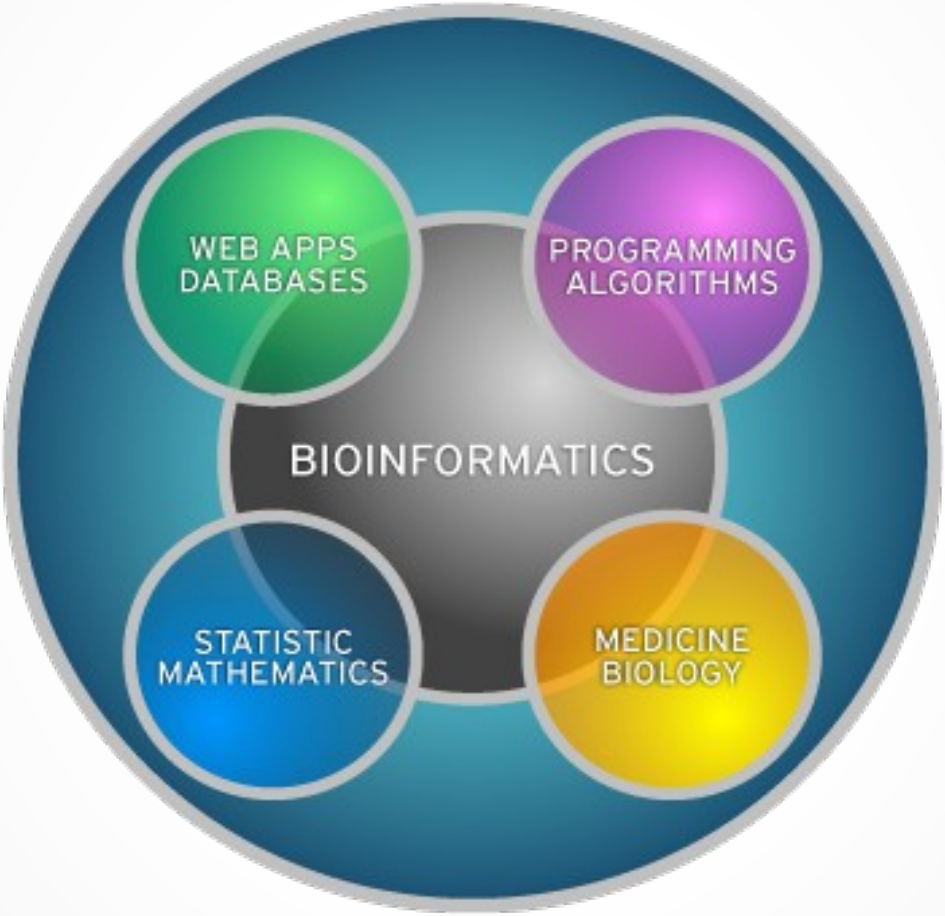
SNP based literature and data retrieval



Werner Veldsman's MSc project

Supervised by Prof. Alan Christoffels

Introduction

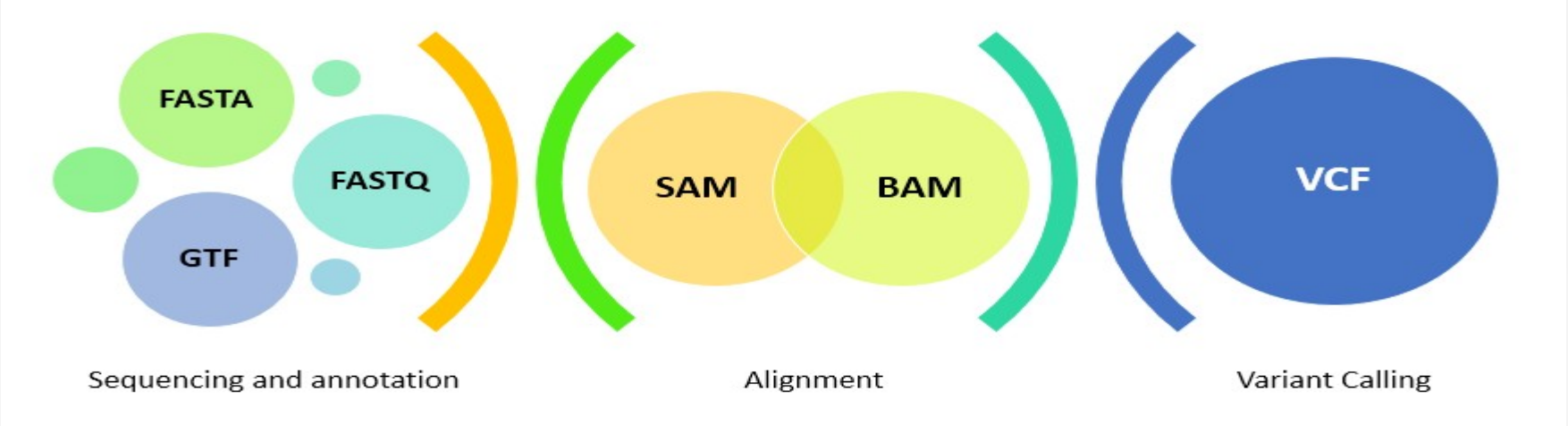
- First citation of digital data storage traced to 1945
- Memex portmanteau = memory + index
- Twenty years later the Cambridge Structural Database (CSD) was created
- CSD served as inspiration for the Protein Data Bank (PDB)
- “Bioinformatics” first used in the 1970's

