베이지안 통계학 중간고사 과제물

- 유방암(Breast invasive carcinoma) 환자의 암세포 내 유전자 발현(mRNA)빈도에 따른 stage 예측 분석

목차

6 조
이정오 : 보고서 작성 및 EDA
이진권 : 보고서 작성 및 EDA
지용기 : 자료 수집 및 전처리, EDA

자료선정

본 과제에서 선택한 자료는 NCI(National Cancer Institute) GDC(Genomic Data Commons) portal 의 TCGA(The Cancer Genome Atlas) dataset 이며, 그 중에서도 유방암(Breast invasive carcinoma)에 관한 부분이다.

관련링크: https://portal.gdc.cancer.gov/projects/TCGA-BRCA

이러한 데이터를 선정한 이유는

1. 해당자료가 신뢰성 있는 기관에서 공개된 자료
2. 연구가 매우 활발히 이루어지는 분야
3. 조원들의 공통 관심분야

이기 때문이었다.

앞서 말한 유방암 데이터에서, mRNA 정보를 바탕으로, 암의 stage 를 예측하는 것을 본 과제의 목표로 정하였다. 따라서 직접적인 단백질 발현에 영향을 미치는 mRNA 의 발현량과, 유방암의 임상적 진행정도인 stage 만을 분석에 사용하였다.

본래 데이터는 다양한 정보(raw sequencing data, single nucleotide variation, DNA methylation 등)를 포함하고 있었기 때문에, 원하는 자료만을 선택하기 위하여, 전처리 과정을 거쳤다.

전처리 과정을 거친 자료는 약 1100 개의 case 와 6 만개 정도의 mRNA 변수를 포함하고 있으며, 수집 및 전처리 과정은 뒤의 "자료 수집"부분에서 설명하였다.

분석 계획

분석 계획은 다음과 같다.

1. 자료수집 및 전처리
   - gdc portal 에서 받은 데이터를 사용하기 쉬운 형태로 변형
   - python 사용

2. 탐색적 데이터 분석(Exploratory Data Analysis)
   - 변수들의 분포 및 도수 파악
   - 선형 관계를 갖는 변수의 유무 및 빈도 파악

3. 차원 축소 (중간 과제 이후)
   - 변수가 6 만개에 달하므로, 모든 변수를 사용하여 회귀분석을 실시하기는 어려울 것으로 생각됨
   - Principal component analysis, correlation analysis, factor analysis 등을 사용하여 변수 개수를 축소

4. 베이지안 회귀 분석 (중간 과제 이후)
   - 축소된 mRNA 변수와 stage(i, ii, iii, iv, 4 종류)에 관한 베이지안 회귀분석 실시
   - 일반 회귀 혹은 deep learning model 과의 비교 시도

자료 수집

1. TCGA-BRCA(유방암) 데이터의 기초 정보 다운로드 및 stage 추출

https://portal.gdc.cancer.gov/projects

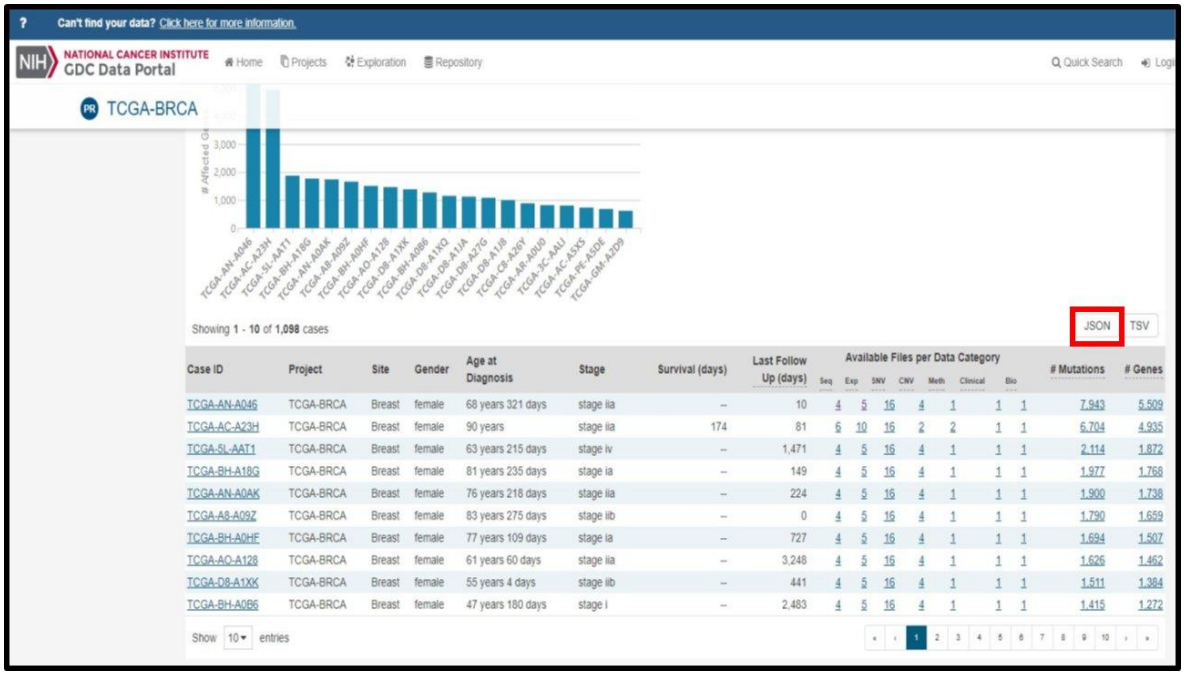
유방암(Breast Invasive Carcinom) 환자의 mRNA 발현데이터는 case 가 1098 개가 있다.

