Haplotype-resolved sweet potato genome traces back its hexaploidization history

Jun Yang1,2,4, M-Hossein Moeinzadeh2, Heiner Kuhl6, Johannes Helmuth7, Peng Xiao2, Stefan Haas2, Guiling Liu5, Jianli Zheng5, Zhe Sun5, Weijuan Fan3, Gaifang Deng3, Hongxia Wang3, Fenhong Hu3, Shanshan Zhao1, Alisdair R. Fernie4, Stefan Boerno6, Bernd Timmermann6, Peng Zhang3* and Martin Vingron2*

1 Shanghai Key Laboratory of Plant Functional Genomics and Resources, Shanghai Chenshan Plant Science Research Center, Chinese Academy of Sciences, Shanghai Chenshan Botanical Garden, 3888 Chenhua Road, Shanghai 201602, China. 2 Department of Computational Molecular Biology, Max Planck Institute for Molecular Genetics, Ihnestraße 63-73, 14195 Berlin, Germany. 3 National Key Laboratory of Plant Molecular Genetics, CAS Center for Excellence in Molecular Plant Sciences, Institute of Plant Physiology and Ecology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, Shanghai 200032, China. 4 Department of Molecular Physiology, Max Planck Institute of Molecular Plant Physiology, Am Mühlenberg 1, 14476 Potsdam, Germany. 5 Tai'an Academy of Agricultural Sciences, 16 Tai’ai Road, Tai’an, 271000 Shandong, China. 6 Max Planck Institute for Molecular Genetics, Sequencing Core Facility, Ihnestraße 63-73, 14195 Berlin, Germany. 7 Max Planck Institute for Molecular Genetics, Otto-Warburg-Laboratory: Computational Epigenomics Group, Ihnestraße 63-73, 14195 Berlin, Germany. Jun Yang and M-Hossein Moeinzadeh contributed equally to this work.

*e-mail: zhangpeng@sibs.ac.cn; vingron@molgen.mpg.de

In the format provided by the authors and unedited.
**Supplementary Figure 1**

**De novo assembly pipeline of haplotype-resolved hexaploid *Ipomoea batatas***. Genomic DNA was extracted from Taizhong6 *in vitro* plant and processed as follows. (1) DNA was fragmented into different sizes according to paired-end (lines) or mate-pair (circles) library requirements. (2) DNA sequences were obtained by Illumina sequencing of paired-end and mate-pair libraries. (3) *De novo* assembly of short reads and seed finding based on variant calling. (4) Phasing haplotypes by extending more strongly supported seed regions. (5) Merging overlapped haplotypes into longer haplotypes. (6) Mapping all raw reads against phased haplotypes. (7) Scaffolding based on haplotypes and generating of consensus genome.
Coverage distribution of genomic regions phased into triploid, tetraploid, pentaploid, and hexaploid during genome survey. Based on the genome survey data and primary assembly, genome regions have been phased in up to six haplotypes. The coverage of each phased region and number of phased haplotypes is summarized here. The peak coverage around 40 in hexaploid indicated the minimal sequencing depth requirement for haplotyping of hexaploid genome. The peak coverage shifting from triploid to hexaploid demonstrated the insufficient sequencing depth in genome survey stage.
Supplementary Figure 3

The insert size distribution of sequenced paired-end libraries. (a) Insert size distribution of paired-end libraries A500, L500, and A1kb. (b) Insert size distribution of mate-pair libraries MP and AMP.
A snapshot of mapping results of contigs against scaffolds. The large number of single nucleotide mismatches indicates the single nucleotide polymorphism between homologous chromosomes in *Ipomoea batatas*. Cigar fields of these mapped scaffolds were listed as follows. (a) 4045M. (b) 33S44M1D136M1I10M6D149M15D21M1I1799M. (c) 579M. (d) 56M13D162M. (e) 15S2396M2I92M. (f) 733M. (g) 1260M. (h) 1014M1I162M1D136M. Among these, (b), (e), (g) & (h) are partially shown here.
Supplementary Figure 5

Summary of variations in scaffolds in preliminary assembly. (a) Number of variations and length of all scaffolds. An isolated red dot represents the fully assembled genome of sweet potato endophyte, *Bacillus pumilus*. (b) High correlation (0.975) between “Number of variations” and “Scaffold length” after excluding the endophyte *Bacillus pumilus* genome.
Comparison between scaffolds, contigs and haplotypes mapping results. (a) Scaffolds were mapped against scaffolds. (b) Contigs were mapped against scaffolds. (c) Haplotypes were mapped back to scaffolds. (d) Collinearity checking before removing of a large scaffold (19kb, red). (e) Collinearity rechecking before removing of a large scaffold (17kb, red). The higher density of mismatches in scaffolds and contigs (a and b) than haplotypes (c) indicated the problems for traditional assembling.
Supplementary Figure 7

Haplotype evaluation by 454 reads. x axis: “Match” is the number of coinciding polymorphic sites between haplotype and 454 read. y axis: “Mismatch” is the number of different polymorphic sites in the overlap. Color indicates frequency of the respective (haplotype, 454 read) pairs, ranging from red to purple (on an exponential scale). (a) Evaluation of haplotypes from single mode. (b) Less than 6 mismatch part of single mode (97.25% of total number of overlaps). There were 64.78% of overlaps between haplotypes and 454 reads are identical at variant loci (y = 0). Even with strict mismatch threshold, a large fraction of 454 reads are supporting haplotypes reconstructed by short reads.
**Supplementary Figure 8**

Distribution of six evolutionary topologies over all phased regions. The majority of the topologies for the 644,360 phased regions are grouped into 2 haplotypes versus 4 haplotypes (green blue and blue bars). Among the 4 haplotypes subgroup, 2 versus 2 is dominant (blue bar).
Haplotype-resolved sweet potato genome traces back its hexaploidization history

Jun Yang\textsuperscript{1,2,4,8}, M-Hossein Moeinzadeh\textsuperscript{2,8}, Heiner Kuhl\textsuperscript{6}, Johannes Helmuth\textsuperscript{7}, Peng Xiao\textsuperscript{2}, Stefan Haas\textsuperscript{2}, Guiling Liu\textsuperscript{5}, Jianli Zheng\textsuperscript{5}, Zhe Sun\textsuperscript{5}, Weijuan Fan\textsuperscript{3}, Gaifang Deng\textsuperscript{3}, Hongxia Wang\textsuperscript{3}, Fenhong Hu\textsuperscript{3}, Shanshan Zhao\textsuperscript{1}, Alisdair R Fernie\textsuperscript{4}, Stefan Boerno\textsuperscript{6}, Bernd Timmermann\textsuperscript{6}, Peng Zhang\textsuperscript{3}*, & Martin Vingron\textsuperscript{2}*

\textsuperscript{1}Shanghai Key Laboratory of Plant Functional Genomics and Resources, Shanghai Chenshan Plant Science Research Center, Chinese Academy of Sciences, Shanghai Chenshan Botanical Garden, 3888 Chenhua Road, Shanghai, 201602, China.
\textsuperscript{2}Max Planck Institute for Molecular Genetics, Department of Computational Molecular Biology, Ihnestraße 63-73, Berlin 14195, Germany.
\textsuperscript{3}Institute of Plant Physiology and Ecology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, 300 Fenglin Road, Shanghai, 200032, China.
\textsuperscript{4}Department of Molecular Physiology, Max Planck Institute of Molecular Plant Physiology, Am Mühlenberg 1, Potsdam 14476, Germany
\textsuperscript{5}Tai'an Academy of Agricultural Sciences, 16 Tailai Road, Tai'an, 271000, Shandong, China.
\textsuperscript{6}Max Planck Institute for Molecular Genetics, Sequencing Core Facility, Ihnestraße 63-73, Berlin 14195, Germany.
\textsuperscript{7}Max Planck Institute for Molecular Genetics, Otto-Warburg-Laboratory: Computational Epigenomics Group, Ihnestrasse 63-73, Berlin14195, Germany.
\textsuperscript{8}These authors contributed equally to this work.

