Microsoft Research

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.

wellcome trust sanger institute

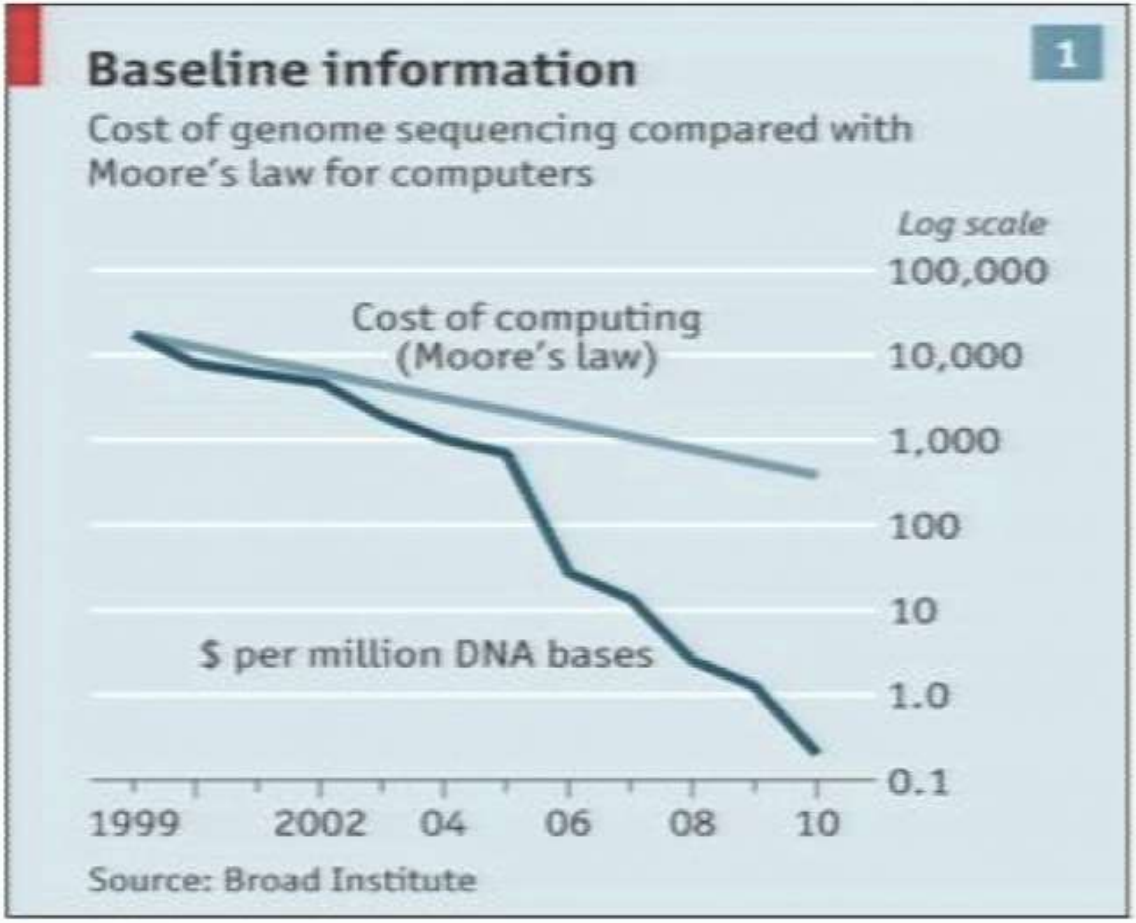
New methods for the analysis of genome variation data

Richard Durbin
Wellcome Trust Sanger Institute
rd@sanger.ac.uk

The era of sequencing genomes

	Size (Mb)	Genes			Completion date
H. influenzae	2	1,700	1/1kb	Bacterium	1995
Yeast	13	6,000	1/2kb	Eukaryotic cell	1996
Nematode	100	18,000	1/6kb	Animal	1998
Human	3000	20,000	1/150kb	Mammal	2000/3

Between 2000 and 2010 DNA sequencing costs dropped by five orders of magnitude



This scaling, now generating PB data per year, challenges data analysis

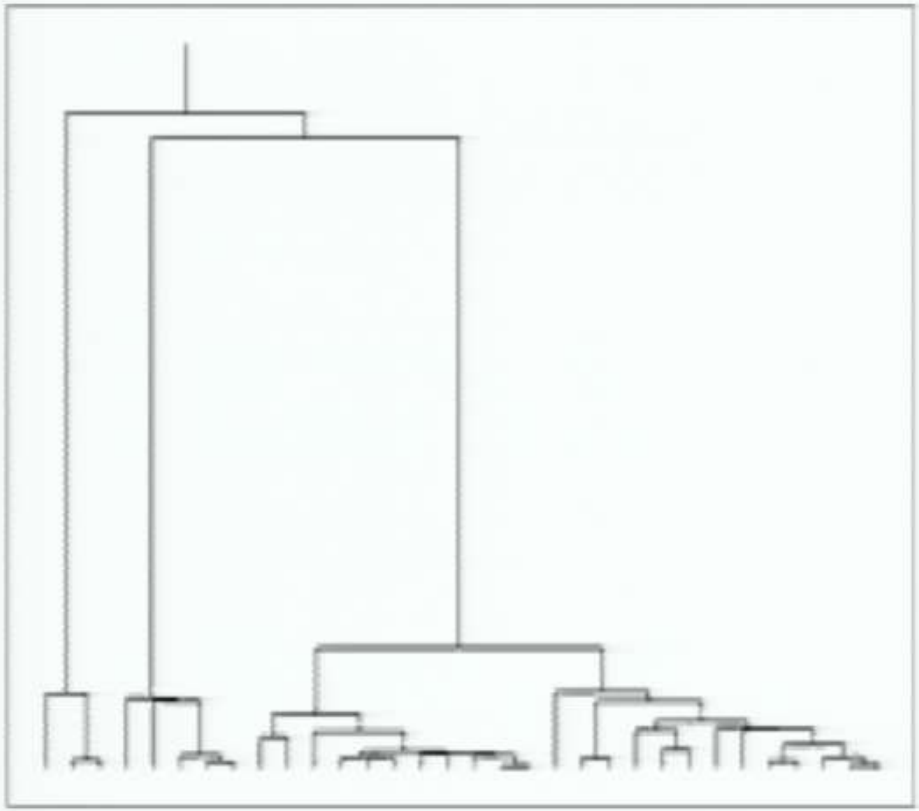
We can now study genetic variation directly by sequencing

- 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

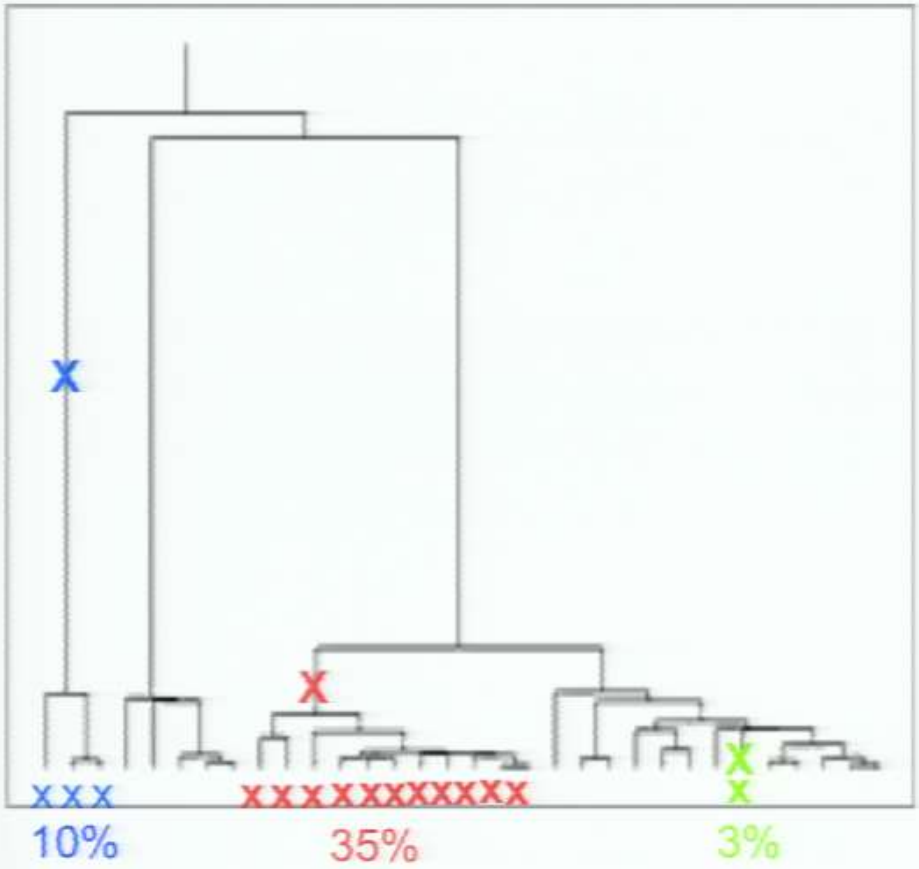
DNA sequences are related by evolution

DNA sequences are related by evolution



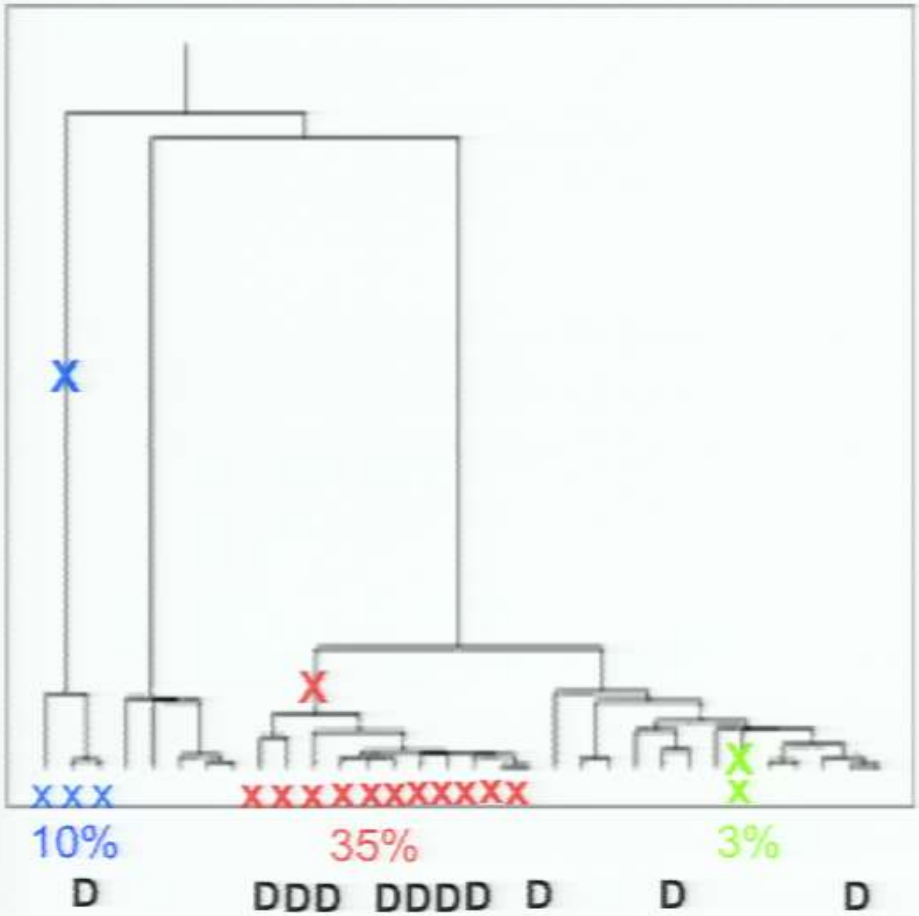
- At each point in the genome we are descended from a common ancestor

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



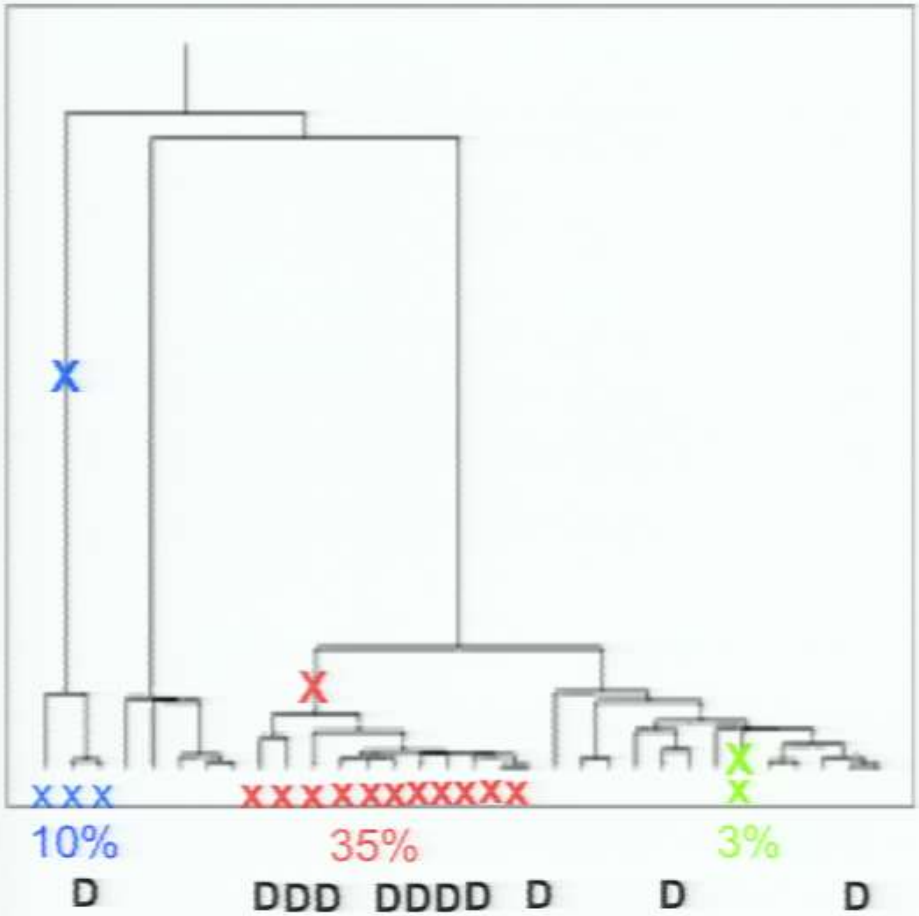
Individuals with a specific disease

- 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

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



Individuals with a specific disease

- 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

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

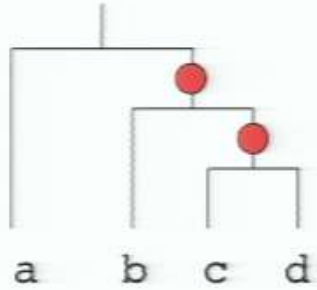
- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting *Ancestral Recombination Graph* describes the way that individual sequences in a sample are related

```
a ..C..G..A..
b ..T..G..C..
c ..T..A..A..
d ..T..A..C..
```

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

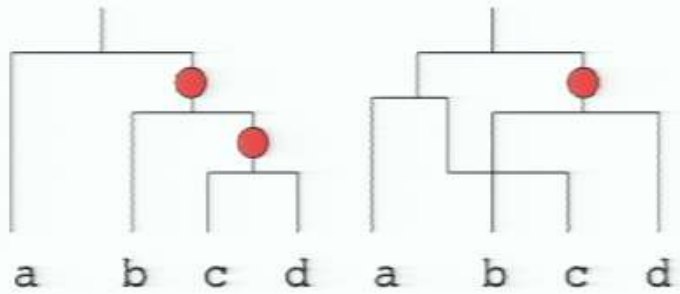
```
a ..C..G..A..   0 0 0
b ..T..G..C..   1 0 1
c ..T..A..A..   1 1 0
d ..T..A..C..   1 1 1
```



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

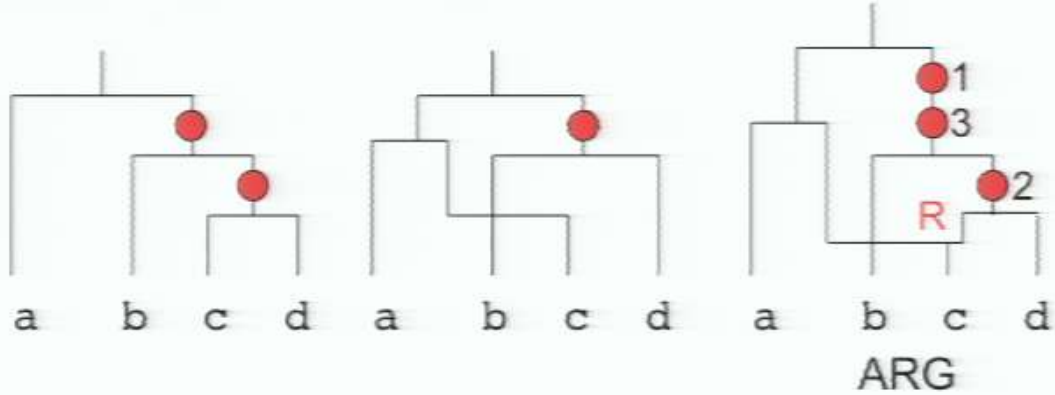
```
a ..C..G..A..   0 0 0
b ..T..G..C..   1 0 1
c ..T..A..A..   1 1 0
d ..T..A..C..   1 1 1
```



The tree changes along the genome

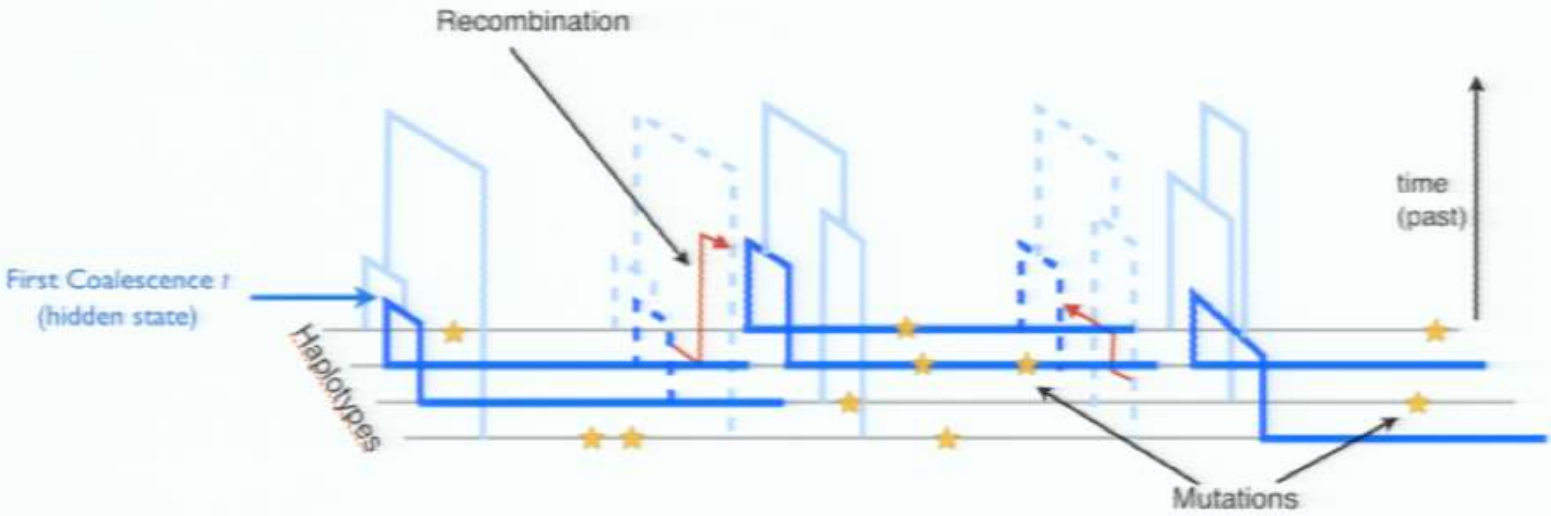
- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting *Ancestral Recombination Graph* describes the way that individual sequences in a sample are related

```
a ..C..G..A..   0 0 0
b ..T..G..C..   1 0 1
c ..T..A..A..   1 1 0
d ..T..A..C..   1 1 1
```



Inference on ARGs is hard

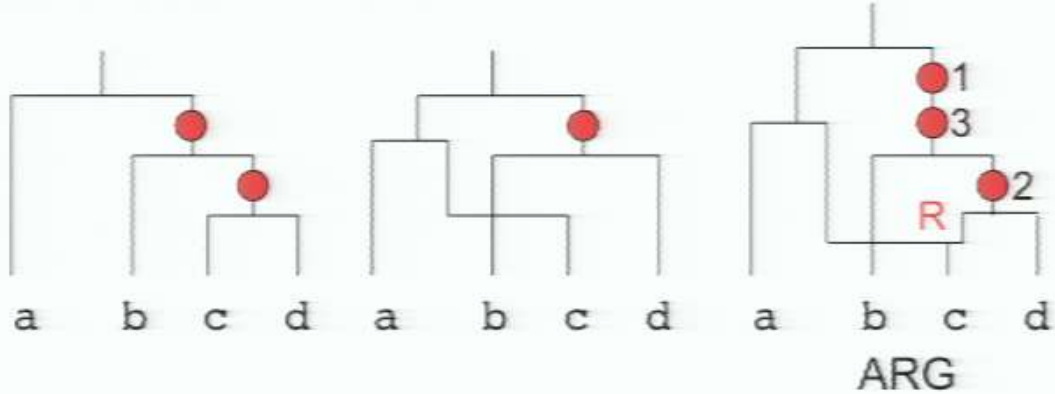
- There is a natural generative model for ARGs
  - Coalescent with recombination (Kingman, Tavare, Griffiths)
  - "Prune and graft" Markovian approximation is very good
- But infinitely many ARGs satisfy a data set
  - Sampling histories is hard, as is conditional sampling



