SyBR Documentation

(1) Overview

SYBR offers a fully automated and scalable workflow that integrates synteny analysis, evolutionary breakpoint detection, functional enrichment, and ancestral genome reconstruction to facilitate the study of genome evolution. Even when working with fragmented or incomplete genome assemblies, the pipeline enables the detailed resolution of genomic conservation, chromosomal rearrangements, and the evolutionary context in which these events occurred by integrating high-confidence alignment filtering with downstream analytical modules. Because of this, SYBR is a valuable tool for comparative genomics and for expanding our understanding of the processes that influence chromosome structure and evolutionary history.

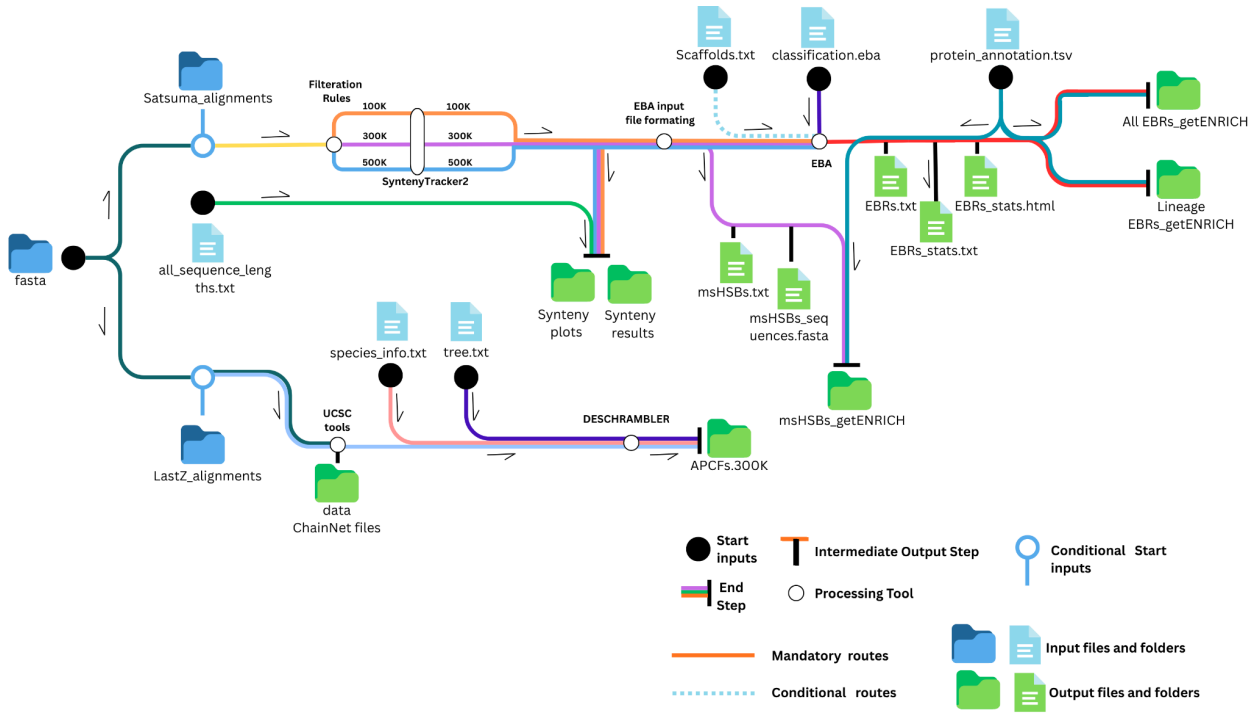


Fig. 1: Schematic illustration of the workflow of the SyBR pipeline. Showing input files/folders (blue in colour) and output files/folders (green in colour). Black-filled dots indicate the input at a particular step. Solid lines with a particular colour show the route of a specific file in the workflow. Dotted lines show the conditional input.

(2) Installation:

The Sybr pipeline can be installed via GitHub, and dependencies can be installed in a conda environment.

(2.1) Prerequisites:

All the dependencies need to be installed via conda. So conda should be installed before installing the SyBR pipeline.

(2.1.1) Conda Installation:

This project requires Conda to manage dependencies. If you already have Conda (Miniconda or Anaconda) installed, skip to this.

Check if Conda is Already Installed

Open a terminal and run:

```
conda --version
```

- If you see something like **conda 24.x.x** — Conda is already installed, skip the section below.
- If you see **'command not found,'** install Conda.

If Conda is not installed on your system, follow this step-by-step guide for Ubuntu:

👉
https://medium.com/@mustafa_kamal/a-step-by-step-guide-to-installing-conda-in-ubuntu-and-creating-an-environment-d4e49a73fc46

(2.2) Installation of Sybr:

(2.2.1) Clone the Repository:

Clone the project from GitHub using the following command:

```
git clone https://github.com/BioinformaticsOnLine/Sybr.git
```

(2.2.2) Set File Permissions:

Grant the necessary permissions to all files and directories:

```
chmod -R 777 Sybr
```

(2.2.3) Navigate to Project Directory:

Change the directory into the project folder

```
cd Sybr
```

(2.2.4) Create Conda Environment:

Install all required dependencies by creating the Conda environment from the provided YAML file

```
conda env create -f install_sybr_dependence.yml
```

(2.2.5) Activate Conda Environment:

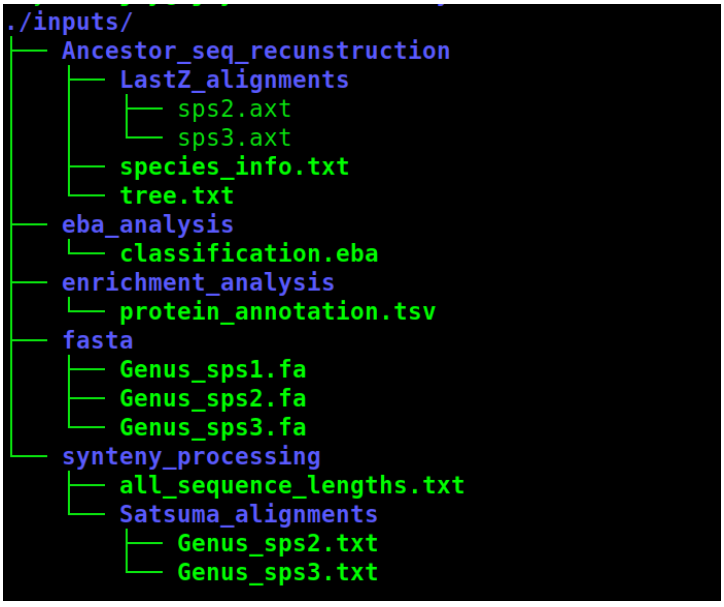
```
conda activate sybr
```

(2.3) Preparation of Input Files:

For the user's reference, we provide an example of input data. User can download the example data from these links:
Fast track: - Example input files with pre-computed alignments:
https://doi.org/10.6084/m9.figshare.32315682

Slow track: -Example input files without pre-computed alignments:
https://doi.org/10.6084/m9.figshare.32315892

For each module, there are separate input files and folders. Users can provide the inputs according to the result requirements, but the file structure is fixed. The user should provide the input in the same structure as this:



This pipeline is run in five modules. Users can select the modules to run based on their needs. Five modules are:

```
# ── Pipeline stages to run ──────────
run_stages:
  run_satsuma_alignment: false
  run_lastz_alignment: false
  synteny_processing: true
  eba_analysis: false
  enrichment_analysis: false
  chainNet_generation: false
  Ancestor_seq_recunstruction: false
```

To activate the module to run, users need to set it to **true** in **run_sybr_config.yaml.** And to deactivate the module, set it to **false**. For every module, specific file formats and structures are required. Explained in detail below:

(2.3.1) Common input files for multiple modules:

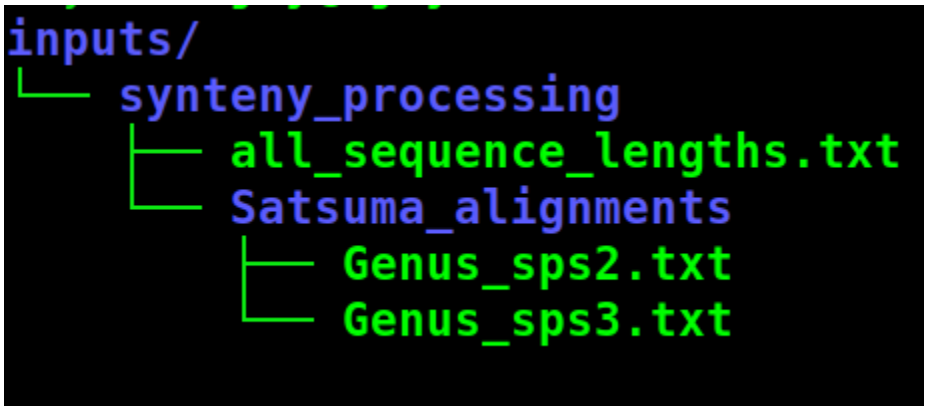
(2.3.2) fasta:

In the input folder, folder named fasta contains the input fasat files which are common for the modules **run_satsuma_alignment, run_lastz_alignment, eba_analysis,** and **chainNet_generation.** The format of input fasta files should be like: **Genus_species.fa** and every header should contain numeric value only.

```
(base) ajay@ajay-SS400TR-34:~/example/slow_track$ ls
Ancestor_seq_recunstruction  eba_analysis  enrichment_analysis  fasta  synteny_processing
(base) ajay@ajay-SS400TR-34:~/example/slow_track$ cd fasta/
(base) ajay@ajay-SS400TR-34:~/example/slow_track/fasta$ ls
Genus_sps1.fa  Genus_sps2.fa  Genus_sps3.fa
(base) ajay@ajay-SS400TR-34:~/example/slow_track/fasta$ head Genus_sps1.fa
>1
TGATAATCTATTATTTTCCAATGTTGgattataaacaaatgttaacatgaaatcaaattt
agcCATTTCATATGATTGTTCTAATGATACTTCTTCTaattttgtattattcatcGATAA
GTAAGTTTCACTTTCGGACAAAGTGATAAAATCGAACATTGTTTCCAAAAAGCTAAGATT
AGATTGATTGAGATGAAAGTCACTGAGAATATGTTGAAGTGGATAATGAGAATGTTCAAG
TATTGATAAACACATCTCTCTTACAGAGTGAACAACTTCATCAAATGACCAATCTCCATT
GAGCTGAACACGATATGGTAATGTTGCCACAAACATTCCTATCATATTATACAATTCAGA
TCGATATCGATTGGCATTTATAGTACCAACACACAAATCAGTTTGATTATGagataattt
gaatagaaatacatAAAATATCGATAGTCCAAGTTGAAACAGTGAAAGATGATGTGATGA
TGCATAATTTAGAAATGATGTACATATAGCATCATCTAATGTTATTTGAGTTCTAGAAGC
(base) ajay@ajay-SS400TR-34:~/example/slow_track/fasta$
```

