```
   nano allPathParameters.ours
   ```

3. **Set WD to your IGGPIPE directory in the allPathParameters.ours file**

   Search for "#@@" in the allPathParameters.ours file and locate the one that is followed by a line setting the value of the "WD" parameter. It looks like:

   ```
   WD := $(BRADYLAB)/Genomes/kmers/IGGPIPE
   ```

   Change the assigned value to the path of your IGGPIPE directory (where you unzipped the IGGPIPE files). For example, maybe it would look like this:

   ```
   WD := /Users/johndoe/Documents/IGGPIPE
   ```

4. **Install make and g++ (Xcode in Mac OSX)**

   IGGPIPE makes use of a utility called *make*, and also, some of the applications used by IGGPIPE are distributed as source code that must be compiled and built into a runnable application on the user’s computer, which requires a C `compiler (g` utility). On Linux, these utilities are already installed so you can skip this step. On Mac OSX, the Apple Developer Toolkit named Xcode provides these utilities, and it is available free from the Apple App Store (Applications, App Store). Xcode is a good thing to have installed anyway, for anyone doing bioinformatics work. If you don’t have it installed already, run the App Store application, search for "Xcode", and double-click the *Install* button to install it, and even if you do have it installed, make sure you are updated with the latest version. I used version 6.4, although later versions should work fine. Installation takes quite a long time, during which it appears nothing is happening. When it is finished, you can verify that it was installed successfully by finding the Xcode application icon in Applications and running it. It may then display a box requesting your computer administrator password so it can install additional components. Then, close the Xcode application and go to the command line and enter the following command, which checks to see if the command line tools such as *make* and *g++* are installed, and if not, installs them:

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

To verify they are installed, you can enter this command:

```
g++
```

and you should see the error message "clang: error: no input files".

5. **Install Jellyfish and set its path**

Jellyfish is a free open-source bioinformatics application that searches FASTA sequence files for k-mers of a specified size and writes them to a file. IGGPIPE uses Jellyfish to extract unique (occurring once) k-mers from the genome sequences being used. You can find the Jellyfish at:

```
http://www.genome.umd.edu/jellyfish.html
```

I chose the "latest source and binaries" link, then downloaded the .tar.gz file. I double-clicked this file in Finder, in the Downloads folder, and it unpacked to produce a jellyfish folder. I moved this folder to a directory I made named *src* under my user root directory:

```
cd ~
pwd
mkdir src
cp Downloads/jellyfish-2.2.3 src
```

This version of IGGPIPE was tested with Jellyfish version 2.2.3. Newer versions should work as well. *Older versions will not work, because Jellyfish changed its output file names. They used to end with _0 but no longer do!*

Now the jellyfish program must be compiled and built into an application, and installed on your computer. I used these commands, which worked without error:

```
cd ~/src/jellyfish-2.2.3
./configure
make
sudo make install
```

The *sudo* command prompts for a password, and I entered my computer’s administrator password. When the above commands are finished, I verified that Jellyfish was installed and that I could run it with these commands:

```
which jellyfish
jellyfish --version
```

Finally, the allPathParameters.ours file must have the path to Jellyfish included in it. Search the files for "#@@" and assign the path to Jellyfish, which was shown when you gave the "which jellyfish" command above, to the parameter "PATH_JELLYFISH". The path will probably already be correct because Jellyfish usually gets installed in a standard location.

```
PATH_JELLYFISH := /usr/local/bin/jellyfish
```

or, if you have Jellyfish on your path, you can optionally use simply:

```
PATH_JELLYFISH := jellyfish
```

Also, set the value JELLYFISH_HASH_SIZE, which follows, to something that seems appropriate for your computer and its memory. Read the comments for each parameter to learn more about it. If you don't know how much memory your Mac computer has, choose Apple Icon, About This Mac, and look for "Memory". The value shown may work fine, but if you are working with k-mer sizes or genome sizes that produce lots more than 24 million k-mers, you may need to increase the size (and have sufficient computer memory).

```
JELLYFISH_HASH_SIZE := 80M
```

