Reproducing Analysis of StartLink

Karl Gemayel

May 21 2020

Contents

1 Introduction **1**

2 Downloading and installing **1**
2.1 Code 1
2.2 External Tools 2
2.3 Data 2

3 Code and data structure **3**

4 Setting up **4**

5 Experiments **5**
5.1 Difference in 5' predictions on Representative Genomes 5
5.2 Theoretical view of Independence 7
5.3 Genomes with genes with verified starts 8
5.4 Larger set of query genomes 9

1 Introduction

This document serves as a step-by-step instruction manual on how to replicate results from the StartLink paper. We note that many of the experiments are time-consuming and/or resource hungry. Therefore, we also provide raw results (from which graphs can be generated) as well as pre-constructed databases for the clades mentioned in the StartLink paper.

2 Downloading and installing

2.1 Code

Downloading the code is fairly straightforward using `git`. The latest version can be downloaded from `WEBSITE`. To install, simply run

```
source config.sh
source install.sh
```

The first command loads all environment variables (including paths to data directories, binaries, etc...), and the second command creates an executable from all python files and stores them in `$bin` for easy access. For non-unix users, you can find the python driver files in `$driverpython`.

2.2 External Tools

StartLink and StartLink+ and their analysis rely on a handful of external tools. The following need to be installed:

- GeneMarkS-2
  - Used for building StartLink+ predictions, and for analysis
  - Link: `http://exon.gatech.edu/GeneMark/license_download.cgi`
- ClustalO:
  - Used for constructing multiple sequence alignments
  - Link: `http://www.clustal.org/omega/#Download`
- Diamond BLAST:
  - Used for generating target databases and finding orthologs
  - Link: `https://github.com/bbuchfink/diamond`
- Prodigal:
  - Used for initial analysis of gene-start prediction status
  - Link: `https://github.com/hyattpd/Prodigal`

2.3 Data

Note: As previously mentioned the data used is on the order of hundreds of gigabytes. As such, if one is only interested in reproducibility, we provide pre-built databases (and even raw statistics from our existing runs).

We provide the databases for *Enterobacterales*, *Actinobacteria*, *Archaea*, and *FCB group*, and the sequence and label files for the genomes with verified starts: *E. coli*, *H. salinarum*, *N. pharaonis*, *M. tuberculosis*, and *R. denitrificans*. We also provide the steps to create a database with for any ancestor using data that can be downloaded from NCBI's website.

2.3.1 Downloading Assembly Summary File

```
$bin/download_assembly_summary_py.sh --database refseq --pf-output
    $metadata/refseq_assembly_summary.txt
```

2.3.2 Constructing Taxonomy Tree

```
pd_work=$tmp/tree

mkdir -p $pd_work

$bin/download_taxonomy_dump_py.sh --pd-output $metadata/taxdump
$bin/build_taxonomy_tree_py.sh --pf-nodes-dmp $metadata/taxdump/nodes.dmp --pf-names-dmp
    $metadata/taxdump/names.dmp --pf-tree $pd_work/tree.pkl
```

2.3.3 Download sequence/label files for different clades, and create Blast databases

```
dbt=refseq     # database type
pf_tree=$tmp/tree/tree.pkl
pf_ass_sum_comb=$metadata/${dbt}_assembly_summary_combined.txt

declare -a clades=("Enterobacterales" "Actinobacteria" "Alphaproteobacteria" "FCB group"
    "Archaea")

# loop over clades; download data under each clade
for cl in ${clades[@]}; do
  dn_cl=$(mk_path_friendly "$cl")
  $bin/download_genomes_for_clade_py.sh --pf-tree $pf_tree --pf_assembly_summary
      $pf_ass_sum_comb --ancestor-id $cl --ancestor-id-type "name_txt"
      --favor-assembly-level-order --number-per-taxid 1 --pf-output-list
      --$lists/${dbt}_${dn_cl}.list
done
```

Construct Diamond Blastp databases

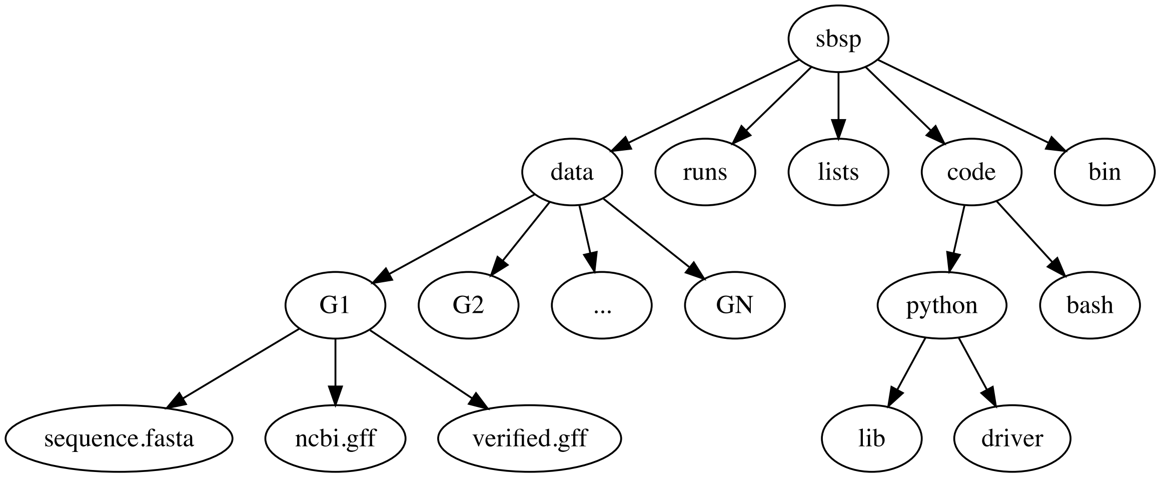
```
for cl in "${clades[@]}"; do
  dn_cl=$(mk_path_friendly "$cl")

  pf_list=$lists/${dbt}_${dn_cl}.list
  pf_faa=$lists/${dbt}_${dn_cl}.list
  pf_db=$db/${dbt}_${dn_cl}.dmnd

  # extract sequences
  $bin/extract_labeled_sequences_py.sh --pf-genome-list $pf_list --pf-output $pf_faa
      --pf-pbs-options $config/pbs_defaults.conf

  # build blast
  $bin/build_blast_db_py.sh --pf_sequences $pf_faa --pf-db $pf_db --pf-pbs-options
      $config/pbs_single_node.conf

  # clean up sequence file
  [[ -f $pf_faa ]] & rm $pf_faa
done
```

2.3.4 Download query genomes from list

```
pf_query_large=$lists/selected_query.list
pf_ass_sum_query_large=$metadata/assembly_summary_query_large.txt
$bin/download_genomes_from_list_py.sh --pf-genome-list $pf_query_large
    --pf-assembly-summary $pf_ass_sum_query_large --pf-pbs-options
    $config/pbs_defaults.conf
```

3 Code and data structure

After installing StartLink, you will have the following structure:

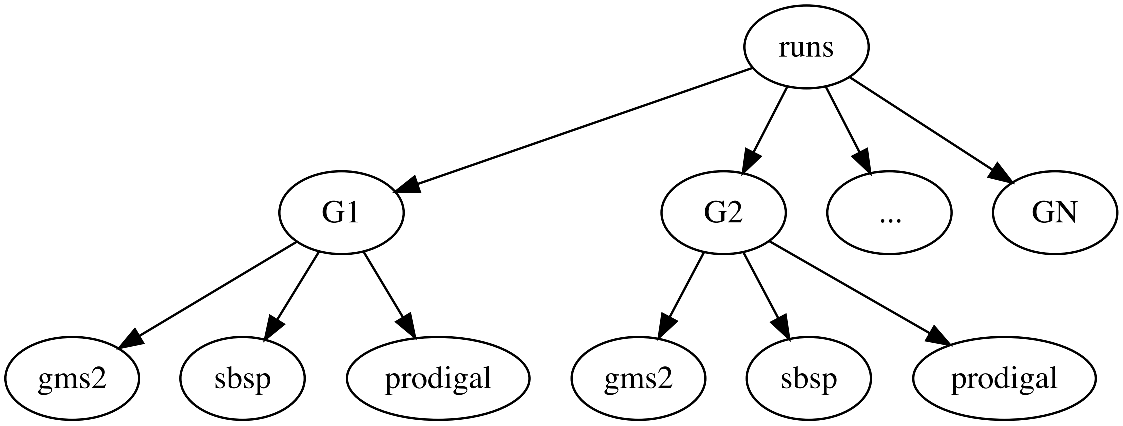


The `bin` directory contains all executables related to StartLink, while the `bin_external` may contain external tools, such as GeneMarkS-2 or Prodigal.

The `data` directory will contain raw genome files (sequence and annotation labels) downloaded from NCBI. In particular, upon initial download of the code, it should contain the genomic sequences for the genomes with experimentally verified gene-starts.

The `list` directory has files that contain different lists of genomes (for example, those with verified genes, those selected as NCBI query genomes, etc. . .)

Finally the `runs` directory will contain runs of different tools, such as StartLink, GeneMarkS-2, or Prodigal (as well as one for NCBI's `PGAP`). These will be placed in a subdirectory per genome, as shown below.