(2.3.2.1) Input Files for synteny_processing:

The file structure of **synteny_processing** should be like this:



(2.3.2.2) all_sequence_lengths.txt:

This is a tab-delimited text file with three columns and without headers. The first column is the **chromosome** or **scaffold** number; this number should be numeric only (e.g., 1, 2, 3…).

The first column is the **chromosome** or **scaffold** number; this number should be **numeric** only (e.g., 1, 2, 3…). The second column contains the size or length of the **chromosome** or **scaffold**. The third column is for the name of the **species**. It should match what is mentioned in the other

input files. This file contains information on all sequences from all species. The ~~lengths of~~ the reference genome's sequences should be written three times, once for each resolution selected for synteny analysis.

In the SYBR tool's home folder, the **genome_length_maker.sh** script is provided, which creates **all_sequence_lengths.txt.** The usage of this script is like this:

```
(sybr) ajay@ajay-SS400TR-34:~/example/sybr$ ./genome_length_maker.sh -h
Usage: genome_length_maker.sh -r <reference_species> [options]

Required:
  -r, --reference   Basename of the reference FASTA (no extension)

Optional:
  -i, --input       Folder containing all FASTA files (default: current directory)
  -o, --output      Output file path (default: <input_folder>/all_sequence_lengths.txt)
  -k, --resolutions Comma- or space-separated resolutions in kb (default: 100,300,500)
  -h, --help        Show this help

Examples:
  genome_length_maker.sh -r Genus_sps1
  genome_length_maker.sh -r Genus_sps1 -i inputs/fasta
  genome_length_maker.sh -r Genus_sps1 -i inputs/fasta -o inputs/synteny_processing/all_sequence_lengths.txt
  genome_length_maker.sh -r Genus_sps1 -i inputs/fasta -k 100,300,500,1000
  genome_length_maker.sh -r Genus_sps1 -i inputs/fasta -o my_lengths.txt -k 200 400
(sybr) ajay@ajay-SS400TR-34:~/example/sybr$
```

```
1       18146847        Genus_sps1:100k
3       20354754        Genus_sps1:100k
5       16930519        Genus_sps1:100k
1       18146847        Genus_sps1:300k
3       20354754        Genus_sps1:300k
5       16930519        Genus_sps1:300k
1       18146847        Genus_sps1:500k
3       20354754        Genus_sps1:500k
5       16930519        Genus_sps1:500k
1       20317728        Genus_sps2
3       22412911        Genus_sps2
5       23051187        Genus_sps2
1       20932626        Genus_sps3
3       21077837        Genus_sps3
5       17364795        Genus_sps3
```

This script calculates the lengths of all sequences across all genomes in the folder where it runs. The name of all the genomes should be written as **Genus_species.fasta.**

(2.3.2.3) Scaffolds.txt:

This file contains the names of species whose genome assemblies are **not at the chromosome level**. If all the genome assemblies are chromosome-level, then modify the **pipeline_paths.yaml** from this

```
scaffolds_file:     "inputs/synteny_processing/Scaffolds.txt"
```

To this:

```
scaffolds_file:     "ALL_CHROMOSOMES"
```

(2.3.2.4) Satsuma_alignments:

```
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/synteny_processing/Satsuma_alignments$ ls
Genus_sps2.txt  Genus_sps3.txt
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/synteny_processing/Satsuma_alignments$ head Genus_sps2.txt
1       60754   61163   3       694776  695185  0.562347        +
1       61245   61679   3       695291  695726  0.571429        +
1       137760  137883  3       4500431 4500554 0.528455        +
1       137897  139971  3       4500616 4502696 0.828833        +
1       158055  158342  3       80801   81088   0.710801        +
1       158813  159447  3       81141   81777   0.701893        +
1       223892  224179  3       47888   48175   0.609756        +
1       224208  224421  3       48148   48361   0.586854        +
1       224259  224418  3       48148   48307   0.503145        +
1       224419  224597  3       48345   48523   0.595506        +
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/synteny_processing/Satsuma_alignments$
```

In the **Satsuma_alignments** folder, all the alignment files present are aligned by the Satsuma tool. The format of the name of alignment files should be **Genus_species.txt**. Each file is a tab-delimited text file with 8 columns; it should not have headers. The first column is the **chromosome number** / **scaffold number** (numeric only), the second column is the **reference start coordinates**, the third column is the **reference end coordinates**, and the fourth column is the **query chromosome/scaffold number (numeric)**. The fifth column is **query start coordinates**, the sixth column is **query end coordinates,** the seventh column is the **score** of Satsuma, and the eighth column is the strand.

