iWGS – version 1.1

in silico Whole Genome Sequencer and Analyzer

Xiaofan Zhou

2016/08/03

Contents

Introduction 1
Pre-requisites/Supported tools 3
Reads simulator 3
QC tools 3
Assembler 4
Evaluation 6
Additional tools 6
Installation 7
iWGS 7
Pre-compiled tools 7
Installation notes for other tools 8
Running iWGS 11
Command line options 11
Control file 12
Advanced configuration files 15
Output 17
Examples 17
References 20
Appendices 22
Appendix A. One complete control file 22
Appendix B. Exemplar configuration files 27

Introduction

Whole genome sequences are rich sources of information about organisms that are superbly useful for addressing a wide variety of evolutionary questions, such as measuring mutation rates [1], characterizing the genomic basis of adaptation [2], and building the tree of life [3, 4]. The rapid advance of DNA sequencing technologies has dramatically reduced the labor and cost required for genome sequencing. Particularly, it enables single investigators to perform *de novo* genome sequencing in virtually any organism they are interested in [5]. Such sequencing efforts may target various organisms with a large diversity of genome architectures. Therefore, to achieve optimal results, the choice of sequencing strategy [i.e. the combination of sequencing technology (e.g. Illumina, Pacific Biosciences), sequencing assay (e.g. paired-end, mate-pair), and other variables, such as sequencing depth] and assembly protocols (e.g. assemblers and the associated parameters) should ideally be tailored to the characteristics of a given genome, such as size and GC/repeat content [6].

The great number of possible ways to combine sequencing technologies, assays, and assembly algorithms poses a great challenge for the experimental design and data analysis in *de novo* genome sequencing projects, which in turn can sometimes lead to poor quality or downright incorrect assemblies [7]. As a consequence, several pipelines have been developed to automate specific steps in the process; for example, the recently developed iMetAMOS [8] and RAMPART [9] have been specifically designed to automate genome assembly. However, as *de novo* genome sequencing is increasingly adopted by single investigator laboratories, there is an urgent need for streamlined approaches that enable investigators to not only efficiently generate high-quality draft genome assemblies but also to predict (via simulation) and identify the most suitable design(s) (i.e. the most suitable combination(s) of sequencing strategy and assembly protocol) currently available for a specific genome.

We have developed an automated pipeline for the design and execution of *de novo* genome sequencing projects that we name iWGS (*in silico* Whole Genome

Sequencer and Analyzer). To approximate the performance of different sequencing strategies and assembly protocols, iWGS simulates high-throughput genome sequencing on user-provided reference genomes (e.g. genomes that closely represent the characteristics of the real targets), facilitating the identification of optimal experimental designs. iWGS allows users to experiment with various combinations of sequencing technologies, assays, assembly tools, and relevant parameters in a single run. iWGS is also designed to work with real data and can be used as a convenient tool for automated selection of the best assembly or genome assembler.

Pre-requisites/Supported tools

This section contains information on all tools supported by iWGS, including the latest supported version and the homepage.

Reads simulator

ART (v03.19.15) [10]:
http://www.niehs.nih.gov/research/resources/software/biostatistics/art/

pIRS (v2.0.1) [11]: https://github.com/galaxy001/pirs

PBSIM (v1.0.3) [12]: https://code.google.com/p/pbsim/

* The PBSIM package distributed with iWGS has been modified (based on version 1.0.3) to model the distribution of per read quality scores using the Weibull distribution instead of the default normal distribution. It also carries a new quality model profile learned from more recent datasets generated by RSII P5/C3 sequencing chemistry (https://github.com/PacificBiosciences/DevNet/wiki/Arabidopsis-lyrata).

Table 1: iWGS will choose the simulator automatically depending on the target read type and read length.

	SE	PE	MP	HMP	PacBio CLR
ART					
pIRS		≤ 100bp	≤ 100bp	≤ 100bp	
PBSIM					

Color key:

	Supported
	Unsupported

QC tools

Trimmomatic (v0.36) [13]:
http://www.usadellab.org/cms/index.php?page=trimmomatic

NextClip (v1.3.1) [14]: https://github.com/richardmleggett/nextclip/

Lighter (v1.1.1) [15]: https://github.com/mourisl/Lighter/

Quake (v0.3.5) [16]: http://www.cbcb.umd.edu/software/quake/

Table 2: The compatibility between QC tools and read types.

		SE	PE	MP	PacBio CLR
Trimmomatic	Quality trimming				
	Adapter trimming				
NextClip	Adapter trimming				
Lighter/Quake	Error correction			(requires NextClip trimming)	

Color key:

	Supported
	Unsupported

Assembler

ABYSS (v1.9.0) [17]: http://www.bcgsc.ca/platform/bioinfo/software/abyss/

ALLPATHS-LG (r52488) [18, 19]: http://www.broadinstitute.org/software/allpaths-lg/blog/

Celera Assembler (v8.3rc2) [20, 21]: http://sourceforge.net/projects/wgs-assembler/

Canu (v1.3) [22]: https://github.com/marbl/canu

DBG2OLC [23]: https://github.com/yechengxi/DBG2OLC

DISCOVAR *de novo* (r52488) [24]: http://www.broadinstitute.org/software/discovar/blog/

FALCON : https://github.com/PacificBiosciences/falcon

MaSuRCA (v3.1.3) [25]: http://www.genome.umd.edu/masurca.html

Meraculous (v2.0.5) [26]: http://jgi.doe.gov/data-and-tools/meraculous

Metassembler (v1.5) [27]: https://sourceforge.net/projects/metassembler

Minia (v2.0.3) [28]: http://minia.genouest.org/

Platanus (v1.2.4) [29]: http://platanus.bio.titech.ac.jp/

SGA (v0.10.14) [30]: https://github.com/jts/sga

SOAPdenovo2 (v2.0.4) [31]: http://sourceforge.net/projects/soapdenovo2/files/SOAPdenovo2/

(dip)SPAdes (v3.9) [32]: http://bioinf.spbau.ru/spades/

Velvet (v1.2.10) [33]: http://www.ebi.ac.uk/~zerbino/velvet/

Table 2: The compatibility between assemblers and read types.

	SE	PE	MP	HQMP[a]	PacBio CLR
ABYSS					
ALLPATHS-LG		overlapping			
Celera Assembler[b]					
Canu					
DBG2OLC					
DISCOVAR *de novo*		(near) overlapping			
FALCON					
MaSuRCA					
Meraculous					
Metassembler					
Minia					
Platanus					
SGA					
SOAPdenovo2					

(dip)SPAdes					
Velvet					

[a]High quality MP data (obtained from MP data after NextClip trimming)
[b]It is recommend to have at least 50x coverage CLR reads for a PacBio only assembly, or at least 20x coverage CLR reads for a hybrid assembly.

