User-friendly genome assembly and gene annotation pipelines for vertebrates

Siwen Wu[1], Sisi Yuan[1], Zhengchang Su[1#]

[1]Department of Bioinformatics and Genomics, the University of North Carolina at Charlotte, Charlotte, NC 28223, USA.

[#]The author supervised this work and should be addressed to zcsu@uncc.edu (ZS).

Abstract

High-quality genome assembly and accurate gene annotation are critical to understand the biology and evolutionary history of a species. With the advance in sequencing technologies, assembling large vertebrate genomes using a combination of low-costing short reads and noisy long reads is in the reach of an individual lab. Although many tools have been developed, no single one is able to produce a chromosome-level assembly. Instead, a pipeline involving multiple steps using different tools is needed to achieve the goal. Nonetheless, it is difficult for a newcomer to choose an appropriate tool for each stage of pipeline and their combination for optimal performance. Moreover, most existing gene annotation tools cannot sufficiently and accurately identify both protein-coding genes and pseudogenes. Although gene annotation pipelines have been described such as those by NCBI and ENSEMBL, they are not available for individual labs due to the complexity of their use and required computational resources. To overcome these obstacles, here, we introduce user-friendly optimized genome assembly and gene annotation pipelines for vertebrates. Our genome assembly pipeline is able to produce chromosome-level assemblies with high quality using a combination of PacBio/Nanopore long reads, Illumina short reads and Hi-C short reads, and our gene annotation pipeline can accurately and sufficiently annotate both protein-coding genes and pseudogenes in an assembled genome using a combination of homology-based and RNA-seq data-based methods. Both pipelines show superior performances to existing ones in applications.

Introduction

Accurate genome assembly of an organism is the first step to fully understand the genetic bluebook of the organism. Subsequent annotation of all genes encoded or pseudogenized in the genome can further the understanding of the biology and evolutionary history of the organism. The rapid development of sequencing technologies has made genome assembly more and more accurate and achievable in individual labs. Since 1977, Sanger sequencing [1] had been the most commonly used technology to sequence genomes before the advent of next-generation sequencing (NGS) technology, and a few genomes including the first two human draft genomes were sequenced by Sanger sequencing technology with ~1,000 bp read length [2, 3]. However, due to its poor quality in the first 15~40 bases of the sequences and relatively high cost and low throughput, Sanger sequencing had been replaced by NGS [4] since 2005. However, NGS, dominated by Illumina technology, is limited by its short-read length (150~300 bp), making it difficult to assemble repetitive regions. In 2014, the single molecule real time (SMRT) sequencing platform was developed by Pacific Biosciences (PacBio) [5], enabling the sequencing length up to 10 kbp [5, 6], allowing high quality *de novo* assembly of large vertebrate genomes. In addition, Oxford Nanopore Technology (ONT) [7, 8] was able to produce reads up to 100 kbp in length [9], making *de novo* genome assembly more complete. However, both PacBio and ONT reads suffer high sequencing errors (10~15%), thus high quality NGS short reads are often used to correct these errors at different stages of assembly pipelines. Furthermore, with the introduction of Hi-C technology [10] that provides information about physical distances between genomic loci, genomes can be scaffolded into chromosomes. More recently, PacBio and ONT have developed technologies to produce highly accurate long reads

[11] and ultralong reads [12], respectively, enabling telomere-to-telomere and phased chromosome assemblies [13]. However, before the costs of such technologies can be largely reduced, combining less accurate long reads with short reads and Hi-C data remains the choice to sequence large genomes in individual labs.

Assembling a genome using a combination of short reads, noisy long reads and Hi-C reads involves multiple steps, including correction of long noisy long reads, assembling error-corrected long reads in contiguous sequences (contigs), and ordering and orienting contigs in scaffolds/chromosomes using Hi-C reads. Additional error correction and polishing steps are also often added between these steps to improve the assembly quality. Many tools have been developed to perform one or more of these steps. For example, Wtdbg2 [14], Canu [15] and FALCON [16] have been developed to assemble contigs using a combination of long and short sequencing reads. SALSA [17, 18], LACHESIS [19] and 3d-DNA [20] and YaHS [21] have also been developed to scaffold the assembled contigs using Hi-C or optical mapping data. Although conceptually simple, it can be challenging for newcomers to choose, combine and optimize tools in each of these steps [22, 23]. Moreover, although sophisticated assembling pipelines have been described, mostly at large consortium such as Vertebrate Genome Consortium [24] and Zoonomia [25], they are usually not available to individual research groups, and also are difficult to follow by most individual research groups due to limited resources available. To fill these gaps, we developed a user-friendly optimized genome assembly pipeline for vertebrates. By using the PacBio/Nanopore long reads, Illumina short reads and Hi-C paired-end reads, our pipeline could assemble the genome at chromosome-level with high-quality. We have

demonstrated the usefulness of the pipeline by assembling the genomes of four indigenous chickens [26] and a white eared pheasant individual [27].

At the same time, like assembling a genome, annotation of protein-coding genes and pseudogenes in an assembled genome also is a complex process. Three approaches and their combinations have been used to annotate protein-coding in a genome, including *de novo* prediction, transcripts-based prediction, and homology-based prediction. Numerous tools of each approach have been developed. Popular tools for *de novo* gene prediction including Augustus [28], GlimmerHMM [29], GeneID [30], Genscan [31], SNAP [32] and GeneMark-ES [33]. Popular homology-based methods include Genomix [34], AUGUSTUS [35], GeMoMa [36], Scipio and WebScipio [37], and OMGene [35]. Popular RNA-seq data-based methods include Braker2 [38], CAARs [39] and FINDER [40]. However, none of these tools alone can sufficiently annotate both protein-coding genes and pseudogenes. Although NCBI (https://www.ncbi.nlm.nih.gov/refseq/annotation_euk/process/#references) and ENSEMBL (https://useast.ensembl.org/info/genome/genebuild/index.html) have developed annotation pipelines which performs well for predicting both protein-coding and pseudogenes, they are not available to individual research groups and difficult to follow. To fill these gaps, we also developed a protein-coding gene and pseudogene annotation pipeline for vertebrates. By using a combination of homology-based and RNA-seq data-based approaches, our pipeline could annotate protein-coding genes and at the same time identify pseudogenes accurately. We have demonstrated the usefulness of the pipeline by annotating genes of four indigenous chicken genomes [41].

Development of the protocol

Most existing genome assembly tools such as Wtdbg2 [14], Canu [15], and FALCON [16] can only achieve the contig-level assembly using a combination of PacBio/Nanopore long reads and Illumina short reads. To obtain the chromosome-level assembly, contigs need to be scaffolded using Hi-C short reads. The gaps introduced by scaffolding need to be filled using PacBio/Nanopore long reads, and errors introduced by gap filling need to be polished using PacBio/Nanopore long reads and Illumina short reads. Thus, we designed a genome assembly pipeline that could achieve all aforementioned tasks in a few steps in a streamlined fashion using well-regarded existing tools including Wtdbg2 [14] for assembling contigs, SALSA [17, 18] for scaffolding contigs, PBJelly [42] for gap filling, Racon [43] and NextPolish [44] for polishing. The pipeline achieved comparable or better assembling quality than state-of-the-art pipelines [26, 27].

For the gene annotation, we have designed a pipeline using a combination of homology-based and RNA-seq data-based methods. By using genes in a large number of closely related species as references as well as large volumes of RNA-seq data from various tissues of the target species or closely related ones, our pipeline could sufficiently identify protein-coding genes and new genes with no homologs, as well as pseudogenes that have open reading frame (ORF) shift or premature nonsense mutation compared with homologous genes. Our pipeline could predict almost all genes and pseudogenes predicted by the NCBI pipeline in the same genome, while predicting more genes and pseudogenes that are either missed or incorrectly predicted by the NCBI pipeline. Moreover, unlike the NCBI and ENSEMBL gene annotation pipelines, our pipeline is an open source and easy to be followed by individual research lab.

