Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available. 2013 © Microsoft Corporation. All rights reserved.
New methods for the analysis of genome variation data

Richard Durbin
Wellcome Trust Sanger Institute
rd@sanger.ac.uk
The era of sequencing genomes

<table>
<thead>
<tr>
<th>Organism</th>
<th>Size (Mb)</th>
<th>Genes</th>
<th>Density (bp/Gen)</th>
<th>Cell Type</th>
<th>Completion date</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>H. influenzae</em></td>
<td>2</td>
<td>1,700</td>
<td>1/1kb</td>
<td>Bacterium</td>
<td>1995</td>
</tr>
<tr>
<td>Yeast</td>
<td>13</td>
<td>6,000</td>
<td>1/2kb</td>
<td>Eukaryotic cell</td>
<td>1996</td>
</tr>
<tr>
<td>Nematode</td>
<td>100</td>
<td>18,000</td>
<td>1/6kb</td>
<td>Animal</td>
<td>1998</td>
</tr>
<tr>
<td>Human</td>
<td>3000</td>
<td>20,000</td>
<td>1/150kb</td>
<td>Mammal</td>
<td>2000/3</td>
</tr>
</tbody>
</table>
Between 2000 and 2010 DNA sequencing costs dropped by five orders of magnitude.

Baseline information
Cost of genome sequencing compared with Moore’s law for computers

Log scale
100,000
10,000
1,000
100
10
1

$ per million DNA bases

Source: Broad Institute

This scaling, now generating PB data per year, challenges data analysis.
We can now study genetic variation directly by sequencing

- **2000:** First human genome ~$1$ billion
- **2009:** ~200 human genomes sequenced
  - Human genome $50k$
- **2012/13:** Thousands of genomes for research
  - Human genome $<5k$
- **2015/16:** Human genome $<1k$
  - Millions of human genomes for clinical usage, research, personal interest

We need to use the structure in the data for representation and inference
DNA sequences are related by evolution.
DNA sequences are related by evolution

- At each point in the genome we are descended from a common ancestor
DNA sequences are related by evolution

- At each point in the genome we are descended from a common ancestor

- Mutations since the common ancestor give rise to genetic variants shared by the descendants
DNA sequences are related by evolution

- At each point in the genome we are descended from a common ancestor
- Mutations since the common ancestor give rise to genetic variants shared by the descendants
- Some mutations are associated with disease: the frequency in disease cases is different from in controls

Individuals with a specific disease
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting Ancestral Recombination Graph describes the way that individual sequences in a sample are related
DNA sequences are related by evolution

- At each point in the genome we are descended from a common ancestor.
- Mutations since the common ancestor give rise to genetic variants shared by the descendants.
- Some mutations are associated with disease: the frequency in disease cases is different from in controls.

Individuals with a specific disease
The tree changes along the genome

• At each locus there is a tree
• Ancestral recombinations change the tree as you move along the genome
• The resulting Ancestral Recombination Graph describes the way that individual sequences in a sample are related
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting *Ancestral Recombination Graph* describes the way that individual sequences in a sample are related

```
```
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting Ancestral Recombination Graph describes the way that individual sequences in a sample are related
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting *Ancestral Recombination Graph* describes the way that individual sequences in a sample are related
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting Ancestral Recombination Graph describes the way that individual sequences in a sample are related
Inference on ARGs is hard

- There is a natural generative model for ARGs
  - Coalescent with recombination (Kingman, Tavare, Griffiths)
  - “Prune and graft” Markovian approximation is very good
- But infinitely many ARGs satisfy a data set
  - Sampling histories is hard, as is conditional sampling
The tree changes along the genome

- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting Ancestral Recombination Graph describes the way that individual sequences in a sample are related
Inference on ARGs is hard

- There is a natural generative model for ARGs
  - Coalescent with recombination (Kingman, Tavare, Griffiths)
  - “Prune and graft” Markovian approximation is very good
- But infinitely many ARGs satisfy a data set
  - Sampling histories is hard, as is conditional sampling
Sequencing accuracy requirements

http://www.ensembl.org/Homo_sapiens/Location/SequenceAlignment?db=core;g=ENSG00000187772;r=6:105569758-105579757
Sequencing accuracy requirements

1% false positives implies $10^{-5}$ errors per bp

http://www.ensembl.org/Homo_sapiens/Location/SequenceAlignment?db=core;g=ENSG00000187772;r=6:105569758-105579757
Sequencing accuracy requirements

1% false positives implies $10^{-5}$ errors per bp

$>1000$ individuals implies $<10^{-8}$ errors per bp per person

http://www.ensembl.org/Homo_sapiens/Location/SequenceAlignment?db=core;g=ENSG00000187772;r=6:105569758-105579757
1000 Genomes
A Deep Catalog of Human Genetic Variation

- International project to construct a next generation open access baseline data set for human genetics
  - Consortium with multiple centres, platforms, funders

- Aims
  - Find >95% accessible SNPs at allele frequencies above 1%, down towards 0.1% in coding regions
  - Also discover and characterize indels, structural variants
  - Identify a reference set of human genome sequences

- Driver for data and methods
Pilot project 179 samples (Nature 2010)

~100 per population: 4x Whole Genome Shotgun + Deep Exomes
Pilot project 179 samples (Nature 2010)
Phase 1: 1,092 samples (Nature 2012)

~100 per population: 4x Whole Genome Shotgun + Deep Exomes
Pilot project 179 samples (Nature 2010)
Phase 1: 1,092 samples (Nature 2012)
Phase 3: >2,500 samples sequence complete

~100 per population: 4x Whole Genome Shotgun + Deep Exomes
<table>
<thead>
<tr>
<th>Call set</th>
<th>Samples</th>
<th>SNPs</th>
<th>Sensitivity (HapMap3.3)</th>
<th>FP (in 59,721 monomorphic OMNI sites)</th>
<th>Genotype accuracy (HapMap 3 hets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pilot</td>
<td>179</td>
<td>15.2M</td>
<td>97.65%</td>
<td>43,606</td>
<td>97.75%</td>
</tr>
<tr>
<td>Phase I</td>
<td>1,092</td>
<td>38.0M</td>
<td>98.87%</td>
<td>1,261</td>
<td>99.24%</td>
</tr>
</tbody>
</table>