The tree changes along the genome

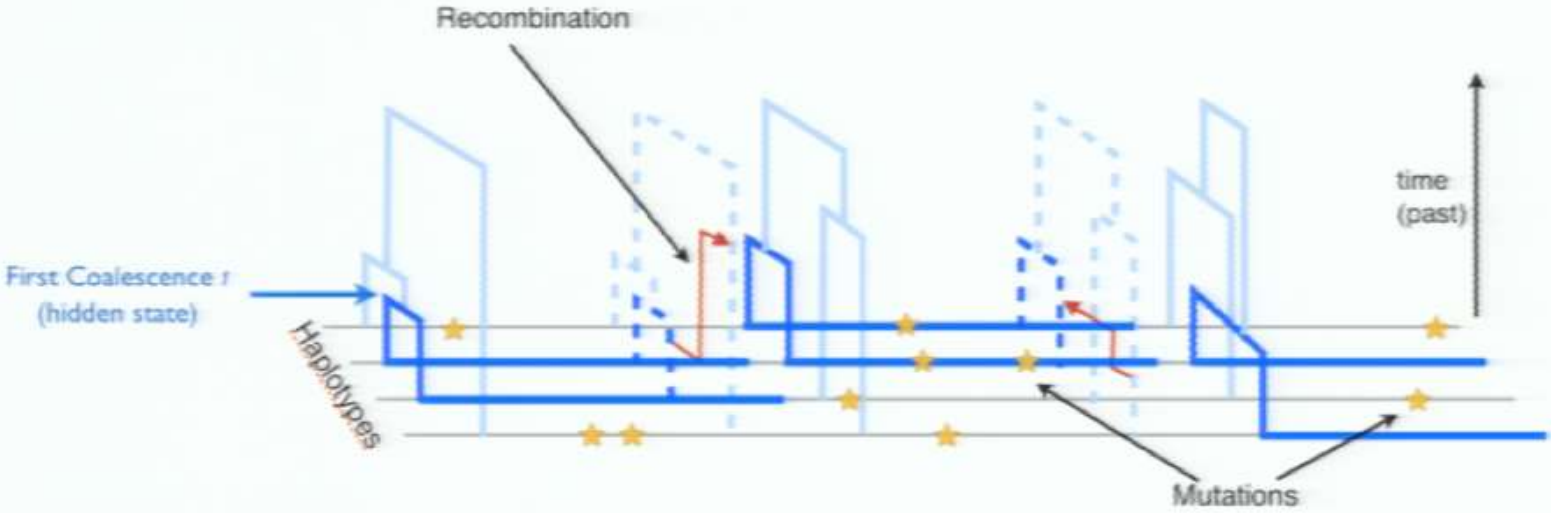
- At each locus there is a tree
- Ancestral recombinations change the tree as you move along the genome
- The resulting *Ancestral Recombination Graph* describes the way that individual sequences in a sample are related

```
a ..C..G..A..   0 0 0
b ..T..G..C..   1 0 1
c ..T..A..A..   1 1 0
d ..T..A..C..   1 1 1
```



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

```
refNCBI36   8881 CCTCCGCCTCCCAGGTTCAACTGATTCTTCTGCCTCAGCCTCCCAAGTAGCTGGGATTACAGGCGAGCGCCACCACACCTGGCTAATTTTTTGTATTTTTTTAGTAGAGACGGGGTTTTG   9000
Venter      8881 ........................................................................................................................   9000
Watson      8881 ........................................................................................................................   9000

refNCBI36   9001 CCACGTTGGCCAGGCTGGTCTTGAACTCCTGACCTCAGGTGATCTGCCCGCCTTGGCCTCCCAAAGTGCTGGGATTACAGGTGTGAGCTACCGCGCTTAGCCCAGTGATAGAGTTTTGT   9120
Venter      9001 ........................................................................................................................   9120
Watson      9001 ........................................................................................................................   9120

refNCBI36   9121 TGCCAAAACAAAACATATGAACATATGATAGCTCTAATAAAAAATGCTGTTTCTTTGTTCTCATAATTTCAGTAGCTGAACTATGCTCCATTTCATCTGTAAAAGAGAAATAATCTGTAC   9240
Venter      9121 ........................................................................................................................   9240
Watson      9121 ........................................................................................................................   9240

refNCBI36   9241 CTTCTTGAGTG          AAGGAAATGAAAGCAAAATAACTATTTATAATTAGTGAAGAGAAAGCTTAGGAAAAAATGAGAAGATGCTTTAGAAATGTTAAAATAGCTTTAATAATA   9360
Venter      9241 ...........          ............................................................................................   9360
Watson      9241 ...........          ............................................................................................   9360

refNCBI36   9361 ATACTTTATTATTTTAAAGAATGTAGGCAGTTGTGATTTTTTCCCCCAACATTTCATCAGAGAGCAGAAAAGGTGCTGCCAAATTGCAGACTTCCACATGAAAGATTTCACTATGCCAGG   9480
Venter      9361 ........................................................................................................................   9480
Watson      9361 ..........................~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   9480

refNCBI36   9481 TGACCTGCACTAGAAGCAGGTTAATAACCCTGTTTTCATGGGCTGTCTTGCTTTCACAATGAATGGTCTCCTTTTGCAATGAATTTTTGAAGTTTTGTTTTTATTTATTCTATGTAATAA   9600
Venter      9481 ........................................................................................................................   9600
Watson      9481 ........................................................................................................................   9600

refNCBI36   9601 TTTGGCTACATGTAATTATTTAGCTGCCTTTAGAGTCATGGACTTGGAGAATTGAAGATTTTCTGGTCCAATATTTCTCTAAGTATGTTCGTCATCTGTACTTAAAGAAGGGGCTCTGGG   9720
Venter      9601 ........................................................................................................................   9720
Watson      9601 ........................................................................................................................   9720

refNCBI36   9721 ATCAATTAGGTCAGGAAACTGCTGAGTTAAAAAAGATTTTTAGTTTTGTGGGAGTTGGGGAGGTGAGAACATTCCTGAGACTTTAATATGCTAATATGAGTCTCTCACAGAGAAACAACA   9840
Venter      9721 ........................................................................................................................   9840
Watson      9721 ........................................................................................................................   9840

refNCBI36   9841 TACAGTGTTTTACAAACTTACTTGACCACAAAACCTTTTTCTCGTGGTCTTGTTGTTACTCAGAACTCAAATCTTTTGAATTTACAAATGAACAGAAACGACAGCAAATGGCTATTCAA   9960
Venter      9841 ........................................................................................................................   9960
Watson      9841 ........................................................................................................................   9960

refNCBI36   9961 AGCAATATAGCAGTTTATTTAAAGAACTGGACTTGAACTA                                                                              10000
Venter      9961 ........................................                                                                               10000
Watson      9961 ........................................                                                                               10000
```

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

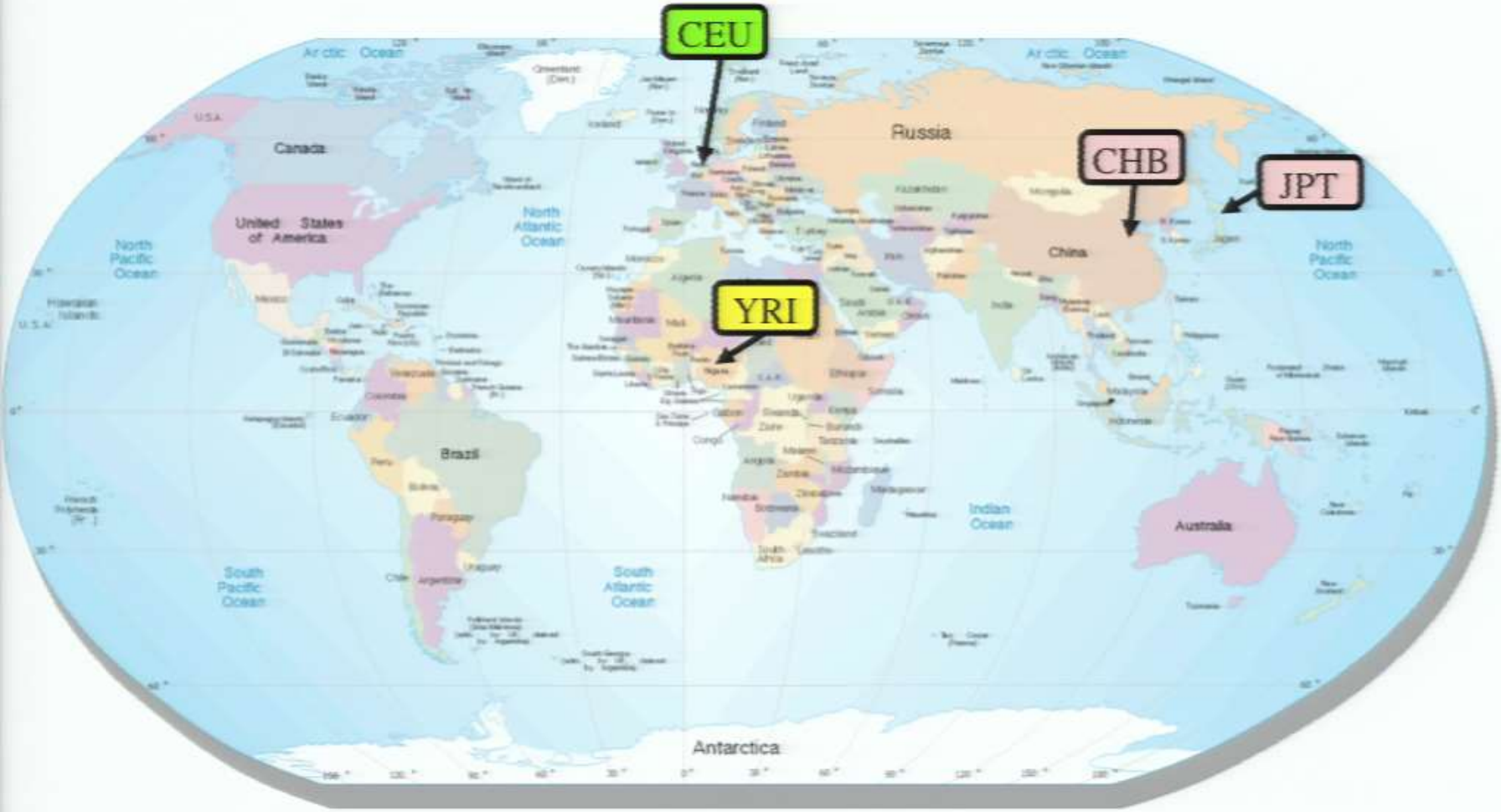
1000 Genomes

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
  - Conceived in 2007, pilot projects published Oct 2010, phase 1 with 1092 samples published November 2012, Completion 2014
- 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

1000 Genomes

Pilot project 179 samples (Nature 2010)

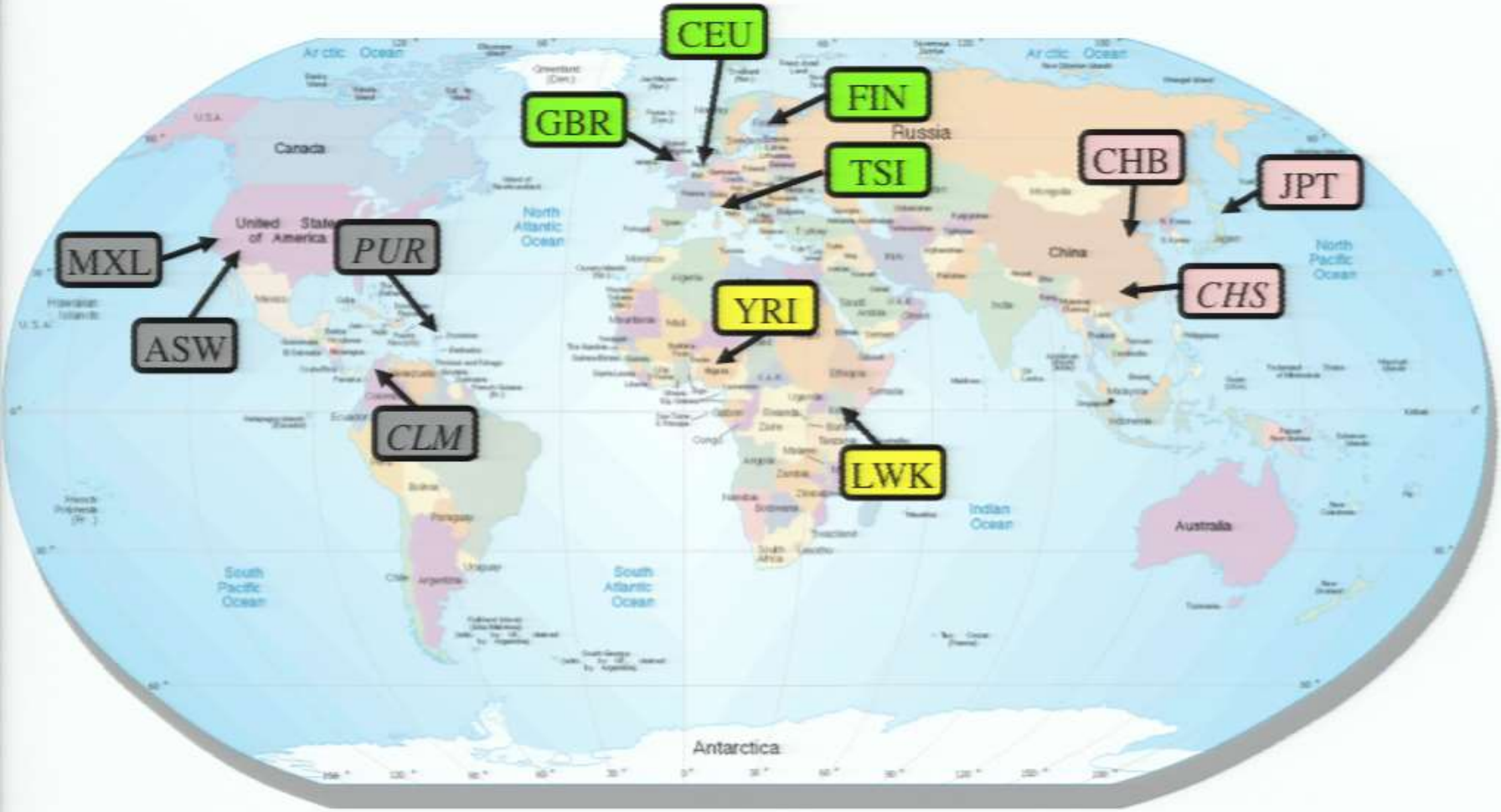


~100 per population: 4x Whole Genome Shotgun + Deep Exomes

1000 Genomes

Pilot project 179 samples (Nature 2010)

Phase 1: 1,092 samples (Nature 2012)



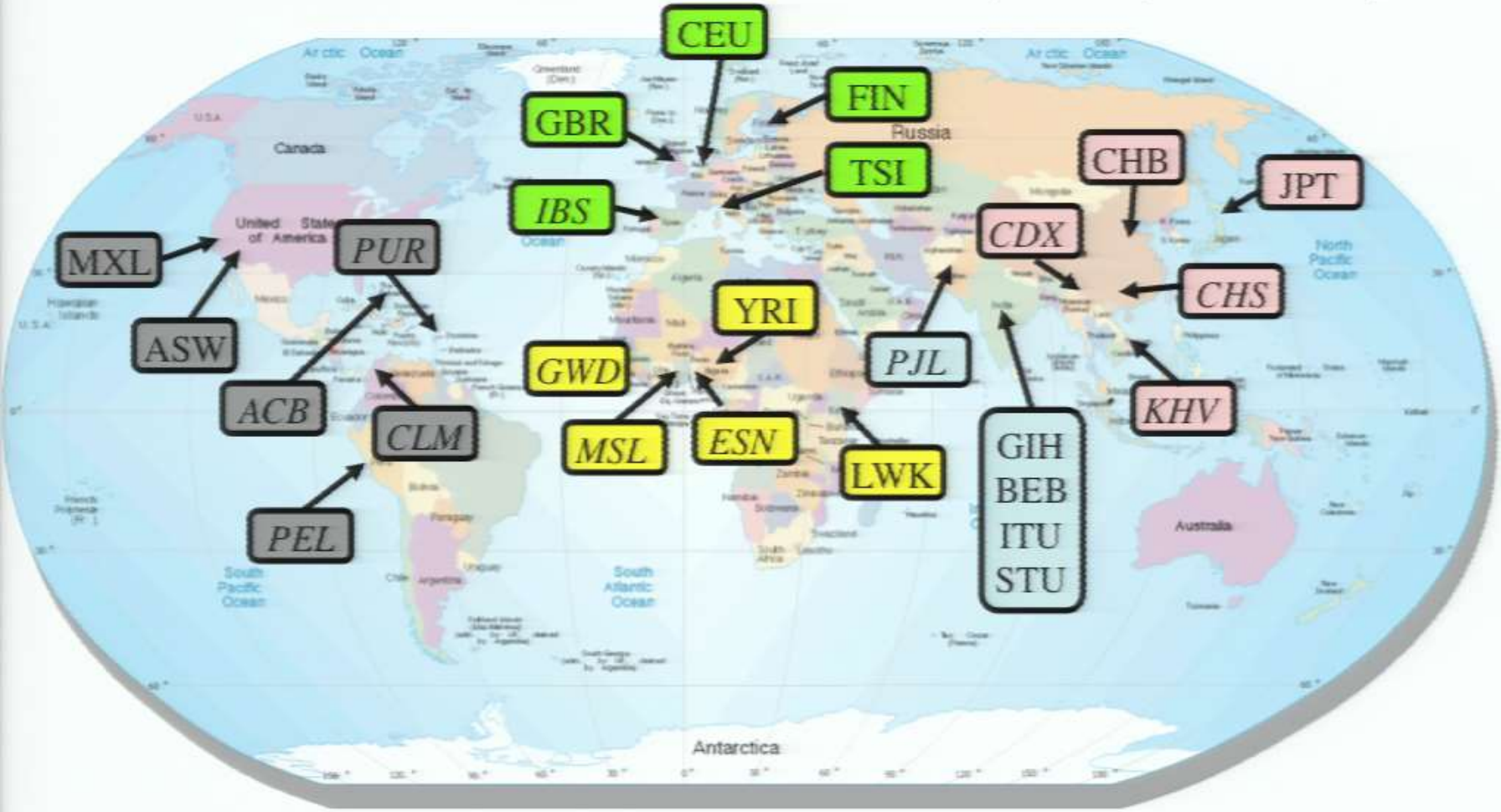
~100 per population: 4x Whole Genome Shotgun + Deep Exomes

1000 Genomes

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

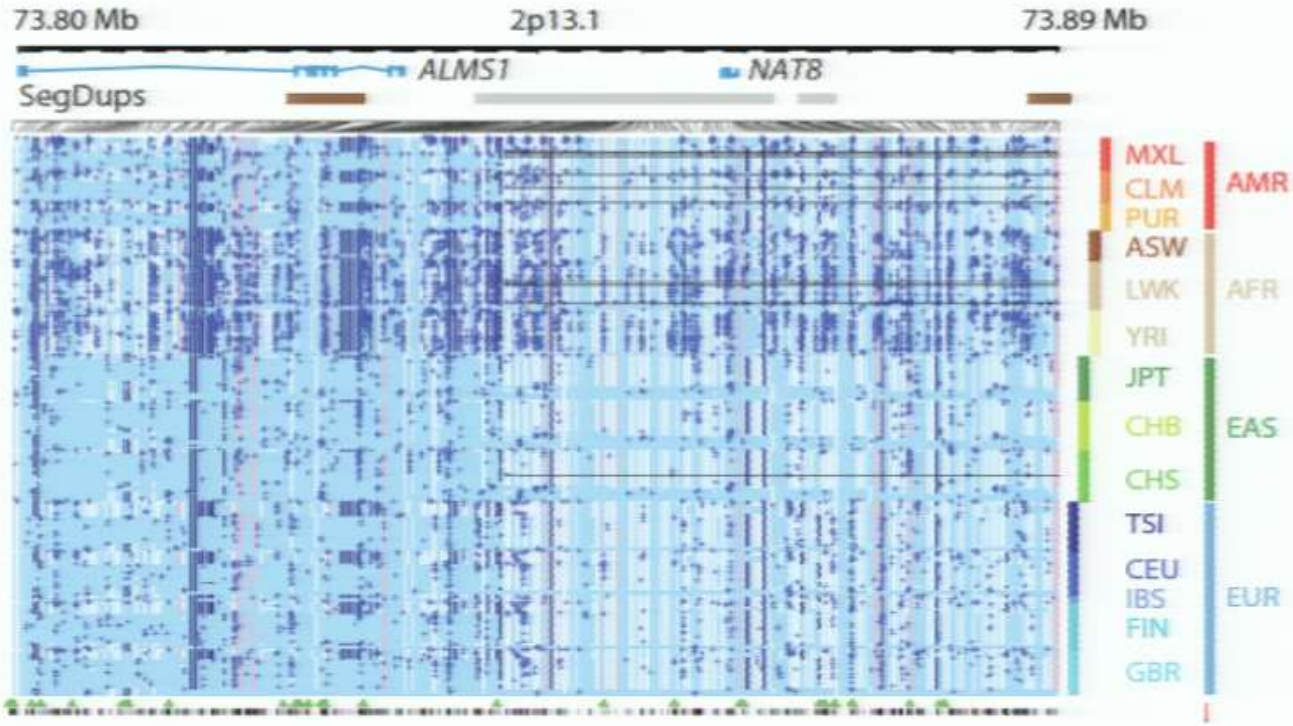
1000 Genomes

A wealth of data

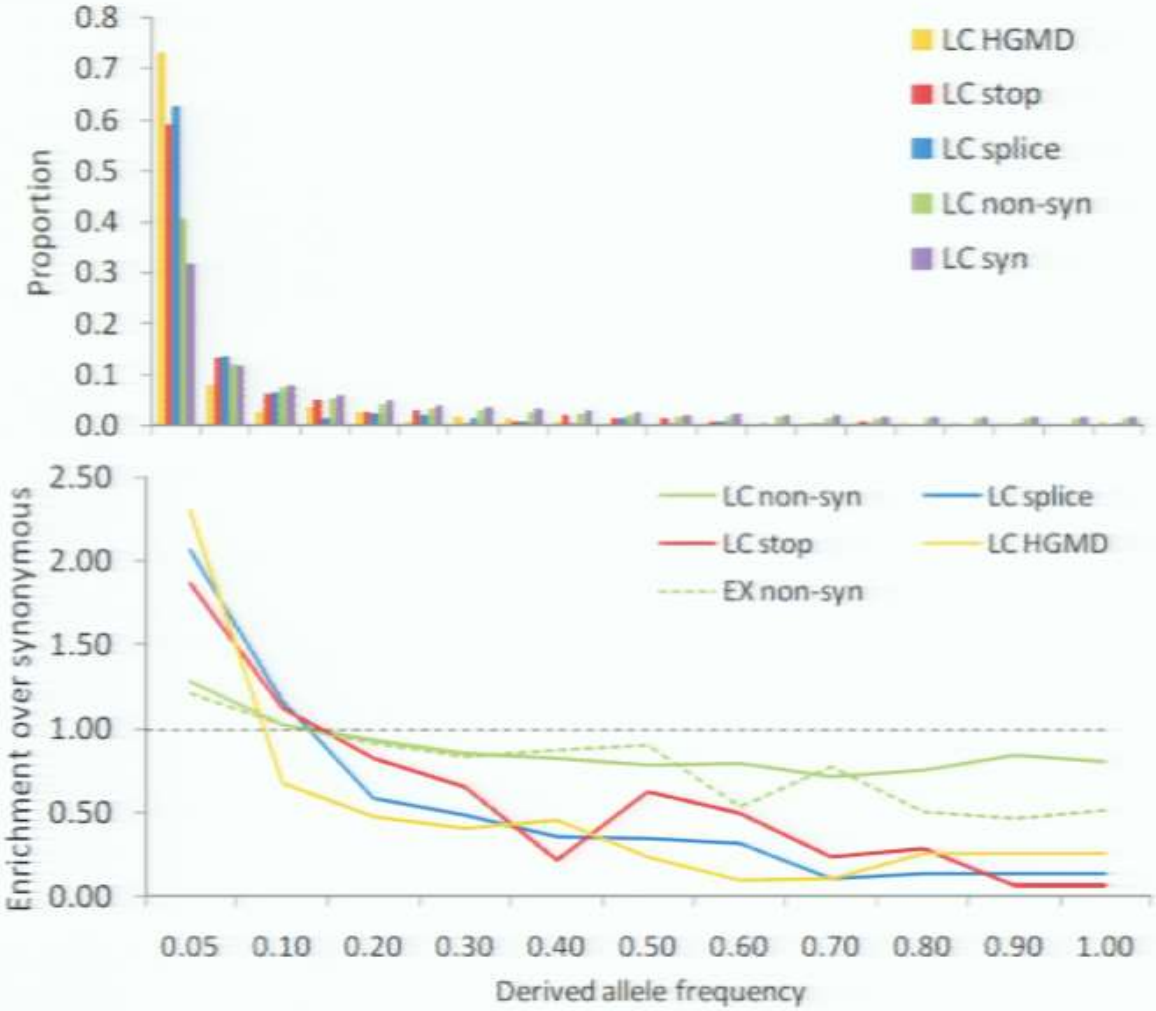
Call set	Samples	SNPs	Sensitivity (HapMap3.3)	FP (in 59,721 monomorphic OMNI sites)	Genotype accuracy (HapMap 3 hets)
Pilot	179	15.2M	97.65%	43,606	97.75%
Phase 1	1,092	38.0M	98.87%	1,261	99.24%