Project	Disease Type	Primary Site	Program	Cases	Seq	Exp
FM-AD	▾ 23 Disease Types	▾ 42 Primary Sites	FM	18,004	0	0
TARGET-NBL	Neuroblastoma	Nervous System	TARGET	1,127	270	151
TCGA-BRCA	Breast Invasive Carcinoma	Breast	TCGA	1,098	1,098	1,097
TARGET-AML	Acute Myeloid Leukemia	Blood	TARGET	988	299	272
TARGET-WT	High-Risk Wilms Tumor	Kidney	TARGET	652	128	128
TCGA-GBM	Glioblastoma Multiforme	Brain	TCGA	617	406	166

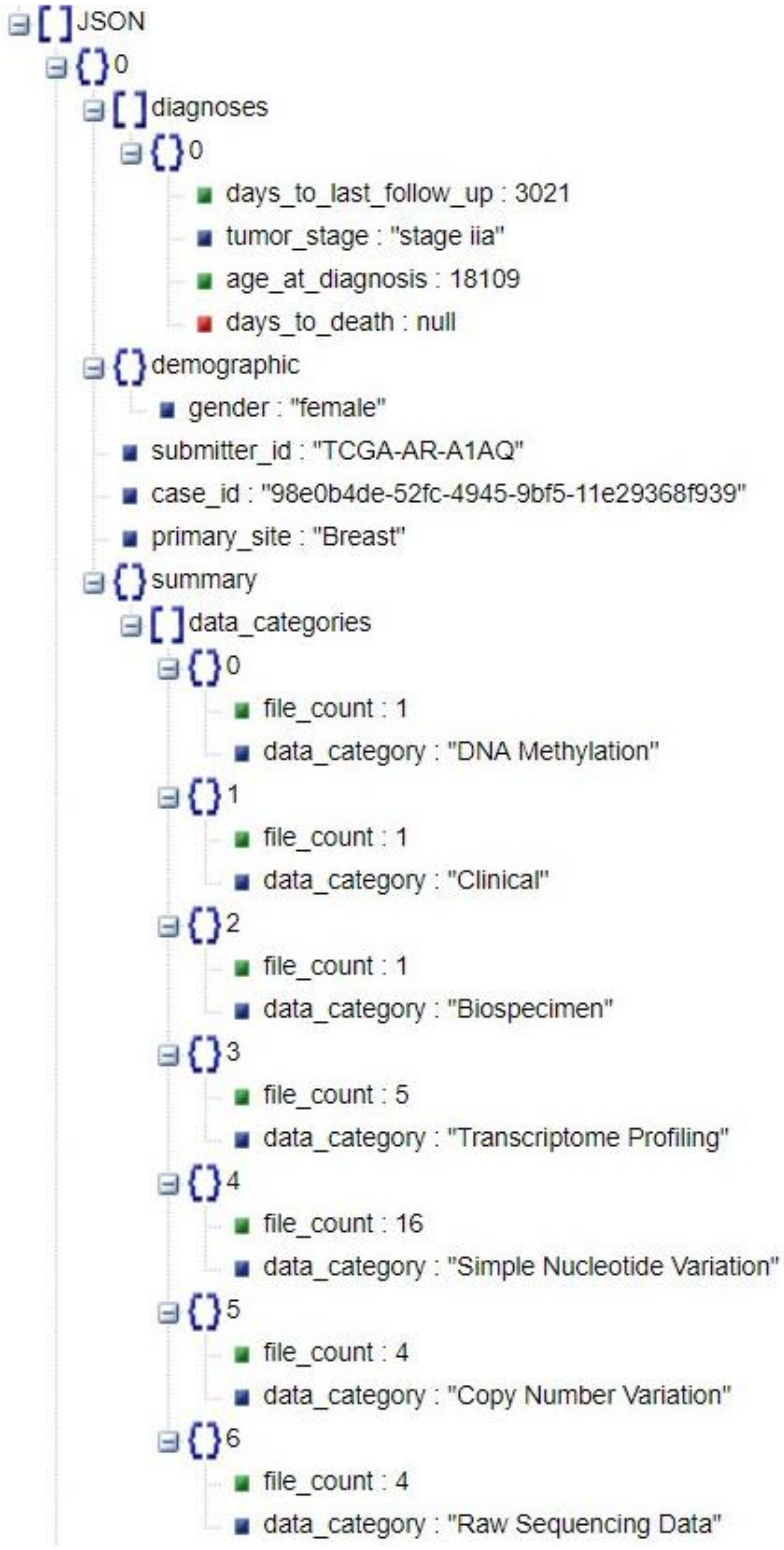
https://portal.gdc.cancer.gov/projects/TCGA-BRCA 페이지로 들어가면 유방암 환자에 대한 데이터현황을 볼 수 있으며, JSON 버튼을 클릭하면, JSON 포맷의 파일이 다운로드 할 수 있다.