(2.3.3) Input Files for eba_analysis:

The file structure to run **eba_analysis** should be like this:

```
./inputs/
└── eba_analysis
    └── classification.eba
```

(2.3.3.1) classification.eba:

```
lineage=
Genus_sps1=Genus_sps2,Genus_sps3,Genus_sps4,Genus_sps5
node1=Genus_sps2,Genus_sps3
```

This is a **classification file** for **EBA**. In this file, the user can classify groups of related species by **node** or **order**. The first line of this file should be lineage=, the second line should start with the name of the reference species, and all remaining species names should follow the = sign. In the next lines, species can be classified into different **groups**.

For a detailed explanation, the user should visit the documentation of the EBA tool:
https://pure.aber.ac.uk/ws/portalfiles/portal/10624966/Narayan_J.pdf

(2.3.4) Input Files for enrichment_analysis:

The file structure for the **enrichment_analysis** module should be like this:

```
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/fasta$ ls
Genus_sps1.fa  Genus_sps2.fa  Genus_sps3.fa
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/fasta$ head Genus_sps1.fa
>1
TGATAATCTATTATTTTCCAATGTTGgattataaacaaatgttaacatgaaatcaaattt
agcCATTTCATATGATTGTTCTAATGATACTTCTTCTaattttgtattattcatcGATAA
GTAAGTTTCACTTTCGGACAAAGTGATAAAATCGAACATTGTTTCCAAAAAGCTAAGATT
AGATTGATTGAGATGAAAGTCACTGAGAATATGTTGAAGTGGATAATGAGAATGTTCAAG
TATTGATAAACACATCTCTCTTACAGAGTGAACAACTTCATCAAATGACCAATCTCCATT
GAGCTGAACACGATATGGTAATGTTGCCACAAACATTCCTATCATATTATACAATTCAGA
TCGATATCGATTGGCATTTATAGTACCAACACACAAATCAGTTTGATTATGagataattt
gaatagaaatacatAAAATATCGATAGTCCAAGTTGAAACAGTGAAAGATGATGTGATGA
TGCATAATTTAGAAATGATGTACATATAGCATCATCTAATGTTATTTGAGTTCTAGAAGC
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/fasta$
```

(2.3.4.1) protein_annotation.tsv:

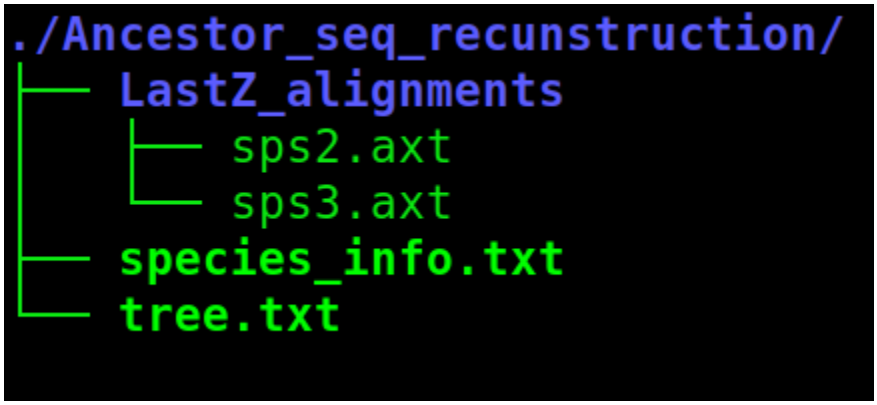
```
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/enrichment_analysis$ ls
protein_annotation.tsv
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/enrichment_analysis$ head protein_annotation.tsv
CP075492.1      4391    5403    3       UJR06537.1
CP075492.1      8205    9286    3       UJR06538.1
CP075492.1      13149   16010   3       UJR06539.1
CP075492.1      19366   33899   3       UJR06540.1
CP075492.1      45201   48426   3       UJR06541.1
CP075492.1      49650   50896   3       UJR06542.1
CP075492.1      51562   53479   3       UJR06543.1
CP075492.1      60237   61256   3       UJR06544.1
CP075492.1      62461   64604   3       UJR06545.1
CP075492.1      66488   70489   3       UJR06546.1
(sybr) ajay@ajay-SS400TR-34:~/example/inputs/enrichment_analysis$
```

This is a **tab-delimited** text file **without headers**, with **five columns**. The first column contains the **chromosome accession number** of the reference genome. The second column represents the **start coordinates** of the gene in the reference genome. The third column is the **end coordinates**. The fourth column contains the **chromosome number (numeric only)**, and the fifth column contains the accession number for **protein-coding genes or gene ID according to the need in getENRICH tool.**

Users can download this file from NCBI.

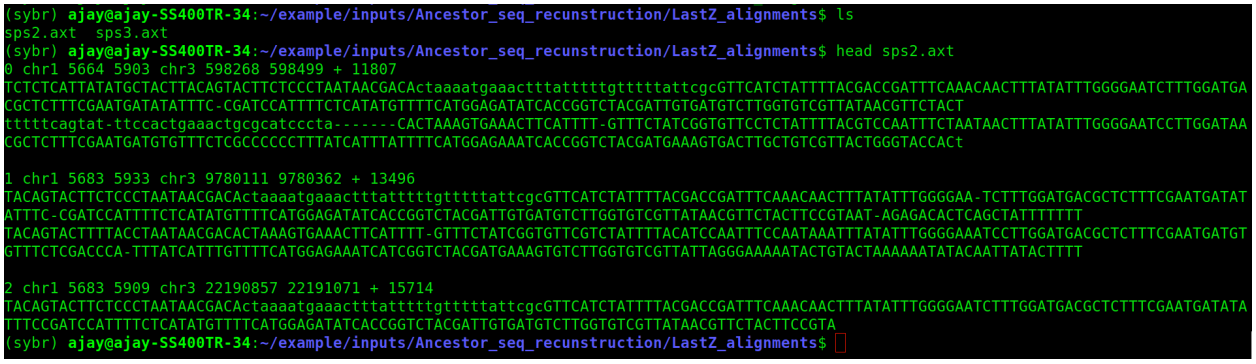
(2.3.5) Input Files for Ancestor_seq_reconstruction:

The file structure to run **Ancestor_seq_reconstruction** should be like this:

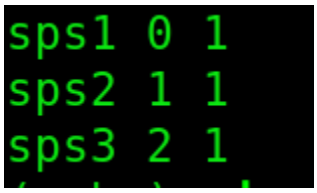


(2.3.5.1) LastZ_alignments:

In the **inputs** folder **LastZ_alignments**, the user should put lastz alignment files with the **.axt** extension. The file name format should be: **species. axt.**



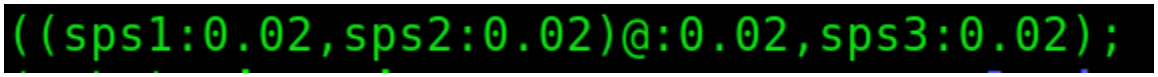
(2.3.5.2) species_info.txt:



This file contains information about the **reference**, **descendant**, **outgroup**, **chromosome-scale**, and **scaffold-level assembly**. There are three columns in this file separated by a **space**. The first column is the **species name**, the same as the file's name in the **seq** and **LastZ_alignments** folders. The second column represents the **species' role**. **0 = reference** , **1 = descendant, 2 = outgroup**. The third column represents the **assembly level**. **1 = chromosome-scale**, **0 = scaffold-level**.
Please refer to https://github.com/jkimlab/DESCHRAMBLER for details.

(2.3.5.3) tree.txt:

This file contains the Newick tree for the species listed in the species_info.txt file. The target ancestor must be specified by the "@" symbol. Please refer to https://github.com/jkimlab/DESCHRAMBLER for details.



(3) How to run the SyBR pipeline:

```
# ──────────────────────────────────────────────────────────────────────
#  run_sybr_config.yaml  –  User-facing config
#  Edit this file to control pipeline behaviour
# ──────────────────────────────────────────────────────────────────────

# ── Base I/O directories ──────────────────────────────────────────────
# Set these to absolute paths if your inputs/outputs live outside the
# workflow directory.  Leave them as "inputs" / "outputs" to use the
# default folders relative to the workflow root.
base_input_dir:  "/home/ajay/example/inputs"
base_output_dir: "/home/ajay/example/outputs"

# ── Pipeline stages to run ────────────────────────────────────────────
run_stages:
  run_satsuma_alignment: true
  run_lastz_alignment: true
  synteny_processing: true
  eba_analysis: true
  enrichment_analysis: true
  chainNet_generation: true
  Ancestor_seq_recunstruction: true

# ── Species / reference names ─────────────────────────────────────────
reference_name: "Genus_sps1"
reference_species: "sps1"

# ── EBA parameters (user-facing) ──────────────────────────────────────
eba:
  n: 2           # number of EBA iterations
  r: "Genus_sps1"  # reference species name
  p: 300         # resolution parameter  # M
```

(3.1) run_sybr_config.yaml:

After setting the file structure in the input folder, the user must create a config file named **run_sybr_config.yaml.** Provide the path of the input folder in **base_input_dir** and the path of the output folder in **base_output_dir**. After setting the paths, set the modules you want to run to true. In **reference_name** and '**r:'** in **eba**, provide the name of the **reference species**.

For **reference_species**, provide the **species name**. In **EBA**, '**n:'** represents the number of species **excluding the reference species**, while **'p:'** denotes the primary resolution for synteny processing. In **getenrich**, **'r:'** represents the **KEGG code. Ko** represents the KEGG ortholog.

If the KEGG organism list includes reference species, provide the KEGG organism code for each.
The user can review the KEGG organism list at this link:

https://www.genome.jp/kegg/tables/br08606.html

(3.2) pipeline_paths.yaml:

This config file remains unchanged. The user needs to change this configuration file if all the genomes in the analysis are chromosome-level. In this case, the value of **scaffolds_file** will be **ALL_CHROMOSOMES** instead of the path to Scaffolds.txt**. If some genomes are not chromosome-level, then leave them.**

```
# ── EBA analysis — paths ──────────────────────────────────────────
eba_format:
  pre_EBA_dir:        "outputs/eba_analysis/pre-EBA"
  scaffolds_file:     "ALL_CHROMOSOMES"   # use path"inputs/synteny_processing/Scaffolds.txt" or "ALL_CHROMOSOMES" to include everything
  eba_input_dir:      "inputs/eba_analysis/EBA-input"
  mshsbs_dir:         "outputs/eba_analysis/msHSBs"
  ebrs_dir:           "outputs/eba_analysis/EBRs"
  copy_destination_dir: "outputs/eba_analysis/EBA_results"
```

