Tag SNP Selection and Disease Gene Location Inference

Limsoon Wong

(Based on works by Guimei Liu, Li Lin, Yue Wang, Li Lin, Carol Lai, & Tze Yun Leong)

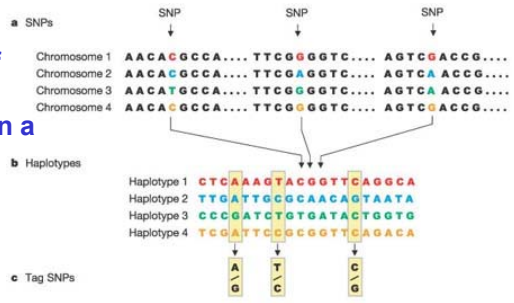
Plan

- **Tag SNP Selection**
  - Motivation & previous works
  - FastTagger
  - Comparison with MMTagger
  - Effectiveness of optimizations
- **Disease Gene Location Inference**
  - Motivation & previous works
  - LinkageTracker
  - Comparison w/ HapMiner, GeneRecon, BLADE

Tag SNP Selection

What is tag SNP?

- **Tag SNP is a representative SNP in a region of genome w/ high linkage disequilibrium**
- **Enable identification of genetic variation w/o genotyping each SNP in a chromosomal region**
- **Useful for discovering genes responsible for various disorders**



Motivation

- Genotyping all SNPs are very expensive
- Adjacent SNPs are often not independent

⇒ Desirable to select a subset of SNPs (the tag SNPs) that are sufficient to infer all the other SNPs

- Existing tag SNP selection algo cannot handle chromosomes containing more than 100k SNPs

r^2 Statistic

- **Nearby SNPs are transmitted together**
- **So they tend to be highly associated**

- **r^2 statistic is a common metric to measure correlation of SNPs**

$$r^2(SNPi, SNPj) = \frac{(P(XY) - P(X)P(Y))^2}{P(X)P(x)P(Y)P(y)}$$

where P(XY), P(Xy), P(xY), P(xy) are freq of possible alleles; P(X) =P(XY)+P(Xy), P(x) =P(xY)+P(xy), etc

Example SNP Data Set

	SNP1	SNP2	SNP3	SNP4	SNP5	SNP6	SNP7	SNP8	SNP9
Major allele	A	G	G	T	A	T	C	A	C
Minor allele	C	A	A	C	T	A	T	T	G
locus	100	200	300	400	500	600	700	800	900
T11	A	G	G	T	A	A	T	A	C
T12	C	A	G	C	T	T	C	A	C
T21	A	G	A	T	A	T	C	A	G
T22	A	G	G	T	A	T	C	T	C
T31	C	A	A	C	T	A	T	A	C
T32	A	G	G	T	A	T	C	T	C
T41	C	A	G	C	T	T	C	A	G
T42	A	G	G	T	A	A	T	T	G
T51	A	A	A	T	T	A	T	A	C
T52	A	G	A	T	A	T	C	T	C

$$r^2(\text{SNP1},\text{SNP2}) = \frac{(P(AG) - P(A)P(G))^2}{P(A)P(C)P(G)P(A)} = \frac{(0.6 - 0.7*0.6)^2}{0.7*0.3*0.6*0.4} = 64.3\%$$

$\{SNP_1, \ldots, SNP_k\} \rightarrow SNP_j$

- **Let Y and y be major and minor alleles of SNP_j**
- **Divide haplotypes H over S = $\{SNP_1, \ldots, SNP_k\}$ into groups X and x where**
  - H ∈ X if P(HY) > P(Hy)
  - H ∈ x otherwise
- **Then**

$$r^2(S, SNPj) = \frac{(P(XY) - P(X)P(Y))^2}{P(X)P(x)P(Y)P(y)}$$

where $P(XY) = \Sigma_{H \in X} P(HY)$, $P(X) = \Sigma_{H \in X} P(H)$, etc

Example SNP Data Set

	SNP1	SNP2	SNP3	SNP4	SNP5	SNP6	SNP7	SNP8	SNP9
Major allele	A	G	G	T	A	T	C	A	C
Minor allele	C	A	A	C	T	A	T	T	G
locus	100	200	300	400	500	600	700	800	900
T11	A	G	G	T	A	A	T	A	C
T12	C	A	G	C	T	T	C	A	C
T21	A	G	A	T	A	T	C	A	G
T22	A	G	G	T	A	T	C	T	C
T31	C	A	A	C	T	A	T	A	C
T32	A	G	G	T	A	T	C	T	C
T41	C	A	G	C	T	T	C	A	G
T42	A	G	G	T	A	A	T	T	G
T51	A	A	A	T	T	A	T	A	C
T52	A	G	A	T	A	T	C	T	C

