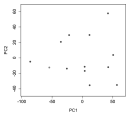
Differentiation of human populations using SNP genotyping data

2019-09-27

Kohji

Machine learning



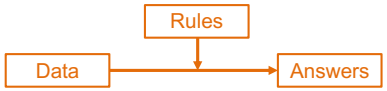
Unsupervised learning

- Hierarchical clustering analysis
- Principal component analysis

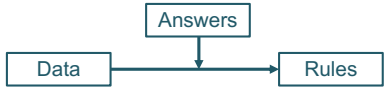
Supervised learning

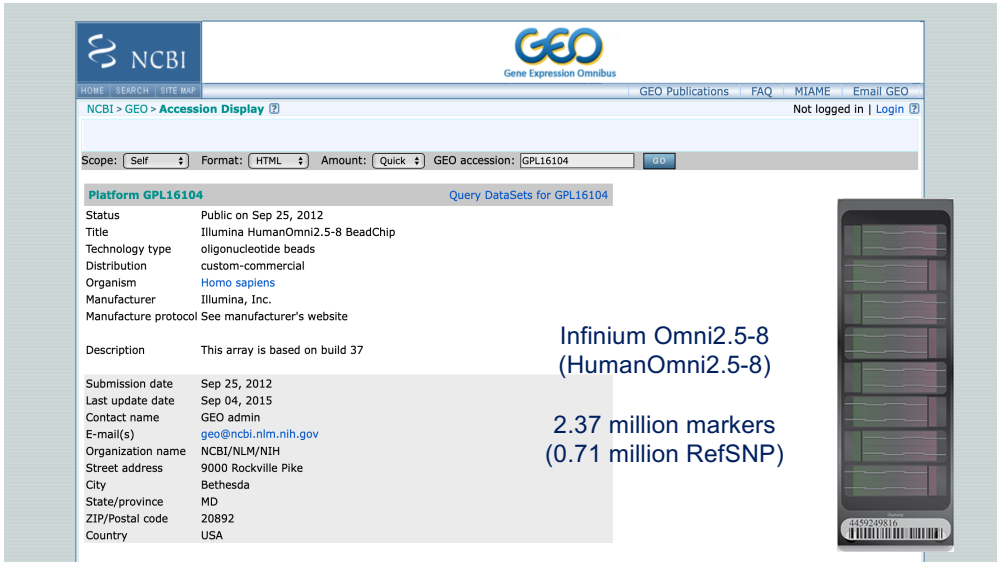
- Linear classification
- Deep learning

Classical programming



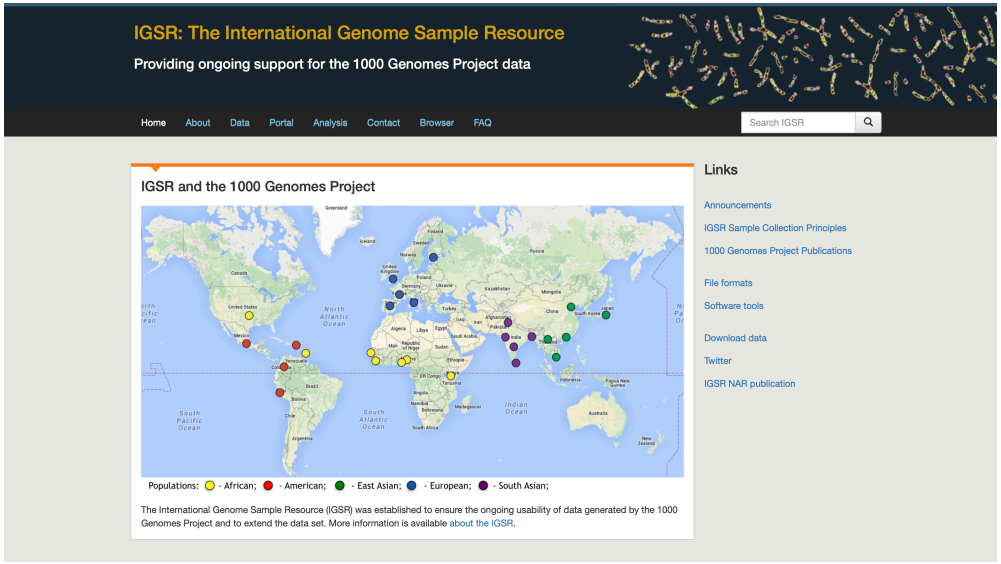
Artificial intelligence





Infinium Omni2.5-8
(HumanOmni2.5-8)