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

20.2 Tbp sequence



Distribution of functional variants



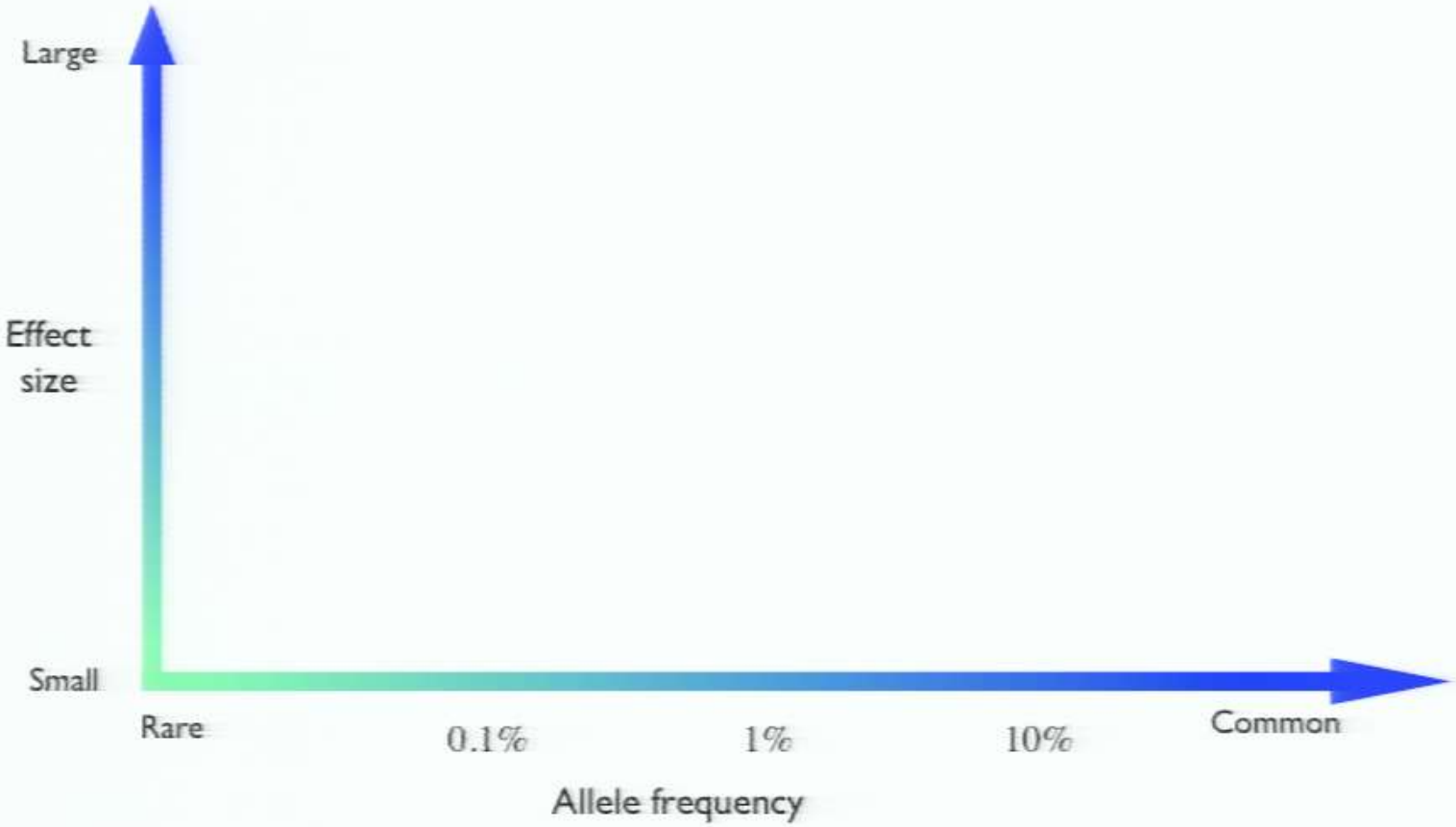
Common deleterious mutations are removed by selection

Pilot paper

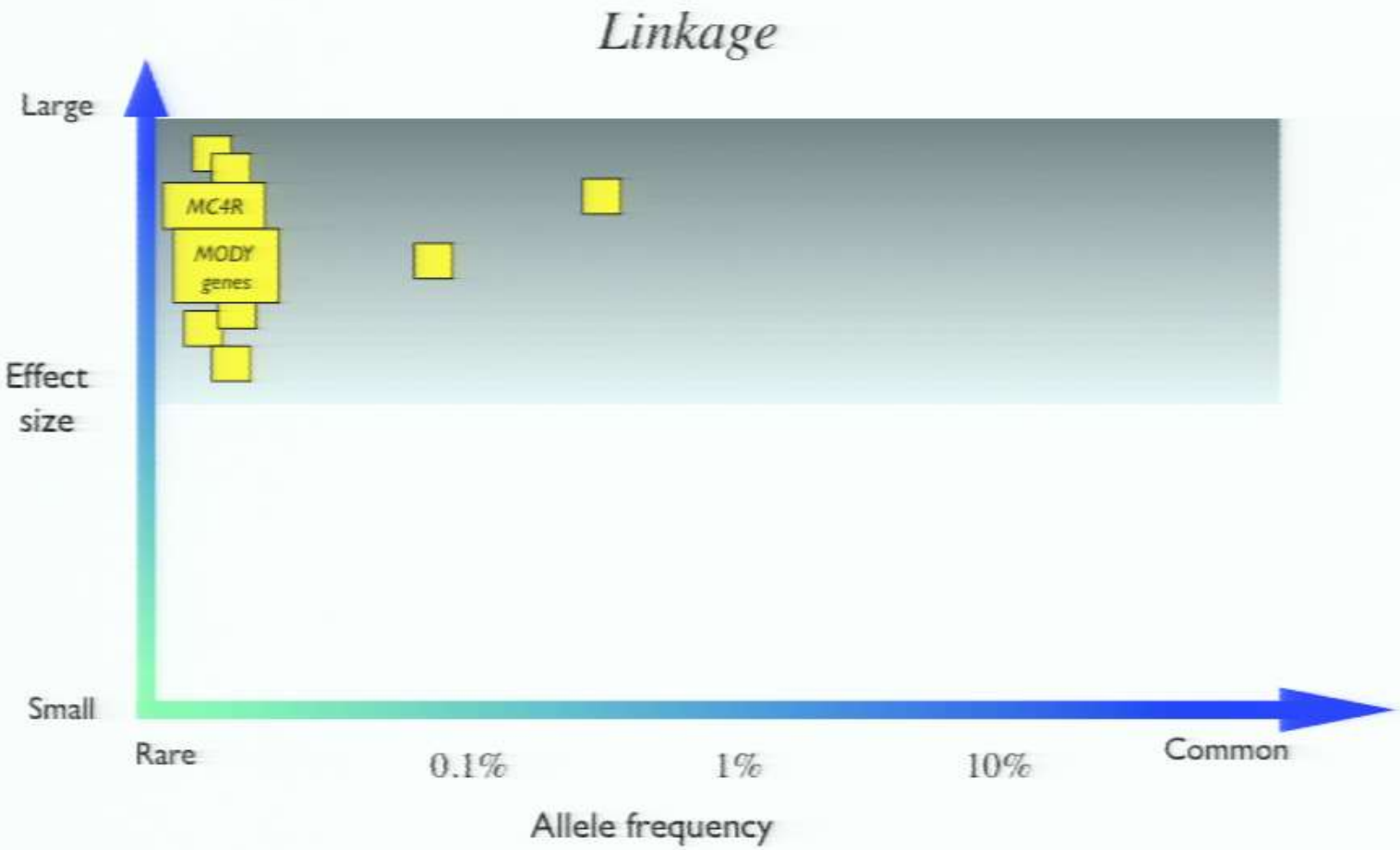
Per individual deleterious variant load

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

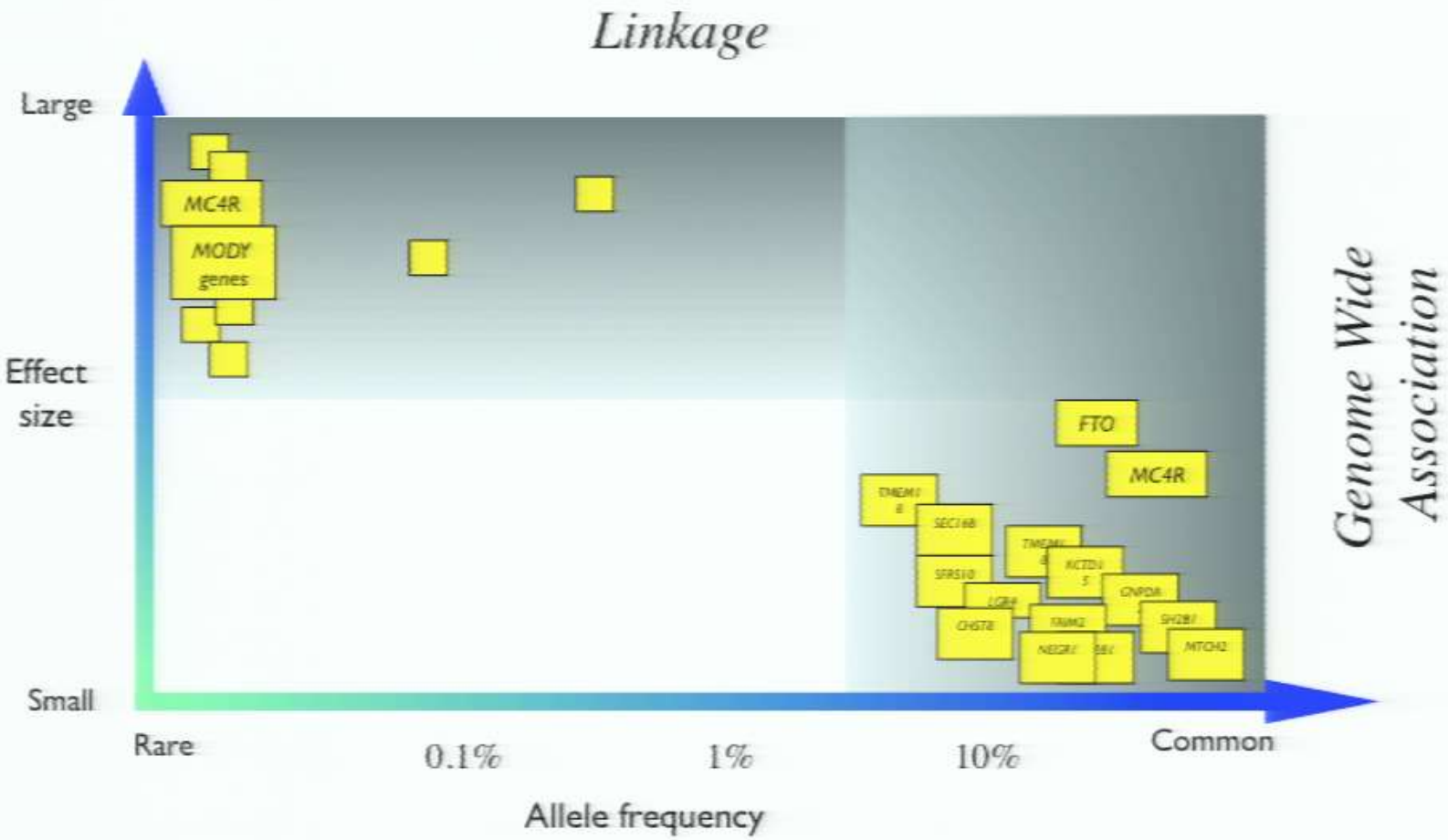
What about identifying functional genetic variants?



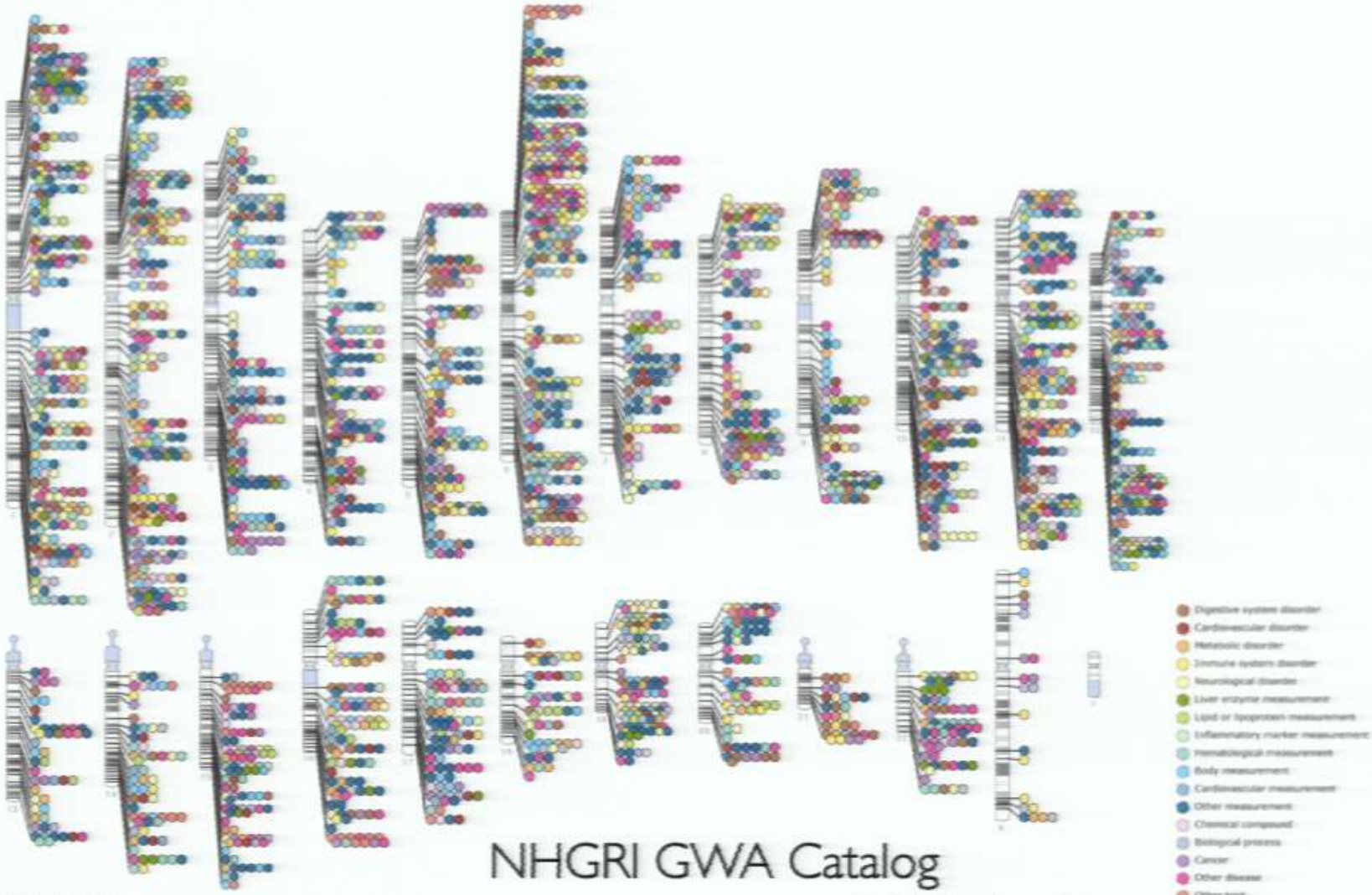
What about identifying functional genetic variants?



What about identifying functional genetic variants?



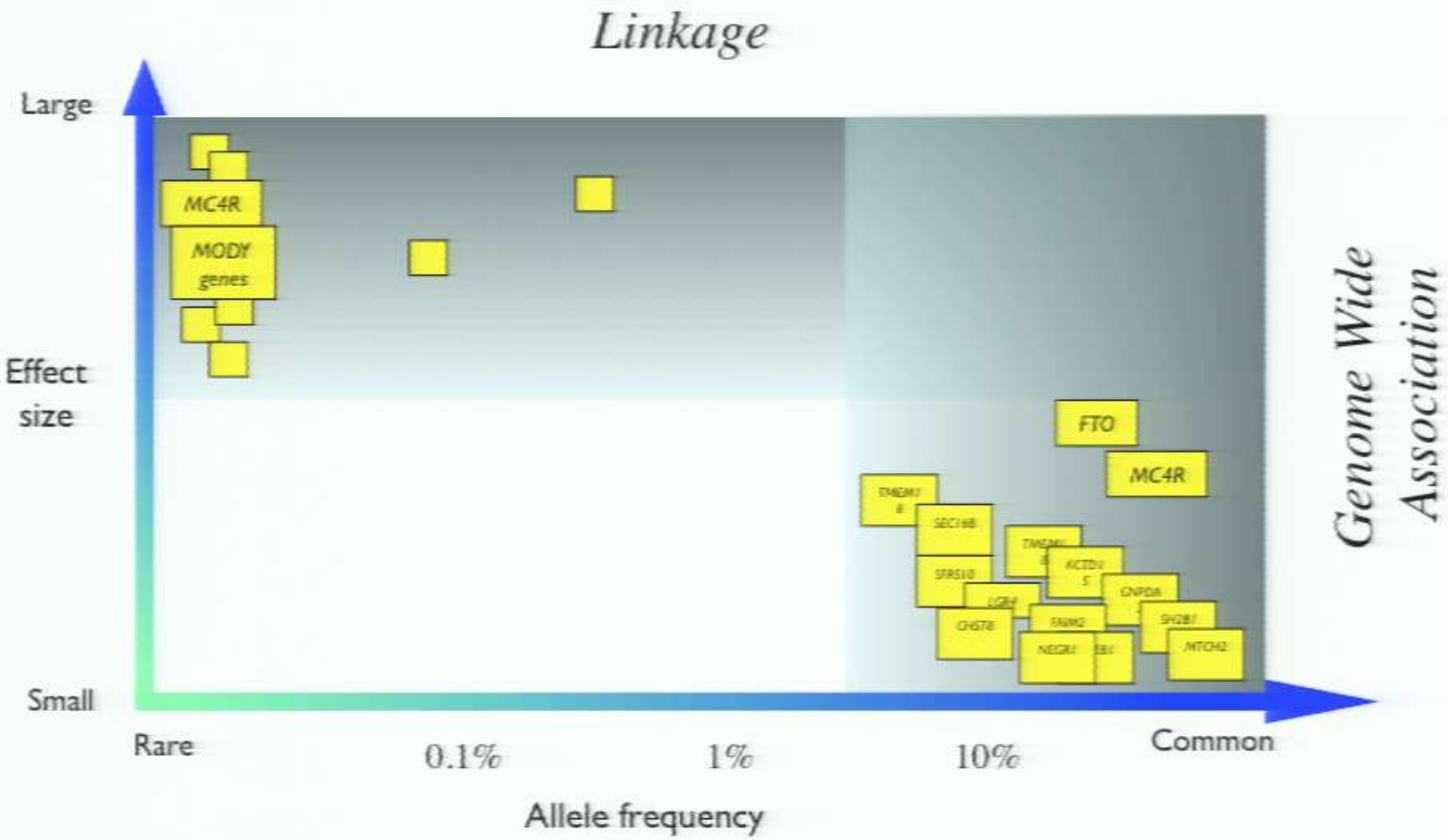
Over a million people genotyped
Published Genome-Wide Associations through 07/2012



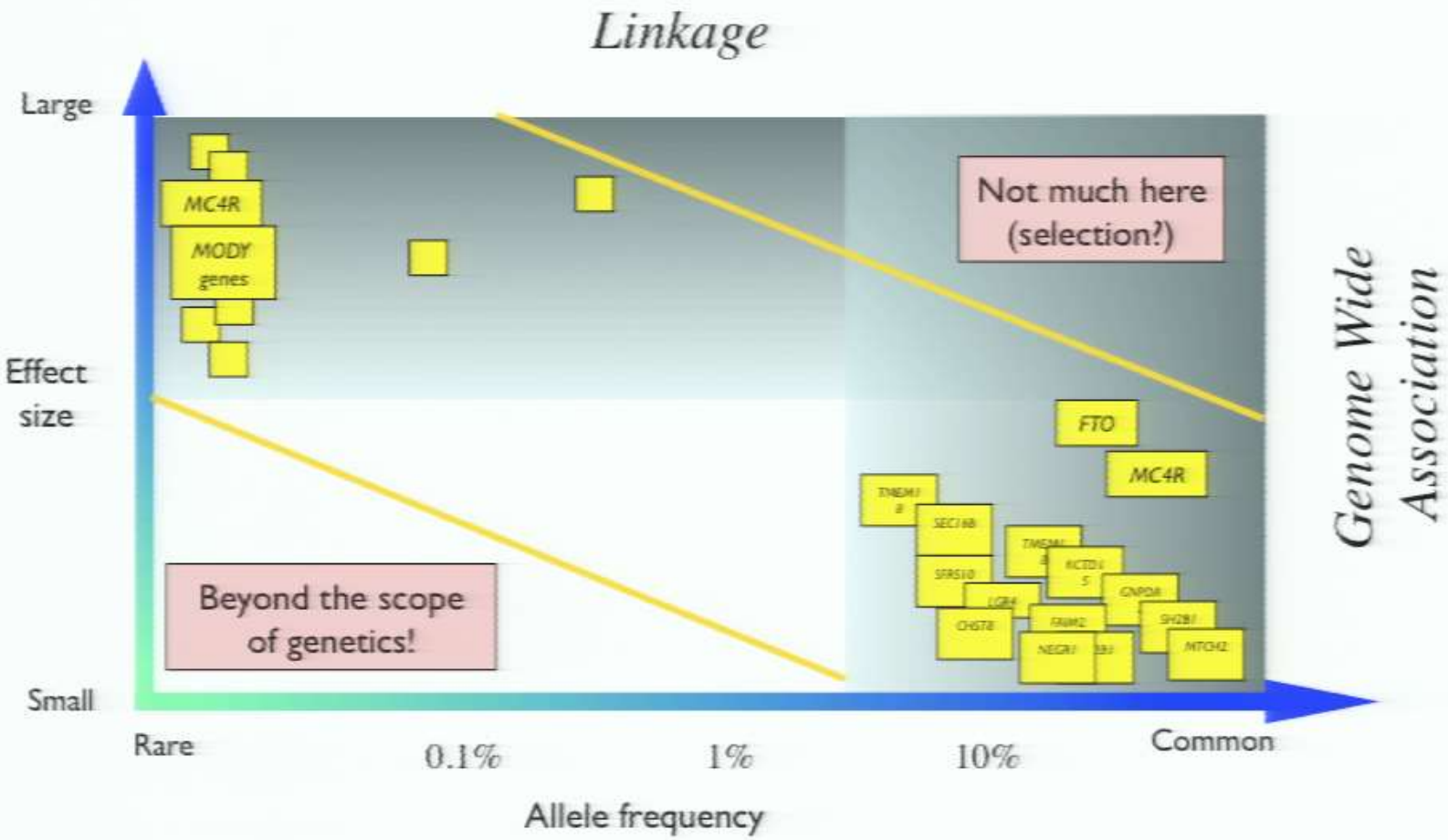
NHGRI GWA Catalog
www.genome.gov/GWAStudies