파일명 : cases.2017-10-25T09_05_38.765714.json

이 파일에는 유방암 환자의 정보(성별, 생존여부, 나이, 암진행단계)와 어떤 종류의 유전자 데이터가 있는지에 대한 정보가 포함되어있다.



cases.2017-10-25T09_05_38.765714.json 파일을 JSON Viewer 로 보면 다음과 같다.



이 cases.2017-10-25T09_05_38.765714.json 파일에서는 case_id와 tumor_stage 정보를 추출하여 사용하였다. 데이터를 추출하기 위해서 python 언어로 추출 프로그램 작성하였다. 코드는 다음과 같다.

```
import json
import pprint
json_data = open('cases.2017-10-25T09_05_38.765714.json').read()
data = json.loads(json_data)
caseCnt = len( data )
print "CASE count=" + str( caseCnt )
not_diagnoses_cnt = 0
not_reported_cnt = 0
stage_count = {}
stage_dic = { 'stage_1' : [], 'stage_2' : [], 'stage_3' : [] , 'stage_4' : []      }
for idx in range( caseCnt ) :
    if data[ idx ].get('diagnoses'):
        diagnoses = data[ idx ]['diagnoses']
        tumor_stage = diagnoses[0]['tumor_stage']
        case_id = data[ idx ]['case_id']
        if tumor_stage == 'not reported' :
            not_reported_cnt = 1 +   not_reported_cnt
        else :
            if tumor_stage in ('stage i', 'stage ia', 'stage ib' ):
                stage_dic['stage_1'].append( case_id )
            elif tumor_stage in ('stage ii', 'stage iia', 'stage iib' ):
                stage_dic['stage_2'].append( case_id )
            elif tumor_stage in ('stage iii', 'stage iiia', 'stage iiib', 'stage iiic' ):
                stage_dic['stage_3'].append( case_id )
            elif tumor_stage in ('stage iv' ):
                stage_dic['stage_4'].append( case_id )

            if tumor_stage in stage_count   :
                stage_count[ tumor_stage   ] = 1 + stage_count[ tumor_stage   ]
            else :
                stage_count[ tumor_stage   ] = 1

        #print "idx=%s, diagnoses count=%s" % ( str(idx),   str( len( diagnoses ) )   )
    else :
        not_diagnoses_cnt = 1 +   not_diagnoses_cnt
```

```
print "not_diagnoses_cnt=" + str( not_diagnoses_cnt )
print "not_reported_cnt=" + str( not_reported_cnt )
print stage_count
print "stage_1_list cnt=" + str( len(stage_dic['stage_1']) )
print "stage_2_list cnt=" + str( len(stage_dic['stage_2']) )
print "stage_3_list cnt=" + str( len(stage_dic['stage_3']) )
print "stage_4_list cnt=" + str( len(stage_dic['stage_4']) )
```

결과로 얻은 stage 정보는 다음과 같다.