2.37 million markers
(0.71 million RefSNP)



Vector data for machine learning

ID	rs7699689	rs13124826	rs10010286	rs10034550	rs11946825	
NA19223_YRI	1.0	0.0	0.0	1.0	1.0	...
NA18975_JPT	0.0	1.0	0.0	0.0	0.0	...
NA12154_CEU	1.0	0.0	0.0	1.0	0.0	...
NA19130_YRI	0.0	0.5	0.0	0.0	0.0	...
NA19118_YRI	0.5	0.0	0.5	0.5	0.5	...
NA18592_CHB	0.0	0.5	0.0	1.0	0.5	...
NA18981_JPT	0.0	1.0	0.0	0.0	0.5	...
NA12005_CEU	1.0	0.0	0.0	1.0	0.0	...
NA18924_YRI	0.0	0.0	0.0	0.0	1.0	...
NA12045_CEU	0.5	0.5	0.0	0.0	0.0	...
NA18988_JPT	0.0	1.0	0.0	0.0	1.0	...
...	...	...	...	...	...	...

Number of individuals (414 samples in total)
108 Africans (YRI), 99 Europeans (CEU), 103 Chinese (CHB), 104 Japanese (JPT)

Selected 4231 SNPs on human chromosome 4

Infinium Omni2.5-8

2.37 million markers

↓

0.71 million RefSNP

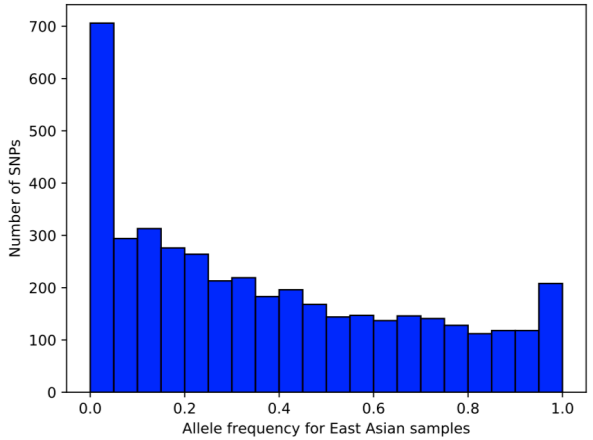
↓

38,738 SNPs (chr4)

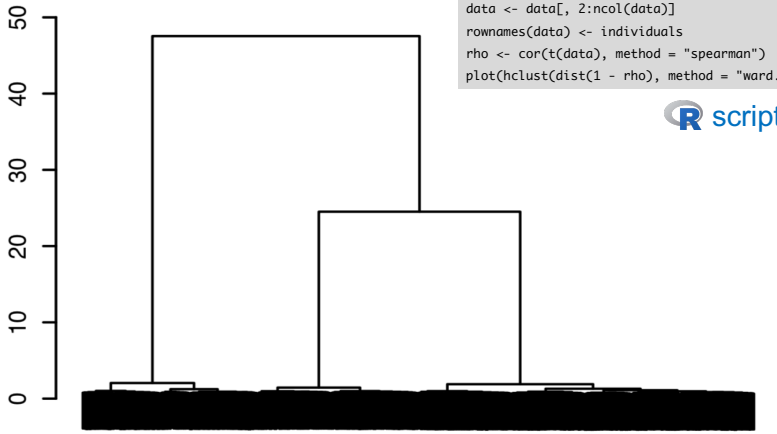
↓

4231 SNPs

(RefSNP multiples of 9)