Also integrate ~500k exome calls, 1.4M indels, 14k large deletions, Omni2.5M genotypes

20.2 Tbp sequence
Distribution of functional variants

Common deleterious mutations are removed by selection.
## Per individual deleterious variant load

<table>
<thead>
<tr>
<th>Variant type</th>
<th>Number of variants (pop range)</th>
<th>Excess rare deleterious &lt;0.5%</th>
<th>Excess low freq deleterious 0.5%-5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>All sites</td>
<td>3.75M-4.7M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conserved sites (GERP&gt;2)</td>
<td>140k-200k</td>
<td>150-510</td>
<td>250-1.3k</td>
</tr>
<tr>
<td>Synonymous conserved</td>
<td>1.4k-1.9k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonSyn conserved</td>
<td>2.7k-4k</td>
<td>76-190</td>
<td>77-130</td>
</tr>
<tr>
<td>Loss-Of-Function conserved</td>
<td>116-175</td>
<td>6-13</td>
<td>11-31</td>
</tr>
<tr>
<td>Non-coding RNA conserved</td>
<td>200-290</td>
<td>1-3</td>
<td>4-13</td>
</tr>
<tr>
<td>Motif loss in TF peak (incomplete)</td>
<td>670-1020</td>
<td>8-22</td>
<td>20-110</td>
</tr>
</tbody>
</table>
What about identifying functional genetic variants?
What about identifying functional genetic variants?

*Linkage*

![Diagram showing allele frequency and effect size with genetic variants like MC4R, MODY genes.](Image)
What about identifying functional genetic variants?

**Linkage**

- Large
- Small

**Effect size**

- Rare
- 0.1%
- 1%
- 10%
- Common

**Allele frequency**

**Genome Wide Association**
Over a million people genotyped
Published Genome-Wide Associations through 07/2012

NHGRI GWA Catalog
www.genome.gov/GWASTudies
What about identifying functional genetic variants?

Linkage

Genome Wide Association

Allele frequency

Rare 0.1% 1% 10%

Small Large

Effect size

MC4R MODY genes FTO MC4R

TMEH1 SEC14B SEC14A TMEM164 A2 DAF LPBP PNMT PRKCA CTNNA5 TRAF1 GNBDA HSD17B11 CHST10 TMEM16A SHOC2 MTO12
What about identifying functional genetic variants?

**Linkage**

- **Large** vs. **Small**
- **Effect size** vs. **Allele frequency**

- **MC4R**
- **MODY genes**
- **Not much here (selection?)**
- **Beyond the scope of genetics!**

**Genome Wide Association**
What about identifying functional genetic variants?

A few genes in the middle zone have been found by candidate studies.
10,000 UK Genomes (2010-13)

- **Sequence**
  - 4,000 cohort samples genome wide: TwinsUK and ALSPAC
  - 6,000 exomes from samples with extreme phenotypes

- **Goals**
  - Direct association in sequenced samples
  - Impute new variants into additional samples, GWAS sets
  - Provide a sequence variation resource for use in further studies

- **Progress**
  - Data collection started late 2010
  - 4,004 WGS samples sequenced, ~6,300 exomes sequenced
  - Calls on ~3,700 WGS and ~6,200 exomes available
  - Greater than 50M variants, many novel
Cohorts initial analysis: 54 traits in 14 groups on 2,453 samples

Fig 3: QQplots and effect size comparison of singlepoint analysis of HDL (MAF>1%)

Fig 4: Association between CETP SNPS and HDL for ALSPAC WGS (N=655), ALSPAC HapMap2 imputed GWAs (N=2,791) and ALSPAC 1000G/UK10K imputed GWAs (N=2,791).

Some potential candidate new loci < 2.5% or < 10% poorly tagged
Alternative strategy: Impute full sequences into GWAS studies from reference sequences

Reference haplotypes via sequencing studies
e.g. 1000 GP, UK10K

Genotype data from GWAS
e.g. T2D, Arthritis, ...
Imputing missing data from reference sequences

Reference haplotypes via sequencing studies
eg. 1000 Genomes Project

Imputation of unobserved alleles via matching of shared haplotypes
Imputing missing data from reference sequences

Reference haplotypes via sequencing studies
eg. 1000 Genomes Project

Hidden Markov Model: “select and copy”, e.g. IMPUTE, MACH

Effectively an approximation to conditional sampling on the ARG

Imputation of unobserved alleles via matching of shared haplotypes
Imputing missing data from reference sequences

GWAS of imputed genotypes
- Increased power
- Better resolution
- Facilitates meta-analysis

CD hit region, chromosome 1

-Genome/Cont.
-Chromosomal position (Mb):

-cM from HAP1

-log10 (P-value)
1000 Genomes Project
Imputation Accuracy

Improvements from pilot to phase 1 using GAIN psoriasis dataset

<table>
<thead>
<tr>
<th>Reference</th>
<th>MAF 1-3%</th>
<th>MAF 3-5%</th>
<th>MAF &gt;5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000G Pilot</td>
<td>.69</td>
<td>.77</td>
<td>.91</td>
</tr>
<tr>
<td>1000G Phase 1</td>
<td>.82</td>
<td>.85</td>
<td>.94</td>
</tr>
</tbody>
</table>
The Haplotype Consortium: Chromosome 20 pilot

