Data Pipelines and Prisms
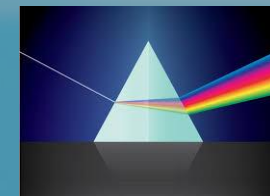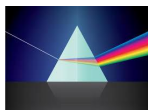## Talk Outline

Case study: A misbehaving pipeline

Data Representation is A Thing !

Semantic and Non-Semantic Data Representation are different things

A Data Prisms project: Express data representations at a high level of abstraction; compute them robustly; deliver hypothesis-neutral synoptic views of data structures
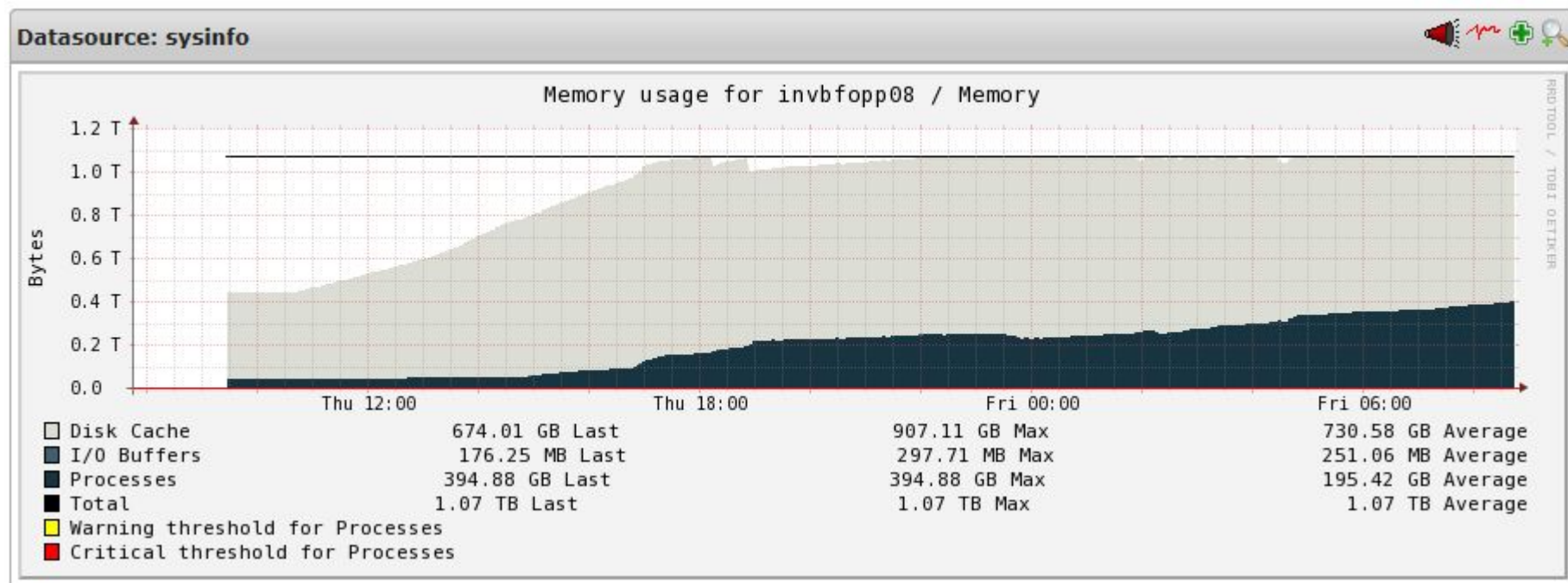
Engineering notes (call-backs, *make*, meta-scheduler, semantic file-naming)

Case study conclusion

Why is this very small bioinformatic computation (fitting together just 420 small (KB size) bits of fungal DNA sequence) misbehaving so badly?

Datasource: sysinfo

Memory usage for invbfopp08 / Memory

	Last	Max	Average
☐ Disk Cache	674.01 GB Last	907.11 GB Max	730.58 GB Average
■ I/O Buffers	176.25 MB Last	297.71 MB Max	251.06 MB Average
■ Processes	394.88 GB Last	394.88 GB Max	195.42 GB Average
■ Total	1.07 TB Last	1.07 TB Max	1.07 TB Average
☐ Warning threshold for Processes			
■ Critical threshold for Processes			

- Continued on to try to use over 800 GB RAM and try to crash a 1TB server
- …whereas it really should run perfectly OK on my Galaxy J8 !