Tumor Stage 별 개수

Tumor Stage	개수	합계
진단 정보가 없는 경우	1	24
not reported	11	
stage x	13	
stage i	90	183
stage ia	86	
stage ib	7	
stage ii	6	621
stage iia	358	
stage iib	257	
stage iii	2	249
stage iiia	155	
stage iiic	65	
stage iiib	27	
stage iv	20	20

2. CASE(환자)에 대한 유전체 정보파일 탐색

https://portal.gdc.cancer.gov/query 페이지에서 유방암환자의 mRNA 발현 데이터를 조회할 수 있으며, Query 입력란에 *cases.project.project_id in ["TCGA-BRCA"] and files.data_category in ["Transcriptome Profiling"]* 라고 입력한 후, "Submit Query" 버튼을 클릭하면 파일정보가 조회된다. 해당 화면에서 JSON 버튼 클릭하여 files.2017-10-25T10_54_46.644542.json 파일을 다운로드한다.

이 JSON 파일에는 case_id 와 유전자 데이터 파일명이 포함되어 있으며, mRNA 발

현량 데이터 파일은 명칭 뒷부분이 FPKM-UQ.txt.gz 로 끝난다. (정규화된 mRNA 발현량을 담고 있는 파일이다.)

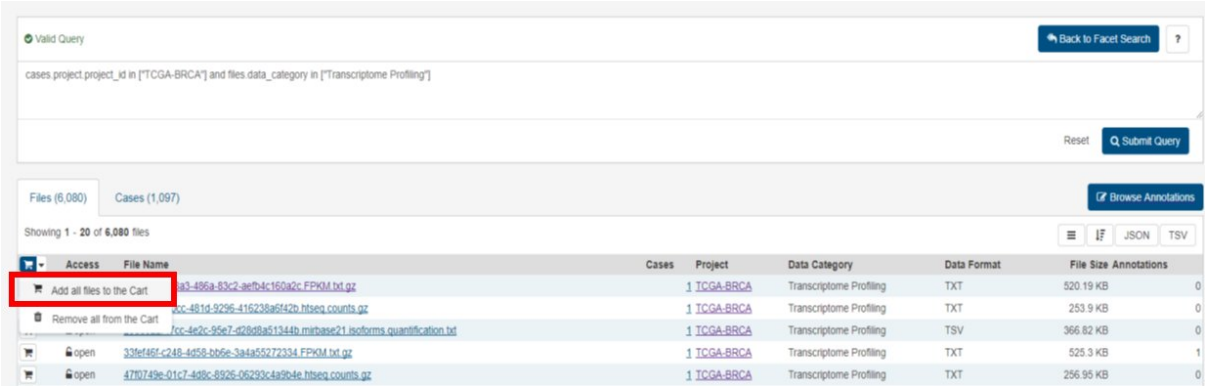
files.2017-10-25T10_54_46.644542.json 에서 각각의 stage 별로 mRNA 발현량 데이터 파일명을 추출하였다. Python 코드는 다음과 같다.

```
with open('files.2017-10-25T10_54_46.644542.json') as data_file:
    files_json = json.load(data_file)

cnt = 0
stage_filename = { 'stage_1' : [], 'stage_2' : [], 'stage_3' : [] , 'stage_4' : []    }
for file_info in files_json :
    file_name = file_info['file_name']
    case_id = file_info['cases'][0]['case_id']
    if 'FPKM-UQ.txt.gz' in file_name :
        for stageKey in stage_dic.keys():
            if case_id in stage_dic[ stageKey ] :
                stage_filename[ stageKey ].append( file_name )

print "stage_1_file cnt=" + str( len(stage_filename['stage_1']) )
print "stage_2_file cnt=" + str( len(stage_filename['stage_2']) )
print "stage_3_file cnt=" + str( len(stage_filename['stage_3']) )
print "stage_4_file cnt=" + str( len(stage_filename['stage_4']) )
```