- **Hapolotypes over S = {SNP7, SNP8} are {CA, TA, CT, TT}**

H	P(HY)	P(Hy)	
CA	0.1	0.2	x
TA	0.3	0.0	X
CT	0.3	0.0	X
TT	0.0	0.1	x

$$r^2(\{\text{SNP7},\text{SNP8}\},\text{SNP9}) = \frac{(P(XY)-P(X)P(Y))^2}{P(X)P(x)P(Y)P(y)} = \frac{(0.6-0.6*0.7)^2}{0.6*0.4*0.7*0.3} = 64.3\%$$

Btw, r^2(SNP7,SNP9) = 0. 79%, r^2(SNP8,SNP9) = 0.79%

Tag SNP Selection

- **Given a set S of SNPs, find the smallest set of tag SNPs S_{tag} such that for every $SNP_j \in S - S_{tag}$, there is at least one SNP set $S_j \subseteq S_{tag}$ such that**
  - **$r^2(S_j, SNP_j) \geq min\_r^2$**
  - **$|S_j| \leq max\_size$**
  - **Distance betw every pair of SNPs in $S_j \cup \{SNP_j\}$ is no larger than max_dist**

Existing Algo

- **Step 1: Correlations betw SNPs within certain distance are calculated**
- **Step 2: Find smallest set of tag SNPs using correlations calculated in Step 1**
- **Most algo use greedy approach to find a near optimal set of tag SNPs in Step 2**

- **Earlier tag SNP selection methods rely on pairwise correlations**
- **MultiTag & MMTagger find multimarker rules**
  - {SNP1, SNP2, SNP3} →SNPx
  - Cannot handle >100k SNP
    - **MultiTag takes hundreds of hours for 30k SNP**
    - **MMTagger takes hours & 1GB memory for 30k SNP**

FastTagger

FastTagger

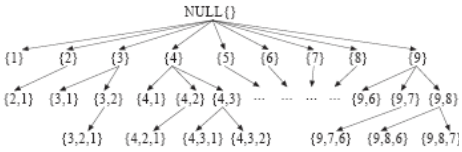
- **Step 1: Use data mining techniques to mine tagging correlation rules**
- **Step 2: Use a greedy algorithm to select tag SNPs using the tagging correlation rules generated**
- **Several techniques are employed to effectively reduce the search space of Step 1 and memory consumption of Step 2**

FastTagger

- **Four ideas to reduce # of rules to be tested and generated**

1. Merge nearby equiv SNPs
2. Prune redundant correlation rules
3. Skip rule if its RHS has been covered many times
4. If total size of rules exceeds memory, divide chromosome into blocks, and then find tag SNPs within each block
   - **Can handle >100k SNPs using <50MB memory**

Mining Tagging Correlation Rules



- **SE-tree of SNP combinations in our running example**
  - SNP_i represented by i
  - Max_dist = 300
  - Max_size =3

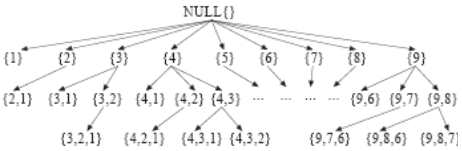
- **Possible combinations of SNPs are tested in depth-first left-to-right manner**
- **Candidate RHS of each SNP set S include all SNPs within max_dist of every SNP in S**
- **r^2 is computed for each S in the SE-tree and each SNP in its candidate RHS**

⇒ **Those above min_r^2 are the tagging correlation rules**

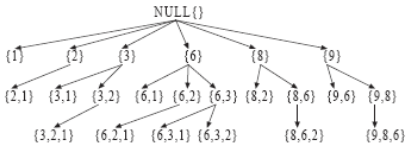
Merge Nearby Equiv SNPs

- **Many SNPs have identical occurrences. The r^2 of these equiv SNPs is always 1**
- **Lemma: If SNP_1 and SNP_2 are equiv, then for any SNP_j, $r^2(SNP_1, SNP_j) = r^2(SNP_2, SNP_j)$**
- **Optimization: Merge equiv SNPs within max_dist of each other, & use one as the representative**

- **Original SE-tree**



- **SE-tree after merging equiv SNPs is a lot smaller!**



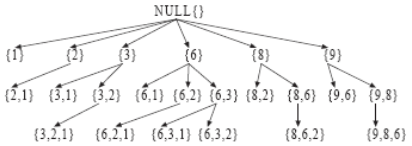
1 merged with 4. 2 merged with 5. 6 merged with 7.