DNA assembly pipeline:
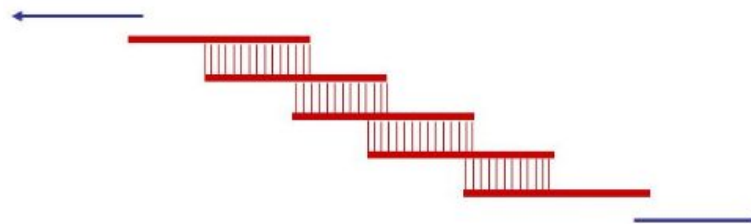
reads

Data Representation
(CGATAGT...)
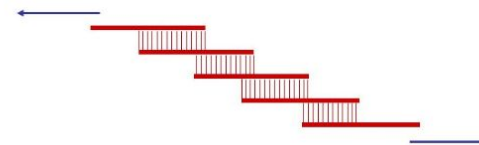
Sequence Quality Filter

Contig-ing

Scaffolding

Adapter etc. removal

Finishing (synteny, stats
summary etc)

reads

Data Representation is A Thing

$\bar{x}$

"To find structure in high-entropy datasets we need to throw away information via entropy-reducing *data representations* that simplify the data while preserving structural features of interest"

McCulloch,A., Jauregui,R., Maclean, P., Ashby, R., Moraga, R., Laugraud A.,Brauning R., Dodds.,K., McEwan, J. (2018) An entropy-reducing data representation approach for bioinformatic data, Database, 2018, 1–16

"Although a picture may be worth a thousand words, a good *representation of data* is priceless. . . . reduce the statistical complexity of the problem—the amount of data needed to solve a given statistical task with a given level of confidence—by approximating the data set by simpler structures"

"Large-Scale Data Representations" (Chapter 5), in National Research Council (2013) Frontiers in Massive Data Analysis. The National Academies Press Washington, DC.
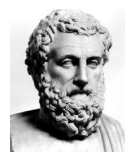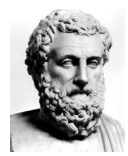
Semantic Data Representation

(very common in biology and bioinformatics)

Example: Reduce entropy via representing DNA sequences by their top hit in a database

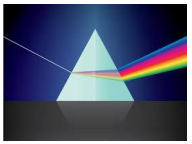| Data	Self-information (bits)	Semantic Representation (top hit in a BLAST database search)	Self Information (bits)
TGCAGCCCACCAGGCTCCTCTG TCCATGGGATTCTCCAGGCAAG AA	92	XM_012177191.3 PREDICTED: Ovis aries family with sequence similarity 180 member A (FAM180A), mRNA	26

(There are 4,951,760,157,141,521,099,596,496,896 possible DNA sequences of that length ($\log_2$ => 92 bits), but only 56 million accessions in the database searched ($\log_2$ => 26 bits)).

Non-Semantic Data Representation

Examples : reduce entropy by representing the data by the group mean, or standard deviation, or by replacing with ranks, binning frequencies or self-information (etc. etc. etc.)

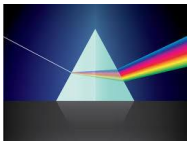| data	mean	stdev	rank (ordering model *i*)	bin frequency (binning model *i*)	self information (probability model *i*)
3.2	3.9	2.1	3	2	0.3
1.8	3.9	2.1	1	1	2.3
6.7	3.9	2.1	5	2	0.3
2.3	3.9	2.1	2	2	0.3
5.5	3.9	2.1	4	2	0.3

A Data Representation Challenge : The Challenge of Primitives

"From the computer systems perspective, it would be very helpful to identify a set of primitive algorithmic tools that (1) provide a framework to express concisely a broad scope of computations; (2) allow programming at the appropriate level of abstraction; and (3) are applicable over a wide range of platforms, hiding architecture-specific details from the users"

"Large-Scale Data Representations" (Chapter 5), in National Research Council (2013) Frontiers in Massive Data Analysis. The National Academies Press Washington, DC.

(see also - similar "Challenge of Primitives" in *visual* data representation (i.e. graphics). Hence developments such as ggplots and ggplots2 packages in R, based on "The Grammar of Graphics"; the wolfram language graphics primitives; etc)

A Grammar of Data Representation?