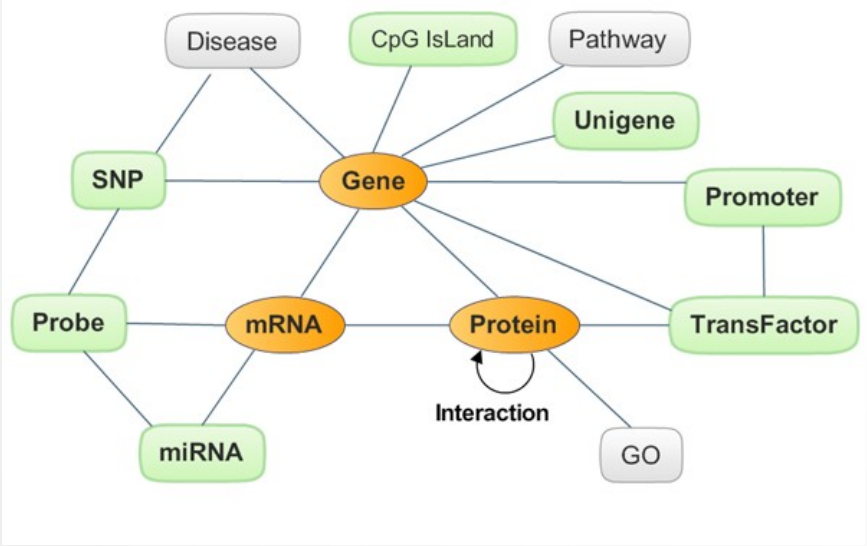
  (aka computational science or genomic data science)

Introduction

- 1980: Rate limiting step shifted from sequencing to information management. Still relevant today
- 2000: Information overload attributed to lack of standardization and coordination rather than volume (Goble & Stevens, 2008)
- Semantic web technology could be a solution
- Abstraction is required for black box medicine

Observation





Structured

Unstructured

Observation

Open Access Initiatives (OAIs)

- Open Access (OA) literature definition:

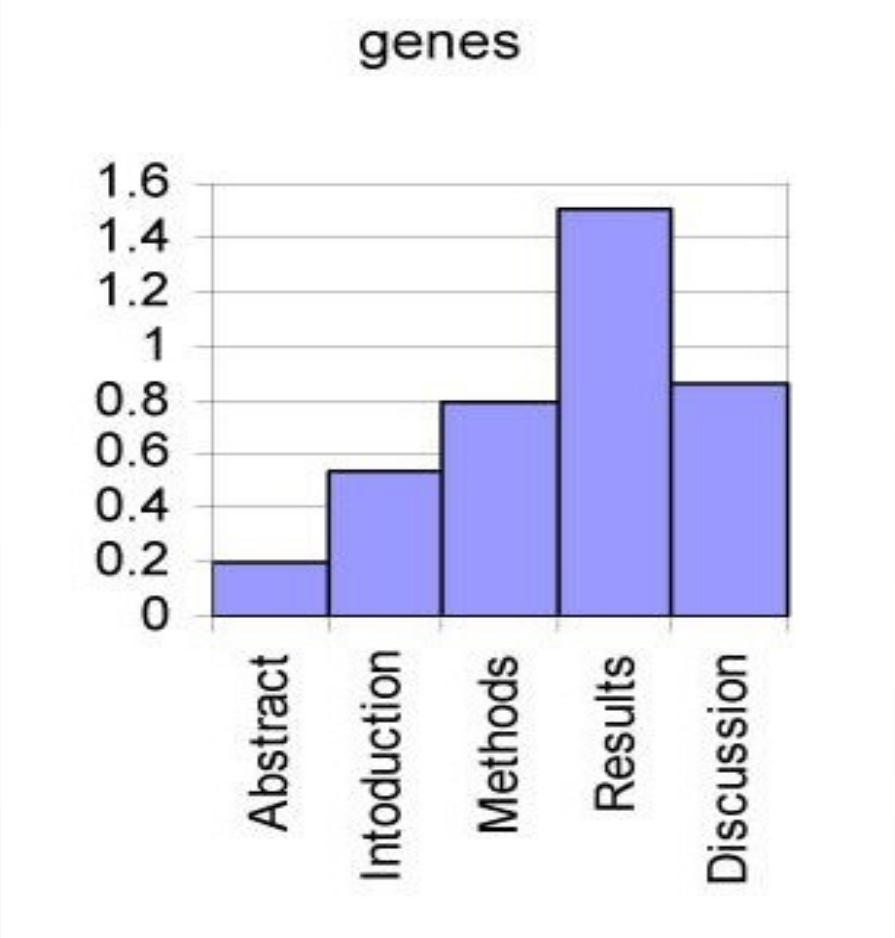
  The literature that should be freely accessible online is that which scholars give to the world without expectation of payment...