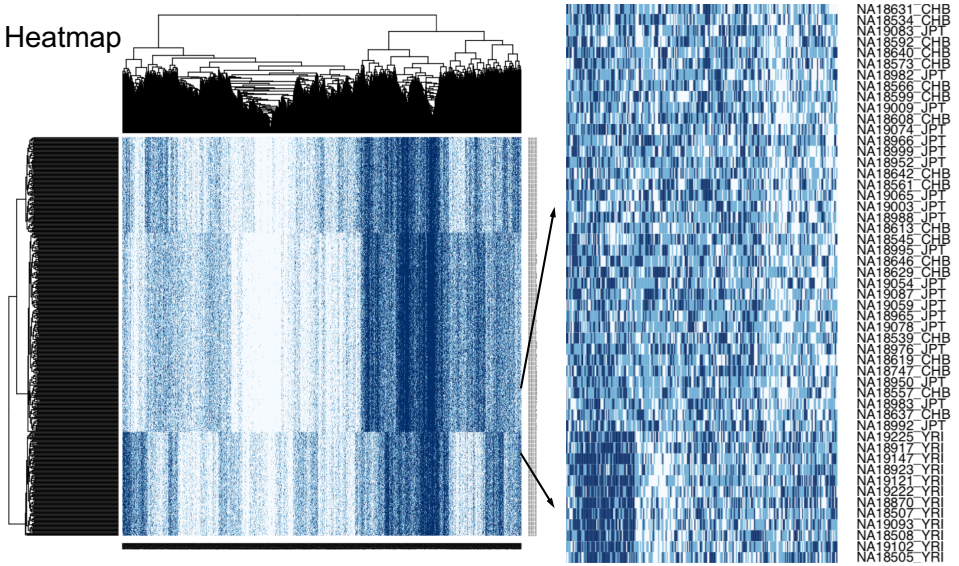
Hierarchical clustering analysis



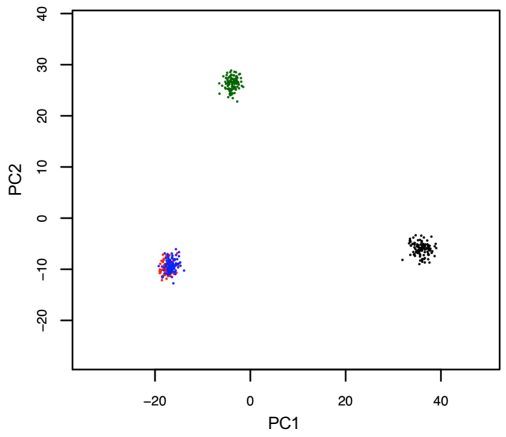

```
data <- read.table("data.tsv", header = TRUE)
individuals <- as.vector(data[,1])
data <- data[, 2:ncol(data)]
rownames(data) <- individuals
rho <- cor(t(data), method = "spearman")
plot(hclust(dist(1 - rho), method = "ward.D2"))
```

R script

Heatmap



Principal component analysis



108 Africans (YRI)

99 Europeans (CEU)

103 Chinese (CHB)

104 Japanese (JPT)

Deep learning

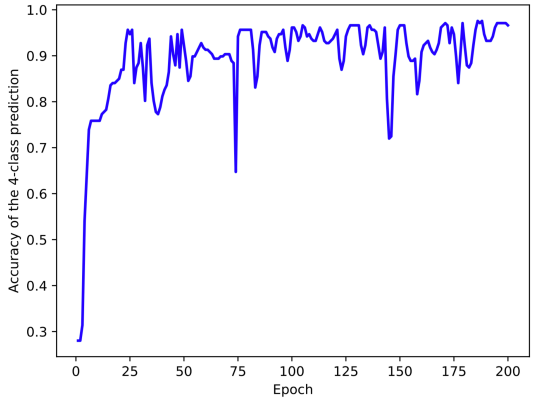
ID	rs7699689	...
NA19223_YRI	1.0	...
NA18975_JPT	0.0	...
NA12154_CEU	1.0	...
NA19130_YRI	0.0	...
NA19118_YRI	0.5	...
NA18592_CHB	0.0	...
NA18981_JPT	0.0	...
NA12005_CEU	1.0	...
NA18924_YRI	0.0	...
NA12045_CEU	0.5	...
NA18988_JPT	0.0	...
...	...	...

Matrix of input training data
414 samples × 4231 SNPs
(rows) (columns)

```
training = numpy.array(data.iloc[id, :], dtype = float)
    else:
      training = numpy.append(training, numpy.array(data.iloc[id, :], dtype = float))
    i += 1
training = numpy.reshape(training, (number_of_data, -1))

model = Sequential()
model.add(Dense(units = 4000, activation = 'sigmoid', input_dim = 4231))
model.add(BatchNormalization())
model.add(Dense(units = 5000, activation = 'sigmoid'))
model.add(Dense(units = 3000, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.08))
model.add(Dense(units = 3500, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(Dense(units = 2000, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization())
model.add(Dense(units = 400, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(Dense(units = 100, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization())
model.add(Dense(units = 40, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(Dropout(0.04))
model.add(Dense(units = 10, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(BatchNormalization())
model.add(Dense(units = 6, kernel_initializer = 'he_normal', activation = 'relu'))
model.add(Dense(units = 4, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = SGD(lr = 0.01,
        decay = 1e-6, momentum = 0.9, nesterov = True), metrics = ['accuracy'])
print(model.summary)
model.fit(training, teacher, epochs = number_of_epochs, validation_split = 0.2,
        batch_size = number_of_batch_size)

predicted = model.predict(testing)

with open(output_results, 'a') as output_result:
```

?