3. CASE(환자)에 대한 유전체파일 다운로드

앞의 "2. CASE(환자)에 대한 유전체 정보파일 탐색" 결과에서 "Add all files to the

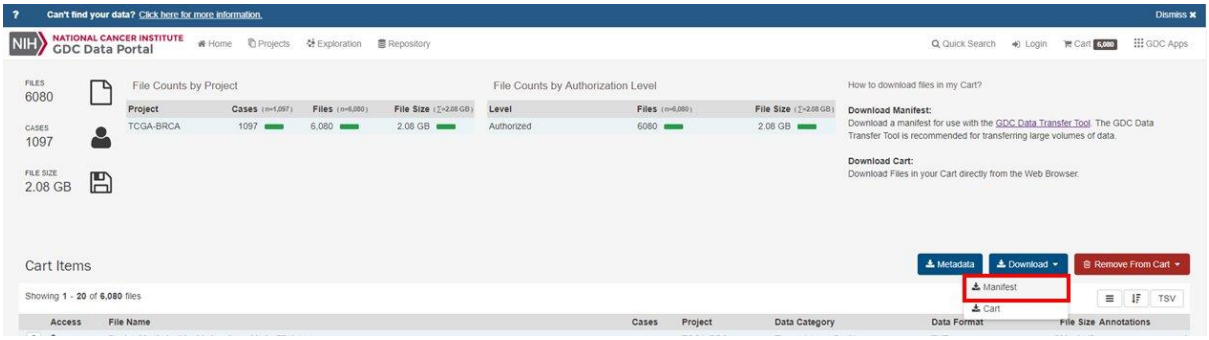
Char" 버튼을 클릭한다.



버튼을 클릭하면, 파일을 다운로드할 수 있는 페이지로 이동하고, Download 버튼을 클릭하면 다운로드가 되지만 시간이 많이 소요되고, 하루 이상이 걸리면 실패가 발생한다.

따라서 많은 데이터를 받기 위해서는 Manifest 과 전용 다운로드 도구를 이용한다.

Manifest 파일(gdc_manifest_20171025_122019.txt)은 GDC Data Portal 에서 제공하는 전용 다운로드 도구를 이용할 때 사용되는 정보 파일이다.



다운 받은 gdc_manifest_20171025_122019.txt 안에는 유전체파일명과 유전체파일의 UUID 가 포함되어 있다. 다운로드 예정 파일을 전용 다운로드 도구에 알려주는 역할을 한다.

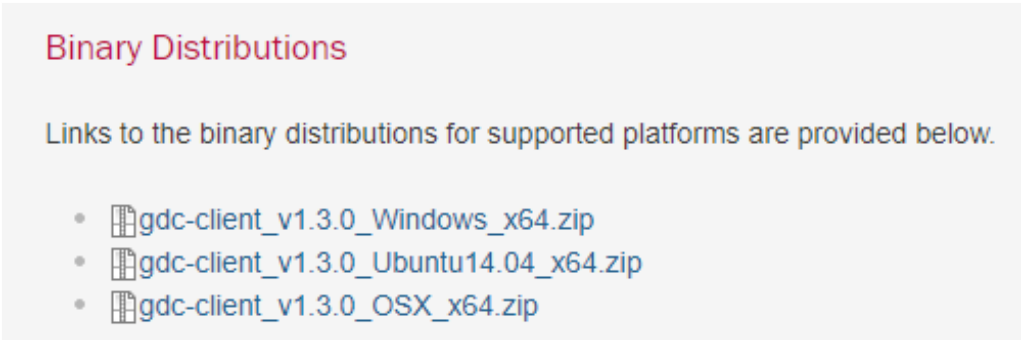
아래 코드는 manifest 파일에서 UUID 를 추출하는 코드이며, 이는 추후에 mRNA 정보와 stage 정보를 합칠 때에 사용하였다.

```
gdc_manifest = {}
with open('gdc_manifest_20171025_122019.txt','r') as gdc :
    cnt = 0
    head_skip = True
    for line in gdc:
```

```
        cnt += 1
        if( head_skip ) :
            head_skip = False
            continue

        #print( line )
        rows = line.strip().split('₩t')
        uuid = str( rows[0] )
        file_name = rows[1]
        if cnt <= 10 :
            print "uuid=%s, filename=%s" %(uuid, file_name )

        gdc_manifest[ file_name ] = uuid
```

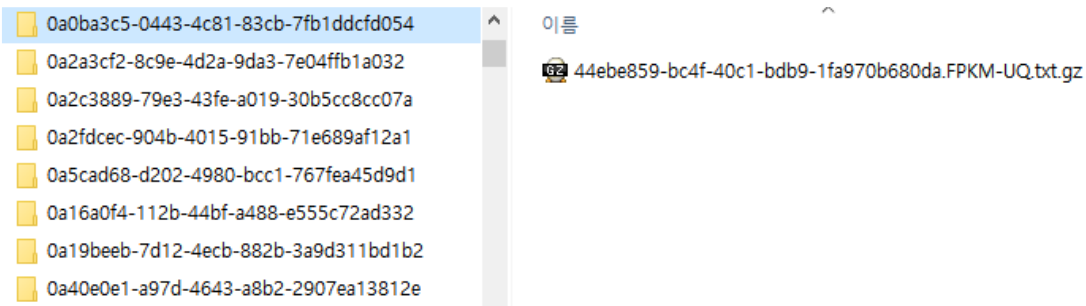
GDC Data Portal 에서 제공하는 전용 다운로드 도구는 https://gdc.cancer.gov/access-data/gdc-data-transfer-tool 페이지에서 받을 수 있으며, 윈도우용, 우분투 리눅스용, 맥용 3 가지가 있다.

