Binning metagenomic contigs with SemiBin2

Luis Pedro Coelho

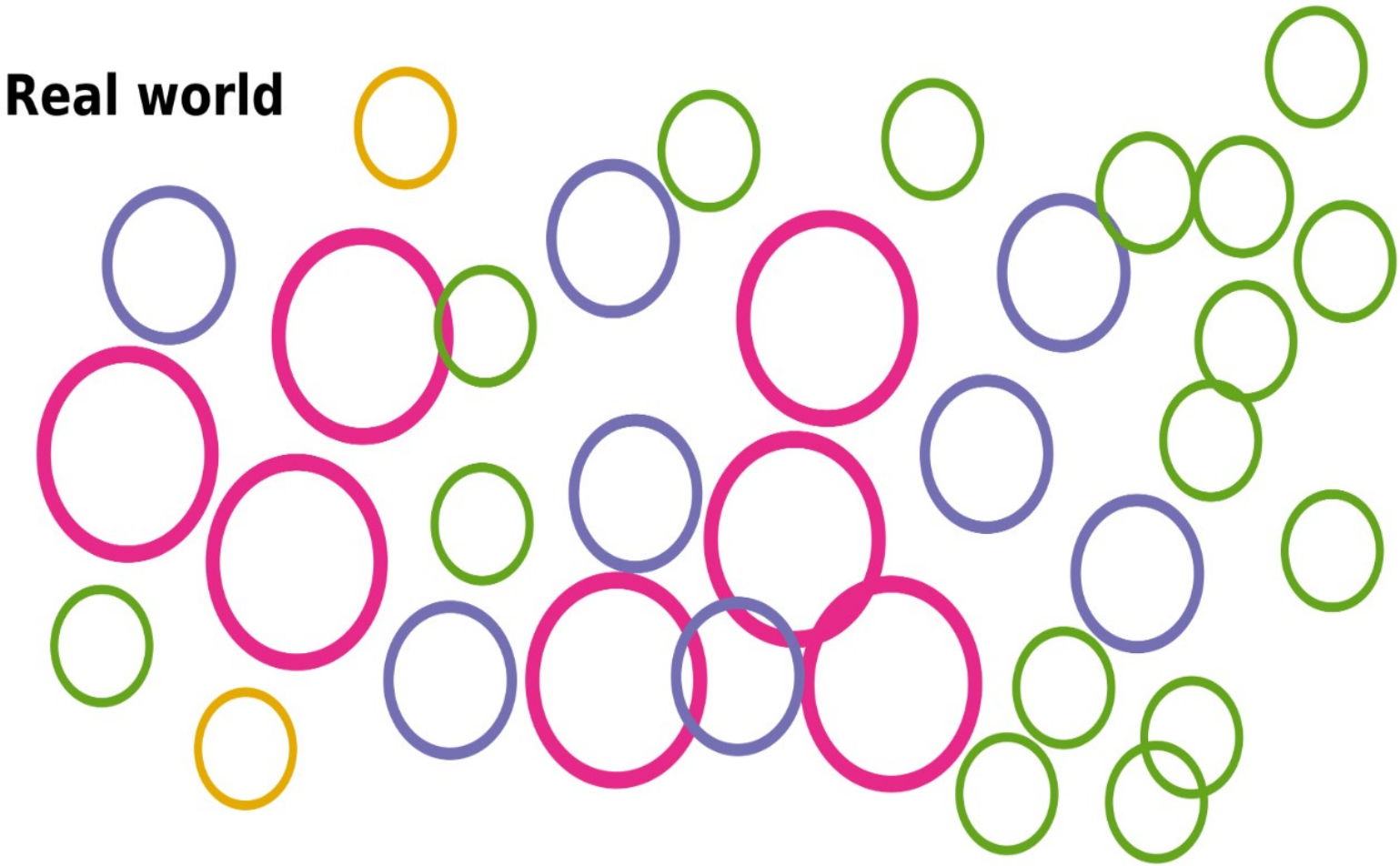
Big Data Biology Lab

luispedro@big-data-biology.org
@luispedrocoelho
@BigDataBiology

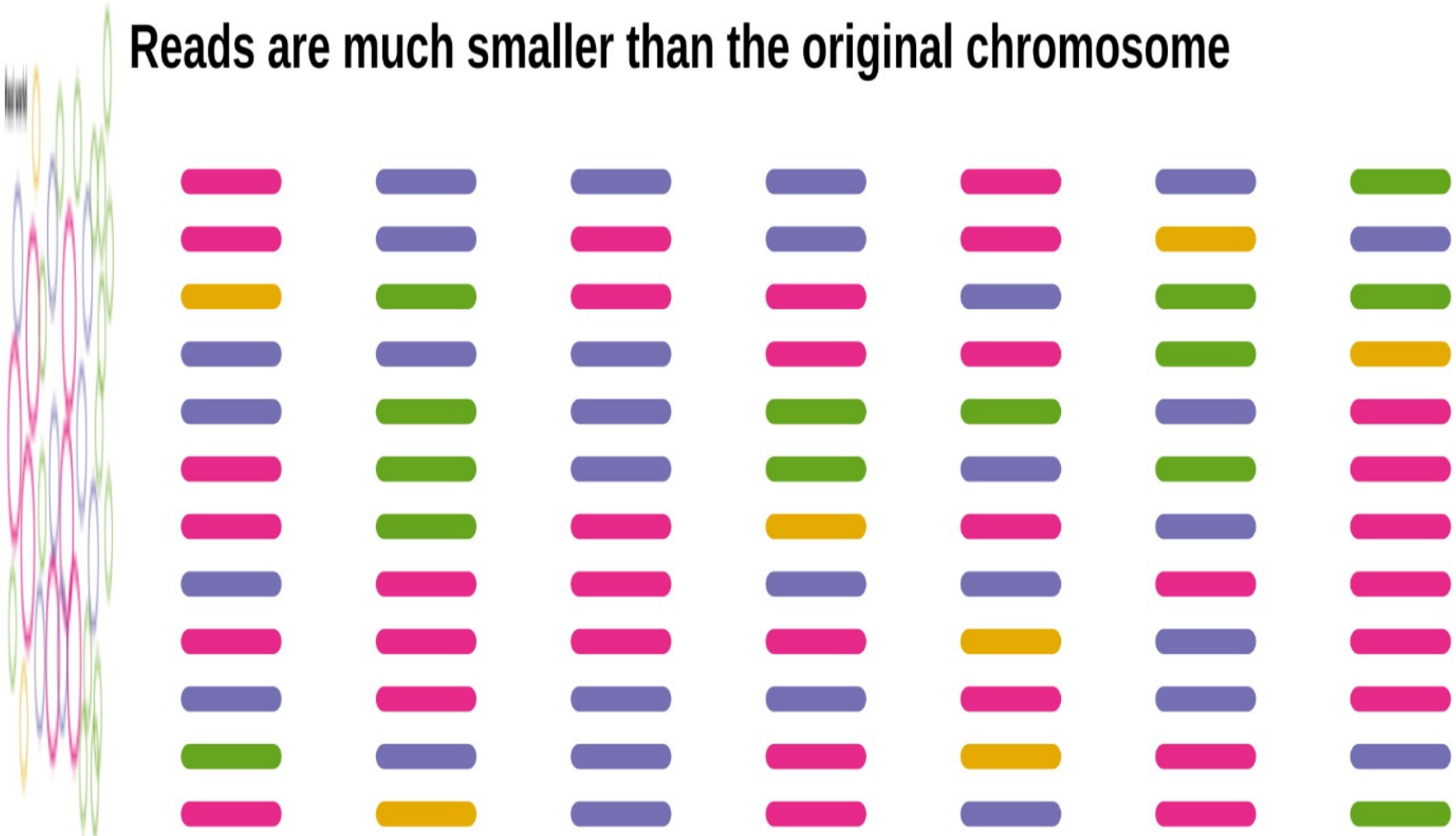
Centre for Microbiome Research

QUT

The metagenomics binning problem

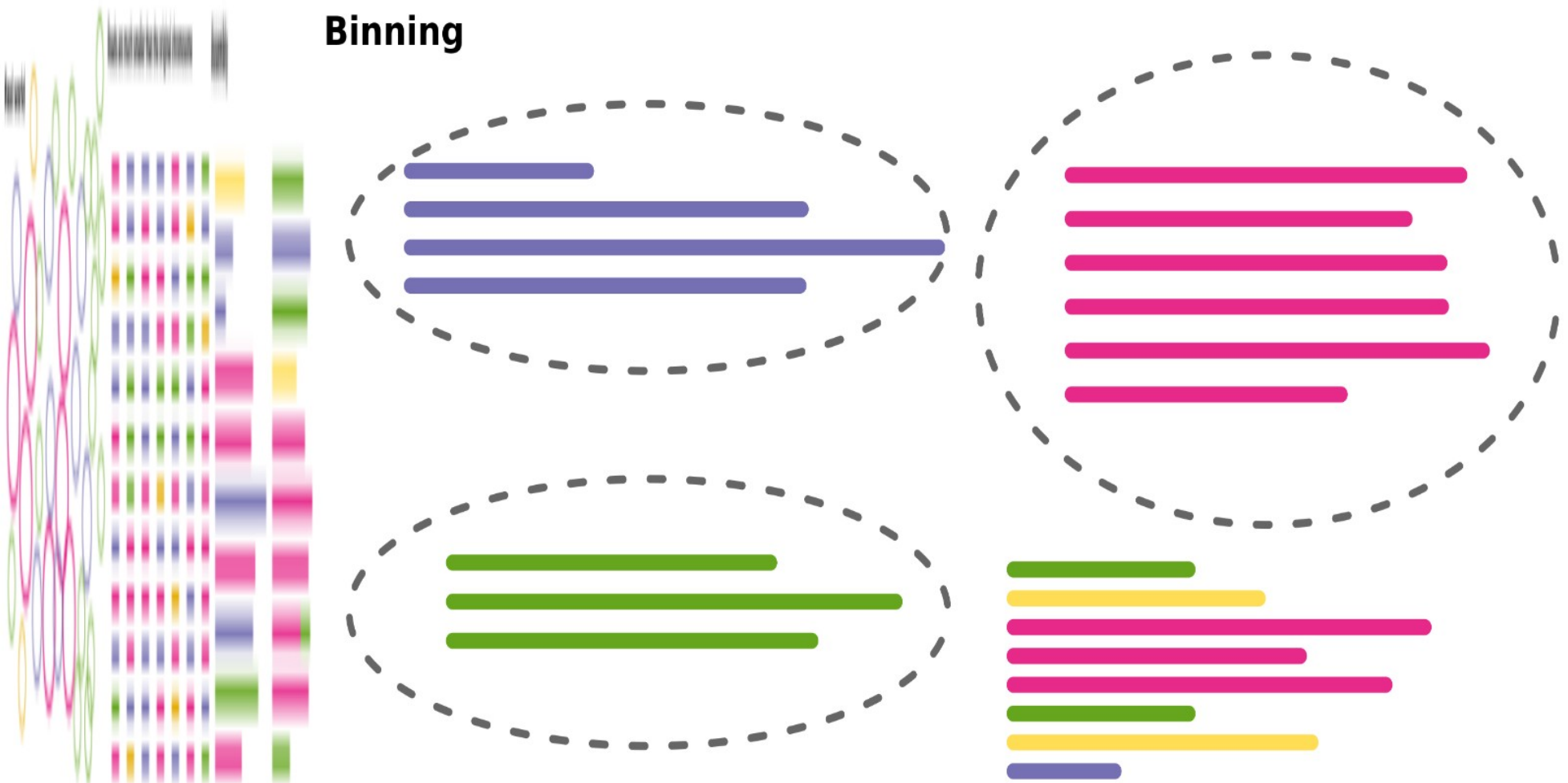


The metagenomics binning problem



The metagenomics binning problem

The metagenomics binning problem



The binning problem is a clustering problem

1. We have a set of contigs
2. We want to group them into bins

The binning problem is a clustering problem