4 Setting up

Since much of the analysis is done by comparing StartLink to NCBI's PGAP, GeneMarkS-2, and/or Prodigal, we first need to run these tools and add the results to the run directory. The following script is capable of doing that (note, depending on which analysis you want to reproduce, you may not need to run the tools on all lists):

```
function run_tools_on_archaea() {
  pf_list="$1"

  $bin/run_tool_on_genome_list_py.sh --tool gms2 --pf-genome-list $pf_list --type archaea
  $bin/run_tool_on_genome_list_py.sh --tool prodigal --pf-genome-list $pf_list --type
      archaea
}

function run_tools_on_bacteria() {
  pf_list="$1"

  $bin/run_tool_on_genome_list_py.sh --tool gms2 --pf-genome-list $pf_list --type bacteria
  $bin/run_tool_on_genome_list_py.sh --tool prodigal --pf-genome-list $pf_list --type
      bacteria
}

# Representative genomes
run_tools_on_archaea $pf_rep_arc
run_tools_on_bacteria $pf_rep_bac

# Verified genomes
run_tools_on_archaea $pf_list_verified_arc
run_tools_on_bacteria $pf_list_verified_bac

# NCBI query genomes
run_tools_on_archaea $pf_list_qncbi_arc
run_tools_on_bacteria $pf_list_qncbi_bac
```

5 Experiments

Unless otherwise noted, these variables (when applicable) will have the following values

```
pf_list_verified=$lists/verified.list  # verified genomes
pf_list_qncbi=$lists/genbank_selected.list   # query genomes

# database and configuration files
pf_db_index=$db/index.csv  # database location files
pf_sbsp_options=$config/sbsp_defaults.conf # sbsp config file
pf_pbs_options=$config/pbs_defaults.conf   # PBS config file

# PBS options
toggle_pbs="--pf-pbs-options $config/$pf_pbs_options"  # if PBS not installed, set this
    option to empty: ""
sg=8   # number of genomes to run simutaneously (low number recommended)
```

5.1 Difference in 5' predictions on Representative Genomes

5.1.1 Data download

```
pf_rep_bac=$lists/refseq_representative_bacteria.list
pf_rep_arc=$lists/refseq_representative_archaea.list
pf_assembly_bac=$metadata/assembly_summary.txt
$bin/download_from_ncbi_py.sh --pf-assembly-summary $pf_assembly_bac --pf-data $data
    --pf-output-list

# link ncbi as "tool" (for easy comparison wwith other tools)
cat $pf_rep_bac $pf_rep_arc | grep -v gcfid | cut -f1 -d, | while read -r line; do
  mkdir -p $runs/$line; mkdir -p $runs/$line/ncbi;
  ln -s $data/$line/ncbi.gff $runs/$line/ncbi/ncbi.gff ;
done
```

5.1.2 Run GMS2 and Prodigal

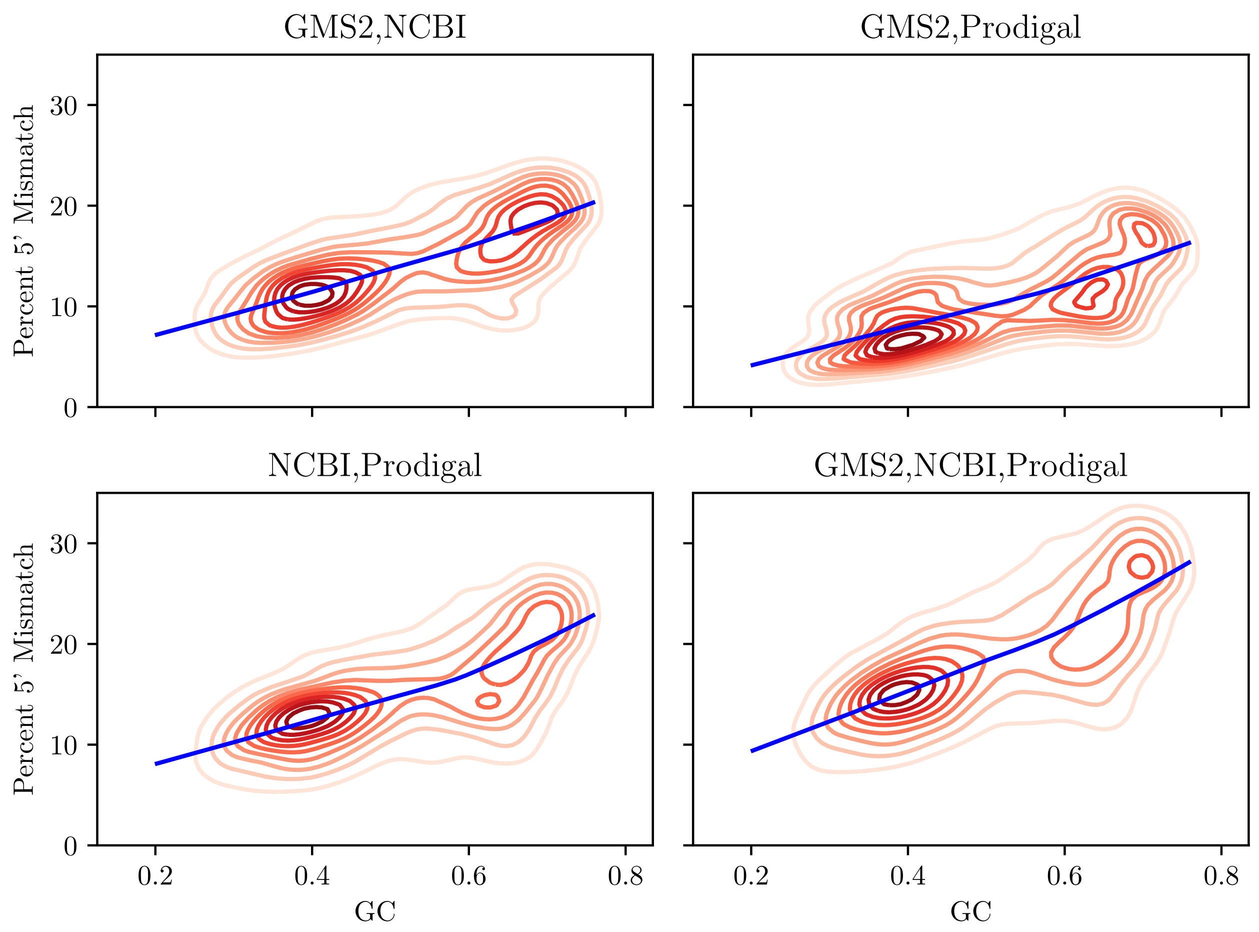
```
# Run on GMS2
$bin/run_tool_on_genome_list_py.sh --tool gms2 --pf-genome-list $pf_rep_bac --type
    bacteria --dn-run gms2
$bin/run_tool_on_genome_list_py.sh --tool gms2 --pf-genome-list $pf_rep_arc --type
    archaea --dn-run gms2

# Run on Prodigal
$bin/run_tool_on_genome_list_py.sh --tool prodigal --pf-genome-list $pf_rep_bac --type
    bacteria --dn-run prodigal
$bin/run_tool_on_genome_list_py.sh --tool prodigal --pf-genome-list $pf_rep_arc --type
    archaea --dn-run prodigal
```

5.1.3 Collect statistics

We can now collect the statistics and create the figures to compare GMS2, Prodigal, and NCBI predictions.

```
pf_stats=$pd_work/stats_tools.csv

$bin/stats_tools_5prime_py.sh --pf-genome-lists $pf_rep_bac $pf_rep_arc --list-names
    Bacteria Archaea --dn-tools gms2 prodigal ncbi --tool-names GMS2 Prodigal NCBI
    --pf-output $pf_stats

# create figures
$bin/viz_stats_tools_5prime_py.sh --pf-stats $pf_stats
```

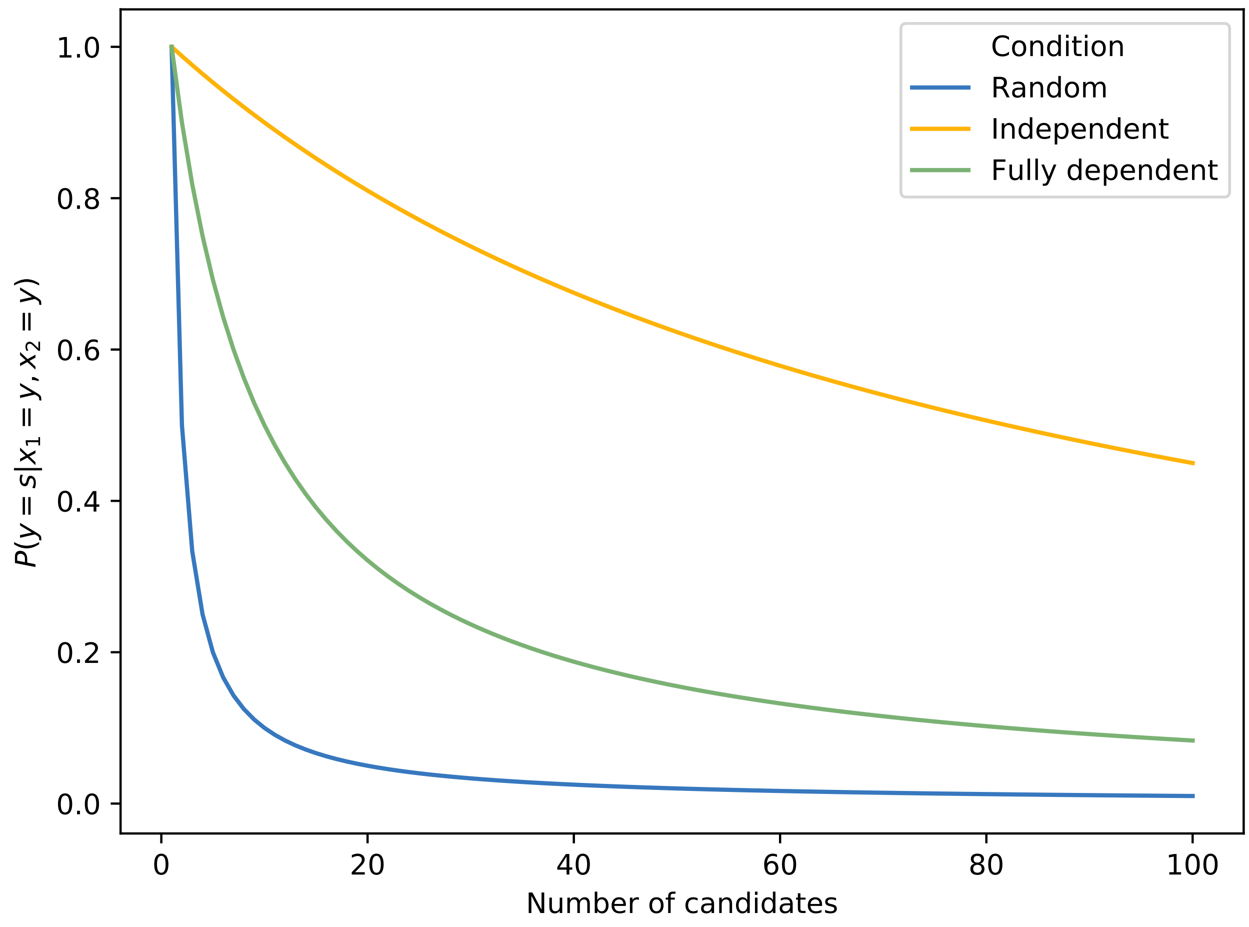
This should now create a file containing the following image



5.2 Theoretical view of Independence

While not technically an experimental result, we provide the code to generate this graph for convenience. The sensitivity of the non-random algorithms A_1 and A_2 are set to 0.9, but the user can easily change them (from within) to observe the change in behavior. What remains constant is the improvement of independent algorithms over fully dependent (and random) algorithms..

```
$bin/independent_predictions_py.sh
```