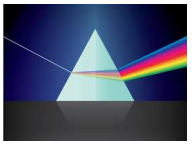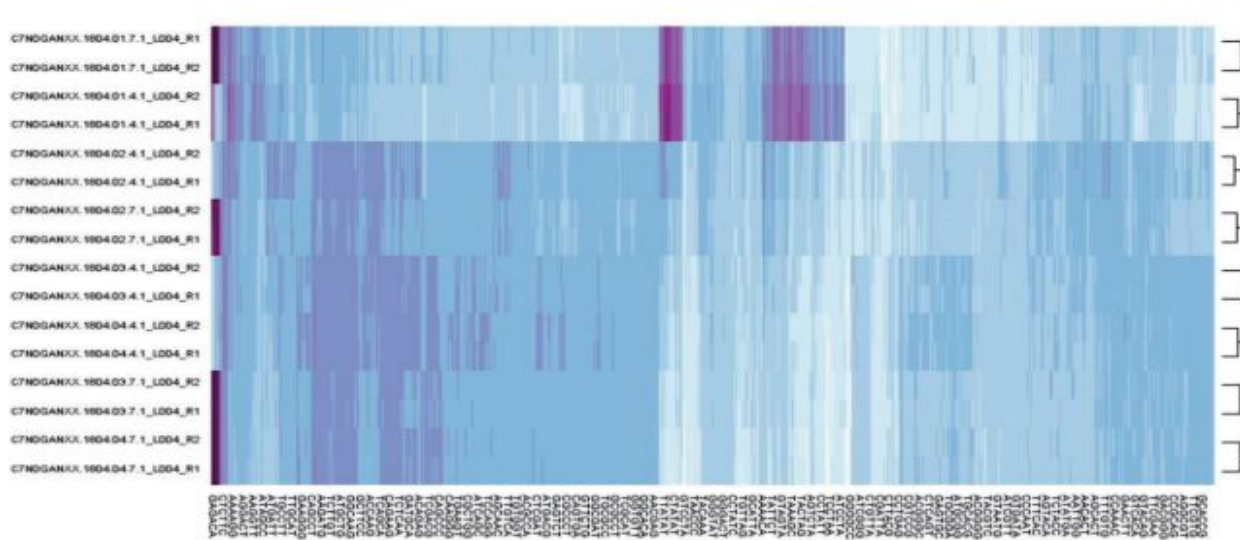$$\mathcal{M}(x^i) \to \bar{x} \qquad \mathcal{S}(x^i) \to s \qquad \mathcal{R}_j(x^i) \to r_j^i \qquad \mathcal{B}_j(x^i) \to c_j^i \qquad \mathcal{I}_j(x^i) \to h_j^i$$

| data	mean	stdev	rank (ordering model *i*)	bin frequency (binning model *i*)	self information (probability model *i*)
3.2	3.9	2.1	3	2	0.3
1.8	3.9	2.1	1	1	2.3
6.7	3.9	2.1	5	2	0.3
2.3	3.9	2.1	2	2	0.3
5.5	3.9	2.1	4	2	0.3

Non-semantic data representation of text (e.g. sequence data)

$$[[x^1, x^2, \ldots x^{4096}] \otimes [\mathcal{I}]](m_j) \to (h_j^1, h_j^2, \ldots, h_j^{4096}) =$$

s_j = a vector co-spectrum for each sequence file

Co-spectra of Fungal sequencing data
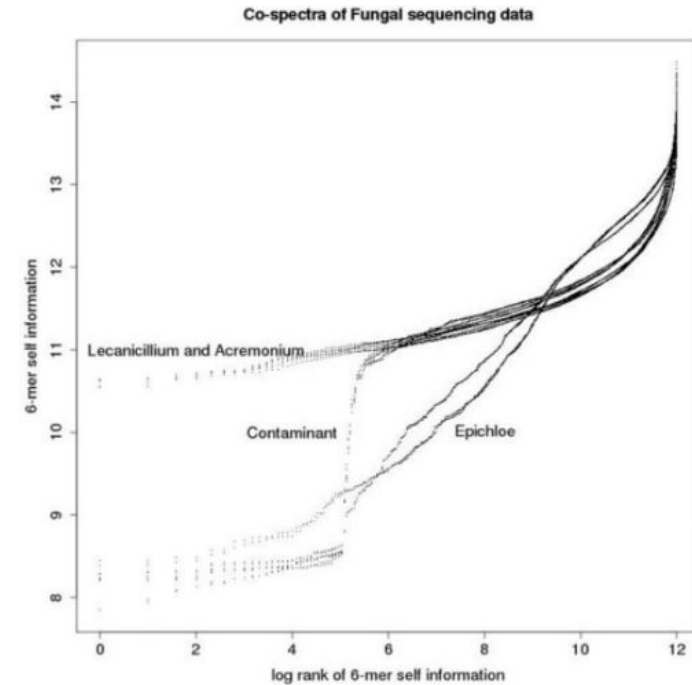
$$[[x^1, x^2, \ldots x^{4096}] \otimes [\mathcal{R}, \mathcal{I}]](m_j) \to$$
$$((r_j^1, h_j^1), (r_j^2, h_j^2), \ldots, (r_j^{4096}, h_j^{4096})) = s_j = \text{a matrix}$$
co-spectrum for each sequence file
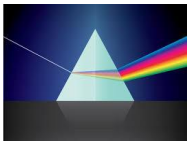
Utilitarian primitives ? A Prisms Project