Prune Redundant Rules

- **Definition:**
  If SNP_j can be tagged by a SNP set S, then rule S′ → SNP_j, such that S′ is a proper superset of S, is redundant

- **Optimization: Prune redundant rules**

- **To prune redundant rules, before generate S → SNPj, check if S′ → SNP_j where S′ ⊂ S is already generated**



- **Easy to do given enumeration order of SE-tree**

Skipping Rules

- **If a SNP can be tagged by many other SNPs, then during tag SNP selection process, the SNP has high probability to be covered by selected tag SNPs**

- **Optimization: If SNP_j occurs in RHS of tagging rules enough # of times, then SNP_j need not be considered as RHS candidate in future rule generation**

Select Tag SNP Greedily

1. S_{tag} = All SNPs not in RHS of any rule
2. $S_{covered}$ = $\{SNP_j \mid S \rightarrow SNP_j, S \subseteq S_{tag}\} \cup S_{tag}$
3. Pick $SNP_i \notin S_{tag}$ and SNP_i covers largest # of SNPs
4. Add SNP_i to S_{tag}
5. Goto Step 2 if there remains SNP $\notin S_{covered}$

- The algo above requires FastTagger to keep all rules in memory. Impossible if too many rules

⇒ **Divide chromosome into chunks. Run FastTagger on each chunk separately**
  - FastTagger can handle >100k SNPs using 50MB

Performance Studies

Data Sets

- **Japanese and Han in HapMap release 21**
  - 45 unrelated individuals
  - 6 chromosomes
    - **chr1. chr2. chr3. chr19. chr21 and chr22**

Chromosome	#SNPs	#representative SNPs
chr1	149,716	70,823
chr2	169,905	84,098
chr3	135,058	63,814
chr19	28,931	11,124
chr21	28,914	13,270
chr22	26,595	11,042

- **Note greatly reduced # of rep SNPs after merging of nearby equiv SNPs!**

Comparison w/ MMTagger

Wang & Jiang. GIW2008

	max_size	*min_r2*	Running time		#tag SNPs		
			FastTagger	MMTagger	FastTagger	MMTagger	Reduction ratio
chr1	1	0.9	0.08	-	50,446	-	-
chr2	1	0.9	0.09	-	52,447	-	-
chr3	1	0.9	0.08	-	44,984	-	-
chr19	1	0.9	0.01	-	12,488	-	-
chr21	1	0.9	0.02	-	10,138	-	-
chr22	1	0.9	0.02	-	10,411	-	-
chr1	2	0.9	2.47	9.47	33,964	57,131	0.406
chr2	2	0.9	2.85	11.88	34,055	60,356	0.436
chr3	2	0.9	2.48	8.09	29,407	50,528	0.418
chr19	2	0.9	0.37	0.53	9,101	13,316	0.317
chr21	2	0.9	0.62	0.80	6,712	11,493	0.416
chr22	2	0.9	0.83	0.94	7,163	11,567	0.381
chr1	3	0.9	74.27	-	28,219	-	-
chr2	3	0.9	91.09	-	27,828	-	-
chr3	3	0.9	82.21	-	24,066	-	-
chr19	3	0.9	9.08	-	7,835	-	-
chr21	3	0.9	20.81	-	5,510	-	-
chr22	3	0.9	34.96	-	5,967	-	-
chr1	3	0.95	77.25	-	35,496	-	-
chr2	3	0.95	95.92	-	35,435	-	-
chr3	3	0.95	85.41	-	30,632	-	-
chr19	3	0.95	9.42	61.90	9,433	10,032	0.060
chr21	3	0.95	21.43	77.56	6,929	7,404	0.064
chr22	3	0.95	35.76	180.00	7,321	7,788	0.06