Color key:

	Require all of these
	Require at least one of these
	Supported
	Unsupported

Evaluation

QUAST (v3.2) [34]: http://bioinf.spbau.ru/quast/

REAPR (v1.0.18) [35]: https://www.sanger.ac.uk/resources/software/reapr/

Additional tools

AMOS (v3.1.0) [36]: http://sourceforge.net/projects/amos/

* the "bank-transact" included in the AMOS package is required for Celera Assembler to perform hybrid assembly using both high-quality short reads (e.g. Illumina) and PacBio CLR reads.

BWA (v0.7.12) [37]: http://sourceforge.net/projects/bio-bwa/
SAMtools (v 1.3) [38]: http://www.htslib.org/

* BWA and SAMtools are both required for SGA to perform scaffolding.

FASTX-Toolkit (v0.0.13): http://hannonlab.cshl.edu/fastx_toolkit

* the "fastx_reverse_complement" in FASTX-Toolkit is required for Velvet to analyze MP data.

KmerGenie (v1.6982) [39]: http://kmergenie.bx.psu.edu/

* KmerGenie can be invoked to estimate the "best" k-mer length for assembly protocols that involve one of these assemblers: ABYSS, SOAPdenovo2, (dip)SPAdes, and Velvet.

Installation

This section contains information on how to install the iWGS pipeline and its pre-requisites.

iWGS

The main iWGS pipeline does not require any installation, and can be run on Linux or Mac OS. To obtain and run iWGS:

1) git clone https://github.com/zhouxiaofan1983/iWGS.git
2) cd iWGS
3) ./iWGS

Pre-compiled tools

Pre-compiled binaries for Linux and/or Mac OS are available for some of the supported tools. We thus provide with iWGS a script to automatically obtain these pre-compile tools so there is no need to install these tools separately:

Reads simulator:

ART: Linux, Mac OS
pIRS: Linux
* The pre-complied binary has a bug, which hopefully will get fixed in future. For now, you will have to compile from source code.
PBSIM: Linux
* The pre-compiled binary is only available for the original version. The source code of the modified version is provided separately.

QC tools:

Trimmomatic: Linux, Mac OS

Assembler:

Celera Assembler: Linux, Mac OS
Canu: Linux, Mac OS
DBG2OLC: Linux
MaSuRCA: Linux
Metassembler: Linux
Platanus: Linux
SOAPdenovo2: Linux, Mac OS
(dip)SPAdes: Linux, Mac OS

Evaluation:

QUAST:	Linux, Mac OS

Additional tools:

BWA	Linux
SAMtools	Linux
FASTX-Toolkit	Linux

Installation notes for other tools

For other tools, please follow their respective instructions for proper installation. Some reminders/tips are provided here to facilitate a smooth installation (and execution).

Potential dependencies:

BamTools: https://github.com/pezmaster31/bamtools (by SGA)

Boost: http://www.boost.org/ (by ABYSS, ALLPATHS-LG)

GCC (version ≥ 4.7.0): (by ALLPATHS-LG, DISCOVAR)

jemalloc: http://www.canonware.com/jemalloc/ (by DISCOVAR, SGA)

Jellyfish: http://www.cbcb.umd.edu/software/jellyfish/ (by Quake)

sparsehash: https://code.google.com/p/sparsehash/ (by ABYSS, SGA)

zlib: http://www.zlib.net/ (by SGA)

Reads simulator:

pIRS:

1) modify the <pirs-dir>/src/pirs/Makfile:
    a. on the first line, replace "/usr/local/share/pirs" with <pirs-dir>/src/Profiles;
    b. requires zlib: if you do not have zlib already, install zlib first, and add the following to the sixth line ("CFLAGS += -static"):
        -L<zlib-lib-dir> –I<zlib-include-dir>.
2) modify the <pirs-dir>/src/stator/gcContCvgBias/Makefile:
    a. requires zlib: if you do not have zlib already, install zlib first, and add the following to the fourth line ("CFLAGS += -static"):
        -L<zlib-lib-dir> –I<zlib-include-dir>.

QC tools:

Quake:

1) requires Boost: if you do not have Boost already, install Boost, and update the second line ("CFLAGS") in <quake-dir>/src/Makefile;
2) requires Jellyfish (use version 1.* only; Quake is not compatible with v2.0 or above): install Jellyfish and place a link of jellyfish binary to the <quake-dir>/bin folder;
3) install VGAM library in R.

Assembler:

ABYSS:

1) requires Boost: if you do not have Boost already, install Boost, and add the "--with-boost=<boost-include-dir>" option while running the "configure" script;
2) requires sparsehash: install sparsehash, and add the "CPPFLAGS=-I<sparsehash-include-dir>" option while running the "configure" script;
3) to enable the use of multi-threads, add the "--with-mpi=<openmpi-dir>" option while running the "configure" script; the exact <openmpi-dir> may vary depending on your environment: e.g. /usr/local/openmpi/latest/x86_64/gcc46 (ACCRE); /usr/lib/openmpi (my PC);
4) by default, ABYSS accepts k-mer sizes up to 64; to allow larger k-mer sizes, add the "--enable-maxk=<value>" (<value> should be a multiple of 32) option while running the "configure" script.

ALLPATHS-LG:

1) requires GCC version ≥ 4.7.0 to compile and **run** (make sure you have the right GCC version when ALLPATHS-LG is executed!);
2) if you do not have access to GCC version ≥ 4.7.0, please use the ALLPATHS-LG version 44837 or earlier;
3) keep in mind that the memory consumption of ALLPATHS-LG will be extremely high when using PacBio data.

DBG2OLC:

requires BLASR and PBDAGCON.

DISCOVAR *de novo*:

1) requires GCC version ≥ 4.7.0 to compile and **run** (make sure you have the right GCC version when DISCOVAR *de novo* is executed!);
2) unfortunately, there is no version of DISCOVAR *de novo* that is compatible with GCC version < 4.7.0;
3) requires jemalloc; if you do not have jemalloc already, install jemalloc (version ≥ 3.6.0), and add the "--with-jemalloc=<jemalloc-lib-dir>" option while running the "configure" script.

Meraculous:

requires Boost version ≥ 1.50.0.

Metassembler:

requires Bowtie2 and MUMmer 3.

SGA:

1) requires sparsehash: install sparsehash, and add the "--with-sparsehash=<sparsehash -dir>" option while running the "configure" script;
2) requires BamTools: install BamTools, and add the "--with-bamtools=<bamtools -dir>" option while running the "configure" script;
3) requires zlib;
4) suggests jemalloc: if you do not have jemalloc already, install jemalloc (version ≥ 3.6.0), and add the "--with-jemalloc=<jemalloc-lib-dir>" option while running the "configure" script.