본 과제에서는 윈도우용을 다운받아서 사용하였다.



이 전용도구를 이용해서 윈도우 명령창에서 아래의 명령어를 입력하면 다운로드가 진행되고, UUID 이름으로 폴더를 만들고 그 안에 파일이 생성된다.

```
>   gdc-client.exe download -m gdc_manifest_20171025_122019.txt
```



4. 다운로드 받은 mRNA 데이터를 분석이 가능한 포맷으로 변환

앞에서 받은 mRNA 데이터 파일에는 한 줄에 유전자명과 발현량 정보가 있다.

```
1  ENSG00000242268.2    0.0
2  ENSG00000270112.3    88.3086344385
3  ENSG00000167578.15   68292.0480165
4  ENSG00000273842.1    0.0
5  ENSG00000078237.5    77729.1256565
6  ENSG00000146083.10   202545.510867
7  ENSG00000225275.4    0.0
8  ENSG00000158486.12   483.289072109
```

이를 과정 1.에서 추출한 stage 정보와 합쳐, 분석 가능한 CSV 포맷으로 변환하였다.

아래의 python 코드를 이용하였으며, 앞의 단계의 추출한 Stage + CASE_ID, CASE_ID + mRNA 파일명, mRNA 파일명 + UUID 정보들을 종합하여, Stage 별 mRNA 파일을 읽어들인 후, CSV 파일로 변환하였다.

```
import gzip
import StringIO

def makeOneLine( stage , filename, makeHead ) :
    SEPARATOR = ','
    NEWLINE = '₩n'

    line = StringIO.StringIO()
    if makeHead == True :
        line.write( 'stage' )
    else :
        line.write( stage )
    with gzip.open(filename,'r') as f:
        for fpkm_line in f:
            rows = fpkm_line.strip('₩n').split('₩t')
            line.write( SEPARATOR )
            if makeHead == True :
                line.write( rows[0] )
```

```
                else :
                    line.write( rows[1] )

    line.write( NEWLINE )
    contents = line.getvalue()
    line.close()
    return contents
```

```
import shutil
import datetime
import os
import gzip
import StringIO

MRNA_CSV = 'mRNA_%s.csv' %( datetime.datetime.today().strftime('%Y%m%d%H%M') )
IS_FIRST = True
print MRNA_CSV
with open(MRNA_CSV, 'w') as f:
    for   stage in stage_filename.keys():
        file_list = stage_filename[ stage ]
        print stage
        for filename in file_list :
            uuid = gdc_manifest[ filename ]
            src_file = './data/%s/%s' %( uuid   , filename )
            dst_file = './%s/' %( stage   )
            if os.path.isfile(src_file):
                #shutil.copy( src_file   , dst_file )
                if True == IS_FIRST :
                    contents = makeOneLine( stage , src_file, True )
                    f.write( contents )
                    f.flush()
                    IS_FIRST = False

                contents = makeOneLine( stage , src_file, False )
                f.write( contents )
                f.flush()
            else :
```

```
print "no file : " + src_file
```

5. 분석가능한 mRNA CSV 파일

위의 결과로 mRNA_201710262036.csv 파일을 만들어졌다.

헤더와 2 번째 라인까지의 대략적인 정보는 다음과 같다.

