Administrativa Week 12

- Assignment 1 marks have been published -> Blackboard
  - check your filled-out marking rubric
- Assignment 2 – currently marked
- **Self-Reflection Survey**
  - This is an integral part of the assignment submission
    => if you have not done so, please enter those today
  - We reserve the right to not mark a submission without self-reflection survey..
- Today's tutorials will concentrate **Assignment 2 demo**

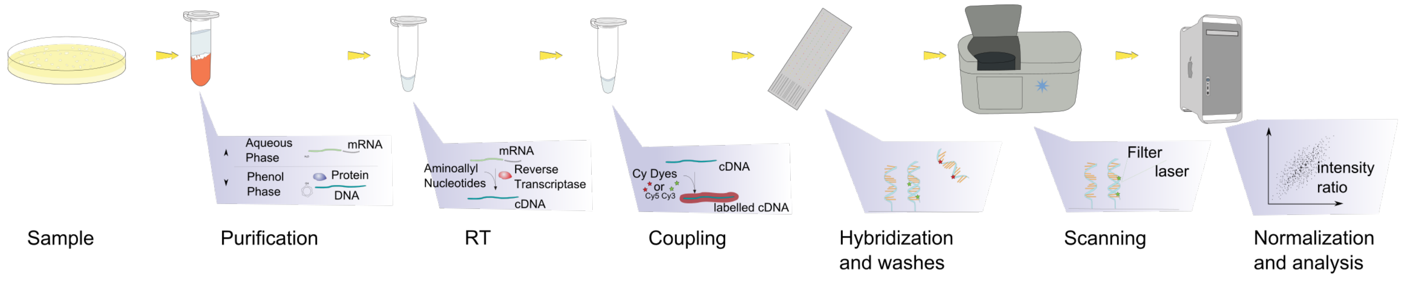
- Assignment 3 has been published since Tuesday in Blackboard

Assignment 3 Submission

- groupwork; worth 15%
- Use either Spark or Flink, depending on how assigned by Bilal (cf. announcement in Piazza)
- Deadline: **Tuesday, 13 June**
  - strictly speaking: Friday, 9 June, but without late penalty till 13 June
- Submission details
  - source code
  - documentation
  - measured execution times on small and large data set
  - *no demo this time*

Assignment 3: Data Mining with Apache Flink/Spark

- Scenario: Gene Expression Study
  - cell samples taken from cohort of patients suffering different diseases
  - analysed with regard to which genes are 'active' in these cells
  - method: gene profiling using DNA microarrays



[Source: Wikipedia]

Assignment 3 Data

- Source Data
  - GEO.txt
    pre-processed gene expression data for each sample
    patientid, geneid, expression value
  - PatientMetaData.txt
    meta-data about each patient
    patientid, age, gender, postcode, diseases, drug_response
  - GeneMedatData.txt
    meta-data about each gene, such as location on genome or which other gene is targeted by the produced protein; not used in this assignment

Note: patients can have multiple diseases

Assignment 3 Tasks

- **Task 1: Explorative Data Analysis**
  "Number of cancer patients with certain active genes per cancer type"

- **Task 2: Frequent Itemset Mining** (Apriori Algorithm)
  Iterative algorithm to identify gene combinations which are often active together (active gene -> for the purpose of this assignment, simply genes with an expression value above a given threshold)

- **Task 3: Association Rule Generation**
  Based on the frequent itemset, create rules *some genes* => *other genes* which hold in the given data set with some confidence