1. We have a set of contigs
2. We want to group them into bins

Features that are widely used:

- abundance
- *k-mer* composition

The binning problem is a clustering problem

1. We have a set of contigs
2. We want to group them into bins

Features that are widely used:

- abundance
- *k-mer* composition

K-mers

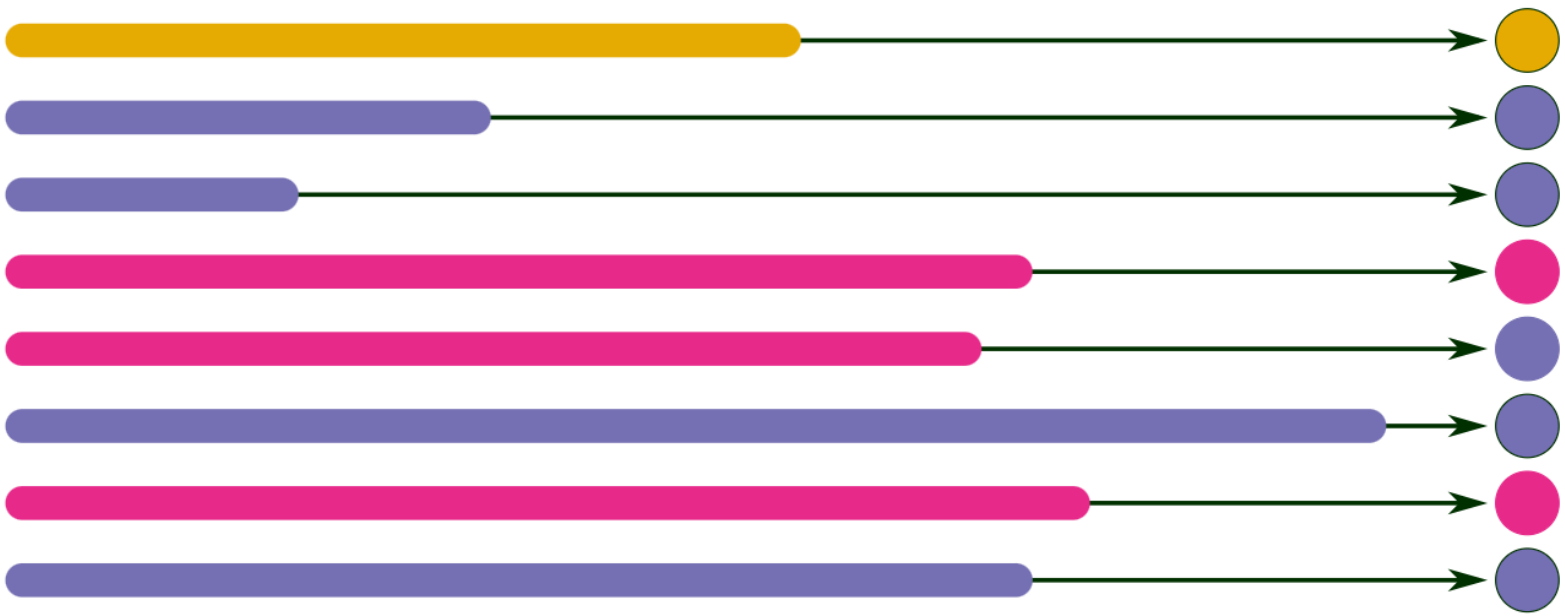
Contigs are DNA sequences, *e.g.*, `ATGTGCTAG` (typical length 3,000-10,000 bp). We can break them into *k-mers* of length *k* (*e*

- `ATGT`
- `TGTG`
- `GTGC`
- `TGCT`
- `GCTA`
- `CTAG`

We can then count the number of times each k-mer appears in each contig.