★ Express data representations at a moderately higher level of abstraction

★ Compute them robustly and at scale, without sacrificing provenance

★ Deliver hypothesis-neutral synoptic views of data

★ Caveat: domain-specific, and (although used in "production"), proof-of-concept

A prisms project

Express data representations at a high level of abstraction; compute them robustly and at scale; deliver hypothesis-neutral synoptic views of data

kmer_prism.sh	`[-h] [-n] [-d] [-s SAMPLE_RATE] [-p kmeroptions ] [ -a fasta|fastq] -O outdir [-C local|slurm ] input_file_names`
sample_prism.sh	`[-h] [-n] [-d] [-s SAMPLE_RATE] [-M minimum sample size] [-t minium_tag_count] [ -T maximum_tag_count]` `                    -a sampler -O outdir [-C local|slurm ] input_file_names`
align_prism.sh	`[-h] [-n] [-d] [-f] [-j num_threads] [-s SAMPLE_RATE] -a aligner -r [ref name	file of ref names]`   ` -p [parameters or file of parameters] -O outdir [-C local|slurm] input_file_names`
sequencing_qc_prism.sh	`[-h] [-n] [-d] [-f] [-C hpctype] [-a analysis] [-s sample rate] -O outdir input_file_names`
demultiplex_prism.sh	`[-h] [-n] [-d] [-x gbsx|tassel3_qc|tassel3] [-l sample_info ] [-e enzymeinfo] -O outdir input_file_names`
genotype_prism.sh	`[-h] [-n] [-d] [-x KGD_tassel] [-p genotyping parameters] -O outdir folder`
gtseq_prism.sh	`[-h] [-n] [-d] [-s species] [-l locus_info ]  -O outdir input_file_names`
melseq_prism.sh	`[-h] [-n] [-d] -a analysis -b blast_database [-w wordsize (16)] [-T blastn|megablast (blastn)] -s similarity (.02)] [-m min_length (40)] [-q min_qual (20)]  [-C local|slurm (slurm)] -O outdir input_file_names`
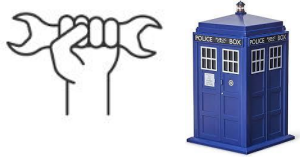
Engineering notes

Call-backs, orchestrated by *make*, with call-back code utilising a meta-scheduler

★ generate (pseudo) semantic targets, and for each target a call-back script

```
rumen_sample1.fa.blastn.nt.taskblastnnum_threads4evalue.02.align_prism
    (make will call                    [ditto]              .align_prism.sh)
rumen_sample2.fa.blastn.nt.taskblastnnum_threads4evalue.02.align_prism
    (make will call                    [ditto]              .align_prism.sh)
```

★ call-back code utilises meta-scheduler for higher level of abstraction

```
tardis --hpctype slurm -d  tag_blast blastn -db  nt -query
_condition_fasta_input_/dataset/GBS_Rumen_Metagenomes/ztmp/melseq_paper_review/tag_blast/rumen_sam
ple1.fa -task blastn -num_threads 4 -evalue .02 \>
_condition_text_output_rumen_sample1.fa.blastn.nt.taskblastnnum_threads4evalue.02.results
```

Engineering notes

★ Call-backs are orchestrated by *make*

```
# align_prism main makefile
#*****************************************************************************
# references:
#*****************************************************************************
# make:
#    http://www.gnu.org/software/make/manual/make.html
#
%.align_prism:
        $@*.sh
        date > $@
```

★ Targets are built concurrently (`make -j N target1 target2 . . . `). The call-back code for each target calls the meta-scheduler which further parallelises the processing by splitting input files and launching chunks on the cluster.