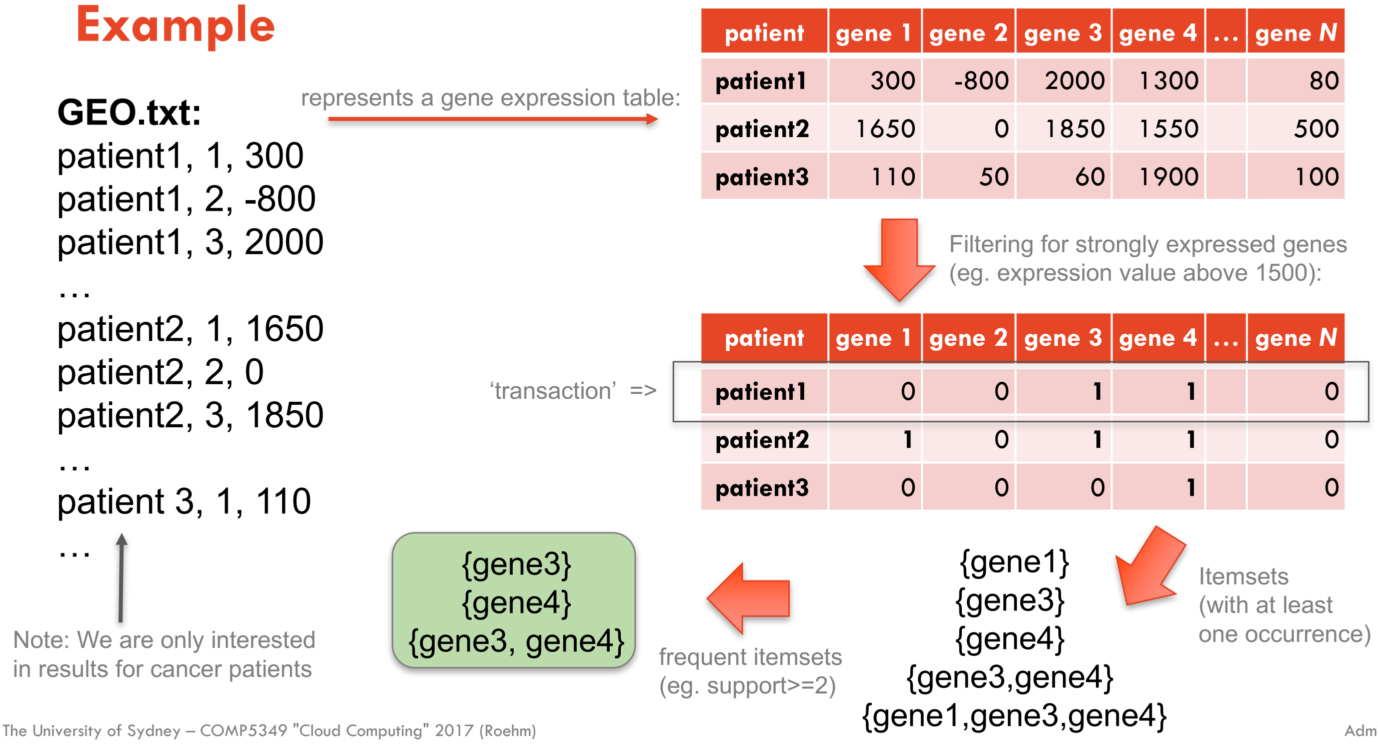
Assignment 3: Apriori Algorithm

- Apriori finds frequent itemset among transactional data set
  - "Which items are frequently bought together?"
- In this scenario:
  - transactions = gene expressions
  - items = (strongly) expressed genes
  - itemset = set of strongly expressed genes

patient	gene 1	gene 2	gene 3	gene 4	…	gene *N*
patient1	300	-800	2000	1300		2500
patient2	1650	0	1850	1550		500

- What is frequent?
  - any itemset with a minimum *support* (= count of occurrences)

Example



GEO.txt:
patient1, 1, 300
patient1, 2, -800
patient1, 3, 2000
…
patient2, 1, 1650
patient2, 2, 0
patient2, 3, 1850
…
patient 3, 1, 110
…

Note: We are only interested in results for cancer patients

represents a gene expression table:

patient	gene 1	gene 2	gene 3	gene 4	…	gene *N*
patient1	300	-800	2000	1300		80
patient2	1650	0	1850	1550		500
patient3	110	50	60	1900		100

Filtering for strongly expressed genes (eg. expression value above 1500):

‘transaction’ =>

patient	gene 1	gene 2	gene 3	gene 4	…	gene *N*
patient1	0	0	1	1		0
patient2	1	0	1	1		0
patient3	0	0	0	1		0

Itemsets (with at least one occurrence)

{gene1}
{gene3}
{gene4}
{gene3,gene4}
{gene1,gene3,gene4}

frequent itemsets (eg. support>=2)

{gene3}
{gene4}
{gene3, gene4}

Assignment 3: Apriori Algorithm (cont'd)

- Apriori is an efficient algorithm to find frequent itemset without the need to brute-force generation of all possible itemsets
- Input:
  - 'transactions' = set of strongly expressed genes per patient sample
  - support threshold (eg. if support threshold of 30%, then minimum support count must be > 0.3 * number of transactions)
- Step 1: Identify frequent 1-itemsets
  - Single genes which have a support count greater than threshold
- Step 2: Iterate
  - in each iteration, try extend current itemsets of size *k* to extend with any of the frequent 1-itsemsets; if support > threshold, keep as *k+1*-itemset
- Stop iterating if either no new *k+1*-itemset found, or after max number of iterations

Assignment 3: Task 3 – Association Rule Generation

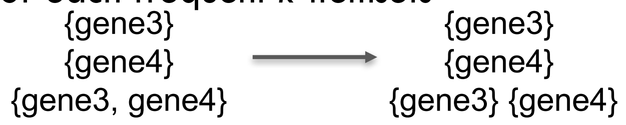
- Input:
  - Frequent itemset from Task 2
  - confidence threshold (eg. At least 60% confidence in rules)
- Step 1: Generate all possible subsets of each frequent itemsets
- Step 2: For each subset *R* of a frequent itemset *S* check:
  - Calculate confidence of rule R => (S – R):
    confidence(R => (S – R)) = support(S) / support(R)
  - If confidence value is above given threshold, then keep in result

Example:

- Input:
  - Frequent *k*-itemset from Task 2:
  - confidence threshold: 80%

{gene3}
{gene4}
{gene3, gene4}

- Step 1: Generate all possible subsets of each frequent *k*-itemsets


{gene3}
{gene4}
{gene3, gene4}

{gene3}
{gene4}
{gene3} {gene4}

- Step 2: For each subset *R* of a frequent itemset *S* check:
  - We can ignore the frequent 1-itemsets
  - Only rules to check hence:

{gene3} => {gene4}	confidence: 2/2 = 100%
{gene4} => {gene3}	confidence: 2/3 = 66%

Above confidence threshold of 80%

  - Calculate confidence of rules:
    support({gene3}) = 2 support({gene4}) = 3 support({gene3,gene4}) = 2