6.874 Lecture 4

- How can we choose an appropriate model for biological data?
  - Structure (graph, clusters, n-ary relationships, classification)
  - Complexity (parameters, edges, variables)
- What biological questions can answer with our models?
  - Predicative ability (effect of perturbations, phenotype)
  - Relationship between models (evolution)
- How can we improve our models by designing new experiments?
  - What experiments will be most informative (active learning)
- What aspects of models can be viewed as mechanistic?
  - Physical models (direct molecular interactions)
Next Generation Sequencing

- High-throughput sequencing produces short DNA sequences ("reads") in huge volumes at low cost.
- Costs are down, adoption is up, next-next generation is coming soon.

Applications of NGS

• Researchers have no problem putting this data to work

- 1000 Genomes
- Global Ocean Survey
- Human Microbiome

• Future growth may be multiplicative
  - Spatial resolution: tissues, individuals, geography
  - Temporal resolution: circadian, seasonal, lifetimes

Short Read Applications

- **Genotyping**
  - Goal: identify variations
  - RNA-seq, ChIP-seq, Methyl-seq
  - Goal: classify, measure significant peaks

Finding the alignments is typically the performance bottleneck.
Short Read Alignment

• Given a reference and a set of reads, report at least one “good” local alignment for each read if one exists
  – Approximate answer to: where in genome did read originate?

• What is “good”? For now, we concentrate on:
  – Fewer mismatches is better
  – Failing to align a low-quality base is better than failing to align a high-quality base

Indexing

- Genomes and reads are too large for direct approaches like dynamic programming

- *Indexing* is required

  Suffix tree

  Suffix array

  Seed hash tables
  Many variants, incl. spaced seeds

- Choice of index is key to performance

Indexing

- Genome indices can be big. For human:
  - > 35 GBs
  - > 12 GBs

- Large indices necessitate painful compromises
  1. Require big-memory machine
  2. Use secondary storage
  3. Build new index each run
  4. Subindex and do multiple passes

Burrows-Wheeler Transform

- Reversible permutation used originally in compression

\[
\begin{align*}
&a c a a c g \\
&a c a a c g \\
&a c a a c g \\
&c g $ a c a a \\
&g $ a c a a c \\
\end{align*}
\]

\[
\begin{align*}
&$ a c a a c g \\
&a c g $ a c a \\
&c a a c g $ a \\
&c g $ a c a a \\
&g $ a c a a c \\
\end{align*}
\]

\[
\begin{align*}
&g c $ a a a c \\
&c a a c g $ a \\
&c g $ a c a a \\
&g $ a c a a c \\
\end{align*}
\]

T

- Once BWT(T) is built, all else shown here is discarded
  - Matrix will be shown for illustration only


Burrows-Wheeler Transform

- Property that makes BWT(T) reversible is “LF Mapping”
  - $i^{th}$ occurrence of a character in Last column is same text occurrence as the $i^{th}$ occurrence in First column

Burrows-Wheeler Transform

- To recreate T from BWT(T), repeatedly apply rule:
  \[ T = BWT[LF(i)] + T; \ i = LF(i) \]
  - Where LF(i) maps row i to row whose first character corresponds to i’s last per LF Mapping

- Could be called “unpermute” or “walk-left” algorithm

FM Index

- Ferragina & Manzini propose “FM Index” based on BWT

- Observed:
  - LF Mapping also allows *exact matching* within T
  - \( \text{LF}(i) \) can be made fast with *checkpointing*
  - ...and more (see FOCS paper)

Exact Matching with FM Index

- To match Q in T using BWT(T), repeatedly apply rule:
  \[ \text{top} = \text{LF}(\text{top}, \text{qc}); \ \text{bot} = \text{LF}(\text{bot}, \text{qc}) \]
  - Where \text{qc} is the next character in Q (right-to-left) and \text{LF}(i, \text{qc}) maps row i to the row whose first character corresponds to i’s last character \text{as if it were qc}
Exact Matching with FM Index

- In progressive rounds, **top** & **bot** delimit the range of rows beginning with progressively longer suffixes of Q

Exact Matching with FM Index

- If range becomes empty ($top = bot$) the query suffix (and therefore the query) does not occur in the text

Rows to Reference Positions

• Once we know a row contains a legal alignment, how do we determine its position in the reference?
Rows to Reference Positions

- Naïve solution 1: Use “walk-left” to walk back to the beginning of the text; number of steps = offset of hit
  
  
  ![Diagram](http://www.cbc.png)

  2 steps, so hit offset = 2

- Linear in length of text in general – too slow

http://www.cbc.png
Rows to Reference Positions

• Naïve solution 2: Keep whole suffix array in memory. Finding reference position is a lookup in the array.