Velvet:

1) by default, Velvet accepts up to two datasets ("categories"); to allow more than two datasets, add the "CATEGORIES=<value>" option while running the "make" command;
2) by default, Velvet accepts k-mer sizes up to 31; to allow larger k-mer sizes, add the "MAXKMERLENGTH=<value>" option while running the "make" command.
3) to allow the assembly of datasets with more than 2.2 billion reads, add the option "BIGASSEMBLY=1" option while running the "make" command;
4) to enable the use of multi-threads, add the "OPENMP=1" option while running the "make" command.

Evaluation tools:

QUAST:

1) Run "quast.py -test" before the first use.

Additional tools:

AMOS:

1) an error encountered when trying to compile with GCC (version ≥ 4.9.0); earlier GCC versions (e.g. 4.6) seem work fine.

KmerGenie:

1) by default, KmerGenie evaluates k-mer sizes up to 121; to allow the evaluation of larger k-mer sizes, do the following:
    a. make clean
    b. make k=<value>

Running iWGS

iWGS uses both command line options and a control file to collect all information it needs to carry out an analysis.

Command line options

Basic options:

-s | --settings <string>	the control file that contains all information for a iWGS run (see the "control file" section for detailed explanation)
-g | --genome <string>	the reference genome sequence in FASTA format
-t | --threads <int>	the number of threads to use (default: 1)
-m | --memory<int>	the number of GBs of memory to use (default: 8)
-o | --out_dir <string>	the directory to store all outputs, will be created if not already exists (default: ./)
-c | --cleanup	whether to remove simulated reads and files generated by assemblers after the iWGS run
-r | --overwrite	whether to redo the simulation/QC/assembly if the corresponding files already exist
-v | --verify	verify the settings and quit
-h | --help	print this help message

Advanced options:

--Mode	iWGS can run in three modes: (default: 1) 1: get parameters from command line / control file, and finish all steps (reads simulation, assembly, and evaluation) in a single run 2: get parameters from command line / control file, and generate three configuration files for library, assembly protocol, and miscellaneous options, respectively, which would allow each library / assembly protocol to have different parameters (see the "advanced configuration files" section for detailed explanation) 3: restart from the configuration files generated in Mode 2 4: quit after reads simulation
--Conf	the configuration files for library, assembly protocol, and miscellaneous options, separated by comma; to be used in Mode 3
--Real	run iWGS with real datasets

Control file

The control file consists of the following sections (see Appendix for the complete control file):

General options:

These general options in the control file are the same as corresponding command line options, and will override the command line options if both are specified.

genome =
out_dir =
threads =
memory =

Library options:

Each library option line starts with the key word "library" and contains the information for a library to be simulated, in the format of four to six parameters separated by comma. The number and meaning of the parameters needed for a library vary with different read types:

	1st parameter	2nd parameter	3rd parameter	4th parameter	5th parameter	6th parameter
SE	Library name	Read type	Coverage	Read length		
PE					Average insert size	SD of insert size
MP						
CLR				Average accuracy	SD of accurary	

Note: there is no need to specify the reads simulator; iWGS will make the choice automatically based on the read type and other specifics of the library.

Example 1: a 100bp SE library "L001" with 50x coverage.
library = L001,SE,50,100

Example 2: a 100bp PE library "L002" with 50x coverage, average insert size of 180bp, and SD of insert size of 9bp.
library = L002,PE,50,100,180,9

Example 3: a 100bp MP library "L003" with 50x coverage, average insert size of 2000bp, and SD of insert size of 100bp.
library = L003,MP,50,100,2000,100

Example 4: a PacBio CLR library "L005" with 50x coverage, average read accuracy of 0.85, and SD of read accuracy of 0.02.

library = L004,CLR,50,0.85,0.02

Simulator options:

Each simulator option line starts with a keyword in the format of: "[name_of_simulator].[option_of_simulator]". These options are used to set parameters for read simulators. Leave the options blank to use the default values.

Note: these parameters are applied universally to all libraries simulated by the corresponding simulator.

Example 1: set the substitution error rate for pIRS
pIRS.error_rate =

Example 2: set the amount to shift every first-read quality score for ART
ART.qual_shift1 =

Example 3: set the maximum read accuracy for PBSIM
PBSIM.accuracy_max =

Quality control options:

The option QC takes a list of library names (separated by comma) or the word "all" to perform preprocessing of selected or all of the libraries. Leave the option blank to skip the QC step for all libraries.

QC =

Each QC tool option line starts with a keyword in the format of: "[name_of_tool].[option_of_tool]". These options are used to set parameters for QC tools. Leave the options blank to use the default values.

Note: these parameters are applied universally to all libraries that will be processed by the corresponding QC tools.

Example 1: set the quality trimming threshold for Trimmomatic
Trimmomatic.trailing =

Example 2: set the adapter sequence for NextClip
NextClip.adapter =

Example 3: set the tool and k-mer size for error correction
Correction.tool =
Correction.kmer =

Assembly protocol options:

Each assembly protocol option line starts with the key word "protocol" and contains the information for an assembly protocol, in the format of three or more parameters

separated by comma. The number and meaning of the parameters needed for an assembly protocol are shown below:

1st parameter	2nd parameter	3rd .. nth parameters
Protocol name	Assembler name	Names of libraries to be included

Example 1: an assembly protocol "P002" using the assembler ALLPAHTS-LG, and libraries L002 (overlapping PE), L003 (MP), and L004 (PacBio CLR).
Protocol = P002,ALLPATHS,L002,L003,L004

Example 2: an assembly protocol "P010" using the assembler SOAPdenovo2, and libraries L001 (SE), L002 (PE), and L003 (MP).
protocol = P010,SOAPdenovo2,L001,L002,L003

Assembler options:

Each assembler option line starts with a keyword in the format of: "[name_of_assembler].[option_of_assembler]". These options are used to set parameters for *de novo* genome assemblers. Leave the options blank to use the default values.

Note: these parameters are applied universally to all assembly protocols using the genome assembler.

Example 1: set the k-mer size to be used for SOAPdenovo2; the default value ("0") indicates that the k-mer size is to be estimated.
SOAPdenovo2.kmer = 0

Example 2: set additional options for SPAdes; the text between quotation marks will be passed to the assembler.
SPAdes.option = "--only-assembler"

Evaluation options:

Each evalution option line starts with a keyword in the format of: "[name_of_evaluator].[option_of_evaluator]". These options are used to set parameters for genome assembly assessment tools. Leave the options blank to use the default values.

Example 1: whether the genome to evaluate is eukaryotic
QUAST.eukaryote = 1

Example 2: whether to run QUAST in "GAGE" mode
QUAST.gage = 1

Example 3: to provide a gene annotation file for QUAST (only effective in full evaluation mode)

QUAST.gene =

Example 4: to provide one PE library and one MP library for REAPR evaluation
REAPR.libs = L002,L003

Executable options:

Each executable option line starts with a keyword in the format of: "bin.[name_of_executable]". These options are used to set path information for needed executables. Leave the options blank if the executables are available in PATH environmental variable.