National Human Genome Research Institute

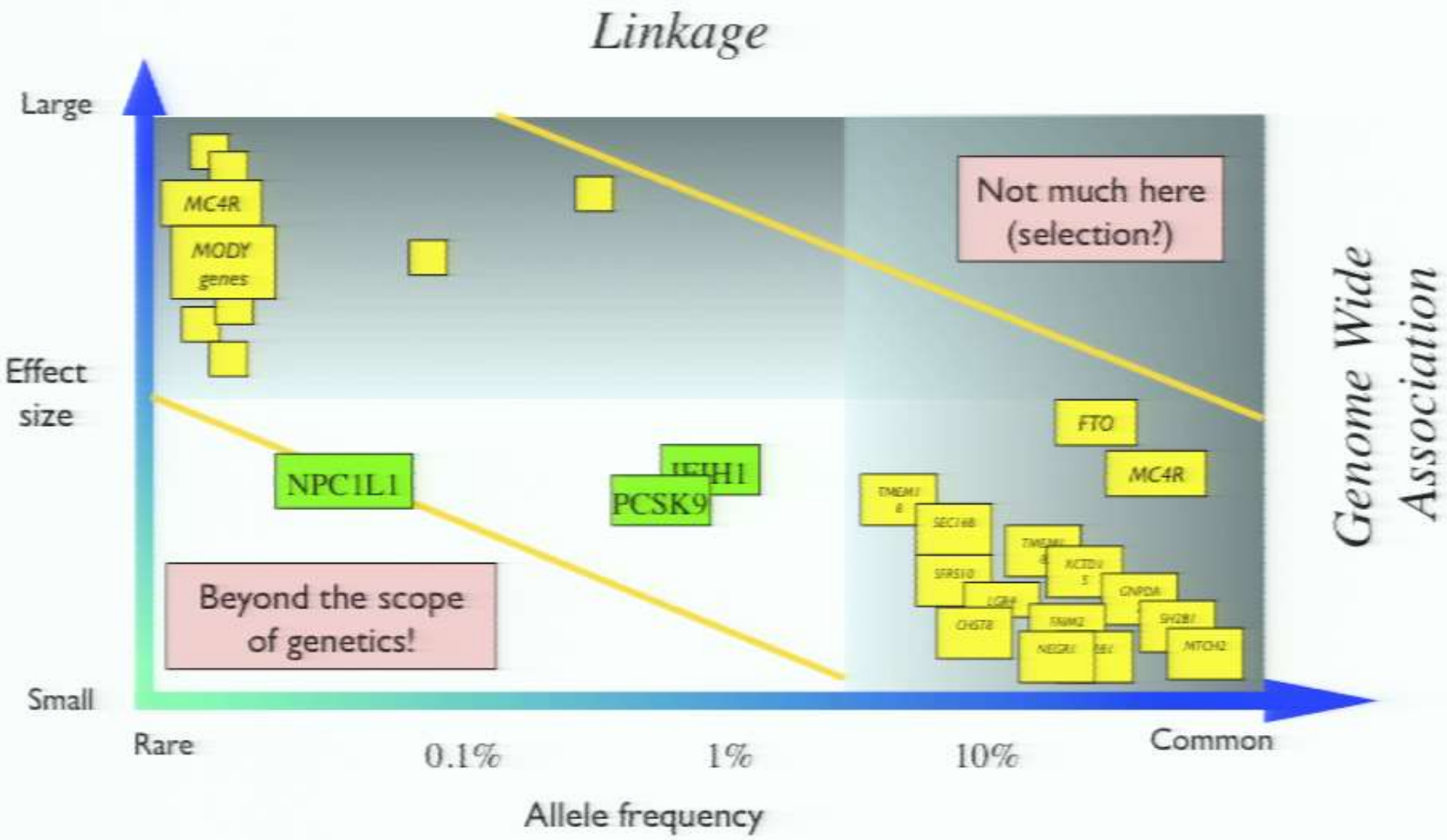
What about identifying functional genetic variants?



What about identifying functional genetic variants?



What about identifying functional genetic variants?



A few genes in the middle zone have been found by candidate studies

UK 10K

RARE GENETIC VARIANTS IN HEALTH AND DISEASE

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

UK 10K

Rare Genetic Variants in Health and Disease

Cohorts initial analysis: 54 traits in 14 groups on 2,453 samples

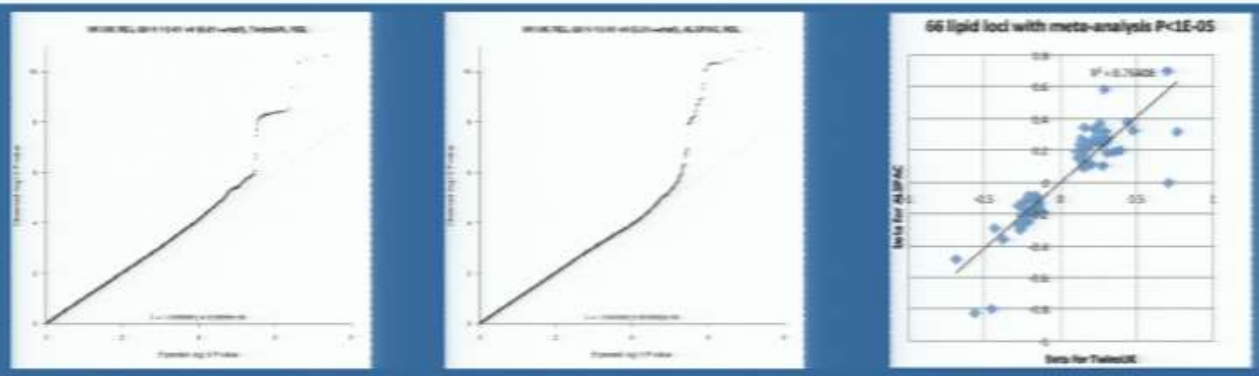


Fig 3: QQplots and effect size comparison of singlepoint analysis of HDL (MAF>1%)

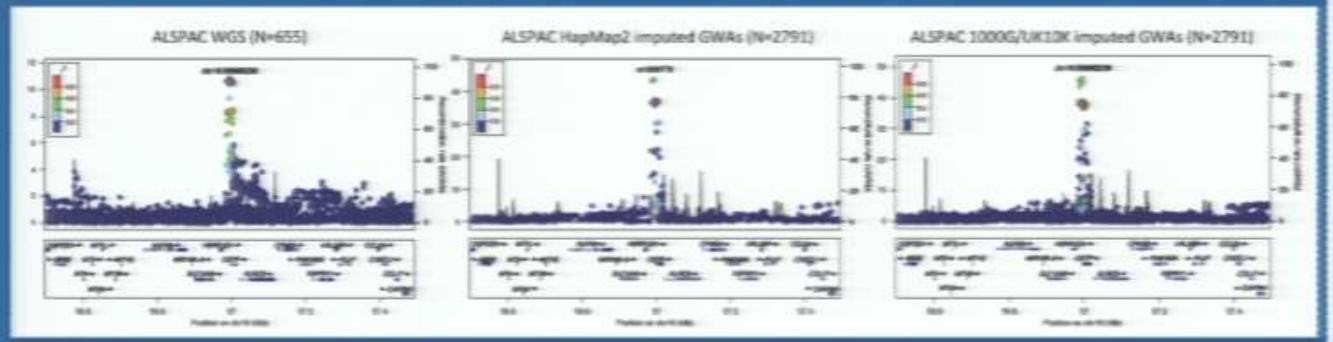


Fig 4: Association between *CETP* SNPS and HDL for ALSPAC WGS (N=655), ALSPAC HapMap2 imputed GWAs (N=2,791) and ALSPAC 1000G/UK10K imputed GWAs (N=2,791).

Some potential candidate new loci < 2.5% or < 10% poorly tagged

Alternative strategy: Impute full sequences into GWAS studies from reference sequences

```
a g a g t t g a g g g a a c c t g a g a a
t g a g a c g a g g g a a a t t g a g a c
t g c g a c g g t g a t t c t c c a g a c
a g c g a c g a t g g t a c t t g a t c a
t a a g t t a g t a a t t c c c g a g c a
t g c a a t g a g g g a a a t t g t t a a
a g a g a c g g g g g a a a t t c t g c c
```

Reference haplotypes via sequencing studies

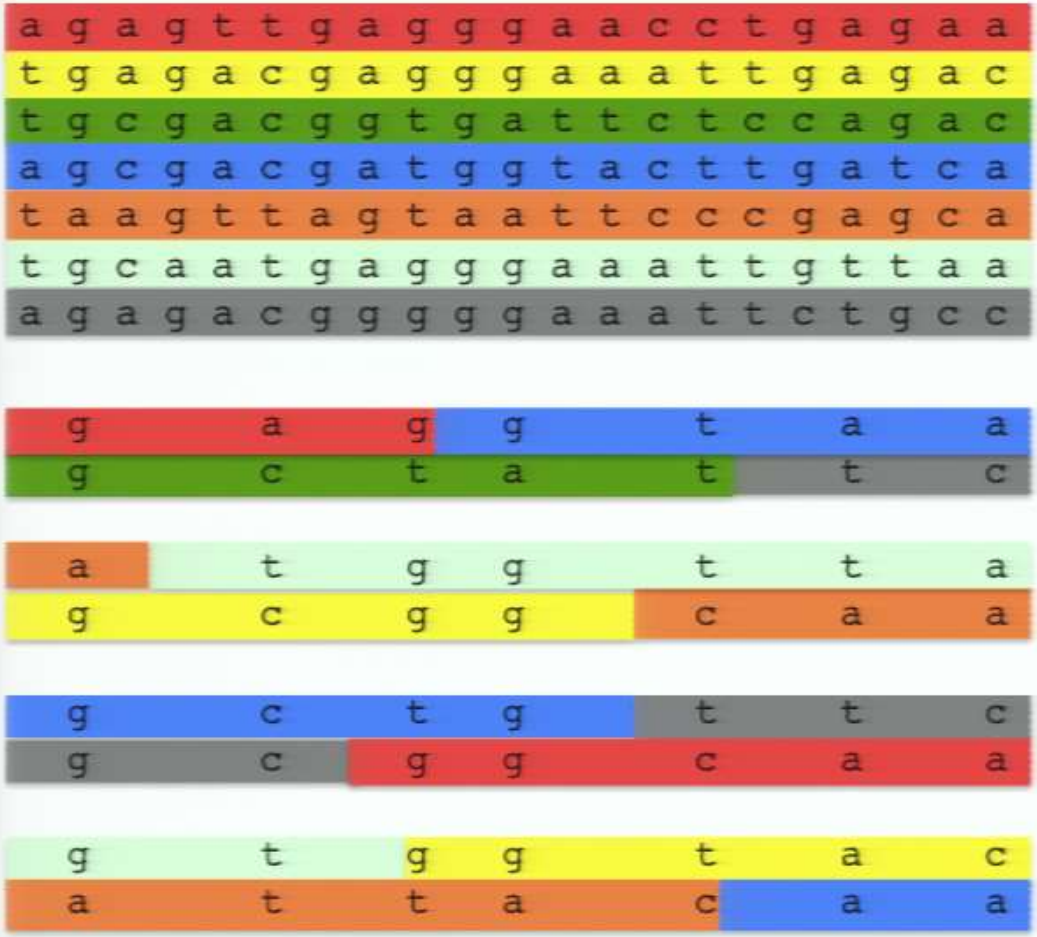
e.g. 1000 GP, UK10K

```
g       a     g   g       t     a     a
g       c     t   a       t     t     c

a       t     g   g       t     t     a
g       c     g   g       c     a     a

g       c     t   g       t     t     c
g       c     g   g       c     a     a

g       t     g   g       t     a     c
a       t     t   a       c     a     a
```

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

```
agagttgagggaacctgagaa
tgagacgagggaaattgagac
tgcgacggtgattctccagac
agcgacgatggtacttgatca
taagttagtaattcccgagca
tgcaatgagggaaattgttaa
agagacggggaaattctgcc
```

Reference haplotypes via sequencing studies

eg. 1000 Genomes Project

```
agagtagaggtacttgatca
tgcgacggtgattcttctgcc

taaaatgagggaaattgttaa
tgagacgagggaacccgagca

agcgacgatggtaattctgcc
agagacgagggaacctgagaa

tgcaatgagggaaattgagac
taagttagtaattcctgatca
```

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

Effectively an approximation to conditional sampling on the ARG

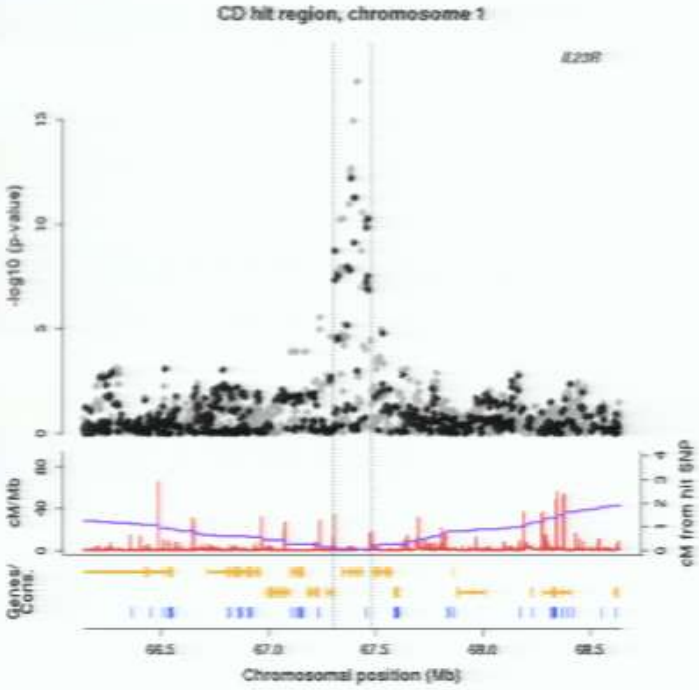
Imputation of unobserved alleles via matching of shared haplotypes

Imputing missing data from reference sequences

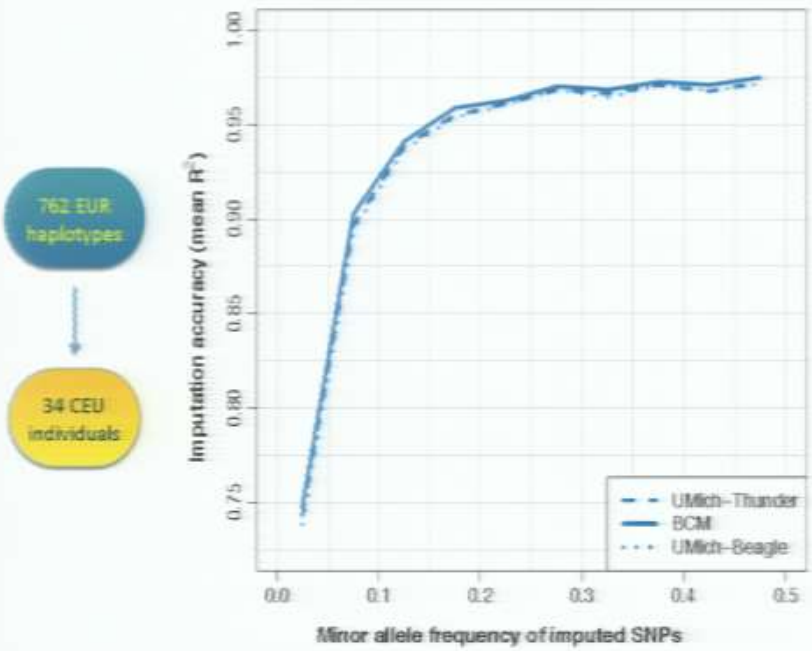
```
agagttgagggaacctgagaa
tgagacgagggaaattgagac
tgcgacggtgattctccagac
agcgacgatggtacttgatca
taagttagtaattcccgagca
tgcaatgagggaaattgttaa
agagacggggaaattctgcc

agagtagaggtacttgatca
tgcgacggtgattcttctgcc

taaaatgagggaaattgttaa
tgagacgagggaaacccgagca

agcgacgatggtaattctgcc
agagacgagggaacctgagaa

tgcaatgagggaaattgagac
taagttagtaattcctgatca
```

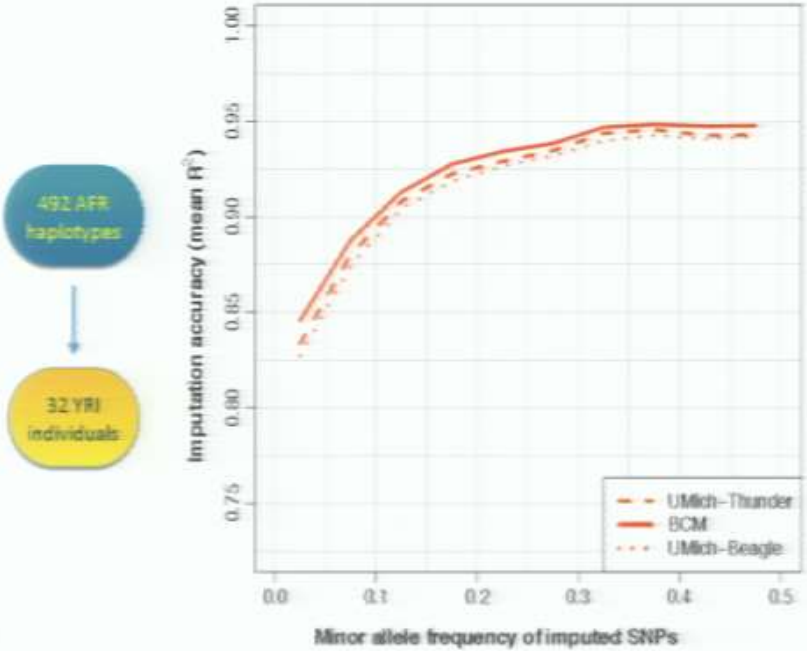
GWAS of imputed genotypes

- **Increased power**
- **Better resolution**
- **Facilitates meta-analysis**



1000 Genomes Project Imputation Accuracy





Improvements from pilot to phase 1 using GAIN psoriasis dataset

Reference	MAF 1-3%	MAF 3-5%	MAF >5%
1000G Pilot	.69	.77	.91
1000G Phase 1	.82	.85	.94

The Haplotype Consortium: Chromosome 20 pilot

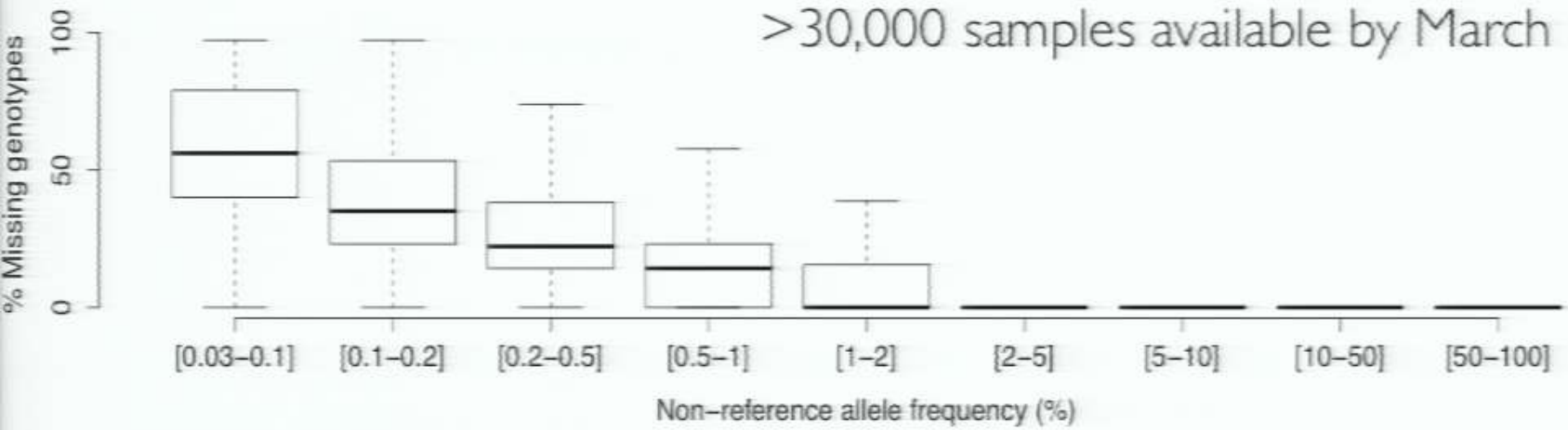
COHORT	#samples	coverage	#SNPs
1000GP	379	4	338,766
UK10K	3,781	6.5	1,004,955
ORCADES	399	4	259,373
FINNS	1,941	4	315,539
GoNL	748	12	434,984
GoT2D	2,874	4	570,847
AMD	630	4	355,438
SARDINIA	2,120	4	372,632
MTFS	687	6	468,672
TOTALS	13,559		1,649,648

