Narrative for “Data Pipelines and Prisms” at eResearch2020

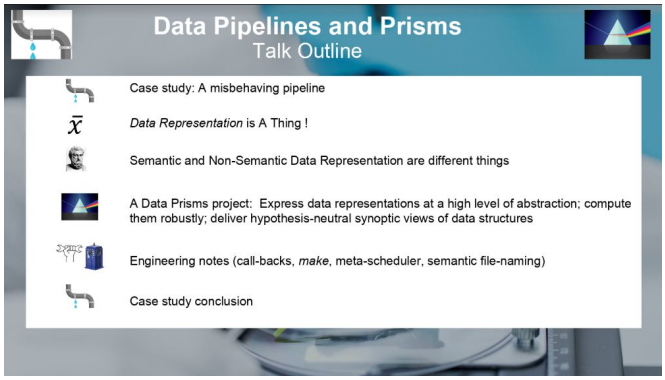
Narrative for “Data Pipelines and Prisms” at eResearch2020 **1**
- Slide 1 - Talk Outline 2
- Slide 2 - The Case Study 3
- Slide 3 - Very Brief Description of Case-Study Pipeline 3
- Slide 4 - Data Representation is A Thing 3
- Slide 5 - Semantic Data Representation 4
- Slide 6 - Non-Semantic Data Representation 5
- Slide 7 - A Data Representation Challenge 5
- Slide 8 - A Grammar of Data Representation ? 6
- Slide 9 - Non-Semantic Data Representation Of Sequence Data 7
- Slide 10 - Utilitarian Primitives ? A Prisms Project 8
- Slide 11 - A Prisms Project 8
- Slide 12 - Engineering Notes: call-backs, make, meta-scheduler 9
- Slide 13 - Engineering Notes: Concurrency and Provenance 10
- Slide 14 - Engineering Notes: The Meta-scheduler 11
- Slide 15 - The Case Study Punchline 12
- Slide 16 - Thanks and Acknowledgements 12

Here are links to the presentation:

Powerpoint:
https://github.com/AgResearch/DataPipelinesAndPrisms/blob/master/DataPipelinesAndPrisms.pptx

PDF:
https://github.com/AgResearch/DataPipelinesAndPrisms/blob/master/DataPipelinesAndPrisms.pdf

Slide 1 - Talk Outline



The “Pipeline” metaphor is often mis-used: in *real* pipelines - be they water, sewerage or oil - it is essential that *all* of what is presented to the input is conducted, without modification, to the output. In data analysis “pipelines”, on the other hand, it is essential that most of the information that is presented at the input is *not* conducted to the output (for example statistical noise, detail that is irrelevant to the kinds of data structures we are looking for), and that what does appear at the output is a highly reduced re-representation of what appeared at the input.

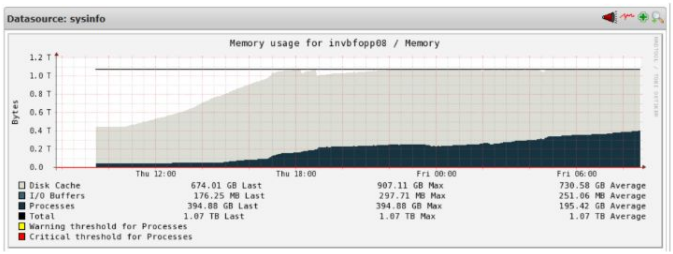
I suggest we code and publish “data prisms” rather than “data pipelines”. (“*Good luck with that*” I hear you say, and you are right ! But let’s fly this kite anyway).

My talk begins with a very brief outline of a case study illustrating application of the philosophy described in the talk; next I try to convince you that “Data Representation” is *a thing*, and that it’s worth thinking about. I distinguish *semantic* from *non-semantic* data representation. (This is perhaps an even more philosophical part of the talk hence represented visually by “The Philosopher”, Aristotle).