Experimental design

The flowchart of our genome assembly pipeline is shown in Figure 1. Firstly, we assemble gapless contigs using long reads using Wtdbg2 [14]. Secondly, we bridge the assembled contigs with gaps (Ns) into scaffolds using Hi-C reads using SALSA [17, 18]. Thirdly, we fill the gaps introduced during scaffolding using PacBio/Nanopore long reads using PBJelly [42]. Fourthly, we polish the scaffolds using PacBio/Nanopore long reads using Racon [43] to correct some sporadically occurring sequencing errors in long reads. Finally, since many sequencing errors remain in the assembled scaffolds, we correct them using Illumina short reads using NextPolish [44]. The final result is a high-quality assembly at chromosome-level with a few gaps.

To annotate the genome, we first need to prepare the required data (Figure 2). For the homology-based annotation, we need to compile sequences of CDSs isoforms of genes in a set of closely related species as reference genes. We map all the CDSs isoforms to the target genome using Splign [45]. For each reference gene whose CDSs could be mapped to the genome, we concatenate all the mapped CDSs and check whether the resulting sequence forms an open reading frame (ORF), i.e., the length of the sequence is an integer time of three and contains no stop codon in the middle. Next, we evaluate each concatenated sequences using a procedure as shown in Figure 3. Specifically, if the concatenated sequence forms an ORF, we predict it to be an intact gene. If the CDSs of a reference gene can be mapped to multiple loci in the genome and form multiple ORFs, we select the one with the highest mapping identity. If the concatenated sequence does not form an ORF, i.e., it contains at least a premature stop codon, or an ORF shift mutation, we map the short reads used to assemble the genome to the sequence with no gaps and mismatch allowed using Bowtie2 [46-48]. If the sequence is

completely covered by the short reads, indicating that the pseudogenization (a premature stop codon, or an ORF shift mutation) is fully supported by the short reads, and we predict the sequence to be a pseudogene. Otherwise, the pseudogenization is not supported by the short reads, and we predict the sequence to be a partially supported gene, since the pseudogenization might be artificially caused by errors of the long reads that could not be corrected by the short reads.

For the RNA-seq data-based annotation, we need RNA-seq reads from various tissues of the target species (Figure 2). Firstly, we map all of the RNA-seq reads from various tissues of the target species to the rRNA database SILVA_138 [49] using Bowtie2 [46-48], and filtering out the mapped reads, thereby removing RNA-seq reads from rRNA genes (Figure 2). Secondly, we map the remaining reads to the genome using STAR [50] and assemble transcripts using the genome-guided method of Trinity [51] (Figure 2). Thirdly, we map the assembled transcripts to the genome using Splign [45], and filter out those that at least partially overlap non-coding RNAs (see below), protein-coding genes or pseudogenes predicted by the homology-based method (Figure 2). Finally, we evaluated each of the remaining transcripts using a procedure as shown in Figure 3. Specifically, for each of such transcripts, we predict its mapped genome regions to be a protein-coding gene if it contains an ORF with a length of at least 300 bp. If multiple ORFs are found in a transcript, we select the longest one. For each of the protein-coding genes predicted by the homology-based and RNA-based methods, we correct the CDS phase using GFF3toolkit [52]. Besides, for completeness, we also predict non-coding RNAs using infernal [53] with Rfam database [54] as the reference.

Limitations

To run our genome assembly pipeline, Illumina paired-end short reads, PacBio/Nanopore long reads and Hi-C paired-end reads with sufficient sequencing depths from the same individual organism are needed. Thus, users without all these types of sequencing data may not obtain a high-quality assembly. Although vertebrate genomes are diploid, our genome assembling pipeline is unable to phase the haploid genomes. Including HiFi data [11] and using a haploid-aware assembler such as HiCanu [55], hifiasm [56] and Verkko [57] in the pipeline in the future would overcome this limitation. To run our gene annotation pipeline, CDS isoforms of genes in a set of closely related species and RNA-seq reads from as many as possible tissues with sufficient sequencing depths of the target species or closely related species are required. Therefore, users need to collect these data, otherwise they may not get the accurate gene annotation result.

Materials

Equipment

1) For the genome assembly pipeline:

- Raw data (see Equipment setup)
- Wtdbg2 [14] software (https://github.com/ruanjue/wtdbg2)
- SALSA [17, 18] software (https://github.com/marbl/SALSA)
- PBJelly [42] software (https://github.com/esrice/PBJelly)
- Racon [43] software (https://github.com/isovic/racon)
- NextPolish [44] software (https://github.com/Nextomics/NextPolish)

- SAMtools [58] software (https://github.com/samtools/)
- Minimap2 [59] software (https://github.com/lh3/minimap2)
- BWA [60] software (https://github.com/lh3/bwa)
- BEDTools [61] software (https://github.com/arq5x/bedtools2)
- Hardware (see Equipment setup)

2) For the gene annotation pipeline:

- Raw data and database (see Equipment setup)
- Splign [45] software (https://www.ncbi.nlm.nih.gov/sutils/splign/splign.cgi)
- Bowtie2 [46-48] software (https://github.com/BenLangmead/bowtie2)
- SAMtools [58] software (https://github.com/samtools/)
- BEDTools [61] software (https://github.com/arq5x/bedtools2)
- Trinity [51] software (https://github.com/trinityrnaseq/trinityrnaseq)
- STAR [50] software (https://github.com/alexdobin/STAR)
- GFF3toolkit [52] software (https://github.com/NAL-i5K/GFF3toolkit)
- Infernal [53] software (https://github.com/EddyRivasLab/infernal)
- Our annotation pipeline (https://github.com/zhengchangsulab/A-genome-assebmly-and-annotation-pipeline)
- Hardware (see Equipment setup)

Equipment setup

1) For the genome assembly pipeline:

- **Required raw data:**

Illumina paired-end sequencing data (>80X), PacBio/Nanopore long reads (>30X) and Hi-C paired-end reads (>100X) with sufficient sequencing depths from the same individual organism.

- **Wtdbg2 [14] software installation:**

  git clone https://github.com/ruanjue/wtdbg2

  cd wtdbg2 && make

  (Wtdbg2 only works on 64-bit Linux.)

- **SALSA [17, 18] software installation:**

  git clone https://github.com/marbl/SALSA

  make

  (Dependencies: Python 2.7, BOOST libraries and Networkx with lower than 1.2 version)

- **PBJelly [42] software installation:**

  git clone https://github.com/esrice/PBJelly

  Edit setup.sh and change $SWEETPATH to the full directory where you've placed the package.

  To automatically place the package into your environment, add source /setup.sh to your .bash_profile.

  Be sure to source your .bash_profile (or just setup.sh) before using Jelly.

  (Dependencies: Python 2.7)

- **Racon [43] software installation:**

  git clone --recursive https://github.com/lbcb-sci/racon.git racon

  cd racon

mkdir build

cd build

cmake -DCMAKE_BUILD_TYPE=Release ..

make

(Dependencies: gcc 4.8+/clang 3.4+ and cmake 3.2+)

- **NextPolish [44] software installation:**

  wget https://github.com/Nextomics/NextPolish/releases/download/v1.4.1/NextPolish.tgz

  pip install paralleltask

  tar -vxzf NextPolish.tgz && cd NextPolish && make

  (Dependencies: Python 2/3)

- **SAMtools [58] software installation:**

  git clone https://github.com/samtools/samtools

  cd samtools

  ./configure

  make

  make install

- **Minimap2 [59] software installation:**

  git clone https://github.com/lh3/minimap2

  cd minimap2 && make

- **BWA [60] software installation:**

  git clone https://github.com/lh3/bwa.git

cd bwa; make

- **BEDTools [61] software installation:**

  wget https://github.com/arq5x/bedtools2/releases/download/v2.30.0/bedtools-2.30.0.tar.gz

  tar -zxvf bedtools-2.30.0.tar.gz

  cd bedtools-2.30.0

  make

- **Hardware setup:**

  A cluster of Linux system of at least 350 GB memory with multiple threads available is needed to use the pipeline to assemble a genome of 1 Gbp size.