We created a **union set of sites across 9 studies**

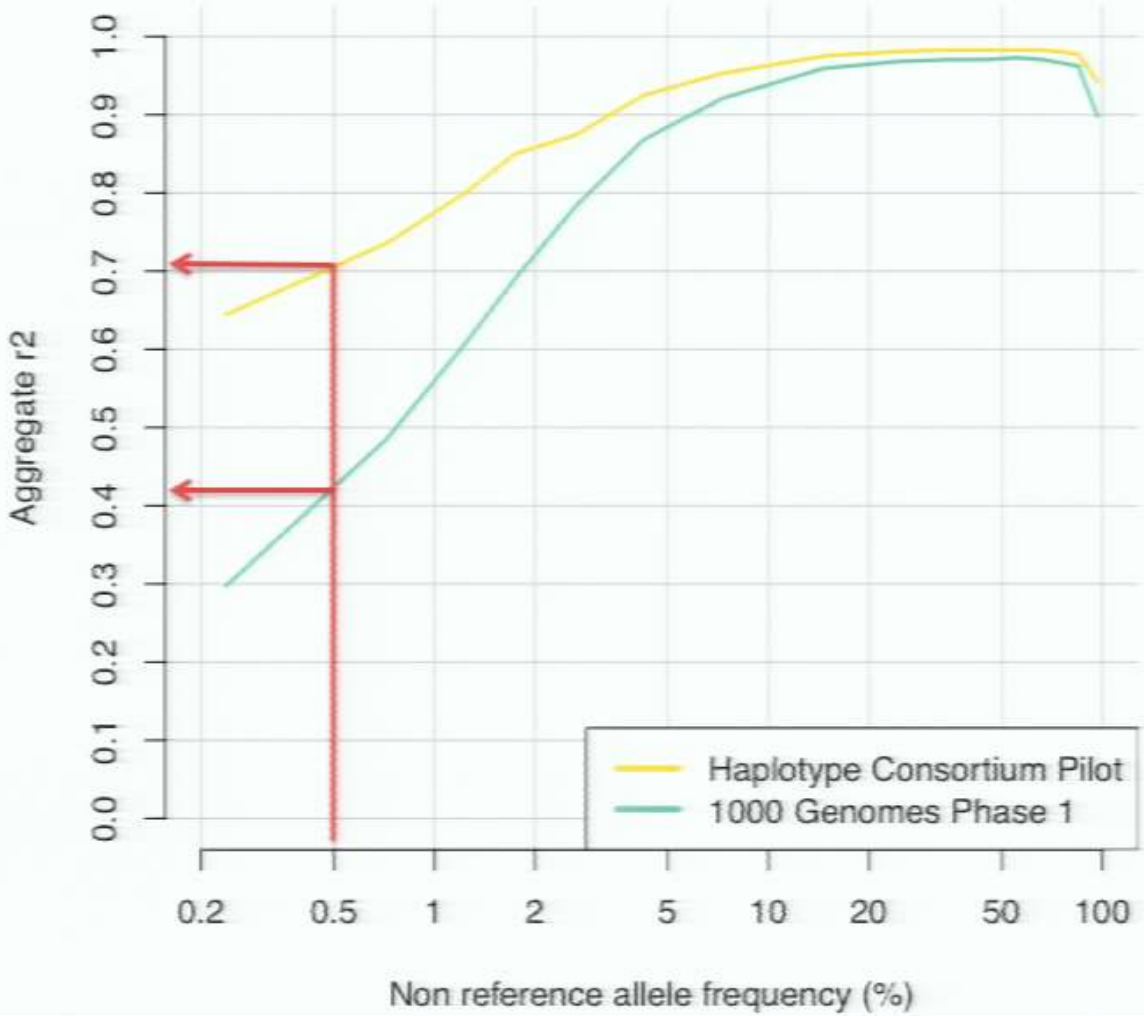
Only use sites with >10 minor alleles i.e. MAF > 0.03%

- 427,589 sites
- 26.5% missing genotypes:
- fill in by imputation

>30,000 samples available by March



Pilot Results: 13,559 sequences chr20



Haplotype Consortium

Jonathan Marchini and Olivier Delaneau, Oxford

1000 Genomes
Goncalo Abecasis
Gil McVean

Sardinia
Francesco Cucca
Serena Sanna
David Schlessinger
Carlo Sidore

Finns
Aarno Palotie

GoNL
Cisca Wijmenga
Paul I.W. de Bakker
Morris A. Swertz
Androniki Menelaou

UK10K
Nicole Soranzo
Nick Timpson
George Davey-Smith
Tim Spector

Orcades
Jim Wilson

AMD
Anand Swaroop
Dwight Stambolian Emily Chew
Xiaowei Zhan

HUNT
Cristen Willer
Kristian Hveem

GoT2D
Mark McCarthy
David Altshuler
Mike Boeknhe

MTFS
Scott Vrieze
Matt McGue
Bill Iacono

Bipolar
Mike Boehnke
Richard Myers

Helic
Eleftheria Zeggini

Crohns/UC
Jeff Barrett
Carl Anderson

Italian Isolates
Paolo Gasparini
Nicole Soranzo
Daniela Toniolo
Nicola Piratsu

Oxford
Warren Kretzschmar

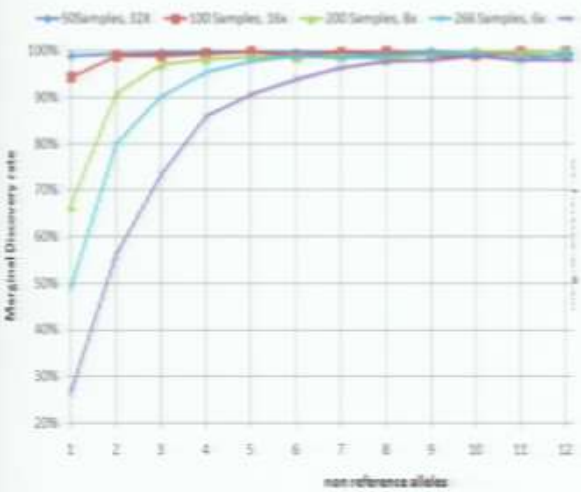
WT Sanger Institute
Shane McCarthy

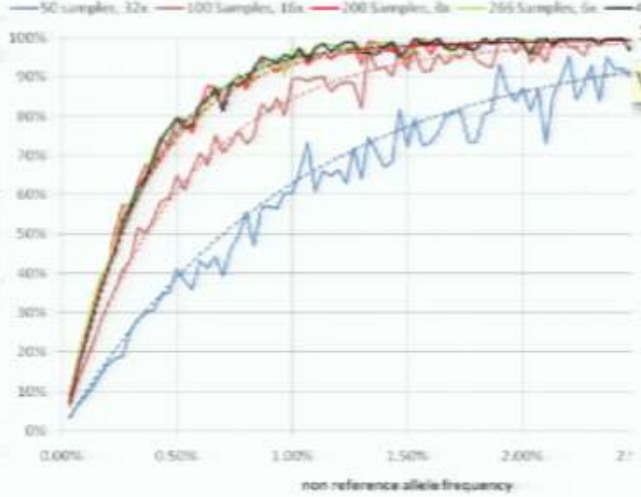
Michigan
Christian Fuchsberger
Hyun Min Kang

Imputation is also used to integrate data from low coverage sequencing

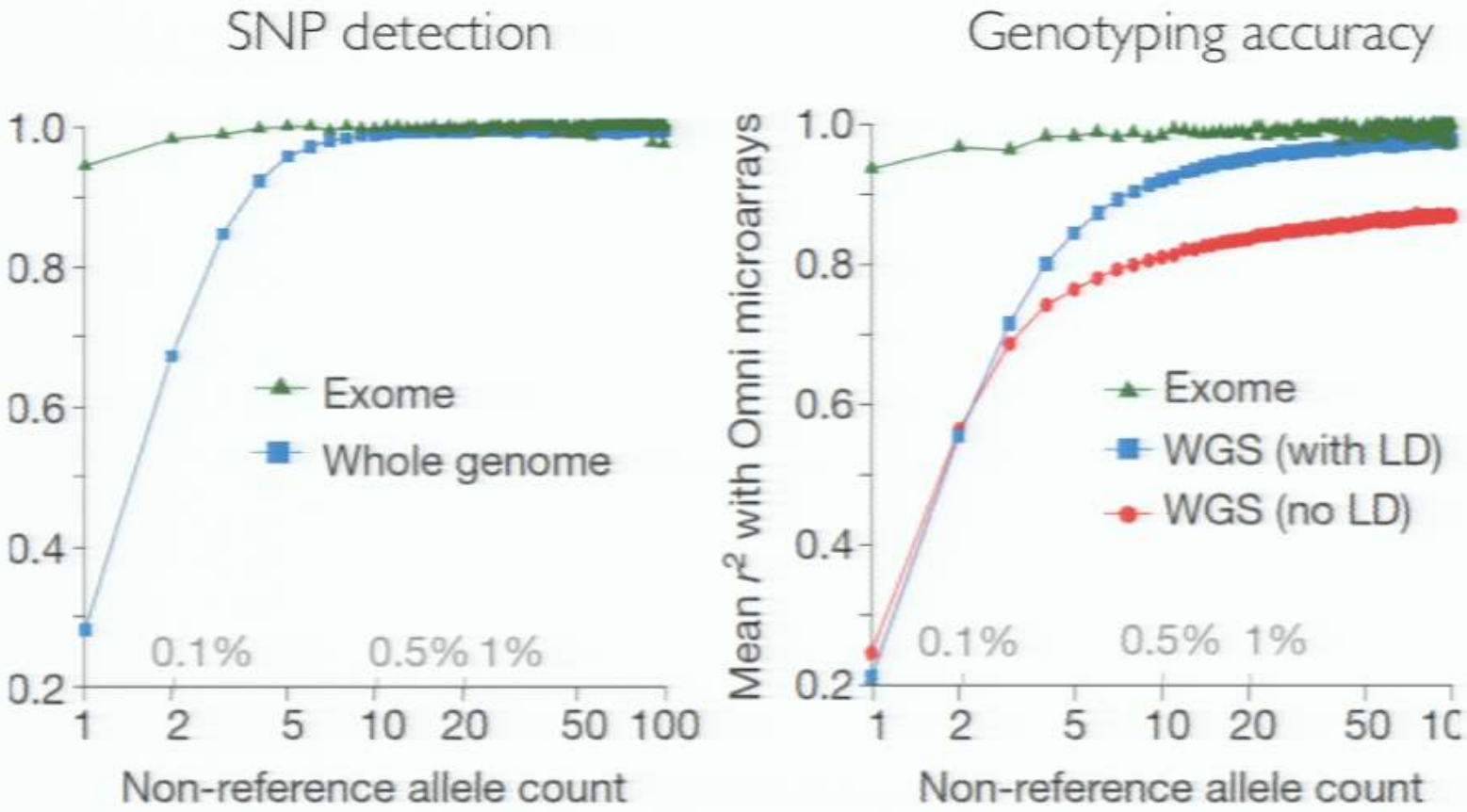
- To find the sequence of a single sample stand alone one needs to sequence at ~30x depth or more
- To find low frequency variants we want to sequence many samples
- Spread sequence across more samples

Fraction of SNPs discovered in 2453 UK10K samples found in 1000GP Phase 1





Phase 1 power and genotyping accuracy



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)

PBW[*k*]

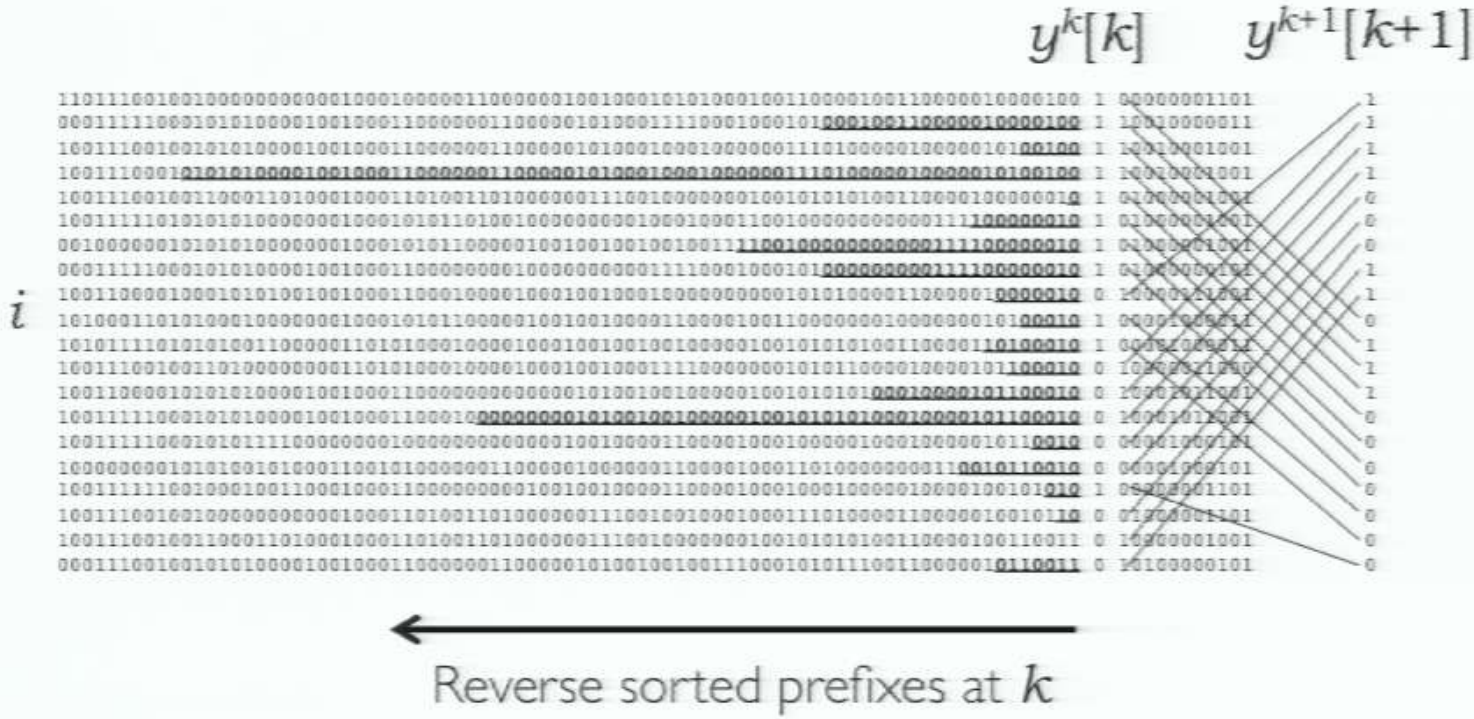
```
0100101010000001110111001001000000000001000100000110000000100100010101000100110000100110000010000100 1 0000000110110010000
0100100101101101100111110001010100001001000110000001100000101000111100010001010000100100001100001110 1 1001000001100001111 0
0110101100010000100111001001010100001001000110000001100000101000100010000001110100000100000101001001 1 1001000100100100010
0100100100010000100111000101010100001001000110000001100000101000100010000001110100000100000101001001 1 1001000100100100010
0110010100010000100111001001100011010001000110100110100000011100100000001001010101001100001000000010 1 0100000100100010010
0100010101010000100111110101010100000001000101011010010000000001000100011001000000000000111100000010 1 0100000100100010000
0110010100010011001000000101010100000001000101011000001001001001001001111100100000000000001111000000010 1 0100000100100010010
1100010100010011000111110001010100001001000110000000010000000000011110001000101000000000001111000000010 1 0100000010101000010
0100101100010000100110000100010101001001000110001000010001001000100000000001010100001100000100000010 0 1000011100100010000
0110010101110000101000110101000100000001000101011000001001001000011000010011000000010000000101000010 1 0000100001100010000
0110010101100000101011110101010011000001101010001000010001001001001000000100101010100110000110100010 1 0000100001100010010
0110010100010000100111001001101000000001101010001000010001001000111100000001010110000100000101100010 0 1000001100000100010
0100100100001100100110000101010100001001000110000000000000010100100100000100101010100010000101100010 0 1000101100100000010
0110010100010101100111110001010100001001000110001000000000101001001000001001010101000100000101100010 0 1000101100100000010
0100100101101101100111110001010111100000001000000000000001001000011000001000100000100100001011 0010 0 0000100010110000110
0100100101100000100000001010100101000110010100000011000001000000110000100011010000000011 0010110010 0 0000100010110000111
0000010100100011100111111001000100110001000110000000001001001000011000010001000100000100001001010 1 0000000110100010000
0100100101101101100111001001000000000001000110100110100000011100100100010001110100001100000100101 0 0100000110110010010
0100010100010000100111001001100011010001000110100110100000011100100000001001010101001100001001100 0 1000000100100000010
0100100110100011000111001001010100001001000110000001100000101001001001110001010111001100000101100 0 1010000010101000000
```

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 $y_i = 1$.

New data structure for fast matching Positional Burrows-Wheeler Transform (PBWT)

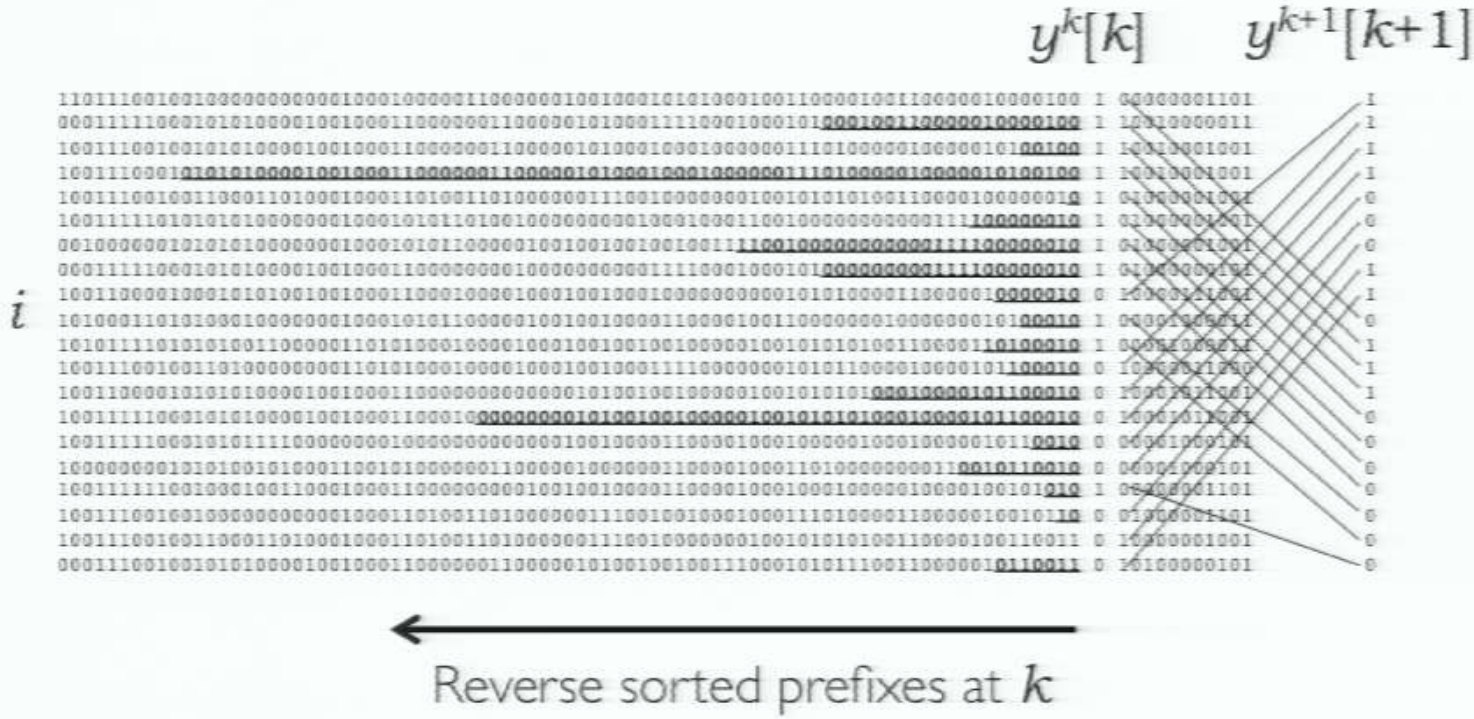
PBW[*k*]

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 $y_i = 1$.

New data structure for fast matching Positional Burrows-Wheeler Transform (PBWT)

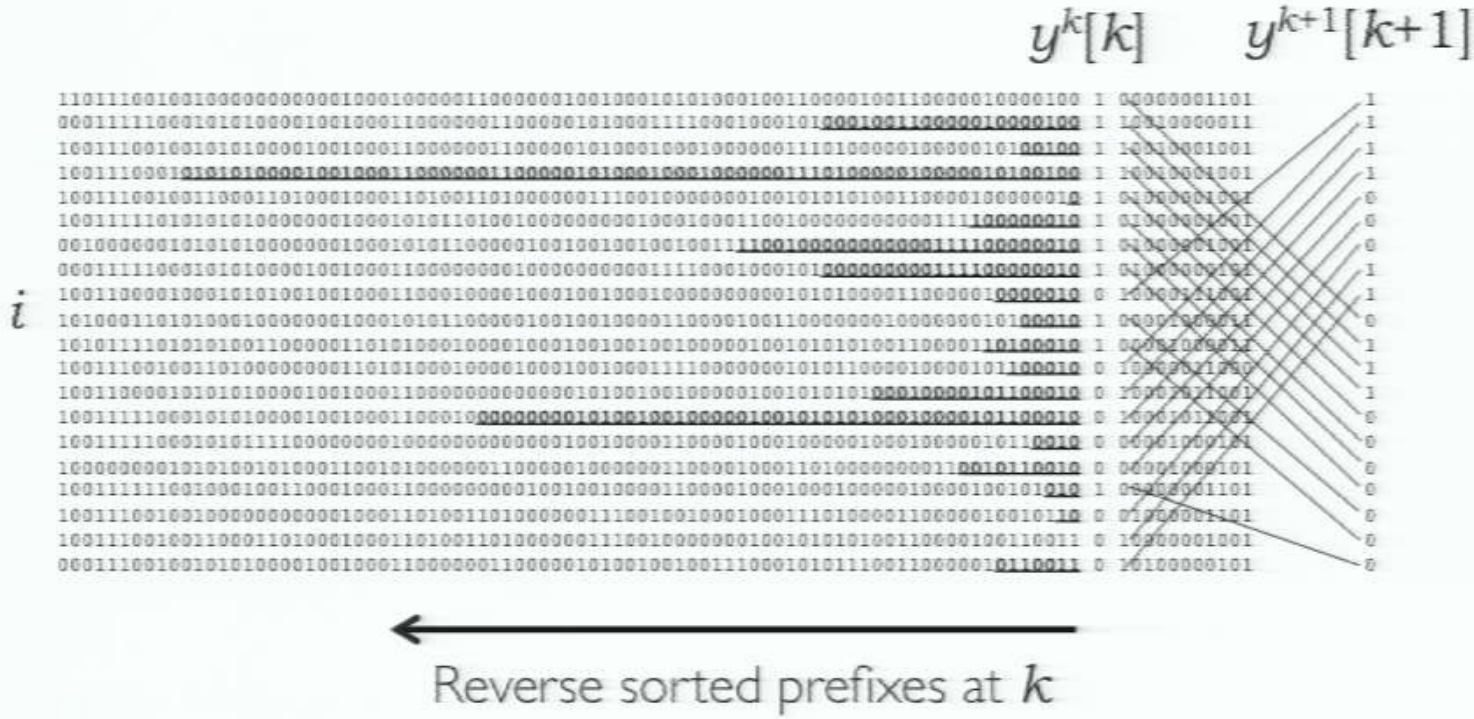
PBW[*k*]

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

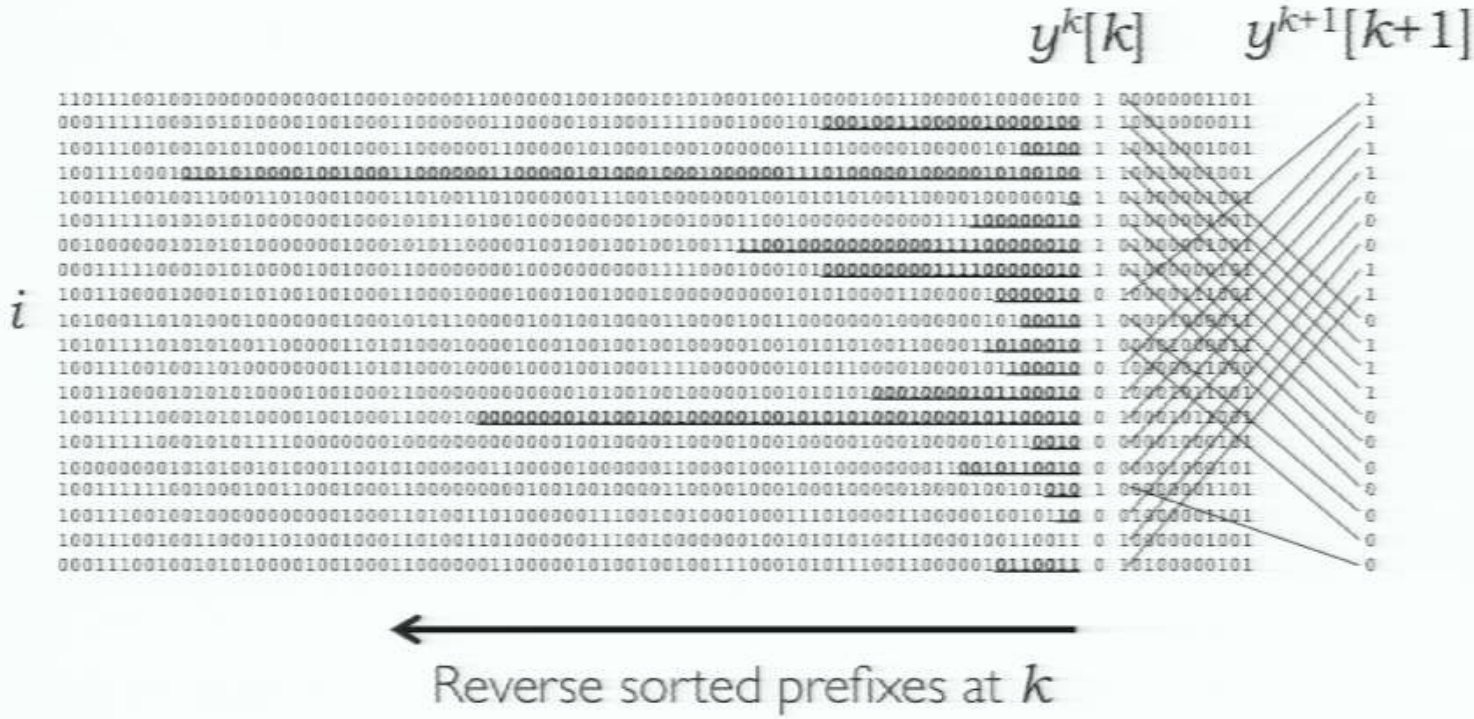


Let u_i be the number of 0s in y before i, i.e. $u_i = i - \sum_{j<i} y_j$
And c be the total number of 1s in y, i.e. $c = \sum y_j$
Then i maps to u_i if $y_i = 0$, and $c + i - u_i$ if $y_i = 1$.