6. **Install Perl and set its path**

Perl is a programming language used by IGGPIPE. Using it requires a Perl interpreter application on your computer. The Mac OSX system comes with a Perl interpreter already installed, and this should be sufficient. This version of IGGPIPE was tested with Perl version 5.16.0, although later versions, and earlier V5 versions, will probably be fine. You can find out if you already have Perl installed, where it is located, and what its version is with this command:

```
which perl
perl --version
```

If you do not have Perl installed, look for it here:

```
https://www.perl.org/get.html
```

Perl is a good thing to have installed anyway, for anyone doing bioinformatics work. Explicit installation instructions are not given here. Follow the instructions provided in the downloaded installation package, then re-run the "which perl" command to find the path to it.

The allPathParameters.ours file must have the path to Perl included in it. Assign the path, which was shown with the "which perl" command, to the parameter "PATH_PERL". For example, maybe your path will be:

```
PATH_PERL := /usr/local/bin/perl
```

or if Perl is on your path, you can optionally use simply:

```
PATH_PERL := perl
```

7. **Install R and set its path**

R is a programming language used by IGGPIPE. Using it requires that the R programming environment be installed on your computer. This version of IGGPIPE was tested with R version 3.2.1, although later versions, and earlier V3 versions, will probably be fine. You can find out if you already have R installed, where it is located, and what its version is with this command, which invokes the command line version of the R interpreter:

```
which Rscript
Rscript --version
```

If you do not have R installed, look for it here:

```
https://www.r-project.org
```

R is a good thing to have installed anyway, for anyone doing bioinformatics work. Explicit installation instructions are not given here. Follow the instructions provided in the downloaded installation package, then re-run the "which Rscript" command to find the path to it.

The allPathParameters.ours file must have the path to RScript included in it. Assign the path, which was shown with the "which RScript" command, to the parameter "PATH_RSCRIPT". For example, maybe your path will be:

```
PATH_RSCRIPT := /usr/bin/Rscript
```

or, if Rscript is on your path (it may be set that way during installation), you can optionally use simply:

```
PATH_RSCRIPT := Rscript
```

8. **Install Primer3 and set its path**

Primer3 is a classic bioinformatics application that generates primers from sequence data. It is used by IGGPIPE to generate primers for candidate IGG markers, so it must be installed on your computer. This version of IGGPIPE was tested with Primer3 version 2.3.6, although later versions, and earlier V2 versions, will probably be fine. You probably know if you already have Primer3 installed. If you don’t know that you do, then you should install it. Look for it here:

```
http://sourceforge.net/projects/primer3
```

It comes pre-built for OSX, so make sure you download the OSX version. Put the downloaded directory wherever you want on your computer. The file named primer3_core in the root directory of the downloaded package is the executable program file. The allPathParameters.ours file must have the path to primer3_core included in it. Assign the path to the parameter "PATH_PRIMER3CORE". For example, maybe you put the downloaded folder into your Documents folder and you set the parameter as follows:

```
PATH_PRIMER3CORE := ~/Documents/primer3-2.3.6/primer3_core
```

or, if primer3_core is on your path, you can optionally use simply:

```
PATH_PRIMER3CORE := primer3_core
```

9. **Install e-PCR and set its path**

e-PCR is an "electronic PCR" application from NCBI that uses primers and sequence data to do an in-silico PCR amplification. It is used by IGGPIPE to test primers of candidate IGG markers to see if they generate unique amplicons of the expected length, so it must be installed on your computer. This version of IGGPIPE was tested with e-PCR version 2.3.12, although later versions will probably be fine. To install e-PCR, look for it here:

```
http://www.ncbi.nlm.nih.gov/tools/epcr
```

The download link uses FTP protocol. Log in as user GUEST with no password. Look for the latest .zip version (Mac), tar.gz (Linux), or binary folder (win32) and copy the file or folder to your computer and unzip it as needed. Put the unzipped directory wherever you want on your computer.