  - Budapest Open Access Initiative (2002)

- Yesterday PMC-OAI contained 1109273 full articles that can be downloaded in XML or text format

Observation

The value of OAIs



Shah *et al.*, 2003

Hypothesis

- *Problem statement*: Obtaining post variant call information from unstructured data is a time consuming process
- *Research question*: Can automated supplementary information retrieval from literature resources using SNP ID's sliced from VCF files benefit science?
- *Working* hypothesis: SNP specific automated literature retrieval will increase research efficiency and scope
- *Expected outcome*: A valuable user friendly biological portal

Benefits and risks

- Benefits:
  - Unique application
  - Low cost (no lab work)
  - Quick access to quality literature relating to SNPs
- Risks:
  - Finding the right software libraries is time consuming
  - Literature resources will become outdated
  - Ongoing updates to system can cause anomalies

Methods

Pre-implementation survey

- *Determine the needs and experiences of biological database users*
- *Temporal, behavioural and spatial parameters in nine observational dimensions (*Reeves *et al.*, 2008*)*
- *Use a google forms template*
- *Post-implementation survey will triangulate results*

Methods

Pre-implementation survey question rubric

		Space	Actor	Activity	Object	Act	Event	Time	Goal	Feeling
How often do you use biological databases?	1	x				x		x		
When querying a database do you?	2					x		x		
What type of information do you collect?	3				x				x	
Do you often have to traverse multiple databases to obtain information on a single biological feature?	4	x		x					x	
Do you know which databases to query for the information you are looking for?	5	x			x				x	
Do you collect data from databases using APIs?	6					x				
How would you like to to learn more about any given database?	7		x				x			x
Have you ever heard of the Pubmed open access initiative?	8	x			x					
Have you ever heard of the Lynx biological database?	9	x			x					
Do you use VCF files in your research?	10				x	x				

Methods

Pubmed OAI corpus:

- Retrieved from http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/ in XML format:

	Total Files scanned	Total size (Mb)
A-B	216411	14600
C-H	187489	12500
I-N	312477	17100
O-Z (I)	189782	17100
O-Z (II)	81199	5000
	987358	66300

Methods

What to use?

```
#Determining genomic variation ID types for chromosome 22 at the 1000 genomes project
idList = open("idnumbers22.txt", "r")
rsCount = 0
esvCount = 0
otherCount = 0
for line in idList:
    if line.startswith("rs"):
        rsCount += 1
    elif line.startswith("esv"):
        esvCount += 1
    else:
        otherCount +=1
rsRatio = str(float(round((rsCount/(rsCount + esvCount \
          + otherCount)*100), 2))) + "%"
esvRatio = str(float(round((esvCount/(rsCount + esvCount \
          + otherCount)*100), 2))) + "%"
otherRatio = str(float(round((otherCount/(rsCount + esvCount \
          + otherCount)*100), 2))) + "%"
print("rs entries (dbSNP): " + rsRatio)
print("esv entries (DGVa): " + esvRatio)
print("Other entries: " + otherRatio)
```

```
rs entries (dbSNP): 99.92%
esv entries (DGVa): 0.08%
Other entries: 0.0%
```