We can also classify contigs

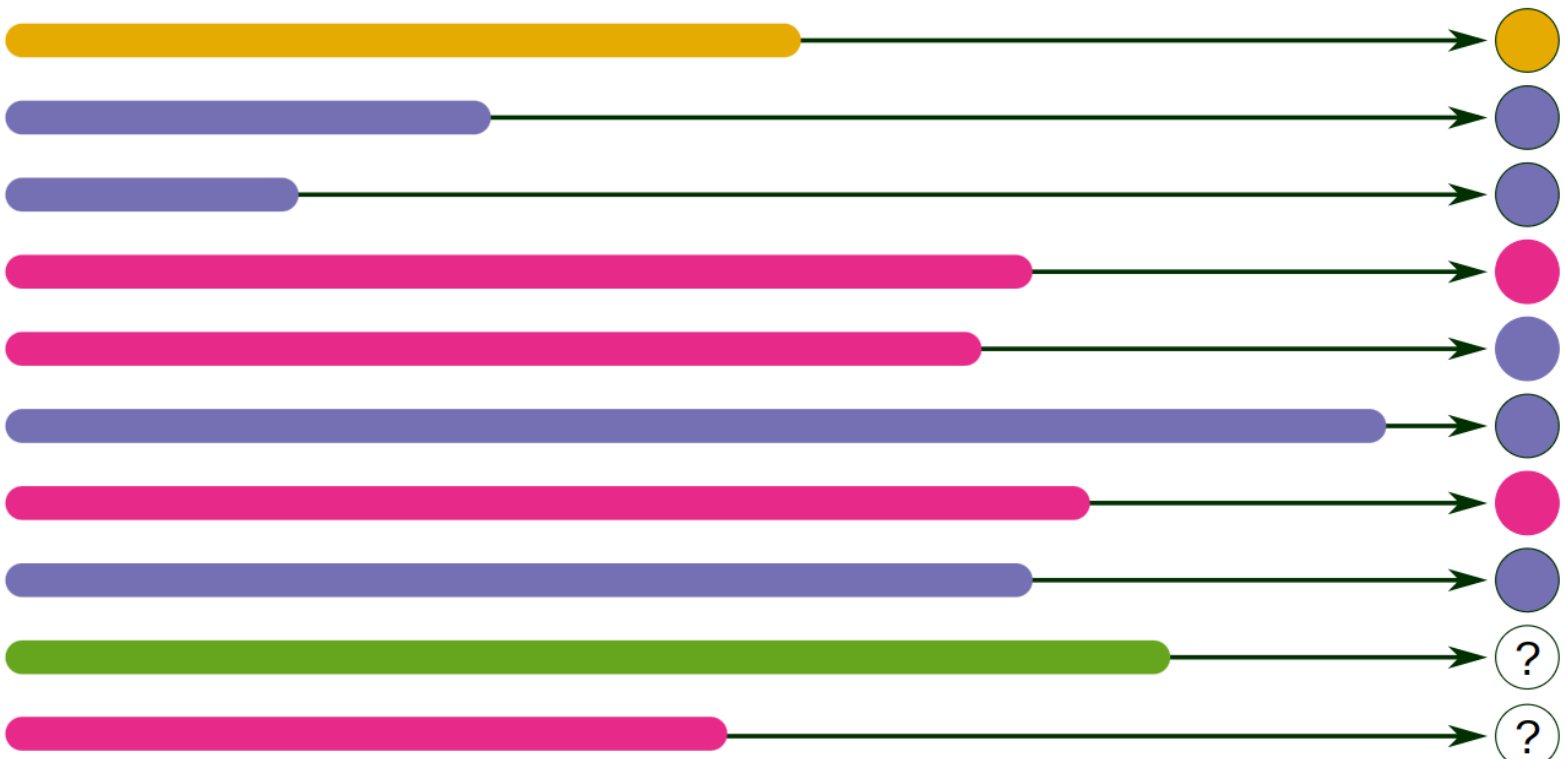
Classification of contigs



We can also classify contigs

We can also classify contigs

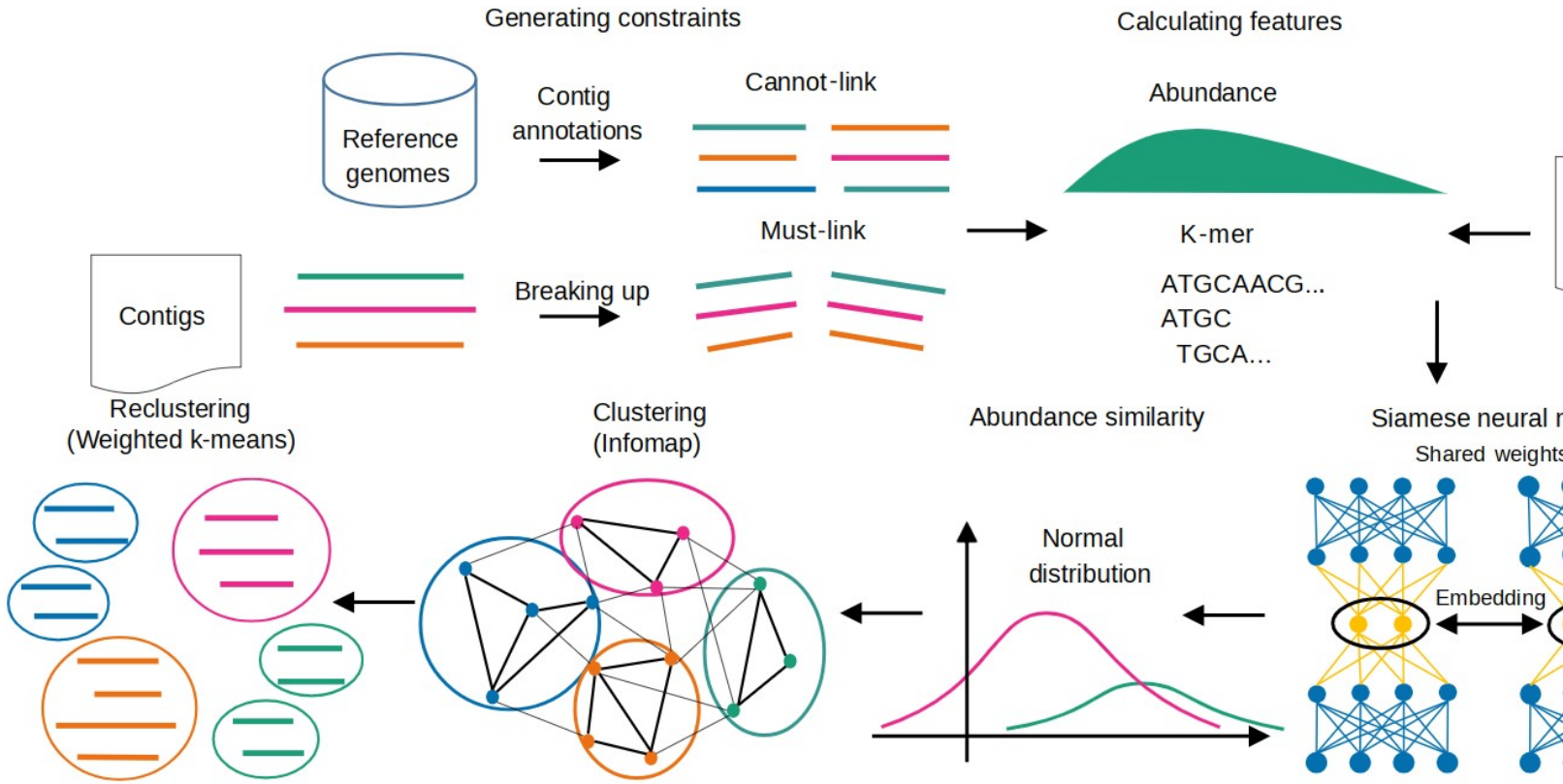
Classification of contigs



In the typical metagenome, the **majority** of contigs will not be classified

SemiBin1's approach was to use semi-supervised learning

Shaojun Pan



Contrastive learning

`input`: $X = \{ x_i \in \mathrm{R}^{136} \}$: *k-mer* & abundances

`output`: $X' = \{ x_i' = f(x_i) \in \mathrm{R}^{100} \}$: embedded

Goal: Learn a function f that maps X to X' such that must-linked contigs are mapped to close points in X' and cannot-linked co
mapped to far points in X'.

Contrastive learning

`input`: $X = \{ x_i \in \mathrm{R}^{136} \}$: *k-mer* & abundances

`output`: $X' = \{ x_i' = f(x_i) \in \mathrm{R}^{100} \}$: embedded

Goal: Learn a function *f* that maps *X* to *X'* such that must-linked contigs are mapped to close points in *X'* and cannot-linked co
mapped to far points in *X'*.

$\texttt{Loss}^{+} = \Sigma_{ij} \|x_i' - x_j'\|^2$ (for all *i* and *j* that are must-linked)

Contrastive learning

`input`: $X = \{ x_i \in \mathrm{R}^{136} \}$: *k-mer* & abundances

`output`: $X' = \{ x_i' = f(x_i) \in \mathrm{R}^{100} \}$: embedded

Goal: Learn a function f that maps X to X' such that must-linked contigs are mapped to close points in X' and cannot-linked co
mapped to far points in X'.

$\texttt{Loss}^{+} = \Sigma_{ij} \|x_i' - x_j'\|^2$ (for all i and j that are must-linked)

$\texttt{Loss}^{-} = \Sigma_{ij} \texttt{max}(0, 1 - \|x_i' - x_j'\|^2)$ (for all i and j that are cannot-linked)

Contrastive learning

`input`: $X = \{ x_i \in \mathrm{R}^{136} \}$: *k-mer* & abundances

`output`: $X' = \{ x_i' = f(x_i) \in \mathrm{R}^{100} \}$: embedded

Goal: Learn a function f that maps X to X' such that must-linked contigs are mapped to close points in X' and cannot-linked co
mapped to far points in X'.

$\texttt{Loss}^{+} = \Sigma_{ij} \|x_i' - x_j'\|^2$ (for all i and j that are must-linked)

$\texttt{Loss}^{-} = \Sigma_{ij} \texttt{max}(0, 1 - \|x_i' - x_j'\|^2)$ (for all i and j that are cannot-linked)

$\texttt{Loss} = \texttt{Loss}^{+} + \texttt{Loss}^{-}$

Contrastive learning

`input`: $X = \{ x_i \in \mathrm{R}^{136} \}$: *k-mer* & abundances

`output`: $X' = \{ x_i' = f(x_i) \in \mathrm{R}^{100} \}$: embedded

Goal: Learn a function *f* that maps *X* to *X′* such that must-linked contigs are mapped to close points in *X′* and cannot-linked co
mapped to far points in *X′*.

$\texttt{Loss}^{+} = \Sigma_{ij} \|x_i' - x_j'\|^2$ (for all *i* and *j* that are must-linked)

$\texttt{Loss}^{-} = \Sigma_{ij} \texttt{max}(0, 1 - \|x_i' - x_j'\|^2)$ (for all *i* and *j* that are cannot-linked)

$\texttt{Loss} = \texttt{Loss}^{+} + \texttt{Loss}^{-}$

Form of *f*: *f* is a simple multilayer network (see manuscript for details).

Learned using standard techniques.

See also (Śmieja, Struski, & Figuei

SemiBin1 outperforms other binning methods

- We compared against other binning methods
- We used CAMI data (simulated) and real data
- We counted the number of high-quality MAGs produced (checkM& GUNC)

Real datasets used

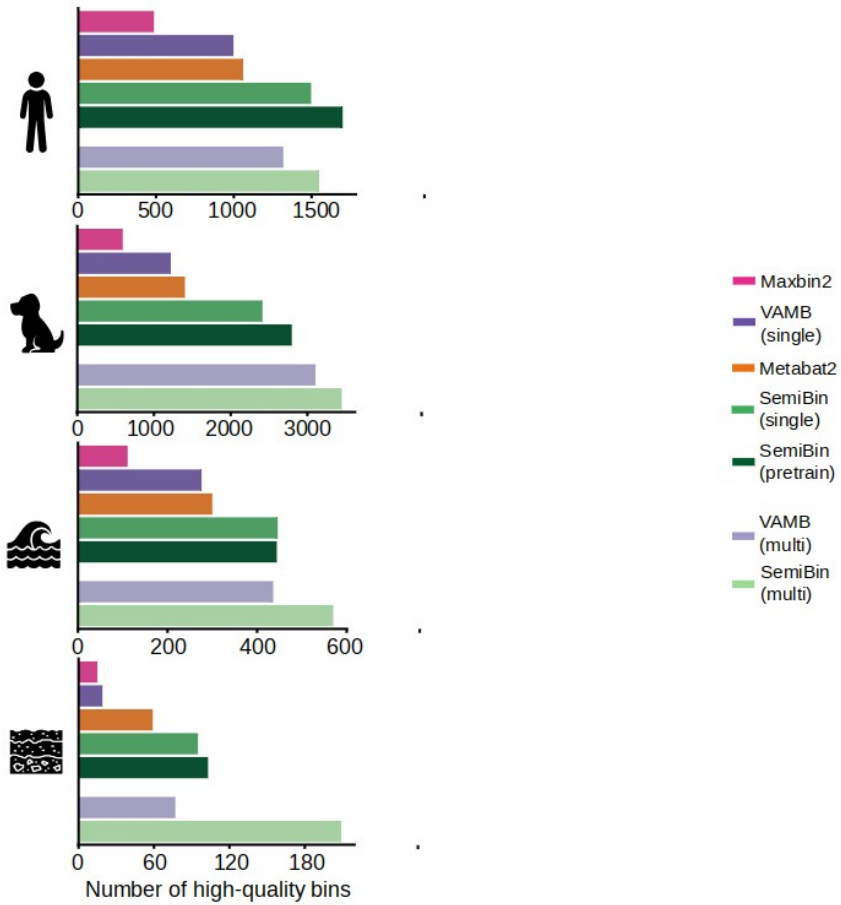
- Human gut (82 samples)
- Dog gut (129 samples)
- Surface marine (109 samples)
- Soil (101 samples)

SemiBin1 outperforms other binning methods

- We compared against other binning methods
- We used CAMI data (simulated) and real data
- We counted the number of high-quality MAGs produced (checkM& GUNC)

Real datasets used

- Human gut (82 samples)
- Dog gut (129 samples)
- Surface marine (109 samples)
- Soil (101 samples)