Correspondence should be addressed to P. Z. (zhangpeng@sibs.ac.cn) or M. V. (vingron@molgen.mpg.de).
Contents

1 Genome sequencing and primary assembly ........................................... 2
  1.1 Plant material .............................................................................. 2
  1.2 Genomic DNA extraction and sequencing ..................................... 2
  1.3 Primary assembly pipeline .......................................................... 2

2 Haplotype phasing of hexaploid genome ........................................... 5
  2.1 Variant calling ........................................................................... 5
  2.2 Haplotype phasing ..................................................................... 5

3 Haplotype aided assembly ................................................................. 11
  3.1 Haplotype connections extraction ................................................. 11
  3.2 SSPACE scaffolding .................................................................. 13
  3.3 Anchoring scaffolds to pseudochromosomes ............................... 15
  3.4 Remove redundancy in preliminary assembly contigs .............. 17
  3.5 BUSCO assessment of assembly quality ..................................... 18

4 Genome annotation and cDNA validation ......................................... 20
  4.1 Gene model identification based on transcriptome ...................... 20
  4.2 Gene model validation by Sanger cDNA sequencing ................. 21
  4.3 Gene annotation using publicly available protein database ....... 21
  4.4 Putative gene clusters identification .......................................... 22
  4.5 Repeat sequence identification based on repeat database ........... 23

5 Updating and validation of haplotypes .............................................. 25
  5.1 Haplotype updating ................................................................. 25
  5.2 Haplotype validation ............................................................... 26
  5.3 Variant updating in phased regions ............................................ 27

6 Phylogenetic analysis of homologous chromosomal regions .......... 28

References ......................................................................................... 30
1 Genome sequencing and primary assembly

1.1 Plant material
A newly bred carotenoid rich cultivar of sweet potato (*Ipomoea batatas*), Taizhong6, China national accession number 2013003, was kindly provided by our collaborator, Ms. Guiling Liu from Tai’an Academy of Agricultural Sciences. *In vitro* cultured plants were established for genome sequencing. The entire *in vitro* plant was collected for genomic DNA extraction.

1.2 Genomic DNA extraction and sequencing
Total genomic DNA was isolated from *in vitro* cultured plants following the method described by Kim and Hamada[1]. In total, six sequencing libraries were constructed and sequenced on Hiseq2500, Nextseq500, Hiseq4000 and GS FLX+ platforms (Supplementary Table 1, A500, A1kb, L500, MP, AMP and A454) according to the manufacturer’s instructions (Illumina, Inc. and Roche Applied Science).

1.3 Primary assembly pipeline
The main steps of preliminary assembly are as follows.

a) Read correction of all Illumina data using the BFC package (https://github.com/lh3/bfc)

```
kmc -ci4 -r -t16 -k33 PAIRED_END-READS KMC-KMER-33-ci4
bfc-kmc -t12 -Q KMC-KMER-33-ci4 READS-file > READS-file.corrected.fa
```

We kept only read pairs that match “ec:Z:T” in the headers, which means correction was successful.

b) Assembling all short reads by IDBA-UD

```
idba_ud_tweaked -r ALL-READS -o IDBA_ALL --mink 100 --maxk 151 --step 23 --num_threads 40 --seed_kmer 100 --similar 0.99 --no_coverage --no_correct
```

The best contig N50 was observed for the largest kmer step, but still highly fragmented:

- **Contigs:** 7,520,052
- **Max:** 1,010,392 bp
- **Mean:** 338 bp
- **N50:** 369 bp
- **N80:** 200 bp
- **Total length:** 2,542,825,653 bp

c) Further assembly of the IDBA outputs by a long read assembler (NEWBLER 3.0)
IDBA_UD outputs for contigs and locally reassembled contigs were used. Large contigs were split into 29kbp chunks that have 4kb overlaps. Local contigs were split at "N" letters. The data was converted to FastQ format.

NEWBLET v3.0 was used for assembly, changing two parameters in the default values:
overlapMinMatchLength=151, overlapMinMatchIdentity=99

Improved assembly:

Contigs: 1,559,267
Max: 406,899 bp
Mean: 834 bp
N50: 1,771 bp
N80: 568 bp
Total length: 1,299,323,989 bp

d) Two scaffolding runs using the PLATANUS scaffoldor were performed on NEWBLET 3.0 output. During the first run the median insert size of the libraries was set to the values observed in the distribution peaks (Supplementary Figure 3). In the second run all of the scaffolds from the first run were re-scaffolded using MP library with median insert size of 6000 allowing larger standard deviations. It thereby used connections inferred from longer mate pairs that were present in the library.

Changing headers of NEWBLET output for compatibility with PLATANUS scaffoldor:

i.e. ">contig00001 length=406899 numreads=299" was changed to ">contig00001_len406899_cov0"

platanus scaffold -o output1 -c Contigs_for_PLATANUS.fa -ip1 PE500 -ip2 PE1000 -ip3 MP2000 -n1 200 -n2 500 -a3 2500 -d3 1000 -t 24
platanus scaffold -t 40 -l 6 -o output2 -c output1.fa -ip1 MP2000 -a1 6000 -d1 2000

e) Gap-closing using all corrected Illumina reads (PLATANUS GAPCLOSER)

platanus gap_close -o PLATANUS_GAPCLOSE -c SP3b.fa -ip1 PE500 -ip2 PE1000 -ip3 MP2000 -t 40 -d 10000

The results of primary assembly are SP3_scaffolds.fa and SP3_contigs.fa which are summarized as follows.

SP3_scaffolds.fa
count: 79,089
total length: 869,598,330 bp
total bases: 799,703,596 bp
total gap N: 69,894,734 bp
largest scaffold: 3,723,026 bp
(a Bacillus pumilis genome that seems to be assembled completely!)
largest scaffold contig: 1,101,905 bp
(part of the B. pumilis scaffold)
scaffold N50 length:  60,299 bp
scaffold N50 count:  3,796
scaffold contig N50 length:  5,649 bp
scaffold contig N50 count:  41,637

SP3_contigs.fa
count:  991,314
total length:  435,867,107 bp
largest contig:  21,429 bp
contig N50 length:  708 bp
contig N50 count:  143,594
contigs larger 1000 bp:  175,679,534 bp in 92,790 contigs

f) Remove redundancy in preliminary assembly scaffolds
A self-to-self blast was employed to identify redundancy in preliminary assembly. Following blat command was parallelized in computer farm. All hits with identity higher than 85 and more than 85% covered by longer scaffold were summarized in all_85_85_nor.

blat SP3_scaffolds.fa scaffolds.tmp -out=blast8 blat.tmp
sort -k2 all_85_85_nor > rm_scflist
cut -f 2 rm_scflist | sort > rm_scflist.sort
Several long candidates in blast results have been manually checked via Circos visualization prior to removing (Supplementary Figure 6d, e).

Exclude the endophyte Bacillus pumilus genome, the largest scaffold in preliminary assembly. vi rm_scflist.sort
Get the clean non-redundancy scaffold sequences.
join -j 1 v3.length.sort rm_scflist.sort -v 1 awk '{if($3>0){print $1}}' > v3finelist
get_contigs.pl v3finelist SP3_scaffolds.fa > v3fine.fa &

g) PLATANUS scaffolding using 20kb insertion library
One scaffolding run using the PLATANUS scaffolder were performed on v3fine.fa using 20kb insertion library.
sed 's/scf.*/&_cov0/g' ../hiseq/v3fine.fa > v3fine_PLATANUS.fna &
platanus scaffold -c v3fine_PLATANUS.fna -o v4fine -OP1 <(zcat ../hiseq/mpimg_L9275-1_B47-1_S35_R1_001.fastq.gz) <(zcat ../hiseq/mpimg_L9275-1_B47-1_S35_R2_001.fastq.gz) -t 16 -a 1 15000 -d 1 5000 2> scaffold.3.log
2 Haplotype phasing of hexaploid genome

2.1 Variant calling

All the Illumina raw reads were mapped back to all scaffolds in the previous step (Scaffolds of preliminary assembly, Table 1) by combination of BWA (Version 0.7.12-r1039) and Samtools (Version 1.4) as follows.

```bash
bwa index -p v4fine v4fine_scaffold.3.fa
bwa mem v4fine AI_500_GTGAAA_L005_R1_001.fastq.gz AI_500_GTGAAA_L005_R2_001.fastq.gz -t 30 | samtools view -@ 5 -Su - | samtools sort -@ 5 -T 500 -o v4fine.500.bam &
bwa mem v4fine AI_1000bp_2_ATTCCT_L003_R1_001.fastq.gz AI_1000bp_2_ATTCCT_L003_R2_001.fastq.gz -t 48 | samtools view -@ 5 -Su - | samtools sort -@ 5 -T 454 -o v4fine.1kb.bam &
bwa mem v4fine mpimg_L3956-2_AI500-2_S1_L001_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L001_R2_001.fastq.gz -t 36 | samtools view -@ 10 -Su - | samtools sort -@ 10 -T L01 -o v4fine.L01.bam &
bwa mem v4fine MP_R1.fastq.gz MP_R2.fastq.gz -t 48 | samtools view -@ 5 -Su - | samtools sort -@ 5 -T MPN -o v4fine.MPN.bam &
bwa mem v4fine mpimg_L9275-1_B47-1_S35_R1_001.fastq.gz mpimg_L9275-1_B47-1_S35_R2_001.fastq.gz -t 48 | samtools view -@ 5 -Su - | samtools sort -@ 5 -T 20k -o v4fine.20k.bam &
```