Methods

Extracting rs ID containing articles from Pubmed corpus:

```
#Extracting "rs" containing XML files from Pubmed OAI set.
"""import os
import sys
import shutil
directory = "/home/werner/Desktop/Source/articles.0-Z (Part II)/" #example subset
filelisting = os.walk(directory)
totalFiles = 0
rsFiles = 0
for root, dirs, files in filelisting:
    for file in files:
        totalFiles += 1
        breakTest = 0
        fileone = open(root + "/" + file)
        if breakTest == 1:
            break
        for line in fileone:
            line1 = line.split()
            if breakTest == 1:
                break
            for word in line1:
                if (word.startswith('rs')): # search clause
                    shutil.copy(os.path.join(root,file), "/home/werner/Desktop/Destination/" + file)
                    breakTest = 1
                    rsFiles += 1
                if breakTest == 1:
                    break
        fileone.close()
print (totalFiles, rsFiles)"""
```

Methods

Reduced Pubmed OAI corpus:

- Using rs ID containing files reduces corpus size significantly:

	Total Files scanned	Total size (Mb)	Containing rs	rs size (Mb)
A-B	216411	14600	3391	311
C-H	187489	12500	2894	325
I-N	312477	17100	3084	338
O-Z (I)	189782	17100	5102	526
O-Z (II)	81199	5000	601	100
	987358	66300	15072	1600

Methods

Creating a JSON database from scratch:

```
#Create flat file database
import xml.etree.ElementTree as ET
import os
directory = "/home/werner/Desktop/Destination/"
fileListing = os.walk(directory)
rslist =[]
for root, dirs, files in fileListing:
    for file in files:
        email = "not available" #reset
        pmid = "not available" #reset
        year = "not available" #reset
        doi = "not available" #reset
        tree = ET.parse(root + "/" + file)
        for node in tree.iter('email'):
            email = node.text
        for node in tree.iter('pub-date'):
            for subnode in node.iter('year'):
                collection = node.attrib
                if "collection" in collection.values():
                    year = subnode.text
        for node in tree.iter('article-id'):
            pubdat = node.attrib
            if "pmid" in pubdat.values():
                pmid = node.text
            if "doi" in pubdat.values():
                doi = node.text
        for node in tree.iter():
            if node.text != None:
                node = node.text.split()
                for rsnumber in node:
                    if len(rsnumber) > 2 and len(rsnumber) < 15: #ensure that digits follow rs
                        if rsnumber.startswith("rs") and rsnumber[2].isdigit():
                            while not rsnumber[-1].isdigit():
                                rsnumber = rsnumber[:-1] #ensure that rs ends with digits
                            rslist.append(rsnumber.strip() + "\t" + email.strip() + "\t" + year
rsdict = {}
for item in rslist:
    if rsdict.get(item,"empty") == "empty":
        rsdict.update({item:1})
    if rsdict.get(item,"empty") != "empty":
        rsdict[item] += 1
rslist = []
writetofile = open("/home/werner/Desktop/TEST1db.csv", "w")
for item in rsdict.keys():
    writetofile.write(item + "\t" + str(rsdict[item]) + "\n")
writetofile.close()
```

Parsing with xml.etree.ElementTree:

- rs ID
- Pubmed ID
- Email address
- Digital Object Identifier (DOI)
- Article file name
- Number of rs occurrences

```
#Convert database to JSON
import json
file = open("/home/werner/Desktop/Test1b.csv", "r")
pre2JSONlist = []
JSONstring = "{\"PMCDAI rs articles\": ["
for line in file:
    lister = line.strip().split("\t")
    rs number = "\"rs number\": " + "\"" + lister[0] + "\""
    email address = "\"email address\": " + "\"" + lister[1] + "\""
    publication date = "\"publication date\": " + "\"" + lister[2] + "\""
    pubmed id = "\"pubmed id\": " + "\"" + lister[3] + "\""
    doi = "\"doi\": " + "\"" + lister[4] + "\""
    pubmed file name = "\"pubmed file name\": " + "\"" + lister[5] + "\""
    rs number cited in article = "\"rs number cited in article\": " + "\"" + lister[6] + "\""
    newdictionary = "{" + rs number + ", " + email address + ", " + publication date + ", " +
    ", " + doi + ", " + pubmed file name + ", " + rs number cited in article + "}"
    JSONstring += newdictionary + ", "
file.close()
with open("/home/werner/Desktop/JSONdbtest.txt", "w") as file1:
    JSONstring = JSONstring[:-2]
    JSONstring += "]}"
    file1.write(JSONstring)
```

Convert data from XML to JSON

Methods

Reduced Pubmed OAI corpus:

- Extracting relevant XML tags reduces corpus size significantly:

	Total Files scanned	Total size (Mb)	Containing rs	rs size (Mb)
A-B	216411	14600	3391	311
C-H	187489	12500	2894	325
I-N	312477	17100	3084	338
O-Z (I)	189782	17100	5102	526
O-Z (II)	81199	5000	601	100
	987358	66300	15072	1600

26 MB

Run in memory?