5.3 Genomes with genes with verified starts

5.3.1 Running StartLink

```
# set this to only run on genes with verified starts
opt_verif="--fn-q-labels verified.gff --fn-q-labels-true verified.gff"

# run SBSP
$bin/sbsp_on_genome_list_py.sh --pf-q-list $pf_list_verified --simultaneous-genomes $sg
    --pd-work $pd_runs --pf-sbsp-options $pf_sbsp_options  --pf-db-index $pf_db_index
    $opt_verif $toggle_pbs
```

5.3.2 Collecting statistics

```
# collect statistics per query gene (comparing SBSP, GMS2, and verified genes)
$bin/stats_per_query_gene_py.sh --pf-genome-list $pf_list_verified --pf-output-summary
    summary.csv --verified
```

5.3.3 Visualizing

```
$bin/viz_stats_genome_level_py.sh --pf-data summary.csv
```

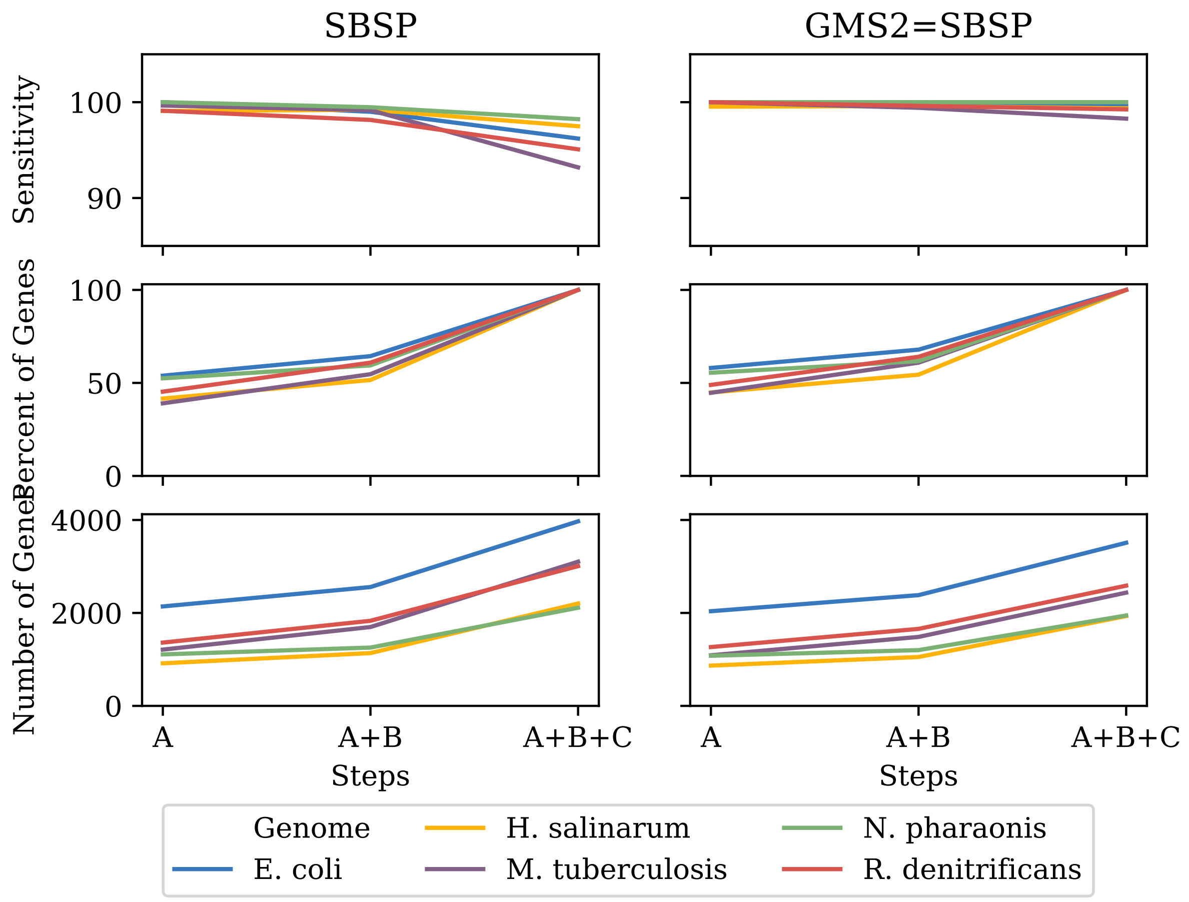
This will produce two files, `error.csv` and `coverage.csv` containing the following two tables.
Error

Genome	Verified	SBSP	GMS2	GMS2=SBSP
E. coli	769	96.204188	97.001304	99.582754
H. salinarum	530	97.489540	98.679245	99.354839
M. tuberculosis	701	93.197279	90.401146	98.282443
N. pharaonis	315	98.226950	99.047619	100.000000
R. denitrificans	526	95.081967	96.571429	99.248120

Coverage

Genome	Verified	SBSP	GMS2	GMS2=SBSP
E. coli	769	99.349805	99.739922	93.498049
H. salinarum	530	90.188679	100.000000	87.735849
M. tuberculosis	701	83.880171	99.572040	74.750357
N. pharaonis	315	89.523810	100.000000	87.301587
R. denitrificans	526	81.178707	99.809886	75.855513

It also produces the per-step analysis on the verified set of genes.



5.4 Larger set of query genomes

5.4.1 Running SBSP

Prewarning, running this analysis can take a long time. Our estimate is roughly 5 days on 20 compute nodes with 8 processors each, though that number can vary based on how databases are setup, where they are located, and the cost of accessing them (e.g. databases can be copied to each node beforehand, making access much cheaper and prevent bottlenecks).

In that respect, we have also provided a CSV file containing the per-query analysis of all genes in this set, which is used for visualization of results.

```
# run SBSP
$bin/sbsp_on_genome_list_py.sh --pf-q-list $pf_list_qncbi --simultaneous-genomes $sg
    --pd-work $pd_runs --pf-sbsp-options $pf_sbsp_options  --pf-db-index $pf_db_index
    $toggle_pbs
```

5.4.2 Collecting statistics

```
# collect statistics per query gene (comparing SBSP, GMS2, and verified genes)
$bin/stats_per_query_gene_py.sh --pf-genome-list $pf_list_qncbi --pf-output-summary
    summary.csv
```

5.4.3 Visualizing

All images regarding the large-scale comparisons can be generated via a single script. Note that the contour plots are computationally expensive and may take ~1 hour to generate. Therefore, they are turned off by default. To enable them, run the command with the option `--with-contours`.

```
$bin/viz_stats_clade_level_py.sh --pf-data summary.csv
```

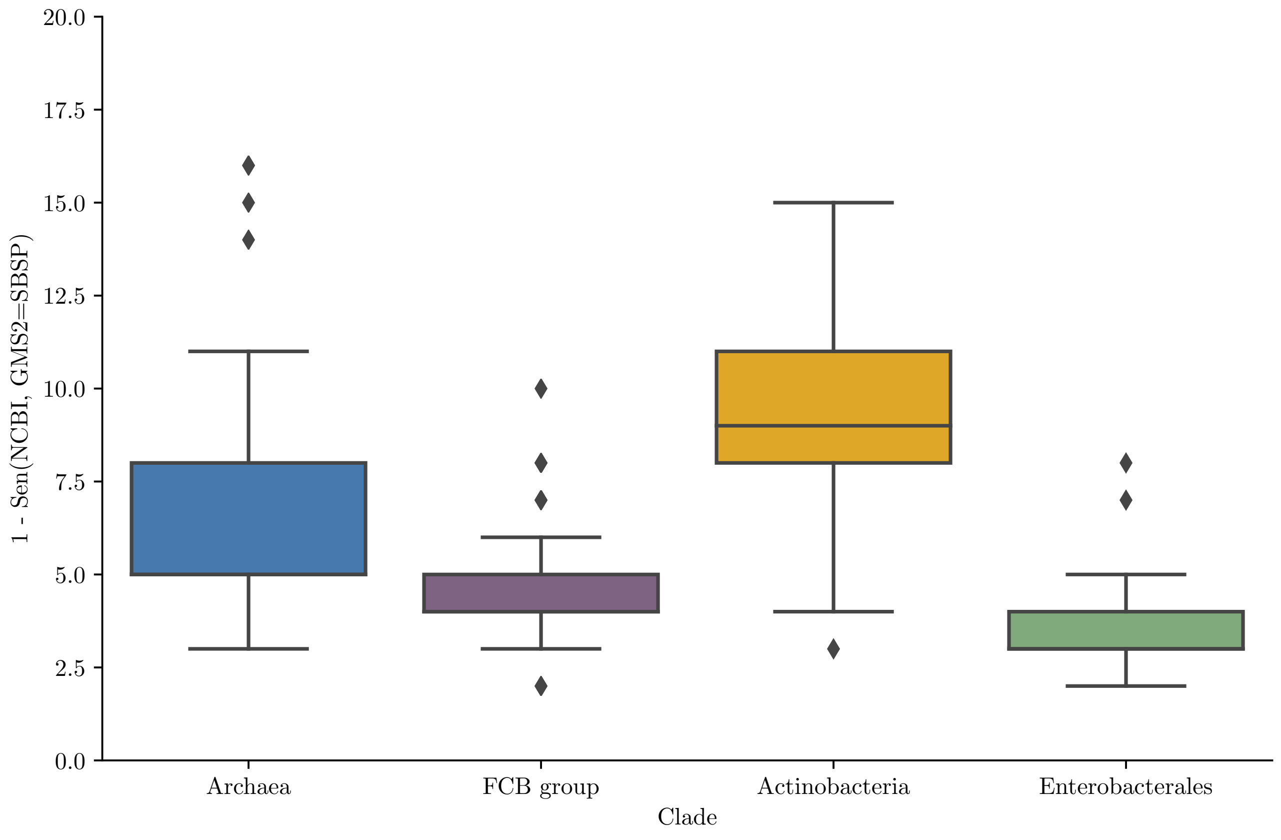


Figure 1: The 5' error rate of NCBI compared to GMS2=SBSP for query genomes in different clades