(3.3) sybr.sh:

```
(sybr) ajay@ajay-SS400TR-34:~/example/sybr$ ./sybr.sh -h

SYBR
============

========================================
    Snakemake Pipeline
========================================

Usage: ./sybr.sh [OPTIONS]

Options:
  -c, --config FILE      Configuration file (default: run_sybr_config.yaml)
  -P, --paths FILE       Static paths file (default: pipeline_paths.yaml)
  -j, --cores N          Number of cores (default: all available)
  -t, --target RULE      Target rule (default: all)
  -l, --log FILE         Log output to file
  -u, --unlock           Unlock working directory
  -n, --dry-run          Dry run (simulate pipeline)
  -k, --keep-going       Keep going on independent job failures
  -v, --verbose          Verbose Snakemake output
  -s, --skip-validation  Skip input validation
  -C, --clean            Remove intermediate files after successful completion
  -w, --window-sizes     Comma-separated window sizes in bp (e.g., 100000,300000,500000)
  -p, --step-size        Step size in bp for synteny assignment (default: 30000)
  -h, --help             Show this help

Examples:
  ./sybr.sh -c config.yaml -j 8                       # Run with default settings
  ./sybr.sh --window-sizes 200000,400000 --step-size 50000  # Custom window sizes and step size
  ./sybr.sh --window-sizes 100000                      # Single window size
  ./sybr.sh --step-size 25000                         # Custom step size with default windows
  ./sybr.sh --clean                                    # Delete intermediate files after run

Note: Window sizes and step size only affect synteny_assign rules. If not specified,
      defaults are: window sizes = 100000,300000,500000 and step size = 30000.
```

The file is a helper script for running this Snakemake pipeline. To see all the options in the pipeline, run this script with the help option: **sybr.sh -h.**

The very basic usage of sybr pipeline is this:

In this command, the **window size** will default to **100000, 300000,** and **500000**. And the **step size** will be **30000**. To use a **custom** window and step size, the user can use the flags **-w** and **-p** as shown in the help section. For a detailed understanding of window size and step size, please refer to https://pmc.ncbi.nlm.nih.gov/articles/PMC2726151/

To get clean output and remove **intermediate files,** the user can use the **-C (capital C)** flag. **-c (small c)** is for **run_sybr_config.yaml** config file, which will be used for the renamed config file.

```
(sybr) ajay@ajay-SS400TR-34:~/example/sybr$ ./sybr.sh -j 8 -C
```

This is the output of **./sybr.sh -j 8 -C** when only the **synteny_processing** module is activated.

(4) Outputs of Sybr:

These are the three folders generated as a result of running sybr pipeline with all modules activated:

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs$ ls
Ancestor_seq_recunstruction  eba_analysis  synteny_processing
(sybr) ajay@ajay-SS400TR-34:~/example/outputs$
```

The results of the **synteny_processing** module will be generated in the **synteny_processing** folder in the **output folder**. The results of the **eba_analysis** and **enrichment_analysis** modules will be generated inside the **eba_analysis** folder in the output folder. The results of the **chainNet_generation** and **Ancestor_seq_reconstruction** modules will be generated in the **Ancestor_seq_reconstruction** folder within the **output folder**.

(4.1) synteny_processing:

The **synteny_processing** output folder has two subfolders: **synteny_plots** and **synteny_results**

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/synteny_processing$ ls
Genus_sps2_out  Genus_sps3_out  Satsuma_alignments  synteny_out  synteny_plots
```

The **synteny_results** output folder contains subfolders for each **resolution;** within each resolution folder, there are subfolders for **each species**. And in each species folder, there is a **blocks_info** file, which contains the processed synteny information of reference and query at a **particular resolution**.

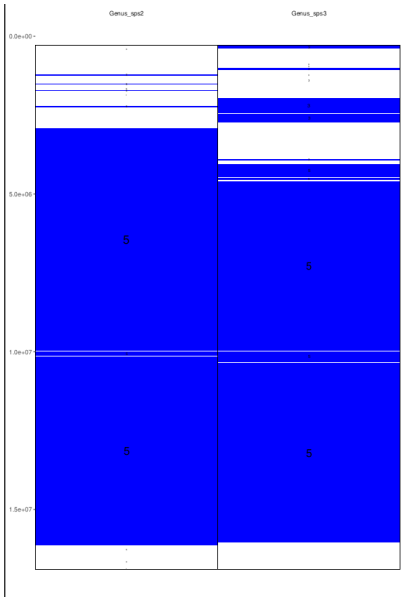
The **blocks_info** file looks like this:

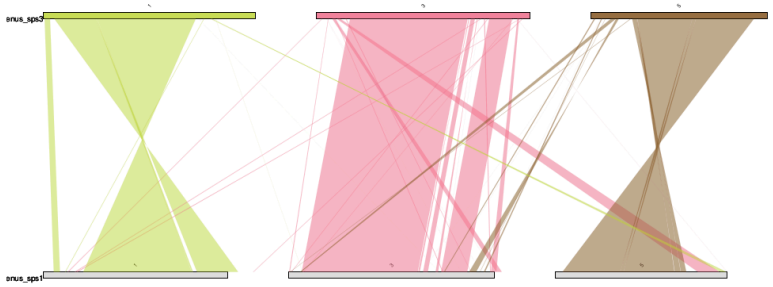
Reference_genome	Genus_sps1_chromosome		Genus_sps1_start		Genus_sps1_end		Genus_sps2_chromosome	Genus_sps2_start		G
enus_sps2_end	Sign	Target_genome	HSB	Comments						
nominal	nominal	int	int	nominal	int	int	nominal	int	nominal	
Genus_sps1	1	187435	252567	3	48148	144775	-	Genus_sps2	1	
Genus_sps1	1	263336	394750	3	287640	365543		Genus_sps2	2	
Genus_sps1	1	651658	14925643		1	4144289	17927843	+	Genus_sps2	3
Genus_sps1	1	15938248		15941708		5	7325799	7329815	Genus_sps2	4
Genus_sps1	1	15952248		15955766		5	21889045	21892509	Genus_sps2	5
Genus_sps1	1	16040269		16041333		3	6745506	6746569	Genus_sps2	6
Genus_sps1	1	16129084		16587181		5	20772627	21899286	Genus_sps2	7
Genus_sps1	1	16624731		16628324		5	19932344	19934912	Genus_sps2	8
Genus_sps1	1	17478154		17481723		3	20998879	21002360	Genus_sps2	9
Genus_sps1	3	144788	145602	3	22411439		22412290	Genus_sps2	10	
Genus_sps1	3	749860	974432	5	5296323	6072366	+	Genus_sps2	11	
Genus_sps1	3	1079412	1080821	5	7119670	7120752		Genus_sps2	12	
Genus_sps1	3	1393190	1461047	5	1907335	1978352		Genus_sps2	13	
Genus_sps1	3	1717468	1797581	3	5283840	5328215		Genus_sps2	14	
Genus_sps1	3	1840205	2082752	1	1340029	2183499	-	Genus_sps2	15	
Genus_sps1	3	2229126	2229953	5	22448139		22448966	Genus_sps2	16	
Genus_sps1	3	2271579	2707482	3	21480577		22401329	-	Genus_sps2	17
Genus_sps1	3	2613766	2634131	5	22452950		22477166	Genus_sps2	18	

The **Synteny_plots** output folder also contains resolution output folders:

```
synteny_plots$ ls
100k  300k  500k
```

In each resolution folder, for each species, two kinds of plots will be generated to visualize synteny results:





(4.2) eba_analysis:

tThe **eba_analysis** output folder has two subfolders: **EBRs** and **msHSBs**. The EBRs output folder contains all the results related to **evolutionary breakpoint regions**. The msHSBs output folder contains all results related to **multispecies homologous synteny blocks**.

```
eba_analysis$ ls
EBRs  msHSBs
```

(4.2.1) EBRs:

This output folder contains a text file (**EBRs_stats.txt**) with statistics for EBRs. This information can be visualized in another HTML file, called **EBRs_stats.html.** A full list of EBRs is available in **EBRs.txt.** The input files for the getENRICH tool for pathway enrichment for all EBRs together and for individual species or nodes are available in folders named after the corresponding species.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs$ head EBRs.txt
1       14920643        15943248        genus_sps2
1       15943249        15950765        genus_sps2
1       15950766        16036332        genus_sps2
1       16036333        16134084        genus_sps2
3       570302  960962  genus_sps3
3       1020232 1139377 genus_sps3
3       1456047 1722468 genus_sps2
3       16264129        16322854        genus_sps2
3       16702089        16731081        genus_sps2
3       17340589        18165376        genus_sps2
```

```
(base) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs$ ls
brkfile2  EBRs_Enrichment_results  EBRs_split_done.txt  EBRs_stats.txt               ebr_stats.log  genus_sps2_lineage
brkfile3  ebrs_processed.marker    EBRs_stats.html      EBRs_subdirs_overlap_done.txt  EBRs.txt       genus_sps3_lineage
(base) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs$ cd EBRs_Enrichment_results/
(base) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs/EBRs_Enrichment_results$ ls
background.txt  EBRs_NCBI_genes_overlap.txt  foreground.txt
(base) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs/EBRs_Enrichment_results$
```

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/EBRs$ cat EBRs_stats.txt
======================================================================
  SYBR Pipeline — EBR Statistics Report
  Input: EBRs.txt
======================================================================

----------------------------------------------------------------------
  GLOBAL SUMMARY
```

(4.2.2) msHSBs:

This output folder contains a list of all msHSB **coordinates** and msHSB **sequences** for the reference genome in **msHSBs.txt** and **msHSB_sequences.fasta. In addition to these,there ate the input files for getENRICH pathway enrichment analysis of msHSBs same as in EBRs**.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/msHSBs$ head msHSBs.txt
1       187435  252567
1       263336  394750
1       1063295 14925643
1       16436853        16475403
1       16500034        16587181
1       16624731        16628324
3       955962  1025232
3       1717468 1797581
3       1840205 1882328
3       1903115 1911559
```

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/eba_analysis/msHSBs$ head msHSBs_sequences.fasta
>1_187435-252567:.
GCCGTATTTATCATGCCCACCCGGCCCGGCCCGAGACCGAAGGATAGTTTATTATAAACA
TTTGCTCTTTTTATTAGGAGTAAAGGATTCTTTTAGTGATAAGCAATGTTTCCAAAAAGC
AAATAGCTAGCATTATGATGTGTCCGGTGGACTTAAATTAAGTGAATAAAACTCTACCAC
TAGAAAATTGTCTGAGTTAAACATGAGTGAACTTAATAAGATGGTATTTCAAAGCATATT
TTGATGTGCTGGTCGTAATCAGACATAACTCATTGTAATATGTCTCTGAATTTTTGGTAA
AATAAGCCTGAACGAAGTGTATCATGTATGTAGATTATATCGTCTTACACTGTAATCGGT
TAATGAAGTAGGACTCAATGAACATTTGTGCAGATGCATGATCAAAAAATGTGGATAACC
ATTTTTTATAGGTTAACTTTAGTCCACTATAGAAACGTATGATTTTTAGATCGCTTTGGC
AATGAGCagtaaaattttcttattgcaTGCGGGCTACTTTCATAAATGAGCTAGcttttc
```