For Linux and Windows, binary distributions of e-PCR are available at that download link, so once downloaded, you should be able to run e-PCR without further ado. With OSX, however, it is now necessary to run *make* to compile and build the program. Version 2.3.12 had two problems with it that required editing of the source code in order for the *make* operation to complete successfully. Perhaps these problems will have been fixed in the version you download (or perhaps a binary version will be available for Mac also at the time you download). Test by trying to build e-PCR. Change into the directory that you unzipped and enter the following command:

```
(Mac only)
cd e-PCR-2.3.12
make LF64LDFLAGS= LF64CCFLAGS=-DNATIVE_LARGEFILES COMMON_CC_FLAGS=-w
```

If the *make* completes without error, there will be a file named "e-PCR" in the directory, and if you run it, it will display a page full of usage info:

```
e-PCR    (Run e-PCR to see if it works)
```

If you get errors from the *make* like I did, here are the changes I made that allowed the *make* to succeed:

a. Edit file mmap.cpp and remove "//" from the start of the line that reads "//#include <sstream>"

b. Edit file minilcs.hpp and insert the following two lines after the line that reads "#include <cstring>":

   ```
   #include <cstdlib>
   #include <sstream>
   ```

Now try the *make* command again, followed by running "e-PCR":

```
make LF64LDFLAGS= LF64CCFLAGS=-DNATIVE_LARGEFILES COMMON_CC_FLAGS=-w
e-PCR    (Run e-PCR)
```

The *make* should succeed and e-PCR should display its usage information, meaning you are good to go.

Now the allPathParameters.ours file must have the path to e-PCR included in it. Assign the path to the parameter "PATH_EPCR". For example, maybe you unzipped the zip file in your Documents folder and you set the parameter as follows:

```
(Mac, Linux, and Windows)
PATH_EPCR := ~/Documents/e-PCR-2.3.12/e-PCR
```

or, if e-PCR is on your path, you can optionally use simply:

```
PATH_EPCR := e-PCR
```

10. **Install MUSCLE and set its path**

MUSCLE is an open-domain multiple sequence aligner. It is used by IGGPIPE only if you choose to search markers or LCRs for InDels by using the *make InDels* command, so if you don’t do that you can skip this step, although you may as well install it. This version of IGGPIPE was tested with MUSCLE version v.8.31, although later versions will probably be fine. To install MUSCLE, look for it here:

```
http://www.drive5.com/muscle/downloads.htm
```

The executable images are already built, so choose the correct download for your system and download the file, putting it wherever you want on your computer, such as a bin folder.

Now the allPathParameters.ours file must have the path to MUSCLE included in it. Assign the path to the parameter "PATH_ALIGNER". For example, maybe you put the downloaded file in a *bin* folder in your user directory and you set the parameter as follows:

```
PATH_ALIGNER := ~/bin/muscle3.8.31_i86darwin64
```

or, if Muscle is on your path and the binary file is named *muscle*, you can optionally use simply:

```
PATH_ALIGNER := muscle
```

11. **Build findMers**

findMers is a C++ program that is part of IGGPIPE. It takes as input a file full of k-mers and a genome FASTA file, and produces as output a file of the k-mers with their genomic position included as additional data columns in the file. It can also locate all contigs in the genome FASTA file and output a file that lists the starting position and length of each contig. IGGPIPE uses both of these functions of findMers to generate a list of common unique k-mers to be analyzed for LCRs (locally conserved regions). The findMers program must be compiled and built using *make*. Its source files are located in subfolders within the code/cpp folder. Change into the code/cpp/findMers directory and enter the command *make*:

```
cd code/cpp/findMers
make
findMers
cd ../../..
```

The *make* should compile C++ files in the findMers folder and other in sister folders. It should complete without error, and there will be a file named "findMers" in the directory, and when that file is run with the *findMers* command shown above following *make*, it will display a page of usage information. The path to "findMers" is already set correctly in the allPathParameters.ours file.