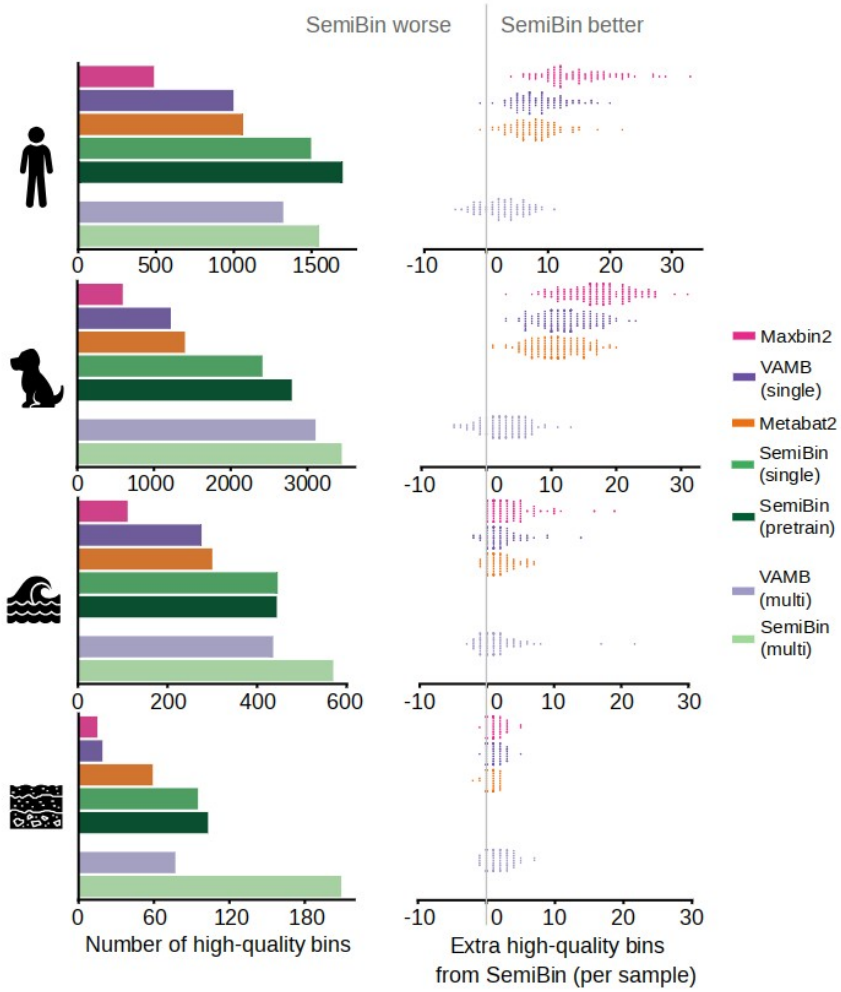


SemiBin1 outperforms other binning methods

- We compared against other binning methods
- We used CAMI data (simulated) and real data
- We counted the number of high-quality MAGs produced (checkM& GUNC)

Real datasets used

- Human gut (82 samples)
- Dog gut (129 samples)
- Surface marine (109 samples)
- Soil (101 samples)



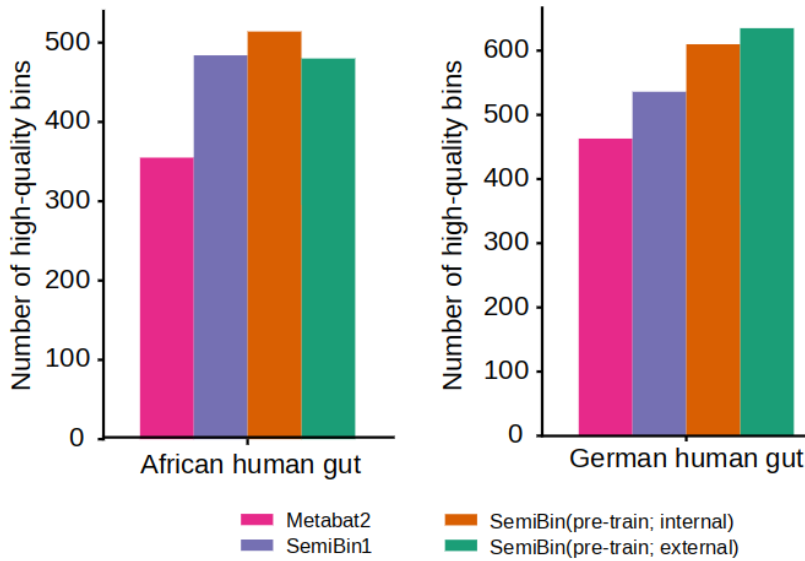
Learning a new model is resource intensive

- SemiBin uses a learned model
- Generating the data for training is expensive (mmseqs2 is slow)
- Learning is slow (a GPU helps, but few people have a GPU cluster)

		Single-sample		
		Human gut	Dog gut	Ocean
Computing features	Time(min)	5	9	12
	Memory(MB)	923	736	1,675
Generating cannot-link	Time(min)	88	81	114
	Memory(MB)	39,070	37,904	46,091
Training(CPU)	Time(min)	181	209	222
	Memory(MB)	2,497	2,373	3,211
Training(GPU)	Time(min)	34	36	45
	Memory(MB)	4,487	4,355	5,222
Binning	Time(min)	2	2	3
	Memory(MB)	4,501	3,641	7,622

SemiBin models can be transferred between datasets

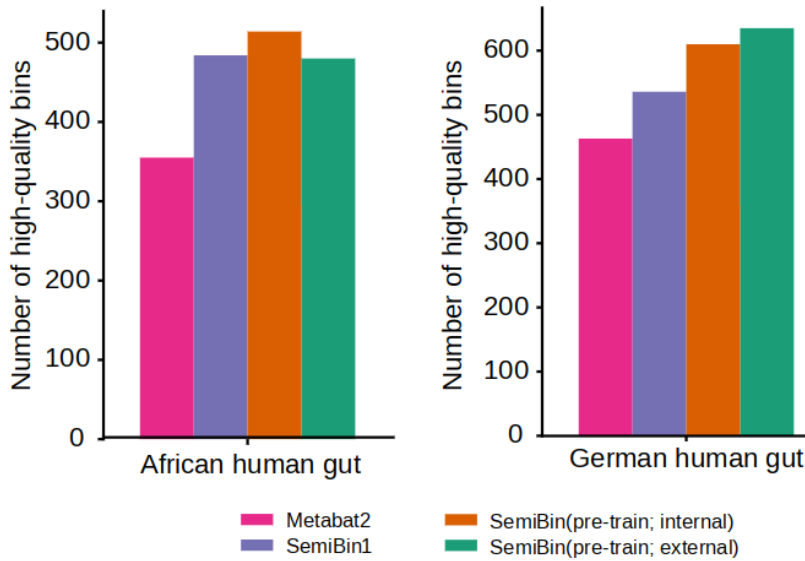
- We tried transferring between datasets
- From a Westernized human gut dataset to another
- From a Westernized human gut dataset to a non-Westernized human gut dataset



Human gut models work well even across geographies

SemiBin models can be transferred between datasets

- We tried transferring between datasets
- From a Westernized human gut dataset to another
- From a Westernized human gut dataset to a non-Westernized human gut dataset



Human gut models work well even across geographies

We make pre-computed models available

We provide pre-computed models for 10 habitats from the GMGC dataset (+ *chicken caecum* contributed by Florian Plaza Oñate)

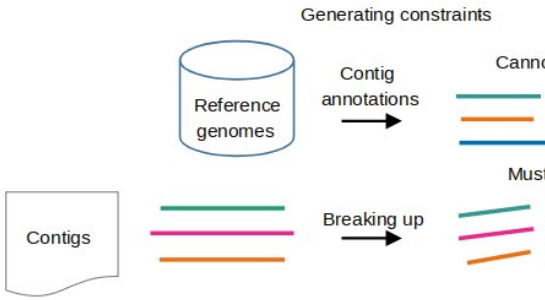
SemiBin1 returns good results, but can be slow

SemiBin1

1. SemiBin1 returns good results: better than the state-of-the-art across a broad range of metrics
2. It is slow to train (and requires a lot of memory)
3. For single-sample binning, we can reuse models

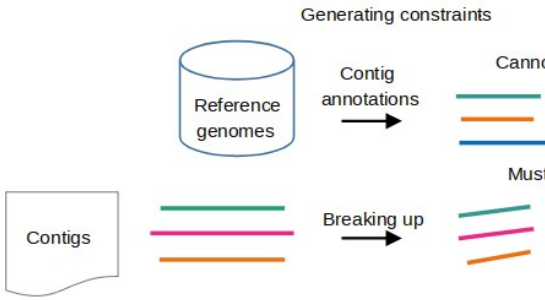
From Semi- to self-supervised learning

- In SemiBin1, must-links were self-supervised
- In SemiBin1, cannot-links were from a reference taxonomy

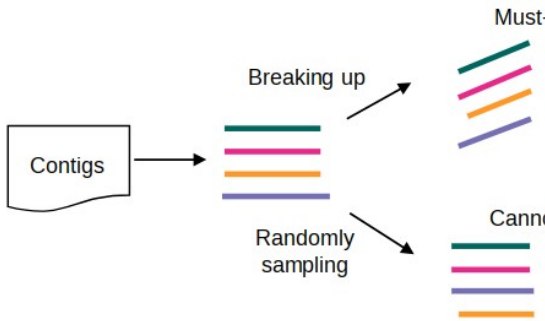


From Semi- to self-supervised learning

- In SemiBin1, must-links were self-supervised
- In SemiBin1, cannot-links were from a reference taxonomy

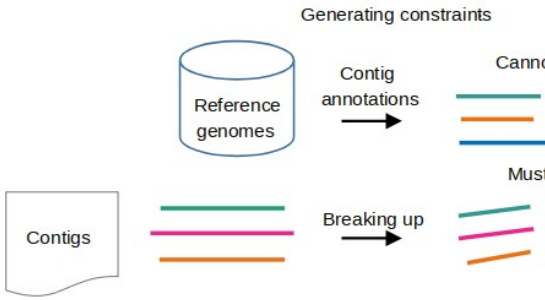


Use random sampling to generate cannot-links



From Semi- to self-supervised learning

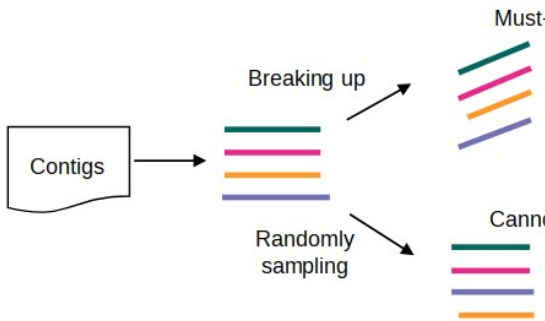
- In SemiBin1, must-links were self-supervised
- In SemiBin1, cannot-links were from a reference taxonomy