Example 1: set the path to the pIRS executable, "pirs"
bin.pIRS =

Example 2: set the path to SPAdes executable, "spades.py"
bin.SPAdes =

Example 3: set the path to QUAST executable, "quast.py"
bin.QUAST =

Advanced configuration files

One limitation of the control file is that the simulator/assembler related parameters will be applied universally to all relevant libraries / assembly protocols. iWGS allows advanced users to set parameters specific to each library / assembly protocol in the following way:

1) run iWGS in Mode 2 (advanced option); iWGS will collect information from command line options and/or the control file, and generate three configuration files for library, assembly protocol, and miscellaneous options, respectively;
2) modify the configuration files to set library /assembly protocol specific parameters;
3) run iWGS in Mode 3 to resume the analysis from the modified configuration files.

The three configuration files are explained as below (see Appendix for exemplar configuration files):

Library configuration file: (default name: libraries.conf)

This file contains one section for each library, and all keywords are in the format of: "[library_name].[option_name]".

All libraries have three common options: (using L001 as example)

L001.read_type = SE
L001.simulator = ART
L001.depth = 50

Note: the "read_type" and "simulator" options are for users' reference only, please do not modify.

Other available options vary depend on the read type, simulator, and QC settings ; for example:

L001.qual_shift1 =
L001.qual_profile1 =
L001.ins_rate1 =
L001.del_rate1=
L001.QC = 1
L001.tm-on = 1
L001.tm-trailing = 3
L001.tm-minlen = 25
L001.ec-on = 1
L001.ec-tool = Quake
L001.ec-kmer = 15

Since L001 is a SE library, only the ART parameters for the first-read are available. Trimmomatic quliaty-trimming and Quake error-correction will also be performed on L001.

Assembly protocol configuration file: (default name: protocols.conf)

This file contains one section for each assembly protocol, and all keywords are in the format of: "[protocol_name].[option_name]".

All protocols have two common options: (using P010 as example)

P008.library = L001,L002,L003
P008.assembler = SOAPdenovo2

Note: the "assembler" options are for users' reference only, please do not modify.

Other available options vary depend on the assembler; for example:

P001.kmer = 0
P001.option = "-F -R -E -w -u"

The default value for "option" (the extra command to pass to the assembler: SOAPdenovo2) is taken from the recipes of GAGE-B project [40].

Miscellaneous option configuration file: (default name: misc.conf)

This file contains all the general options, evaluation options, and executable options, which are in the same format as in the control file.

Output

iWGS will create five directories under the "out_dir" directory:

1) $out_dir/libraries/ - this directory contains all simulated reads; each simulated library will be stored in a separate subdirectory with the same name as the library; this folder will be removed if the "cleanup" command line option is turned on;
2) $out_dir/preprocessed/ - this directory contains all reads after preprocessing; each library will be stored in a separate subdirectory with the same name as the library; this folder will be removed if the "cleanup" command line option is turned on;
3) $out_dir/protocols/ - this directory contains the results of all assembly protocols; files generated by each assembly protocol will be stored in a separate subdirectory with the same name as the protocol; this folder will be removed if the "cleanup" command line option is turned on;
4) $out_dir/logs/ - this directory contains log files for all library simulation, assembly, and evaluation steps;
5) $out_dir/assemblies/ - this directory contains copies of all successfully assembled contig/scaffold files;
6) $out_dir/evaluation/ - this directory contains the results of QUAST evaluations; copies of the main evaluation reports can be found in this directory; two subdirectores, "contigs/" and "scaffolds/", will be created to store the detailed evaluation results on assembled contigs and scaffolds, respectively.

Examples

A quick and simple iWGS analysis:

A quick and simple iWGS analysis can be carried out without using a control file, in which case the only required option is the reference genome ("-g"). iWGS will simulate the following two libraries:

Name	Read type	Simulator	Read length	Coverage	Average insert size	SD of insert size
L001	PE	pIRS	100bp	50x	180bp	9bp

L002	MP	pIRS	100bp	50x	3000bp	150bp

and evaluate the following two assembly protocols:

Name	Assembler	Libraries	Note
P001	SOAPdenovo2	L001, L002	k-mer size to be estimated
P002	ALLPATHS-LG	L001, L002	

iWGS will behave in the same way in cases where a control file is provided but no library and assembly protocol is specified.

Exemplar command:
iWGS -g Kazachestania_africana.fa -t 8 -m 32 -o K_africana

Run with library / assembly protocol specific parameters:

1) collect information from the control file and generate configuration files:
   iWGS -s K_africana.ctl --Mode 2
2) modify the configuration files;
3) resume the analysis:
   iWGS --Mode 3 --Conf libraries.conf,protocols.conf,misc.conf

Using iWGS to analyze real datasets:

iWGS can also be used to analyze real datasets, providing a convenient way for users to try multiple assembler/parameter combinations on their own data and evaluate the resulting assemblies.

1) create the output directory ($out_dir) and the "real_data" subdirectory ($out_dir/real_data);
2) place all real dataset under the "real_data" directory; the names of the data files should follow the naming convention of iWGS:

Read type	Data file name
SE, PacBio CLR	[library_name].fq
PE, MP	[library_name]_1.fq, [library_name]_2.fq

[a]the [library_name] can be any legitimate name for the library.

3) in the iWGS control file, create a library option line for each real dataset; real specs (e.g. coverage, read length) of the libraries should be provided;

4) in the iWGS control file, either set the "genome_size" option or provide a reference genome (only used for the purpose of genome size estimation)
5) run iWGS with the "Real" option:
   # Exemplar command:
   iWGS -s real_data.ctl -r 0 -c 0 --Real

Note: similarly, with proper library options in the control file, iWGS can analyze real datasets together with simulated data. The users can use this feature to evaluate the potential improvements additional datasets may have on existing real data.

