StrainGE

Lucas van Dijk, Bruce Walker, Tim Straub, Colin Worby, Alexandra

Sep 29, 2023

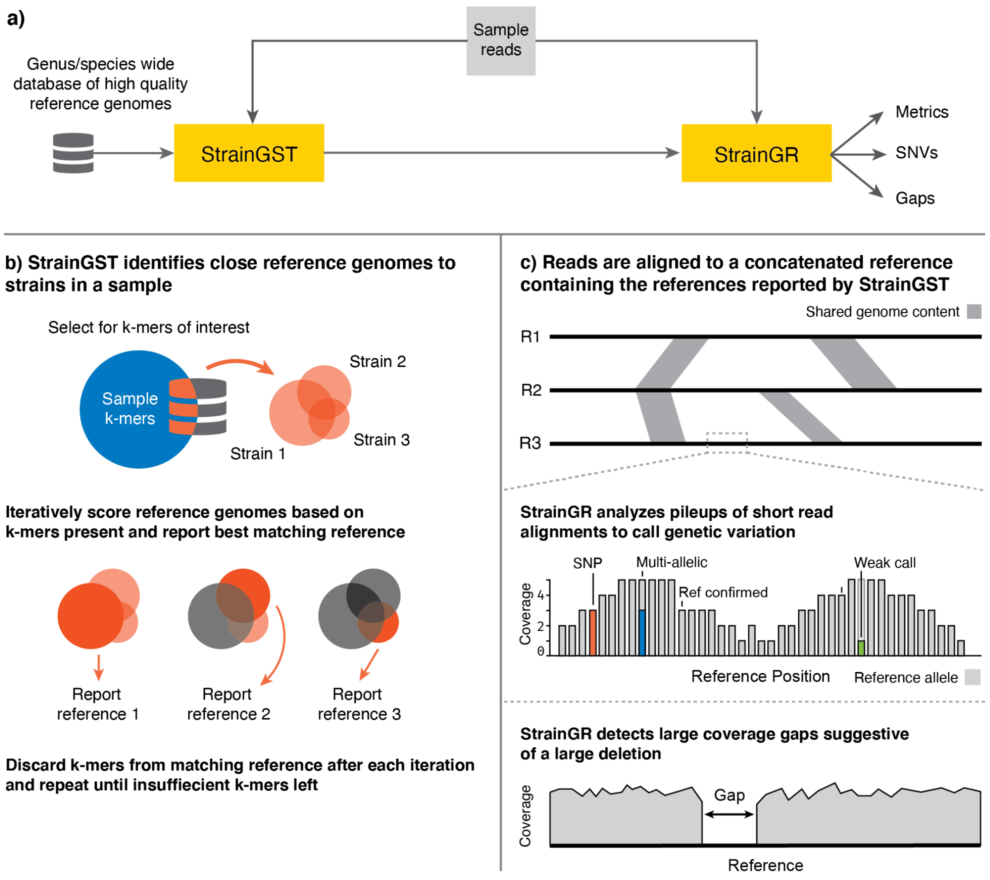
CONTENTS

CHAPTER

ONE

A TOOLKIT TO TRACK AND CHARACTERIZE LOW-ABUNDANCE STRAINS USING METAGENOMIC DATA

StrainGE is a set of tools to analyse conspecific strain diversity in bacterial populations. It consists of two main components: 1) Strain Genome Search tool (StrainGST), a tool to find close reference genomes to strain(s) present in a sample and 2) Strain Genome Recovery (StrainGR), a tool to perform strain-aware variant calling at low coverages, which in turn can be used to track strains across samples.


a)
Sample reads
Genus/species wide database of high quality reference genomes
StrainGST
StrainGR
Metrics
SNVs
Gaps
b) StrainGST identifies close reference genomes to strains in a sample
Select for k-mers of interest
Sample k-mers
Strain 1
Strain 2
Strain 3
Iteratively score reference genomes based on k-mers present and report best matching reference
Report reference 1
Report reference 2
Report reference 3
Discard k-mers from matching reference after each iteration and repeat until insuffiecient k-mers left
c) Reads are aligned to a concatenated reference containing the references reported by StrainGST
Shared genome content
R1
R2
R3
StrainGR analyzes pileups of short read alignments to call genetic variation
SNP
Multi-allelic
Ref confirmed
Weak call
Coverage
0
2
4
Reference Position
Reference allele
StrainGR detects large coverage gaps suggestive of a large deletion
Gap
Coverage
Reference

CHAPTER TWO

INSTALLATION

StrainGE requires Python >= 3.7 and depends on the following packages:

- NumPy
- SciPy
- matplotlib
- scikit-bio
- pysam
- h5py
- intervaltree

These packages will be automatically installed when installing through pip.

2.1 Install through *pip*

```
pip install strainge
```

Make sure *numpy* is already installed before installing StrainGE.

2.2 Install from bioconda

1. Create a new conda environment and activate it

   ```
   conda create -n strainge python=3.9
   source activate strainge
   ```

2. Add bioconda and conda-forge channels

   ```
   conda config --add channels bioconda
   conda config --add channels conda-forge
   ```

3. Install StrainGE

   ```
   conda install strainge
   ```

Tip: also consider installing Mamba for much faster conda operations.

2.3 Install manually from github

1. Clone the repository

   ```
   git clone https://github.com/broadinstitute/StrainGE
   ```

2. Install StrainGE

   ```
   cd StrainGE
   python setup.py install
   ```

CHAPTER

THREE

USAGE

3.1 StrainGST database creation

3.1.1 1. Download high quality reference genomes for your genus/species of interest

This tutorial assumes you have activated the *strainge* conda environment. The first step is to obtain high quality reference genomes for your genus or species of interest, any method suffices. We've found the tool ncbi-genome -download useful, and will use that tool for this step.

For example, to download all *Escherichia* genomes:

```
mkdir ref_genomes
ncbi-genome-download bacteria -l complete -g Escherichia,Shigella -H -F all \
    -o ref_genomes
```

The `-H` flag automatically organizes all downloaded files in a nice human -readable folder structure. Besides downloading references, this command downloads all associated metadata like gene annotations too, which is useful for downstram analyses.

Next, we organize all references in a single directory using a script available in the `bin/` directory of this repository: `prepare_strainge_db.py`. This script serves two main purposes: 1) it organizes all references in a single directory, 2) it optionally splits chromosomes and plasmids into separate files. When tracking strains we're usually more interested in tracking the chromosome, and we don't want StrainGST to report a strain as present because it shares a plasmid (although its algorithm should already prevent most of those cases.)

So download the prepare_strainge_db.py script to your analysis folder, and run it as follows:

```
mkdir strainge_db
python3 prepare_strainge_db.py ref_genomes/human_readable -s \
    -o strainge_db > strainge_db/references_meta.tsv
```

The `-s` flag enables splitting chromosomes and plasmids. The file `references_meta.tsv` contains metadata on each reference (for example its accession no.)

3.1.2 2. K-merize your reference sequences

Next, we k-merize each genome:

```
for f in strainge_db/*.fa.gz; do straingst kmerize -o ${f%.fa.gz}.hdf5 $f; done;
```

These steps can run in parallel, so use your favorite parallelization method if desired (e.g., cluster task array, GNU parallel).

The syntax `${f%.fa.gz}` removes the `.fa.gz` extension from the filename in `$f`, thus the output filename for each kmerset HDF5 will follow the format `REF_NAME.hdf5`. StrainGE will infer the strain name from the HDF5 filename in the steps below, thus by removing the `.fa.gz` extension we remove clutter.

3.1.3 3. Compare the k-mer sets and cluster similar references

The goal of StrainGST is to identify close reference genomes to strains present in a sample. These reference genomes are in turn used for variant calling and sample comparisons. Here lies a trade-off: the reference genome should be close enough for accurate variant calling, but sample comparisons are more easy to perform when the variant calling step is done using the same reference genome, so you don't want to be too specific. Furthermore, limiting the database size reduces computational time. The database of reference genomes should cover the diversity of the species of interest but not contain too many highly similar genomes. Therefore a clustering step is performed to reduce redundancy in the database.