Use random sampling to generate cannot-links

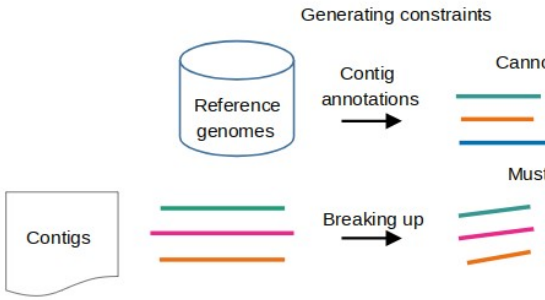
Random sampling advantages

1. Random sampling is faster than MMSeqs2
2. It has no database biases



From Semi- to self-supervised learning

- In SemiBin1, must-links were self-supervised
- In SemiBin1, cannot-links were from a reference taxonomy



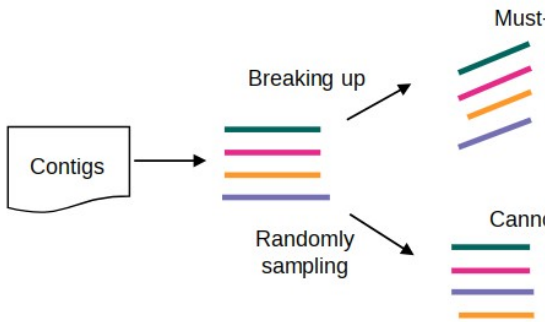
Use random sampling to generate cannot-links

Random sampling advantages

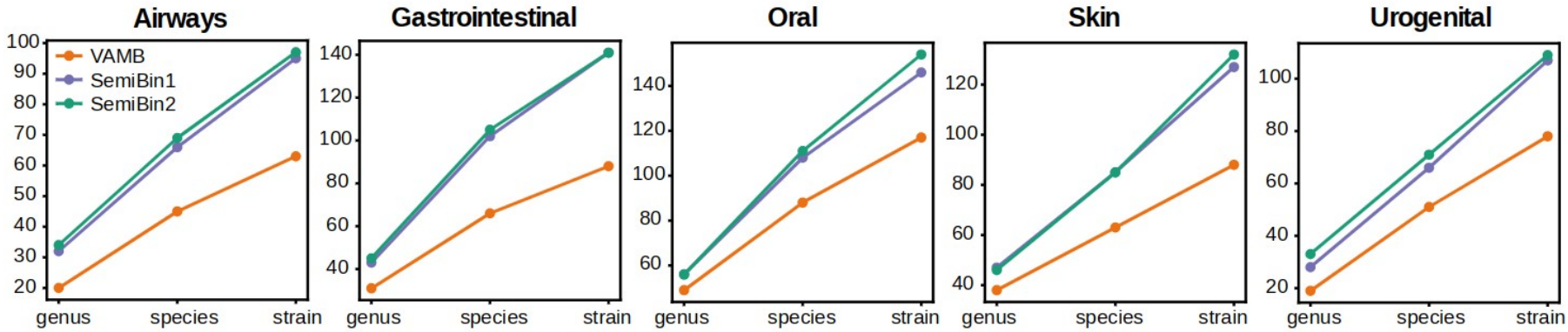
1. Random sampling is faster than MMSeqs2
2. It has no database biases

Random sampling disadvantages

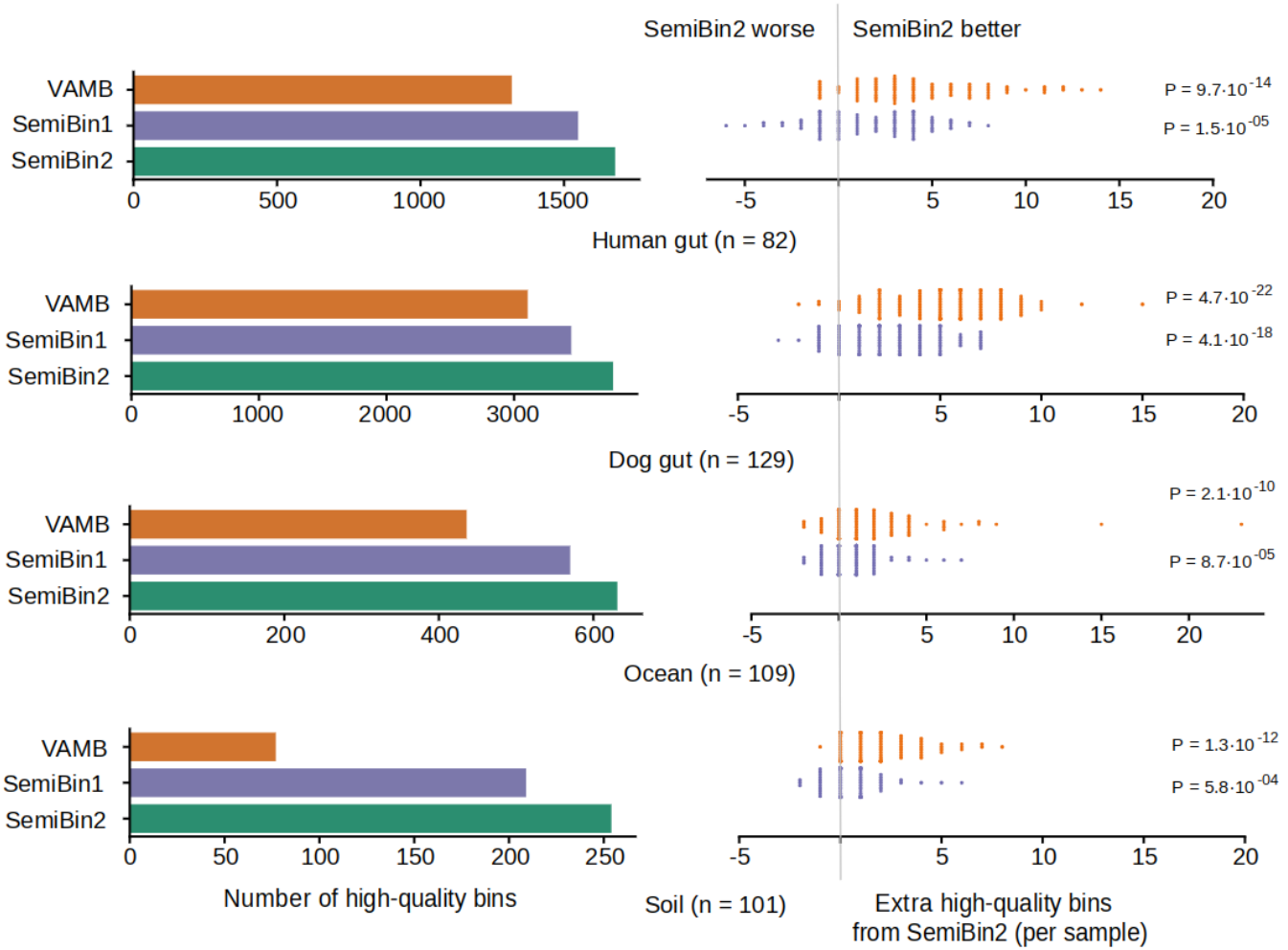
1. Random sampling introduces errors (naturally)



The proof of the pudding is in the eating I: simulated data



The proof of the pudding is in the eating II: real data



We also wanted to extend SemiBin to long-reads

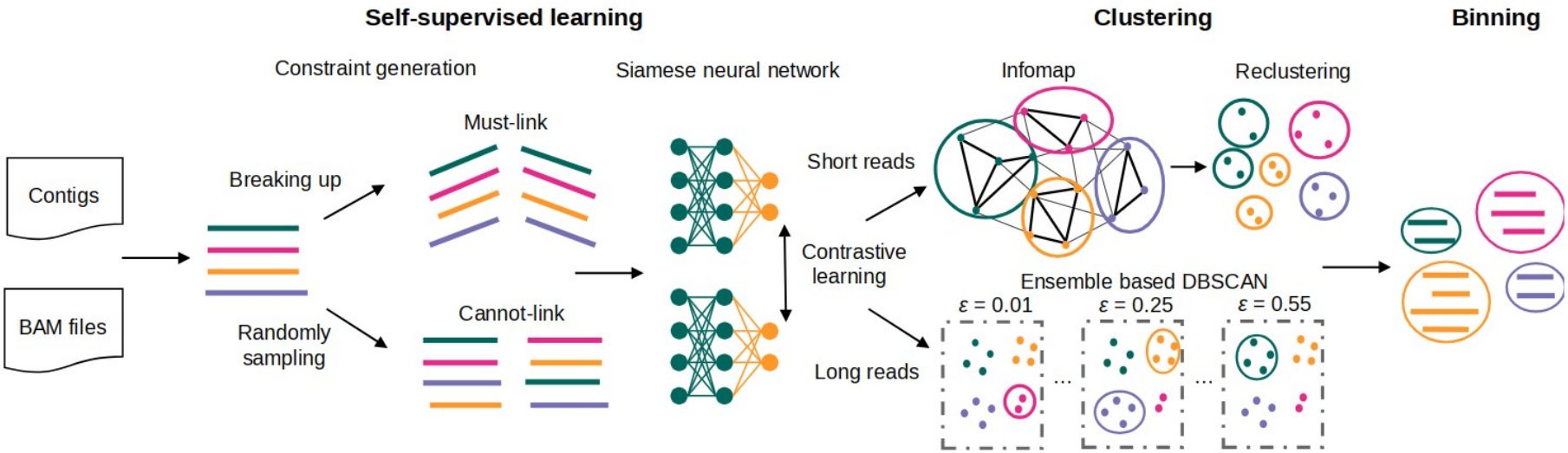
- Everything I presented so far was for short-reads (Illumina)
- We wanted to extend SemiBin to long-reads (PacBio, Oxford Nanopore)
- You could always use the system, but we did not test it