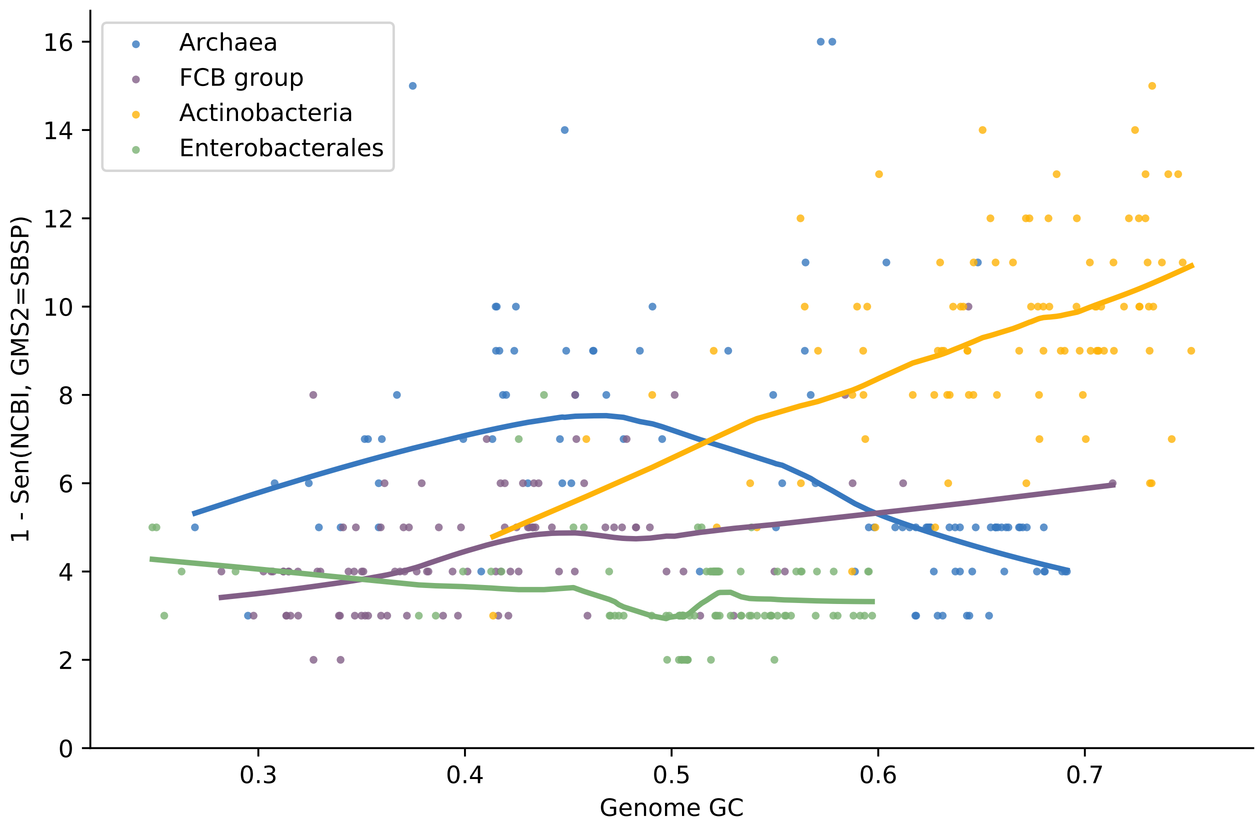


Figure 2: The 5' error rate of NCBI compared to GMS2=SBSP, as a function of genome GC

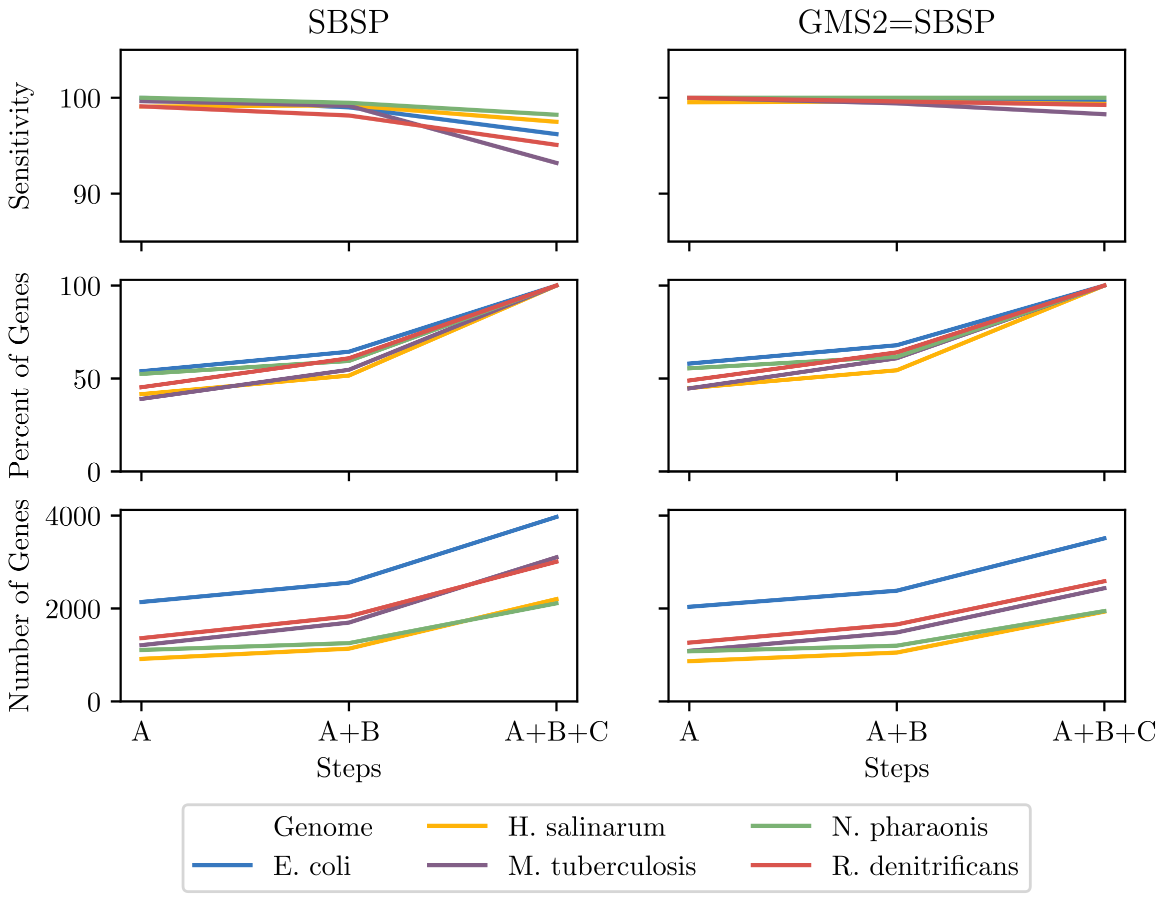


Figure 3: Left: The sensitivity for each SBSP step on the set of verified genes (top), and the percentage (middle) and number (bottom) of SBSP genes predicted by step A alone, steps A and B, and all steps together. Right: Same analysis, for GMS2=SBSP.

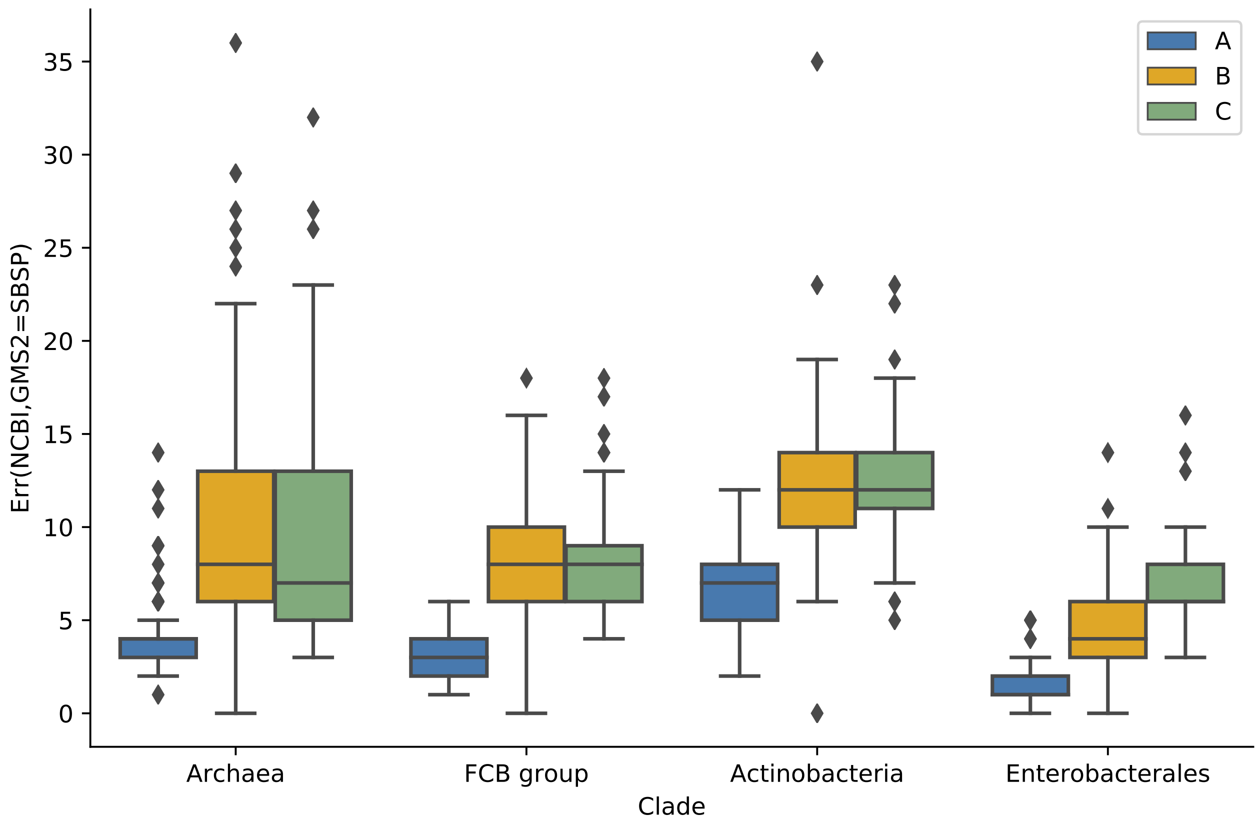


Figure 4: The 5' error rate of NCBI compared to GMS2=SBSP, shown per step of SBSP