Heuristic for skipping rules turned off for fair comparison

Comparison w/ MMTagger

	FastTagger		MMTagger
	step1	step2	
chr1	12.17MB	375.83MB	-
chr2	18.03MB	613.57MB	-
chr3	11.73MB	424.84MB	-
chr19	2.97MB	47.35MB	657MB
chr21	3.27MB	83.59MB	1210MB
chr22	3.45MB	81.56MB	1216MB

Max_size =3, min_r^2 = 0.95

- **MMTagger consumes much more memory**
  - Failed on large chromosomes when max_size = 3
- **Step 2 of FastTagger consumes much more memory than Step 1 because this step needs to store rules generated in the memory**

Effectiveness of Merging Nearby Equiv SNPs

			merging SNPs				without merging			
	max_size	*min_r2*	time	mem	#tag SNPs	#rules	time	mem	#tag SNPs	#rules
chr1	2	0.9	2.47	55.87MB	33,964	937,806	27.57	221.56MB	34,121	13,363,677
chr2	2	0.9	2.85	69.14MB	34,055	1,340,978	34.69	304.55MB	34,224	20,559,496
chr3	2	0.9	2.48	53.10MB	29,407	974,249	25.30	216.87MB	29,539	13,155,695
chr19	2	0.9	0.37	8.84MB	9,101	134,855	3.47	29.10MB	9,132	1,565,499
chr21	2	0.9	0.62	11.59MB	6,712	182,416	6.87	44.62MB	6,733	2,605,213
chr22	2	0.9	0.83	10.75MB	7,163	170,649	7.26	37.63MB	7,188	2,377,089
chr19	3	0.95	9.42	47.35MB	9,433	1,025,160	138.29	472.10MB	9,476	17,863,615
chr21	3	0.95	21.43	83.59MB	6,929	1,859,282	318.32	937.55MB	6,959	35,365,338
chr22	3	0.95	35.76	81.56MB	7,321	1,775,501	409.39	790.65MB	7,342	31,025,297

- **# of rules, tag SNPs, and runtime are significantly reduced**

Effectiveness of Pruning Redundant Rules

	with pruning		without pruning	
	mem	#rules	mem	#rules
chr1	375.83MB	8,473,070	1000.59MB	28,784,330
chr2	613.57MB	15,611,939	1473.02MB	46,202,676
chr3	424.84MB	9,942,665	1076.30MB	31,521,519

Max_size =3, min_r^2 = 0.95

- **Memory usage and # rules are significantly reduced**

Effectiveness of Skipping Rules

	no skipping				*cover_thres*=5			
	time	mem	#tag SNPs	#rules	time	mem	#tag SNPs	#rules
chr1	77.25	375.83MB	35,496	8,473,070	59.43	193.90MB	36,180	3,466,008
chr2	95.92	613.57MB	35,435	15,611,939	68.04	276.89MB	36,403	5,263,179
chr3	85.41	424.84MB	30,632	9,942,665	64.13	207.24MB	31,393	3,845,950
chr19	9.42	47.35MB	9,433	1,025,160	7.87	27.87MB	9,545	478,412
chr21	21.43	83.59MB	6,929	1,859,282	16.98	43.68MB	7,073	773,891
chr22	35.76	81.56MB	7,321	1,775,501	29.25	44.11MB	7,445	776,435

- **Memory usage and runtime are significantly reduced, while # of tag SNPs is marginally increased**

Conclusions

- **Compared to existing genome-wide tag SNP selection algo using multi-marker correlations, FastTagger is**
  - Many times faster
  - Consumes much less memory
  - Can work on chromosomes with > 100k SNPs
- **Merging equiv SNPs together is most effective technique in reducing running time and memory consumption**

Disease Gene Location Inference

Motivation

- **Identification of disease gene location has big impact on patient treatment planning**
- **Major challenge: How to maximize haplotype info extraction in association mapping of disease under extreme conditions**
  - # of samples with mutation of interest is very low
  - Samples contain lots of errors and noise

Some Previous Works

- **BLADE [Liu, 2001]**
  - MCMC-based Bayesian parameter estimation
  - Assume all mutations occur in same location