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 compresssion)
    - 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 compresssion)
    - Native order run length encodes to ~2GB

Compression performance

Simulated data subsets

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

"Real" data: 1000 Genomes phase1 chromosome 1

2184 chromosomes, 3007196 sites

PBWT **51186641**

gzip **302883517**

factor **5.9**

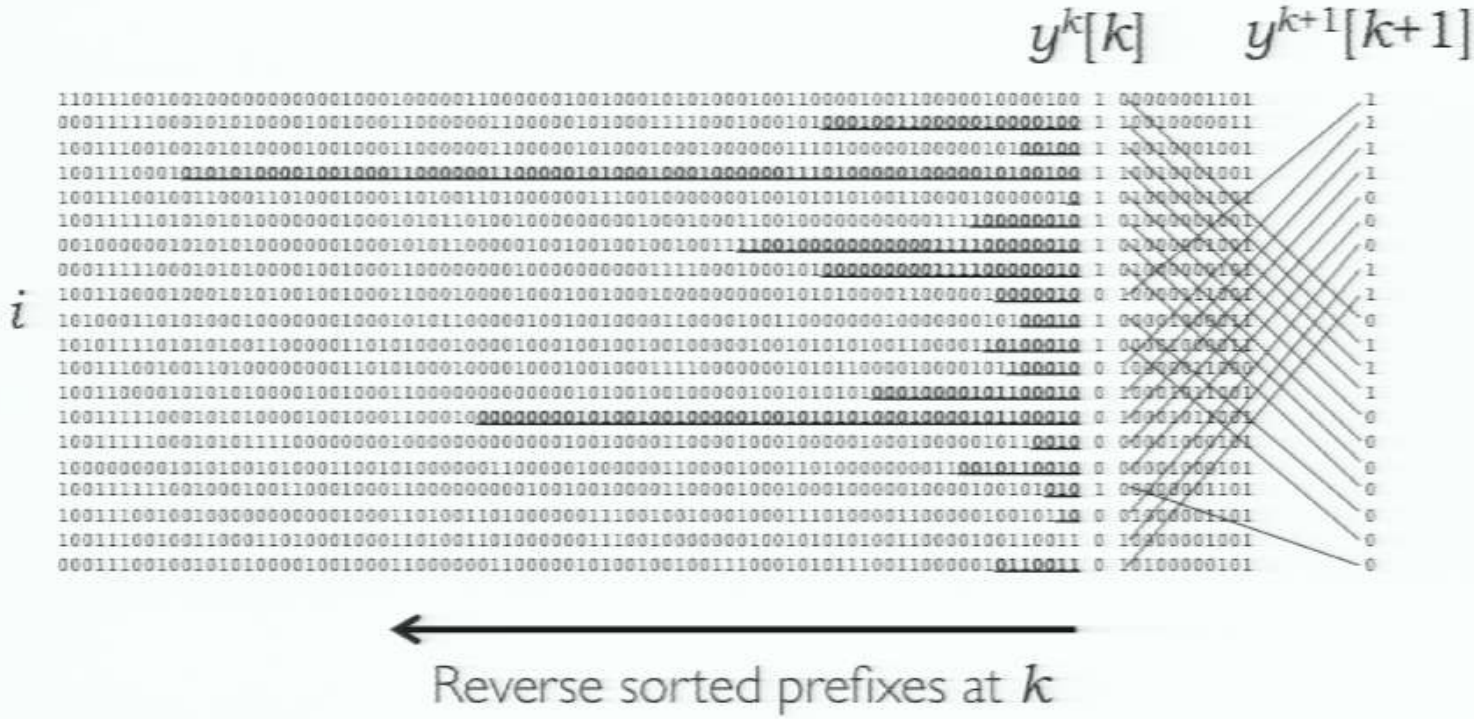
Maximal matches of new sequences to a reference panel

- 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, ~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 compresssion)
    - 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 compresssion)
    - Native order run length encodes to ~2GB

Compression performance

Simulated data subsets

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

"Real" data: 1000 Genomes phase1 chromosome 1

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, ~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

Reference panel size	1,000	2,000	5,000	10,000	20,000	50,000
Naïve	52.1	103.8	258.9	519.2	1035.5	2582.6
Indexed	0.9u 0.1s	0.9u 0.1s	0.9u 0.1s	0.9u 0.2s	1.1u 0.5s	1.7u 15s
Batch	2.3u 0.1s	2.5u 0.1s	3.5u 0.1s	4.8u 0.1s	6.8u 0.1s	12.1u 0.1s

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.

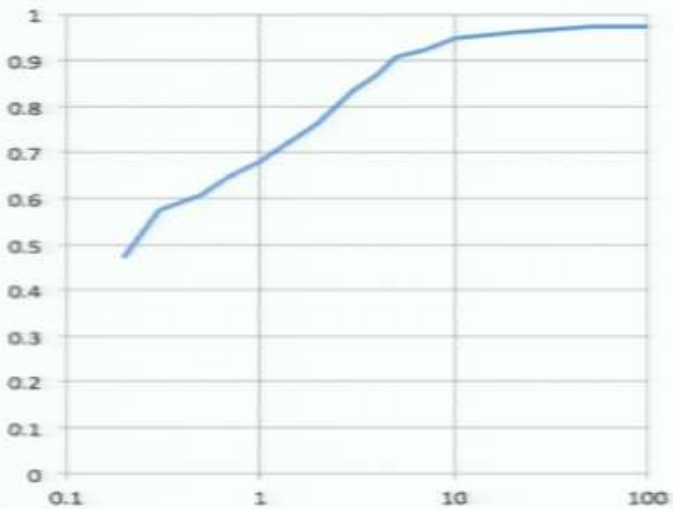
What about inference?

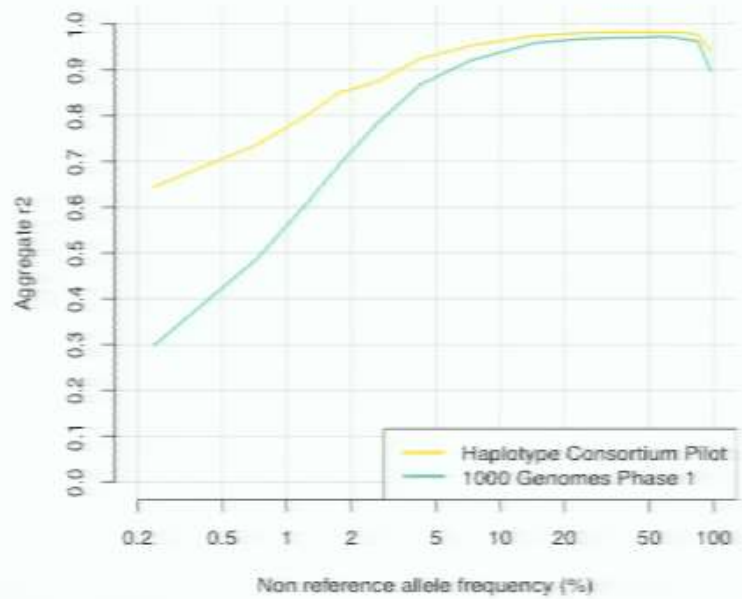
- Prediction is closely related to compression

- Consider a generative model for y given d
- e.g. Markov given by $\mathrm{p}(y_{i+1} | y_i, d_i)$
- 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 ~ 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

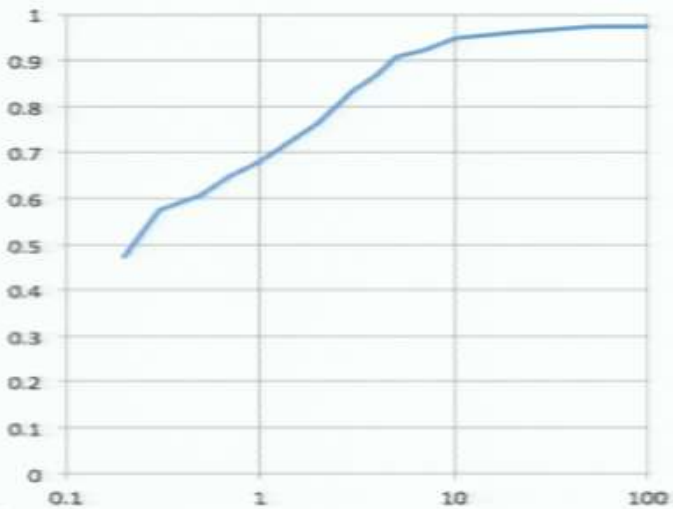
What about inference?

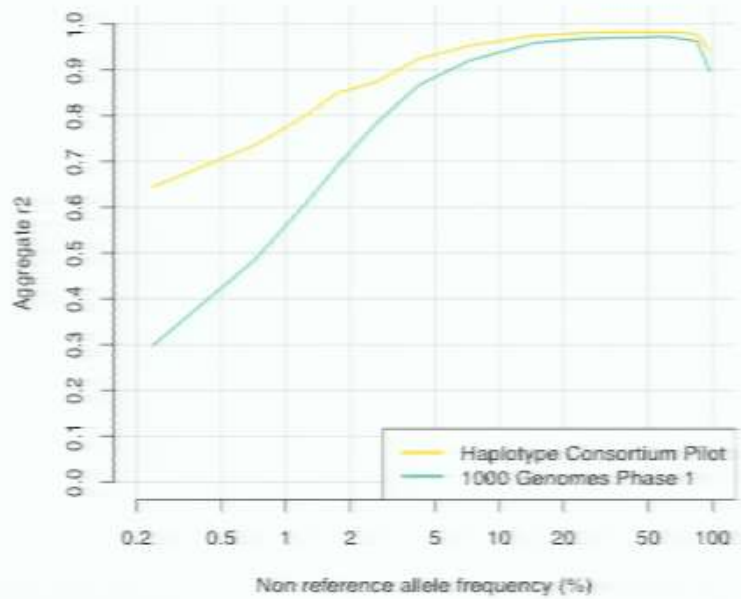
- Prediction is closely related to compression

- Consider a generative model for y given d
- e.g. Markov given by $\mathrm{p}(y_{i+1} | y_i, d_i)$
- 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 ~ Gumbel extreme value
- Imputation under Markov model is local, so very fast





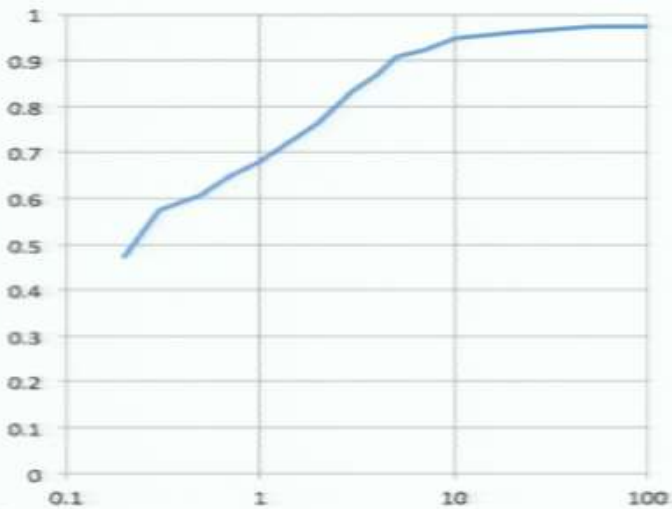
26,618 haps, 428,131 sites (~60Mb)
25 mins on Mac air – results a bit worse than using IMPUTE

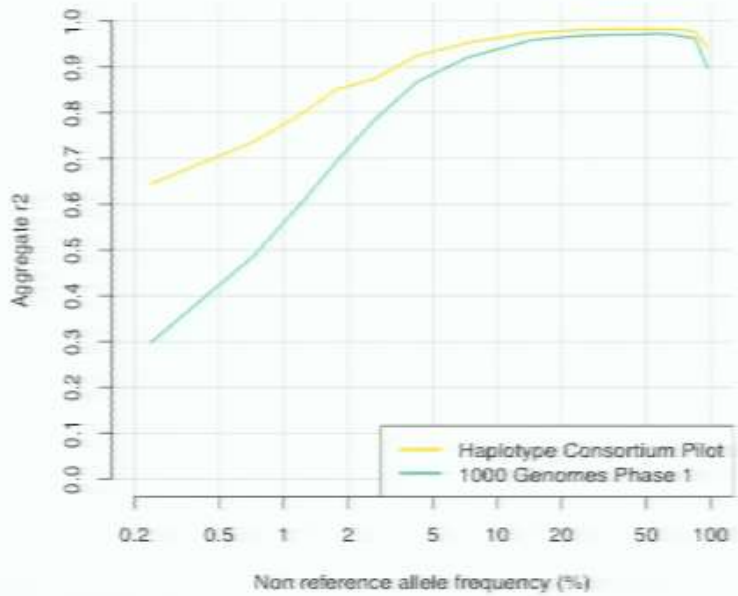
Alternative model: tree defined by d_i?

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

mississippi

i	SA[i]	
0	11	$
1	10	i$
2	7	ippi$
3	4	issippi$
4	1	ississippi$
5	0	mississippi$
6	9	pi$
7	8	ppi$
8	6	sippi$ ←
9	3	sissippi$
10	5	ssippi$ ←
11	2	ssissippi$

L(si)=8, U(si)=10

Many modern mappers (e.g. bwa) use the standard Burrows-Wheeler Transform

Non-unique match is an *interval* in Suffix Array

mississippi

i	SA[i]	
0	11	$
1	10	i$
2	7	ippi$
3	4	issippi$
4	1	ississippi$
5	0	mississippi$
6	9	pi$
7	8	ppi$
8	6	sippi$
9	3	sissippi$
10	5	ssippi$
11	2	ssissippi$

i si

L(si)=8, U(si)=10

FM approach to matching:

update L() and U() as you extend the search string

L()=0, U()=12

L(i)=1, U(i)=5

L(si)=8, U(si)=10

Constant time update gives O(M) search

Update uses FM-index
Ferragina and Manzini 2000

mississippi

i	SA[i]	
0	11	i $
1	10	p i$
2	7	s ippi$
3	4	s issippi$
4	1	m ississippi$
5	0	$ mississippi$
6	9	p pi$
7	8	i ppi$
8	6	s sippi$
9	3	s sissippi$
10	5	i ssippi$
11	2	i ssissippi$

BW[i]

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

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

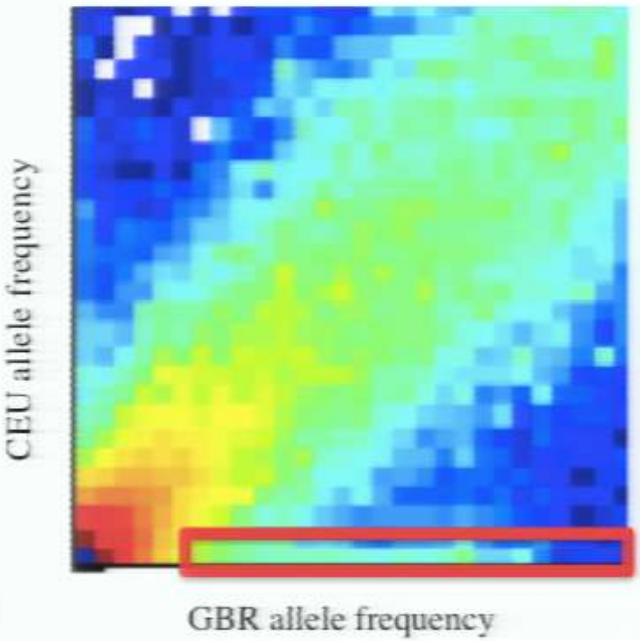
Where

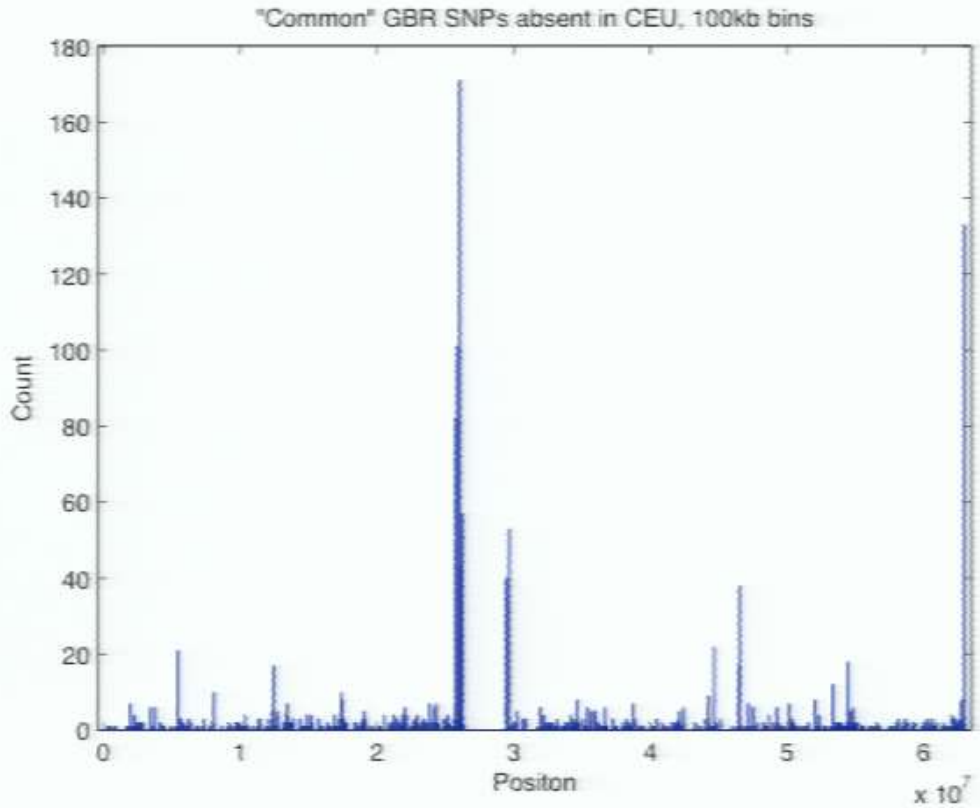
$C[a]$ is the number of letters less than a in the target string X

$P[a,i]$ 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

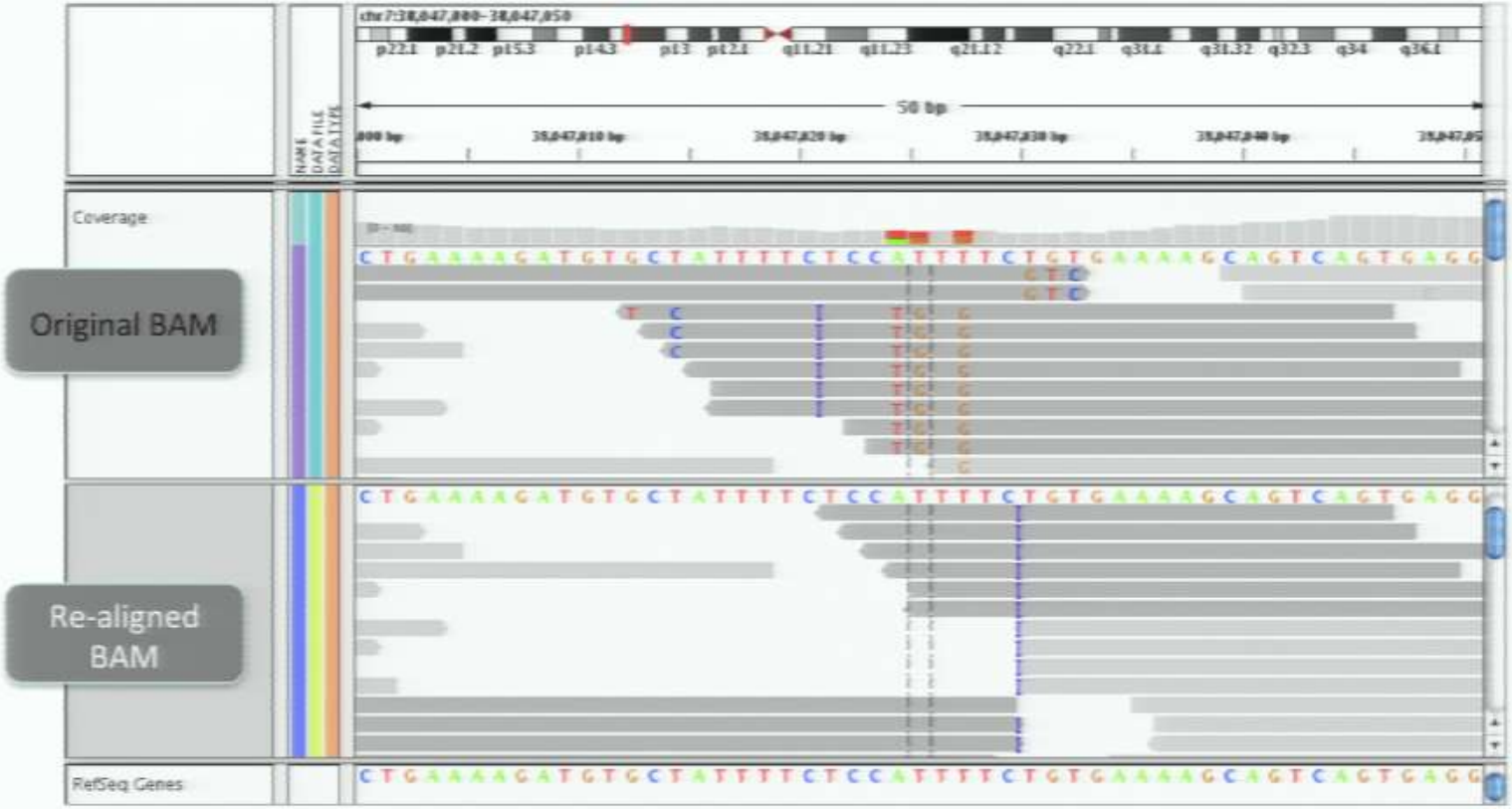




Rare, population specific "SNPs" are clustered near centromere
Many of these are likely to be artefacts, e.g. ~1% of the total