Next I describe “A Data Prisms Project”, which aspires to lift the level of abstraction at which we think about and code information-reducing data representations. (This is not yet “*The* Data Prisms Project”: using the definite article would raise expectations too high, suggesting the existence of a “data prisms” community, consensus, respectability, funding, etc. , when there is none of those. I hope one day somebody does *The* Data Prisms project - if and when she does, I will buy her a beer :)). Finally, if time allows (which fortunately for the audience it probably won’t), I’ll run through some notes on software engineering approaches we have found useful, and conclude by wrapping up the case study, delivering the punchline about how a novel non-semantic DNA sequence data representation resolved the problem, and thanking the audience for their time and attention, and my colleagues for their support

Slide 2 - The Case Study

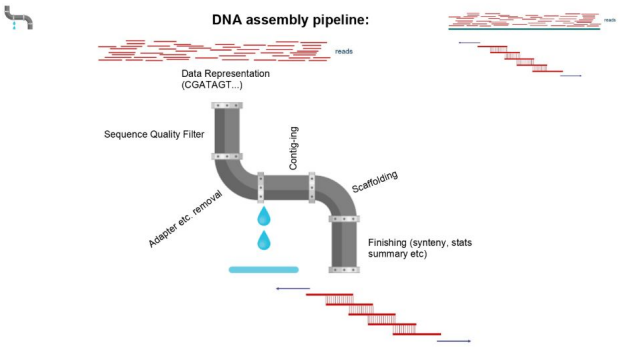
Why is this very small bioinformatic computation (fitting together just 420 small (KB size) bits of fungal DNA sequence) misbehaving so badly?



- 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 !

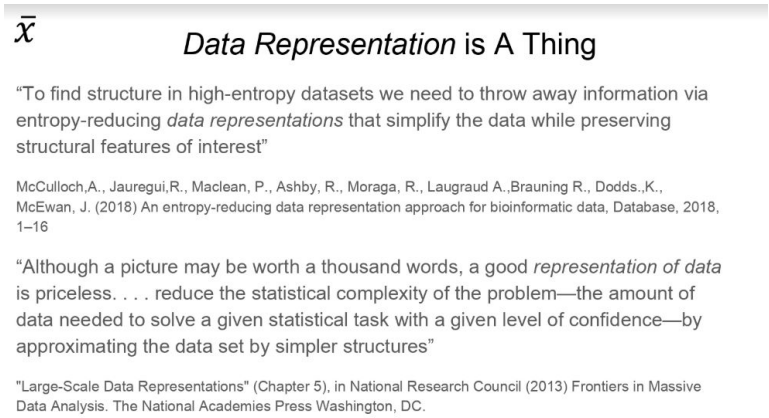
An analyst runs a standard assembly pipeline on a really very small amount of data, and is surprised to find that the job uses a ridiculously large amount of RAM - indeed it eventually renders our (at the time) biggest compute node (1 TB of RAM, 64 cores) unresponsive, requiring a reboot. The pipeline is mature, includes data quality checking and filtering, and does not log any errors or warnings. What is going on ? Do we *really* need a bigger computer for this analysis ?

Slide 3 - Very Brief Description of Case-Study Pipeline



A brief explanation of DNA sequence assembly and what the pipeline is doing. It is a mature pipeline, including thorough data checking and cleaning, and no errors or warnings are logged.

Slide 4 - Data Representation is *A Thing*



Introducing the idea of *Data Representation*, (DR) citing two publications on this topic. The National Research Council document devotes a whole chapter to DR - (their publication "Frontiers in Massive Data Analysis" is freely downloadable and well worth a read, it is full of interesting content). Both citations emphasize the role of DR in reducing the information content (entropy) of a dataset.

Slide 5 - Semantic Data Representation

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)).

As we are working in the field of bioinformatics we distinguish *Semantic* and *Non-Semantic* DR - semantic DR is common in bioinformatics. (This distinction is not made by the NRC DR chapter). A *Semantic* data representation is one where the data is represented by one or more items of text, rather than one or more numbers. Consider for example this typical example from bioinformatics - we searched a sequence database (nt nucleotide in this case), for sequence-near-matches (using BLAST), and then represented the sequence, by the accession of the top hit. It is easy to see that this is an entropy-reducing DR - the information content of the sequence was 92 bits, and of the representation only 26 bits; if we have (say) two files of sequence data, and we represent the sequences in each file by their top-hit accessions, it's obvious that we have reduced the entropy of the data, and also that this will enable us to find

structure in the dataset - for example, some accessions may occur more frequently in one file than the other. (Here we are emphasising the immediate *entropy-reduction* that is achieved by representing sequences by their top hit from a database, rather than any biological interpretation this might support, which is the more usual way in which we justify obtaining the top hit).