All sorted BAM files were merged as one file:

```bash
samtools merge -@ 32 ai.bam v4fine.500.bam v4fine.1kb.bam v4fine.L01.bam v4fine.L02.bam v4fine.L03.bam v4fine.L04.bam v4fine.MPN.bam v4fine.20k.bam &
```

PCR duplicates were removed:

```
Samtools rmdup ai.bam ai.rmdup.bam &
```

Freebayes (Version 0.9.14-19-g8a407cf) was employed for variant calling in the hexaploid genome.

```bash
samtools faidx v4fine_scaffold.3.fa &
fasta_generate_regions.py v4fine_scaffold.3.fa.fai 100000 > v4fine.100kb &
freebayes-parallel v4fine.100kb 48 -f v4fine_scaffold.3.fa -p 6 -F 0.08 ai.rmdup.bam > ais6.08.vcf &
```

The variant calling result was summarized in Figure 2.

2.2 Haplotype phasing

We developed a method, called ‘Ranbow’, for:
1) Haplotype reconstruction of polyploid genomes
2) Correcting the sequence variants (the vcf file)
3) Investigating the evolutionary history of the organism

Rainbow is implemented in Python and is available from https://www.molgen.mpg.de/ranbow. The whole haplotyping pipeline can be run from the script (ranbow.py) in the Linux shell. Parameters need to be set in a parameter file or be passed as shell arguments. Rainbow has three main modes of function namely,

- ranbow hap: for haplotyping
- ranbow eval: for evaluating of the assemble haplotypes by gold standard reads
- ranbow phylo: for the phylogenetic analysis

The parameter file path is given to the main script with the argument ‘-par’. Therefore ranbow for haplotyping can be run with the following command:

```
python2.7 ranbow.py hap -par input.params
```

Rainbow hap takes fasta, bam, and vcf formatted files as inputs to generate phased haplotypes in hap (This format will be explained later), fasta, and bam formats. The input paths and the other parameters can be passed as arguments in command line as well as being edited in parameter file. All parameters and settings are categorized as follows:

**Main input:**

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-refFile</td>
<td>path to file</td>
<td>references (.fasta)</td>
</tr>
<tr>
<td>-bamFile</td>
<td>path to file</td>
<td>mapped reads (.bam)</td>
</tr>
<tr>
<td>-vcfFile</td>
<td>path to file</td>
<td>Sequence variants (.vcf)</td>
</tr>
<tr>
<td>-selectedScf</td>
<td>path to file</td>
<td>selected scaffolds or regions for phasing</td>
</tr>
<tr>
<td>-bamFileEval (only for eval mode)</td>
<td>path to file</td>
<td>Mapped reads (.bam)</td>
</tr>
<tr>
<td>-outputFolderBase</td>
<td>path to folder</td>
<td>All the resulting files will be generated in this folder</td>
</tr>
</tbody>
</table>
Behavior:

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-mode</td>
<td>index, hap, collect, modVCF</td>
<td>Modes of functions</td>
</tr>
<tr>
<td>-ploidy</td>
<td>number</td>
<td>The ploidy of the polyploid genome</td>
</tr>
<tr>
<td>-noProcessor</td>
<td>number</td>
<td>The number of available processors</td>
</tr>
<tr>
<td>-processorIndex</td>
<td>A number in the range of [0, noProcessor)</td>
<td>only for multiprocessing in...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rainbow hap –mode hap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rainbow eval –mode run</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rainbow phylo –mode run</td>
</tr>
</tbody>
</table>

To clarify, here we show the whole pipeline step by step.

The content of parameter file (input.params) for the haplotyping could be as follows:
- ploidy 6
- noProcessor 4
- bamFile path to/file.bam
- refFile path to/file.fasta
- vcfFile path to/file.vcf
- selectedScf path to/file.scaffolds.list
- outputFolderBase path to folder~/RANBOW/toy/result

**Indexing (mode: index)**
The command for indexing the fasta, vcf, and bam files is:

```bash
python2.7 rainbow.py hap-par input.params –mode index
```

**Runing haplotyper (mode: hap)**
In order to run the haplotyper “-noProcessor” may be adjusted according to the number of available processors. For example if the code is running on cluster with 200 cores, “-noProcessor” can be set to 200. Then, 200 independent jobs will be executed with different set of scaffolds. These 200 jobs are independent and can be run on different machines as well. The -processorIndex is a compulsory parameter in command line if the number of processors is more than 1 (-noProcessor > 1). Otherwise -noProcessor is set to 1 and -processorIndex is set
to 0 by default, meaning that the output is generated with one processor and the result is going to be generated in a folder named 0.

The command for running haplotyper using one processor (as an example it is set to 7) is:

```python
python2.7 ranbow.py hap -par input.params --mode index --processorIndex 7
```

To run the code in parallel on one machine the following code can be executed:

```bash
for i in {0..30}
do
   python2_7_13 ranbow.py hap --mode hap --par hap.params --processorIndex $i > $i.log &
done
```

The standard outputs for each processor is collected in “0.log”, “1.log”, to “30.log” files.

It is also possible to run Ranbow partly in one machine and partly in another machine. Let’s assume that two machine (machine A and B) are available with 30 and 20 processors respectively.

**Machin A:**

```bash
for i in {0..29}
do
   python2_7_13 ranbow.py hap --mode hap --par hap.params --processorIndex $i > $i.log &
done
```

**Machin B:**

```bash
for i in {30..49}
do
   python2_7_13 ranbow.py hap --mode hap --par hap.params --processorIndex $i > $i.log &
done
```

**Collecting the generated data from different processors (mode: collect)**
As all jobs get finished in all machines, “-mode collect” can be executed to collect the haplotypes from different machines:

```python
python2.7 ranbow.py hap -par input.params --mode collect
```

This command generates the fasta and bam files for the assembled haplotypes. Moreover it generates the hap formatted file. These files includes Haplotype name, Start position, Haplotype, Number of error correction, Haplotype quality, and Supporting fragments. These fields are explained in more details as follows.
**Haplotype name**: Each block and each haplotype in a block have a tag number. We code each haplotype name with its block and haplotype tags, e.g. in “12_0”, block number is 12 and haplotype number is 0. Then, if two or more haplotypes are merged in downstream analysis (bridge and pair), we merge the names with putting “N” symbols in between, e.g. 12_0N29_3.

**Start position**: Haplotype beginning in ordered list of all variant positions.

**Haplotype**: A haplotype coded by “0” to “5”.

**Number of error correction**: Each haplotype has a number of supporting fragments. This field represents the number of mismatch among these fragments.

**Quality of haplotype**: The supporting fragment coverage of each position in the haplotype. Length of quality equals to the length of haplotype.

**Supporting fragments**: A full list of supporting fragments.

Haplotyping results are summarized as follows.

<table>
<thead>
<tr>
<th>File</th>
<th>Count</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_exten.fasta</td>
<td>4,169,940</td>
<td>1,734,469,040 bp</td>
</tr>
<tr>
<td>all_bridge.fasta</td>
<td>3,614,700</td>
<td>1,646,394,644 bp</td>
</tr>
<tr>
<td>all_pair.fasta</td>
<td>2,565,258</td>
<td>2,237,680,867 bp</td>
</tr>
</tbody>
</table>

To evaluate the haplotyping accuracy, Roche 454 trimmed reads were mapped against SP3_scaffolds.fa.

```bash
bwa mem SP3_scaffolds.fa 454TrimmedReads.fastq -t 48 &
samtools view -@ 32 -b -o 454.bam -S 454.sam &
samtools sort -@ 32 454.bam 454.sorted &
samtools index 454.sorted.bam &
```

The content of parameter file for comparing 454 reads and the assembled haplotypes is as follows:

```
-ploidy 6
-noProcessor 4
-bamFile path to/file.bam
```
-refFile path to/file.fasta
-vcfFile path to/file.vcf
-selectedScf path to/file.scaffolds.list
-outputFolderBase path to folder~>/RANBOW/toy/result
-bamFileEval path to/454.bam

Ranbow eval has also this mode of being executed in different machine. For obtaining the evaluation result the following steps needs to be done.