Adam Auton

Errors due to alignment at indels



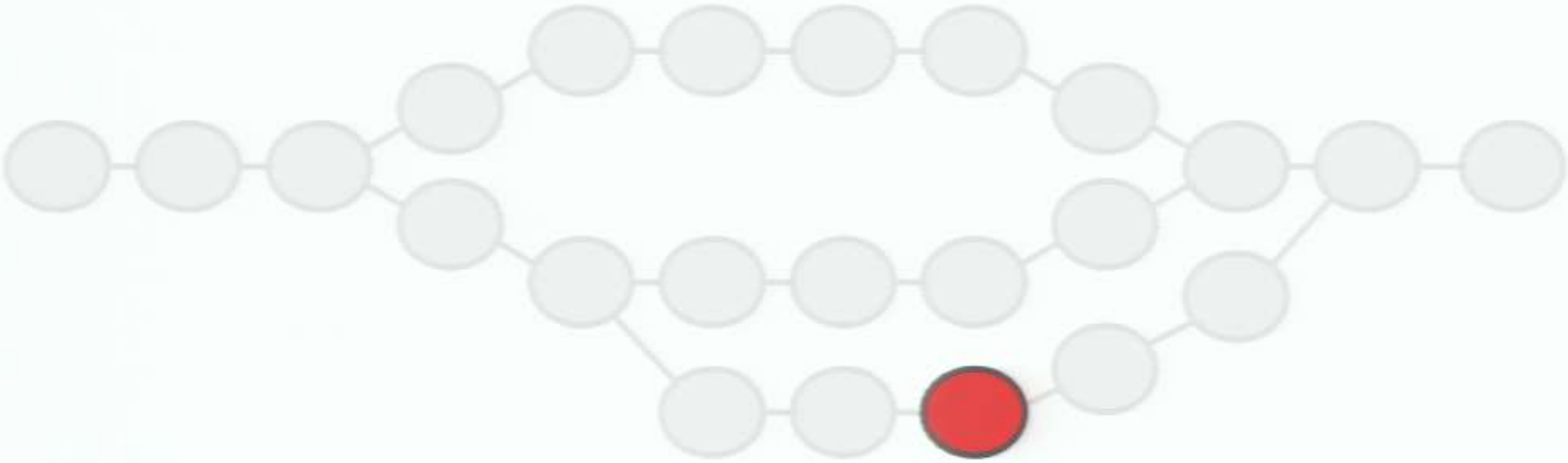
- Control using local alignment uncertainty (BAQ: Li Heng), realignment, reassembly …

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

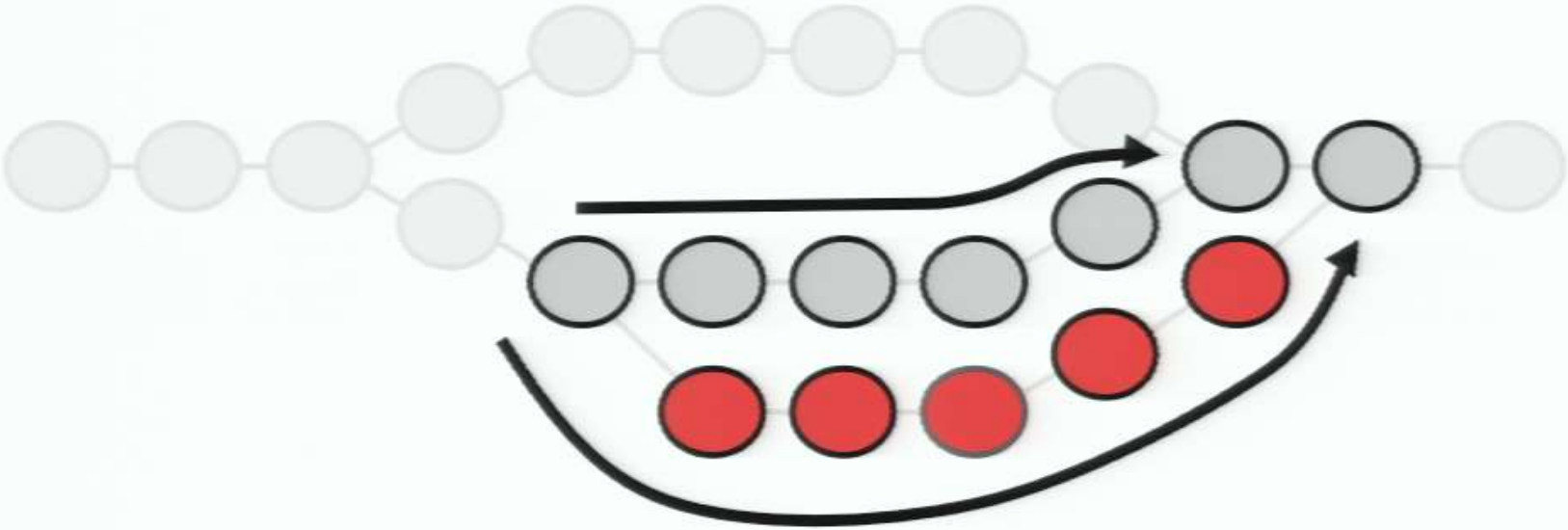
Genetic variation by assembly

Find unique reads/k-mers, then *locally* construct the string graph around reads containing these k-mers



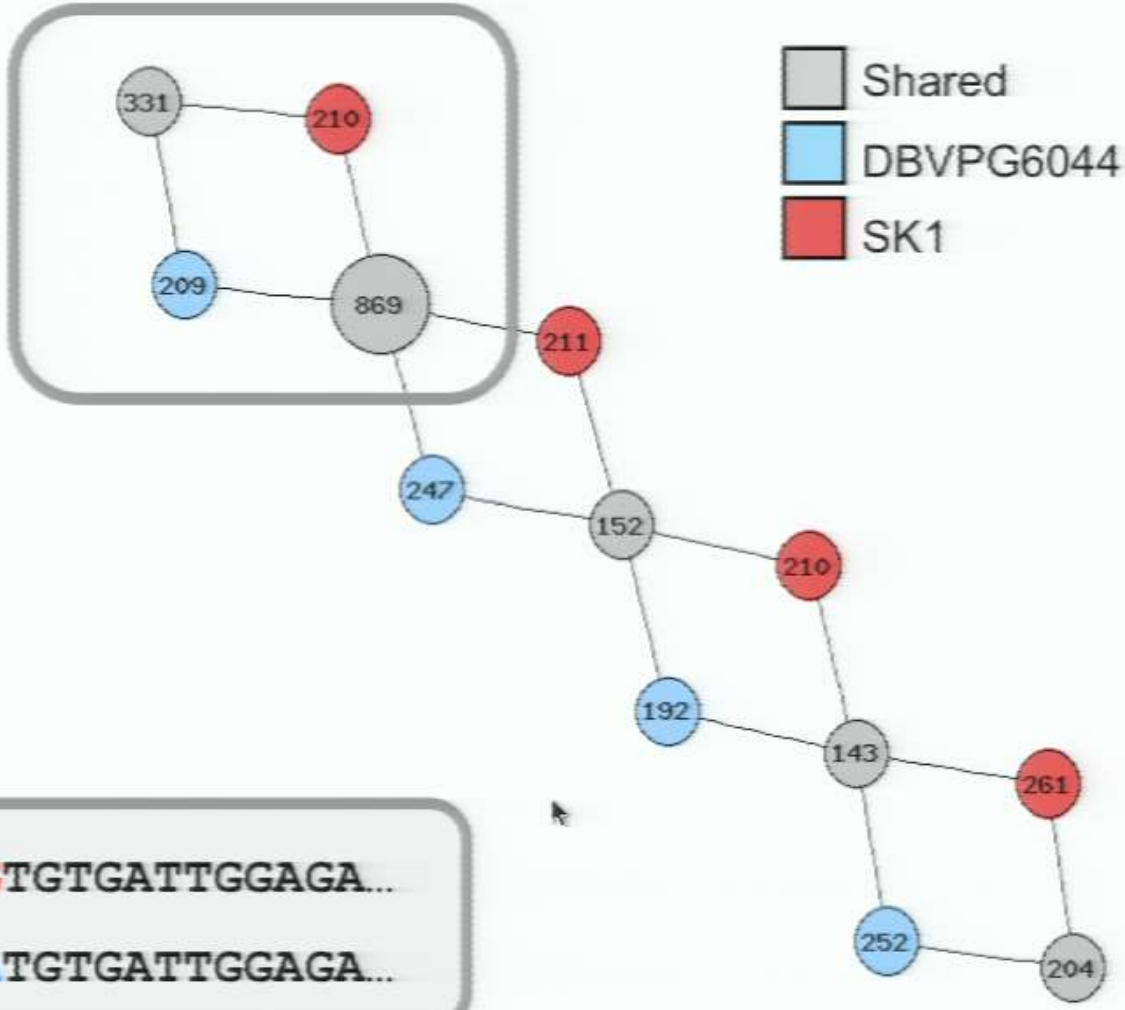
Genetic variation by assembly

...GATACATACGATCGATCTATGCATCCGAGCTGCTAGCTATCGATT... parent
...GATACATACGATCGATCTATGCATCCGAGCAGCTAGCTATCGATT... child



Evaluate evidence for haplotypes with Dindel (Albers et al. 2010) Bayesian methods

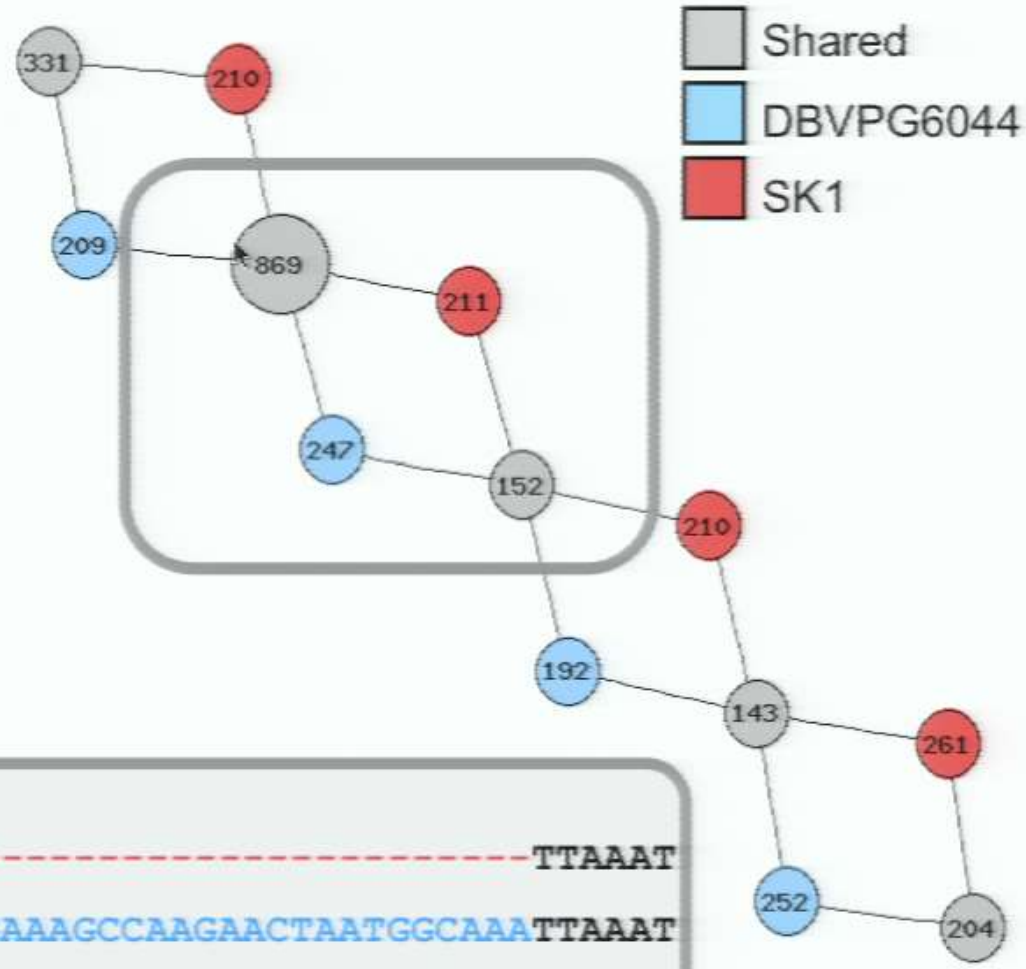
Two yeast strains co-assembled



...AAAGATATGCGCGTGTGATTGGAGA...

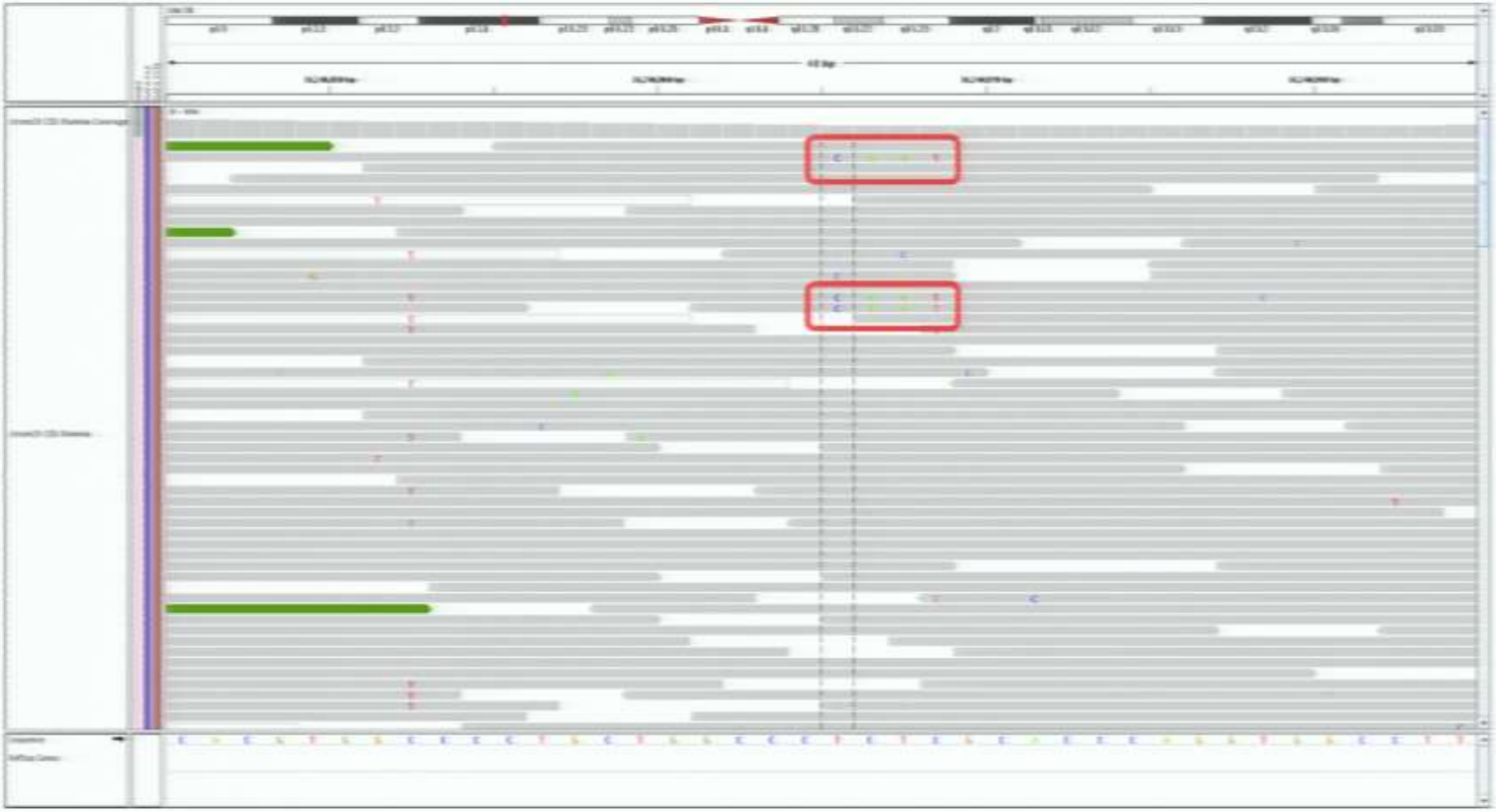
...AAAGATATGCGCATGTGATTGGAGA...

Two yeast strains co-assembled




```
CCTTGTGTT-----------------------------------------TTAAAT
CCTTGTGTTAGTAATCATATCTTCAGGAAAGCCAAGAACTAATGGCAAATTAAAT
```

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

wellcome trust sanger institute

UK 10K
Rare Genetic Variants in Health and Disease

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

1000 Genomes
A Deep Catalog of Human Genetic Variation


N
I
P
S

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

FMRI Brain Analysis

- 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.

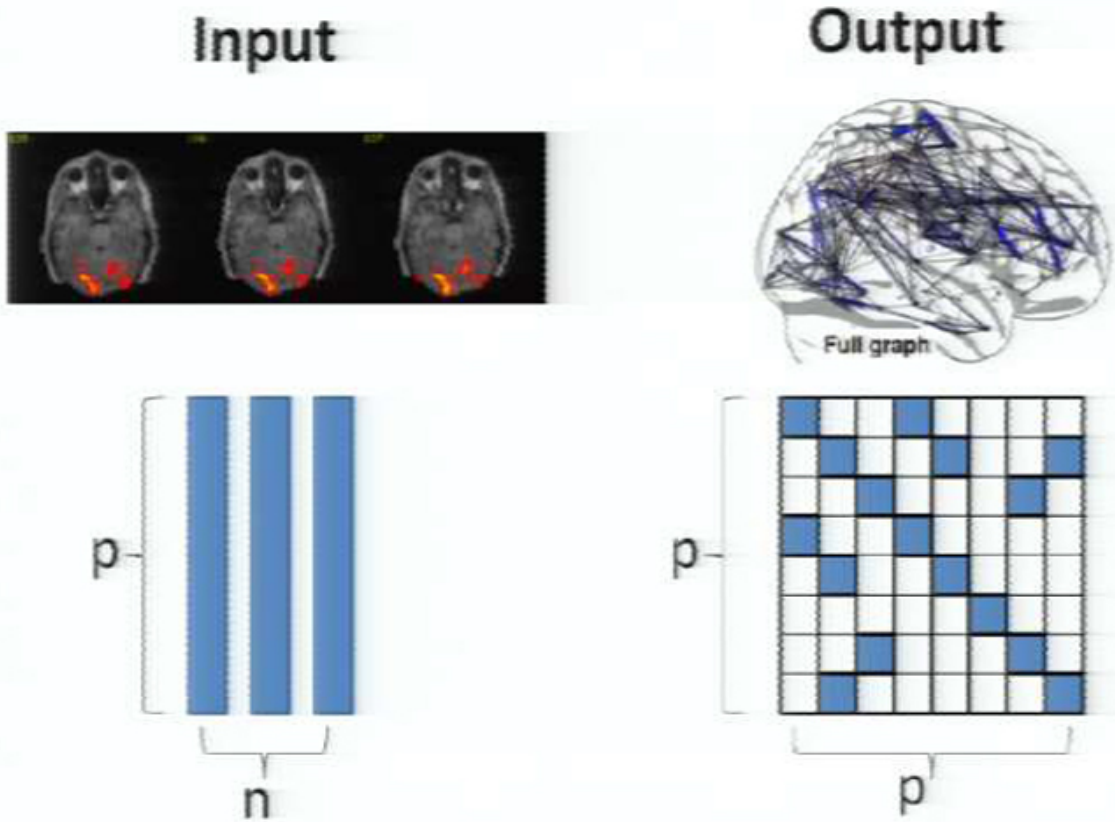


Figure from (Varoquaux et al, 2010)

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

FMRI Brain Analysis

- 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.

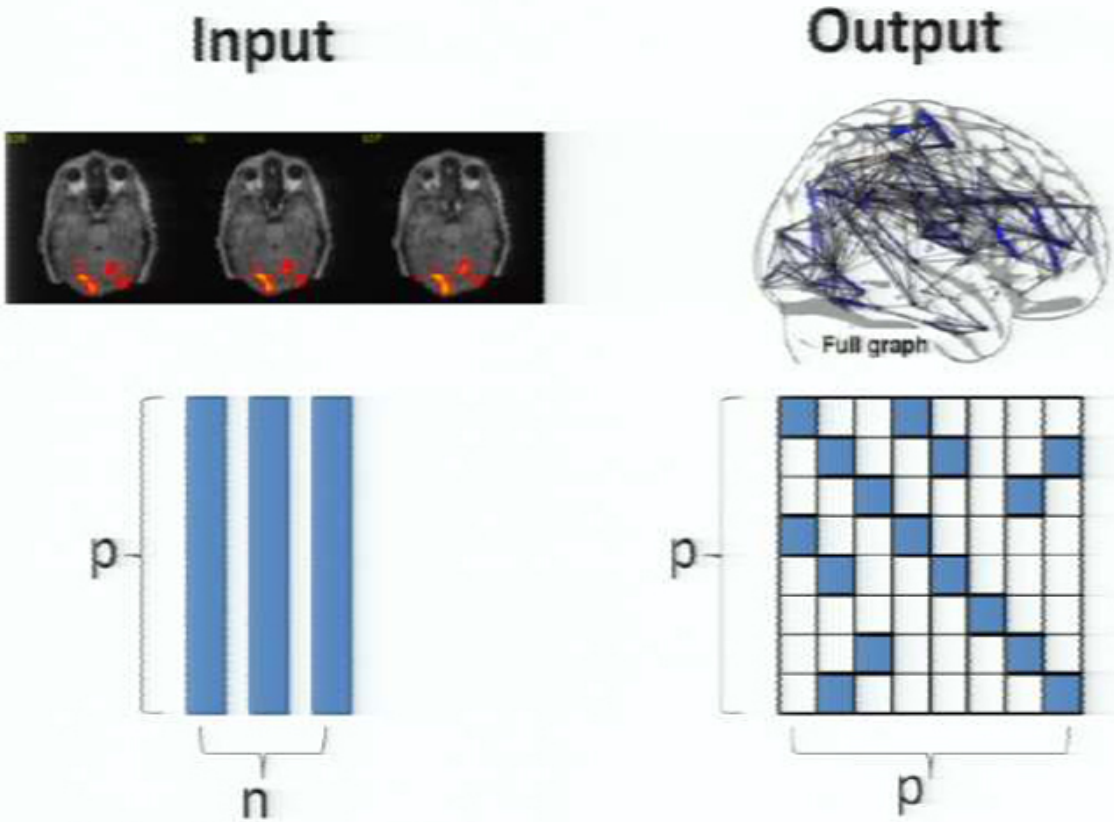


Figure from (Varoquaux et al, 2010)

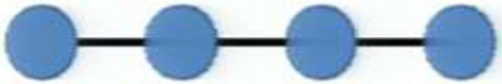
Other Applications

- 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).
- Signal Processing (Zhang & Fung, 2013).
- Anomaly Detection (Ide et al, 2009).