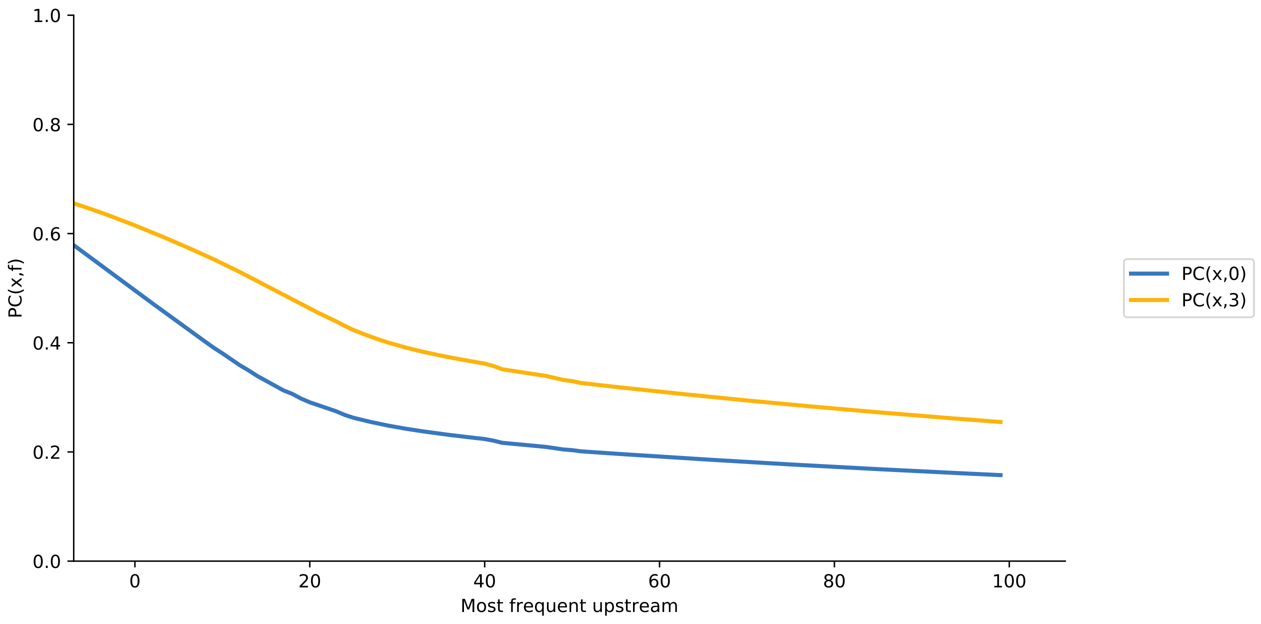


Figure 5: The variation in proximity consistency as the distance to the upstream gene increases

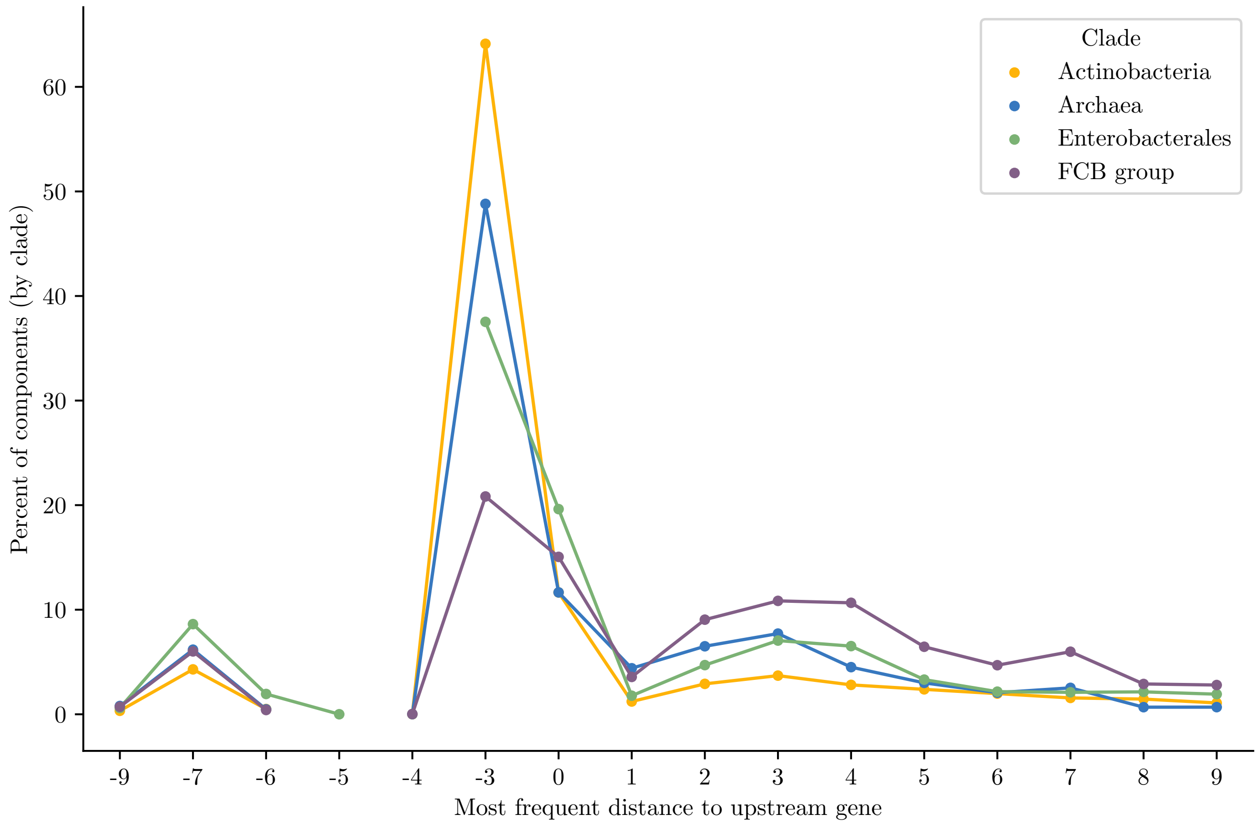


Figure 6: The percentages of components whose most frequent upstream distance lies within the -10 and +10 *nt* range. A component is defined as a single query and its targets

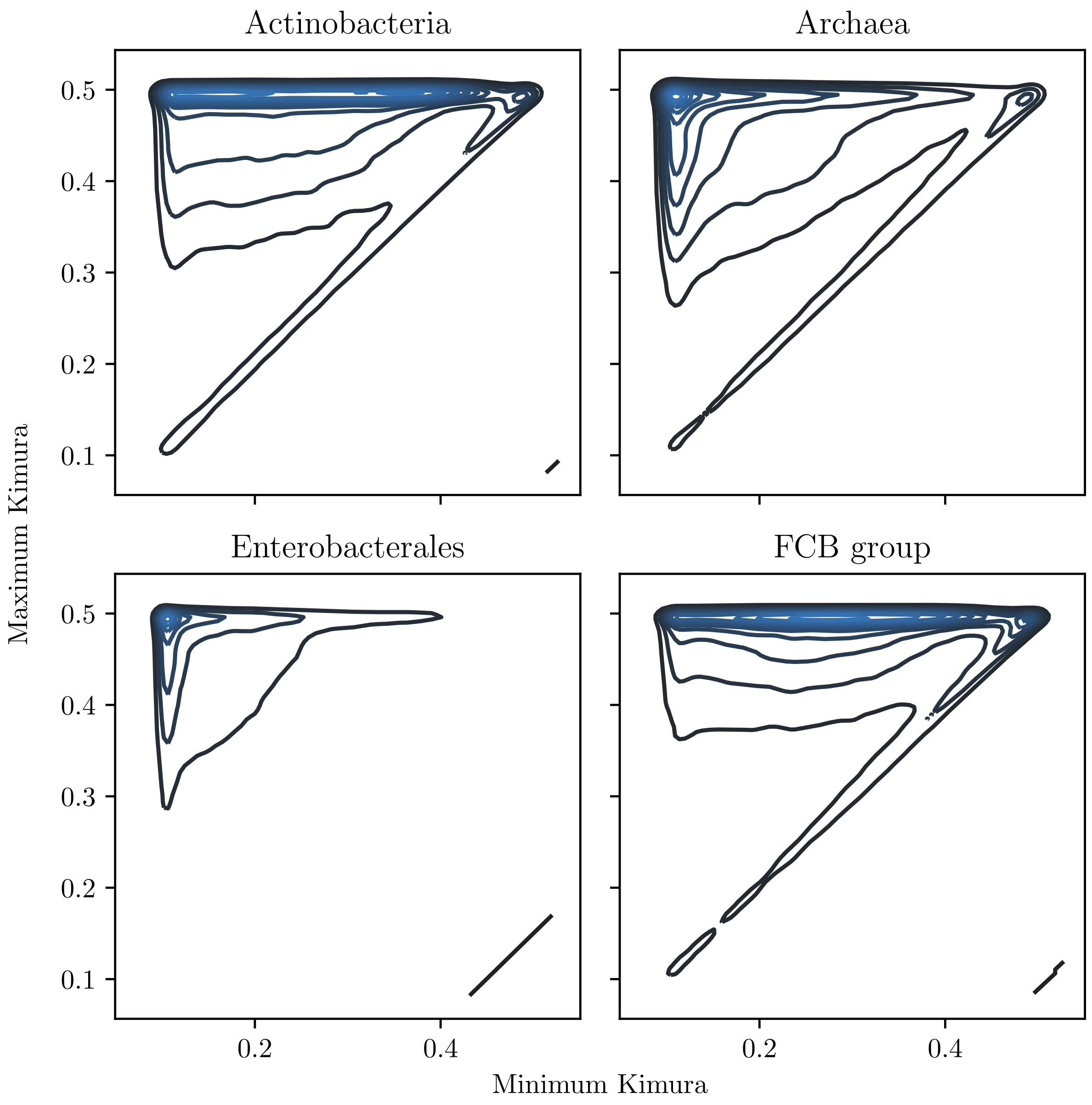


Figure 7: The distribution of queries by minimum and maximum Kimura distance to their orthologs. This shows that most query genes in *Enterobacterales* will find an orthologs that spread the range from 0.1 to 0.5 Kimura, whereas many in *Actinobacteria* have a minimum Kimura distance of above 0.3 and even 0.4