References

1. Kumar, S. and S. Subramanian, *Mutation rates in mammalian genomes.* Proc Natl Acad Sci U S A, 2002. **99**(2): p. 803-8.
2. Roux, J., et al., *Patterns of positive selection in seven ant genomes.* Mol Biol Evol, 2014. **31**(7): p. 1661-85.
3. Rokas, A., et al., *Genome-scale approaches to resolving incongruence in molecular phylogenies.* Nature, 2003. **425**(6960): p. 798-804.
4. Salichos, L. and A. Rokas, *Inferring ancient divergences requires genes with strong phylogenetic signals.* Nature, 2013. **497**(7449): p. 327-31.
5. Rokas, A. and P. Abbot, *Harnessing genomics for evolutionary insights.* Trends Ecol Evol, 2009. **24**(4): p. 192-200.
6. Nagarajan, N. and M. Pop, *Sequence assembly demystified.* Nat Rev Genet, 2013. **14**(3): p. 157-67.
7. Denton, J.F., et al., *Extensive error in the number of genes inferred from draft genome assemblies.* PLoS Comput Biol, 2014. **10**(12): p. e1003998.
8. Koren, S., et al., *Automated ensemble assembly and validation of microbial genomes.* BMC Bioinformatics, 2014. **15**: p. 126.
9. Mapleson, D., N. Drou, and D. Swarbreck, *RAMPART: a workflow management system for de novo genome assembly.* Bioinformatics, 2015. **31**(11): p. 1824-6.
10. Huang, W., et al., *ART: a next-generation sequencing read simulator.* Bioinformatics, 2012. **28**(4): p. 593-4.
11. Hu, X., et al., *pIRS: Profile-based Illumina pair-end reads simulator.* Bioinformatics, 2012. **28**(11): p. 1533-5.
12. Ono, Y., K. Asai, and M. Hamada, *PBSIM: PacBio reads simulator--toward accurate genome assembly.* Bioinformatics, 2013. **29**(1): p. 119-21.
13. Bolger, A.M., M. Lohse, and B. Usadel, *Trimmomatic: a flexible trimmer for Illumina sequence data.* Bioinformatics, 2014. **30**(15): p. 2114-20.
14. Leggett, R.M., et al., *NextClip: an analysis and read preparation tool for Nextera Long Mate Pair libraries.* Bioinformatics, 2014. **30**(4): p. 566-8.
15. Song, L., L. Florea, and B. Langmead, *Lighter: fast and memory-efficient sequencing error correction without counting.* Genome Biol, 2014. **15**(11): p. 509.
16. Kelley, D.R., M.C. Schatz, and S.L. Salzberg, *Quake: quality-aware detection and correction of sequencing errors.* Genome Biol, 2010. **11**(11): p. R116.
17. Simpson, J.T., et al., *ABySS: a parallel assembler for short read sequence data.* Genome Res, 2009. **19**(6): p. 1117-23.
18. Gnerre, S., et al., *High-quality draft assemblies of mammalian genomes from massively parallel sequence data.* Proc Natl Acad Sci U S A, 2011. **108**(4): p. 1513-8.
19. Ribeiro, F.J., et al., *Finished bacterial genomes from shotgun sequence data.* Genome Res, 2012. **22**(11): p. 2270-7.

20. Myers, E.W., et al., *A whole-genome assembly of Drosophila.* Science, 2000. **287**(5461): p. 2196-204.
21. Koren, S., et al., *Reducing assembly complexity of microbial genomes with single-molecule sequencing.* Genome Biol, 2013. **14**(9): p. R101.
22. Berlin, K., et al., *Assembling large genomes with single-molecule sequencing and locality-sensitive hashing.* Nat Biotechnol, 2015. **33**(6): p. 623-30.
23. Ye, C., et al. *DBG2OLC: Efficient Assembly of Large Genomes Using Long Erroneous Reads of the Third Generation Sequencing Technologies.* ArXiv e-prints, 2014. **1410**.
24. Weisenfeld, N.I., et al., *Comprehensive variation discovery in single human genomes.* Nat Genet, 2014. **46**(12): p. 1350-5.
25. Zimin, A.V., et al., *The MaSuRCA genome assembler.* Bioinformatics, 2013. **29**(21): p. 2669-77.
26. Chapman, J.A., et al., *Meraculous: de novo genome assembly with short paired-end reads.* PLoS One, 2011. **6**(8): p. e23501.
27. Wences, A.H. and M.C. Schatz, *Metassembler: merging and optimizing de novo genome assemblies.* Genome Biol, 2015. **16**(1): p. 207.
28. Salikhov, K., G. Sacomoto, and G. Kucherov, *Using cascading Bloom filters to improve the memory usage for de Brujin graphs.* Algorithms for Molecular Biology, 2014. **9**.
29. Kajitani, R., et al., *Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads.* Genome Res, 2014. **24**(8): p. 1384-95.
30. Simpson, J.T. and R. Durbin, *Efficient de novo assembly of large genomes using compressed data structures.* Genome Res, 2012. **22**(3): p. 549-56.
31. Luo, R., et al., *SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler.* Gigascience, 2012. **1**(1): p. 18.
32. Bankevich, A., et al., *SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing.* J Comput Biol, 2012. **19**(5): p. 455-77.
33. Zerbino, D.R. and E. Birney, *Velvet: algorithms for de novo short read assembly using de Bruijn graphs.* Genome Res, 2008. **18**(5): p. 821-9.
34. Gurevich, A., et al., *QUAST: quality assessment tool for genome assemblies.* Bioinformatics, 2013. **29**(8): p. 1072-5.
35. Hunt, M., et al., *REAPR: a universal tool for genome assembly evaluation.* Genome Biol, 2013. **14**(5): p. R47.
36. Schatz, M.C., et al., *Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies.* Brief Bioinform, 2013. **14**(2): p. 213-24.
37. Li, H. and R. Durbin, *Fast and accurate short read alignment with Burrows-Wheeler transform.* Bioinformatics, 2009. **25**(14): p. 1754-60.
38. Li, H., et al., *The Sequence Alignment/Map format and SAMtools.* Bioinformatics, 2009. **25**(16): p. 2078-9.
39. Chikhi, R. and P. Medvedev, *Informed and automated k-mer size selection for genome assembly.* Bioinformatics, 2014. **30**(1): p. 31-7.
40. Magoc, T., et al., *GAGE-B: an evaluation of genome assemblers for bacterial organisms.* Bioinformatics, 2013. **29**(14): p. 1718-25.