Index:
```
python2_7 ranbow.py eval -par hap.params -mode index
```

mode run in parallel:
```
python2_7 ranbow.py eval -par hap.params -mode run -processorIndex i
```

mode collect:
```
python2_7 ranbow.py eval -par hap.params -mode collect
```

The results can be found in:
```
path to: outputFolderBase/eval/result.sing
path to: outputFolderBase/eval/result.pair
```

File “result.sing” is the evaluation of haplotypes which are assembled from single reads while in file “result.pair” pair end information is also utilized. The two column files indicate the “Match”, “Mismatch” of the overlaps between phased haplotypes and 454 reads.
3 Haplotype aided assembly

3.1 Haplotype connections extraction

All the Illumina raw reads were mapped against “all_bridge.fasta” sequences. Only perfect matched paired-end reads were considered as haplotype connections. The inter-scaffolds and inner-scaffolds connections were separated for haplotype-based scaffolding and haplotype elongation, respectively.

Mapping raw reads back to all_bridge.fasta.

bwa index all_bridge.fasta
bwa mem all_bridge.fasta MP_R1.fastq.gz MP_R2.fastq.gz -t 48 >mpb.sam &

bwa mem all_bridge.fasta AI_500_GTGAAA_L005_R1_001.fastq.gz AI_500_GTGAAA_L005_R2_001.fastq.gz -t 48 >a500b.sam &

bwa mem all_bridgefasta AI_1000bp_2_ATTCCT_L003_R1_001.fastq.gz AI_1000bp_2_ATTCCT_L003_R2_001.fastq.gz -t 48 >a1kbb.sam &

bwa mem all_bridgefasta mpimg_L3956-2_AI500-2_S1_L001_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L001_R2_001.fastq.gz -t 28 >l001b.sam &

bwa mem all_bridgefasta mpimg_L3956-2_AI500-2_S1_L002_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L002_R2_001.fastq.gz -t 28 >l002b.sam &

bwa mem all_bridgefasta mpimg_L3956-2_AI500-2_S1_L003_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L003_R2_001.fastq.gz -t 28 >l003b.sam &

bwa mem all_bridgefasta mpimg_L3956-2_AI500-2_S1_L004_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L004_R2_001.fastq.gz -t 32 >l004b.sam &

All the SAM files were converted into BAM files for sorting and removing of PCR duplicates.

samtools view -@ 48 -b -o mpb.bam -S mpb.sam &
samtools sort -@ 48 mpb.bam mpb.sorted &
samtools rmdup mpb.sorted mpb.rmdup.bam &
samtools view -@ 48 -b -o a500b.bam -S a500b.sam &
samtools sort -@ 20 a500b.bam a500b.sorted &
samtools rmdup a500b.sorted a500b.rmdup.bam &
samtools view -@ 48 -b -o a1kbb.bam -S a1kbb.sam &
samtools sort -@ 20 a1kbb.bam a1kbb.sorted &
samtools rmdup a1kbb.sorted a1kbb.rmdup.bam &
samtools view -@ 48 -b -o l001b.bam -S l001b.sam &
samtools sort -@ 20 l001b.bam l001b.sorted &
samtools rmdup l001b.sorted l001b.rmdup.bam &
samtools view -@ 48 -b -o l002b.bam -S l002b.sam &
samtools sort -@ 20 l002b.bam l002b.sorted &
samtools rmdup l002b.sorted l002b.rmdup.bam &
samtools view -@ 48 -b -o l003b.bam -S l003b.sam &
samtools sort -@ 20 l003b.bam l003b.sorted &
samtools rmdup l003b.sorted l003b.rmdup.bam &

depending on the size of reads

Extracting haplotype connections.
A python script is employed to convert all perfect matched paired-end reads information into SSPACE tab format.

```
python sam2tab mpb.perfect.sam>mpb.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab a500b.perfect.sam>a500b.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab a1kbb.perfect.sam>a1kbb.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab l001b.perfect.sam>l001b.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab l002b.perfect.sam>l002b.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab l003b.perfect.sam>l003b.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
python sam2tab l004b.perfect.sam>l004b.rmdup.sam.nodeNames-pos.sorted.justPE.justConnections
```
3.2 SSPACE scaffolding

Paired-end reads connect haplotypes from different scaffolds were extracted to generate connection tables for SSPACE (Version 3.0) scaffolding.

```
sort v3finelist >v3finelist.sort &
sort mp.sspace.tab >mp.sspace.tab.sort
join -j 1 mp.sspace.tab.sort v3finelist.sort >mp.sspace.tab.sort1 &
sort -k4 mp.sspace.tab.sort1 >mp.sspace.tab.sort2 &
join -1 4 -2 1 mp.sspace.tab.sort2 v3finelist.sort | awk '{print $2,$3,$4,$1,$5,$6}' > mp.sspace.tab.rmscf &
```

```
sort a500bp.sspace.tab >a500bp.sspace.tab.sort
join -j 1 a500bp.sspace.tab.sort v3finelist.sort >a500bp.sspace.tab.sort1 &
sort -k4 a500bp.sspace.tab.sort1 >a500bp.sspace.tab.sort2 &
join -1 4 -2 1 a500bp.sspace.tab.sort2 v3finelist.sort | awk '{print $2,$3,$4,$1,$5,$6}' >a500bp.sspace.tab.rmscf &
```

```
sort samfiles/inter/a1kbp.sspace.tab >a1kbp.sspace.tab.sort
join -j 1 a1kbp.sspace.tab.sort v3finelist.sort >a1kbp.sspace.tab.sort1 &
sort -k4 a1kbp.sspace.tab.sort1 >a1kbp.sspace.tab.sort2 &
join -1 4 -2 1 a1kbp.sspace.tab.sort2 v3finelist.sort | awk '{print $2,$3,$4,$1,$5,$6}' >a1kbp.sspace.tab.rmscf &
```

Based on the insert size distribution of all paired-end libraries (Supplementary Figure 3), the library settings for SSPACE were as follows.

```
a500 TAB a500v4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 350 0.75 FR
l500 TAB l500v4fine_hap_pair.rmdup.bam150M.sam.sspaceIn 550 0.75 FR
a1kbp TAB a1kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 950 0.75 FR
amp1 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 1500 0.75 RF
mp-1 TAB mpv4fine_hap_pair.rmdup.bam150M.sam1.sspaceIn 1500 0.75 RF
amp2 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 3000 0.75 RF
mp-2 TAB mpv4fine_hap_pair.rmdup.bam150M.sam1.sspaceIn 3000 0.75 RF
amp3 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 6000 0.75 RF
mp-3 TAB mpv4fine_hap_pair.rmdup.bam150M.sam1.sspaceIn 6000 0.75 RF
amp4 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 9000 0.75 RF
mp-4 TAB mpv4fine_hap_pair.rmdup.bam150M.sam1.sspaceIn 9000 0.75 RF
amp5 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 12000 0.75 RF
mp-5 TAB mpv4fine_hap_pair.rmdup.bam150M.sam1.sspaceIn 12000 0.75 RF
amp6 TAB a20kv4fine_hap_pair.rmdup.bam100M.sam.sspaceIn 15000 0.75 RF
```

Summary of input sequence file: v4fine.fa.

Total number of contigs = 57051
Sum (bp) = 831919670
Total number of N's = 95841843
Sum (bp) no N's = 736077827
GC Content = 35.15%
Max contig size = 1152062
Min contig size = 392
Average contig size = 14582
N25 = 288887
N50 = 142553
N75 = 26854

Scaffolding command and parameter settings
SSPACE.Standard_v3.0.pl -l lib_v4fine6.txt -s v4fine.fa -k 5 -a 0.75 -T 8 -S 0 -b lib_v4fine6 &

Summary of final output file: lib_v4fine6.final.scaffolds.fasta.

Total number of scaffolds = 35919
Sum (bp) = 836316092
Total number of N's = 100239550
Sum (bp) no N's = 736076542
GC Content = 35.15%
Max scaffold size = 1335955
Min scaffold size = 392
Average scaffold size = 23283
N25 = 369546
N50 = 200728
N75 = 71586
3.3 Anchoring scaffolds to pseudochromosomes

To anchor scaffolds to pseudochromosomes according to gene synteny, we need to extract the coding regions of the scaffolds.