We also wanted to extend SemiBin to long-reads

- Everything I presented so far was for short-reads (Illumina)
- We wanted to extend SemiBin to long-reads (PacBio, Oxford Nanopore)
- You could always use the system, but we did not test it

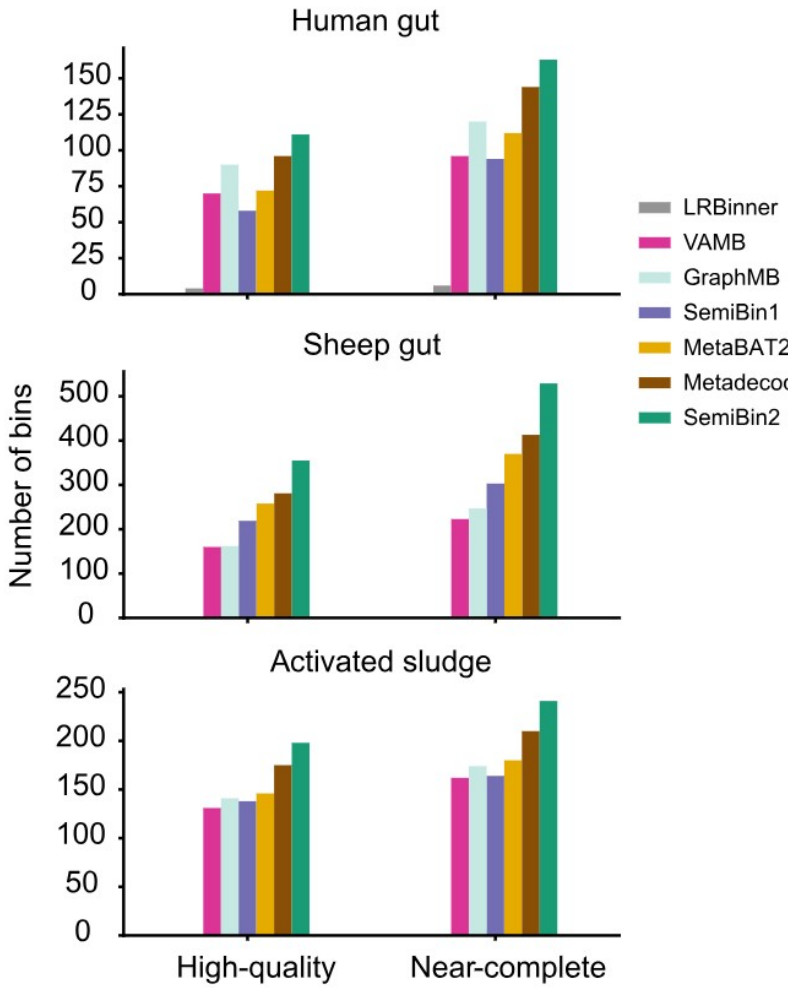
		#contig	Average length	N50
Sample A	mNGS	277,884	1,865	8,778
	PacBio-HiFi	2,575	99,842	269,406
	Nanopore	3,414	65,685	199,639
Sample B	mNGS	146,857	2,402	28,310
	PacBio-HiFi	1,447	157,228	1,081,788
	Nanopore	1,795	123,512	658,841
Sample C	mNGS	170,813	1,956	18,485
	PacBio-HiFi	822	171,215	1,270,126
	Nanopore	1,370	126,533	891,411

SemiBin2 introduces an alternative clustering method (after embedding



SemiBin2 surpasses alternatives for long-read binning

- Three datasets: human gut, sheep gut, and activated sludge
- Some of the alternatives (like SemiBin1) are not designed for long-reads
- We used CheckM2 & GUNC to evaluated the results
- SemiBin2 is also best using checkM1, but the two tools share some of the same marker genes (so evaluation is partly circular)



Summary

SemiBin1

- Semi-supervised binning of metagenomic contigs
- Using a combination of self-supervised learning & reference training
- Better results across a range of habitats

SemiBin2

- Fully self-supervised binning of metagenomic contigs
- Similar (or better) results as SemiBin1 at a fraction of the cost
- Add support for long reads

Ongoing & Future work

- Applying SemiBin to metagenomic datasets (particularly those with long reads)
- Extending to micro-eukaryotes
- Exploring how to make the preprocessing faster (particularly when working with many samples)

Acknowledgements

Fudan University co-authors

- **Shaojun Pan**
- Chengkai Zhu
- Xing-Ming Zhao+ (co-corresponding)

SemiBin users

- Florian Plaza Oñate (INRAE)
- Bérénice Batut (University of Freiburg)
- Rhys Newell (Microba)
- Silas Kieser (Nestlé Health Sciences)
- Marco Gabrielli (Eawag)
- ...

Funding

Thank you

Find out more

Open office hours

- Dec 10 @ 9.30am UTC: Register now
- Dec 12 @ 9.30pm UTC: Register now

About SemiBin: https://semibin.readthedocs.io

SemiBin manuscripts:

Pan, S.; Zhu, C.; Zhao, XM.+; Coelho, LP.+ A deep siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments. *Nat Commun* **13,** 2326 (2022). https://doi.org/10.1038/s41467-022-29843-y Pan, S.; Zhao, XM.+; Coelho, LP.+ SemiBin2: self-supervised contrastive learning leads to better MAGs for short- and long-read sequencing. *Bioinformatics* **39** (2023), i21–i29. https://doi.org/10.1093/bioinformatics/btad209

About our group

- Main website: https://big-data-biology.org
- YouTube: https://youtube.com/@BigDataBiolo
- Twitter: https://twitter.com/BigDataBiology
- Quarterly Newsletter: https://bigdatabiology.substack.com/.

Email us!

- luispedro@big-data-biology.org

Join us in Brisbane!

- PhD studentships open

Thank you

The semi-supervised step is important

We first embed and then cluster.

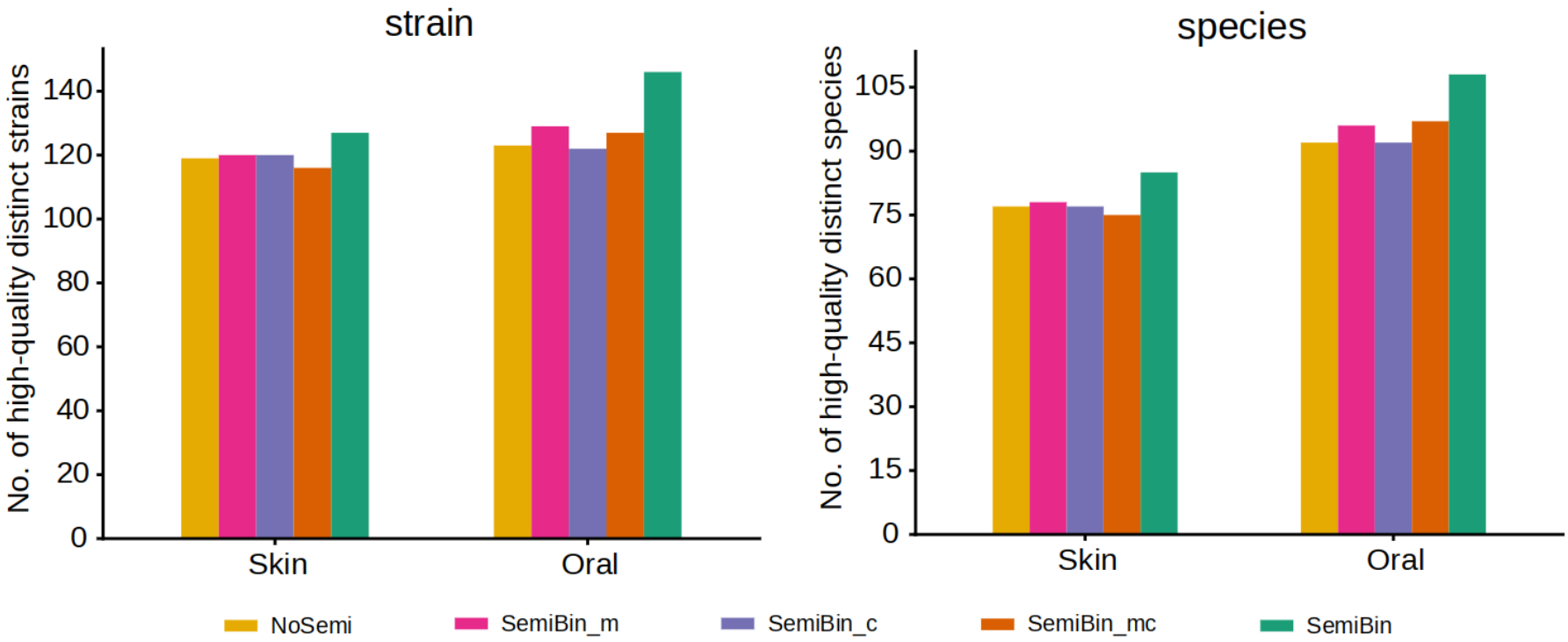
The semi-supervised step is important

Maybe we just used a better clustering method?

The semi-supervised step is important

Or maybe the supervised information should be used directly?