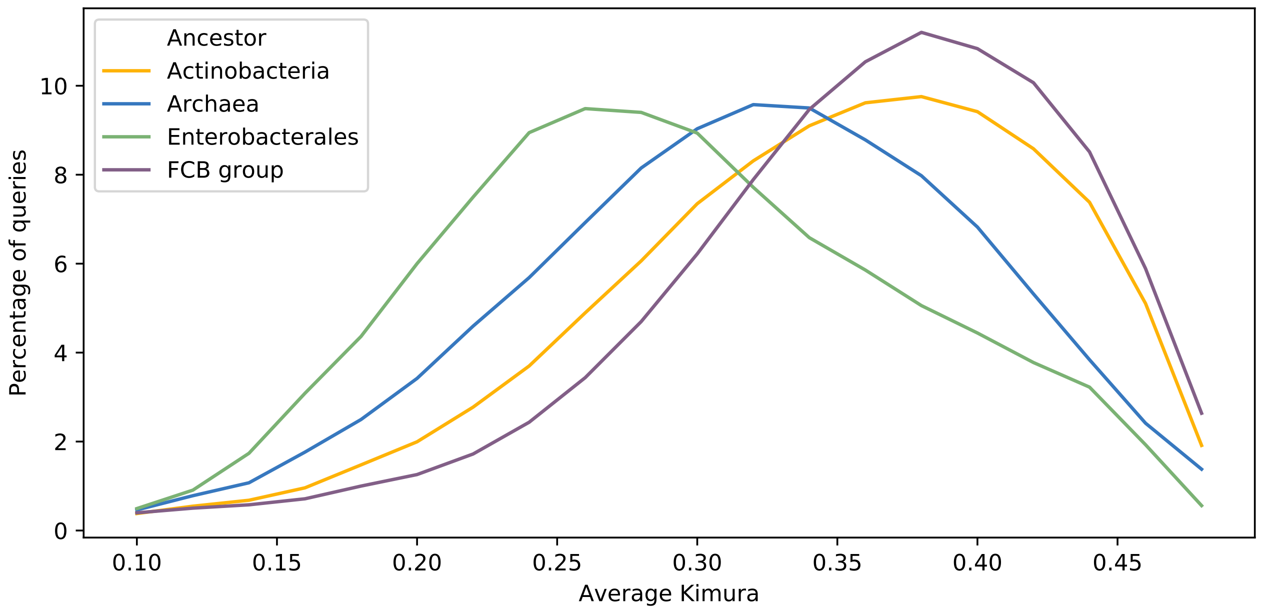


Figure 8: The distribution of average Kimura distances (per component). The y-axis shows the percentage of queries (and thus, components) that have a particular average Kimura distance to its orthologs

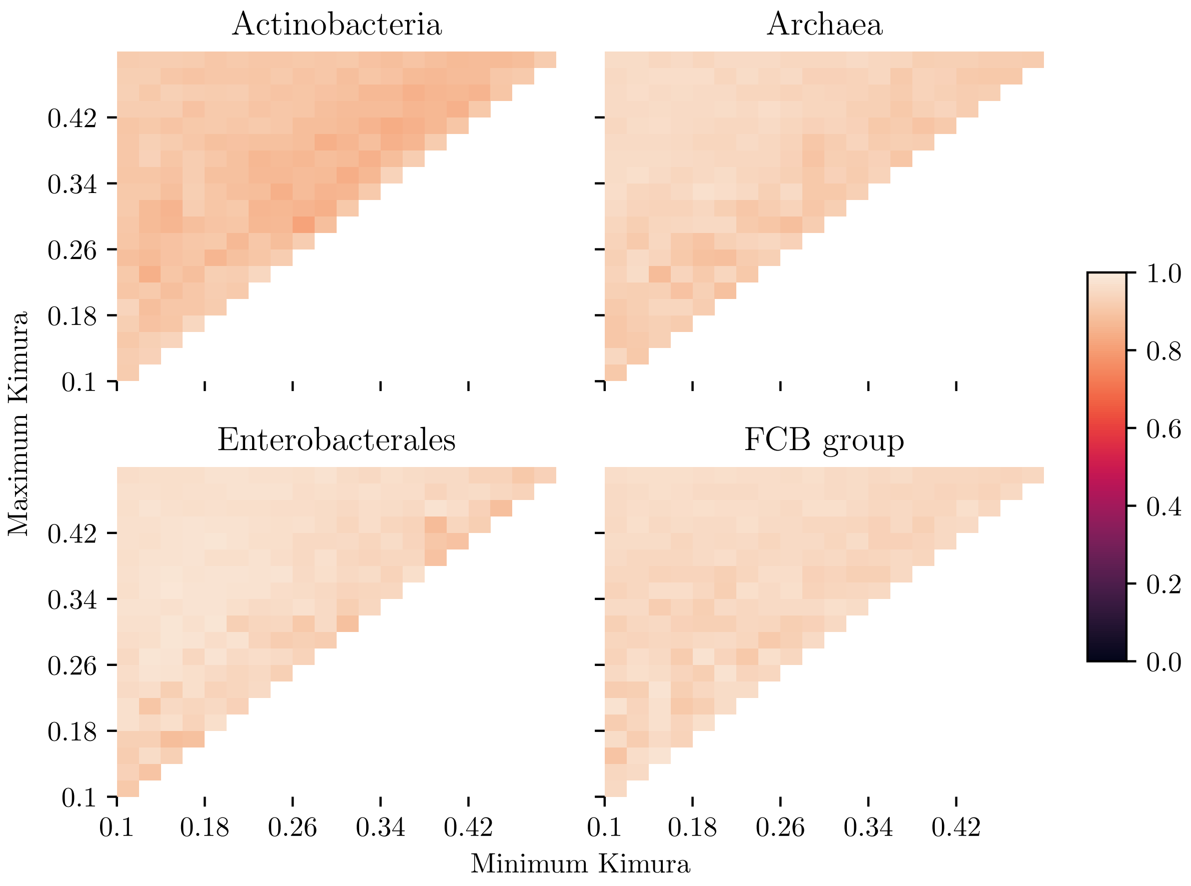


Figure 9: The 5' sensitivity rate of NCBI compared to GMS2=SBSP (i.e. (NCBI, GMS2=SBSP)) based on the minimum and maximum Kimura distances between a query and its targets. The color bar measures the sensitivity rate, with brighter colors indicating higher sensitivity

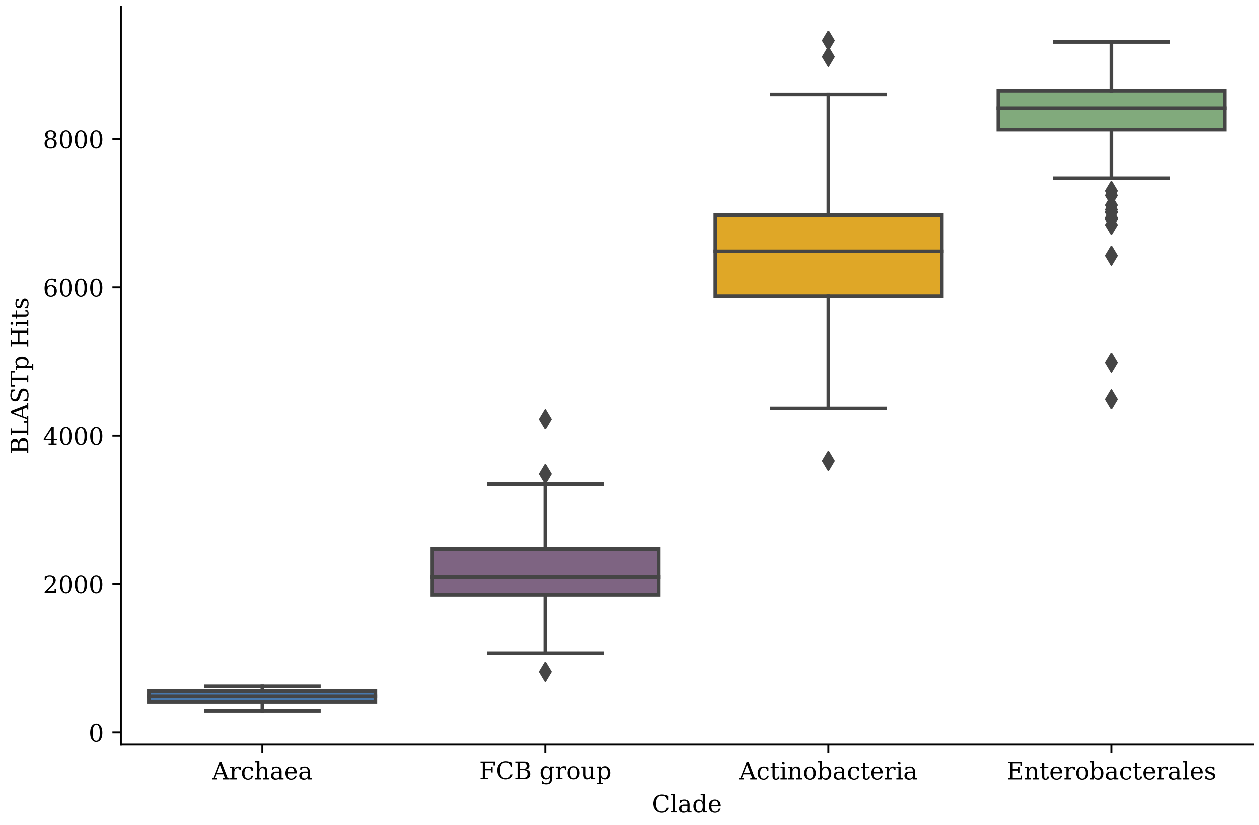


Figure 10: Distribution of raw blast hits across clades for the set of query genomes in Table~**??**. Left: The raw number of BLAST hits per clade. Right: The cumulative percentage of queries with *at most N* BLASTp hits, where N varies from 0 to 5,000. The shaded band shows the standard deviation (per clade) across query genomes