  ⇒ No locus heterogeneity
- **HPM [Toivonen, 2000]**
  - Mine freq patterns in cases
  - $\chi 2$ test to discriminate cases vs controls
  - Markers w/ largest freq in significant patterns = disease gene location
- **HapMiner [Li & Jiang, 2005]**
  - Density-based clustering
  - "Model free", no need genealogy info
  - Very fast, but sensitive to clustering parameters
- **GeneRecon [Mailund, 2006]**
  - "Shattered coalescent", allow multiple founding mutations
  - Take long time to process a few hundred samples with a few tens of markers

LinkageTracker

LinkageTracker

- **Model free; no need any population ancestry info about disease and genealogy of haplotypes**
- **No need to set complex parameters prior to the disease gene location inference process**
- **Two steps**
  1. Discover linkage disequilibrium patterns by constrained level-wise search
  2. Marker inference via Fisher's P-value estimation method

Constrained Level-Wise Search

- **LinkageTracker mine patterns <dx1, dx2, …, dxk>**
  - dxi = allele of marker i of sample x
  - dxi = * means missing marker allele
  - E.g., (3,5,6,*,*,4)
- **Allelic association beyond 20cM is weak [Long & Langley, 1999]**

⇒ **Enumerate all possible patterns <dx1, dx2, …, dxk> where markers in a pattern is no more than 20cM apart**

- **Score patterns using odds ratio**
- **Pick patterns w/ significant P-value**

Marker Inference

- **Σ(c) follows χ2 distribution w/ df=2n [Fisher 1970]**

⇒ **Can infer combined P-value from Σ(c)**

⇒ **Pick marker with best combined P-value**

Marker	1 2 3 4 5 6	*P-Value*	c = -2 * ln(*P*)
Pattern01	* 4 3 * * *	0.0090	9.4211
Pattern02	2 4 * * 6 1	0.0065	10.0719
Pattern03	2 4 3 5 * *	0.0030	11.6183
Pattern04	* * 3 5 * 1	0.0100	9.2103
Pattern05	2 4 * 5 6 *	0.0045	10.8074

	Freq	Σ(c)	Combine *P-Value*
Marker 1 allele 2	3	32.4975	1.3098E-05
Marker 2 allele 4	4	41.9186	1.4027E-06
Marker 3 allele 3	3	30.2497	3.5236E-05
Marker 4 allele 5	3	31.6390	1.9160E-05
Marker 5 allele 6	2	10.0719	0.0392
Marker 6 allele 1	2	19.2822	0.007

Performance Studies

Cystic Fibrosis Data Set [Kerem, 1989]

- **23 bi-allelic markers around CFTCR gene**
- **92 control haplotypes, 94 disease haplotypes**
- **Founder mutation is betw marker 17 and 18**
- **67% of disease haplotypes carry founder mutation**
- **Disease haplotypes have 39% missing info**
- **To study disease gene location inference w/ noise, we divide this CF data set into 3 subsets**
  - Set A: Disease haplotypes carrying founder mutation, 63 samples
  - Set B: Disease haplotypes w/o founder mutation, 31 samples
  - Set C: Control group haplotypes, 92 samples

Standard Conditions

- **Pick 50 from Sets A & B as cases**
- **Pick 50 from Set C as controls**

	Standard Deviation of SSE	Avg SSE	Avg Time (in seconds)
Blade	**0.0036**	0.01564	58.0202
HapMiner	0.0129	**0.00588**	**1.986**
HapMiner (*Modified*)	0.0354	0.04376	-
LinkageTracker	0.0043	0.00811	125.534
GeneRecon	0.0149	0.01466	4775.6566

- **HapMiner is most accurate & very fast, but parameter sensitive**
- **LinkageTracker is 2nd, but more consistent**

Low Occurrence

- **Can we find the founder mutation under conditions of low occurrence?**
- **Combine Set A and Set C to form data sets with x% founder mutations**
- **E.g., for x=20, we take**
  - 10 from Set A and 40 from Set C & label them as "cases"
  - 50 from Set C & label them as "controls"

Low Occurrence

Avg SSE	10%	20%	30%	40%	50%	Standard deviation of SSE over 5 different %	Avg SSE over 5 different %
Blade	0.41200	0.42290	0.02427	0.02025	0.00691	0.21938	0.17727
HapMiner	0.11264	0.02765	0.13234	**0.00380**	0.01647	0.05936	0.05858
HapMiner (*Modified*)	0.28143	0.21786	0.06756	0.24051	0.05967	0.10282	0.17341
LinkageTracker	**0.01860**	0.02751	0.04065	0.01047	**0.00035**	0.01549	**0.01952**
GeneRecon	0.03386	**0.016987**	**0.01810**	0.02246	0.01255	**0.00811**	0.02079