The semi-supervised step is important



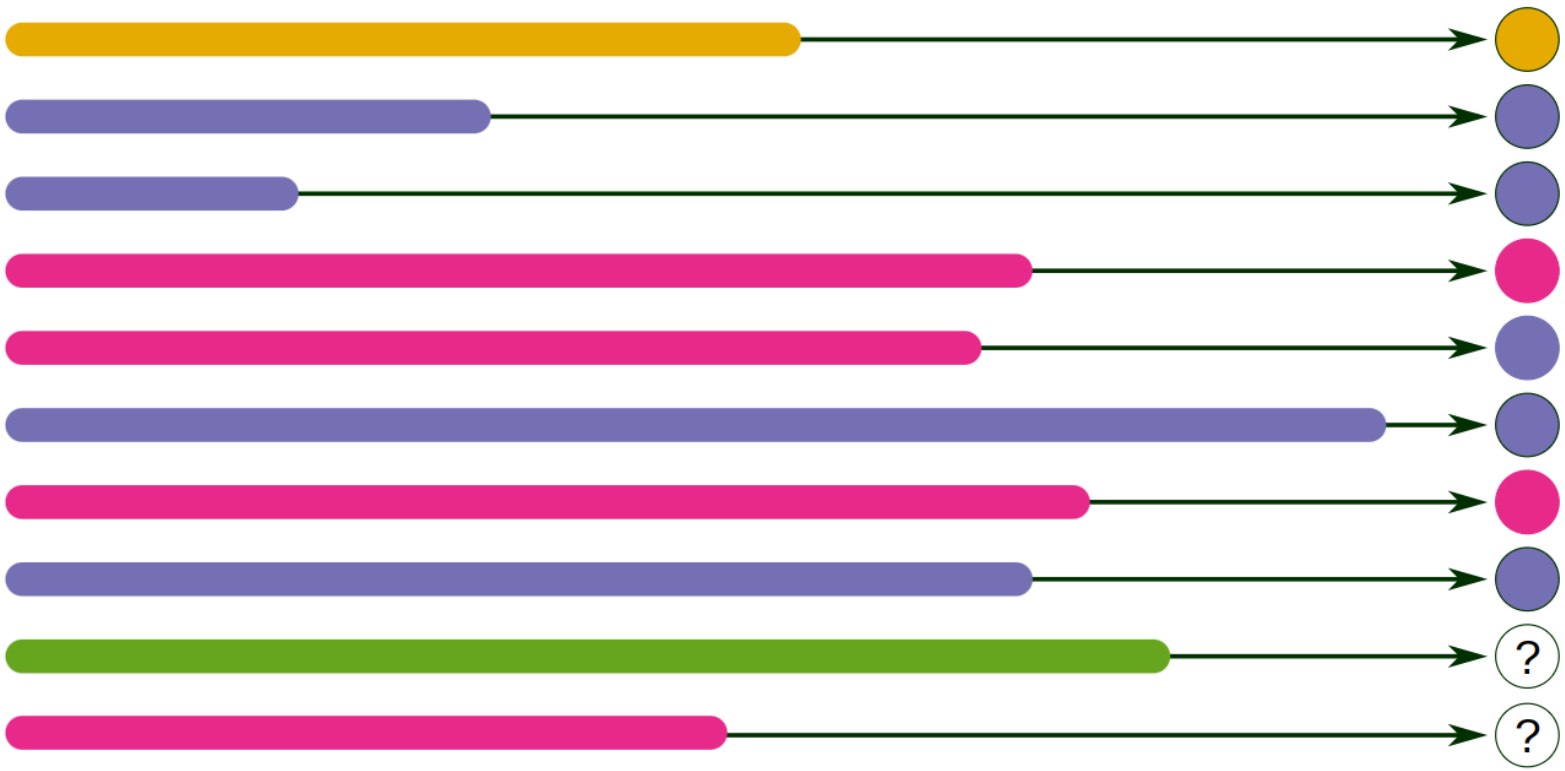
There is real learning going on

Maybe we are just recovering strains from species that are already known?

There is real learning going on

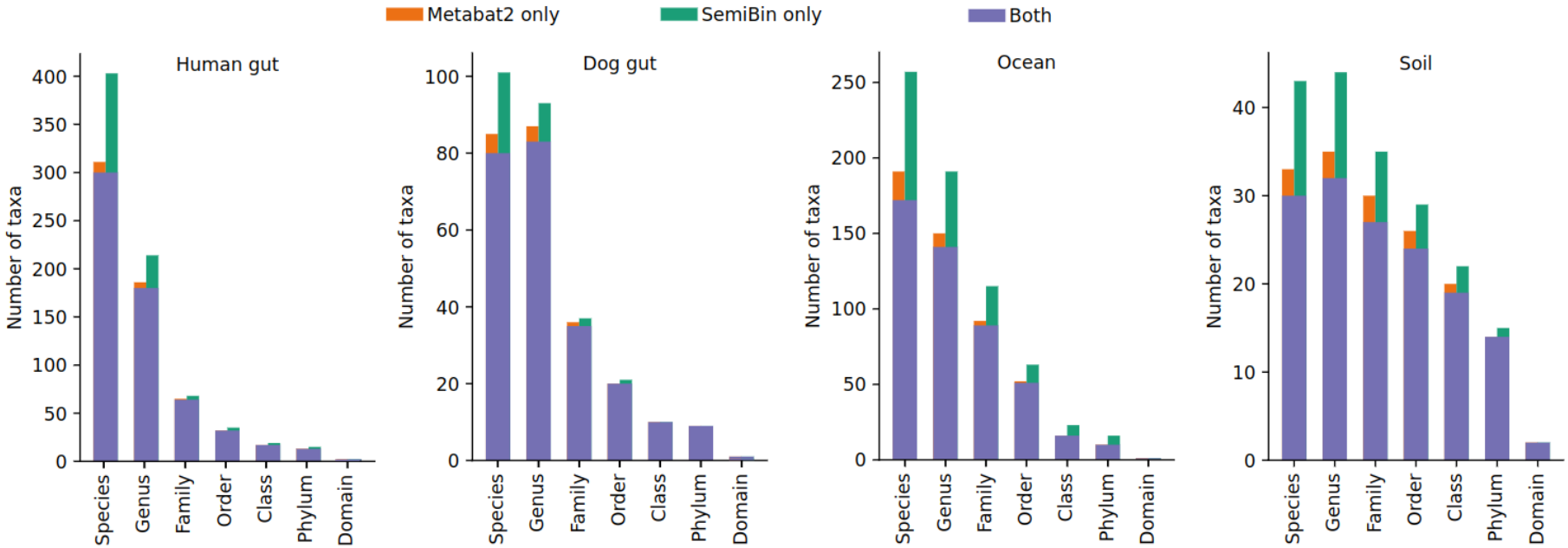
Maybe we are just recovering strains from species that are already known?

Classification of contigs



There is real learning going on

We recover more **taxonomic diversity** overall

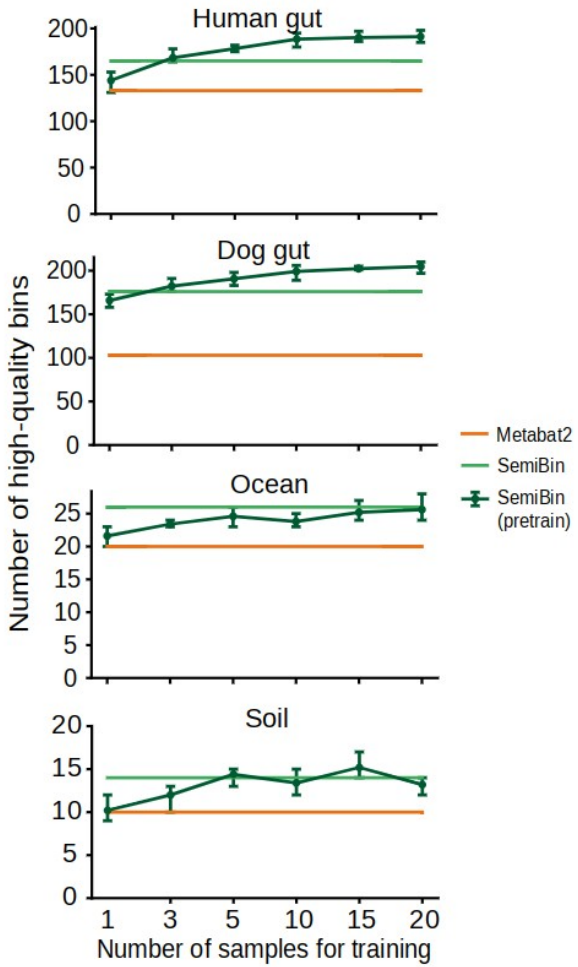


SemiBin models can be transferred between samples

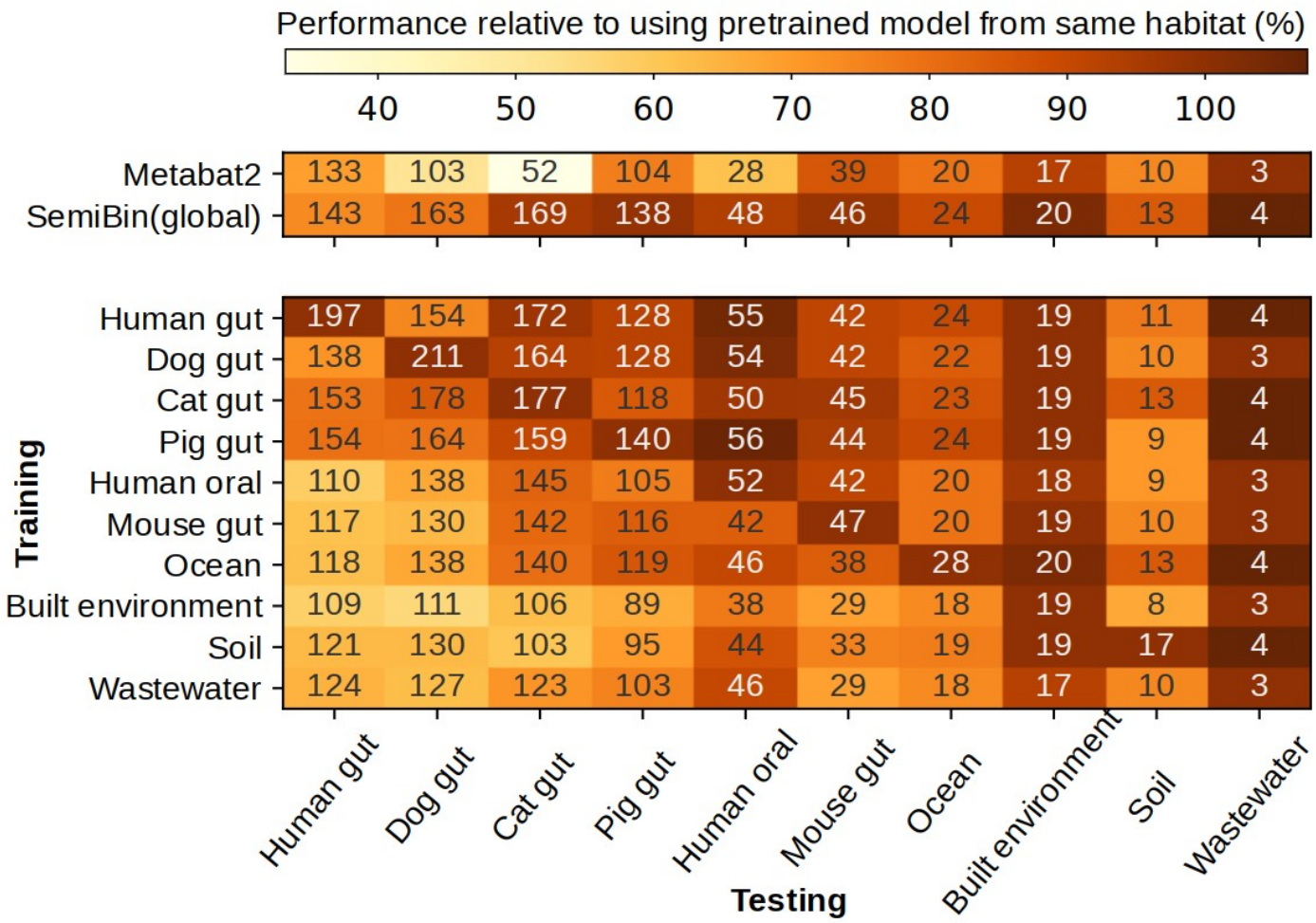
For a large dataset, maybe we can learn a single model instead of one per sample?