We remove redundant reference genomes two ways:

1. Remove reference genomes that are a near perfect subset of another genome. An example of this is an *E. coli* strain used for synthetic biology applications that was basically a K-12 strain with many genes removed.
2. Cluster closely related genomes based on k-mer similarity and pick one representative.

To do this, we need to compute the pairwise similarities between k-mer sets, and a metric to identify whether a k-mer set is a subset of another. Both can be obtained using `straingst kmersim`.

```
straingst kmersim --all-vs-all -t 4 -S jaccard -S subset strainge_db/*.hdf5 >␣
↪similarities.tsv
```

This command produces as tab separated file, where each line contains a pair of k-mer sets with their accompanying similarity scores. With the `-S` flag we enable which scoring metrics to calculate, and in this case we enable the *Jaccard* similarity and the *subset* score. The output file contains for each pair of k-mer sets the requested scores, sorted by the first scoring metric (in our case the jaccard similarity). With the parameter `-t` you specify the number of processes to spawn, to allow for parallel computation of these pairwise similarities.

We can now cluster our references using the `straingst cluster` command.

```
straingst cluster -i similarities.tsv -d -C 0.99 -c 0.90 \
   --clusters-out clusters.tsv \
    strainge_db/*.hdf5 > references_to_keep.txt
```

The cluster command reads our previously created file `similarities.tsv` to determine which references to keep. The first step is to discard any genome where more than 99% of its kmers are present in another genome, as enabled by `-d` and `-C 0.99`. Afterwards, we cluster similar genomes based on the *Jaccard* similarity between k-mersets: if the Jaccard similarity between two k-mer sets is higher than 0.90 (`-c 0.90`), those two genomes will be clustered together (approximate ANI: ~99.8%). For each cluster we pick one representative genome: the genome with the smallest mean distance to the other cluster members. Each genome to keep is written to `references_to_keep.txt`. With the option `--clusters-out` we specify another file where we write the clustering results. Each line in this file specifies a cluster along with its entries, separated by a tab. The genomes in the first column represent the cluster representatives. This option is optional, but can be useful for debugging purposes.

3.1.4 4. Create pan-genome k-mer database

Using our list of references, we finally create a single database file which will contain all k-mers of the given references.

```
straingst createdb -f references_to_keep.txt -o pan-genome-db.hdf5
```

Now our database lives in the file `pan-genome-db.hdf5`, created from reference sequences read from the file given by `-f`.

It is also possible to give the list of k-mer sets to include in the database as positional arguments, like in the following example:

```
straingst createdb -o pan-genome-db.hdf5 ref1.hdf5 ref2.hdf5 ...
```

Combining the two methods described above works too.

3.2 Running StrainGST

Identify close reference genome(s) to strain(s) in a sample.

3.2.1 Prerequisites

1. A pre-built database for the genus or species of interest
2. A whole metagenomic sequencing (WMS) sample

3.2.2 Usage

1. K-merize the sample reads

StrainGST iteratively compares the k-mer profiles of references in the database to the k-mers in the sample to identify close reference genomes to strains in a sample.

Our first step is to kmerize the sample reads. For example, if you have a FASTQ file named `patient1.fastq` with all reads, then we generate its corresponding k-mer set as follows:

```
straingst kmerize -k 23 -o patient1.hdf5 patient1.fastq
```

Similar to the first step of the database creation section, this will generate a HDF5 file named `patient1.hdf5` with all k-mers and their corresponding counts. Make sure the value of `k` is the same as used in the database creation step.

You can specify multiple FASTQ files to the command above, which is useful if you have paired-end reads. Furthermore, it will also automatically decompress *gzipped* files. For example, if you have gzipped paired-end FASTQ files, then the following command also works:

```
straingst kmerize -k 23 -o patient1.hdf5 \
    patient1.1.fastq.gz patient1.2.fastq.gz
```

2. Run StrainGST

We can now run `straingst run` with our database HDF5 and our sample HDF5:

```
straingst run -o results.tsv pan-genome-db.hdf5 patient1.hdf5
```

This will output a *tab separated values* (tsv) file, containing statistics about the sample k-mer set and a list of identified reference strains with accompanying metrics.

New in version 1.3: instead of writing both sample statistics and the identified strains to a single TSV file, which is generally not as easily read in Python's `pandas` or R, you can now enable the option to write sample statistics and strains to separate files when enabling the `--separate-output` (`-O`) option. If enabling this option, use `-o` to specify the output filename prefix.

Example:

```
straingst run -O -o PREFIX pan-genome-df.hdf5 patient1.hdf5
```

This will result in two files: `PREFIX.stats.tsv` (sample statistics), and `PREFIX.strains.tsv` (list of identified strains).

3.2.3 Output file description

Example output (single file output)

```
sample    totalkmers      distinct   pkmers  pkcov   pan%
UMB11_01  2277023860      380759656  50090   6.984   1.536
i         strain          gkmers     ikmers  skmers  cov    kcov   gcov   acct   even  ␣
↪spec   rapct  wscore  score
0         Esch_coli_NGF1  49631      49622   50090   0.985  7.009  6.831  0.980  0.987 ␣
↪1.000  1.507  0.940   0.940
```

Example output (separate file output; new in version 1.3)

`PREFIX.stats.tsv`

```
sample    totalkmers      distinct   pkmers  pkcov   pan%
UMB11_01  2277023860      380759656  50090   6.984   1.536
```

`PREFIX.strains.tsv`

```
i         strain          gkmers     ikmers  skmers  cov    kcov   gcov   acct   even  ␣
↪spec   rapct  wscore  score
0         Esch_coli_NGF1  49631      49622   50090   0.985  7.009  6.831  0.980  0.987 ␣
↪1.000  1.507  0.940   0.940
```

Sample statistics

The first two lines contain statistics on the whole sample.

Columns:

- *sample*: Sample name, derived from the k-mer set filename
- *totalkmers*: total number of k-mers counted in the sample, including k-mers that occur multiple times
- *distinct*: total *unique* number of k-mers
- *pkmers*: total unique number of k-mers that are also present in the database
- *pkcov*: average "coverage" (multiplicity) of each unique k-mer in the sample that is also present in the database.
- *pan%*: total number of k-mers (including duplicates) that are present in both the sample and database divided by the total number of k-mers in the sample (*totalkmers*), i.e. an estimation of the relative abundance of the species/genus of interest in this sample.

Reference strain statistics

The next lines contain the close reference genomes identified by StrainGST.

Columns:

- *i*: Iteration number
- *strain*: Reference strain name
- *gkmers*: Total number of unique k-mers in the original reference genome (or its fingerprint).
- *ikmers*: Remaining unique k-mers in the genome after discarding k-mers excluded in an earlier iteration or because their average coverage was too high
- *skmers*: Remaining unique k-mers from the sample
- *cov*: Breadth of coverage of this reference, i.e. what fraction of k-mers in the reference is also present in the sample
- *kcov*: Average depth of coverage of k-mers both present in the reference and in the sample
- *gcov*: Average depth of coverage of *all* k-mers in the reference
- *acct*: What fraction of the sample k-mers can be explained by this reference?
- *even*: Evenness of coverage. A value close to 1 indicates that the coverage is evenly distributed along the genome, a value close to zero indicates that only a small part of the genome is well covered.
- *spec*: Obsolete
- *rapct*: Estimated strain relative abundance (relative to the whole sample).
- *wscore*: Obsolete
- *score*: Score used to rank each reference in the database at each iteration. A high score represents high confidence in this prediction. Scores cannot be compared across iterations or across samples, and it is possible that a strain in a second iteration has a higher score than the strain in the first iteration.

