`generate_simulated_samples_byConf.py` – Detailed Guide

Automated generator for GSV-benchmarking bash scripts

1. Purpose

Genomic Sequence Variant (**GSV**) benchmarking evaluates how well different classifier settings distinguish closely related genomes.
This script clusters reference genomes from a target species into GSVs, simulates metagenomic samples from those references, and tests multiple Kraken2 and mSWEEP configurations **to maximise recall and precision**. Off-target genomes from other species in the same family are processed in parallel to measure **false-positive rates**, allowing the pipeline to pinpoint the best combination of:

- Number of GSV clusters
- Kraken2 classifier confidence thresholds
- Count filtration to account for false positive classification

2. Quick Usage

Follow the GSV_benchmarking_tutorial.md document for detailed instructions. In short, you prepare a bunch of files and databases. Then you run the benchmarking script to take that information and create a series of 3 scripts which will make simulated samples and take them through our VARIANT++ classification workflow.

3. What the Script Generates

Bash Script	Stage	Key Tasks
script_build_reads_.sh	Sample generation	• Concatenate target / off-target genomes • Simulate reads with ISS • Merge reads with FLASH • Standardise filenames & headers • Produce one merged FASTQ per iteration
script_kraken_extract_split_.sh	Taxonomic classification	• Classify merged FASTQs at each `--confidence` • Extract target reads with extract_kraken_reads.py • Split merged vs unmerged read sets
script_themisto_msweep_.sh	GSV quantification	• Pseudo-align extracted reads with Themisto • Run mSWEEP (merged & unmerged) to estimate GSV abundances

4. Workflow Breakdown

1. **Parse configuration** (`params.txt`): key/value file with paths, iteration counts, GSV list, Kraken confidences, etc.

- This file can be found in `VARIANT++/bin/database_creation/Benchmarking_code`

2. **Create directories**:

```
<PREFIX>_cat_reads/     # merged FASTQs
<PREFIX>_split_reads/   # unmerged FASTQs
<PREFIX>_merged_reads/  # extracted merged reads
<PREFIX>_results/       # reports and outputs
```

3. **Generate mSWEEP annotation files** (`k_#.msweep.txt`) from the metadata TSV.
4. **Write `script_build_reads`** – loops over `num_GSV_list × num_iters × k_columns`.
   - `numGSV == 0` → simulate off-target samples.
   - `numGSV > 0` → select N GSVs, concatenate genomes per GSV, simulate reads.
   - Once genomes are selected, they are all concatenated => reads are simulated using the Novaseq error profile and the ISS tool based on a random selection of counts based on the `num_reads_options` parameter. Then, we attempt to merge reads using flash. To reduce the number of times we run kraken2 downstream, we modify sample headers to include information about their source and number of reads, then we concatenate the simulated samples for each iteration.
5. **Write `script_kraken_extract_split`** – classify each merged FASTQ for every confidence value, then extract & split reads.
   - These set of scripts take the concatenated sample files and classifies them using kraken2. Then we extract the reads matching the `extract_reads_taxid` parameter and split them up back up into their individual simulated samples for classification with themisto/mSWEEP on the next step.
6. **Write `script_themisto_msweep`** – pseudo-align reads and run mSWEEP (all k-columns for off-target, single column for on-target).
   - This takes each of our individual simulated samples and runs them through themisto and mSWEEP.
   - Off-target simulated samples get processed with all unique ANI cluster annotations.
7. **Finish** – print a summary of the three generated bash scripts.

---

5. Key Parameters (`params.txt`)

Key	Type	Example	Description
`target_genome_dir`	str	`/data/targets/`	`.fna[.gz]` of target species
`nontarget_genome_dir`	str	`/data/offtarget/`	Off-target family genomes
`input_file`	str	`ani_clusters.tsv`	TSV with genome filenames + GSV codes per `k_col`
`bin_dir`	str	`/usr/local/bin/`	Path to helper scripts (`rename_*`, split scripts)
`themisto_index`	str	`/indices/themisto/`	Prefix of existing Themisto index
`krakendb`	str	`/databases/kraken2/`	Kraken2 database directory

Key	Type	Example	Description
`kraken_confidence`	list / int	`[0,0.1,0.5]`	Confidence thresholds to test
`num_iters`	int	`100`	Iterations per `numGSV` setting
`num_GSV_list`	list[int]	`[0,1,3,5,7,9]`	GSV counts to test
`num_reads_options`	list[int]	`[10000,20000]`	Possible read counts per sample
`threads`	int	`48`	CPU threads for all tools
`tmp_build`	str	`$TMPDIR`	Scratch directory (can use environment variable)

6. Expected Outputs

```
<PREFIX>_cat_reads/      # merged FASTQs for Kraken2
<PREFIX>_split_reads/    # unmerged FASTQs for Themisto/mSWEEP
<PREFIX>_merged_reads/   # merged extracted reads by confidence
<PREFIX>_results/        # Kraken reports, Themisto outputs, mSWEEP tables
script_build_reads_<PREFIX>.sh
script_kraken_extract_split_<PREFIX>.sh
script_themisto_msweep_<PREFIX>.sh
```

---

7. Dependencies

- Python ≥3.6
- **ISS** (read simulator)
- **FLASH** (read merger)
- **Kraken2** ≥2.1
- **extract_kraken_reads.py** (KrakenTools)
- **Themisto** ≥2.5
- **mSWEEP** ≥1.9
- Helper scripts in `bin_dir` (`rename_sample_files.py`, `rename_headers.py`, `split_extracted_reads_by_conf.py`)

---

8. Troubleshooting Tips

- **Empty output?** Verify `num_GSV_list` values exist in your TSV.
- **File-not-found errors**: Check all paths in `params.txt`.
- **Kraken2 memory issues**: Add `--memory-mapping` to reduce memory, but request a lot more time.
- **mSWEEP annotation errors**: Ensure each `k_#.msweep.txt` has one entry per genome.