2) For the gene annotation pipeline:

- **Required raw data and database:**

  CDS isoforms in a set of closely related species, RNA-seq paired-end short reads from as many as possible tissues of the target species, Illumina paired-end sequencing reads with enough sequencing (>80X) depth of the same individual organism used for genome assembly, rRNA database, and Rfam database.

- **Splign [45] software installation:**

  wget https://ftp.ncbi.nlm.nih.gov/genomes/TOOLS/splign/linux-i64/splign.tar.gz

  tar zxvf splign.tar.gz

- **Bowtie2 [46-48] software installation:**

  conda install bowtie2

  (Dependency: anaconda3)

- **SAMtools [58] software installation:**

  git clone https://github.com/samtools/samtools

  cd samtools

  ./configure

  make

  make install

- **BEDTools [61] software installation:**

  wget https://github.com/arq5x/bedtools2/releases/download/v2.30.0/bedtools-2.30.0.tar.gz

  tar -zxvf bedtools-2.30.0.tar.gz

  cd bedtools-2.30.0

  make

- **Trinity [51] software installation:**

  wget https://github.com/trinityrnaseq/trinityrnaseq/releases/download/Trinity-v2.15.0/trinityrnaseq-v2.15.0.FULL.tar.gz

  tar zxvf trinityrnaseq-v2.15.0.FULL.tar.gz

  cd trinityrnaseq-v2.15.0.FULL

  make

- **STAR [50] software installation:**

  wget https://github.com/alexdobin/STAR/archive/2.7.10b.tar.gz

  tar -xzf 2.7.10b.tar.gz

  cd STAR-2.7.10b /source

make STAR

- **GFF3toolkit [52] software installation:**

pip install gff3tool

(Dependencies: Python 3.x and Perl)

- **Infernal [53] software installation:**

wget http://eddylab.org/software/infernal/infernal.tar.gz

tar zxf infernal.tar.gz

cd infernal-1.1.4

./configure

make

- **Our annotation pipeline installation:**

git clone https://github.com/zhengchangsulab/A-genome-assebmly-and-annotation-pipeline.git

cd A-genome-assebmly-and-annotation-pipeline

chmod -R 711 scripts

chmod -R 711 bin

cd bin

export PATH=$PATH:$PWD

cd ../scripts

export PATH=$PATH:$PWD

- **Hardware setup:**

A cluster of Linux system of at least 350 GB memory with multiple threads available is needed to use the pipeline to annotate a genome of 1 Gbp size.

Procedure

1) For the genome assembly pipeline:

- **Step 1:** Use Wtdbg2 [14] to generate contigs using PacBio/Nanopore long reads and polish the contigs using Illumina paired-end short reads. An example of Linux commands is shown below:

```
NANOPORE=nanopore long reads.fastq
SHORTREAD1=Illumina paired-end-1.fastq
SHORTREAD2=Illumina paired-end-2.fastq
PREFIX=F025
threads=32
wtdbg2 -x ont -g 1g -X 109 -e 10 -i $ NANOPORE -t $threads -fo $PREFIX
wtpoa-cns -t $threads -i $PREFIX.ctg.lay.gz -fo $PREFIX.ctg.fa
minimap2 -t $threads -ax map-ont $PREFIX.ctg.fa $NANOPORE | samtools view -Sb |
samtools sort -@ $threads-1 - > $PREFIX.ctg.crt.bam
samtools view -F 0x900 $PREFIX.ctg.crt.bam | wtpoa-cns -t $threads -d $PREFIX.ctg.fa -i
- -fo $PREFIX.ctg.2nd.fa
bwa index $PREFIX.ctg.2nd.fa
bwa mem -t $threads $PREFIX.ctg.2nd.fa $SHORTREAD1 $SHORTREAD2 | samtools sort
```

```
-@ $threads-1 -O SAM | wtpoa-cns -t $threads -x sam-sr -d $PREFIX.ctg.2nd.fa -i - -fo
$PREFIX.ctg.3rd.fa
```

(The genome size is 1 Gbp and the sequencing depth of the nanopore long reads is 109X in the example. The final assembled contigs are written into the file F025.ctg.3rd.fa, which will be used in Step 2.)

- **Step 2:** Use SALSA [17, 18] to bridge the contigs obtained in Step 1 into scaffolds using Hi-C paired-end reads. An example of Linux commands is shown below:

```
CONTIGFILE=F025.ctg.3rd.fa
HICFILE1=Hi-C pair-end-1.fastq
HICFILE2= Hi-C pair-end-2.fastq
PREFIX=F025arima
TEMPDIR=temp
threads=36
MAPQ_FILTER=10
SALSADIR=SALSA_DIR
FILTER=SALSA_DIR/mapping_pipeline/filter_five_end.pl
COMBINER= SALSA_DIR /mapping_pipeline/two_read_bam_combiner.pl
bwa index $CONTIGFILE
bwa mem -t $threads $CONTIGFILE $HICFILE1 | samtools view -Sb - > aln-
$PREFIX_1.bam
bwa mem -t $threads $CONTIGFILE $HICFILE2 | samtools view -Sb - > aln-
$PREFIX_2.bam
```

```
samtools view -h aln-$PREFIX_1.bam | perl $FILTER | samtools view -Sb - > flt-$PREFIX_1.bam
samtools view -h aln-$PREFIX_2.bam | perl $FILTER | samtools view -Sb - > flt-$PREFIX_2.bam
samtools faidx $CONTIGFILE
perl $COMBINER flt-$PREFIX_1.bam flt-$PREFIX_2.bam samtools $MAPQ_FILTER | samtools view -Sb -t $CONTIGFILE.fai - | samtools sort -@ $threads-1 -o aln-$PREFIX.bam -
bamToBed -i aln-$PREFIX.bam > aln-$PREFIX.bed
sort -k 4 --parallel=16 --temporary-directory=$TEMPDIR aln-$PREFIX.bed > aln-$PREFIX.srt.bed
python $SALSADIR/run_pipeline.py -a $CONTIGFILE -l $CONTIGFILE.fai -b aln-$PREFIX.srt.bed -e AAGCTT -o $PREFIX -m yes -i 4 -s 1000000000 -c 500
```

(The final scaffolds are written into file scaffolds_FINAL.fasta, which will be used in Step 3.)

- **Step 3:** Use PBjelly [42] to fill the gaps introduced in Step 2 using PacBio/Nanopore long reads. An example of Linux commands is shown below:

```
Prepare your Protocol.xml
Jelly.py setup Protocol.xml
Jelly.py mapping Protocol.xml
Jelly.py support Protocol.xml
Jelly.py extraction Protocol.xml
```

```
Jelly.py assembly Protocol.xml
Jelly.py output Protocol.xml
```

(The final assembly is written into file jelly.out.fasta, which will be used in Step 4.)

- **Step 4:** Use Racon [43] to polish the scaffolds obtained in Step 3 using PacBio/Nanopore long reads. An example of Linux commands is shown below:

```
GENNAME=jelly.out.fasta
NANOPORE=nanopore long reads.fastq
PREFIX=F025
threads=48
types=map-ont
minimap2 -x $types -t $threads $GENNAME $NANOPORE > $PREFIX.paf
racon -t $threads -u $NANOPORE $PREFIX.paf $GENNAME > racon.fasta
minimap2 -x $types -t $threads racon.fasta $NANOPORE > $PREFIX\2.paf
racon -t $threads -u $NANOPORE $PREFIX\2.paf racon.fasta > racon.2nd.fasta
minimap2 -x $types -t $threads racon.2nd.fasta $NANOPORE > $PREFIX\3.paf
racon -t $threads -u $NANOPORE $PREFIX\3.paf racon.2nd.fasta > racon.3rd.fasta
```

(There are three rounds of polish in the example, and the final results are written into file racon.3rd.fasta, which will be used to Step 5.)