3.2.4 Tips and Tricks

Easily parse StrainGST file in Python (mainly useful for single file output):

```python
from strainge.io.utils import parse_straingst

results = ['sample1.tsv', 'sample2.tsv']

for sample in results:
    print('#', sample)
    with open(sample) as f:
        for strain in parse_straingst(f):
            print(strain)  # strain is a dict with above columns
```

With sample statistics:

```python
from strainge.io.utils import parse_straingst

results = ['sample1.tsv', 'sample2.tsv']

for sample in results:
    print('#', sample)
    with open(sample) as f:
        straingst_iter = iter(parse_straingst(f, return_sample_stats=True))
        sample_stats = next(straingst_iter)
        print(sample_stats)

        for strain in straingst_iter:
            print(strain)  # strain is a dict with above columns
```

3.3 Running StrainGR

Characterize strains in a metagenomic sample.

3.3.1 Prerequisites

- StrainGST results on one or more samples
- Directory containing the reference genomes used to create the StrainGST database
- BWA-MEM
- Mummer4
- Optional: Picard (for MarkDuplicates)

3.3.2 Usage

1. Prepare a concatenated reference FASTA with `straingr prepare-ref`

Our strategy to deconvolve strains in a mixture sample is to create a FASTA containing a close reference genome for each strain present in a sample, and then aligning the sample reads to this concatenated FASTA file. StrainGR provides a tool `straingr prepare-ref` to automatically create and analyze a concatenated reference genome from a list of StrainGST result files.

By including multiple reference genomes into a single FASTA file, reads with an allele specific to a strain will be placed to the optimal location in the concatenated reference. On the other hand, the reference genomes included in the concatenated reference may share (conserved) parts of their genome because they are the same species, and an aligner will be unable to unambiguously place reads in those regions. This is a trade-off: include as many reference genomes as required to deconvolve strains in a sample, without combining too closely related reference genomes such that they share a vast chunk of their genomes. StrainGR will not call variants in shared regions.

The `prepare-ref` subcommand aids in building a concatenated reference from StrainGST result files. It determines which strains have been reported by StrainGST, and performs another clustering step on the reported strains to ensure the included reference strains are not too closely related. For example, sometimes it happens that a patient has a strain that's somewhat in the middle between two reference genomes sitting next to each other on the tree. Due to stochasticity in sequencing, StrainGST may report one reference genome in one sample, while reporting the other reference in another sample with the same strain, but taken at a different time point. Here the clustering step ensures that only one of these two closely related strains gets included in the concatenated reference.

After concatenating the selected references, `prepare-ref` runs `nucmer` from the MUMmer toolkit to estimate how "repetitive" the concatenated reference is, i.e. how much sequence do the genomes concatenated share, by computing maximal exact matches of at least a configurable size within the concatenated reference. By default the minimum exact match size is 250 bp, but its recommended to change this value to the average insert size of the sample read set to most accurately estimate the actual repetitiveness. These estimates are used to normalize strain abundances in a later step.

To create a concatenated reference, use `straingr prepare-ref` as follows:

```
straingr prepare-ref -s path/to/straingst/*.tsv \
   -p "path/to/refdir/{ref}.fa.gz" \
   -S path/to/straingst_db/similarities.tsv
   -o refs_concat.fasta
```

We give multiple StrainGST TSV result files to `prepare-ref` with the `-s` flag. Usually these are all StrainGST results file belonging to a single patient, or an other related set of samples. Next, we need to specify how `prepare-ref` can find the actual FASTA files belonging to strains reported by StrainGST, this is done using the "path-template" switch `-p`: in this given path "{ref}" will be replaced **by StrainGR** (so don't replace it yourself) with the actual strain name. Don't forgot to use quotes, because { and } are special characters in many shells. We specify the similarities.tsv file created at the StrainGST database construction step, to reuse the calculated k-mer similarities again for clustering. The resulting concatenated reference will be written to `refs_concat.fasta`.

New in version 1.3: If you use the new split StrainGST output format introduced in version 1.3, only specify the files listing the predicted strains. So, replace `straigr prepare-ref -s path/to/straingst/*.tsv ...` with `straingr prepare-ref -s path/to/straingst/*.strains.tsv ...`.

2. Align reads to the reference

StrainGR is built to be used with bwa mem, as it uses the supplied information on alternative alignment locations encoded in the XA SAM tag to deal with shared regions introduced by concatenating reference genomes.

The following command aligns the reads with bwa mem and outputs a sorted BAM file:

```
bwa mem -I 300 -t 2 refs_concat.fasta sample1.1.fq.gz sample1.2.fq.gz \
    | samtools sort -@ 2 -O BAM -o sample1.bam -

# Also create BAM index
samtools index sample1.bam
```

We specify a fixed insert size to bwa mem, because if the species of interest in a metagenomic sample is at low abundance, there may be not enough reads per batch for bwa mem to infer the mean insert size, and reads in such a batch will be marked as improperly paired. Optionally you can run picard MarkDuplicates on your alignment file.

3. Analyze read alignments to call variants

To call any variants in your sample run the StrainGR variant caller:

```
straingr call refs_concat.fasta sample1.bam --hdf5-out sample1.hdf5 --summary sample1.
↪tsv --tracks all
```

All variant calling data will be stored in the given HDF5 file sample1.hdf5. A table with summary statistics like coverage, SNP rate, gaps and more is written to sample1.tsv. You can also specify to output this table to a TSV file with the -s switch in the above command. There are more options for data output, it can output VCF files, BED tracks and more, see the CLI reference documentation below.

You can recreate many of the additional data files from the HDF5 file using straingr view.

3.3.3 Output files

StrainGR summary

Example output

```
ref                                                 name           length    coverage ␣
↪uReads  abundance  median  callable  callablePct  confirmed  confirmedPct  snps ␣
↪snpPct  multi  multiPct  lowmq    lowmqPct  high    highPct  gapCount  gapLength
Esch_coli_H3                                        NZ_CP010167.1  4630919   0.247     18    ␣
↪   0.000      0       281       0.006       275         97.865       6      2.135    0  ␣
↪    0.000     308810   6.668     21045    0.454    13          447553
Esch_coli_H3                                        NZ_CP010168.1  48243     0.025     0     ␣
↪   0.000      0       0         0.000       0           0.000        0      0.000    0  ␣
↪    0.000     263      0.545     0        0.000    0           0
Esch_coli_NGF1                                      NZ_CP016007.1  5026105   3.549     85824␣
↪   0.823      3       2506998   49.880      2506921     99.997       77     0.003    70 ␣
↪    0.003     1668501  33.197    3681     0.073    1           16868
Esch_coli_NGF1                                      NZ_CP016008.1  40158     6.942     2458 ␣
↪   0.008      7       39096     97.355      39094       99.995       2      0.005    2  ␣
↪    0.005     982      2.445     12       0.030    0           0
```

(continues on next page)

(continued from previous page)

```
Esch_coli_NGF1                                        NZ_CP016009.1  8556      0.000     0    ␣
↪  0.000      0       0         0.000        0           0.000        0     0.000   0  ␣
↪   0.000     0        0.000     0        0.000    1          8556
Esch_coli_clone_D_i14                                 NC_017652.1    5038386   1.341     210  ␣
↪  0.002      0       5022      0.100        5018        99.920       4     0.080   0  ␣
↪   0.000     1792289  35.573    196601   3.902    30         575694
Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  NZ_LR536430.1  4975029   0.224     49   ␣
↪  0.000      0       548       0.011        548         100.000      0     0.000   0  ␣
↪   0.000     298901   6.008     18373    0.369    16         767577
Esch_coli_1190                                        NZ_CP023386.1  4900891   0.260     24   ␣
↪  0.000      0       351       0.007        334         95.157       17    4.843   0  ␣
↪   0.000     342936   6.997     24117    0.492    25         848801
Esch_coli_1190                                        NZ_CP023387.1  86147     0.000     0    ␣
↪  0.000      0       0         0.000        0           0.000        0     0.000   0  ␣
↪   0.000     0        0.000     0        0.000    1          86147
TOTAL                                                 -              24754434  1.148     88583␣
↪  0.834      0       2552296   10.310       2552190     99.996       106   0.004   72 ␣
↪   0.003     4412682  17.826    263829   1.066    87         2751196
```

Column descriptions

For each scaffold in the concatenated reference StrainGR outputs several metrics.

