Approaches to standardizing parallel evaluation in *Bioconductor*

Martin Morgan (mtmorgan@fredhutch.org)
Fred Hutchinson Cancer Research Center

26 January 2015
Abstract: The *Bioconductor* project represents almost 1000 core and contributed packages for the analysis and comprehension of high-throughput genomic data. The project core provides software infrastructure tailored to our common use cases, including facilities for parallel evaluation via the *BiocParallel* and other packages. *BiocParallel* has had mixed success, simplifying cross-platform compatibility but imperfectly exploiting heterogeneous computational environments and inspiring creative parallel computation.
Outline: \textit{R / Bioconductor} for Integrative Analysis

1. The \textit{Bioconductor} project
2. Use cases
3. \textit{BiocParallel} & Friends
4. Critique
5. Prospects
I’m sorry to have left so suddenly. I was taken ill during your talk and had to go home. I am still ill in fact one week later. – Martyn Plummer, President of the R Foundation, 15 Jan 2015.
**Bioconductor**

**Goal**  Analysis and comprehension of high-throughput genomic data

**Focus**
- Sequencing; RNA-Seq, ChIP-Seq, Variants, . . .
- Expression and other microarrays; flow cytometry; proteomics, imaging

**Themes**
- ‘Core’ and (primarily academic) community contributions.
- R – statistics, visualization, interoperability
- Reproducible – data structures, scripts, vignettes, packages
- Interoperable – formal classes in ‘core’ packages
- Accessible: affordable, transparent, usable

Huber et al., Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*: soon!
Project status (December, 2014)

- 320,000 unique IP address package downloads / year
- 1,300 support site contributors / year, 8,200 visitors / month
- 10,500 PubMed Central mentions of ‘Bioconductor’; ≈ 22,000 citations to Bioconductor packages
- Funding from US NIH & NSF, and EC
Use Cases: High-Throughput Sequencing

Questions

- Which genes are differentially expressed in cancer versus normal tissue?
- Which transcription factors are regulating gene expression?
- What single nucleotide polymorphisms (SNPs) are present in a population / associated with a disease?
- What is the ChIP-seq regulatory signal along a linear genome?

Sample sizes

- Designed experiments – e.g., 10’s or 100’s of samples
- Cohorts – e.g., 100’s or 1000’s of patients
- Populations – 1000’s - 10000’s of individuals

Attributes

- 10,000’s of genes
- Millions of variants
Use Cases

Patterns

1. Reduction – large idiosyncratic (‘BAM’) files reduced to e.g., count matrix of $100,000 \times 100$.

2. Intermediate expansion (e.g., pairwise interactions between SNPs...) & reduction (...reaching statistical significance) – MatrixEQTL.

3. Query-like, e.g., predict SNP effects; drill down on subsets

4. 1-dimensional linear dependency

‘Academic’ work environment

- Local or shared computer with 10’s of cores and moderate memory.
- Cluster with possibly idiosyncratic batch scheduler.
- More than 1/2 of our web site visitors are Windows users!
BiocParallel & Friends: Strategies for Large Data

Memory management
- Restrict input to relevant ‘columns’.
- Select relevant rows.
- Iterate: read in and operate on successive chunks.

Speed
- Efficient R code $– 10-100\times$ speed-up. All gravy.
- C implementation $– 1-5\times$ speed-up. Tedium, error-prone, multiple languages.
- Parallel evaluation $– 2-10\times$ speed-up. Debugging & error recovery; local expertise. Implies memory management.

- BiocParallel, GenomicFiles, Streamer

BiocParallel...

How

```r
register(MulticoreParam(workers=4))  # stack
ans <- bplapply(X, FUN, ..., BPPARAM=bpparam())
```

Why

- Easy(er) cross-platform use – registry of OS-specific back ends.
- Standardized front end (`bplapply`, `bpvec`, ...) to diverse back-ends (*BiocParallelParam*).
  - **Multicore**, `snow`, `Rmpi`, **BatchJobs** (reasonable interface to cluster schedulers)
  - Familiar (?) functional style.
- Spawned jobs: interactive; direct use of existing code.
- Registration stack supports coarse-grained nested parallelism.
... & Friends

**GenomicFiles**

- Manage files underlying many biological applications – references, iteration, restriction, ...
- Distribution of file references (paths) and shared file system as ‘state of the art’

**Streamer**

- Compose work flows connecting iterative data input functions through serial and parallel operations to data output.
- ‘Yield’ on the stream pulls a chunk of data through the stream.

**rhdf5, h5vc**

- Transform idiosyncratic files to intermediate form.
- Basis for spoke-like down-stream exploration
Critique

- User R code is often very inefficient. 100 – 1000-fold gains in R; 1 – 10-fold gain in C.
- Non-multicore parallel programming is very challenging to support – heterogenous user environments; lack of shared state.
- Shared memory implies memory management in concert with parallelization – iteration, restriction, sampling.
- Clunky R-level nested parallelism.
- High-throughput computing may be good enough.
- Interactive debugging and error recovery would be great!
- The illusion of performance would be a great goal for interactive exploratory analysis
- SIMD appealing conceptually but less consistent with interactive user expectations.
Prospects

Clouds & virtualization

- Controlled environment enables advanced configuration, e.g., our *StarCluster* AMI

Grammar of high throughput computing

- Verbs: what are they? `yield`, `restrict` (columns), `select` (rows), `query`; `map`, `reduce`; `tiles`, `aggregate`, `splice`
- Lazy exploratory evaluation

```r
d %>% restrict() %>% select() %>% aggregate() %>% display()
```

```r
aggregate() %>% data.frame
```
Acknowledgments

Core (Seattle): Sonali Arora, Marc Carlson, Nate Hayden, Valerie Obenchain, Hervé Pagès, Paul Shannon, Dan Tenenbaum.


Scientific Advisory Board: Robert Gentleman, Wolfgang Huber, Rafael Irizarry, Vincent Carey, Paul Flicek, Simon Tavaré, Simon Urbanek.

Funding: US NHGRI/NIH U41HG004059; US NCI U24CA180996; US NSF BIGDATA; EC H2020-PHC-2014