<table>
<thead>
<tr>
<th>COHORT</th>
<th>#samples</th>
<th>coverage</th>
<th>#SNPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000GP</td>
<td>379</td>
<td>4</td>
<td>338,766</td>
</tr>
<tr>
<td>UK10K</td>
<td>3,781</td>
<td>6.5</td>
<td>1,004,955</td>
</tr>
<tr>
<td>ORCADES</td>
<td>399</td>
<td>4</td>
<td>259,373</td>
</tr>
<tr>
<td>FINNS</td>
<td>1,941</td>
<td>4</td>
<td>315,539</td>
</tr>
<tr>
<td>GoNL</td>
<td>748</td>
<td>12</td>
<td>434,984</td>
</tr>
<tr>
<td>GoT2D</td>
<td>2,874</td>
<td>4</td>
<td>570,847</td>
</tr>
<tr>
<td>AMD</td>
<td>630</td>
<td>4</td>
<td>355,438</td>
</tr>
<tr>
<td>SARDINIA</td>
<td>2,120</td>
<td>4</td>
<td>372,632</td>
</tr>
<tr>
<td>MTF5</td>
<td>687</td>
<td>6</td>
<td>468,672</td>
</tr>
<tr>
<td>TOTALS</td>
<td>13,559</td>
<td>1,649,648</td>
<td></td>
</tr>
</tbody>
</table>

We created a union set of sites across 9 studies.

Only use sites with >10 minor alleles i.e. MAF > 0.03%:
- 427,589 sites
- 26.5% missing genotypes:
  - fill in by imputation

>30,000 samples available by March
<table>
<thead>
<tr>
<th><strong>1000 Genomes</strong></th>
<th><strong>UK10K</strong></th>
<th><strong>GoT2D</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Goncalo Abecasis</td>
<td>Nicole Soranzo</td>
<td>Mark McCarthy</td>
</tr>
<tr>
<td>Gil McVean</td>
<td>Nick Timpson</td>
<td>David Altshuler</td>
</tr>
<tr>
<td><strong>Sardinia</strong></td>
<td>George Davey-Smith</td>
<td>Mike Boekhne</td>
</tr>
<tr>
<td>Francesco Cucca</td>
<td>Tim Spector</td>
<td><strong>MTFS</strong></td>
</tr>
<tr>
<td>Serena Sanna</td>
<td></td>
<td>Scott Vrieze</td>
</tr>
<tr>
<td>David Schlessinger</td>
<td></td>
<td>Matt McGue</td>
</tr>
<tr>
<td>Carlo Sidore</td>
<td></td>
<td><strong>Bipolar</strong></td>
</tr>
<tr>
<td><strong>Finns</strong></td>
<td></td>
<td>Mike Boehnke</td>
</tr>
<tr>
<td>Aarno Palotie</td>
<td></td>
<td>Richard Myers</td>
</tr>
<tr>
<td><strong>GoNL</strong></td>
<td></td>
<td><strong>Helic</strong></td>
</tr>
<tr>
<td>Cisca Wijmenga</td>
<td></td>
<td>Eleftheria Zeggini</td>
</tr>
<tr>
<td>Paul I.W. de Bakker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morris A. Swertz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Androniki Menelaou</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HUNT</strong></td>
<td>Cristen Willer</td>
<td><strong>Crohns/UC</strong></td>
</tr>
<tr>
<td></td>
<td>Kristian Hveem</td>
<td>Jeff Barrett</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Carl Anderson</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Italian Isolates</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paolo Gasparini</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nicole Soranzo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniela Toniolo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nicola Piratsu</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Oxford</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warren Kretzschmar</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WT Sanger Institute</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shane McCarthy</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Michigan</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Christian Fuchsberger</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyun Min Kang</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Imputation is also used to integrate data from low coverage sequencing:

- To find the sequence of a single sample stand alone one needs to sequence at ~30x depth or more.
- To find low frequency variants we want to sequence many samples.
- Spread sequence across more samples.

Fraction of SNPs discovered in 2453 UK10K samples found in 1000GP Phase I:
- 98% at 10% MAF
- 96% at 1% MAF
- 47% at 0.1% MAF
Phase I power and genotyping accuracy

SNP detection

Genotyping accuracy

Exome
Whole genome

Mean $r^2$ with Omni microarrays

Exome
WGS (with LD)
WGS (no LD)

Non-reference allele count

Non-reference allele count
How to improve/speed up imputation?

- HMM based methods such as IMPUTE are computationally heavy
  - Quadratic (linear) in number of reference sequences
  - Typically use MCMC sampling
  - 1000s of CPU days for current data sets

- They will not scale to millions of genomes
New data structure for fast matching
Positional Burrows-Wheeler Transform
(PBWT)

\[
\text{PBW}[k]
\]

Reverse sorted prefixes at \( k \)

Matches are adjacent in the sort order
Analogous to BWT used by BWA, BowTie etc. for sequence matching
Updating sort order is linear time

\[
y^k[k] \quad y^{k+1}[k+1]
\]

Reverse sorted prefixes at \(k\)

Let \(u_i\) be the number of 0s in \(y\) before \(i\), i.e.
\[
u_i = i - \sum_{j<i} y_j
\]

And \(c\) be the total number of 1s in \(y\), i.e.
\[
c = \sum y_j
\]

Then \(i\) maps to \(u_i\) if \(y_i = 0\), and \(c + i - u_i\) if \(y_i = 1\).
New data structure for fast matching
Positional Burrows-Wheeler Transform (PBWT)

PBW[k]

Reverse sorted prefixes at k

Matches are adjacent in the sort order
Analogous to BWT used by BWA, BowTie etc. for sequence matching
Updating sort order is linear time

$$y^k[k] \quad y^{k+1}[k+1]$$

Let $$u_i$$ be the number of 0s in $$y$$ before $$i$$, i.e. $$u_i = i - \sum_{j<i} y_j$$

And $$c$$ be the total number of 1s in $$y$$, i.e. $$c = \sum y_j$$

Then $$i$$ maps to $$u_i$$ if $$y_i = 0$$, and $$c + i - u_i$$ if $$y_i = 1$$. 
New data structure for fast matching
Positional Burrows-Wheeler Transform (PBWT)

\[ PBW[k] \]

Reverse sorted prefixes at \( k \)