- *ref*: original reference genome this scaffold belongs to
- *name*: Scaffold name
- *length*: Scaffold length
- *coverage*: Average depth of coverage along this scaffold. **includes multimapped reads, and multimapped reads are counted multiple times (for each alternative alignment location)**
- *uReads*: Number of reads uniquely aligned to this scaffold
- *abundance*: Estimated relative abundance of this scaffold. Calculated by dividing the number uniquely aligned reads to this scaffold by the total number of reads uniquely aligned, normalized by the estimated repetitiveness in the `prepare-ref` stage. Generally, we trust the abundances calculated by StrainGST a lot more.
- *median*: median depth of coverage
- *callable* (*callablePct*): Number (percentage) of positions in this scaffold with *strong* evidence for an allele (i.e. two good reads supporting a single allele)
- *confirmed* (*confirmedPct*): Number (percentage) of positions where there's *strong* evidence for the reference allele (does not exclude positions with multiple alleles).
- *snps* (*snpPct*): Number (percentage) of positions with strong evidence for a **single** allele **different** than the reference. Our best estimate of ANI.
- *multi* (*multiPct*): Number (percentage) of positions with strong evidence for **multiple alleles** (whether it includes the reference or not).
- *lowmq* (*lowmqPct*): Number (percentage) of positions where the majority of reads are mapped with low mapping quality, i.e. representing shared or repetitive regions.
- *high* (*highPct*): Number (percentage) of positions with abnormally high coverage.
- *gapCount*: Number of gaps predicted

- *gapLength*: Number of positions in the genome marked as gap

3.4 Comparing strains across samples

3.4.1 Prerequisites

- StrainGR call data (HDF5 files) for the samples of interest

3.4.2 Comparing strains in different samples

Strains in different samples that match the same close reference genome can be compared in more detail (at the nucleotide level) using StrainGR.

To compare strains run `straingr compare`:

```
straingr compare sample1.hdf5 sample2.hdf5 \
   -o sample1.vs.sample2.summary.tsv -d sample1.vs.sample2.details.tsv
```

`straingr compare` takes in two HDF5 files as generated by `straingr call`, and the compares the base calls in each sample for each scaffold in the concatenated reference. If different concatenated references were used for each sample, only the scaffolds the two concatenated references have in common will be compared.

3.4.3 Output file description

Summary TSV

This file contains several metrics that summarizes the comparisons of each strain (scaffold).

Warning: this file currently contains a ton of metrics, several of which are slight variations on others. In the final version of StrainGE we will likely remove a few and only keep the most relevant ones.

Columns:

- *sample1, sample2*: Sample names (from filename)
- *ref*: The name of the original reference this scaffold belongs to
- *scaffold*: scaffold name
- *length*: length of the scaffold
- *common* (*commonPct*): Number (percentage) of positions of this scaffold that's callable in both samples
- *single* (*singlePct*): Number (percentage) of positions where both samples have a single strong call (i.e. no evidence for multiple alleles)
- *singleAgree* (*singleAgreePct*): Number (percentage) of positions where both sample have single strong call, and the base call is the same. *singleAgreePct* is the *ACNI* metric as described in the paper.
- *sharedAlleles* (*sharedAllelesPct*): Number (percentage) of positions where both samples share an allele. This allows for positions to have multiple alleles, and at least one allele should match.
- *variants* (*variantsPct*): Number (percentage) of positions where either sample has an allele other than the reference.
- *commonVariant* (*commonVariantPct*): Number (percentage) of variants where both samples share an allele

- *variantExact* (*variantExactPct*): Number (percentage) of variants that are exactly the same in both samples (including the same positions with multiple alleles).
- *AnotB* (*AnotBPct*): Number (percentage) of variants in Sample A but not in Sample B
- *BnotA* (*BnotAPct*): Number (percentage) of variants in Sample B but not in Sample A
- *gapJaccardSimilarity*: Jaccard similarity between samples of set of positions **not** marked as gap (i.e. analogous to gene content similarity).

3.5 Analyze StrainGE output in Python

Now that we have run StrainGST and StrainGR (including the compare step), how do we analyze the outputs? This page uses Python and its commonly used data science stack (NumPy, SciPy, Pandas and matplotlib+seaborn) to parse the data, plot the relative abundances of strains over time, and generate an ACNI/gap similarity plot.

3.5.1 Download data

We download an archive containing StrainGE outputs part of the vignette described in the paper on the persistence of an *E. coli* strain in the gut of a woman with recurrent urinary tract infections. The extracted data is organized in a `straingst` and `straingr` folder.

[2]:
```
!curl --output umb_data.tar.gz https://raw.githubusercontent.com/broadinstitute/strainge-
↪paper/master/umb/umb_data.tar.gz
!tar -xzvf umb_data.tar.gz
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 12196  100 12196    0     0  42347      0 --:--:-- --:--:-- --:--:-- 42347
x straingst/UMB11_01.tsv
x straingst/UMB11_02.tsv
x straingst/UMB11_03.1.tsv
x straingst/UMB11_03.tsv
x straingst/UMB11_04.1.tsv
x straingst/UMB11_04.tsv
x straingst/UMB11_05.tsv
x straingst/UMB11_06.tsv
x straingst/UMB11_07.tsv
x straingst/UMB11_08.tsv
x straingst/UMB11_11.tsv
x straingst/UMB11_12.tsv
x straingr/UMB11_01.tsv
x straingr/UMB11_02.tsv
x straingr/UMB11_03.1.tsv
x straingr/UMB11_03.tsv
x straingr/UMB11_04.1.tsv
x straingr/UMB11_04.tsv
x straingr/UMB11_05.tsv
x straingr/UMB11_06.tsv
x straingr/UMB11_07.tsv
x straingr/UMB11_08.tsv
x straingr/UMB11_11.tsv
```

(continues on next page)

(continued from previous page)

```
x straingr/UMB11_12.tsv
x straingr/compare.summary.chrom.txt
```

3.5.2 Import required modules

[3]:
```
from pathlib import Path

import numpy
import pandas
import matplotlib.pyplot as plt
from IPython.display import display
```

3.5.3 StrainGST

Read StrainGST outputs and combine it in a DataFrame

The TSV files written by StrainGST contain both sample statistics (the first two lines), and statistics for each identified strain (see *StrainGST* page). In this tutorial, we are mainly interested in the identified strains. In the code below, when calling `pandas.read_csv`, we give the argument `skiprows=2` to skip the sample statistics.