	Avg time over 5 different %	Avg time with Linkage Tracker as base unit
Blade	1m 11.47s	0.74
HapMiner	**2.57s**	**0.03**
LinkageTracker	1m 36.66s	1
GeneRecon	2hrs 54m 32.23s	108.33

- **LinkageTracker is consistent, accurate & fast**

Noisy Data

- **Can we find the founder mutation under conditions of high confounding noise?**
- **Use Set B to generate confounding noise**

Mutation level	Data type	Set-A	Set-B	Set-C	Total
10%	Disease set	5/63	All 31	14/92	50
	Control set	-	-	50/(92-14)	50
20%	Disease set	10/63	All 31	9/92	50
	Control set	-	-	50/(92-9)	50
30%	Disease set	15/63	All 31	4/92	50
	Control set	-	-	50/(92-4)	50
40%	Disease set	20/63	30/31	-	50
	Control set	-	-	50/92	50
50%	Disease set	25/63	25/31	-	50
	Control set	-	-	50/92	50

Noisy Data

Avg SSE	10%	20%	30%	40%	50%	Stan-dard deviation of SSE over 5 different %	Avg SSE over 5 different %
Blade	0.12414	0.13140	0.18466	0.10704	0.13875	0.02902	0.13720
HapMiner	0.42124	**0.00010**	**0.00010**	**0.00010**	**0.00010**	0.18833	0.08433
HapMiner (*Modified*)	0.04986	0.23604	0.05109	0.05604	0.03199	0.08492	0.08501
Link-ageTracker	**0.00627**	0.01580	0.01004	0.00232	0.00619	**0.00501**	**0.00835**
GeneRecon	0.02467	0.01305	0.01078	0.02759	0.02283	0.00742	0.01979

	Avg time over 5 different %	Avg time with Linkage Tracker as base unit
Blade	47.85s	0.31
HapMiner	**1.53s**	**0.01**
LinkageTracker	2m 33.29s	1
GeneRecon	1hr 21m 18.63s	31.83

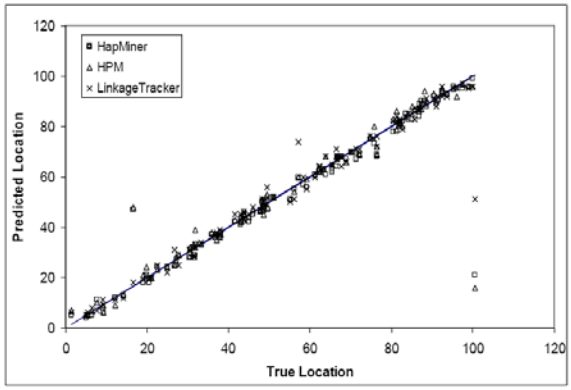
- **LinkageTracker is consistent, accurate, & fast**

Simulated Data Sets [Toivonen, 2000]

- **100 data sets**
- **Each data set consists of**
  - 200 seq labeled “abnormal”
  - 200 seq labeled “normal”
  - Each seq consists of 101 markers
- **Each dataset has a diff disease gene location**
- **The main task is to predict the marker that nearest to the disease gene for each dataset**

Simulated Data Sets

- **LinkageTracker is accurate & consistent. It is least affected by outliers**



	Avg SSE over 100 datasets	Avg SSE over 99 data-sets (after removing the dataset causing the outlier)
HPM	86.71	15.47
HapMiner	76.91	13.85
LinkageTracker	34.30	9.90

Conclusions

Method	Accuracy	Consistency	Speed
BLADE	x		xx
GeneRecon	xx	xx	
HapMiner	xxx	xx	xxx
Linkage Tracker	xxx	xxx	xx

- **LinkageTracker is consistently accurate under extreme conditions of low occurrence & high noise**
- **It is fast enough for data sets of small/medium size**

Acknowledgements

- **Tag SNP Selection**
  - Guimei Liu
  - Yue Wang

- **Disease Gene Inference**
  - Li Lin
  - Coral Lai
  - Tze Yun Leong