• Suffix array is ~12 gigabytes for human – too big
Rows to Reference Positions

- Hybrid solution: Store *sample* of suffix array; “walk left” to next sampled (“marked”) row to the left
  - Due to Ferragina and Manzini

- Bowtie marks every 32\textsuperscript{nd} row by default (configurable)

Put It All Together

- Algorithm concludes: “aac” occurs at offset 2 in “acaacg”
Checkpointing in FM Index

- **LF(i, qc)** must determine the *rank* of *qc* in row i
- Naïve way: count occurrences of *qc* in all previous rows
  - This LF(i, qc) is linear in length of text – too slow

Checkpointing in FM Index

- **Solution:** pre-calculate cumulative counts for A/C/G/T up to periodic **checkpoints** in BWT

- **LF**(i, qc) is now constant-time
  (if space between checkpoints is considered constant)

FM Index is Small

- Entire FM Index on DNA reference consists of:
  - BWT (same size as T)
  - Checkpoints (~15% size of T)
  - SA sample (~50% size of T)

- Total: ~1.65x the size of T

Assuming 2-bit-per-base encoding and no compression, as in Bowtie
Assuming a 16-byte checkpoint every 448 characters, as in Bowtie
Assuming Bowtie defaults for suffix-array sampling rate, etc

FM Index in Bioinformatics

- Oligomer counting

- Whole-genome alignment

- Smith-Waterman alignment to large reference
Short Read Alignment

- FM Index finds exact sequence matches quickly in small memory, but short read alignment demands more:
  - Allowances for mismatches
  - Consideration of quality values

- Lam et al try index-assisted Smith-Waterman
  - Slower than BLAST

- We tried index-assisted “seed-and-extend”
  - Competitive with other aligners, but not much faster

- Bowtie’s solution: backtracking quality-aware search

Backtracking

- Consider an attempt to find $Q = \text{“agc”}$ in $T = \text{“acaacg”}$:

- Instead of giving up, try to “backtrack” to a previous position and try a different base

Backtracking

- Backtracking attempt for $Q = \text{“agc”}$, $T = \text{“acaacg”}$:

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

"gc" does not occur in the text

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

Substitution

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

"g"

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

"a"

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

"a"

```
$ a c a a c g$
a a c g $ a c$
a c a a c g$
ac g $ a c a$
caacg $ a$
c g $ a c a a$
g $ a c a a c$
```

Found this alignment:

```
acaacg
  \[
  \text{agc}
  \]
```

Backtracking

• May not be so lucky

Found this alignment (eventually):

\texttt{acaacg}

\[\texttt{agc}\]

Backtracking

- Relevant alignments may lie along multiple paths
  - E.g., Q = “aaa”, T = “acaacg”
Backtracking

- Bowtie’s \(-v \ <\text{int}\) option allows alignments with up to \(<\text{int}\) mismatches
  - Regardless of quality values
  - Max mismatches allowed: 3
  - Equivalent to SOAP’s* \(-v\) option

* Li R et al: SOAP: short oligonucleotide alignment program. 
Qualities

- When backtracking is necessary, Bowtie will backtrack to *leftmost just-visited position* with minimal quality

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Phred Quals</th>
</tr>
</thead>
<tbody>
<tr>
<td>G C C A T A C G G A T T A G C C</td>
<td>40 40 35 40 40 40 30 30 20 15 15 40 40 40 40</td>
</tr>
<tr>
<td>G C C A T A C G G A C T A G C C</td>
<td>40 40 35 40 40 40 40 30 30 20 15 15 40 40 40</td>
</tr>
<tr>
<td>G C C A T A C G G G C T A G C C</td>
<td>40 40 35 40 40 40 40 30 30 20 15 15 40 40 40</td>
</tr>
</tbody>
</table>

- Greedy, depth-first, not optimal, but simple

Qualities

- Bowtie supports a Maq*-like alignment policy
  - ≤ N mismatches allowed in first L bases on left end
  - Sum of mismatch qualities may not exceed E
  - N, L and E configured with -n, -l, -e
  - E.g.:

  L=12
  E=50, N=2

  If N < 2
  If E < 45
  If L < 9 and N < 2

  Maq-like is Bowtie’s default mode (N=2, L=28, E=70)


Excessive Backtracking

- But how to match left-to-right?
- Double indexing:
  - Reverse read and use “mirror index”: index for reference with sequence reversed
Implementation