- **Step 5:** Use Nextpolish [44] to further polish the scaffolds obtained in Step 4 using Illumina paired-end short reads. An example of Linux commands is shown below:

```
nextpolish=/Nextpolish_Dir/nextpolish1.py
input=racon.3rd.fasta
```

```
read1=Illumina paired-end-1.fastq
read2=Illumina paired-end-2.fastq
threads=48
round=2
for ((i=1; i<=$[62];i++)); do
bwa index $input
bwa mem -t $threads $input $read1 $read2|samtools view --threads 3 -F 0x4 -b -
|samtools fixmate -m --threads 3 - -|samtools sort -m 2g --threads 5 -|samtools
markdup --threads 5 -r - sgs.sort.bam
samtools index -@ $threads sgs.sort.bam
samtools faidx $input
python $nextpolish -g $input -t 1 -p $threads -s sgs.sort.bam > genome.polishtemp.fa
input=genome.polishtemp.fa
bwa index $input
bwa mem -t $threads $input $read1 $read2|samtools view --threads 3 -F 0x4 -b -
|samtools fixmate -m --threads 3 - -|samtools sort -m 2g --threads 5 -|samtools
markdup --threads 5 -r - sgs.sort.bam
samtools index -@ $threads sgs.sort.bam
samtools faidx $input
python $nextpolish -g $input -t 2 -p $threads -s sgs.sort.bam > genome.nextpolish.fa
input=genome.nextpolish.fa
done;
```

(There are two rounds of polish in the example, and the final assembly is written into file genome.nextpolish.fa)

2) For the gene annotation pipeline:

- **Step 1:** Data preparations

  1) For the homology-based method:

  - **Step H1:** Use Splign [45] to map the reference CDS isoforms to the target genome. An example of Linux commands is shown below:

```
reference_cds=reference_CDS.fa
genome=my_genome.fa
mkdir fasta_dir
cp $genome fasta_dir
cp $reference_cds fasta_dir
splign -mklds fasta_dir
cd fasta_dir
makeblastdb -dbtype nucl -parse_seqids -in reference_CDS.fa
makeblastdb -dbtype nucl -parse_seqids -in my_genome.fa
compart -qdb reference_CDS.fa -sdb my_genome.fa > cdna.compartments
cd ..
splign -ldsdir fasta_dir -comps ./fasta_dir/cdna.compartments > splign.output.ref
```

    (The results are written into the file splign.output.ref.)

  - **Step H2:** Use Bowtie2 [46-48] to map the Illumina paired-end sequencing reads to the target genome allowing no mismatch and gaps to get the region not

supported by the short reads on the target genome. An example of Linux commands is shown below:

```
genome=my_genome.fa
r1=Illumina paired-end-1.fastq
r2=Illumina paired-end-2.fastq
threads=48
bowtie2-build $genome chicken
bowtie2 -p $threads -x chicken -1 $r1 -2 $r2 --score-min L,0,0 | samtools view -Sb -@ $threads-1 | samtools sort -@ $threads-1 > out.bam
bedtools genomecov -ibam out.bam -bga > out.bed
awk '$4=="0"{print $0}' out.bed > notsupport.region
```

(The results are written into the file notsupport.region.)

2) For the RNA-based method:

- **Step R1:** Use Infernal [53] to predict non-coding RNAs against Rfam database. An example of Linux commands is shown below:

```
Rfam_path=Path of Rfam database
Genome=my_genome.fa
esl-seqstat $Genome
cmscan --cpu 48 --tblout result.tbl $Rfam_path/Rfam.cm $Genome > result_final.cmscan
```

(The results are written into the file result_final.cmscan.)

- **Step R2:** Use Bowtie2 [46-48] to map the RNA-seq short reads to the rRNA database to get the unaligned reads, and assemble the unaligned reads into transcripts using STAR [50] and Trinity [51] genome-guided method. An example of Linux commands is shown below:

```
rrna=rrna_database.fa
left=RNA-seq paired-end-1.fastq
right=RNA-seq paired-end-2.fastq
bowtie2-build $rrna rrna_data
bowtie2 -p 48 --very-sensitive-local -x rrna_data -1 $left -2 $right --un-conc-gz
paired_unaligned.fq.gz --un-gz unpaired_unaligned.fq.gz
genome=my_genome.fa
left=paired_unaligned.fq.1
right=paired_unaligned.fq.2
PREFIX=F025
threads=32
mkdir star
STAR --runThreadN $threads --runMode genomeGenerate --genomeDir ./star --
genomeFastaFiles $genome
STAR --genomeDir ./star --runThreadN $threads --readFilesIn $left $right --
outFileNamePrefix $PREFIX --outSAMtype BAM SortedByCoordinate --
outBAMsortingThreadN $threads --limitBAMsortRAM 214748364800
RNAbam=$PREFIX\Aligned.sortedByCoord.out.bam
```

```
Trinity --output Trinity_GG --genome_guided_bam $RNAbam --
genome_guided_max_intron 200000 --CPU $threads --max_memory 350G --
verbose
```

(The assembled transcripts are written into the file Trinity-GG.fasta.)

- **Step R3:** Use Splign [45] to map the transcripts obtained in step R2 to the target genome. An example of Linux commands is shown below:

```
genome=my_genome.fa
rna=Trinity-GG.fasta
mkdir fasta_dir
cp $genome fasta_dir
cp $rna fasta_dir
splign -mklds fasta_dir
cd fasta_dir
makeblastdb -dbtype nucl -parse_seqids -in Trinity-GG.fasta
makeblastdb -dbtype nucl -parse_seqids -in my_genome.fa
compart -qdb transcripts.fa -sdb my_genome.fa > rna.compartments
cd ..
splign -ldsdir fasta_dir -comps ./fasta_dir/rna.compartments -type est >
splign.output.rna
```

(The results are written into the file splign.output.rna.)

- **Step 2:** Annotate protein-coding genes and pseudogenes for the target genome

- **Step A1:** Use our gene finding procedure (https://github.com/zhengchangsulab/A-genome-assebmly-and-annotation-pipeline) to get the primary gene annotation results. The inputs of the pipeline are the results from Step H1~H2 and Step R1~R3 of Data preparations (Step 1). An example of Linux commands is shown below:

  prepare your parameter.txt file

  annotation.pip

- **Step A2:** Use GFF3toolkit [52] to correct CDS phase of the protein-coding genes from the result of step A1. An example of Linux commands is shown below:

  genome=my_genome.fa

  gff3_QC -g final.use.truegene.gff3 -f $genome -o error.txt -s statistic.txt

  gff3_fix -qc_r error.txt -g final.use.truegene.gff3 -og final.right.truegene.gff3

  (The final annotation of protein coding genes and pseudogenes are written into the file final.right.truegene.gff3 and final.right.pseudogene.gff3, respectively.)

Timing

The time of each step of the pipelines depends on the genome size of the target species, sequencing depth of the raw data and the available memory and number of CUPs of the computer. Here we used the Daweishan chicken with about 1 Gbp genome size that we assembled [26] and annotated before [41] as an example to show the timeline of our pipeline.

Here is the timeline for the genome assembly pipeline:

- **Step 1:** ~20 hours with 350 GB memory and 32 CPUs.

- **Step 2:** ~18 hours with 350 GB memory and 32 CPUs.
- **Step 3:** ~20 hours with 350 GB memory and 32 CPUs.
- **Step 4:** ~5 hours with 350 GB memory and 32 CPUs.
- **Step 5:** ~ 20 hours with 350 GB memory and 32 CPUs.

Here is the timeline for the gene annotation pipeline:

- **Step 1:** Data preparations

  1) For the homology-based method:

  - **Step H1:** ~four days with 100 GB memory and 1 CPU.
  - **Step H2:** ~20 hours with 300 GB memory and 48 CPUs.

  2) For the RNA-based method:

  - **Step R1:** ~20 hours with 300 GB memory and 48 CPUs.
  - **Step R2:** ~four days with 350 GB memory and 36 CPUs.
  - **Step R3:** ~20 hours with 350 GB memory and 1 CPU.
- **Step 2:** Annotate protein-coding genes and pseudogenes for the target genome
  - **Step A1:** ~15 minutes with 1 GB memory and 1 CPU.
  - **Step A2:** ~15 minutes with 1 GB memory and 1 CPU.