Matches are adjacent in the sort order
Analogous to BWT used by BWA, BowTie etc. for sequence matching
Updating sort order is linear time

Let $u_i$ be the number of 0s in $y$ before $i$, i.e. $u_i = i - \sum_{j<i} y_j$

And $c$ be the total number of 1s in $y$, i.e. $c = \sum y_j$

Then $i$ maps to $u_i$ if $y_i = 0$, and $c + i - u_i$ if $y_i = 1$. 
Data compression – run length encoding

- Encode runs in bytes using
  - 1 bit for the value
  - 2 bits select run length units: 1 or 64 or 2048
  - 5(6) bits give number of units

- Simulate 100k sequences of length 20Mbp with ARG simulator MaCS (Chen et al. 2009)
  - 370,264 sites (one per 54bp): 37GB raw output
  - gzip compresses to 1.02GB (~35x compression)
  - PBWT compresses to 7.7MB (~4800x compression)

- Native order run length encodes to ~2GB
Updating sort order is linear time

\[ y^k[k] \quad y^{k+1}[k+1] \]

Reverse sorted prefixes at \( k \)

Let \( u_i \) be the number of 0s in \( y \) before \( i \), i.e. \( u_i = i - \sum_{j<i} y_j \)

And \( c \) be the total number of 1s in \( y \), i.e. \( c = \sum y_j \)

Then \( i \) maps to \( u_i \) if \( y_i = 0 \), and \( c + i - u_i \) if \( y_i = 1 \).
Data compression – run length encoding

- Encode runs in bytes using
  - 1 bit for the value
  - 2 bits select run length units: 1 or 64 or 2048
  - 5(6) bits give number of units

- Simulate 100k sequences of length 20Mbp with ARG simulator MaCS (Chen et al. 2009)
  - 370,264 sites (one per 54bp): 37GB raw output
  - gzip compresses to 1.02GB (~35x compression)
  - PBWT compresses to 7.7MB (~4800x compression)
- Native order run length encodes to ~2GB
Compression performance

Simulated data subsets

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>50000</th>
<th>100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBWT</td>
<td>1685629</td>
<td>1956360</td>
<td>2783732</td>
<td>3372188</td>
<td>4145516</td>
<td>5688290</td>
<td>7697194</td>
</tr>
<tr>
<td>haps.gz</td>
<td>10515008</td>
<td>21340464</td>
<td>53246970</td>
<td>105558782</td>
<td>209332249</td>
<td>517432833</td>
<td>1024614062</td>
</tr>
<tr>
<td>factor</td>
<td>6.2</td>
<td>10.9</td>
<td>19.1</td>
<td>31.3</td>
<td>50.5</td>
<td>91.0</td>
<td>133.1</td>
</tr>
<tr>
<td>bytes/site</td>
<td>4.6</td>
<td>5.3</td>
<td>7.5</td>
<td>9.1</td>
<td>11.2</td>
<td>15.4</td>
<td>20.8</td>
</tr>
</tbody>
</table>

"Real" data: 1000 Genomes phase I chromosome I
2184 chromosomes, 3007196 sites
PBWT 51186641
gzip 302883517
factor 5.9
Maximal matches of new sequences to a reference panel

- Update rule the same as for sort update
- Option 1: build indexes on top of PBWT
  - Very fast, effectively independent of the size of the reference panel so $O(N)$ time
  - Quite memory hungry, $\sim 13NM$ bytes
- Option 2: match a batch of new sequences as you pass once through the PBWT
  - $O(NM)$ time, but very memory efficient
Data compression — run length encoding

- Encode runs in bytes using
  - 1 bit for the value
  - 2 bits select run length units: 1 or 64 or 2048
  - 5(6) bits give number of units

- Simulate 100k sequences of length 20Mbp with ARG simulator MaCS (Chen et al. 2009)
  - 370,264 sites (one per 54bp): 37GB raw output
  - gzip compresses to 1.02GB (~35x compression)
  - PBWT compresses to 7.7MB (~4800x compression)

- Native order run length encodes to ~2GB
Updating sort order is linear time

Let $u_i$ be the number of 0s in $y$ before $i$, i.e. $u_i = i - \sum_{j<i} y_j$

And $c$ be the total number of 1s in $y$, i.e. $c = \sum y_j$

Then $i$ maps to $u_i$ if $y_i = 0$, and $c + i - u_i$ if $y_i = 1$. 
Data compression – run length encoding

- Encode runs in bytes using
  - 1 bit for the value
  - 2 bits select run length units: 1 or 64 or 2048
  - 5(6) bits give number of units

- Simulate 100k sequences of length 20Mbp with ARG simulator MaCS (Chen et al. 2009)
  - 370,264 sites (one per 54bp): 37GB raw output
  - gzip compresses to 1.02GB (~35x compression)
  - PBWT compresses to 7.7MB (~4800x compression)
- Native order run length encodes to ~2GB
Compression performance

Simulated data subsets

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>50000</th>
<th>100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBWT</td>
<td>1685629</td>
<td>1956360</td>
<td>2783732</td>
<td>3372188</td>
<td>4145516</td>
<td>5688290</td>
<td>7697194</td>
</tr>
<tr>
<td>haps.gz</td>
<td>10515008</td>
<td>21340464</td>
<td>53246970</td>
<td>105558782</td>
<td>209332249</td>
<td>517432833</td>
<td>1024614062</td>
</tr>
<tr>
<td>factor</td>
<td>6.2</td>
<td>10.9</td>
<td>19.1</td>
<td>31.3</td>
<td>50.5</td>
<td>91.0</td>
<td>133.1</td>
</tr>
<tr>
<td>bytes/site</td>
<td>4.6</td>
<td>5.3</td>
<td>7.5</td>
<td>9.1</td>
<td>11.2</td>
<td>15.4</td>
<td>20.8</td>
</tr>
</tbody>
</table>

"Real" data: 1000 Genomes phase I chromosome I
2184 chromosomes, 3007196 sites
PBWT 51186641
gzip 302883517
factor 5.9
Maximal matches of new sequences to a reference panel