- Free, Open Source under Artistic License
- C++
- Uses SeqAn* library (http://www.seqan.de)
- Uses POSIX threads to exploit parallelism
- `bowtie-build` is the indexer
- `bowtie` is the aligner
- `bowtie-convert` converts Bowtie’s alignment output format to Maq’s `.map` format
  - Users may leverage tools in the Maq suite, e.g., `maq assemble`, `maq cns2snp`
  - Uses code from Maq

Bowtie is an ultrafast, memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of 25 million reads per hour on a typical workstation with 2 gigabytes of memory. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: 1.3 GB for the human genome. It supports alignment policies equivalent to Maq and SOAP but is much faster: about 35x faster than Maq and over 350x faster than SOAP when aligning to the human genome.

0.9.7 release - 11/8/08
- Added new reporting option --n <int> which suppresses all alignments for a particular read if more than <int> reportable alignments exist for it.
- Threads now buffer all alignments for a particular read/phase then output all alignments in one critical section. This guarantees that all alignments for a given read/phase appear in one consecutive block of the output, even when multiple threads are operating in parallel.
- Separated the quality-conversion and parsing aspects of the old --solexa-quals argument into separate arguments: --solexa-quals (quality conversion) and --integer-quals (parsing).
- bowtie-convert now handles the new (post-0.7.0) Maq alignment format. The new format allows Maq tools to handle reads up to 127 bases, whereas the old format was limited to 63 bases. Added a -o option to opt for the old Maq format.
- New --refout argument sends alignments to a set of files named refXXXXX.map, where XXXXX is the 0-padded index of the reference sequence aligned to. Useful for dealing with large datasets aligned to, e.g., the assembled human genome.
- Improved tutorial to use a simple simulated read set (included) to do SNP calls with Maq.
- Added --nota option to bowtie-build
- Fixed make_h_sapiens_asm.sh script to include mitochondrial DNA.

TopHat released - 11/8/08
- Cole Trapnell has completed the initial release of TopHat, a fast splice junction mapper for RNA-Seq reads. TopHat aligns RNA-Seq reads to mammalian-sized genomes using Bowtie, and then analyzes the mapping results to identify splice junctions between exons.
Indexing Performance

- Bowtie employs a indexing algorithm* that can trade flexibly between memory usage and running time
- For human (NCBI 36.3) on 2.4 GHz AMD Opteron:

<table>
<thead>
<tr>
<th>Physical memory Target</th>
<th>Actual peak memory footprint</th>
<th>Wall clock time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 GB</td>
<td>14.4 GB</td>
<td>4h:36m</td>
</tr>
<tr>
<td>8 GB</td>
<td>5.84 GB</td>
<td>5h:05m</td>
</tr>
<tr>
<td>4 GB</td>
<td>3.39 GB</td>
<td>7h:40m</td>
</tr>
<tr>
<td>2 GB</td>
<td>1.39 GB</td>
<td>21h:30m</td>
</tr>
</tbody>
</table>

Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- PC: 2.4 GHz Intel Core 2, 2 GB RAM
- Server: 2.4 GHz AMD Opteron, 32 GB RAM
- Bowtie v0.9.6, Maq v0.6.6, SOAP v1.10
- SOAP not run on PC due to memory constraints
- Reads: FASTQ 8.84 M reads from 1000 Genomes (Acc: SRR001115)
- Reference: Human (NCBI 36.3, contigs)

## Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td><strong>33.8 M</strong></td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td><strong>351x</strong></td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td><strong>29.5 M</strong></td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td><strong>59.8x</strong></td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td><strong>28.8 M</strong></td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td><strong>107x</strong></td>
<td>74.7</td>
</tr>
</tbody>
</table>

- Bowtie delivers about 30 million alignments per CPU hour

Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td><strong>67.4</strong></td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td><strong>67.3</strong></td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td><strong>71.9</strong></td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td><strong>74.7</strong></td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td><strong>71.9</strong></td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td><strong>74.7</strong></td>
</tr>
</tbody>
</table>

- Disparity in reads aligned between Bowtie (67.4%) and SOAP (67.3%) is slight

Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td>1,149 MB</td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td>13,619 MB</td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td>804 MB</td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td>1,353 MB</td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td>804 MB</td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- Disparity in reads aligned between Bowtie (71.9%) and Maq (74.7%) is more substantial (2.8%)
  - Mostly because Maq –n 2 reports some, but not all, alignments with 3 mismatches in first 28 bases
  - Fraction (<5%) of disparity is due to Bowtie’s backtracking limit (a heuristic not discussed here)