[16]:
```
STRAINGST_DIR = Path("straingst/")

df_list = []
sample_names = []
for f in STRAINGST_DIR.glob("*.tsv"):
    sample_name = f.stem
    df = pandas.read_csv(f, sep='\t', comment='#', skiprows=2, index_col=1)

    df_list.append(df)
    sample_names.append(sample_name)


# Combine all StrainGST results from each sample into a single DataFrame.
straingst_df = pandas.concat(df_list, keys=sample_names, names=["sample"])

sample_names = list(sorted(sample_names, key=lambda e: float(e.replace("UMB11_", ""))))
straingst_df.sort_index()
```

[16]:
```
                                                          i gkmers ikmers  \
sample     strain
UMB11_01   Esch_coli_NGF1                                 0  49631  49622
UMB11_02   Esch_coli_NGF1                                 0  49631  49623
UMB11_03   Esch_coli_1190                                 0  48261  48249
UMB11_03.1 Esch_coli_1190                                 0  48261  48254
UMB11_04.1 Esch_coli_1190                                 0  48261  48250
UMB11_05   Esch_coli_1190                                 0  48261  48248
UMB11_06   Esch_coli_1190                                 1  48261  21600
           Esch_coli_H3                                   0  45610  45560
```

(continues on next page)

(continued from previous page)

```
UMB11_07   Esch_coli_1190                               0  48261  48237
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  1  47727  21265
UMB11_08   Esch_coli_1190                               0  48261  48248
UMB11_11   Esch_coli_1190                               0  48261  48248
UMB11_12   Esch_coli_1190                               0  48261  48235
           Esch_coli_26561                              1  46249  19738

                                                       skmers    cov  \
sample     strain
UMB11_01   Esch_coli_NGF1                               50090  0.985
UMB11_02   Esch_coli_NGF1                                5358  0.103
UMB11_03   Esch_coli_1190                               37144  0.711
UMB11_03.1 Esch_coli_1190                               31201  0.595
UMB11_04.1 Esch_coli_1190                               19042  0.362
UMB11_05   Esch_coli_1190                               40411  0.775
UMB11_06   Esch_coli_1190                               30714  0.794
           Esch_coli_H3                                 74449  0.960
UMB11_07   Esch_coli_1190                               58276  0.854
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  17074  0.441
UMB11_08   Esch_coli_1190                               31599  0.592
UMB11_11   Esch_coli_1190                               49462  0.920
UMB11_12   Esch_coli_1190                               66509  0.941
           Esch_coli_26561                              21112  0.853

                                                           kcov     gcov  \
sample     strain
UMB11_01   Esch_coli_NGF1                                 7.009    6.831
UMB11_02   Esch_coli_NGF1                                 1.546    0.158
UMB11_03   Esch_coli_1190                                 2.814    1.975
UMB11_03.1 Esch_coli_1190                                 2.152    1.264
UMB11_04.1 Esch_coli_1190                                 1.870    0.668
UMB11_05   Esch_coli_1190                                 2.741    2.097
UMB11_06   Esch_coli_1190                                 7.102    5.565
           Esch_coli_H3                                 114.343  109.038
UMB11_07   Esch_coli_1190                                 4.557    3.846
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8    2.588    1.077
UMB11_08   Esch_coli_1190                                 2.243    1.309
UMB11_11   Esch_coli_1190                                 6.107    5.545
UMB11_12   Esch_coli_1190                                 9.061    8.431
           Esch_coli_26561                                4.773    3.983

                                                         acct   even  \
sample     strain
UMB11_01   Esch_coli_NGF1                               0.980  0.987
UMB11_02   Esch_coli_NGF1                               0.932  0.707
UMB11_03   Esch_coli_1190                               0.926  0.826
UMB11_03.1 Esch_coli_1190                               0.918  0.830
UMB11_04.1 Esch_coli_1190                               0.908  0.743
UMB11_05   Esch_coli_1190                               0.923  0.884
UMB11_06   Esch_coli_1190                               0.283  0.797
           Esch_coli_H3                                 0.921  0.960
UMB11_07   Esch_coli_1190                               0.803  0.873
```

(continues on next page)

(continued from previous page)

```
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  0.526  0.668
UMB11_08   Esch_coli_1190                                  0.904  0.810
UMB11_11   Esch_coli_1190                                  0.920  0.923
UMB11_12   Esch_coli_1190                                  0.800  0.941
           Esch_coli_26561                                 0.780  0.869

                                                            spec   rapct  \
sample     strain
UMB11_01   Esch_coli_NGF1                                  1.000   1.536
UMB11_02   Esch_coli_NGF1                                  1.032   0.048
UMB11_03   Esch_coli_1190                                  0.972   0.395
UMB11_03.1 Esch_coli_1190                                  1.030   0.711
UMB11_04.1 Esch_coli_1190                                  1.047   0.135
UMB11_05   Esch_coli_1190                                  0.967   0.484
UMB11_06   Esch_coli_1190                                  0.552   1.005
           Esch_coli_H3                                    0.976  16.420
UMB11_07   Esch_coli_1190                                  0.794   0.411
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  1.012   0.613
UMB11_08   Esch_coli_1190                                  0.980   0.315
UMB11_11   Esch_coli_1190                                  0.965   1.180
UMB11_12   Esch_coli_1190                                  0.734   0.978
           Esch_coli_26561                                 0.968   1.548

                                                           old_rapct  wscore  \
sample     strain
UMB11_01   Esch_coli_NGF1                                      1.505   0.940
UMB11_02   Esch_coli_NGF1                                      0.045   0.047
UMB11_03   Esch_coli_1190                                      0.366   0.437
UMB11_03.1 Esch_coli_1190                                      0.652   0.365
UMB11_04.1 Esch_coli_1190                                      0.122   0.173
UMB11_05   Esch_coli_1190                                      0.447   0.540
UMB11_06   Esch_coli_1190                                      0.391   0.079
           Esch_coli_H3                                       16.042   0.795
UMB11_07   Esch_coli_1190                                      0.822   0.415
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8      0.106   0.102
UMB11_08   Esch_coli_1190                                      0.285   0.344
UMB11_11   Esch_coli_1190                                      1.086   0.696
UMB11_12   Esch_coli_1190                                      2.020   0.489
           Esch_coli_26561                                     0.395   0.486

                                                           score
sample     strain
UMB11_01   Esch_coli_NGF1                                  0.940
UMB11_02   Esch_coli_NGF1                                  0.048
UMB11_03   Esch_coli_1190                                  0.449
UMB11_03.1 Esch_coli_1190                                  0.376
UMB11_04.1 Esch_coli_1190                                  0.182
UMB11_05   Esch_coli_1190                                  0.558
UMB11_06   Esch_coli_1190                                  0.142
           Esch_coli_H3                                    0.814
UMB11_07   Esch_coli_1190                                  0.522
           Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  0.103
```

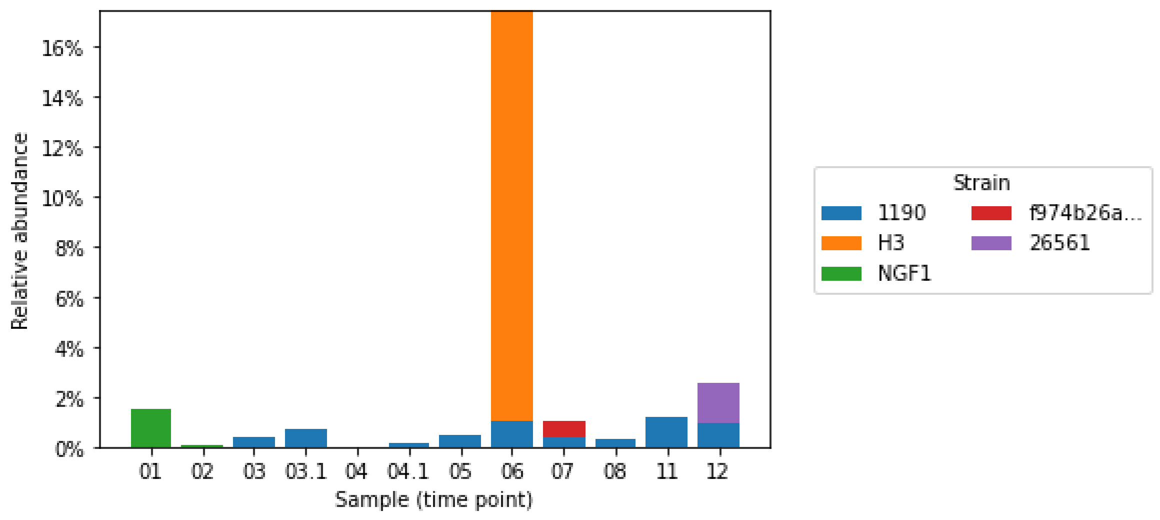
(continues on next page)

(continued from previous page)

```
UMB11_08    Esch_coli_1190                                              0.351
UMB11_11    Esch_coli_1190                                              0.721
UMB11_12    Esch_coli_1190                                              0.667
            Esch_coli_26561                                             0.502
```