SemiBin models can be transferred between samples

For a large dataset, maybe we can learn a single model instead of one per sample?



SemiBin models work best within similar habitats



Models can be learned in *one dataset* and **transferred** to another, but it works best *within the same habitat*

How do the random samples compare to the reference taxonomy?

		Taxonomic annotation			Randomly sampling		
		Nr. Cannot	Acc %	Nr. Covered	Nr. Cannot	Acc %	Nr. Covered
mNGS (multi-sample binning)	Airways	35,772,061	99.64	885	39,007,000	96.81	1085
	GI	25,628,395	98.67	291	28,843,000	92.23	511
	Oral	37,794,493	99.75	1,135	40,000,000	98.19	1,393
	Skin	21,908,477	99.47	614	25,814,000	95.71	822
	Urog	15,229,369	98.61	271	18,710,000	93.03	408
PacBio (co-assembly binning)	Airways	4,000,000	99.96	806	4,000,000	99.68	934
	GI	4,000,000	99.74	128	4,000,000	97.13	281
	Oral	4,000,000	99.95	654	4,000,000	99.54	799
	Skin	4,000,000	99.92	569	4,000,000	99.39	725
	Urog	4,000,000	99.81	192	4,000,000	98.25	300

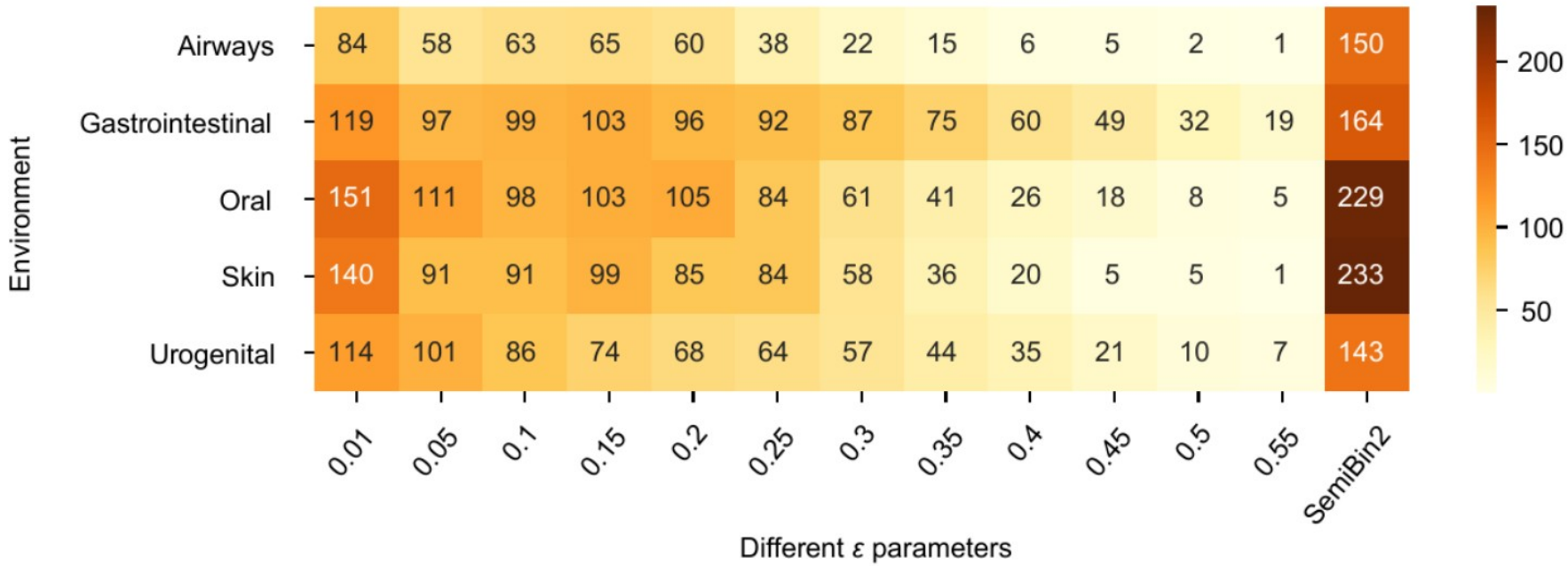
This is on simulated data

Binning long-reads: Ensemble-DBSCAN

1. DBSCAN can return very small clusters as well as large ones, depending on the value of its main parameter, ε.
2. It is not trivial to choose the right ε
3. We use an ensemble of DBSCANs with different ε values

Binning long-reads: Ensemble-DBSCAN

1. DBSCAN can return very small clusters as well as large ones, depending on the value of its main parameter, ε.
2. It is not trivial to choose the right ε
3. We use an ensemble of DBSCANs with different ε values



Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER** (and the majority are not used)

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

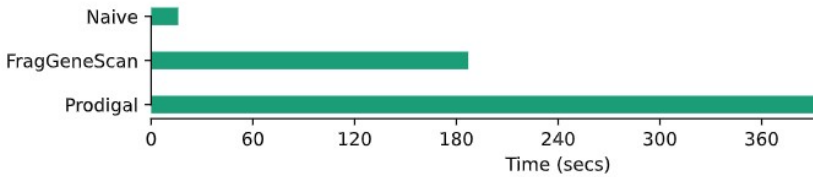
- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER** (and the majority are not used)

We implemented *the simplest possible* ORF finder (START ... STOP)

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER** (and the majority are not used)

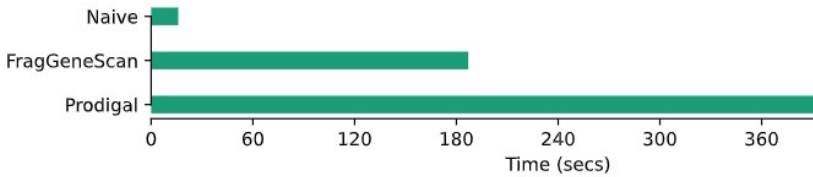


We implemented *the simplest possible* ORF finder (START ... STOP)

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER** (and the majority are not used)



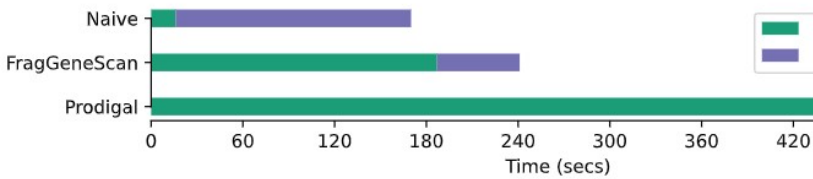
We implemented *the simplest possible* ORF finder (START ... STOP)

- Prodigal returns 129,657 ORFs (total 41.3 Mbps)
- FragGeneScan returns 145,909 ORFs (total 41.7 Mbps)
- Naive returns 1,214,842 ORFs (total 108.1 Mbps)

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER** (and the majority are not used)



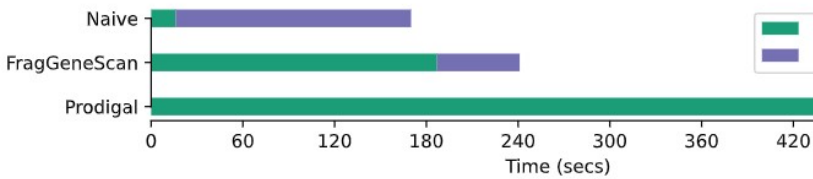
We implemented *the simplest possible* ORF finder (START ... STOP)

- Prodigal returns 129,657 ORFs (total 41.3 Mbps)
- FragGeneScan returns 145,909 ORFs (total 41.7 Mbps)
- Naive returns 1,214,842 ORFs (total 108.1 Mbps)

Bad ORF finding is better than good ORF finding

Vapnik's dictum: do not solve, as an intermediate step, a more general problem than the one you are trying to solve.

- We use gene annotation to find a few specific single-copy genes
- We were using Prodigal, but it is slow
- **All predicted ORFs get fed to HMMER**
  (and the majority are not used)



We implemented *the simplest possible* ORF finder (START ... STOP)

- Prodigal returns 129,657 ORFs (total 41.3 Mbps)
- FragGeneScan returns 145,909 ORFs (total 41.7 Mbps)
- Naive returns 1,214,842 ORFs (total 108.1 Mbps)

Almost no difference in results: >99% identical

(and not clear that the difference is in favor of the more complex ORF finders)