Anticipated results

We demonstrated the usage of our genome assembly pipeline by assembling the genome of a Daweishan chicken individual using Illumina paired-end short reads of 150 bp length with 143X sequencing depth, Nanopore long reads with an average read length of 14.5 kbp and 109X sequencing depth, and Hi-C paired-end reads of 150 bp length with 108X sequencing depth. As

549 shown in Table 1, after Step 1 (assemble contigs and polish the contigs using short reads), we
550 obtained 616 contigs with a contig N50 of 23 Mbp and BUSCO [63] completeness of 94.20%. To
551 evaluate the quality of the contigs, we mapped the short reads from the Daweishan chicken to
552 these assembled contigs using Bowtie2 [47] allowing no mismatch and gaps, and found that
553 3.56% of nucleotides in the contigs could not be covered by the short reads. After Step 2
554 (scaffolding), the number of contigs decreased and contig N50 increased compared with the
555 results of Step 1, but BUSCO completeness decreased, and the number of Ns and percentage of
556 assembled nucleotides that cannot be covered by short reads increased. After Step 3 (fill gaps
557 using long reads), the number of contigs and number of Ns decreased, and the contig N50
558 increased compared with the results of Step 2. Meanwhile, the percentage of assembled
559 nucleotides that cannot be covered by short reads increased. After Step 4 (polish using long
560 reads), the number of contigs remained the same (368) but their total length and contig N50
561 increased, while the number of Ns decreased, compared with the results of Step 3. However,
562 BUSCO [63] completeness decreased, and percentage of assembled nucleotides that cannot be
563 covered by short reads increased. After Step 5 (polish using short reads), the number of contigs
564 remained the same (368), the number of Ns increase slightly but was only 1,452 with 157 gaps,
565 the contig N50 increased to 74 Mbp, the BUSCO [63] completeness increased to 96.70%, and
566 percentage of assembled nucleotides that could not be covered by short reads increased to
567 1.69%, suggesting that the assembly is of very high-quality. The Hi-C interaction heatmap of the
568 final assembly (Figure 4) shows that most of the scaffolds form a squared box along the main
569 diagonal of the heatmap matrix, indicating that the assembly is at chromosome-level. Taken

570 together, all these measures indicate the assembly is at chromosome-level of high-quality, and
571 thus our genome assembly pipeline works well.

572 Our gene annotation pipeline works seamlessly with our genome assembly pipeline for
573 accurately identifying protein-coding genes and pseudogenes. We demonstrated the usage of
574 our annotation pipeline by annotating the genes and pseudogenes on chromosome 10 (chr10)
575 of the assembled Daweishan chicken's genome as an example. For the homology-based
576 approach (Figures 2 and 3), we used CDS isoforms previously annotated in the red jungle fowl
577 (GRCg6a) and two commercial chickens (GRCg7b and GRCg7w) genomes as the references.
578 Applying the method to the target chr10 of Daweishan chicken, we found 421 intact genes, 6
579 partially supported genes and 13 pseudogenes on the chromosome (Table 2).

580 For the RNA-seq data-based approach (Figures 2 and 3), we used the RNA-seq reads of
581 150 bp with ~20X sequencing depth from multiple tissues of the Daweishan. Applying the
582 method to chr10 of Daweishan chicken, we found 26 new genes on chr10 of Daweishan chicken
583 (Table 2). Thus, we have identified a total of 453 protein-coding genes (including 26 new genes
584 not annotated in the reference genome) and 13 pseudogenes on chr10 of Daweishan chicken
585 (Table 2). In comparison, 432, 433 and 445 protein-coding genes, and 4, 2 and 4 pseudogenes,
586 were previously annotated in GRCg6a, GRCg7b and GRCg7w, respectively (Table 2). Therefore,
587 we identified more pseudogenes and also found new genes that were not annotated in the
588 chicken reference genomes by the NCBI annotation pipeline, suggesting that our pipeline could
589 find more putative new genes and pseudogenes.

590 To further demonstrate the accuracy of our gene annotation pipeline, we annotated the
591 genes for a Tibetan ground-tit individual based on its assembly from NCBI (GCF_000331425.1)

using our pipeline and compared our result with the that from NCBI. By using the CDS isoforms of genes in 53 well-annotated avian genomes from NCBI (Supplementary Table 1) as the reference genes, our homology-based method annotated 14,659 intact genes, 1,165 partially supported genes and 765 pseudogenes (Table 3). By using the RNA-seq reads from multiple tissues (liver and muscle) of the Tibetan ground-tit individual (SRA accession number: SRP018927), our RNA-seq data-based method annotated 118 new genes (Table 3). In total, we annotated 15,942 genes and 765 pseudogenes (Table 3), while NCBI annotates 15,814 genes and 83 pseudogenes in the genome (Table 3). There are 15,058 genes shared by the results of the two pipelines, thus, our pipeline could annotate almost all the genes (95.2%) predicted by the NCBI pipeline (Table 3). Our pipeline predicted substantially more pseudogenes than the NCBI pipeline, and only 41 pseudogenes are predicted by both pipelines (Table 3). For the inconsistent genes and pseudogenes predicted by the two pipelines, we found that NCBI pipeline often mistakenly predicts genes as pseudogenes, or mistakenly predict pseudogenes as genes. For examples, gene *DIO*3 is annotated as a protein-coding gene by the NCBI pipeline, but it is annotated as a pseudogene by our pipeline since there is a stop codon (TGA) in the middle (position: 411) of the ORF of the gene (Figure 5a); gene *LOC*102109485 is annotated as a pseudogene by the NCBI pipeline, but it is annotated as a protein-coding gene (named *LOC*107212260) by our pipeline since there is neither a stop codon nor a ORF shift mutation in the ORF of the gene (Figure 5b). Thus, these results suggest that our gene annotation pipeline is at least comparable to the NCBI pipeline in accuracy.

Author contributions statements

ZS supervised and conceived the project; SW, SY and ZS designed the pipeline; SW and ZS performed data analysis; and SW and ZS wrote the manuscript.

ORCID for corresponding authors
The ORCID for ZS is 0000-0003-4636-3440.

Acknowledgement
This work was supported by the Department of Bioinformatics and Genomics of the University of North Carolina at Charlotte.

Competing interests
The authors declare that they have no competing interests.

References

1. Sanger, F., S. Nicklen, and A.R. Coulson, *DNA sequencing with chain-terminating inhibitors.* Proc Natl Acad Sci U S A, 1977. **74**(12): p. 5463-7.
2. Lander, E.S., et al., *Initial sequencing and analysis of the human genome.* Nature., 2001. **409**(6822): p. 860-921.
3. Venter, J.C., et al., *The sequence of the human genome.* Science., 2001. **291**(5507): p. 1304-51.
4. Mardis, E.R., *Next-generation DNA sequencing methods.* Annu Rev Genomics Hum Genet, 2008. **9**: p. 387-402.
5. van Dijk, E.L., et al., *Ten years of next-generation sequencing technology.* Trends Genet, 2014. **30**(9): p. 418-26.
6. Schadt, E.E., S. Turner, and A. Kasarskis, *A window into third-generation sequencing.* Hum Mol Genet, 2010. **19**(R2): p. R227-40.
7. Clarke, J., et al., *Continuous base identification for single-molecule nanopore DNA sequencing.* Nat Nanotechnol, 2009. **4**(4): p. 265-70.
8. Eisenstein, M., *Oxford Nanopore announcement sets sequencing sector abuzz.* Nat Biotechnol, 2012. **30**(4): p. 295-6.
9. Jain, M., et al., *Nanopore sequencing and assembly of a human genome with ultra-long reads.* Nat Biotechnol, 2018. **36**(4): p. 338-345.
10. Belton, J.M., et al., *Hi-C: a comprehensive technique to capture the conformation of genomes.* Methods, 2012. **58**(3): p. 268-76.
11. Wenger, A.M., et al., *Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome.* Nat Biotechnol, 2019. **37**(10): p. 1155-1162.
12. Sereika, M., et al., *Oxford Nanopore R10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing.* Nat Methods, 2022. **19**(7): p. 823-826.
13. Li, H. and R. Durbin, *Genome assembly in the telomere-to-telomere era.* Nat Rev Genet, 2024.