- Update rule the same as for sort update
- Option 1: build indexes on top of PBWT
  - Very fast, effectively independent of the size of the reference panel so $O(N)$ time
  - Quite memory hungry, $\sim 13NM$ bytes
- Option 2: match a batch of new sequences as you pass once through the PBWT
  - $O(NM)$ time, but very memory efficient
# Time to match 1000 new sequences

<table>
<thead>
<tr>
<th>Reference panel size</th>
<th>1,000</th>
<th>2,000</th>
<th>5,000</th>
<th>10,000</th>
<th>20,000</th>
<th>50,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>52.1</td>
<td>103.8</td>
<td>258.9</td>
<td>519.2</td>
<td>1035.5</td>
<td>2582.6</td>
</tr>
<tr>
<td>Indexed</td>
<td>0.9u</td>
<td>0.9u</td>
<td>0.9u</td>
<td>0.9u</td>
<td>1.1u</td>
<td>1.7u</td>
</tr>
<tr>
<td></td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.2s</td>
<td>0.5s</td>
<td>15s</td>
</tr>
<tr>
<td>Batch</td>
<td>2.3u</td>
<td>2.5u</td>
<td>3.5u</td>
<td>4.8u</td>
<td>6.8u</td>
<td>12.1u</td>
</tr>
<tr>
<td></td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
<td>0.1s</td>
</tr>
</tbody>
</table>

The reference panel here is pseudo-genotype array data, made of 10% of sites with MAF > 0.05 from the full sequence simulation.

The naïve method compares each sequence to each previous sequence, efficiently.
What about inference?

- Prediction is closely related to compression

- Consider a generative model for $y$ given $d$
  - e.g. Markov given by $p(y_{i+1} \mid y_i, d_i)$
  - Iterate to generate whole sequence sets
Imputation (very preliminary)

- **Empirically** \( p(y_{i+1} \neq y_i \mid d_i) \approx e^{-\alpha - \beta d} \)
  - Expected given \( d \sim \text{Gumbel extreme value} \)
- Imputation under Markov model is local, so very fast

26,618 haps, 428,131 sites (~60Mb)
25 mins on Mac air – results a bit worse than using IMPUTE
What about inference?

• Prediction is closely related to compression

• Consider a generative model for $y$ given $d$

• e.g. Markov given by $p(y_{i+1} | y_i, d_i)$

• Iterate to generate whole sequence sets
Imputation (very preliminary)

- Empirically $p(y_{i+1} \neq y_i \mid d_i) \approx e^{-\alpha - \beta d}$
  - Expected given $d \sim$ Gumbel extreme value

- Imputation under Markov model is local, so very fast

26,618 haps, 428,131 sites (~60Mb)
25 mins on Mac air – results a bit worse than using IMPUTE
Alternative model: tree defined by $d_i$?

- Potentially attractive relationship to ARG
- But data not consistent with tree in hard form
  - Sample trees consistent with data weighting via $d_i$
Imputation (very preliminary)

- Empirically $p(y_{i+1} \neq y_i | d_i) \approx e^{-\alpha - \beta d}$
  - Expected given $d \sim$ Gumbel extreme value
- Imputation under Markov model is local, so very fast

26,618 haps, 428,131 sites (~60Mb)
25 mins on Mac air – results a bit worse than using IMPUTE
Alternative model: tree defined by $d_i$?

- Potentially attractive relationship to ARG
- But data not consistent with tree in hard form
  - Sample trees consistent with data weighting via $d_i$
Back to the raw data
BW methods help there too

- Collect shotgun sequencing reads
  - Random fragments from the whole genome
  - Can be enriched e.g. for exome
- Map the reads to the reference genome
  - Potential problems in repetitive areas
  - Potential alignment problems
- Detect variants based on the multiple alignment of reads
  - Statistical issues allowing for errors and sampling
Many modern mappers (e.g. bwa) use the standard Burrows-Wheeler Transform. Non-unique match is an interval in Suffix Array.

```plaintext
\[ i \quad SA[i] \]

\begin{align*}
0 & \quad 11 & \$ & \text{L(si)=8, U(si)=10} \\
1 & \quad 10 & \text{i}\$ \\
2 & \quad 7 & \text{ippi}\$ \\
3 & \quad 4 & \text{issippi}\$ \\
4 & \quad 1 & \text{ississippi}\$ \\
5 & \quad 0 & \text{mississippi}\$ \\
6 & \quad 9 & \text{pi}\$ \\
7 & \quad 8 & \text{ppi}\$ \\
8 & \quad 6 & \text{sippi}\$ \\
9 & \quad 3 & \text{sissippi}\$ \\
10 & \quad 5 & \text{ssippi}\$ \\
11 & \quad 2 & \text{ssissippi}\$
\end{align*}
```
Many modern mappers (e.g. bwa) use the standard Burrows-Wheeler Transform. Non-unique match is an interval in Suffix Array.

```
  i  SA[i]
0   11  $          L(si)=8, U(si)=10
1   10  i$          FM approach to matching:
2    7  ippi$       update L() and U() as you extend the search string
3    4  issippi$    L()=0, U()=12
4    1  issississippi$ L(i)=1, U(i)=5
5    0  mississippi$ L(si)=8, U(si)=10
6    9  pi$        Constant time update gives O(M) search
7    8  ppi$
8    6  sippi$
9    3  sissississippi$
10   5  ssippi$
11   2  ssissississippi$
```

mississippi
Update uses FM-index
Ferragina and Manzini 2000

mississippi

\[ L(aS) = C[a] + P[a, L(S)] \]
\[ U(aS) = C[a] + P[a, U(S)] \]

Where

- \( C[a] \) is the number of letters less than \( a \) in the target string \( X \)
- \( P[a, i] \) is the number of times \( a \) occurs in \( BW[j] \) for \( j < i \)
- \( BW[i] = X[SA[i]-1] \) is the Burrows-Wheeler transform
Errors due to mapping problems