Plot relative abundances

```
[5]: plt.figure(figsize=(6, 4))

strain_order = ['Esch_coli_1190', 'Esch_coli_H3', 'Esch_coli_NGF1', 'Esch_coli_f974b26a-
↪5e81-11e8-bf7f-3c4a9275d6c8', "Esch_coli_26561"]
strain_labels = ['1190', 'H3', 'NGF1', 'f974b26a...', "26561"]
xlabels = [s.replace("UMB11_", "") for s in sample_names]

x = numpy.arange(len(sample_names))
bottom = numpy.zeros((len(sample_names),))
for ref, label in zip(strain_order, strain_labels):
    # Create an array with all relative abundances for the current reference in each␣
↪sample. If not available, set to zero.
    rel_abun = numpy.array([
        straingst_df.loc[(sample, ref), 'rapct'] if (sample, ref) in straingst_df.index␣
↪else 0.0
        for sample in sample_names
    ])

    plt.bar(x, rel_abun, bottom=bottom, tick_label=xlabels, label=label, width=0.8)
    bottom += rel_abun

plt.xlabel("Sample (time point)")
plt.ylabel("Relative abundance")
plt.gca().yaxis.set_major_formatter("{x:g}%")
plt.legend(title="Strain", loc="center left", bbox_to_anchor=(1.05, 0.5), ncol=2)

plt.show()
```



3.5.4 StrainGR

Load `call` data in a DataFrame

To load StrainGR outputs, we use a similar approach as descried above. In this case, the StrainGR TSV files can be directly loaded with pandas without `skiprows`.

One thing to note, StrainGR outputs metrics for every contig in the concatenated reference used for alignment. The output file thus contains metrics for **contigs from strains that were not predicted to be present by StrainGST**. We use the presence/absence predictions of StrainGST as our "truth" and remove the contigs from strains that weren't present.

We apply a few other filters, including removing plasmid contigs, and contigs with less coverage than 0.5x.

[28]:
```
STRAINGR_DIR = Path("straingr/")

df_list = []
sample_names = []
for f in STRAINGR_DIR.glob("*.tsv"):
    df = pandas.read_csv(f, sep='\t', index_col=0)
    df = df.drop(index='TOTAL')  # Remove TOTAL statistics

    df_list.append(df)
    sample_names.append(f.stem)

straingr_df = pandas.concat(df_list, keys=sample_names, names=["sample"])
straingr_df['straingst_present'] = straingr_df.index.map(lambda ix: ix in straingst_df.
↪index)
straingr_df['is_plasmid'] = straingr_df['length'] < 4e6
straingr_df['enough_cov'] = straingr_df['coverage'] > 0.5

# Filter and re-index
straingr_df = straingr_df[straingr_df['straingst_present'] & ~straingr_df['is_plasmid'] &
↪ straingr_df['enough_cov']].reset_index().set_index(['sample', 'ref'])
straingr_df
```

[28]:
```
                                                                  name  \
sample     ref
UMB11_11   Esch_coli_1190                                     NZ_CP023386.1
UMB11_06   Esch_coli_H3                                       NZ_CP010167.1
           Esch_coli_1190                                     NZ_CP023386.1
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  NZ_LR536430.1
           Esch_coli_1190                                     NZ_CP023386.1
UMB11_03   Esch_coli_1190                                     NZ_CP023386.1
UMB11_01   Esch_coli_NGF1                                     NZ_CP016007.1
UMB11_03.1 Esch_coli_1190                                     NZ_CP023386.1
UMB11_08   Esch_coli_1190                                     NZ_CP023386.1


                                                            length  coverage  \
sample     ref
UMB11_11   Esch_coli_1190                                  4900891     2.492
UMB11_06   Esch_coli_H3                                    4630919    47.519
           Esch_coli_1190                                  4900891     1.963
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  4975029     0.700
           Esch_coli_1190                                  4900891     1.591
```
(continues on next page)

(continued from previous page)

```
UMB11_03   Esch_coli_1190                                  4900891     0.822
UMB11_01   Esch_coli_NGF1                                  5026105     3.549
UMB11_03.1 Esch_coli_1190                                  4900891     0.596
UMB11_08   Esch_coli_1190                                  4900891     0.580

                                                             uReads  abundance  \
sample     ref
UMB11_11   Esch_coli_1190                                    106232      0.449
UMB11_06   Esch_coli_H3                                     1331869      7.902
           Esch_coli_1190                                     69465      0.264
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8     14686      0.132
           Esch_coli_1190                                     59112      0.347
UMB11_03   Esch_coli_1190                                     35131      0.143
UMB11_01   Esch_coli_NGF1                                     85824      0.823
UMB11_03.1 Esch_coli_1190                                     24708      0.278
UMB11_08   Esch_coli_1190                                     24145      0.118

                                                            median  callable  \
sample     ref
UMB11_11   Esch_coli_1190                                        2   2819899
UMB11_06   Esch_coli_H3                                         48   3863102
           Esch_coli_1190                                        2   1515111
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8        0    378975
           Esch_coli_1190                                        1   1894740
UMB11_03   Esch_coli_1190                                        1    859998
UMB11_01   Esch_coli_NGF1                                        3   2506998
UMB11_03.1 Esch_coli_1190                                        0    547015
UMB11_08   Esch_coli_1190                                        0    505713

                                                            callablePct  \
sample     ref
UMB11_11   Esch_coli_1190                                        57.538
UMB11_06   Esch_coli_H3                                          83.420
           Esch_coli_1190                                        30.915
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8         7.618
           Esch_coli_1190                                        38.661
UMB11_03   Esch_coli_1190                                        17.548
UMB11_01   Esch_coli_NGF1                                        49.880
UMB11_03.1 Esch_coli_1190                                        11.162
UMB11_08   Esch_coli_1190                                        10.319

                                                            confirmed  \
sample     ref
UMB11_11   Esch_coli_1190                                     2818902
UMB11_06   Esch_coli_H3                                       3861892
           Esch_coli_1190                                     1513886
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8      377743
           Esch_coli_1190                                     1893628
UMB11_03   Esch_coli_1190                                      859671
UMB11_01   Esch_coli_NGF1                                     2506921
UMB11_03.1 Esch_coli_1190                                      546816
UMB11_08   Esch_coli_1190                                      505494
```

(continues on next page)

(continued from previous page)

```
                                                     confirmedPct  ...  \
sample     ref                                                     ...
UMB11_11   Esch_coli_1190                                  99.965  ...
UMB11_06   Esch_coli_H3                                    99.969  ...
           Esch_coli_1190                                  99.919  ...
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8       99.675  ...
           Esch_coli_1190                                  99.941  ...
UMB11_03   Esch_coli_1190                                  99.962  ...
UMB11_01   Esch_coli_NGF1                                  99.997  ...
UMB11_03.1 Esch_coli_1190                                  99.964  ...
UMB11_08   Esch_coli_1190                                  99.957  ...

                                                     multiPct    lowmq  \
sample     ref
UMB11_11   Esch_coli_1190                               0.004   435239
UMB11_06   Esch_coli_H3                                 0.004  1439048
           Esch_coli_1190                               0.015  1161951
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8    0.013   495996
           Esch_coli_1190                               0.010   323110
UMB11_03   Esch_coli_1190                               0.001   160054
UMB11_01   Esch_coli_NGF1                               0.003  1668501
UMB11_03.1 Esch_coli_1190                               0.001    99001
UMB11_08   Esch_coli_1190                               0.001   115822

                                                     lowmqPct    high  \
sample     ref
UMB11_11   Esch_coli_1190                               8.881     488
UMB11_06   Esch_coli_H3                                31.075   84996
           Esch_coli_1190                              23.709  699045
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8    9.970   17165
           Esch_coli_1190                               6.593    3384
UMB11_03   Esch_coli_1190                               3.266      26
UMB11_01   Esch_coli_NGF1                              33.197    3681
UMB11_03.1 Esch_coli_1190                               2.020     114
UMB11_08   Esch_coli_1190                               2.363      64

                                                     highPct  gapCount  \
sample     ref
UMB11_11   Esch_coli_1190                              0.010         9
UMB11_06   Esch_coli_H3                                1.835         9
           Esch_coli_1190                             14.264         9
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8   0.345        17
           Esch_coli_1190                              0.069        12
UMB11_03   Esch_coli_1190                              0.001         9
UMB11_01   Esch_coli_NGF1                              0.073         1
UMB11_03.1 Esch_coli_1190                              0.002         7
UMB11_08   Esch_coli_1190                              0.001         6

                                                     gapLength  \
sample     ref
UMB11_11   Esch_coli_1190                                165998
```