14. Ruan, J. and H. Li, *Fast and accurate long-read assembly with wtdbg2.* Nat Methods, 2020. **17**(2): p. 155-158.
15. Koren, S., et al., *Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation.* Genome Res, 2017. **27**(5): p. 722-736.
16. Chin, C.S., et al., *Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data.* Nat Methods, 2013. **10**(6): p. 563-9.
17. Ghurye, J., et al., *Scaffolding of long read assemblies using long range contact information.* BMC Genomics, 2017. **18**(1): p. 527.
18. Ghurye, J., et al., *Integrating Hi-C links with assembly graphs for chromosome-scale assembly.* PLoS Comput Biol, 2019. **15**(8): p. e1007273.
19. Burton, J.N., et al., *Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions.* Nat Biotechnol, 2013. **31**(12): p. 1119-25.
20. Dudchenko, O., et al., *De novo assembly of the Aedes aegypti genome using Hi-C yields chromosome-length scaffolds.* Science, 2017. **356**(6333): p. 92-95.
21. Zhou, C., S.A. McCarthy, and R. Durbin, *YaHS: yet another Hi-C scaffolding tool.* Bioinformatics, 2023. **39**(1).
22. Cosma, B.M., et al., *Evaluating long-read de novo assembly tools for eukaryotic genomes: insights and considerations.* Gigascience, 2022. **12**.
23. Hou, Y., L. Wang, and W. Pan, *Comparison of Hi-C-Based Scaffolding Tools on Plant Genomes.* Genes (Basel), 2023. **14**(12).
24. Rhie, A., et al., *Towards complete and error-free genome assemblies of all vertebrate species.* Nature, 2021. **592**(7856): p. 737-746.
25. Christmas, M.J., et al., *Evolutionary constraint and innovation across hundreds of placental mammals.* Science, 2023. **380**(6643): p. eabn3943.
26. Wu, S., et al., *High quality assemblies of four indigenous chicken genomes and related functional data resources.* Sci Data, 2024. **11**(1): p. 300.
27. Wu, S., et al., *High-quality genome assembly of a C. crossoptilon and related functional and genetics data resources.* Sci Data, 2024. **11**(1): p. 247.
28. Stanke, M. and S. Waack, *Gene prediction with a hidden Markov model and a new intron submodel.* Bioinformatics, 2003. **19 Suppl 2**: p. ii215-25.
29. Majoros, W.H., M. Pertea, and S.L. Salzberg, *TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders.* Bioinformatics, 2004. **20**(16): p. 2878-9.
30. Blanco, E., G. Parra, and R. Guigó, *Using geneid to identify genes.* Curr Protoc Bioinformatics, 2007. **Chapter 4**: p. Unit 4.3.
31. Burge, C. and S. Karlin, *Prediction of complete gene structures in human genomic DNA.* J Mol Biol, 1997. **268**(1): p. 78-94.
32. Korf, I., *Gene finding in novel genomes.* BMC Bioinformatics, 2004. **5**: p. 59.
33. Lomsadze, A., et al., *Gene identification in novel eukaryotic genomes by self-training algorithm.* Nucleic Acids Res, 2005. **33**(20): p. 6494-506.
34. Coghlan, A. and R. Durbin, *Genomix: a method for combining gene-finders' predictions, which uses evolutionary conservation of sequence and intron-exon structure.* Bioinformatics, 2007. **23**(12): p. 1468-75.
35. Nachtweide, S. and M. Stanke, *Multi-Genome Annotation with AUGUSTUS.* Methods Mol Biol, 2019. **1962**: p. 139-160.

36. Keilwagen, J., F. Hartung, and J. Grau, *GeMoMa: Homology-Based Gene Prediction Utilizing Intron Position Conservation and RNA-seq Data.* Methods Mol Biol, 2019. **1962**: p. 161-177.
37. Kollmar, M., *Predicting Genes in Closely Related Species with Scipio and WebScipio.* Methods Mol Biol, 2019. **1962**: p. 193-206.
38. Brůna, T., et al., *BRAKER2: automatic eukaryotic genome annotation with GeneMark-EP+ and AUGUSTUS supported by a protein database.* NAR Genom Bioinform, 2021. **3**(1): p. lqaa108.
39. Rey, C., et al., *CAARS: comparative assembly and annotation of RNA-Seq data.* Bioinformatics, 2019. **35**(13): p. 2199-2207.
40. Banerjee, S., et al., *FINDER: an automated software package to annotate eukaryotic genes from RNA-Seq data and associated protein sequences.* BMC Bioinformatics, 2021. **22**(1): p. 205.
41. Wu, S., et al., *Annotations of four high-quality indigenous chicken genomes identify more than one thousand missing genes in subtelomeric regions and micro-chromosomes with high G/C contents.* BMC Genomics, 2024. **25**(1): p. 430.
42. English, A.C., et al., *Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology.* PLoS One, 2012. **7**(11): p. e47768.
43. Vaser, R., et al., *Fast and accurate de novo genome assembly from long uncorrected reads.* Genome Res, 2017. **27**(5): p. 737-746.
44. Hu, J., et al., *NextPolish: a fast and efficient genome polishing tool for long-read assembly.* Bioinformatics, 2020. **36**(7): p. 2253-2255.
45. Kapustin, Y., et al., *Splign: algorithms for computing spliced alignments with identification of paralogs.* Biol Direct, 2008. **3**: p. 20.
46. Langmead, B., et al., *Scaling read aligners to hundreds of threads on general-purpose processors.* Bioinformatics, 2019. **35**(3): p. 421-432.
47. Langmead, B. and S.L. Salzberg, *Fast gapped-read alignment with Bowtie 2.* Nat Methods, 2012. **9**(4): p. 357-9.
48. Langmead, B., et al., *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.* Genome Biol, 2009. **10**(3): p. R25.
49. Quast, C., et al., *The SILVA ribosomal RNA gene database project: improved data processing and web-based tools.* Nucleic Acids Res, 2013. **41**(Database issue): p. D590-6.
50. Dobin, A., et al., *STAR: ultrafast universal RNA-seq aligner.* Bioinformatics, 2013. **29**(1): p. 15-21.
51. Grabherr, M.G., et al., *Full-length transcriptome assembly from RNA-Seq data without a reference genome.* Nat Biotechnol, 2011. **29**(7): p. 644-52.
52. Chen, M.M., et al., *The GFF3toolkit: QC and Merge Pipeline for Genome Annotation.* Methods Mol Biol, 2019. **1858**: p. 75-87.
53. Nawrocki, E.P. and S.R. Eddy, *Infernal 1.1: 100-fold faster RNA homology searches.* Bioinformatics, 2013. **29**(22): p. 2933-5.
54. Kalvari, I., et al., *Rfam 14: expanded coverage of metagenomic, viral and microRNA families.* Nucleic Acids Res, 2021. **49**(D1): p. D192-d200.
55. Nurk, S., et al., *HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads.* Genome Res, 2020. **30**(9): p. 1291-1305.