Slide 6 - Non-Semantic Data Representation

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

Non-Semantic Data Representations represent the data by one or more numbers. (Usually the original data is itself numeric, but non-semantic representation of text data is not uncommon - an example of non-semantic representation of sequence data is given below). For example, calculating the average of a collection of numbers is a very common kind of non-semantic DR - in representing the original values by the group mean, we reduce the information content of the data, thereby potentially disclosing structure. Other examples of non-semantic DR which reduce the information content of the data in order to support finding structure are: representing the data by the group standard deviation; ordering the data and then representing each item by its rank; binning the data, and representing each item by the number of items falling in its parent bin; representing each item by it's self-information. Some of these representations are model-dependent: ranks depend on an ordering model, bins depend on a binning model, and self-information depends on a probability model; whereas taking the mean or standard deviation are not model-dependent. (Where we calculate a model-dependent data representation, we could consider deploying more than one model, thus obtaining vectorial data representations - this idea is demonstrated below)

Slide 7 - A Data Representation Challenge

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)

Everybody in the audience will be familiar with one specific and important kind of data representation, *visual* data representation, *aka* data visualisation, and of the logistical challenges often involved in producing high quality visualisations (e.g. the many different and often arcane graphics packages, API, unique obscure recipes, and high cost in time and frustration). In their chapter on DR in "Frontiers…", the NRC identified a similar challenge in relation to data-representation in general, which they called the "Challenge of Primitives", and outlined the kind of future work that might be needed to meet this challenge - for example, development of the ability to program at a high level of abstraction.

In the field of Visual Data Representation, the "Challenge of Primitives" has been tackled by ideas such as "The Grammar of Graphics", and implementations such as R's *ggplots* and *ggplots2* packages, which provide an API at a high level of abstraction.

So - what might a "Grammar of Data Representation (in general)" look like ?

Slide 8 - A Grammar of Data Representation ?

A Grammar of Data Representation?

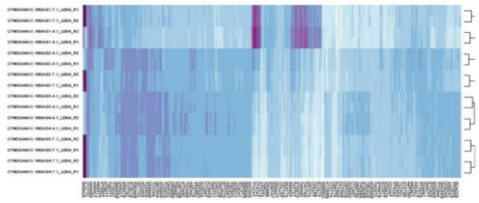
$\mathcal{M}(x^i) \to \bar{x}$ $\quad \mathcal{S}(x^i) \to s$ $\quad \mathcal{R}_j(x^i) \to r_j^i$ $\quad \mathcal{B}_j(x^i) \to c_j^i$ $\quad \mathcal{I}_j(x^i) \to h_j^i$

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

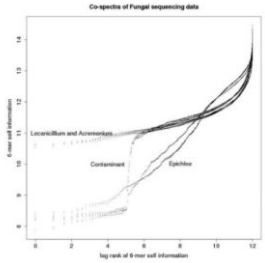
Well - we could for example consider a consistent notation for data representations, as functional operators yielding output representations from input data values, indexed by model in the case of model-dependent representations such as self-information etc.

Slide 9 - Non-Semantic Data Representation Of Sequence Data

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



$[[x^1, x^2, \ldots x^{4096}] \otimes [\mathcal{I}]](m_j) \rightarrow (h_j^1, h_j^2, \ldots, h_j^{4096}) =$
s_j = a vector co-spectrum for each sequence file



$[[x^1, x^2, \ldots x^{4096}] \otimes [\mathcal{R}, \mathcal{I}]](m_j) \rightarrow$
$((r_j^1, h_j^1), (r_j^2, h_j^2), \ldots, (r_j^{4096}, h_j^{4096})) =$ s_j = a matrix co-spectrum for each sequence file

Given a palette of such operators, we can combine them in various ways to obtain new data representations , for example by taking abstract formal products of tuples of models with tuples of operators. In the example on the left, we have considered each of 16 files of sequence data associated with our case-study, as providing a probability model of how often DNA 6mers are likely to appear, and have designed a non-semantic DR of this data by taking the cross product of the self-information operator with tuples of 6-mers to yield a numeric co-spectrum vector for each file - these vectors are the rows in the heatmap. The co-spectrum vectors have lower entropy than the original data, and clearly elicit structure in the collection of files that we could not see before.

A benefit of expressing this data representation at a high level of abstraction (as a formal algebraic product), is that we can now easily tweak the formula in various ways to motivate other representations that might not have been considered. For example - what if we substitute a ranking operator, for the self-information operator, in the formula on the left ? Or, what if we substitute in a tuple of *(ranking operator, self-information operator)*, obtaining the abstract data representation formula on the right ? This representation yields a co-spectrum matrix (rather than vector) for each file. These matrix co-spectra can be visualised as X-Y plots as on the right, eliciting interesting new structures such as different files having different slopes in this representation, and the presence of a turning point for some of the files.