Mapping transcriptome data as follows.

```bash
hisat-build v4fine_lib_v4fine6.final.scaffolds.fasta
hisat -p 20 -x v4fine -1 CHG006968-1605096-xushu18-10_S11_L006_R1_001.fastq.gz,CHG006968-1605096-
xushu18-2_S12_L006_R1_001.fastq.gz,CHG006968-1605096-xushu18-6_S13_L006_R1_001.fastq.gz,CHG006968-
1605096-yuzi7hao-3_S15_L006_R1_001.fastq.gz,CHG006968-1605096-yuzi7hao-
-S16_L006_R1_001.fastq.gz
hisat -p 20 -x v4fine -1 CHG006968-1605096-xushu18-10_S11_L006_R2_001.fastq.gz,CHG006968-
1605096-xushu18-2_S12_L006_R2_001.fastq.gz,CHG006968-1605096-xushu18-6_S1
3_L006_R2_001.fastq.gz,CHG006968-1605096-yuzi7hao-1_S14_L006_R2_001.fastq.gz,CHG006968-1605096-
yuzi7hao-3_S15_L006_R2_001.fastq.gz,CHG006968-1605096-yuzi7hao-6_S16_L006
_R2_001.fastq.gz
samtools view -@ 20 - Su xt.sam &
``` 

Extracting coding regions using StringTie (Version 1.2.2).

```
stringtiexy.bam -l v4fine -f 0.18 -o v4fine.xy.gtf
```

Dealing with hanging reads at the end of scaffolds.

```
length.pl lib_v4fine6.final.scaffolds.fasta | sort | join -j 1 - <(sort v4fine.xy.gtf)|awk '{if($3<$7){print $1 "	"$4,$5,$6,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18} else {print $1"\t"$2"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9
$s10,$11,$12}'} >v4fine6.xy.clean.gtf &
```

Preparing input files for ALLMAPS runs.

```
python -m jcvi.formats.gfffromgtf v4fine6.xy.clean.gtf -o v4fine6.clean.gff
python -m jcvi.formats.gffbed --type=mRNA --key=ID v4fine6.clean.gff -o v4fine6.ori.bed
sed -e 's/\s/-/1' -e 's/\s/-/1' v4fine6.ori.bed >v4fine6.bed
gffreadv4fine6.clean.gtf -g lib_v4fine6.final.scaffolds.fasta -w v4fine6.clean.fa &
```

The input files for ALLMAPS runs are four files: v4fine6.bed, v4fine6.cds, nil.bed and nil.cds.

```
python -m jcvi.compara.catalogortholog v4fine6 nil
python -m jcvi.library.allmapsmergebed v4fine6.nil.converted.bed -o v4fine6.nil.merged.bed
python -m jcvi.library.allmaps path v4fine6.nil.merged.bed lib_v4fine6.final.scaffolds.fasta
```
By changing parameters of ortholog finding process, via built-in MCscan (Python version) of ALLMAPS or standalone MCScan (http://chibba.pgml.uga.edu/mcscan2/), one can optimize the anchoring process. Finally, there are 7,470 relatively larger scaffolds, which account for 75.7% total length of all scaffolds, has been anchored to 15 pseudochromosomes.

Summary of the 15 pseudochromosomes is as follows.

**Number of Sequences** = 15

**Length Statistics [bp]**

- *Total Length* = 633,423,954 (633Mb)
- *Largest* = 52,844,149
- *Smallest* = 32,277,352
- *Mean* = 42,228,263
- *Median* = 41,911,220
- *StdDev* = 6,076,957

**Base Composition**

- A = 28.2%
- C = 15.2%
- G = 15.2%
- T = 28.2%
- G+C = 30.4%
- A+T = 56.5%
- N = 13.1%

The 15 pseudochromosomes and remaining scaffolds serve as our final assembly: *Ipomoea_batatas.fasta*.

Summary is as follows.

**Number of Sequences** = 28,464

**Length Statistics [bp]**

- *Total Length* = 837,061,592 (837Mb)
- *Largest* = 52,844,149
- *Smallest* = 392
- *Mean* = 29,407
- *Median* = 2,726
- *StdDev* = 979,310

**Base Composition**
\[A = 28.5\%\]
\[C = 15.5\%\]
\[G = 15.4\%\]
\[T = 28.5\%\]
\[G+C = 30.9\%\]
\[A+T = 57.0\%\]
\[N = 12.1\%\]

3.4 Remove redundancy in preliminary assembly contigs

All the contigs in preliminary assembly were mapped against the scaffolds of HI-assembly produced by SSPACE scaffolding in previous step.

```
bwa index lib_v4fine6.final.scaffolds.fasta -p v4
bwa mem v4 SP3_contigs.fa -t 32 >c2v4.sam &
```

Based on the Cigar information in c2v4.sam file, all contigs with more than 85\% covered by the scaffold of HI-assembly, were summarized in c2v4.sam.rm_cand (M length/Cigar length >= 85\%).

```
sort c2v4.sam.rm_cand >c2v4.sam.rm.sort
length.pl SP3_contigs.fa | sort >SP3_contigs.sort
join -j 1 SP3_contigs.sort c2v4.sam.rm.sort -v 1|cut -f1 -d " " >v4.contigs.list
get_contigs.pl v4.contigs.list SP3_contigs.fa >v4.contigs.fasta
```

Summary of final output file: v4.contigs.fasta.

*Total number of contigs* = 41487

*Sum (bp)* = 13861310

*Max contig size* = 8691

*Min scaffold size* = 100

*N50* = 480

*Contigs larger than 1000 bp*: 2094 contigs containing 3511745 bp
3.5 BUSCO assessment of assembly quality

Firstly, we performed a BUSCO testing run in Arabidopsis considering the Bate-version of BUSCO_plants.py.

```python
python3 BUSCO_plants.py -g at.fa -l plantae/ -c 4 -a at -m all -f -sp arabidopsis &
```

The output:

```bash
#Summarized BUSCO benchmarking for file: at.fa
#BUSCO was run in mode: genome
Summarized benchmarks in BUSCO notation:
   C:98%[D:26%],F:0.4%,M:1.5%,n:956
Representing:
   937 Complete Single-Copy BUSCOs
   256 Complete Duplicated BUSCOs
   4 Fragmented BUSCOs
   15 Missing BUSCOs
   956 Total BUSCO groups searched
```

The runs for sweet potato

Using a HMM model from tomato:

```python
python3 BUSCO_plants.py -g ../v4.scaffolds.fasta -l plantae/ -c 4 -a v4 -m all -f -sp tomato --long &
```

The output:

```bash
#Summarized BUSCO benchmarking for file: ../v4.scaffolds.fasta
#BUSCO was run in mode: genome
Summarized benchmarks in BUSCO notation:
   C:88%[D:58%],F:5.4%,M:6.2%,n:956
Representing:
   844 Complete Single-Copy BUSCOs
   559 Complete Duplicated BUSCOs
   52 Fragmented BUSCOs
   60 Missing BUSCOs
   956 Total BUSCO groups searched
```

Fast mode:

```python
python3 BUSCO_plants.py -g ../v4.scaffolds.fasta -l plantae/ -c 4 -a v4 -m all -f -sp tomato
```

The output:

```bash
Summarized benchmarks in BUSCO notation:
   C:87%[D:57%],F:5.2%,M:7.2%,n:956
```
Representing:

- 837 Complete Single-Copy BUSCOs
- 548 Complete Duplicated BUSCOs
- 50 Fragmented BUSCOs
- 69 Missing BUSCOs
- 956 Total BUSCO groups searched

Using a default HMM model:

```bash
python3 BUSCO_plants.py -g ../v4.scaffolds.fasta -l plantae/ -c 4 -a v4 -m all -f -sp v4 -long &
```

The output:

Summarized benchmarks in BUSCO notation:

- C:86%, D:50%, F:6.2%, M:7.1%, n:956

Representing:

- 828 Complete Single-Copy BUSCOs
- 483 Complete Duplicated BUSCOs
- 60 Fragmented BUSCOs
- 68 Missing BUSCOs
- 956 Total BUSCO groups searched
4 Genome annotation and cDNA validation