56. Cheng, H., et al., *Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm.* Nat Methods, 2021. **18**(2): p. 170-175.
57. Rautiainen, M., et al., *Telomere-to-telomere assembly of diploid chromosomes with Verkko.* Nat Biotechnol, 2023. **41**(10): p. 1474-1482.
58. Li, H., et al., *The Sequence Alignment/Map format and SAMtools.* Bioinformatics, 2009. **25**(16): p. 2078-9.
59. Li, H., *Minimap2: pairwise alignment for nucleotide sequences.* Bioinformatics, 2018. **34**(18): p. 3094-3100.
60. Li, H. and R. Durbin, *Fast and accurate short read alignment with Burrows-Wheeler transform.* Bioinformatics, 2009. **25**(14): p. 1754-60.
61. Quinlan, A.R. and I.M. Hall, *BEDTools: a flexible suite of utilities for comparing genomic features.* Bioinformatics, 2010. **26**(6): p. 841-2.
62. Lindhorst, D. and M.S. Halfon, *Reporter gene assays and chromatin-level assays define substantially non-overlapping sets of enhancer sequences.* BMC Genomics, 2023. **24**(1): p. 17.
63. Simão, F.A., et al., *BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs.* Bioinformatics, 2015. **31**(19): p. 3210-2.

Table 1: Results of each step for assembling the Daweishan chicken genome

Steps	# Congits (Total bp) (# Ns)	Contig N50 (bp) (L50)	BUSCO completeness	Percentage of assembled nucleotides that cannot be covered by short reads
Step 1 -Wtdbg2	616 (1,018,887,028) (0)	22,952,302 (13)	94.20%	3.56%
Step 2 -SALSA	407 (1,018,887,028) (108,500)	73,917,222 (5)	94.10%	3.57%
Step 3 -Pbjelly	368 (1,026,115,452) (14,224)	73,988,966 (5)	94.10%	4.22%
Step 4 -Racon	368 (1,036,024,190) (1,306)	74,243,019 (5)	85.00%	14.95%
Step 5 -NextPolish	368 (1,037,025,916) (1,452)	74,334,266 (5)	96.70%	1.69%

Table 2: Annotation result of chr10 of Daweishan chicken in comparison with those of the three reference chicken genomes

Chicken	Homology-based			RNA-seq data based	# Total protein-coding genes	# Total pseudogenes
	# Intact genes	# Partially supported genes	# Pseudogenes	# New genes		
Daweishan	421	6	13	26	453	13
GRCg6a	-	-	-	-	432	4
GRCg7b	-	-	-	-	433	2
GRCg7w	-	-	-	-	445	4

Table 3: Comparison of gene annotation results of the Tibetan ground-tit genomes by our pipeline and the NCBI pipeline

Methods	Homology-based			RNA-seq data based	# Total protein-coding genes	# Total pseudogenes
	# Intact genes	# Partially supported genes	# Pseudogenes	# New genes		
Our pipeline	14,659	1,165	765	118	15,942	765
NCBI pipeline	-	-	-	-	15,814	83
Common	-	-	-	-	15,058	41

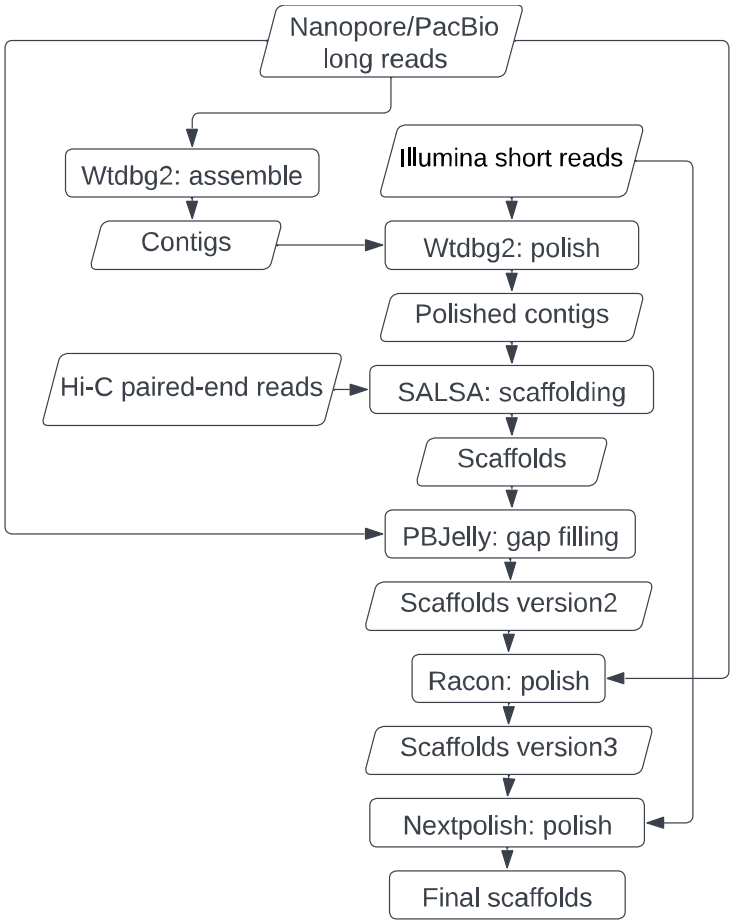

Figure 1. Flowchart of the genome assembly pipeline

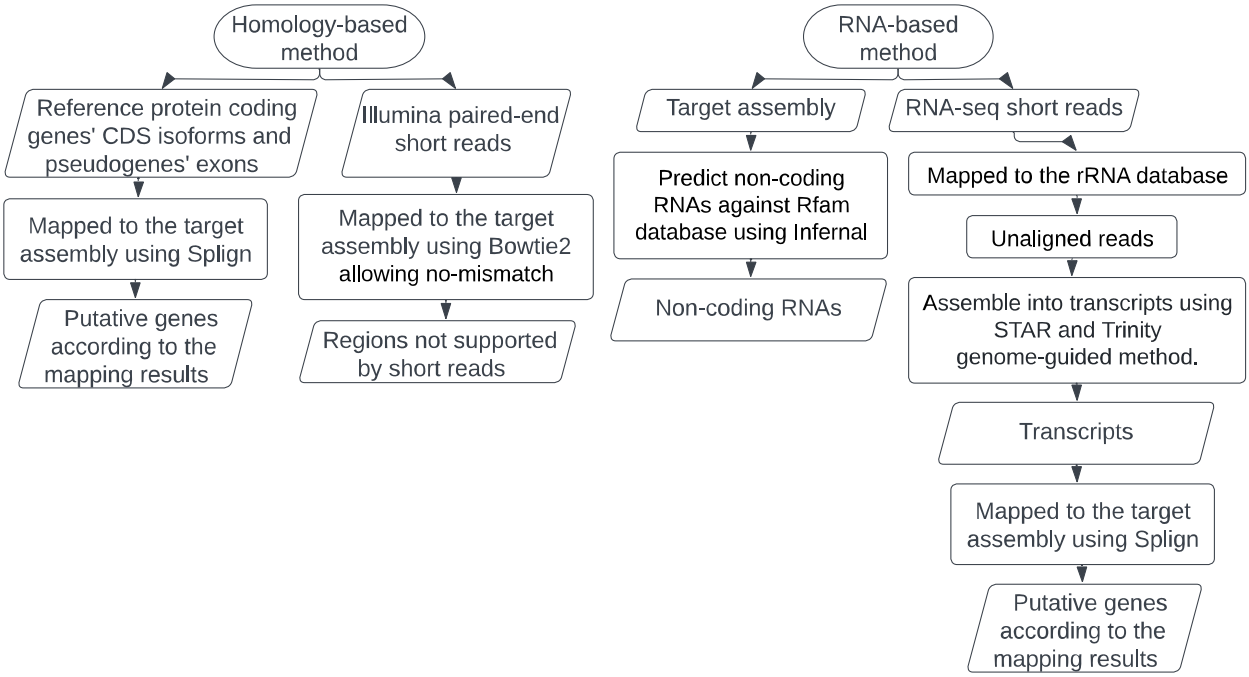

Figure 2. Flowcharts of data preparation for our homology-based and RNA-seq data-based methods in the annotation pipeline