Methods

Extracting rs ID's from VCF files using PyVCF 0.6.7:

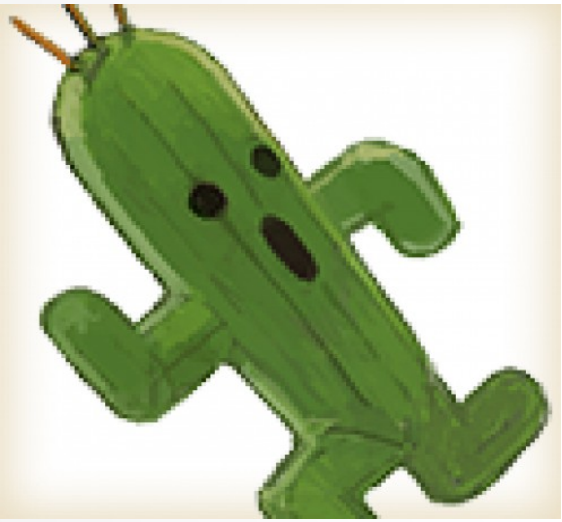
```
#Extracting ID references from 1000 genomes VCF file for chromosome Y
import vcf
vcf_reader = vcf.Reader(open('ALL.chrY.phase3_integrated_v1a.20130502.genotypes.vcf', 'r'))
vcf_writer = open('idnumbers.txt', 'a')
for record in vcf_reader:
    if record.ID != None:
        vcf_writer.write(record.ID + "\n")
vcf_writer.close()
```

- *Extraction is simple, but takes time*

Methods

Serving the data:

- Use node.js JSON-server (with API) by Typicode:



This is typicode's
Github profile icon

- **Dependencies**: yargs, update-notifier, underscore-db, pluralize, node-uuid, morgan, method-override, lowdb, lodash, got, express, errorhandler, cors, connect-pause, chalk, body-parser

Methods

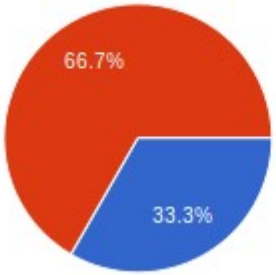
Client side (HTML, CSS, Bootstrap, JS and jQuery):

- Create a web interface with search and upload functionality
- Interactivity handled by AJAX calls to JSON database
- jQuery statements update DOM dynamically
- Use Bootstrap library to implement a minimalistic design (also have a look at Google's Material Design Language)

Preliminary results

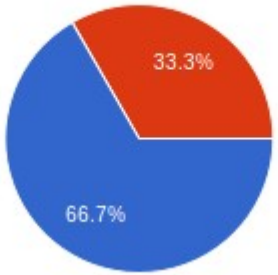
Pre-implementation questionnaire

Do you collect data from databases using APIs?



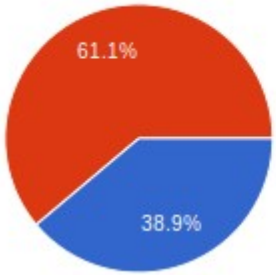
Yes	**6**	33.3%
No	**12**	66.7%

Do you often have to traverse multiple databases to obtain information on a single biological feature?



Yes	**12**	66.7%
No	**6**	33.3%

When querying a database do you?