4.1 Gene model identification based on transcriptome

Six transcriptome data sets were mapped to the genome by HISAT2[^2] (Version 2.0.4) and HISAT (Version 0.1.5-beta) according to the available data format.

```
hisat2-build Ipomoea_batatas.fasta Ipomoea_batatas &
hisat2 -p 10 -x ../Ipomoea_batatas -U 23196105L.fq.gz,23196105P.fq.gz,23196105S.fq.gz,23196105R.fq.gz,23204017L.fq.gz,23204017P.fq.gz,23204017S.fq.gz,23204017R.fq.gz |samtools view -@ 10 -Su - | samtools sort -@ 10 | samtools rmdup - ../Xu22.bam &
hisat2 -p 10 -x ../Ipomoea_batatas -U RFR.fq.gz,WFR.fq.gz,RDR.fq.gz,WDR.fq.gz |samtools view -@ 10 -Su - | samtools sort -@ 10 | samtools rmdup - ../Ji26.bam &
hisat2 -p 10 -x ../Ipomoea_batatas -U S_PB1.fastq,S_PL1.fastq | samtools view -@ 10 -Su - | samtools sort -@ 10 - | samtools rmdup - ../Gjet.bam &
```

```
hisat-build Ipomoea_batatas.fasta l.batatas4sra &
hisat -p 10 -x ../l.batatas4sra --sra-acc ./SRR331231YL.sra,./SRR331743ML.sra,./SRR331744Stem.sra,./SRR33191745TR.sra,./SRR331746TR.sra,./SRR331824ETR.sra,./SRR331863HTR.sra,./SRR331947.sra |samtools view -@ 10 -Su - | samtools sort -@ 10 - | samtools rmdup- ../Ju18.bam &
hisat -p 10 -x ../l.batatas4sra --sra-acc ./SRR329935.sra |samtools view -@ 10 -Su - | samtools sort -@ 10 - | samtools rmdup- ../Js06.bam &
hisat -p 10 -x ../l.batatas4sra --sra-acc ./SRR063318.sra |samtools view -@ 10 -Su - | samtools sort -@ 10 - | samtools rmdup- ../Gs87.bam &
```

Extracting coding regions using StringTie (Version 1.2.2).

```
stringtie Xu22.bam -l Xu22 -f 0.18 -o Xu22.6.18.gtf -j 6 -c 6 -p 10 &
stringtieGjet.bam -l Gjet -f 0.18 -o Gjet.6.18.gtf -j 6 -c 6 -p 10 &
stringtie Ji26.bam -l Ji26 -f 0.18 -o Ji26.6.18.gtf -j 6 -c 6 -p 10 &
stringtie Js06.bam -l Js06 -f 0.18 -o Js06.6.18.gtf -j 6 -c 6 -p 10 &
stringtie Xu18.bam -l Xu18 -f 0.18 -o Xu18.6.18.gtf -j 6 -c 6 -p 10 &
stringtie Gs87.bam -l Gs87 -f 0.18 -o Gs87.6.18.gtf -j 6 -c 6 -p 10 &
```

A consensus transcriptome was generated by TACO ([https://tacorna.github.io/](https://tacorna.github.io/)), which is designed for multi-sample RNA-seq data.

```
ls ????.6.18.gtf > 6gtf_files
nohup taco_run -p 7 -o taco7 6gtf_files &
bedtools getfasta -fi ../Ipomoea_batatas.fasta -bed assembly.bed -name -split -s -fo Ipomoea_batatas.transcript.fa &
```

Dealing with 50 transcripts that contain hanging reads at the end of scaffolds.

```bash
cut -f 2 skipping -d " "|sed -e 's/(//1'- e 's/)/1'- e 's/:.*/\./
*/1' |grep -f assembly.bed |paste <(cut -f 1 skipping|cut -f 4 -d e) -|awk '{print $2"\t"$3"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12"\t"$13'} > fix.bed

cut -f 2 skipping -d " "|sed -e 's/(//1'- e 's/)//1' -e 's/:.*/\.
*"/1' |grep -f assembly.bed |paste <(cut -f 1 skipping|cut -f 4 -d e) -|awk '{print $4"\t"$0}' |cut -f 1,13|awk '{print $1","$2}' |awk -F , '{ print $0$(NF - 1)-1$1"," }'|awk -F, '{$1="";$(NF - 2)="";print $0}' |sed -e 's/ /,/g' -e 's,,/1'|cut -f 2 -d , |paste fix.bed |awk '{print $2"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7"\t"$8"\t"$9"\t"$10"\t"$11"\t"$12"\t"$13'} > realfix.bed

cat fine.bed realfix.bed > clean.bed
```

4.2 Gene model validation by Sanger cDNA sequencing

To validate the transcript sequences extracted by transcriptome data. We obtained additional 10063 cDNA Sanger sequences (EST) and mapped back to the genome.

```bash
blat Ipomoea_batatas.fasta ../../xu22cdna/sp.fas.screen -out=blast8 clib2 ref.blat8 &
```

There are 9425 ESTs located on current genome and have corresponding predicted gene models.

Haplotypes impacts on exons,

```bash
grep -v # Ipomoea_batatas_Version1.vcf |awk '{if(length($4)!=length($5)){print}}' > complexVar.vcf
cut -f 1,2,4,5 complexVar.vcf |sort > complexVar
grep exon Ipomoea_batatas_Version1.Tair.gtf|cut -f 1,4,5 |sort > exons
join -j 1 exons complexVar|awk '{if ($3==$4&&$4==$2){print}}' > putativeFramshift &
sed 's' ./g' putativeFramshift|awk '{if((length($5)-length($6))%3){print}}' > framshift
sort -k1 -k4g framshift |uniq > framshift.su
```

Using a python code we found 172,830 frame shifted events due to variants in haplotypes.

4.3 Gene annotation using publicly available protein database

Gene annotation based on NCBI protein database via lastal (http://last.cbrc.jp/) ungapped search. All annotated protein information in *Nicotiana sylvestris*, *Nicotiana tomentosiformis*, *Solanum indicum*, *Solanum tuberosum*, *Solanum lycopersicum*, *Nicotiana tabacum* and *Sesamum indicum* genomes were downloaded from FTP.

```bash
cat *.faa -project/sweet/mapping/proteins.faa
lastdb -p -cR01 protdb proteins.faa &
```
There are 89% transcripts (70166/78781) have hits in this protein database from taxonomy close plants.

We searched the homologous proteins of rest 11% transcripts (8615/78781) in Uniprot database. After these procedure, there are still 9% transcripts (7461/78781) have no available homologous protein information. To improve the gene annotation accuracy, all the transcripts with lastal search Score smaller than 300 were searched again in Uniprot database.

4.4 Putative gene clusters identification

There were 57 genes in various gene clusters of different species have been collected to identify putative gene clusters in current assembly. Firstly, a maximum likelihood phylogenetic tree of clustering genes was established in MEGA6 for windows. The relative divergence time estimation employed the RelTime method[5] in MEGA. Four main groups were found divergent at the similar time point. Then the similar strategy as gene annotation in previous step was applied on gene clusters identification.
awk '{if($3=="transcript"){print $1}}' v4Tai6clean-gc.gtf | sort | uniq -c | sort -gr > v4gcScf.list

All scaffolds with more than three hits were manually checked in IGV_2.3.32 to reduce false positive gene cluster number. Finally, there were 76 putative gene clusters were found and summarized in Supplementary Table 2. One may easily notice that most of identified cluster genes of *I. batatas* are belong to group II.

### 4.5 Repeat sequence identification based on repeat database

*De novo* repeat sequence identification was conducted using RepeatModeler (Version 1.0.8)\(^8\). RepeatMasker (Version open-4.0.5) was employed to classify and mask the repeats in the present genome.

RepeatModeler -pa 24 -engine ncbi -database ipoBat3
RepeatMasker -pa 24 -lib Libraries/ipoBat3_consensi.fa.classified -e ncbi ipoBat3.fasta

The repeat sequence classifications are summarized as follows. A brief summary was presented as Table 3. All the detail information and repeat sequences can be found in our Genome Browser.