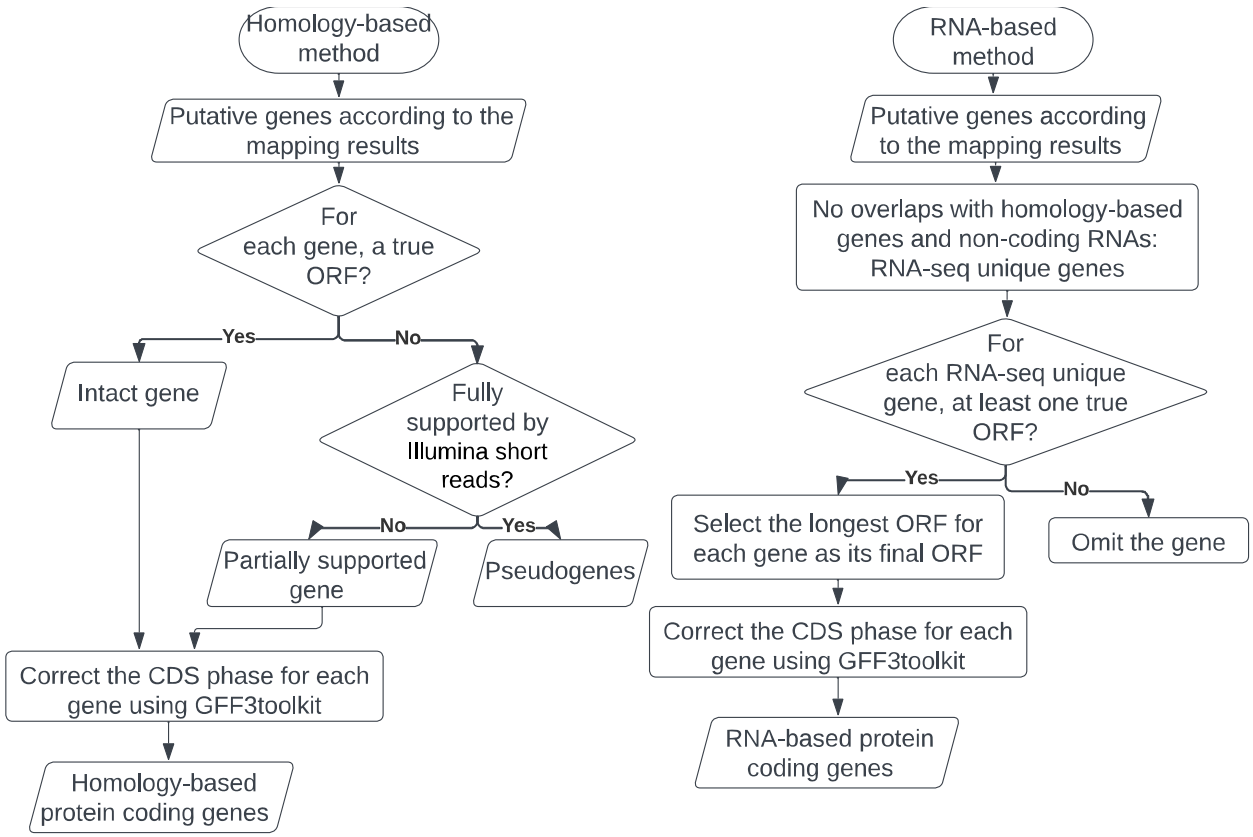

Figure 3. Flowcharts of our homology-based and RNA-seq data-based gene finding procedures of the annotation pipeline

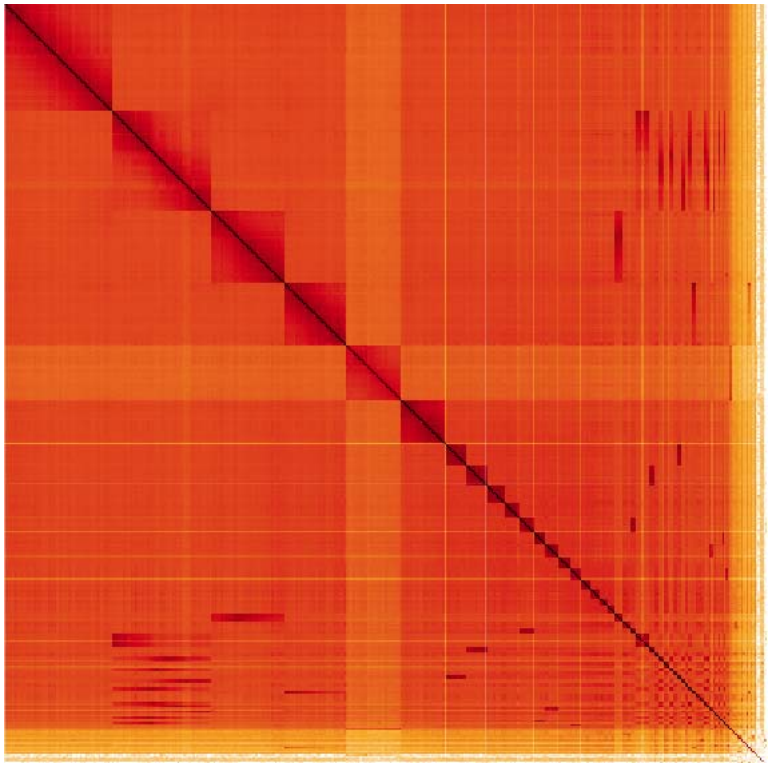

Figure 4. Hi-C interaction heatmap of the final genome assembly of the Daweishan chicken individual

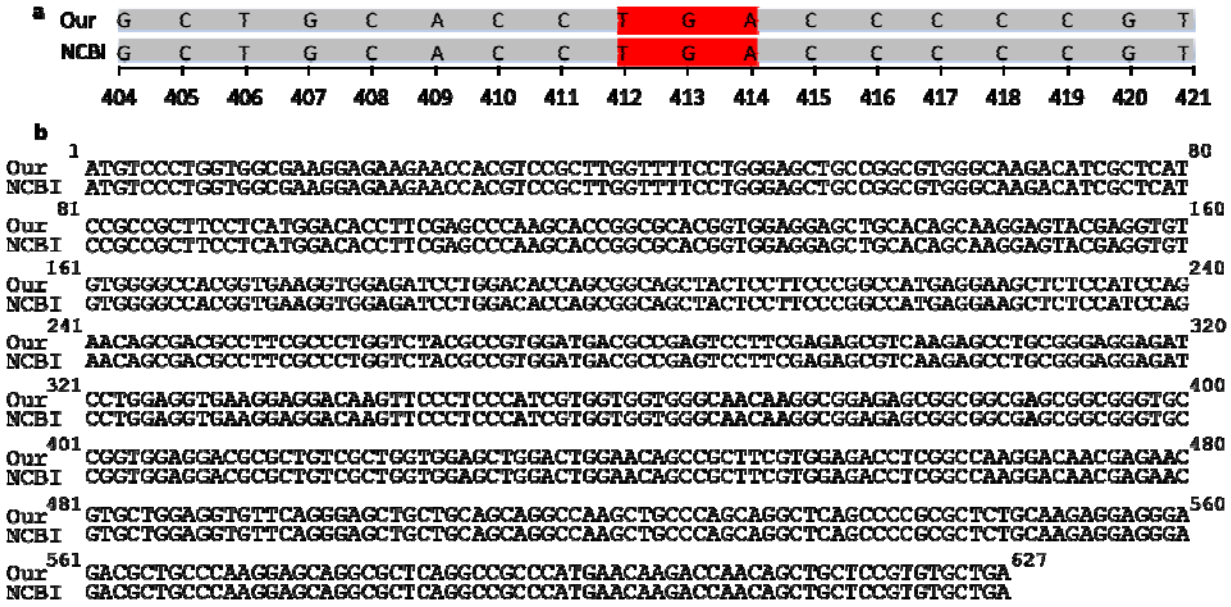

Figure 5. Examples of different annotations for the same gene/pseudogene from the two gene annotation pipelines. **a.** Part of ORFs of gene *DIO*3 annotated by the two pipelines contains a stop codon labeled in red, which is missed by the NCBI pipeline. **b.** The ORF of gene *LOC*102109485 annotated by the two pipelines. The sequence contains no pseudogenization mutations, however, NCBI pipeline considers it as a pseudogene.