★ Provenance is important - i.e. the ability to drill-down to see what's actually going on, and if necessary tweak and rerun parts of the job. So the target call-backs are just shell-scripts and can be run stand-alone
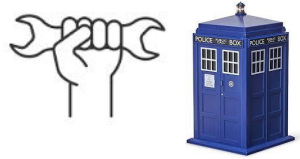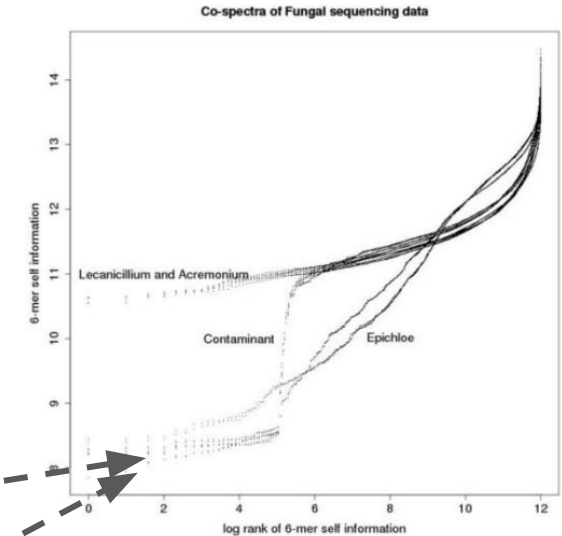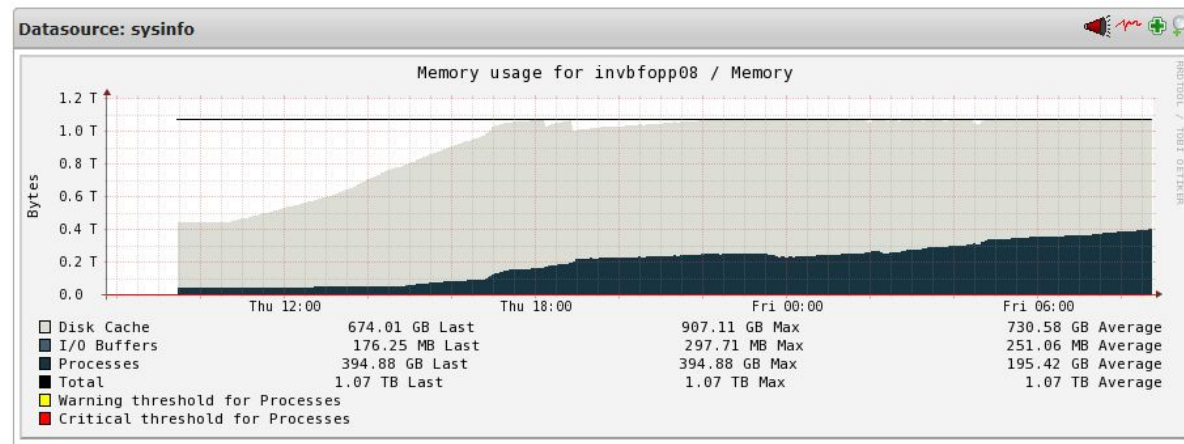
Meta-scheduler notes

- Abstract the details of the underlying grid/computational resource, as well as administrivia such as uncompression/compression, file-splitting etc.
- But ⬆ abstraction often means ⬇ flexibilty and provenance. Assumption: for many users and applications, the unix command-line is about the right compromise between level-of-abstraction, and flexibility.
- Example: original commands (searching a big file for some patterns, plus administrivia)

```
gunzip big_file.gz
grep -f big_pattern-file big_file > big_match_file
gzip big_file
gzip big_match_file
```

- only a single marked-up command is needed to do the above *and* distribute the job over the cluster if using the meta-scheduler

```
tardis grep -f big_pattern-file _condition_text_input_big_file.gz > _condition_text_output_big_match_file
```

Datasource: sysinfo

Memory usage for invbfopp08 / Memory

	Last	Max	Average
Disk Cache	674.01 GB Last	907.11 GB Max	730.58 GB Average
I/O Buffers	176.25 MB Last	297.71 MB Max	251.06 MB Average
Processes	394.88 GB Last	394.88 GB Max	195.42 GB Average
Total	1.07 TB Last	1.07 TB Max	1.07 TB Average
Warning threshold for Processes			
Critical threshold for Processes			

Co-spectra of Fungal sequencing data

A tensorial data representation helped solve the mystery by highlighting a low-complexity feature in the data (the problem was due to an "exploding graph structure" – a hugely over-represented short contaminant sequence caused the graph-based assembly algorithm to create cross-links between almost all possible pairs of sequences)

Graph of contig links →
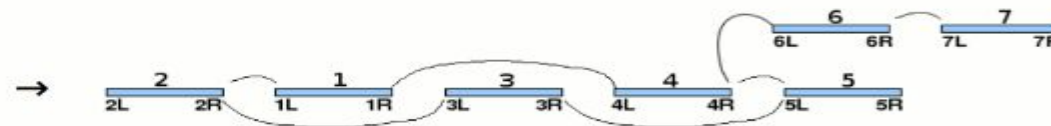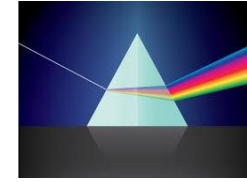
Thank you for your time and attention