12. **Test trashing and choose deletion method**

IGGPIPE uses *make* to run data through its pipeline. A command can be given to cause *make* to delete files that it has generate by running the pipeline. There are two different ways it can delete files: it can actually delete them, or it can move them to the Mac trash can where they can be found and undeleted if necessary. You must choose which of these methods you want. Since the trash can method seems more useful and flexible, it is the default method. You select the method by setting the parameter CMD_DELETE_WHEN_CLEANING to either $(CMD_DELETE) or $(CMD_TRASH). You should make sure it is set the way you want. Also, you should test the shell script that moves files to the trash, to make sure it works. To do this, use these commands:

```
cp help.txt junk.txt
$SHELL code/shell/trash.sh junk.txt
```

Now look in the trash can to see if file "junk.txt" is there. If this doesn’t work, you should set the $(CMD_DELETE) method as the delete method:

```
CMD_DELETE_WHEN_CLEANING := $(CMD_DELETE)
```

13. **Copy primer3settings.default.txt**

Primer3 uses a settings file to control many of the settings it uses to generate primers. Several sample settings files come with Primer3, in its root directory. One of these, **primer3web_v4_0_0_default_settings.txt**, was copied and modified for use with IGGPIPE. The file is named **primer3settings.default.txt**, in the main IGGPIPE directory. The following required changes were made to it: . P3_FILE_ID was set to a descriptive settings title. . PRIMER_EXPLAIN_FLAG was changed from 1 to 0. . PRIMER_PRODUCT_SIZE_RANGE was set to a simplified 36-300 (primers are designed with most intervening DNA sequence removed) . PRIMER_NUM_RETURN was changed from 5 to 1. . PRIMER_GC_CLAMP was changed from 0 to 1 (optional but recommended).

Copy primer3settings.default.txt to primer3settings.txt:

```
cp primer3settings.default.txt primer3settings.txt
```

This makes primer3settings.txt available for editing by you, should you want to change Primer3 settings for your needs, while keeping a pristine copy in the original primer3settings.default.txt file.

This file copy is all you need to do, IGGPIPE will work with this version, and this is the required version for running the test of IGGPIPE.

The RUN instructions for IGGPIPE indicate that primer3settings.txt should be edited if you want to change primer settings for your needs. However, whenever you want to run the test of IGGPIPE as shown below, you should re-do the above copy to use the pristine file for testing.

14. **Enable Access to FileMerge (optional, Mac only)**

Parameter settings files (allParameters.* and allPathParameters.*) and Primer3 settings files (primer3settings.txt) can be edited by the user. You might at some time wish to see what changes were made to a file by comparing it to another similar file. The *diff* command can be used on the command line to do this. Another program, available on Mac OSX, is *FileMerge*, a great file comparison and merging tool that comes with Xcode. It is initially hidden within Xcode, but you can put it in your dock to make it more easily accessible.

To run FileMerge, start Xcode, then on the menu choose Xcode, Open Developer Tool, FileMerge. When it opens up, find its icon on the dock and set it to stay put in the dock, then you can close Xcode and in the future get to it directly from the dock.

When you run FileMerge, it prompts for two or three or four file names. To see an example of use, enter the first two file names, "left" and "right", setting "left" to allParameters.template and "right" to allParameters.test, then click "Compare". You will see a comparison of the two files, with the differences clearly shown. If you wanted to incorporate changes from one of these files into the other, you can do this easily by using the up/down arrow keys to go through the differences one by one, and use the left/right arrow keys to select whether you want the left or right side file text in the output, and you can also click in the box on the bottom that shows the merged text and edit it; when finished you can save the merged text to a new file or overwrite one of the two compared files, using File, Save Merge. Since we don’t want to merge these files, exit FileMerge without saving anything.