Prefer to be supplied with information in real time (less informative)	**7**	38.9%
Not mind waiting for a processing completion email from the database (more informative)	**11**	61.1%

Preliminary results

- Start web server, database server and API server:

  > `node index.js --watch JSONdb.json`

- Serving a website and API

- Website takes single and multiple rs queries

- API:

  > `http://localhost:3000/PMCOAI_rs_articles?rs_number=rs34014629`

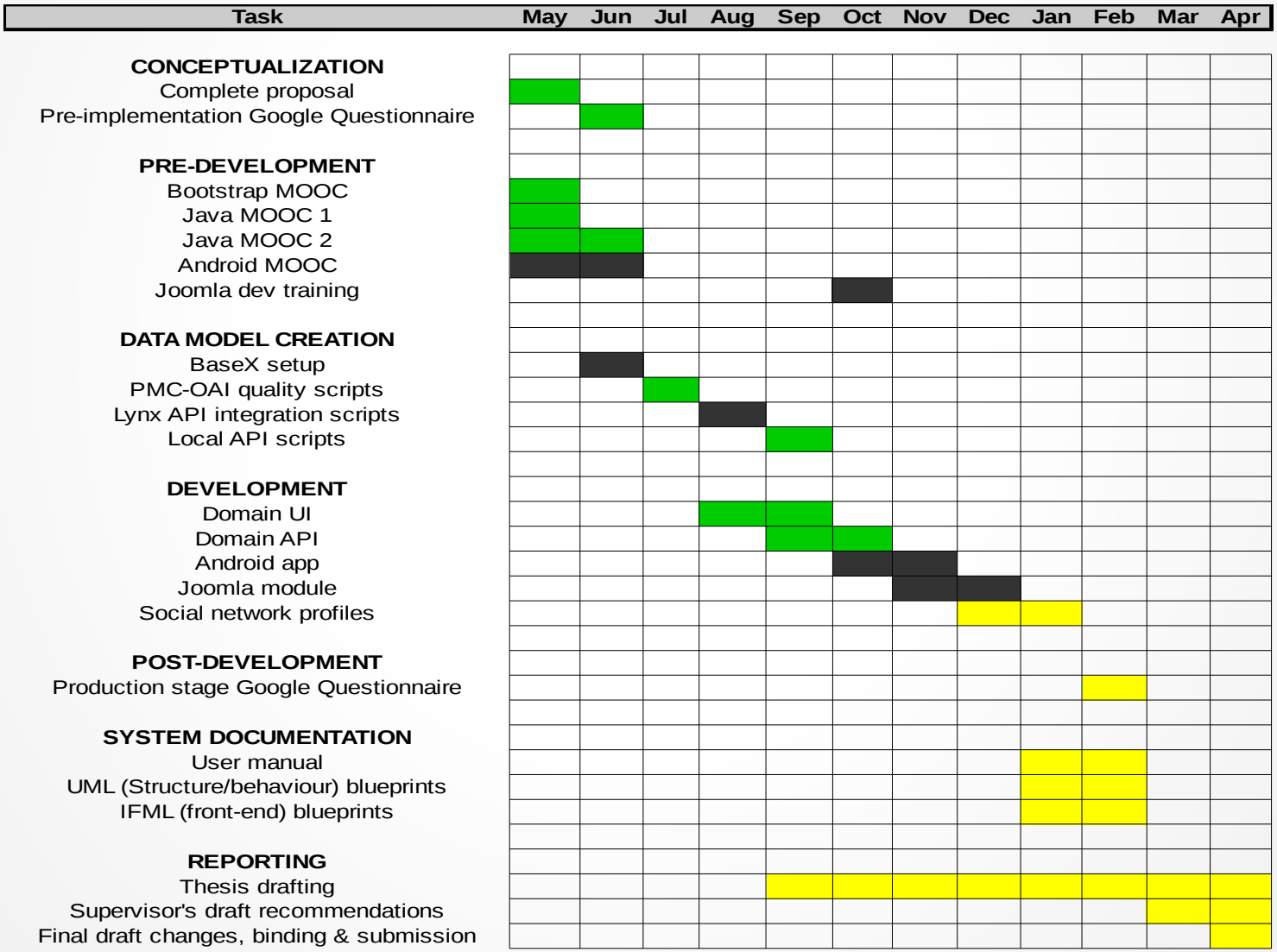
SNPhunter

What's next?

- Semantic web integration (RDFa, JSON-LD, Ontology stores)
- Create galaxy work flow for VCF input with wrapper or iFrame
- Attend Coursera ML & BD MOOCs starting on 15 September
- Finish literature review (Chapter 2 of thesis) by mid October
- Move SNPhunter to production phase by end October

Progress status

Ahead of original schedule



Acknowledgements

- Funding: NRF and MRC
- Inspiration: Prof. Alan Christoffels
- Online resources: Contributors to Github, Stack Overflow etc.
- MOOC providers: UCSD, HKUST, MIT, Microsoft
- SANBI's bioinformatics course lecturers and organizers

Questions