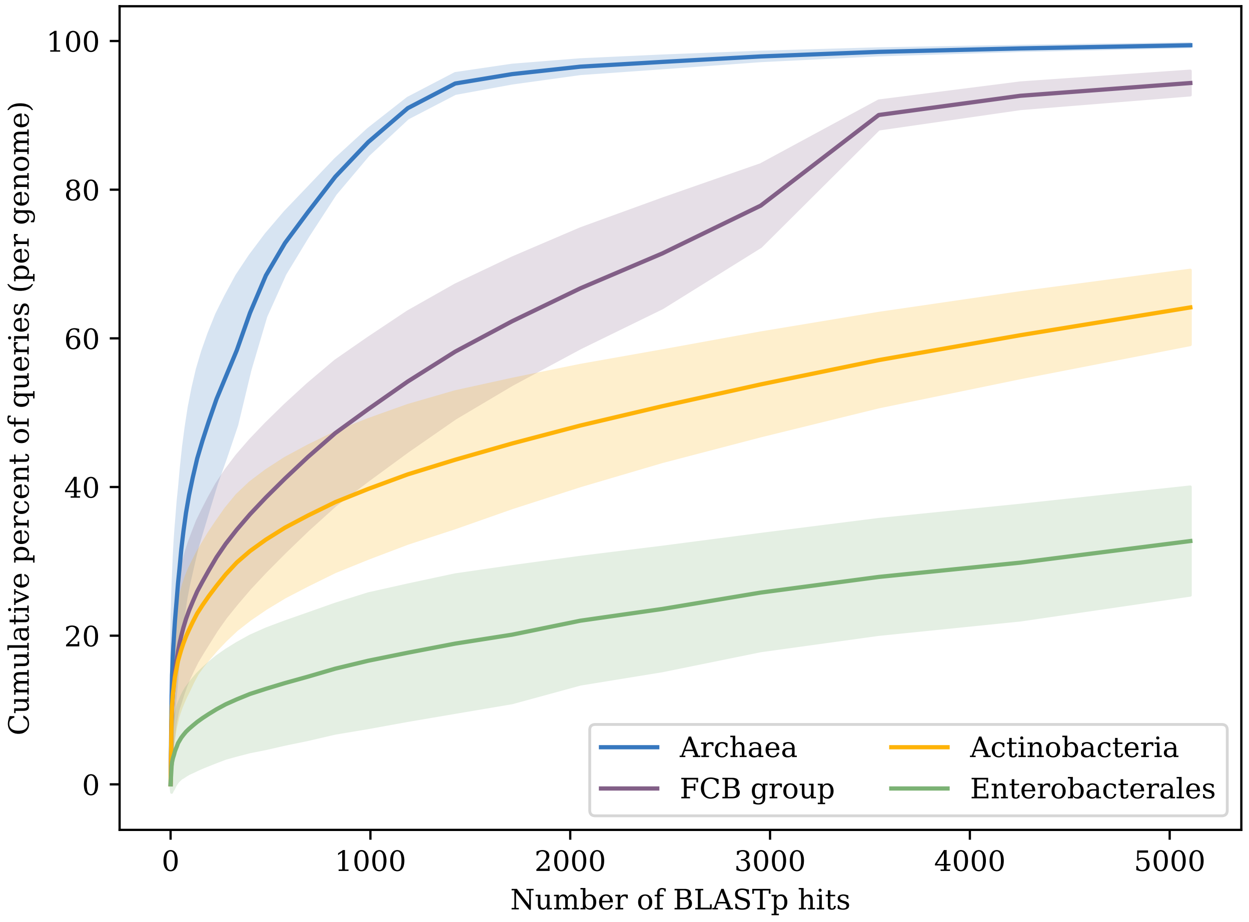


Figure 11: Distribution of raw blast hits across clades for the set of query genomes in Table~**??**. Left: The raw number of BLAST hits per clade. Right: The cumulative percentage of queries with *at most* N BLASTp hits, where N varies from 0 to 5,000. The shaded band shows the standard deviation (per clade) across query genomes

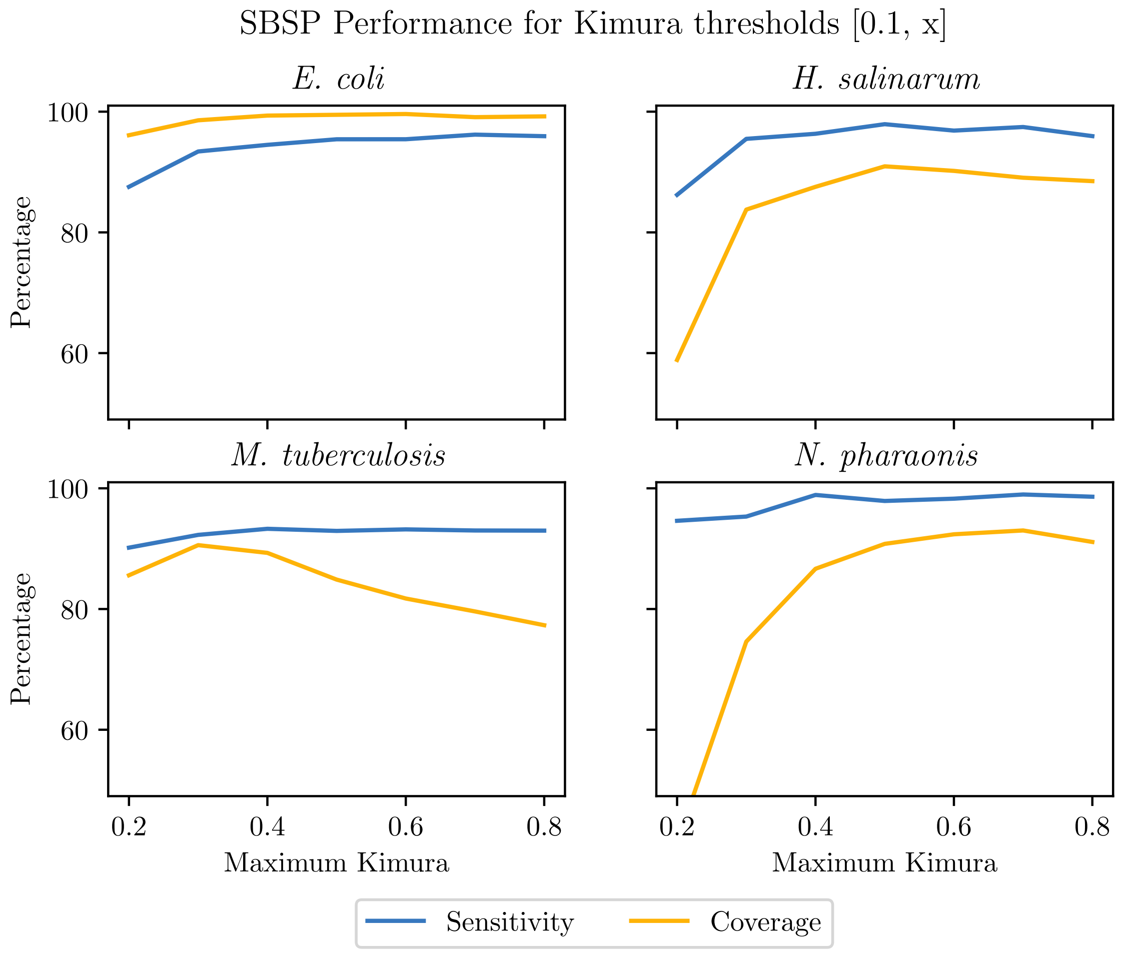


Figure 12: The effect of changing the maximum Kimura threshold on SBSP's sensitivity and coverage rates. The minimum Kimura threshold is fixed to 0.1, and $x \in \{0.2, 0.3, ..., 0.8\}$

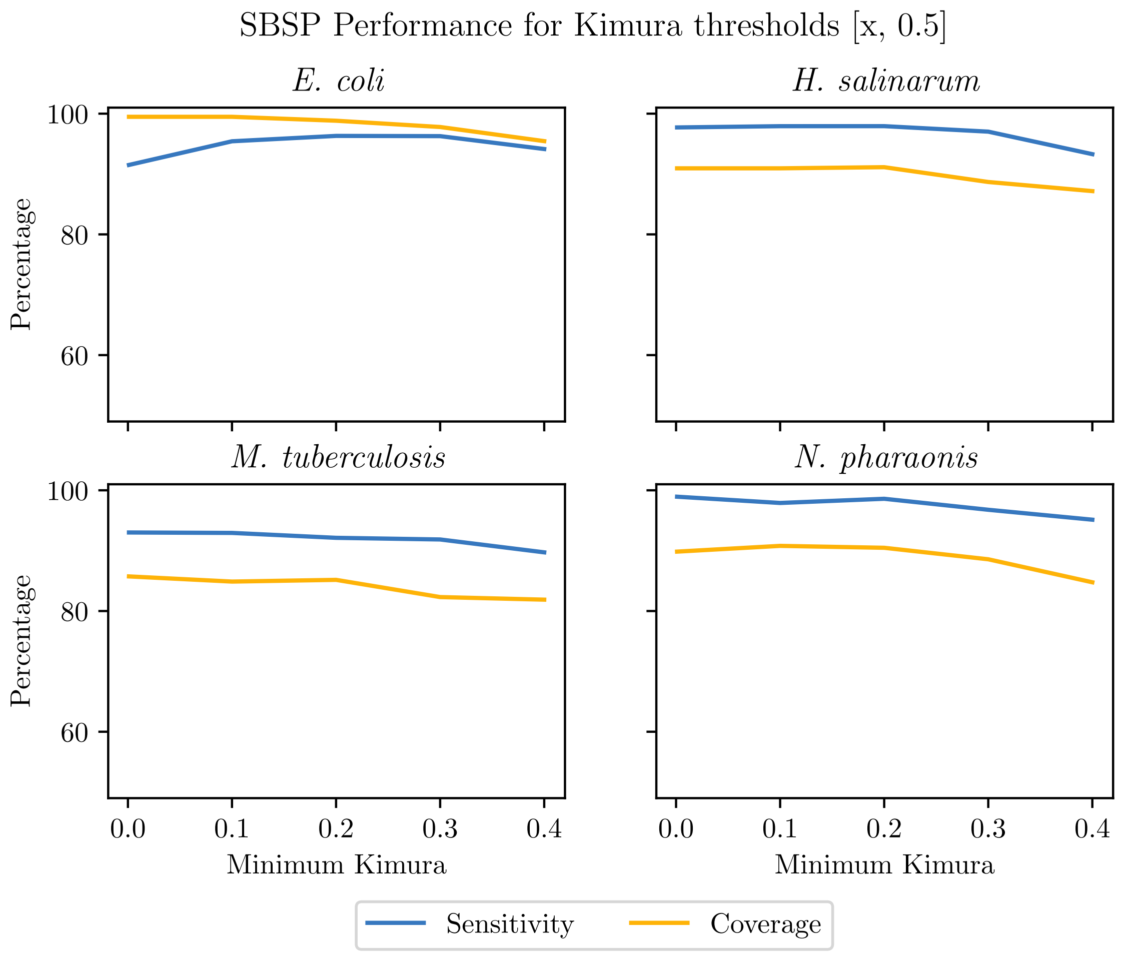


Figure 13: The effect of changing the minimum Kimura threshold on SBSP's sensitivity and coverage rates. The maximum Kimura threshold is fixed to 0.5, and $x \in \{0.001, 0.1, 0.2, 0.3, 0.4\}$