Inverse Covariance Estimation

- Given: n i.i.d. samples $\{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$, $\mathbf{y}_i \in R^p$, $\mathbf{y_i} \sim \mathcal{N}(\mu, \Sigma)$.
- An example – Chain graph: $y_j = 0.5 y_{j-1} + \mathcal{N}(0, 1)$



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

- Conditional independence is reflected as zeros in Σ^{-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 ℓ_1 regularization – a sparse inverse covriance matrix is preferred.
- The ℓ_1-regularized Maximum Likelihood Estimator:

$$\Sigma^{-1} = \arg\min_{X \succ 0} \Big\{ \underbrace{-\log\det X + \mathrm{tr}(SX)}_{\text{negative log likelihood}} + \lambda \|X\|_1 \Big\} = \arg\min_{X \succ 0} f(X),$$

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

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.

QUIC – proximal Newton method

- Split smooth and non-smooth terms: $f(X) = g(X) + h(X)$, where

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

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

$$\begin{aligned}\bar{g}_{X_t}(\Delta) = &\,\mathrm{tr}((S - W_t)\Delta) + (1/2)\,\mathrm{vec}(\Delta)^T (W_t \otimes W_t)\,\mathrm{vec}(\Delta) \\ &- \log\det X_t + \mathrm{tr}(SX_t),\end{aligned}$$

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

- Define the generalized Newton direction:

$$D_t = \arg\min_{\Delta} \bar{g}_{X_t}(\Delta) + \lambda\|X_t + \Delta\|_1.$$

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

Coordinate Descent Updates

- Use coordinate descent to solve:

$$\arg\min_{D}\{\bar{g}_X(D) + \lambda\|X + D\|_1\}.$$

- Closed form solution for each coordinate descent update:

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

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

- The main cost is in computing $\mathbf{w}_i^T D\mathbf{w}_j$,

where $\mathbf{w}_i, \mathbf{w}_j$ are i-th and j-th columns of $W = X^{-1}$.

Algorithm

QUIC: QUadratic approximation for sparse Inverse Covariance estimation

Input: Empirical covariance matrix S, scalar λ, 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: $D_t = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over set of free variables, (A *Lasso* problem.)
3. Line Search: use an *Armijo*-rule based step-size selection to get α 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

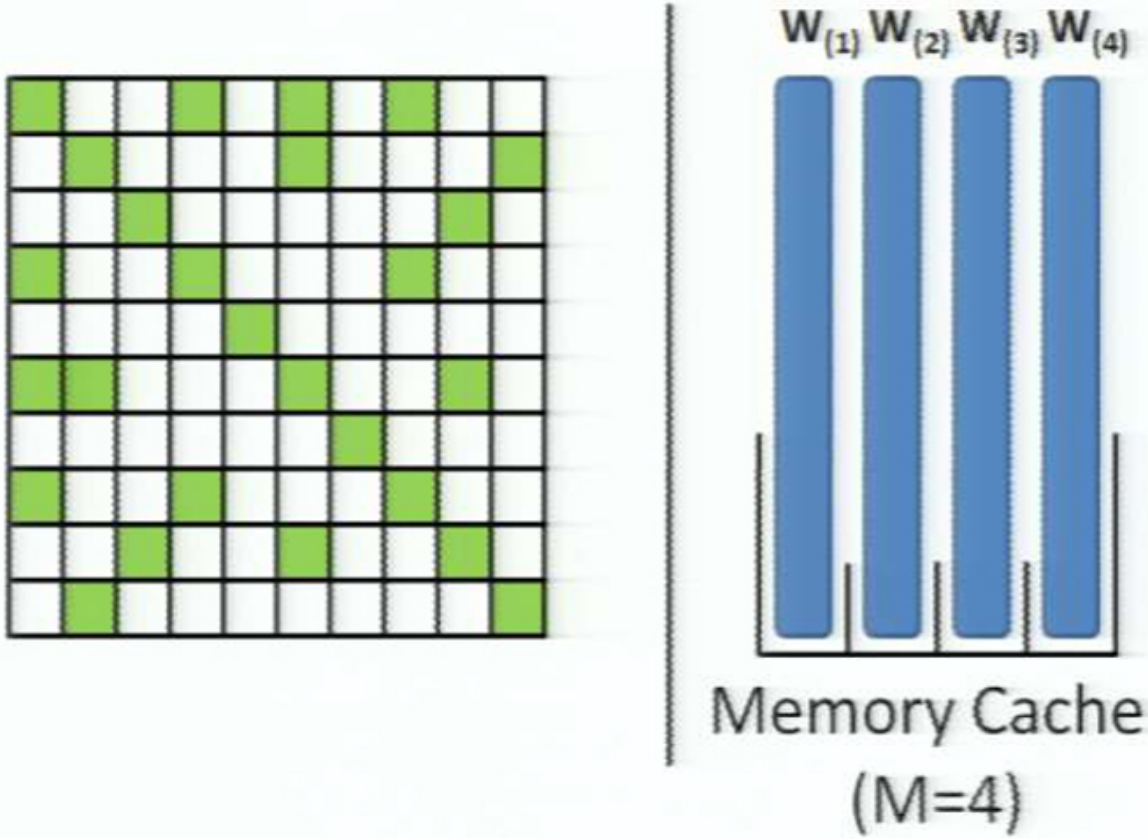
Coordinate Updates with Memory Cache

- Assume we can store M columns of W in memory.
- Coordinate descent update (i, j): compute $\mathbf{w}_i^T D \mathbf{w}_j$.
- If $\mathbf{w}_i, \mathbf{w}_j$ are not in memory: recompute by CG:

  $X\mathbf{w}_i = \mathbf{e}_i$: $O(T_{CG})$ time.

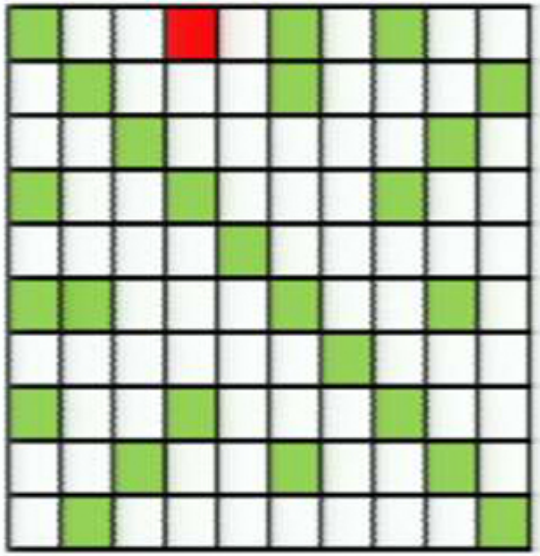
Coordinate Updates with Memory Cache

$\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4$ stored in memory.



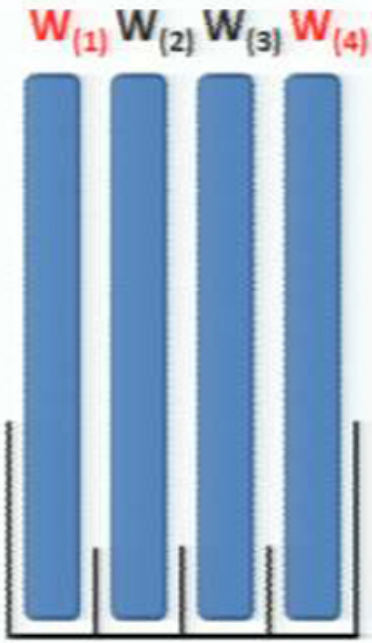
Coordinate Updates with Memory Cache

Cache hit, do not need to recompute $\mathbf{w}_i, \mathbf{w}_j$.



Update (1,4)

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

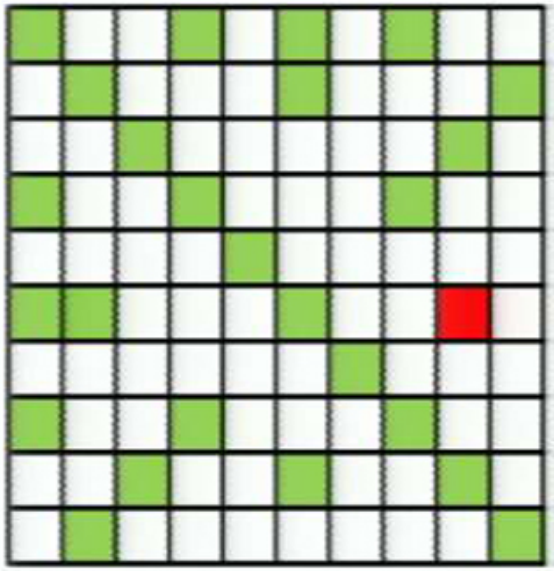


Memory Cache

(M=4)

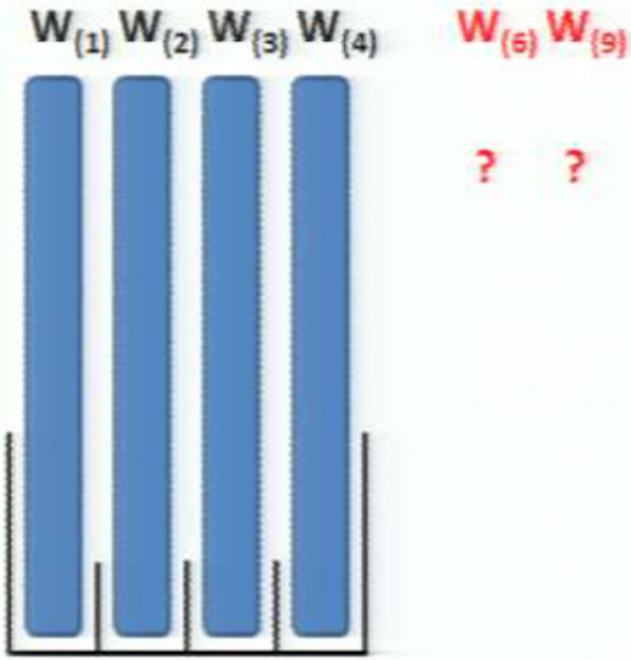
Coordinate Updates with Memory Cache

Cache miss, recompute $\mathbf{w}_i, \mathbf{w}_j$.



Update (6,9)

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

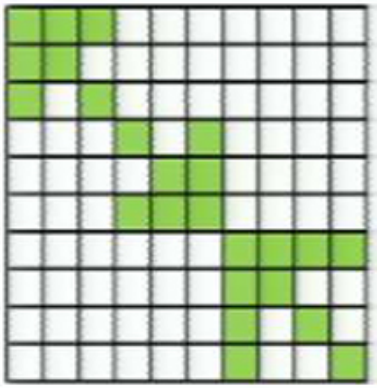


Memory Cache

(M=4)

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

- Only requires p column evaluations.

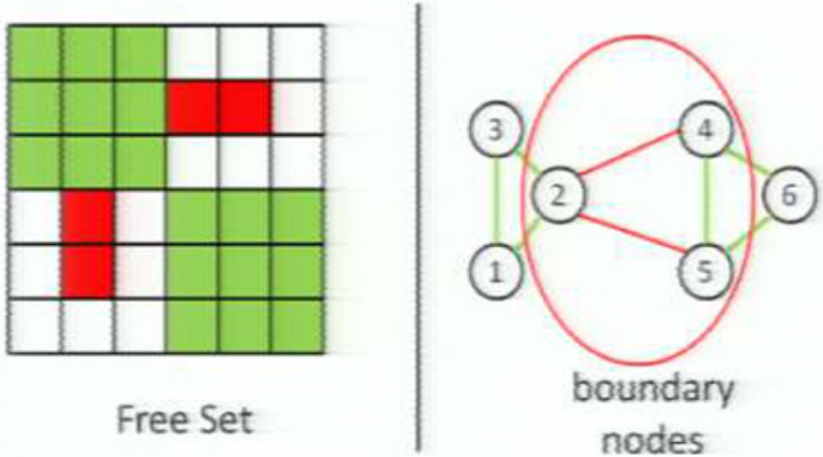
General case: block diagonal + sparse

- 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.



Graph Clustering Algorithm

- The number of columns to be computed in one sweep is

$$p + \sum_q |B(S_q)|.$$

- Can be upper bounded by

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

- Use Graph Clustering (METIS or Graclus) to find the partition.
- Example: on fMRI dataset ($p = 0.228$ million) with 20 blocks,

  random partition: need 1.6 million column computations.

  graph clustering: need 0.237 million column computations.

- 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 $O(p^2)$ storage
  - needs $O(p^3)$ computation

Line Search

- 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 $A = \begin{pmatrix} a & \mathbf{b}^T \\ \mathbf{b} & C \end{pmatrix}$,
  - $\det(A) = \det(C)(a - \mathbf{b}^T C^{-1}\mathbf{b})$
  - A is positive definite iff C is positive definite and $(a - \mathbf{b}^T C^{-1}\mathbf{b}) > 0$.
- C is sparse, so can compute $C^{-1}\mathbf{b}$ using Conjugate Gradient (CG).
- Time complexity: $T_{CG} = O(mT)$, where T is number of CG iterations.

BigQUIC

- 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

Algorithm

BigQUIC

Input: Samples Y, scalar λ, 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: $D_t = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over set of free variables.
4. Line Search: use an *Armijo*-rule based step-size selection to get α s.t. $X_{t+1} = X_t + \alpha D_t$ is
   - positive definite,
   - satisfies a sufficient decrease condition $f(X_t + \alpha D_t) \leq f(X_t) + \alpha\sigma\Delta_t$.

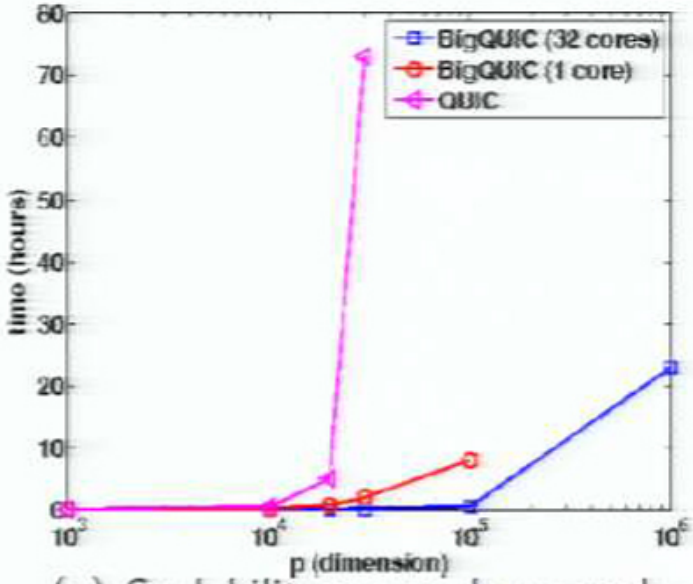
(Schur complement with conjugate gradient method.)

BigQUIC Convergence Analysis

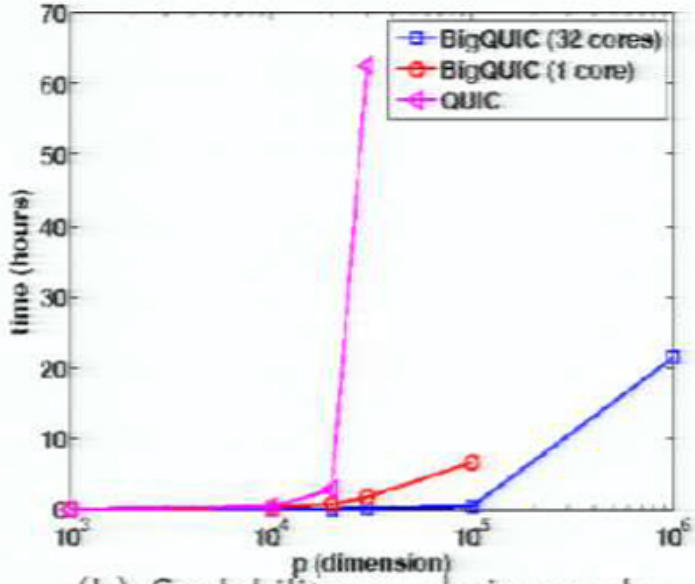
- Recall $W = X^{-1}$.
- When each $\mathbf{w}_i$ is computed by CG ($X\mathbf{w}_i = \mathbf{e}_i$):
  - The gradient $\nabla_{ij} g(X) = S_{ij} - W_{ij}$ on free set can be computed once and stored in memory.
  - Hessian ($\mathbf{w}_i^T D \mathbf{w}_j$ in coordinate updates) needs to be repeatedly computed.
- To reduce the time overhead, Hessian should be computed approximately.
- **Theorem:** the convergence rate is quadratic if $\|X\hat{\mathbf{w}}_i - \mathbf{e}_i\| = O(\|\nabla^S f(X_t)\|)$, 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)



(a) Scalability on random graph



(b) Scalability on chain graph

Figure: BIGQUIC can solve one million dimensional problems.

Experimental results

BIGQUIC is faster even for medium size problems.

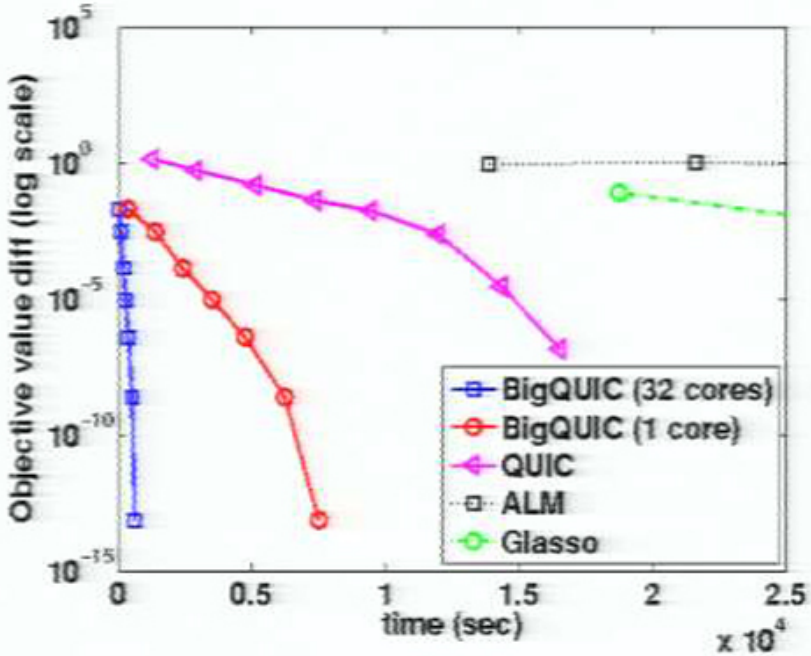
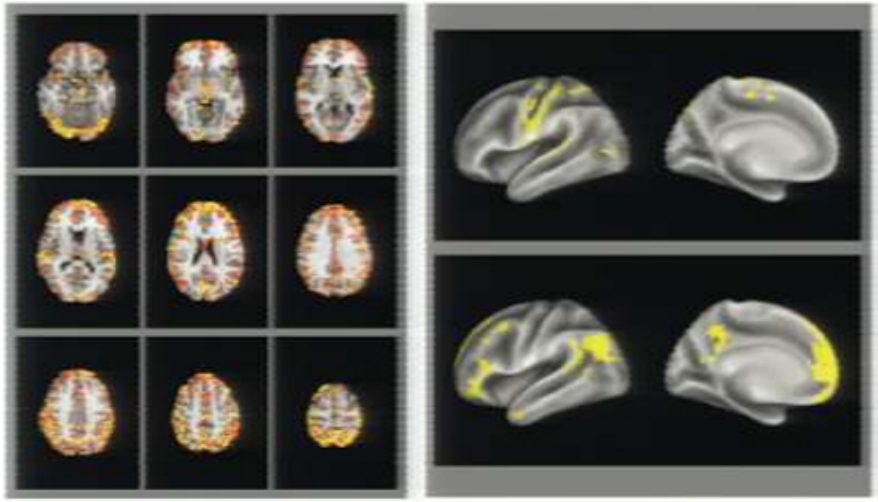


Figure: Comparison on FMRI data with a $p = 20000$ subset (maximum dimension that previous methods can handle).

Results on FMRI dataset

- 228,483 voxels, 518 time points.
- $\lambda = 0.6 \Longrightarrow$ average degree 8, BigQUIC took 5 hours.

  $\lambda = 0.5 \Longrightarrow$ 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.**

References

[1] C. J. Hsieh, M. Sustik, I. S. Dhillon, P. Ravikumar, and R. Poldrack. *BIG & QUIC: Sparse inverse covariance estimation for a million variables* . NIPS (oral presentation), 2013.

[2] C. J. Hsieh, M. Sustik, I. S. Dhillon, and P. Ravikumar. *Sparse Inverse Covariance Matrix Estimation using Quadratic Approximation.* NIPS, 2011.

[3] C. J. Hsieh, I. S. Dhillon, P. Ravikumar, A. Banerjee. *A Divide-and-Conquer Procedure for Sparse Inverse Covariance Estimation.* NIPS, 2012.

[4] P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie. *Newton-Like Methods for Sparse Inverse Covariance Estimation.* Optimization Online, 2012.

[5] O. Banerjee, L. El Ghaoui, and A. d'Aspremont *Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data.* JMLR, 2008.

[6] J. Friedman, T. Hastie, and R. Tibshirani. *Sparse inverse covariance estimation with the graphical lasso.* Biostatistics, 2008.

[7] L. Li and K.-C. Toh. *An inexact interior point method for l1-reguarlized sparse covariance selection.* Mathematical Programming Computation, 2010.

[8] K. Scheinberg, S. Ma, and D. Glodfarb. *Sparse inverse covariance selection via alternating linearization methods.* NIPS, 2010.

[9] K. Scheinberg and I. Rish. *Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach.* Machine Learning and Knowledge Discovery in Databases, 2010.