Appendices

Appendix A. One complete control file

```
############################
# General options
############################
genome = /home/xiaofan/iWGS/example/Kazachstania_africana.fa    # the reference genome sequence in fasta format
out_dir = K_africana                                # all outputs will be written to the folder ./K_african
threads = 4                                         # number of CPUs to use (default: 1)
memory = 32                                         # number of GBs of memory to use (default: 8)

############################
# Library options
############################
library = L001,PE,50,100,180,9                      # Illumina 100bp PE library "L001" - coverage: 50x; average insert size: 180bp; SD of insert size: 9bp
library = L002,MP,50,100,8000,400                   # Illumina 100bp MP library "L002" - coverage: 50x; average insert size: 8000bp; SD of insert size: 400bp
library = L003,CLR,60,0.856,0.029                   # PacBio CLR library "L003" - coverage: 60x; average read accuracy: 0.85; SD of read accuracy: 0.02
library = L004,PE,50,250,450,23                     # Illumina 250bp PE library "L004" - coverage: 50x; average insert size: 450bp; SD of insert size: 23bp
library = L005,CLR,10,0.856,0.029                   # PacBio CLR library "L005" - coverage: 50x; average read accuracy: 0.856; SD of read accuracy: 0.029

############################
# Simulator options
############################
pIRS.error_rate =                                   # substitution error rate: 0, 1, or 0.0001-0.63 (default: 1 - indicate that the default setting of pIRS should be used)
pIRS.error_profile =                                # the base-calling profile for simulating substitution-error and quality score (default: the default profile of pIRS)
pIRS.gc =                                           # whether to simulate GC bias: 1 - yes; 0 - no (default: 1)
pIRS.gc_profile =                                   # the GC content-coverage file for simulating GC bias (default: the default profile of pIRS)
pIRS.indel =                                        # whether to simulate indel errors: 1 - yes; 0 - no (default: 1)
pIRS.indel_profile =                                # the InDel-error profile for simulating InDel-error (default: the default profile of pIRS)

ART.qual_shift1 =                                   # the amount to shift the quality score of all first-reads (default: 0)
ART.qual_shift2 =                                   # the amount to shift the quality score of all second-reads (default: 0)
ART.qual_profile1 =                                 # the quality profile of first-reads (default: the default profile of ART)
```

```
ART.qual_profile2 =                    # the quality profile of second-reads (default: the
default profile of ART)

ART.ins_rate1 =                        # the insertion rate of first-reads (default: 0.00009)
ART.ins_rate2 =                        # the insertion rate of second-reads (default:
0.00015)
ART.del_rate1 =                        # the deletion rate of first-reads (default: 0.00011)
ART.del_rate2 =                        # the deletion rate of second-reads (default:
0.00023)

PBSIM.model_qc =                       # the model of quality code for simulating read
accuracy (default: the default profile of PBSIM)
PBSIM.ratio =                          # the ratio of substitution:insertion:deletion errors
(default: 10:60:30)
PBSIM.accuracy_max =                   # the maximum read accuracy (default: 0.90)
PBSIM.accuracy_min =                   # the minimum read accuracy (default: 0.75)
PBSIM.length_mean =                    # the mean of read length (default: 3000)
PBSIM.length_sd =                      # the standard deviation of read length (default:
2300)
PBSIM.length_max =                     # the maximum read length (default: 25000)
PBSIM.length_min =                     # the minimum read length (default: 100)

############################
# Quality control options
############################
QC = L001                              # provide a list of library names (or "all") to perform
quality control on selected (or all) libraries

Trimmomatic.trailing =                 # default: 3; the threshold for quality trimming from
3'-end of reads
Trimmomatic.adapters =                 # a file containing adapter sequence files for each
library to perform adapter trimming
Trimmomatic.minlen =                   # default: 25; minimum read length after
Trimmomatic trimming

NextClip.adapter =                     # default: "CTGTCTCTTATACACATCT"; adapter
sequence for adapter trimming
NextClip.minlen =                      # minimum read length after NextClip trimming

Correction.tool =                      # default: Lighter
Correction.kmer =                      # default: 0 - to estimate based on genome size

############################
# Assembly protocol options
############################
protocol = P001,ABYSS,L001,L002        # assembly protocol "P001" that uses the ABYSS
assembler and libraries "L001" and "L002"
protocol = P002,ALLPATHS,L001,L002     # assembly protocol "P002" that uses the
ALLPATHS-LG assembler and libraries "L001" and "L002"
```

```
protocol = P003,MaSuRCA,L001,L002                # assembly protocol "P003" that uses the MaSuRCA assembler and libraries "L001" and "L002"
protocol = P004,Meraculous,L001,L002             # assembly protocol "P004" that uses the Meraculous assembler and libraries "L001" and "L002"
protocol = P005,Minia,L001,L002                  # assembly protocol "P005" that uses the Minia assembler and libraries "L001" and "L002"
protocol = P006,Platanus,L001,L002               # assembly protocol "P006" that uses the Platanus assembler and libraries "L001" and "L002"
protocol = P007,SGA,L001,L002                    # assembly protocol "P007" that uses the SGA assembler and libraries "L001" and "L002"
protocol = P008,SOAPdenovo2,L001,L002            # assembly protocol "P008" that uses the SOAPdenovo2 assembler and libraries "L001" and "L002"
protocol = P009,SPAdes,L001,L002                 # assembly protocol "P009" that uses the SPAdes assembler and libraries "L001" and "L002"
protocol = P010,Velvet,L001,L002                 # assembly protocol "P010" that uses the Velvet assembler and libraries "L001" and "L002"
protocol = P011,Metassembler,L002                # assembly protocol "P011" that uses the Metassembler assembler and the library "L002" to combine all Illumina-based assemblies
protocol = P012,DISCOVAR,L004                    # assembly protocol "P012" that uses the DISCOVAR de novo assembler and the library "L004"
protocol = P013,CA,L003                          # assembly protocol "P013" that uses the Celera Assembler and the library "L003" - assembly of only PacBio data
protocol = P014,Canu,L003                        # assembly protocol "P014" that uses the Canu assembler and the library "L003" - assembly of only PacBio data
protocol = P015,FALCON,L003                      # assembly protocol "P015" that uses the FALCON assembler and the library "L003" - assembly of only PacBio data
protocol = P016,DBG2OLC,L005                     # assembly protocol "P016" that uses the DBG2OLC assembler and libraries "L005" – hybrid assembly of Illumina (best Illumina-based contig-level assembly) and PacBio data
protocol = P017,SPAdes,L001,L002,L005            # assembly protocol "P017" that uses the SPAdes assembler and libraries "L001", "L002", and "L005" – hybrid assembly of Illumina and PacBio data

#############################
# Assembler options
#############################
ABYSS.kmer =                                     # default: 0 - to be estimated using KmerGenie
ABYSS.option = "l=1 n=5 s=100"                   # GAGE-B recipe

ALLPATHS.ploidy =                                # default: 1; set to 2 for heterozygous assembly

CA.pbCNS =                                       # default 1; set to 0 to use "falcon" instead of "pbdagcon"
CA.sensitive =                                   # default: 0; set to 1 for lower-quality PacBio data

Canu.sensitive =                                 # default: 0; set to 1 for lower-quality PacBio data

MaSuRCA.kmer =                                   # default: 0 - to be selected automatically

Meraculous.kmer =                                # default: 0 - to be selected automatically
Meraculous.diploid =                             # default: 0; set to 1 for diploid dataset
```

```
Minia.kmer =                                    # default: 0 - to be estimated using KmerGenie
Minia.min-abundance =                           # default: 0 - to be estimated using KmerGenie

Platanus.kmer =                                 # default: 0 – to be estimated using KmerGenie

SGA.kmer =                                      # default: 31; the k-mer size for the “correct”
module
SGA.min-overlap =                               # default: 45; the overlap size for the “overlap”
module
SGA.assemble-overlap =                          # default: 75; the overlap size for the “assemble”
module

SPAdes.kmer =                                   # default: 0 - to be estimated using KmerGenie
SPAdes.multi-kmer =                             # default: 1 - to use multiple k-mer sizes
SPAdes.option = “--only-assembler”              # set it empty (i.e. “ ”) to enable the error-correction
module

SOAPdenovo2.kmer =                              # default: 0 - to be estimated using KmerGenie
SOAPdenovo2.option = “-F -R -E -w -u”           # GAGE-B recipe

Velvet.kmer =                                   # default: 0 - to be estimated using KmerGenie
Velvet.option = “-exp_cov auto -scaffolding yes”  # GAGE-B recipe

############################
# Evaluation options
############################
QUAST.eukaryote =                               # whether the reference genome is eukaryotic: 1 -
yes; 0 - no (default: 1)
QUAST.gage =                                    # whether to generate GAGE report: 1 - yes; 0 - no
(default: 1)
QUAST.gene = example/Kazachstania_africana.genes          # gene annotations to be used for
evaluation (default: NA)
REAPR.libs =                                    # use PE library L001 and MP library L002 for REAPR
evaluation (default: NA)

############################
# Executable options
############################
bin.ART =/home/xiaofan/iWGS/tools/ART/art_illumina
bin.pIRS = /home/xiaofan/iWGS/tools/pIRS/pirs
bin.PBSIM = /home/xiaofan/iWGS/tools/PBSIM/bin/pbsim
bin.Trimmomatic = /home/xiaofan/iWGS/tools/Trimmomatic/trimmomatic.jar
bin.NextClip = /home/xiaofan/iWGS/tools/NextClip/bin/nextclip
bin.Lighter = /home/xiaofan/iWGS/tools/Lighter
bin.Quake = /home/xiaofan/iWGS/tools/Quake/bin/quake.py
bin.KmerGenie = /home/xiaofan/iWGS/tools/kmergenie/kmergenie
bin.ABYSS = /home/xiaofan/iWGS/tools/ABYSS/bin/abyss-pe
bin.ALLPATHS = /home/zhoux8/usr/bin/allpathslg/bin/RunAllPathsLG
bin.PBcR = /home/xiaofan/iWGS/tools/CA/bin/PBcR
```