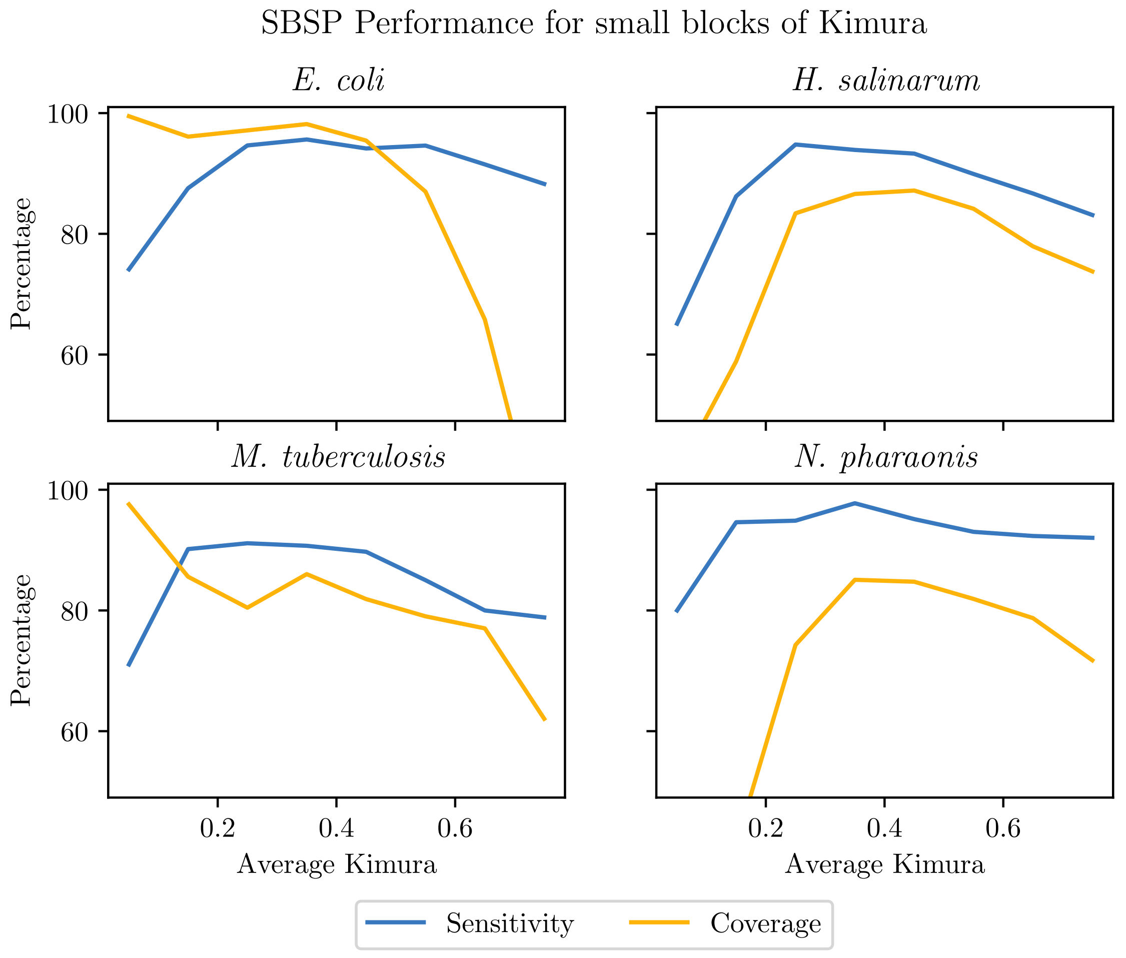


Figure 14: The performance of SBSP on small intervals of Kimura ranges: $[0.001, 0.1], [0.1, 0.2], [0.2, 0.3] \ldots [0.7, 0.8]$. The x-axis shows the mean Kimura of a block; e.g., for range $[a, b]$, the average is $(b + a)/2$

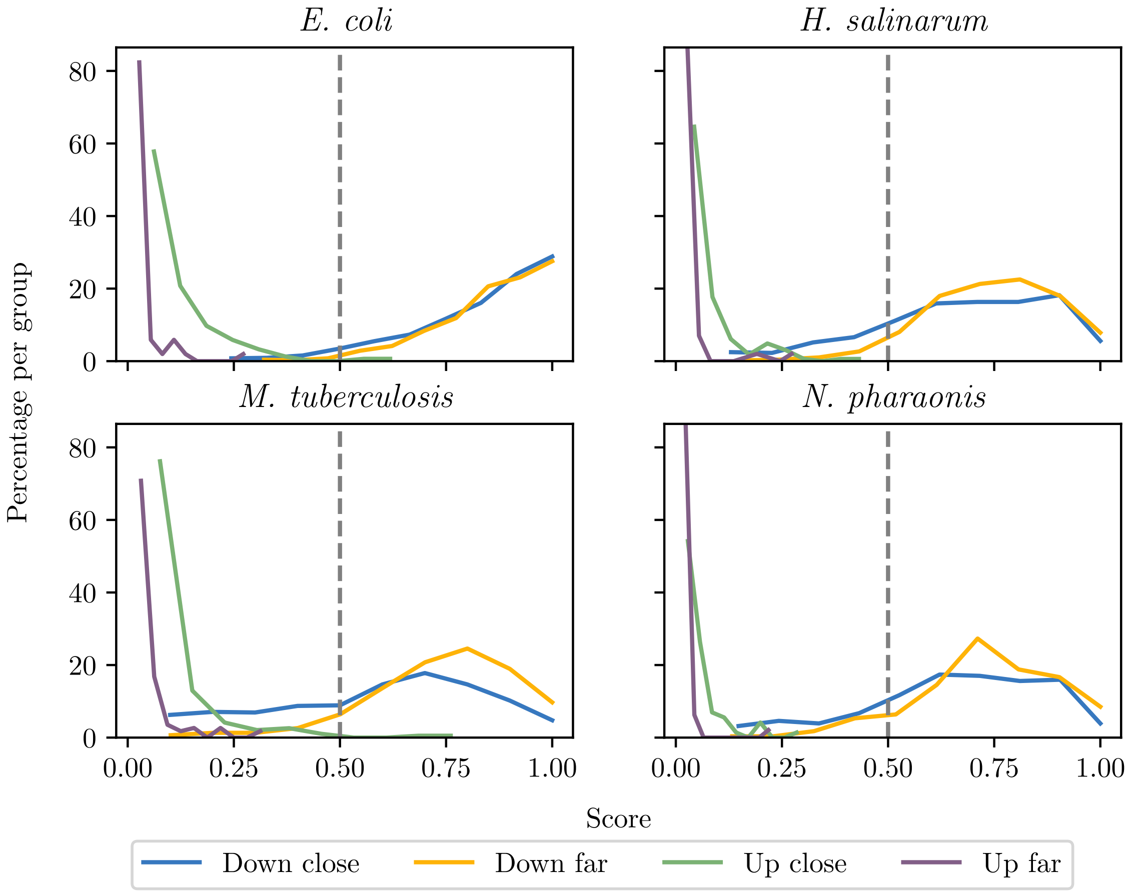


Figure 15: Distribution of block conservation scores in regions around verified starts

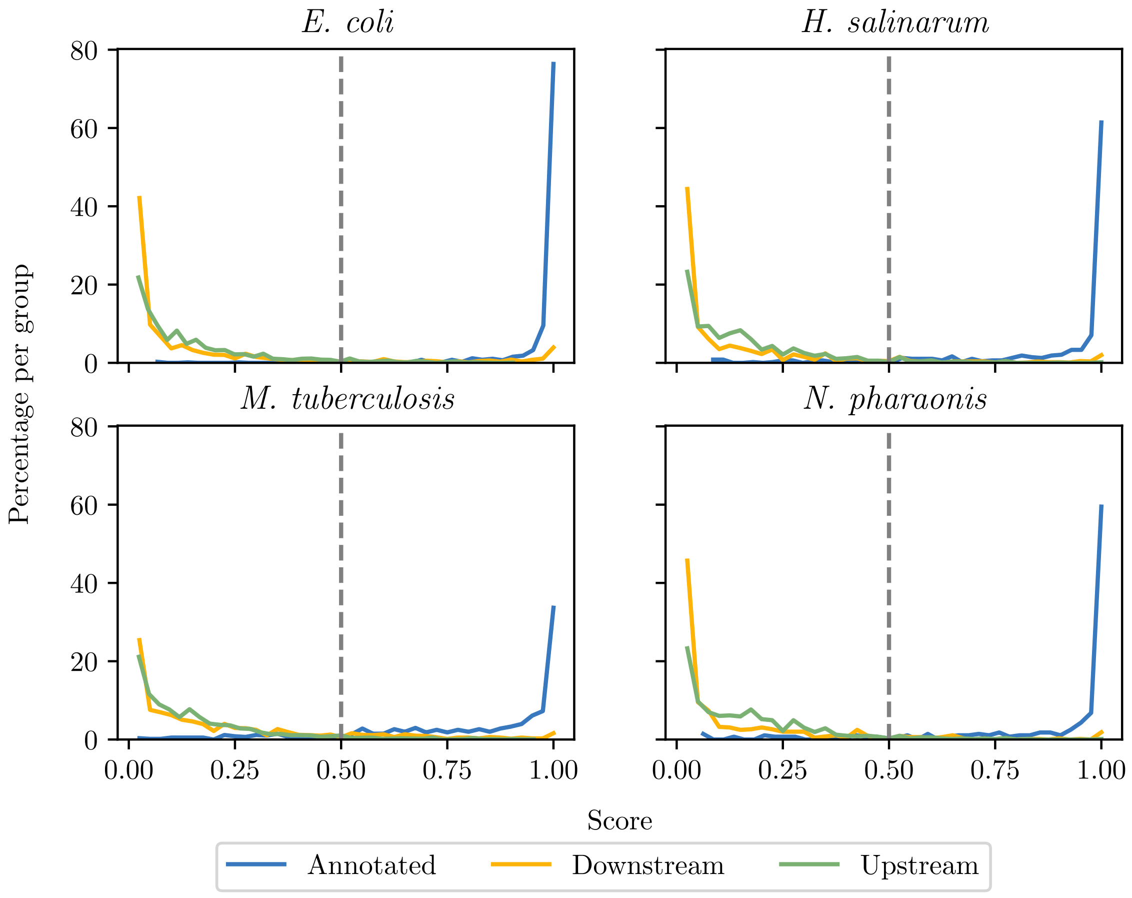


Figure 16: Distribution of 5' identity for verified starts, and upstream and downstream false 5' candidates