15. **Run IGGPIPE using the test parameters in allParameters.test and check for success**

Everything is now ready to run the IGGPIPE pipeline. Data for testing it is provided in the testFASTA folder. This consists of two FASTA files that are truncated versions of the S. lycopersicum (tomato) and S. pennellii genomes, with only two chromosomes (1 and 2) and only about 14 Mbp for each one. The parameter file allParameters.test has parameters set for using these FASTA files and doing the test. It is more-or-less a copy of the allParameters.template file, modified for testing IGGPIPE.

To test IGGPIPE, from the command line in the IGGPIPE main directory, enter this command:

```
make PARAMS=allParameters.test ALL | tee logFiles/makeLog.test.txt
```

If all goes well, the pipeline will run quickly, and after four or five minutes, it should finish with the message **ALL files are up to date**.

The *tee* command routes the piped log output from *make* to the console and to the file logFiles/makeLog.test.txt. You can examine this file after the run to see what specifically happened at each step. Note that the output includes timestamps telling how long each step took to run.

There should be several files in the output folder "outTestHP11", including files starting with the prefixes BadKmers_, NonvalidatedMarkers_, IndelGroupsNonoverlapping_, IndelGroupsOverlapping_, LCRs_, MarkerCounts_, MarkerDensity_, MarkerErrors_, MarkersNonoverlapping_, and MarkersOverlapping_. The last two are the final output files containing the markers. The .pdf and .png files should be examined to see how they depict marker counts and densities.

To make sure the pipeline ran correctly, compare the file of output markers to the expected result, which is in file MarkersOverlapping.test.tsv in the IGGPIPE main directory:

```
diff MarkersOverlapping.test.tsv outTestHP11/MarkersOverlapping_K11k2L100D10_2000A100_
```

This command should not produce any output, indicating the two files are identical.

Note that there are other sample "allParameters" files in subdirectory *allParameters* which have been used for testing IGGPIPE on various genomes.

16. **Run make InDels to align markers and find InDels**

An R program that is NOT run as part of the pipeline when the *make … ALL* target is built, but which can be run using *make … InDels*, is able to read a file of LCRs, non-overlapping InDelGroups, or non-overlapping Markers, extract the DNA sequences from the genomes in each LCR or Marker region and align them, then locate all InDels in the aligned sequences and write their positions to a file. The program is called alignAndGetIndels.R. Run it as follows:

```
make PARAMS=allParameters.test InDels
```

Check that the output file exists with:

```
ls outTestHP11/Markers*.indels.tsv
```

This should list the file outTestHP11/MarkersNonoverlapping_K11k2L100D10_2000A100_2000d10_100N2F0X20V3000W8M3G1.indels.tsv

You can examine it with Excel or a text editor to see the InDel data it contains.

17. **Run make plotInDels to plot InDel information**

Another R program that is NOT run as part of the pipeline when the *make … ALL* target is built, but which can be run using *make … plotInDels*, reads the InDels file produced by *make … InDels* and plots information from it in a pdf file. The program is called plotIndels.R. Run it as follows:

```
make PARAMS=allParameters.test plotInDels
```

Check that the output file exists with:

```
ls outTestHP11/Markers*.indels.pdf
```

This should list the file outTestHP11/MarkersNonoverlapping_K11k2L100D10_2000A100_2000d10_100N2F0X20V3000W8M3G1.indels.pdf

You might want to open it and look at the plots.

18. **Run dotplot.R to make a dot plot**

The LCRs_ file contains a list of common unique k-mers assigned to locally conserved regions (LCRs), and it can be used to make a dotplot depicting alignment of the two genomes. The R program dotplot.R is provided to do this. It is driven by a parameter file, a sample of which has been provided, dotplot.template, that is set for using the test data just produced. Run dotplot.R as follows:

```
Rscript code/R/dotplot.R dotplot.template
```

Check that the output file exists with:

```
ls outTestHP11/LCRs_*.dotplot.png
```

This should list the file outTestHP11/LCRs_K11k2L100D10_2000.dotplot.png, an image file. You may want to examine it (e.g. in the OSX Preview app) to see the dot plot.