Slide 10 - Utilitarian Primitives ? A Prisms Project

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

Unfortunately we don’t currently have a compiler or interpreter capable of translating abstract formal schema like these into machine code ! Hence for now a much less ambitious approach to the challenge of primitives : A Prisms Project - a collection of (domain-specific) utilities aiming to overcome some of the logistical challenges in generating data representations, in the domain of bioinformatics.

Slide 11 - A Prisms Project

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

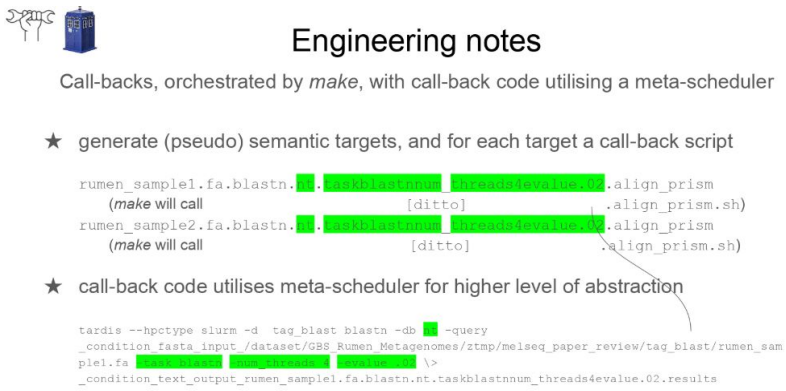
This prisms project consists of a collection of prisms and shared libraries, with a consistent API. Each prism handles a specific broad class of data representations. For example *kmer_prism.sh* handles non-semantic representation of sequence data by using kmer self-information as described above; *align_prism.sh* handles the common case of semantic representation of sequence data by top hits in a database; *sample_prism.sh* handles representation of sequence and other text data, and also numeric datasets, by a random-sample of the data - another way to lower the information content of a dataset.

Typical logistical challenges in computing data representations, that this prisms project tackles, include: handling large numbers of input and output files at a high level of abstraction; output file naming and disposition; summarising large numbers of output files to obtain the final representations; efficient submission of large numbers of tasks to a compute cluster, at a high level of abstraction; restarts of interrupted runs and automated discovery of which targets can be launched concurrently and which consecutively.

Ideally it should be easy for a user to express what is needed, at a high level of abstraction, for example “Please randomly sample all those text/sequence files *over here* , put the results *over there* (“you” figure out safe output-file-naming, how to submit jobs to the cluster, whether files need uncompressing, splitting files to distribute over cluster, etc.).

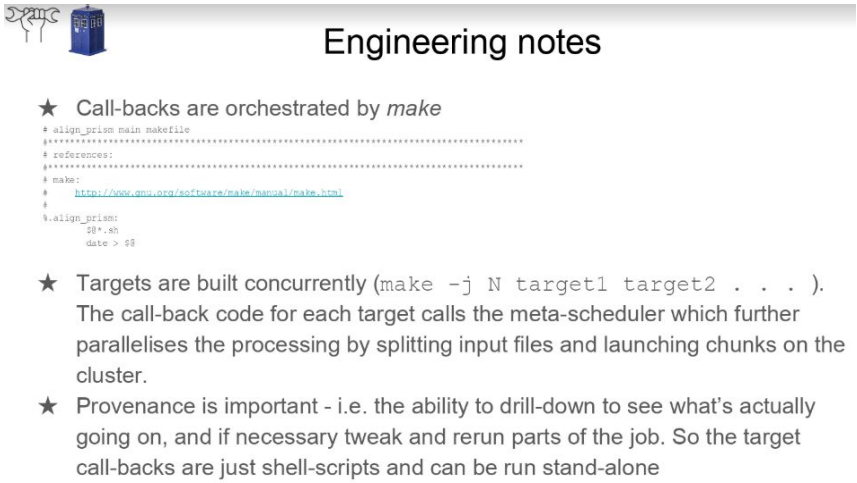
Broad classes of similar data representation are handled at a higher level of abstraction, for example *align_prism.sh* exposes the same API and provides the same logistical support for representations using a number of underlying database search utilities such as blast (blastn, blastx, megablast etc.), bwa, bowtie. Coding of specialist data representations such as genotyping by sequencing (in which sequence data are represented by numeric tuples coding probable variant alleles) benefit from the improved logistical support developed for more general data representations, as these specialist prisms face many of the same logistical challenges as the generalists.