(continues on next page)

(continued from previous page)

```
UMB11_06   Esch_coli_H3                                    120001
           Esch_coli_1190                                  165099
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8  347002
           Esch_coli_1190                                  171056
UMB11_03   Esch_coli_1190                                  185085
UMB11_01   Esch_coli_NGF1                                   16868
UMB11_03.1 Esch_coli_1190                                  172806
UMB11_08   Esch_coli_1190                                  158445

                                                        straingst_present  \
sample     ref
UMB11_11   Esch_coli_1190                                           True
UMB11_06   Esch_coli_H3                                             True
           Esch_coli_1190                                           True
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8           True
           Esch_coli_1190                                           True
UMB11_03   Esch_coli_1190                                           True
UMB11_01   Esch_coli_NGF1                                           True
UMB11_03.1 Esch_coli_1190                                           True
UMB11_08   Esch_coli_1190                                           True

                                                        is_plasmid  \
sample     ref
UMB11_11   Esch_coli_1190                                    False
UMB11_06   Esch_coli_H3                                      False
           Esch_coli_1190                                    False
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8    False
           Esch_coli_1190                                    False
UMB11_03   Esch_coli_1190                                    False
UMB11_01   Esch_coli_NGF1                                    False
UMB11_03.1 Esch_coli_1190                                    False
UMB11_08   Esch_coli_1190                                    False

                                                        enough_cov
sample     ref
UMB11_11   Esch_coli_1190                                     True
UMB11_06   Esch_coli_H3                                       True
           Esch_coli_1190                                     True
UMB11_07   Esch_coli_f974b26a-5e81-11e8-bf7f-3c4a9275d6c8     True
           Esch_coli_1190                                     True
UMB11_03   Esch_coli_1190                                     True
UMB11_01   Esch_coli_NGF1                                     True
UMB11_03.1 Esch_coli_1190                                     True
UMB11_08   Esch_coli_1190                                     True

[9 rows x 23 columns]
```

Load `compare` data in a DataFrame

The above data mainly contains data per sample of individual strains as compared to its closest reference. In general, we are often more interested how strains in each sample relate to each other. These kind of relationships are computed with the `straingr compare` command. Here, we load the data from `compare`, make sure we only include comparisons between strains that were predicted to be present by StrainGST, and plot the ACNI/gap similarity.

[27]:
```
compare_df = pandas.read_csv(STRAINGR_DIR / "compare.summary.chrom.txt", sep='\t', index_
↪col=[0, 1, 2])

def both_straingst_present(ix):
    sample1, sample2, ref = ix

    return (sample1, ref) in straingr_df.index and (sample2, ref) in straingr_df.index

compare_df['both_present'] = compare_df.index.map(both_straingst_present)
compare_df = compare_df[compare_df['both_present']].copy()
compare_df
```

[27]:
```
                                         scaffold   length   common  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190  NZ_CP023386.1  4900891   126732
           UMB11_06   Esch_coli_1190  NZ_CP023386.1  4900891   336010
           UMB11_07   Esch_coli_1190  NZ_CP023386.1  4900891   405110
           UMB11_08   Esch_coli_1190  NZ_CP023386.1  4900891   117635
           UMB11_11   Esch_coli_1190  NZ_CP023386.1  4900891   611275
UMB11_03.1 UMB11_06   Esch_coli_1190  NZ_CP023386.1  4900891   215863
           UMB11_07   Esch_coli_1190  NZ_CP023386.1  4900891   252557
           UMB11_08   Esch_coli_1190  NZ_CP023386.1  4900891    75899
           UMB11_11   Esch_coli_1190  NZ_CP023386.1  4900891   390913
UMB11_06   UMB11_07   Esch_coli_1190  NZ_CP023386.1  4900891   718395
           UMB11_08   Esch_coli_1190  NZ_CP023386.1  4900891   199668
           UMB11_11   Esch_coli_1190  NZ_CP023386.1  4900891  1059446
UMB11_07   UMB11_08   Esch_coli_1190  NZ_CP023386.1  4900891   235328
           UMB11_11   Esch_coli_1190  NZ_CP023386.1  4900891  1284089
UMB11_08   UMB11_11   Esch_coli_1190  NZ_CP023386.1  4900891   358765

                                    commonPct   single  singlePct  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190     2.5859   126732   100.0000
           UMB11_06   Esch_coli_1190     6.8561   335954    99.9833
           UMB11_07   Esch_coli_1190     8.2660   405067    99.9894
           UMB11_08   Esch_coli_1190     2.4003   117635   100.0000
           UMB11_11   Esch_coli_1190    12.4727   611224    99.9917
UMB11_03.1 UMB11_06   Esch_coli_1190     4.4046   215823    99.9815
           UMB11_07   Esch_coli_1190     5.1533   252524    99.9869
           UMB11_08   Esch_coli_1190     1.5487    75897    99.9974
           UMB11_11   Esch_coli_1190     7.9764   390894    99.9951
UMB11_06   UMB11_07   Esch_coli_1190    14.6585   718228    99.9768
           UMB11_08   Esch_coli_1190     4.0741   199626    99.9790
           UMB11_11   Esch_coli_1190    21.6174  1059249    99.9814
UMB11_07   UMB11_08   Esch_coli_1190     4.8017   235291    99.9843
           UMB11_11   Esch_coli_1190    26.2011  1283913    99.9863
```

(continues on next page)

(continued from previous page)

```
UMB11_08   UMB11_11   Esch_coli_1190     7.3204   358752    99.9964

                                     singleAgree  singleAgreePct  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190       126725         99.9945
           UMB11_06   Esch_coli_1190       335809         99.9568
           UMB11_07   Esch_coli_1190       405023         99.9891
           UMB11_08   Esch_coli_1190       117629         99.9949
           UMB11_11   Esch_coli_1190       611203         99.9966
UMB11_03.1 UMB11_06   Esch_coli_1190       215758         99.9699
           UMB11_07   Esch_coli_1190       252490         99.9865
           UMB11_08   Esch_coli_1190        75894         99.9960
           UMB11_11   Esch_coli_1190       390880         99.9964
UMB11_06   UMB11_07   Esch_coli_1190       717916         99.9566
           UMB11_08   Esch_coli_1190       199556         99.9649
           UMB11_11   Esch_coli_1190      1058911         99.9681
UMB11_07   UMB11_08   Esch_coli_1190       235271         99.9915
           UMB11_11   Esch_coli_1190      1283763         99.9883
UMB11_08   UMB11_11   Esch_coli_1190       358744         99.9978

                                     sharedAlleles  sharedAllelesPct  ...  \
sample1    sample2    ref                                             ...
UMB11_03   UMB11_03.1 Esch_coli_1190         126725           99.9945  ...
           UMB11_06   Esch_coli_1190         335865           99.9568  ...
           UMB11_07   Esch_coli_1190         405066           99.9891  ...
           UMB11_08   Esch_coli_1190         117629           99.9949  ...
           UMB11_11   Esch_coli_1190         611254           99.9966  ...
UMB11_03.1 UMB11_06   Esch_coli_1190         215798           99.9699  ...
           UMB11_07   Esch_coli_1190         252523           99.9865  ...
           UMB11_08   Esch_coli_1190          75896           99.9960  ...
           UMB11_11   Esch_coli_1190         390899           99.9964  ...
UMB11_06   UMB11_07   Esch_coli_1190         718083           99.9566  ...
           UMB11_08   Esch_coli_1190         199598           99.9649  ...
           UMB11_11   Esch_coli_1190        1059108           99.9681  ...
UMB11_07   UMB11_08   Esch_coli_1190         235308           99.9915  ...
           UMB11_11   Esch_coli_1190        1283939           99.9883  ...
UMB11_08   UMB11_11   Esch_coli_1190         358757           99.9978  ...

                                     BnotAweak  BnotAweakPct   Agaps  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190          5       17.8571  185085
           UMB11_06   Esch_coli_1190        160       57.3477  185085
           UMB11_07   Esch_coli_1190         65       32.5000  185085
           UMB11_08   Esch_coli_1190          2        3.8462  185085
           UMB11_11   Esch_coli_1190         29       11.9835  185085
UMB11_03.1 UMB11_06   Esch_coli_1190         95       71.4286  172806
           UMB11_07   Esch_coli_1190         56       43.0769  172806
           UMB11_08   Esch_coli_1190          2       10.0000  172806
           UMB11_11   Esch_coli_1190         14       14.0000  172806
UMB11_06   UMB11_07   Esch_coli_1190        151       22.1083  165099
           UMB11_08   Esch_coli_1190         18       11.1111  165099
           UMB11_11   Esch_coli_1190         50        6.2500  165099
```