Rare, population specific “SNPs” are clustered near centromere. Many of these are likely to be artefacts, e.g. ~1% of the total.
Errors due to alignment at indels

- Control using local alignment uncertainty (BAQ: Li Heng), realignment, reassembly ...
We are mapping to the wrong reference!
Genetic variation by assembly

- Reference free variation detection
  - De novo mutations by comparing child to parents
  - Somatic cancer mutations by comparing tumour to normal
  - Population variation by identifying segregating sequence
  - Individual variation by comparing to reference

- Could assemble first then compare contig sets
  - Time consuming

- We only want to assemble the differences between the samples
Genetic variation by assembly

Find unique reads/k-mers, then *locally* construct the string graph around reads containing these k-mers.
Genetic variation by assembly

Evaluate evidence for haplotypes with Dindel (Albers et al. 2010) Bayesian methods
Two yeast strains co-assembled

...AAAGATATGCGCAGTGTTGATTGGAGA...
...AAAGATATGCGCAGTGTTGATTGGAGA...
Two yeast strains co-assembled

![Diagram showing two yeast strains co-assembled with shared and unique sequences indicated.]
Example from 1000 Genomes: 4bp MNP

20 16240965 . TCTC CAAT 120 PASS AF=0.31894;NumReads=1107;VarDP=43

NB: only 3 reads in BAM: 43 reads in assembly
Using BWT/FM index for assembly

- The FM-Index of the reads is closely related to an assembly graph
  - It efficiently encodes all 1 base extensions of a k-mer
  - It can compute non-transitive overlap structure of the reads and hence string graph unitigs in $O(N)$ time
  - Also can derive all de Bruijn graphs for $k$ up to read length
- SGA assembler (Jared Simpson)
  - Compression of read BWT vastly reduces memory usage

“Efficient de novo assembly of large genomes using compressed data structures”, Simpson and Durbin, 2012

- Open question: is Myers string graph the same as de Bruijn graph after perfect read threading?
Final big question

- Can we merge PBWT and BWT assembly?
- Build a model from all existing primary data to use efficiently for interpreting new data
  - Use ML techniques, perhaps on probabilistic model capturing genetic structure
- Very large distributed data sets (PB now)
  - The more people, the more information

Is anyone interested to help? Collaborators, postdocs....

- Of course, there are lots of other big data machine learning problems in genomics
Acknowledgements

Andrew Brown, Milan Malinsky, Stephan Schiffels, Vladimir Shchur, Vagheesh Narasimhan, Zhihao Ding, Yasin Memari, Jared Simpson, Heng Li

Thomas Keane, Sendu Bala, Petr Danecek, Shane McCarthy, core sequencing teams
BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables

Cho-Jui Hsieh
The University of Texas at Austin

NIPS
Lake Tahoe, Nevada
Dec 8, 2013

Joint work with M. Sustik, I. Dhillon, P. Ravikumar and R. Poldrack
Goal: Reveal functional connections between regions of the brain. 
(Sun et al, 2009; Smith et al, 2011; Varoquaux et al, 2010; Ng et al, 2011)

\[ p = 228,483 \text{ voxels.} \]

Figure from (Varoquaux et al., 2010)
BIG & QUIC: Sparse Inverse Covariance Estimation for a Million Variables

Cho-Jui Hsieh
The University of Texas at Austin

NIPS
Lake Tahoe, Nevada
Dec 8, 2013

Joint work with M. Sustik, I. Dhillon, P. Ravikumar and R. Poldrack
Goal: Reveal functional connections between regions of the brain.
(Sun et al, 2009; Smith et al, 2011; Varoquaux et al, 2010; Ng et al, 2011)

$p = 228,483$ voxels.

**Input**

**Output**

Figure from (Varoquaux et al, 2010)
Other Applications

- Gene regulatory network discovery:
  (Schafer & Strimmer 2005; Andrei & Kendziorski 2009; Menendez et al, 2010; Yin and Li, 2011)

- Financial Data Analysis:
  - Model dependencies in multivariate time series (Xuan & Murphy, 2007).
  - Sparse high dimensional models in economics (Fan et al, 2011).

- Social Network Analysis / Web data:
  - Model co-authorship networks (Goldenberg & Moore, 2005).
  - Model item-item similarity for recommender system (Agarwal et al, 2011).

- Climate Data Analysis (Chen et al., 2010).
- Anomaly Detection (Ide et al, 2009).
Given: $n$ i.i.d. samples $\{y_1, \ldots, y_n\}$, $y_i \in \mathbb{R}^p$, $y_i \sim \mathcal{N}(\mu, \Sigma)$,

An example – Chain graph: $y_j = 0.5y_{j-1} + \mathcal{N}(0, 1)$

$$
\Sigma = \begin{pmatrix}
1.33 & 0.67 & 0.33 & 0.17 \\
0.67 & 1.33 & 0.67 & 0.33 \\
0.33 & 0.67 & 1.33 & 0.67 \\
0.17 & 0.33 & 0.67 & 1.33 \\
\end{pmatrix}, \quad \Sigma^{-1} = \begin{pmatrix}
1 & -0.5 & 0 & 0 \\
-0.5 & 1.25 & -0.5 & 0 \\
0 & -0.5 & 1.25 & -0.5 \\
0 & 0 & -0.5 & 1 \\
\end{pmatrix}
$$

Conditional independence is reflected as zeros in $\Sigma^{-1}$:

$\Sigma^{-1}_{ij} = 0 \Leftrightarrow y_i$ and $y_j$ are conditionally independent given other variables.
L1-regularized inverse covariance selection

- Goal: Estimate the inverse covariance matrix in the high dimensional setting: $p(\# \text{ variables}) \gg n(\# \text{ samples})$
- Add $\ell_1$ regularization – a sparse inverse covariance matrix is preferred.
- The $\ell_1$-regularized Maximum Likelihood Estimator:

$$
\Sigma^{-1} = \arg \min_{X > 0} \left\{ -\log \det X + \text{tr}(SX) + \lambda \|X\|_1 \right\} = \arg \min_{X > 0} f(X),
$$

where $\|X\|_1 = \sum_{i,j=1}^{n} |X_{ij}|$. 