<table>
<thead>
<tr>
<th>Type of elements</th>
<th>Number of elements</th>
<th>Length occupied</th>
<th>Percentage of genome*</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA/MULE-MuDR</td>
<td>54,322</td>
<td>10,471,659</td>
<td>1.250</td>
</tr>
<tr>
<td>DNA/CMC-EnSpm</td>
<td>42,257</td>
<td>10,306,018</td>
<td>1.230</td>
</tr>
<tr>
<td>DNA/hAT-Ac</td>
<td>39,478</td>
<td>7,920,540</td>
<td>0.945</td>
</tr>
<tr>
<td>DNA/Unclassified</td>
<td>41,831</td>
<td>7,020,622</td>
<td>0.838</td>
</tr>
<tr>
<td>DNA/PIF-Harbinger</td>
<td>43,575</td>
<td>6,640,535</td>
<td>0.792</td>
</tr>
<tr>
<td>DNA/hAT-Tag1</td>
<td>14,175</td>
<td>3,304,599</td>
<td>0.394</td>
</tr>
<tr>
<td>DNA/Mule-MuDR</td>
<td>5,995</td>
<td>2,731,660</td>
<td>0.326</td>
</tr>
<tr>
<td>DNA/hAT-Tip100</td>
<td>11,867</td>
<td>2,041,655</td>
<td>0.244</td>
</tr>
<tr>
<td>DNA/Ginger</td>
<td>921</td>
<td>165,282</td>
<td>0.020</td>
</tr>
<tr>
<td>DNA/IS3EU</td>
<td>1,080</td>
<td>161,489</td>
<td>0.019</td>
</tr>
<tr>
<td>DNA/TcMar-Tc2</td>
<td>759</td>
<td>99,076</td>
<td>0.012</td>
</tr>
<tr>
<td>LINE/L1</td>
<td>29,348</td>
<td>18,532,568</td>
<td>2.212</td>
</tr>
<tr>
<td>Repeat Class</td>
<td>Number</td>
<td>Estimated Size</td>
<td>Repeat Modeler %</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>LINE/CR1</td>
<td>5,481</td>
<td>1,091,341</td>
<td>0.130</td>
</tr>
<tr>
<td>LINE/L2</td>
<td>1,703</td>
<td>361,185</td>
<td>0.043</td>
</tr>
<tr>
<td>LINE/CRE</td>
<td>665</td>
<td>232,100</td>
<td>0.028</td>
</tr>
<tr>
<td>LINE/R1</td>
<td>828</td>
<td>221,489</td>
<td>0.026</td>
</tr>
<tr>
<td>LINE/RTE-BovB</td>
<td>1,562</td>
<td>180,020</td>
<td>0.021</td>
</tr>
<tr>
<td>LINE/RTE-X</td>
<td>368</td>
<td>136,783</td>
<td>0.016</td>
</tr>
<tr>
<td>LINE/L1-Tx1</td>
<td>191</td>
<td>58,779</td>
<td>0.007</td>
</tr>
<tr>
<td>Low_complexity</td>
<td>45,912</td>
<td>2,324,352</td>
<td>0.277</td>
</tr>
<tr>
<td>LTR/Copia</td>
<td>114,589</td>
<td>46,880,934</td>
<td>5.595</td>
</tr>
<tr>
<td>LTR/Gypsy</td>
<td>90,015</td>
<td>41,064,422</td>
<td>4.901</td>
</tr>
<tr>
<td>LTR/Unclassified</td>
<td>4,496</td>
<td>2,778,922</td>
<td>0.332</td>
</tr>
<tr>
<td>LTR/Caulimovirus</td>
<td>738</td>
<td>700,712</td>
<td>0.084</td>
</tr>
<tr>
<td>LTR/Cassandra</td>
<td>3,421</td>
<td>569,290</td>
<td>0.068</td>
</tr>
<tr>
<td>LTR/ERV1</td>
<td>180</td>
<td>72,223</td>
<td>0.009</td>
</tr>
<tr>
<td>RC/Helitron</td>
<td>12,067</td>
<td>5,610,834</td>
<td>0.670</td>
</tr>
<tr>
<td>rRNA</td>
<td>322</td>
<td>97,450</td>
<td>0.012</td>
</tr>
<tr>
<td>Satellite</td>
<td>456</td>
<td>33,443</td>
<td>0.004</td>
</tr>
<tr>
<td>Simple_repeat</td>
<td>324,390</td>
<td>14,578,880</td>
<td>1.740</td>
</tr>
<tr>
<td>SINE/Unclassified</td>
<td>2,481</td>
<td>380,194</td>
<td>0.045</td>
</tr>
<tr>
<td>SINE/ID</td>
<td>619</td>
<td>104,407</td>
<td>0.012</td>
</tr>
<tr>
<td>SINE/tRNA</td>
<td>698</td>
<td>95,214</td>
<td>0.011</td>
</tr>
<tr>
<td>SINE/tRNA-RTE</td>
<td>370</td>
<td>38,908</td>
<td>0.005</td>
</tr>
<tr>
<td>snRNA</td>
<td>219</td>
<td>29,584</td>
<td>0.004</td>
</tr>
<tr>
<td>Unknown</td>
<td>985,978</td>
<td>195,232,724</td>
<td>23.299</td>
</tr>
<tr>
<td>Total</td>
<td>1,883,357</td>
<td>382,269,893</td>
<td>45.619</td>
</tr>
</tbody>
</table>

* Scaffolds and Unplaced Contigs in HI-assembly were taken as input sequences of RepeatModeler.
5 Updating and validation of haplotypes

5.1 Haplotype updating

Taken the scaffolds of haplotype-aided assembly, by reusing same pipeline and algorithm, haplotypes were updated.

```
bwa index ../potato/v4.scaffolds.fasta -p v4
bwa mem v4 MP_R1.fastq.gz MP_R2.fastq.gz -t 48 >v4mp.sam &
bwa mem v4 AI_500_GTGAAA_L005_R1_001.fastq.gz AI_500_GTGAAA_L005_R2_001.fastq.gz -t 24 >v4a500.sam &
bwa mem v4 mpimg_L3956-2_AI500-2_S1_L001_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L001_R2_001.fastq.gz -t 28 >v4l001.sam &
bwa mem v4 mpimg_L3956-2_AI500-2_S1_L002_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L002_R2_001.fastq.gz -t 28 >v4l002.sam &
bwa mem v4 mpimg_L3956-2_AI500-2_S1_L003_R1_001.fastq.gz mpimg_L3956-2_AI500-2_S1_L003_R2_001.fastq.gz -t 28 >v4l003.sam &
bwa mem v4 AI_1000bp_2_ATTCCT_L003_R1_001.fastq.gz AI_1000bp_2_ATTCCT_L003_R2_001.fastq.gz -t 24 >v4a1kb.sam &

samtools view -@ 48 -b -o v4mp.bam -S v4mp.sam &
samtools view -@ 24 -b -o v4a1kb.bam -S v4a1kb.sam &
samtools view -@ 24 -b -o v4a500.bam -S v4a500.sam &
samtools view -@ 24 -b -o v4l001.bam -S v4l001.sam &
samtools view -@ 24 -b -o v4l002.bam -S v4l002.sam &
samtools view -@ 24 -b -o v4l003.bam -S v4l003.sam &
samtools view -@ 24 -b -o v4l004.bam -S v4l004.sam &

samtools sort -@ 50 v4mp.bam v4mp.sorted &
samtools sort -@ 24 v4a1kb.bam v4a1kb.sorted &
samtools sort -@ 48 v4a500.bam v4a500.sorted &
samtools sort -@ 24 v4l001.bam v4l001.sorted &
samtools sort -@ 24 v4l002.bam v4l002.sorted &
samtools sort -@ 24 v4l003.bam v4l003.sorted &
samtools sort -@ 24 v4l004.bam v4l004.sorted &

samtoolsrmdup v4mp.sorted.bam v4mp.rmdup.bam &
samtoolsrmdup v4a1kb.sorted.bam v4a1kb.rmdup.bam &
samtoolsrmdup v4a500.sorted.bam v4a500.rmdup.bam &
samtoolsrmdup v4l001.sorted.bam v4l001.rmdup.bam &
samtoolsrmdup v4l002.sorted.bam v4l002.rmdup.bam &
samtoolsrmdup v4l003.sorted.bam v4l003.rmdup.bam &
samtoolsrmdup v4l004.sorted.bam v4l004.rmdup.bam &

samtools merge -@ 54 v4.rmdup.bam v4*.rmdup.bam&
samtools view -@ 40 -h -o v4.rmdup.sam v4.rmdup.bam &
samtools faidx v4.scaffolds.fasta
fasta_generate_regions.py v4.scaffolds.fasta.fai 100000 >v4.100k
freebayes -parallel v4.100k 48 -f v4.scaffolds.fasta -p 6 -F 0.08 v4.rmdup.bam >v4.vcf &

After indexing and reads converting as usual, here are settings for generating HAP, FASTA and SAM files for all haplotypes.
-refFile path_to/v4.scaffolds.fasta
-bamFile . path_to /v4.rmdup.bam
-vcfFile . path_to /v4.vcf
-scfLenFile . path_to /v4.length
-noProcessor 1500
-outputFolderBase . path_to /result
for i in $(seq 0 1 1499); do mxqsub --group-name=samFix --stdout ./$i -t 300 python2.7 ./ranbow.py -processorIndex $i -par ./input.params --mode hap; done

Result summary.

exten.hap.fasta:4691130 (1973261074bp)
bridge.hap.fasta:3994080 (1862372729bp)>>ranbow.single.hap.fasta: 3868860 (1928359400bp)
pair.hap.fasta:2603102 (2814504087bp) >>ranbow.pair.hap.fasta: 2406913 (8130577820bp)

5.2 Haplotype validation

To validate the haplotyping accuracy, Roche 454 trimmed reads were mapped against the genome

bwa mem v4 454TrimmedReads.fastq -t 48 >v4.454.sam &
samtools view -@ 48 -b -o v4.454.bam -S v4.454.sam &
samtools sort -@ 48 v4.454.bam v4.454.sorted &

Settings for comparing v4.454.bam and HAP files of phased haplotypes
-454File path_to /v4.454.bam

python2.7 rainbow.py phylo -par input.params --mode index

for i in $(seq 0 1 1499)
do
python2.7 rainbow.py phylo -par input.params --mode run --processorIndex $i
done
python2.7 rainbow.py phylo -par input.params –mode run
Validation results were summarized in Supplementary Figure 7.

5.3 Variant updating in phased regions
We update variants according to assembled haplotypes. On the other words, we used haplotyping to detect the variant caller possible errors. In current VCF file, each SNP is obtained according to the reads mapped to a locus. The reference allele and the alternative ones are inferred just based on the specific locus. The contribution of our method to variant calling is that our method not only looks at a specific SNP, but also integrates the information of the neighbor alleles as well. This information is revealed from the consideration of connection between neighbor SNPs and the current one. For instance, two consecutive SNPs with 2 and 4 alleles, which are called by variant callers with just consideration of each locus separately, could give raise to 8 combinations. Just 6 out of these 8 combinations are correct. By haplotyping we take into account and evaluate the supports for each combination. This means that two combinations with lowest support are going to be deleted. Therefore, if one of the alleles in the second column (the one with 4 alleles) would not be supported by reads, we filter it. Then there is going to be 6 combinations which are matched by our hexaploid genome.

There are three possibilities for one phased SNP. (1) all 6 haplotypes covered the SNP, (2) some of the haplotypes covered the SNP, and (3) the SNP located outside of haplotype blocks. There is no sufficient information to update alleles in the second and third categories. Our method distinguishes the first group and categorizes them in the following subgroups: (a) all alleles are the same and (b) alleles are different. Then, the first group is put aside as an error in variant calling. This means that there is no sufficient support to consider this position as a variant position. We remove the variant calling result of this position from original VCF file. For the second group, our method ranks alleles firstly based on the number of supporting haplotypes. Then the first rank is picked as the reference allele and the rests are considered as the alternative ones.

Taken “bridge.hap.sam” and “v4.vcf” as the phased haplotypes and original VCF file, respectively, the variant positions in phased regions were updated according to the ranking method we proposed and final VCF file was uploaded as Ipomoea_batatas_Version1.vcf
6 Phylogenetic analysis of homologous chromosomal regions

ranbow phylo is employed to extract and tune the alignments from all phased regions in “ranbow.single.hap.bam”. As well as the other modes of functions, ranbow phylo can be executed in parallel mode. Therefore the files need to be indexed first