bin.runCA = /home/xiaofan/iWGS/tools/CA/bin/runCA
bin.BLASR = /home/xiaofan/iWGS/tools/CA/bin/blasr
bin.PBDAGCON = /home/xiaofan/iWGS/tools/CA/bin/pbdagcon
bin.Canu = /home/xiaofan/iWGS/tools/Canu/bin/canu
bin.DBG2OLC = /home/xiaofan/iWGS/tools/DBG2OLC/DBG2OLC
bin.Sparc = /home/xiaofan/iWGS/tools/DBG2OLC/Sparc
bin.DISCOVAR = /home/xiaofan/iWGS/tools/Discovar/bin/DiscovarDeNovo
bin.FALCON = /home/xiaofan/iWGS/tools/FALCON/fc_run.py
bin.MaSuRCA = /home/xiaofan/iWGS/tools/MaSuRCA/bin/masurca
bin.Meraculous = /home/xiaofan/iWGS/tools/Meraculous/bin/run_meraculous.sh
bin.Metassembler = /home/xiaofan/iWGS/tools/Metassembler/bin/metassemble
bin.Minia = /home/xiaofan/iWGS/tools/Minia/bin/minia
bin.Platanus = /home/xiaofan/iWGS/tools/Platanus/Platanus
bin.SGA = /home/xiaofan/iWGS/tools/SGA/bin/sga
bin.SOAPdenovo2 = /home/xiaofan/iWGS/tools/SOAPdenovo2/SOAPdenovo2
bin.SPAdes = /home/xiaofan/iWGS/tools/SPAdes/bin/spades.py
bin.dipSPAdes = /home/xiaofan/iWGS/tools/SPAdes/bin/dipspades.py
bin.velvetg = /home/xiaofan/iWGS/tools/Velvet/velvetg
bin.velveth = /home/xiaofan/iWGS/tools/Velvet/velveth
bin.QUAST = /home/xiaofan/iWGS/tools/QUAST/quast.py
bin.AMOS = /home/xiaofan/iWGS/tools/dependencies/bank-transact
bin.REAPR= /home/xiaofan/iWGS/tools/REAPR/reapr
bin.BWA = /home/xiaofan/iWGS/tools/dependencies/bwa
bin.SAMtools = /home/xiaofan/iWGS/tools/dependencies/samtools
bin.FASTX = /home/xiaofan/iWGS/tools/dependencies/fastx_reverse_complement

Appendix B. Exemplar configuration files

Library configuration file: (one example for each read type)

```
##############################
# Library options
##############################

# parameters for the library L001
L001.read_type = PE
L001.depth = 50
L001.simulator = ART
L001.read_length = 100
L001.frag_mean = 180
L001.frag_sd = 9
L001.qual_shift1 = 0
L001.qual_shift2 = 0
L001.qual_profile1 =
L001.qual_profile2 =
L001.ins_rate1 = 0.00009
L001.ins_rate2 = 0.00015
L001.del_rate1 = 0.00011
L001.del_rate2 = 0.00023
L001.QC = 1
L001.tm-trailing = 3
L001.tm-adapter = /home/xiaofan/iWGS/example/TruSeq-PE.fa
L001.tm-minlen = 25
L001.ec-tool = Quake
L001.ec-kmer = 15

# parameters for the library L002
L002.read_type = MP
L002.depth = 50
L002.simulator = ART
L002.read_length = 100
L002.frag_mean = 180
L002.frag_sd = 9
L002.qual_shift1 = 0
L002.qual_shift2 = 0
L002.qual_profile1 =
L002.qual_profile2 =
L002.ins_rate1 = 0.00009
L002.ins_rate2 = 0.00015
L002.del_rate1 = 0.00011
L002.del_rate2 = 0.00023
L002.QC = 0

# parameters for the library L003
L003.read_type = CLR
L003.depth = 60
```

```
L003.simulator = PBSIM
L003.accuracy_mean = 0.856
L003.accuracy_sd = 0.029
L003.accuracy_max = 0.90
L003.accuracy_min = 0.75
L003.model_qc = /home/xiaofan/iWGS/tools/PBSIM/data/model_qc_clr
L003.ratio = 10:60:30
L003.length_mean = 3000
L003.length_sd = 2300
L003.length_max = 25000
L003.length_min = 100
L003.QC = 0

# parameters for the library L004
L004.read_type = PE
L004.depth = 50
L004.simulator = ART
L004.read_length = 250
L004.frag_mean = 450
L004.frag_sd = 23
L004.qual_shift1 = 0
L004.qual_shift2 = 0
L004.qual_profile1 =
L004.qual_profile2 =
L004.ins_rate1 = 0.00009
L004.ins_rate2 = 0.00015
L004.del_rate1 = 0.00011
L004.del_rate2 = 0.00023
L004.QC = 0

# parameters for the library L005
L005.read_type = CLR
L005.depth = 10
L005.simulator = PBSIM
L005.accuracy_mean = 0.856
L005.accuracy_sd = 0.029
L005.accuracy_max = 0.90
L005.accuracy_min = 0.75
L005.model_qc = /home/xiaofan/iWGS/tools/PBSIM/data/model_qc_clr
L005.ratio = 10:60:30
L005.length_mean = 3000
L005.length_sd = 2300
L005.length_max = 25000
L005.length_min = 100
L005.QC = 0
```