Big & Quic: Sparse Inverse Covariance Estimation
Scalability

- Block coordinate ascent (Banerjee et al., 2007), Graphical Lasso (Friedman et al., 2007).
- VSM, PSM, SINCO, IPM, PQN, ALM (2008-2010).
  ALM solves $p = 1000$ in 300 secs.
- QUIC: Newton type method (Hsieh et al., 2011)
  Solves $p = 1000$ in 10 secs, $p = 10,000$ in half hour.
- All the above methods require $O(p^2)$ memory, cannot solve problems with $p > 30,000$.
- Need for scalability: FMRI dataset has more than 220,000 variables.
Scalability

- Block coordinate ascent (Banerjee et al, 2007), Graphical Lasso (Friedman et al, 2007).
- VSM, PSM, SINCO, IPM, PQN, ALM (2008-2010).
  ALM solves $p = 1000$ in 300 secs.
- QUIC: Newton type method (Hsieh et al, 2011)
  Solves $p = 1000$ in 10 secs, $p = 10,000$ in half hour.
- All the above methods require $O(p^2)$ memory, cannot solve problems with $p > 30,000$.
- Need for scalability: FMRI dataset has more than 220,000 variables
- BIGQUIC (2013):
  $p = 1,000,000$ (1 trillion parameters) in 22.9 hrs with 32 GBytes memory (using a single machine with 32 cores).
Our innovations

- **Main Ingredients:**
  1. Second-order Newton-like method (QUIC) → quadratic convergence rate.
  2. Memory-efficient scheme using block coordinate descent (BigQUIC) → scale to one million variables.
  3. Approximate Hessian computation (BigQUIC) → super-linear convergence rate.
- Split smooth and non-smooth terms: \( f(X) = g(X) + h(X) \), where

\[
g(X) = -\log \det X + \text{tr}(SX) \quad \text{and} \quad h(X) = \lambda \|X\|_1.
\]

- Form quadratic approximation for \( g(X_t + \Delta) \):

\[
\bar{g}_X(\Delta) = \text{tr}((S - W_t)\Delta) + (1/2) \text{vec}(\Delta)^T (W_t \otimes W_t) \text{vec}(\Delta)
- \log \det X_t + \text{tr}(SX_t),
\]

where \( W_t = (X_t)^{-1} = \frac{\partial}{\partial X} \log \det(X) \big|_{X=X_t} \).

- Define the generalized Newton direction:

\[
D_t = \arg \min_{\Delta} \bar{g}_X(\Delta) + \lambda \|X_t + \Delta\|_1.
\]

- Solve by coordinate descent (Hsieh et al, 2011) or other methods (Olsen et al, 2012).
Coordinate Descent Updates

- Use coordinate descent to solve:

\[
\arg \min_D \{\tilde{g}_X(D) + \lambda \|X + D\|_1\}.
\]

- Closed form solution for each coordinate descent update:

\[
D_{ij} \leftarrow -c + S(c - b/a, \lambda/a),
\]

where \( S(z, r) = \text{sign}(z) \max\{|z| - r, 0\} \) is the soft-thresholding function, \( a = W_{ij}^2 + W_{ii}W_{jj} \), \( b = S_{ij} - W_{ij} + w_i^T D w_j \), and \( c = X_{ij} + D_{ij} \).

- The main cost is in computing \( w_i^T D w_j \),

where \( w_i, w_j \) are \( i \)-th and \( j \)-th columns of \( W = X^{-1} \).
QUIC: QUadratic approximation for sparse Inverse Covariance estimation

**Input:** Empirical covariance matrix $S$, scalar $\lambda$, initial $X_0$.

For $t = 0, 1, \ldots$

1. **Variable selection:** select a free set of $m \ll p^2$ variables.
2. **Use coordinate descent to find descent direction:**
   
   \[ D_t = \arg \min_{\Delta} \tilde{f}_{X_t}(X_t + \Delta) \text{ over set of free variables, (A Lasso problem.)} \]
3. **Line Search:** use an Armijo-rule based step-size selection to get $\alpha$ s.t. $X_{t+1} = X_t + \alpha D_t$ is
   - positive definite,
   - satisfies a sufficient decrease condition $f(X_t + \alpha D_t) \leq f(X_t) + \alpha \sigma \Delta_t$.

(Cholesky factorization of $X_t + \alpha D_t$)
Difficulties in Scaling QUIC

Consider the case that $p \approx 1$ million, $m = \|X_t\|_0 \approx 50$ million.

- Coordinate descent requires $X_t$ and $W = X_t^{-1}$,
  - needs $O(p^2)$ storage
  - needs $O(mp)$ computation per sweep, where $m = \|X_t\|_0$
- Line search (compute determinant using Cholesky factorization).
  - needs $O(p^2)$ storage
  - needs $O(p^3)$ computation
- Assume we can store $M$ columns of $W$ in memory.
- Coordinate descent update $(i, j)$: compute $w_i^T D w_j$.
- If $w_i, w_j$ are not in memory: recompute by CG:
  \[ Xw_i = e_i : O(T_{CG}) \text{ time.} \]
\( w_1, w_2, w_3, w_4 \) stored in memory.
Cache hit, do not need to recompute $w_i, w_j$.

Update $(1,4)$
Need $w_{(1)}, w_{(4)}$

Memory Cache ($M=4$)
Cache miss, recompute $w_i, w_j$.