```
python2.7 ranbow.py phylo -mode index -par hap.params
```

There were 644360 phased regions in “ranbow.bridge.hap.bam”. MEGA-CC[6] was employed to compute the phylogenetic trees on computer farm with the following command.

```
python2.7 ranbow.py phylo -mode index -par hap.params -processorIndex i
```

then then collect mode is developed in order to report the final statistics for different phylogenetic tree topologies.

```
python2.7 ranbow.py phylo -mode collect -par hap.params
```

more over the tree modes calculates the mutation rate in the trees and reports the branch lengths

```
python2.7 ranbow.py phylo -mode tree -par hap.params
```

The results are summarized as follows.

<table>
<thead>
<tr>
<th>Tree</th>
<th>Mean (haplotype length)</th>
<th>Mean (Sequence length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(((((()))))</td>
<td>9</td>
<td>343</td>
</tr>
<tr>
<td>(((()))))</td>
<td>10</td>
<td>341</td>
</tr>
<tr>
<td>(())())()</td>
<td>6</td>
<td>342</td>
</tr>
<tr>
<td>(((()))))</td>
<td>6</td>
<td>347</td>
</tr>
<tr>
<td>(())())()</td>
<td>9</td>
<td>330</td>
</tr>
<tr>
<td>(((())))</td>
<td>9</td>
<td>340</td>
</tr>
<tr>
<td>(((((()))))</td>
<td>12</td>
<td>366</td>
</tr>
<tr>
<td>(())())()</td>
<td>6</td>
<td>321</td>
</tr>
<tr>
<td>(())())()</td>
<td>8</td>
<td>326</td>
</tr>
</tbody>
</table>
All topology structure distributed as follows.

<table>
<thead>
<tr>
<th># trees</th>
<th>%</th>
<th>Topology structure</th>
<th>Binary proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>48547</td>
<td>7.2928434082</td>
<td>(((((XX)X)X)X)X)X</td>
<td>1:5(1:4(1:3(1:2)))</td>
</tr>
<tr>
<td>38582</td>
<td>5.7958778993</td>
<td>(((XX)(XX))X)X</td>
<td>1:5(1:4(2:2))</td>
</tr>
<tr>
<td>122594</td>
<td>18.4163562072</td>
<td>(((XX)((XX)X))X)X</td>
<td>1:5(2:3(2:1))</td>
</tr>
<tr>
<td>141838</td>
<td>21.3072347074</td>
<td>(((XX)X)((XX)X))</td>
<td>3:3</td>
</tr>
<tr>
<td>130339</td>
<td>19.5798281457</td>
<td>(((XX)X)(XX))X</td>
<td>2:4(1:3(2:1))</td>
</tr>
<tr>
<td>183780</td>
<td>27.6078596323</td>
<td>(((XX)(XX))(XX))</td>
<td>2:4(2:2)</td>
</tr>
</tbody>
</table>

Focus on these 2:4(2:2) tree to draw a consensus tree with average branch lengths.

(((X:0.647499727936,X:0.647499727936):0.587670720427,(X:0.670078354554,X:0.670078354554):0.565092093808,(X:0.801373925346,X:0.801373925346):1.29386086625);

There were 644360 phased regions in “rainbow.single.hap.bam”. MEGA-CC[6] was employed to compute all the phylogenetic trees of such huge number of alignments on computer farm.

for i in {1..644360..1};do megacc -a infer_UPGMA_nucleotide.mao -d meg/tree$i.meg -o nwk/tree$i -n; done &

All topology structure distributed as follows.

<table>
<thead>
<tr>
<th># trees</th>
<th>%</th>
<th>Topology structure</th>
<th>Binary proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>48087</td>
<td>7.462522114</td>
<td>(((((XX)X)X)X)X)X</td>
<td>1:5(1:4(1:3(1:2)))</td>
</tr>
<tr>
<td>37238</td>
<td>5.778888234</td>
<td>(((XX)(XX))X)X</td>
<td>1:5(1:4(2:2))</td>
</tr>
<tr>
<td>118694</td>
<td>18.41987647</td>
<td>(((XX)((XX)X))X)X</td>
<td>1:5(2:3(2:1))</td>
</tr>
<tr>
<td>140245</td>
<td>21.76433161</td>
<td>(((XX)X)((XX)X))</td>
<td>3:3</td>
</tr>
<tr>
<td>136082</td>
<td>21.11828424</td>
<td>(((XX)X)(XX))X</td>
<td>2:4(1:3(2:1))</td>
</tr>
<tr>
<td>164014</td>
<td>25.45299358</td>
<td>(((XX)(XX))(XX))</td>
<td>2:4(2:2)</td>
</tr>
</tbody>
</table>

Focus on these 2:4(2:2) tree to draw a new consensus tree with average branch lengths.

(((X:0.667683856256,X:0.667683856256):0.622155730608,(X:0.69290121575,X:0.69290121575):0.596938371115):0.782279256649,(X:0.831069298962,X:0.831069298962):1.33390442279);

The SNP difference in B1 sub-genome or B2 sub-genome is 0.721598345337% and 0.601696135787%, respectively.

The final tree and behind evolutionary meanings were interpreted and presented in Figure 5.
References