Comparison to Maq & SOAP

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Bowtie speedup</th>
<th>Reads aligned (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie –v 2 (server)</td>
<td>15m:07s</td>
<td>15m:41s</td>
<td>33.8 M</td>
<td><strong>1,149 MB</strong></td>
<td>-</td>
<td>67.4</td>
</tr>
<tr>
<td>SOAP (server)</td>
<td>91h:57m:35s</td>
<td>91h:47m:46s</td>
<td>0.08 M</td>
<td><strong>13,619 MB</strong></td>
<td>351x</td>
<td>67.3</td>
</tr>
<tr>
<td>Bowtie (PC)</td>
<td>16m:41s</td>
<td>17m:57s</td>
<td>29.5 M</td>
<td><strong>1,353 MB</strong></td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (PC)</td>
<td>17h:46m:35s</td>
<td>17h:53m:07s</td>
<td>0.49 M</td>
<td><strong>804 MB</strong></td>
<td>59.8x</td>
<td>74.7</td>
</tr>
<tr>
<td>Bowtie (server)</td>
<td>17m:58s</td>
<td>18m:26s</td>
<td>28.8 M</td>
<td><strong>1,353 MB</strong></td>
<td>-</td>
<td>71.9</td>
</tr>
<tr>
<td>Maq (server)</td>
<td>32h:56m:53s</td>
<td>32h:58m:39s</td>
<td>0.27 M</td>
<td><strong>804 MB</strong></td>
<td>107x</td>
<td>74.7</td>
</tr>
</tbody>
</table>

- Bowtie and Maq have memory footprints compatible with a typical workstation with 2 GB of RAM
  - Maq builds non-reusable spaced-seed index on reads; recommends segmenting reads into chunks of 2M (which we did)
- SOAP requires a computer with >13 GB of RAM
  - SOAP builds non-reusable spaced-seed index on genome

Multithreaded Scaling

<table>
<thead>
<tr>
<th></th>
<th>CPU time</th>
<th>Wall clock time</th>
<th>Reads per hour</th>
<th>Peak virtual memory footprint</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bowtie, 1 thread (server)</td>
<td>18m:19s</td>
<td>18m:46s</td>
<td>28.3 M</td>
<td>1,353 MB</td>
<td>-</td>
</tr>
<tr>
<td>Bowtie, 2 threads (server)</td>
<td>20m:34s</td>
<td>10m:35s</td>
<td>50.1 M</td>
<td>1,363 MB</td>
<td>1.77x</td>
</tr>
<tr>
<td>Bowtie, 4 threads (server)</td>
<td>23m:09s</td>
<td>6m:01s</td>
<td>88.1 M</td>
<td>1,384 MB</td>
<td>3.12x</td>
</tr>
</tbody>
</table>

- Bowtie uses POSIX threads to exploit multi-processor computers
  - Reads are distributed across parallel threads
  - Threads synchronize when fetching reads, outputting results, etc.
  - Index is shared by all threads, so footprint does not increase substantially as # threads increases
- Table shows performance results for Bowtie v0.9.6 on 4-core Server with 1, 2, 4 threads

• Bowtie aligns all 1000-Genomes (Build 2) reads for human subject NA12892 on a 2.4 Ghz Core 2 workstation with 4 GB of RAM with 4 parallel threads:
  – 14.3x coverage, 935 M reads, 42.9 Gbases
  – Running time: 14 hrs – 1 overnight

Future Work

• Paired-end alignment
• Finding alignments with insertions and deletions
• ABI color-space support
Bowtie creators

Cole Trapnell  Steven Salzberg  Mihai Pop

Demand

“True understanding of how genes function requires knowledge of their expression patterns, their impact on all other genes and their effects on DNA structure and modifications. These data will have to be obtained across large numbers of cell types, individuals, environments and time points.”


Wanted: Scalable Algorithms

“...the overwhelming amounts of data being produced are the equivalent of taking a drink from a fire hose...”

“...grant-awarding bodies should start focusing on the back-end bioinformatics as much as the sequencing technology itself. And as the bioinformatics bottleneck threatens to limit instrument sales, manufacturers as a group have a massive incentive to unblock it.”

- Editorial, Nature Biotechnology 26, 1099 (Oct 2008)
PubMed was “searched in two-year increments for key words and the number of hits plotted over time.”

From the following article
What would you do if you could sequence everything?
Avak Kahvejian, John Quackenbush & John F Thompson
doi:10.1038/nbt1494