Update (6,9)  
Need $w_{(6)}, w_{(9)}$

Memory Cache  $(M=4)$

$W_{(1)} W_{(2)} W_{(3)} W_{(4)}  W_{(6)} W_{(9)}$

?  ?
Coordinate Updates – ideal case

- Want to find update sequence that **minimizes** number of cache misses: probably NP Hard.
- Our strategy: update variables **block by block**.
- The ideal case: there exists a partition \( \{S_1, \ldots, S_k\} \) such that all free sets are in diagonal blocks:

![Free Set](image)

- Only requires \( p \) column evaluations.
General case: block diagonal + sparse

- If the block partition is not perfect:
  extra column computations can be characterized by boundary nodes.
- Given a partition \( \{S_1, \ldots, S_k\} \), we define boundary nodes as

\[
B(S_q) \equiv \{j \mid j \in S_q \text{ and } \exists i \in S_z, z \neq q \text{ s.t. } F_{ij} = 1\},
\]

where \( F \) is adjacency matrix of the free set.

![Diagram showing a grid and a graph with boundary nodes highlighted.](image)
The number of columns to be computed in one sweep is

\[ p + \sum_{q} |B(S_q)|. \]

Can be upper bounded by

\[ p + \sum_{q} |B(S_q)| \leq p + \sum_{z \neq q} \sum_{i \in S_z} \sum_{j \in S_q} F_{ij}. \]

Use Graph Clustering (METIS or Graclus) to find the partition.

Example: on fMRI dataset \((p = 0.228\, \text{million})\) with 20 blocks, random partition: need \textbf{1.6 million} column computations. graph clustering: need \textbf{0.237 million} column computations.
• Block co-ordinate descent with clustering,
  • needs $O(p^2) \to O(m + p^2/k)$ storage
  • needs $O(mp) \to O(mp)$ computation per sweep, where $m = \|X_t\|_0$
• Line search (compute determinant of a big sparse matrix).
  • needs $O(p^2)$ storage
  • needs $O(p^3)$ computation
Given sparse matrix \( A = X_t + \alpha D \), we need to
1. Check its positive definiteness.
2. Compute \( \log \det(A) \).

Our approach computes \( \log \det(A) \) in \( O(mp) \) time.

Cholesky factorization in QUIC requires \( O(p^3) \) computation.

If \( A = \begin{pmatrix} a & b^T \\ b & C \end{pmatrix} \),

- \( \det(A) = \det(C)(a - b^T C^{-1}b) \)
- \( A \) is positive definite iff \( C \) is positive definite and \( (a - b^T C^{-1}b) > 0 \).
- \( C \) is sparse, so can compute \( C^{-1}b \) using Conjugate Gradient (CG).
- Time complexity: \( T_{CG} = O(mT) \), where \( T \) is number of CG iterations.
• Block co-ordinate descent with clustering,
  • needs $O(p^2) \rightarrow O(m + p^2/k)$ storage
  • needs $O(mp) \rightarrow O(mp)$ computation per sweep, where $m = \|X_t\|_0$

• Line search (compute determinant of a big sparse matrix).
  • needs $O(p^2) \rightarrow O(p)$ storage
  • needs $O(p^3) \rightarrow O(mp)$ computation
**BIGQUIC**

**Input:** Samples $Y$, scalar $\lambda$, initial $X_0$.

For $t = 0, 1, \ldots$

1. **Variable selection:** select a free set of $m \ll p^2$ variables.
2. **Construct a partition by clustering.**
3. Run **block coordinate descent** to find descent direction:
   $$D_t = \arg \min_{\Delta} \tilde{f}_{X_t}(X_t + \Delta)$$
   over set of free variables.
4. **Line Search:** use an Armijo-rule based step-size selection to get $\alpha$ s.t.
   $$X_{t+1} = X_t + \alpha D_t$$
   is
   - positive definite,
   - satisfies a sufficient decrease condition
     $$f(X_t + \alpha D_t) \leq f(X_t) + \alpha \sigma \Delta_t.$$  
   (Schur complement with conjugate gradient method.)
Recall $W = X^{-1}$.

When each $w_i$ is computed by CG ($Xw_i = e_i$):

- The gradient $\nabla_{ij} g(X) = S_{ij} - W_{ij}$ on free set can be computed once and stored in memory.
- Hessian ($w_i^T D w_j$ in coordinate updates) needs to be repeatedly computed.

To reduce the time overhead, Hessian should be computed approximately.

**Theorem:** the convergence rate is quadratic if $\| Xw_i - e_i \| = O(\| \nabla^S f(X_t) \|)$, where

$$
\nabla^S_{ij} f(X) = \begin{cases} 
\nabla_{ij} g(X) + \text{sign}(X_{ij})\lambda & \text{if } X_{ij} \neq 0, \\
\text{sign}(\nabla_{ij} g(X)) \max(\| \nabla_{ij} g(X) \| - \lambda, 0) & \text{if } X_{ij} = 0.
\end{cases}
$$
Experimental results (scalability)

Figure: BigQUIC can solve one million dimensional problems.
**BigQUIC** is faster even for medium size problems.

**Figure:** Comparison on FMRI data with a $p = 20000$ subset (maximum dimension that previous methods can handle).
Results on FMRI dataset

- 228,483 voxels, 518 time points.
- $\lambda = 0.6 \rightarrow$ average degree 8, BIGQUIC took 5 hours.
- $\lambda = 0.5 \rightarrow$ average degree 38, BIGQUIC took 21 hours.

Findings:
- Voxels with large degree were generally found in the gray matter.
- Can detect meaningful brain modules by modularity clustering.
Conclusions

- **BigQUIC**: Memory efficient quadratic approximation method for sparse inverse covariance estimation.

- **Our contributions:**
  - Computing Newton direction:
    - Coordinate descent $\rightarrow$ **block coordinate descent with clustering**.
    - Memory complexity: $O(p^2) \rightarrow O(m + p^2/k)$.
    - Time complexity: $O(mp) \rightarrow O(mp)$.
  - Line search (computing determinant of a big sparse matrix)
    - Cholesky factorization $\rightarrow$ **Schur complement with conjugate gradient method**.
    - Memory complexity: $O(p^2) \rightarrow O(p)$.
    - Time complexity: $O(p^3) \rightarrow O(mp)$.
  - **Inexact Hessian computation with super-linear convergence.**