Whole population, genome-wide mapping of hidden relatedness

Authors: Alexander Gusev, Jennifer K. Lowe, Markus Stoffel, Mark J. Daly, David Altshuler, Jan L. Breslow, Jeffrey M. Friedman, and Itsik Pe'er

Presenters: Seth Temple, Manisha Ramaseshan, Cameron Haas

Methods

"A pair of haplotypes (i, i`) is defined to be sharing a segment in a SNP region [j … j`] if their included SNPs are identical and length L(j,j`) exceeds" a hyperparameter L*.

1. Split haplotypes by SNPs matrix H into smaller matrices with same # of columns
2. Determine matching haplotypes for each matrix (Algorithm 1)
3. Extend matrices (Algorithm 2)
4. Combine matching matrices
5. Prune matching matrix according to length rule

Under reasonable assumptions, algorithm has linear time complexity in expectation

```
In [3]: def germline_match(H):
            M = {}
            D = {}
            for i in range(len(
                try:
                    D[germline_
                except KeyError
                    D[germline_

            for k in D.keys():
                for i in range(
                    for j in r
                        M[(i,j
            return M
```
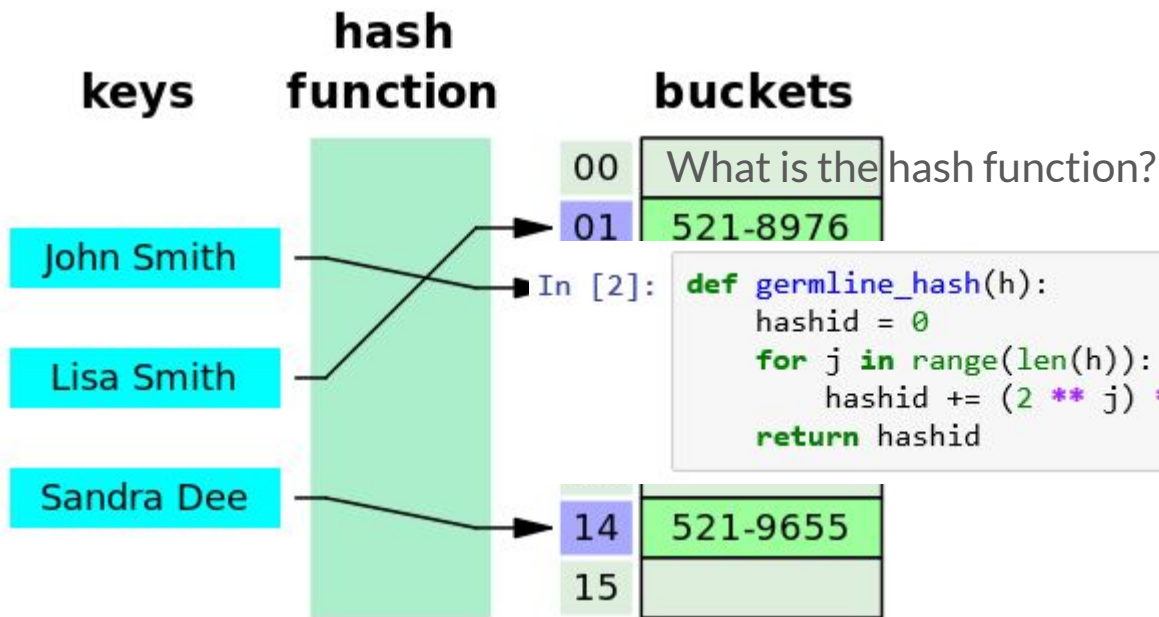
```
In [4]: H1 = [[0,0,0,0],[0,0,0,
        H2 = [[0,0,0,0],[0,0,0,
```

```
In [5]: germline_match(H1)
```

Out

(2, 5): [0, 0, 0, 0]}

```
In [6]: germline_match(H2)
```

Out[6]: {(0, 1): [0, 0, 0, 0]}

keys   **hash function**   **buckets**

00	What is the hash function?
01	521-8976

John Smith

Lisa Smith

```
In [2]: def germline_hash(h):
            hashid = 0
            for j in range(len(h)):
                hashid += (2 ** j) * h[j]
            return hashid
```

Sandra Dee

14	521-9655
15	

In computing, a **hash table** (**hash map**) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an *index*, also called a *hash code*, into an array of *buckets* or *slots*, from which the desired value can be found.

```
In [13]: def germline_extend(m1,m2):
             M = deepcopy(m2)
             for k in m2.keys():
                 if k in m1.keys():
                     M[k] = m2[k] + m1[k]
             return M
```

```
In [14]: H21 = [[0,0],[0,0],[0,0],[0,0]]
         H22 = [[0,0],[0,0],[1,0],[0,1]]
```

```
In [15]: H21m = germline_match(H21)
         H21m
```

Out[15]: {(0, 1): [0, 0],
          (0, 2): [0, 0],
          (0, 3): [0, 0],
          (1, 2): [0, 0],
          (1, 3): [0, 0],
          (2, 3): [0, 0]}

```
In [16]: H22m = germline_match(H22)
         H22m
```

Out[16]: {(0, 1): [0, 0]}

```
In [17]: germline_extend(H21m,H22m)
```

Out[17]: {(0, 1): [0, 0, 0, 0]}

```
In [18]: germline_extend(H22m,H21m)
```

Out[18]: {(0, 1): [0, 0, 0, 0],
          (0, 2): [0, 0],
          (0, 3): [0, 0],
          (1, 2): [0, 0],
          (1, 3): [0, 0],
          (2, 3): [0, 0]}