(4.3) Ancestor_seq_recunstruction:

The ancestor_seq_reconstruction output folder contained two subfolders: the **data** folder, which contains the **chain** and **net** files used by the **DESCHRAMBLER** tool for **ancestor seq reconstruction**. And **APCFs. 300K** contains the final reconstruction results.

```
outputs/Ancestor_seq_recunstruction$ ls
APCFs.300K  data
```

Many files are generated in the **APCFs. 300K** directory. Among them, the following files are the most important and useful.

(4.3.1) block_list.txt (in the "SFs" subdirectory)

This file contains the coordinates and identifiers of syntenic fragments (SFs) used in reconstruction.

Column 1: chromosome or scaffold
Column 2: start position (0-based); add 1 to obtain the actual start position
Column 3: end position (1-based)
Column 4: orientation
Column 5: identifier

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K/SFs$ head block_list.txt
chr1    566521  15747356        +       3
chr3    2887155 16277172        +       1
chr5    2406227 16209185        +       2
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K/SFs$
```

(4.3.2) Conserved.Segments (in the "SFs" directory)

This file contains the coordinates of the genomic regions for each species in each SF.

Lines starting with ">": ><identifier of an SF>

Other lines: the coordinates of the genomic regions for each species in the SF. The
The format is as follows.

<species>.<chromosome or scaffold>:<start position>-<end position> <orientation>

Here, the start position is 0-based, and the end position is 1-based. You need to add 1 to the start position to obtain the actual start position.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K/SFs$ head Conserved.Segments
>1
sps1.chr3:2887155-16277172 +
sps2.chr3:7111500-19748262 +
sps3.chr3:955059-14099127 +

>2
sps1.chr5:2406227-16209185 +
sps2.chr5:7720321-20466878 -
sps3.chr5:6699106-13868197 -
sps3.chr5:572601-6385072 -
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K/SFs$
```

(4.3.3) Ancestor.APCF

This file contains the list of ancestral predicted chromosome fragments (APCFs) and the order and orientation of SFs in each APCF.

Line 1: >ANCESTOR <total number of SFs>
Lines starting with "#": # APCF <identifier of APCF>

Other lines: the order and orientation of SFs in the APCF are shown immediately above each line.
SFs are shown using their identifiers (see the block_list.txt file).

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$ head Ancestor.APCF
>ANCESTOR       0
# APCF 1
3 $
# APCF 2
2 $
# APCF 3
1 $
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$
```

(4.3.4) Ancestor.ADJS

This file contains the predicted adjacencies of pairs of SFs and their scores used in the Ancestor.APCF file. SFs are shown using their identifiers (see the block_list.txt file).

Column 1: the first SF identifier
Column 2: the second SF identifier
Column 3: adjacency score of the first and the second identifiers

In the first and second columns, 0 means the end of APCFs. For example, "0 64" means the SF 64 is placed at the end of a current APCF.

(4.3.5) APCF_size.txt

This file contains the total length of APCFs.

Column 1: APCF identifier
Column 2: total length

The last line shows the total APCF length.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$ head APCF_size.txt
1       15180835
2       13802958
3       13390017
total   42373810
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$
```

(4.3.6) APCFs

This file contains the coordinates of genomic regions for each species that match each APCF.

Lines starting with "#": #<identifier of APCF>
The coordinates of genomic regions of each species (grouped by species) follow this line.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$ head APCFs
#1

sps1.chr1:566521-15747356 + [3]

sps2.chr1:3973734-19114999 + [3]

#2

sps1.chr5:2406227-16209185 + [2]

(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$
```

(4.3.7) APCF_<SPC>.merged.map

This file contains the mapping of genomic regions of the SPC species for APCFs. Each mapping is shown by using the following three lines.

><identifier of mapping>
APCF.<identifier of APCF>:<start position in APCF>-<end position in APCF> <orientation>
<SPC>.<chromosome or scaffold>:<start position in SPC>-<end position in SPC> <orientation>

The start position is 0-based, and the end position is 1-based. You need to add 1 to the start position to obtain the actual start position.

```
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$ head APCF_sps1.merged.map
>1
APCF.1:0-15180835 +
sps1.chr1:566521-15747356 +

>2
APCF.2:0-13802958 +
sps1.chr5:2406227-16209185 +

>3
APCF.3:0-13390017 +
(sybr) ajay@ajay-SS400TR-34:~/example/outputs/Ancestor_seq_recunstruction/APCFs.300K$
```