(continues on next page)

(continued from previous page)

```
UMB11_07   UMB11_08   Esch_coli_1190         6        5.1724  171056
           UMB11_11   Esch_coli_1190        26        3.8981  171056
UMB11_08   UMB11_11   Esch_coli_1190         5        3.6765  158445

                                        AsharedGaps   AgapPct   Bgaps  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190       163905   88.5566  172806
           UMB11_06   Esch_coli_1190       163905   88.5566  165099
           UMB11_07   Esch_coli_1190       175075   94.5917  171056
           UMB11_08   Esch_coli_1190       145541   78.6347  158445
           UMB11_11   Esch_coli_1190       185085  100.0000  165998
UMB11_03.1 UMB11_06   Esch_coli_1190       172806  100.0000  165099
           UMB11_07   Esch_coli_1190       172806  100.0000  171056
           UMB11_08   Esch_coli_1190       148091   85.6978  158445
           UMB11_11   Esch_coli_1190       172806  100.0000  165998
UMB11_06   UMB11_07   Esch_coli_1190       158136   95.7825  171056
           UMB11_08   Esch_coli_1190       151355   91.6753  158445
           UMB11_11   Esch_coli_1190       158136   95.7825  165998
UMB11_07   UMB11_08   Esch_coli_1190       131119   76.6527  158445
           UMB11_11   Esch_coli_1190       154895   90.5522  165998
UMB11_08   UMB11_11   Esch_coli_1190       146928   92.7312  165998

                                        BsharedGaps   BgapPct  gapJaccardSim  \
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190       172806  100.0000         0.9919
           UMB11_06   Esch_coli_1190       158136   95.7825         0.9895
           UMB11_07   Esch_coli_1190       154895   90.5522         0.9872
           UMB11_08   Esch_coli_1190       146928   92.7312         0.9971
           UMB11_11   Esch_coli_1190       165998  100.0000         0.9889
UMB11_03.1 UMB11_06   Esch_coli_1190       158136   95.7825         0.9922
           UMB11_07   Esch_coli_1190       148033   86.5407         0.9879
           UMB11_08   Esch_coli_1190       146928   92.7312         0.9916
           UMB11_11   Esch_coli_1190       154893   93.3102         0.9916
UMB11_06   UMB11_07   Esch_coli_1190       148033   86.5407         0.9898
           UMB11_08   Esch_coli_1190       158445  100.0000         0.9951
           UMB11_11   Esch_coli_1190       154893   93.3102         0.9964
UMB11_07   UMB11_08   Esch_coli_1190       146928   92.7312         0.9909
           UMB11_11   Esch_coli_1190       160523   96.7018         0.9900
UMB11_08   UMB11_11   Esch_coli_1190       139943   84.3040         0.9945

                                        both_present
sample1    sample2    ref
UMB11_03   UMB11_03.1 Esch_coli_1190           True
           UMB11_06   Esch_coli_1190           True
           UMB11_07   Esch_coli_1190           True
           UMB11_08   Esch_coli_1190           True
           UMB11_11   Esch_coli_1190           True
UMB11_03.1 UMB11_06   Esch_coli_1190           True
           UMB11_07   Esch_coli_1190           True
           UMB11_08   Esch_coli_1190           True
           UMB11_11   Esch_coli_1190           True
UMB11_06   UMB11_07   Esch_coli_1190           True
```

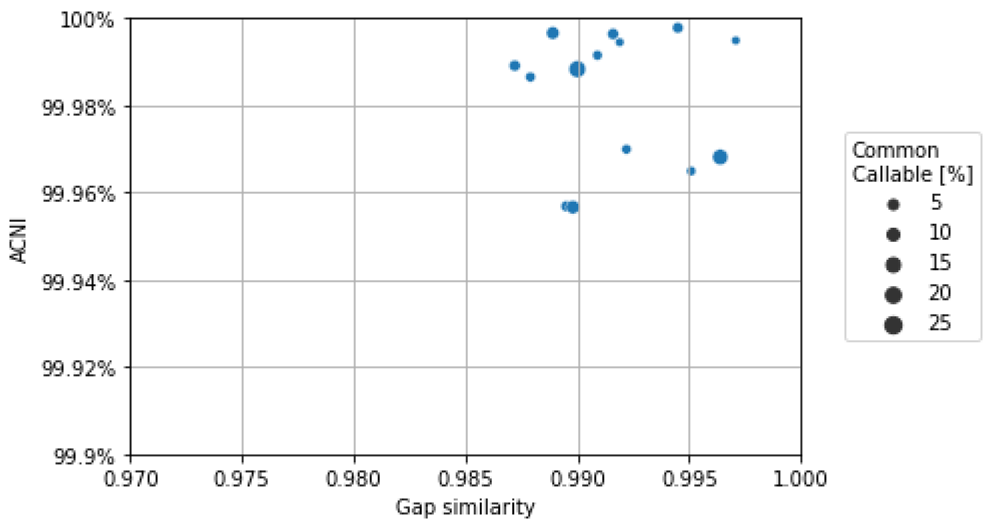
(continues on next page)

(continued from previous page)

```
           UMB11_08   Esch_coli_1190          True
           UMB11_11   Esch_coli_1190          True
UMB11_07   UMB11_08   Esch_coli_1190          True
           UMB11_11   Esch_coli_1190          True
UMB11_08   UMB11_11   Esch_coli_1190          True

[15 rows x 32 columns]
```

[41]:
```
import seaborn

seaborn.scatterplot(x="gapJaccardSim", y="singleAgreePct", size="commonPct",␣
↪data=compare_df)

plt.xlim(0.970, 1.0)
plt.xlabel("Gap similarity")

plt.ylim(99.9, 100)
plt.ylabel("ACNI")
plt.gca().yaxis.set_major_formatter("{x:g}%")

plt.grid('on')
plt.legend(title="Common\nCallable [%]", loc="center left", bbox_to_anchor=(1.05, 0.5))
```

[41]:
```
<matplotlib.legend.Legend at 0x160142700>
```



CHAPTER FOUR

CITATION

If you use StrainGE in your project, please consider citing our publication:

> Dijk, Lucas R. van, Bruce J. Walker, Timothy J. Straub, Colin J. Worby, Alexandra Grote, Henry L. Schreiber, Christine Anyansi, et al. 2022. "StrainGE: A Toolkit to Track and Characterize Low-Abundance Strains in Complex Microbial Communities." Genome Biology 23 (1): 74. https://doi.org/10.1186/s13059-022-02630-0.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search