There are other sample parameter files in subdirectory *dotplot*, although the parameter file is fairly straightforward and you probably don't need other examples.

19. **Run annotateFile.R to make new files containing annotated marker data in different formats**

A common need is to add additional annotation information the table of markers. For example, you might be working with an introgression line population (as I was) and wish to annotate each marker with the names of the lines whose introgressions that marker lies within, along with the marker position relative to the introgression. Or, you might want to annotate each marker with the ID of the nearest gene and its distance away. You may also want to change file format, from .tsv (tab-separated) to .gff3 or .gtf for adding the markers to a browser track. All this can be done with the

R program annotateFile.R that is provided with IGGPIPE. It is driven by a parameter file, a sample of which has been provided, annotate.template, that is set for using the test data just produced along with additional annotation test data in folder code/R/test_GFFfuncsAndMergeData. Run annotateFile.R as follows:

```
Rscript code/R/annotateFile.R annotate.template
```

Check that the output file exists with:

```
ls MarkersAnnotated.*
```

This should list file MarkersAnnotated.test.tsv in the root IGGPIPE folder. You can examine this file with a text editor or Excel to see the new column.

There are other sample parameter files in subdirectory *annotate* which produce other types of files or do other types of file data manipulation.

20. **Edit primer3settings.txt (optional)**

After finishing installation, and prior to any run of IGGPIPE, you may want to edit primer3settings.txt file and make any changes that are important for your needs. For example, you might change the parameters that determine the acceptable *range of primer Tm values*. If you have several different setting values you use, you will probably want to keep a directory of different primer3settings.txt files and copy the needed one prior to each run of IGGPIPE.

The Primer3 user manual (http://primer3.sourceforge.net/primer3_manual.htm) describes all the parameters.

An explanation of the sequence data IGGPIPE gives Primer3 in order to generate primers will be helpful, particularly in understanding the setting of the parameter PRIMER_PRODUCT_SIZE_RANGE. Since IGGPIPE is making primers to be used in different genomes with different sequences and sequence lengths between the two primer sites, it cannot use the typical method of giving Primer3 the entire sequence between the two primer sites. Instead, IGGPIPE gives Primer3 the concatenation of two short sequences, one around each of the two k-mers that define and anchor the candidate IGG marker. Each sequence is equal to K plus twice EXTENSION_LEN in length. Both K (the k-mer length) and EXTENSION_LEN (the number of bases to add on each side of the k-mer) are defined in allParameters.template. Thus, the sequence that Primer3 uses for designing the primers is equal to 2K + 4*EXTENSION_LEN in length. IGGPIPE also gives Primer3 a value for its parameter SEQUENCE_PRIMER_PAIR_OK_REGION_LIST. This tells Primer3 to design one primer in the left half of the sequence and one primer in the right half. Thus, the primer product size will appear to Primer3 to be much smaller than the actual amplicon size will be, which is why PRIMER_PRODUCT_SIZE_RANGE can be set to a smaller value than the amplicon sizes.

Although Primer3 is a stable program and unlikely to change a lot, if new versions of Primer3 add parameters, you might want to incorporate them into primer3settings.txt. You will see new parameters if you compare primer3settings.txt to Primer3’s file primer3web_v4_0_0_default_settings.txt (for example by using *diff* or *FileMerge*).

That completes the installation of IGGPIPE.

4. To run IGGPIPE to generate markers

- Find file RUN.pdf or RUN.html in the IGGPIPE folder on your computer and open either one and follow the instructions.

4.1. For problems and help:

- Post an issue on GitHub under BradyLab/IGGPIPE repository
- Contact me, Ted Toal, twtoal@ucdavis.edu [mailto:twtoal@ucdavis.edu]