HEADER LINE


```
stage,ENSG00000242268.2,ENSG00000270112.3,ENSG00000167578.15,ENS
G00000273842.1,ENSG00000078237.5,ENSG00000146083.10,ENSG00000225
275.4,ENSG00000158486.12,ENSG00000198242.12,ENSG00000259883.1,EN
SG00000231981.3,ENSG00000269475.2,ENSG00000201788.1,ENSG00000134
108.11,ENSG00000263089.1,ENSG00000172137.17,ENSG00000167700.7,EN
SG00000234943.2,ENSG00000240423.1,ENSG00000060642.9,ENSG00000271
616.1,ENSG00000234881.1,ENSG00000236040.1,ENSG00000231105.1,ENSG
00000243044.1,ENSG00000182141.8,ENSG00000269416.4,ENSG0000026498
1.1,ENSG00000275265.1,ENSG00000185105.4,ENSG00000233540.1,ENSG00
000102174.8,ENSG00000271647.1,ENSG00000166391.13,ENSG00000270469
.1,ENSG00000070087.12,ENSG00000262950.1,ENSG00000255420.1,ENSG00
000280038.1,ENSG00000266261.1,ENSG00000153561.11,ENSG00000269148
.1,ENSG00000274458.1,ENSG00000273406.1,ENSG00000179262.8,ENSG000
00214198.6,ENSG00000278099.1,ENSG00000166368.2,ENSG00000206072.1
1,ENSG00000234900.1,ENSG00000258630.1,ENSG00000127511.8,ENSG0000
0220993.1,ENSG00000225269.2,ENSG00000214 ...
```


1st LINE


```
stage_4,1379.03136837,191.344981735,86877.4583502,0.0,66073.0742
29,419726.632776,0.0,632.670585443,5926102.44898,846.616750143,0
.0,0.0,0.0,483988.849364,1067.31852383,375623.41989,203556.52820
3,0.0,1807.43299071,77394.8395205,0.0,0.0,984.032433709,13033.76
56155,0.0,58873.1753267,1386.28355598,0.0,2151.89510229,1991.684
84854,0.0,20653.7885183,0.0,2207.87175672,2414.58020109,662608.5
37252,484.22961006,0.0,97.4241066209,4572.62006557,268090.264827
,8116.55024401,0.0,1371.30570244,735367.022507,2625.79407102,0.0
,294.027709171,706.430210347,895.393023814,0.0,195294.430524,0.0
,187.031952539,450.788338647,2042.06951493,0.0,1208.78058215,707
693.96767,509440.758942,568326.516398,5342.65742656,0.0,0.0,0.0,
54316.7093881,279782.646872,53130.1232619,0.0,196.372296739,0.0,
475.759121254,27778.1109191,4421.05497757,0.0,0.0,8857.19334154,
0.0,0.0,0.0,0.0,0.0,540.707019848,0.0,365645.529522,0.0,44844.83
68644,0.0,0.0,0.0,56163.7594352,0.0,0.0,0.0,13054.8302872,0.0,0.
0,11674.8381789,805.522230803,0.0,0.0,0. ...
```


2nd LINE


```
stage_4,0.0,0.0,147167.536817,0.0,98540.7461541,212511.50162,0.0
,1950.17808205,4271222.44395,2655.44079072,0.0,0.0,0.0,1069617.4
5396,0.0,18032.9845829,434906.359312,0.0,404.933712351,203554.02
```

2973,0.0,0.0,0.0,1238.81293847,0.0,28297.6437048,3261.09237277,0
.0,2249.82950511,416.465591873,2585.03137582,8258.00906617,0.0,6
3067.5207369,0.0,582808.850005,0.0,0.0,0.0,3585.54264387,270135.
778773,1339.88537281,0.0,1075.28616053,734628.128472,2260.831984
15,0.0,0.0,0.0,0.0,2708.128108,124212.274403,0.0,293.315881038,0
.0,7472.51984751,0.0,0.0,638833.824542,95872.6440168,703076.9785
95,4115.85696368,0.0,0.0,0.0,24308.7357008,972062.036313,59515.8
386526,0.0,0.0,0.0,186.529231929,44116.6581355,83485.6245519,0.0
,1815.02202983,644.703805855,0.0,0.0,0.0,0.0,1218.6576486,0.0,0.
0,244088.930114,0.0,9605.0688716,3046.6441215,0.0,0.0,170708.968
189,0.0,0.0,0.0,6551.50351887,0.0,0.0,5760.21460914,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,7874.40326788,0.0,0.0,581.632059559,0.0,
0.0,24200.4342997,0.0,0.0,0.0,27317.4755 ...

EDA(탐색적 데이터 분석)

R 을 이용하여 앞에서 얻은 csv 파일에 대하여 탐색적 분석을 실시하였다.

다음 코드를 이용해서 자료를 불러오면, 총 1195 개의 행과, 60484 개의 열을 가지고 있음을 알 수 있다.

```
library(data.table)
big_data <- fread("~/downloads/mRNA_201710262036.csv")
```

```
big_data                1195 obs. of 60484 variables
```

Stage 에 해당하는 첫 번째 열을 살펴보면 다음과 같다.