Protocol configuration file: (one example for each assembler)

```
##############################
# Assembly protocol options
```

```
##############################

# parameters for the assembly protocol P001
P001.library = L001,L002
P001.assembler = ABYSS
P001.kmer = 0
P001.option = “l=1 n=5 s=100”

# parameters for the assembly protocol P002
P002.library = L001,L002
P002.assembler = ALLPATHS
P002.ploidy = 1

# parameters for the assembly protocol P003
P003.library = L001,L002
P003.assembler = MaSuRCA
P003.kmer = 0

# parameters for the assembly protocol P004
P004.library = L001,L002
P004.assembler = Meraculous
P004.kmer = 0
P004.diploid = 0

# parameters for the assembly protocol P005
P005.library = L001,L002
P005.assembler = Minia
P005.kmer = 0
P005.min-abundance = 0

# parameters for the assembly protocol P006
P006.library = L001,L002
P006.assembler = Platanus
P006.kmer = 0

# parameters for the assembly protocol P007
P007.library = L001,L002
P007.assembler = SGA
P007.kmer = 31
P007.min-overlap = 45
P007.assemble-overlap = 75

# parameters for the assembly protocol P008
P008.library = L001,L002
P008.assembler = SOAPdenovo2
P008.kmer = 0
P008.option = "-F -R -E -w -u"

# parameters for the assembly protocol P009
P009.library = L001,L002
```

```
P009.assembler = SPAdes
P009.kmer = 0
P009.multi-kmer = 1
P009.option = "--only-assembler"

# parameters for the assembly protocol P010
P010.library = L001,L002
P010.assembler = Velvet
P010.kmer = 0
P010.option = "-exp_cov auto -scaffolding yes”

# parameters for the assembly protocol P011
P011.library = L002
P011.assembler = Metassembler

# parameters for the assembly protocol P012
P012.library = L004
P012.assembler = DISCOVAR

# parameters for the assembly protocol P013
P0013.library = L001,L002
P0013.assembler = Velvet
P0013.kmer = 0
P0013.option = “-exp_cov auto -scaffolding yes”

# parameters for the assembly protocol P013
P013.library = L003
P013.assembler = CA
P013.pbCNS = 1
P013.sensitive = 0

# parameters for the assembly protocol P014
P014.library = L003
P014.assembler = Canu
P014.sensitive = 0

# parameters for the assembly protocol P015
P015.library = L003
P015.assembler = FALCON

# parameters for the assembly protocol P016
P016.library = L005
P016.assembler = DBG2OLC

# parameters for the assembly protocol P017
P017.library = L001,L002,L005
P017.assembler = SPAdes
P017.kmer = 0
P017.multi-kmer = 1
P017.option = "--only-assembler"
```

Miscellaneous options configuration file:

```
############################
# General options
############################
genome = /home/xiaofan/iWGS/example/Kazachstania_africana.fa
threads = 4
memory = 32
out_dir = /home/xiaofan/iWGS/example/K_africana

############################
# Evaluation options
############################
QUAST.eukaryote = 1
QUAST.gage = 1
QUAST.gene = /home/xiaofan/iWGS /example/Kazachstania_africana.genes
REAPR.libs =

############################
# Executable options
############################
bin.ART =/home/xiaofan/iWGS/tools/ART/art_illumina
bin.pIRS = /home/xiaofan/iWGS/tools/pIRS/pirs
bin.PBSIM = /home/xiaofan/iWGS/tools/PBSIM/bin/pbsim
bin.Trimmomatic = /home/xiaofan/iWGS/tools/Trimmomatic/trimmomatic.jar
bin.NextClip = /home/xiaofan/iWGS/tools/NextClip/bin/nextclip
bin.Lighter = /home/xiaofan/iWGS/tools/Lighter
bin.Quake = /home/xiaofan/iWGS/tools/Quake/bin/quake.py
bin.KmerGenie = /home/xiaofan/iWGS/tools/kmergenie/kmergenie
bin.ABYSS = /home/xiaofan/iWGS/tools/ABYSS/bin/abyss-pe
bin.ALLPATHS = /home/zhoux8/usr/bin/allpathslg/bin/RunAllPathsLG
bin.PBcR = /home/xiaofan/iWGS/tools/CA/bin/PBcR
bin.runCA = /home/xiaofan/iWGS/tools/CA/bin/runCA
bin.BLASR = /home/xiaofan/iWGS/tools/CA/bin/blasr
bin.PBDAGCON = /home/xiaofan/iWGS/tools/CA/bin/pbdagcon
bin.Canu = /home/xiaofan/iWGS/tools/Canu/bin/canu
bin.DBG2OLC = /home/xiaofan/iWGS/tools/DBG2OLC/DBG2OLC
bin.Sparc = /home/xiaofan/iWGS/tools/DBG2OLC/Sparc
bin.DISCOVAR = /home/xiaofan/iWGS/tools/Discovar/bin/DiscovarDeNovo
bin.FALCON = /home/xiaofan/iWGS/tools/FALCON/fc_run.py
bin.MaSuRCA = /home/xiaofan/iWGS/tools/MaSuRCA/bin/masurca
bin.Meraculous = /home/xiaofan/iWGS/tools/Meraculous/bin/run_meraculous.sh
bin.Metassembler = /home/xiaofan/iWGS/tools/Metassembler/bin/metassemble
bin.Minia = /home/xiaofan/iWGS/tools/Minia/bin/minia
bin.Platanus = /home/xiaofan/iWGS/tools/Platanus/Platanus
bin.SGA = /home/xiaofan/iWGS/tools/SGA/bin/sga
bin.SOAPdenovo2 = /home/xiaofan/iWGS/tools/SOAPdenovo2/SOAPdenovo2
bin.SPAdes = /home/xiaofan/iWGS/tools/SPAdes/bin/spades.py
bin.dipSPAdes = /home/xiaofan/iWGS/tools/SPAdes/bin/dipspades.py
```

bin.velvetg = /home/xiaofan/iWGS/tools/Velvet/velvetg
bin.velveth = /home/xiaofan/iWGS/tools/Velvet/velveth
bin.QUAST = /home/xiaofan/iWGS/tools/QUAST/quast.py
bin.AMOS = /home/xiaofan/iWGS/tools/dependencies/bank-transact
bin.REAPR= /home/xiaofan/iWGS/tools/REAPR/reapr
bin.BWA = /home/xiaofan/iWGS/tools/dependencies/bwa
bin.SAMtools = /home/xiaofan/iWGS/tools/dependencies/samtools
bin.FASTX = /home/xiaofan/iWGS/tools/dependencies/fastx_reverse_complement