Slide 12 - Engineering Notes: call-backs, make, meta-scheduler



A prism generates a series of semantic pseudo-targets, and associated call-backs, one call-back per target. A *pseudo-target* is a land-mark file that will be written when the build of a target is completed, simply containing a date-stamp, not actual outputs. The *semantic* target's name is a human-readable hash of the input file name, program used, and program options. The make-file rules are simple - to build a target, *make* simply invokes the call-back whose name matches the target. The call-back code is expressed at a high level of abstraction by delegating the details of job splitting and scheduling to the meta-scheduler.

Slide 13 - Engineering Notes: Concurrency and Provenance



make automatically determines targets that are independent and can be built concurrently; and each build in turn calls the meta-scheduler which encapsulates file-splitting and job scheduling. These two layers of parallelisation cooperate to achieve a high degree of parallelisation of computation of data representations, with minimal effort from the user.

The call-backs corresponding to each target are shell-scripts that can be run stand-alone if necessary. This allows users to easily drill down and see exactly what were the low level calls (e.g. to applications such as blast, bwa etc.) , and gives fine-grained control (e.g. call-backs can be tweaked and re-run manually if necessary). This approach provides a high level of provenance.

The technique of automated output file-naming, via construction of a human-readable hash of input filename, low-level application used (blast etc.), and program options, is extremely useful as a pragmatic booster of provenance and reproducibility - it is immediately apparent from the name of an output file, what input it originated from and what was done to it.

Slide 14 - Engineering Notes: The Meta-scheduler

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

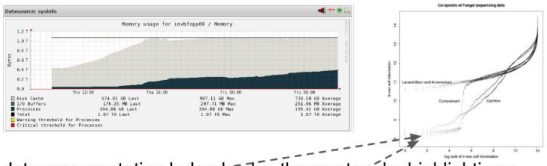
```
cardiff grep -f big_pattern-file _condition_text_input_big_file.gz > _condition_text_output_big_match_file
```

The meta-scheduler provides an interface to HPC resources at a high level of abstraction, thus simplifying call-back code (and interactive use of HPC by users). For example: which HPC pool to use (slurm ? condor ? launch processes on local host ?), how to split the input files ?, how to join back together output chunks ?, preparation of job submission files; all these are encapsulated by the meta-scheduler.

The meta-scheduler used in this prisms project uses a command-line markup approach to deliver access to HPC resources at a high level of abstraction, playing a pre-compiler role: marked-up commands are pre-compiled, outputting a series of actual commands to be launched on the cluster (for , in turn, further processing by language interpreters and operating system shells). Mark-up hints indicate which are the input and output command-line arguments, thus specifying input file-splitting and output file joining details for the pre-compiler.

To the user (or prisms call-back code) , a marked-up command processed by the meta-scheduler usually behaves just like an ordinary command processed by the shell - the user is not aware of file-splitting, job scheduling, etc., and the command appears to have been processed on the local machine. If the job completed without error, the meta-scheduler cleans up its working folder (for example, deleting the split-data files etc.), and the command simply returns (and the exit code is 0); if there were any errors, the meta-scheduler command exits with a non-zero exit code, displays error messages, and the path to the working folder (and no clean-up is done so that the problem can be diagnosed, and the job relaunched if possible)

Slide 15 - The Case Study Punchline



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)

This non-semantic data representation of sequence data (introduced above), related to the case study, highlighted a low-complexity feature in the data - a small set of around 23 6-mers that are all highly over-represented. These can be assembled into a longer over-represented sequence, which turned out to be an adapter present in the data but not known to the pipeline. This feature interacted perversely with the graph-based assembler used, as it causes the number of edges in the graph to explode.

This illustrates how a “data prisms perspective” - information-reducing synoptic data representations - can highlight features in the data which may not be visible from a pipeline perspective.

Slide 16 - Thanks and Acknowledgements

Thank you for your time and attention

Thanks to the awesome AgResearch GBS team, and Invermay lab, bioinformatics and stats geniuses, and many other AgResearch colleagues, on whose coat-tails I’ve ridden. (Any embarrassing errors, misconceptions, time-wasting and Ig-Nobel-worthiness are most definitely mine alone) !