Change in accuracy by repetitive training



Linux

ANACONDA

TensorFlow

Keras

python

Results of leave-one-out cross validation

NA19143	YRI	0.99958	0.00003	0.00025	0.00013	correct
NA19159	YRI	0.99916	0.00045	0.00037	0.00002	correct
NA19184	YRI	0.99873	0.00127	0.00000	0.00000	correct
NA19189	YRI	0.99865	0.00076	0.00034	0.00025	correct
NA11933	CEU	0.00010	0.99957	0.00022	0.00012	correct
NA11992	CEU	0.00005	0.99995	0.00000	0.00001	correct
NA11995	CEU	0.00089	0.99084	0.00099	0.00728	correct
NA12003	CEU	0.00185	0.99757	0.00045	0.00013	correct
NA18570	CHB	0.00013	0.00028	0.99931	0.00029	correct
NA18574	CHB	0.00006	0.00107	0.97022	0.02864	correct
NA18592	CHB	0.00021	0.00003	0.00472	0.99504	wrong
NA18596	CHB	0.00146	0.00563	0.95536	0.03754	correct
NA18959	JPT	0.00012	0.00006	0.00027	0.99956	correct
NA18964	JPT	0.00001	0.00000	0.00051	0.99947	correct
NA18971	JPT	0.00638	0.01263	0.04557	0.93542	correct
NA18977	JPT	0.00058	0.00010	0.99803	0.00129	wrong

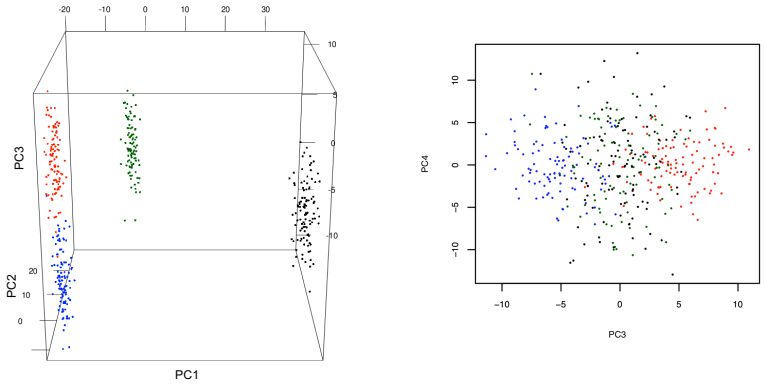
Accuracies of some classifications

	Deep learning		Linear classification
	4-class	2-class	2-class
Africans (YRI)	108 / 108 = 100%	N/a	N/a
Europeans (CEU)	99 / 99 = 100%	N/a	N/a
Chinese (CHB)	90 / 103 = 87.3%	91 / 103 = 88.3%	102 / 103 = 99.0%
Japanese (JPT)	90 / 104 = 86.5%	92 / 104 = 88.5%	103 / 104 = 97.1%

SNPs differentiated the two populations

Principal component analysis		Linear classification	
rs4693005	6.696675e-02	rs4865148	0.0217184
rs6853554	5.412930e-02	rs4693005	0.0211772
rs12499677	5.264641e-02	rs4699395	0.0201767
rs4956372	5.110434e-02	rs7695738	0.0188612
rs7695738	4.934608e-02	rs2390022	0.0183215
rs1874565	-4.706829e-02	rs4861584	-0.0172726
rs9997830	-4.809371e-02	rs4352481	-0.0182920
rs243981	-4.856684e-02	rs2663289	-0.0184255
rs2090025	-4.912147e-02	rs1874565	-0.0194939
rs10212894	-5.279501e-02	rs12643884	-0.0209435

PC3 and PC4



Summary

Following data processing by yourself, I would like you to feel

- the difference between classical programing and AI
- importance of quantity of data for deep learning
- usefulness of linear classification for small data set
- that AI does not always provide correct answer