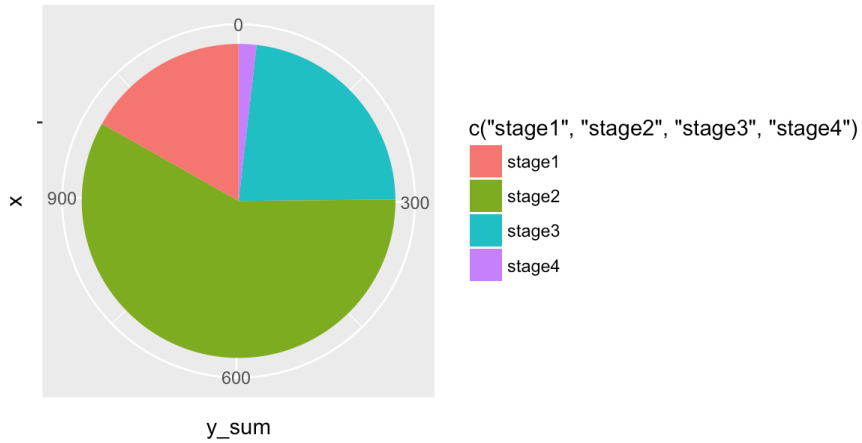
```
stage <- big_data[,1]
y <- as.factor(stage$stage)
summary(y)
```

```
> summary(y)
stage_1 stage_2 stage_3 stage_4
    201     697     275      22
```

이를 비율로 변형해보고, 도표로 그리면 다음과 같다.

```
round(y_sum/sum(y_sum), 2)
library(ggplot2)
bp<- ggplot( y_sum, aes( x="", y = y_sum, fill = c( "stage1", "stage2", "stage3", "stage4" ))) + geom_bar(width = 1, stat = "identity")
pie <- bp + coord_polar("y", start=0)
pie
```

```
stage_1 stage_2 stage_3 stage_4
   0.17    0.58    0.23    0.02
```



stage 4 는 거의 없고, 나머지 1,2,3 이 대부분을 차지함을 알 수 있다.

mRNA 의 발현량 해당하는 나머지 데이터는 전체를 살펴보기에는 너무 많으므로, 일부분을 살펴보며 탐색적 분석을 실시하였다.

```
mRNA <- big_data[,-1]
head(mRNA[,1:10],15)
```

```
> head(mRNA[,1:10],15)
   ENSG00000242268.2 ENSG00000270112.3 ENSG00000167578.15 ENSG00000273842.1
1          1379.0314         191.34498           86877.46                 0
2             0.0000           0.00000          147167.54                 0
3          1243.3906         517.57318          111668.29                 0
4             0.0000           0.00000           37754.13                 0
5             0.0000           0.00000           92800.19                 0
6          2957.3768           0.00000           44666.76                 0
7          1723.4025          59.78189           41413.97                 0
8             0.0000           0.00000          115353.39                 0
9             0.0000           0.00000          111928.45                 0
10            0.0000           0.00000           62395.45                 0
11            0.0000           0.00000           56397.79                 0
12          606.3068           0.00000           62652.73                 0
13          683.8799           0.00000          162673.79                 0
14            0.0000           0.00000           53852.62                 0
15            0.0000           0.00000           56669.73                 0
   ENSG00000078237.5 ENSG00000146083.10 ENSG00000225275.4 ENSG00000158486.12
1           66073.07          419726.63                 0          632.67059
2           98540.75          212511.50                 0         1950.17808
3          113891.78          201985.07                 0         2065.39147
4          100629.86          245029.72                 0          210.50203
5           23562.54           99051.22                 0          226.73019
6           90296.35          188086.03                 0          456.15956
7           75286.98          186623.03                 0          715.68437
8           52124.11          202234.87                 0         1274.67417
```

위의 결과에서 mRNA 의 발현량이 0 에 해당하는 값이 많음을 알 수 있다. 따라서 0 의 비율이 높은 mRNA 들은 얼마나 되는지 알아보기로 하였다.

다음과 같은 코드에서 ratio(0 값의 비율)를 바꿔가며, 0 의 비율이 높은 열의 개수는 얼마나 되는지 살펴보았다.

```
torf <- vector()

for(i in mRNA){
  zero_num <- 0
  ratio <- 0.9
  for(j in i){
    if(j ==0){
      zero_num <- zero_num + 1
    }
  }

  if(zero_num/length(i)>ratio){
    torf <- append(torf, FALSE)
  } else {
    torf <- append(torf, TRUE)
  }
}

length(torf) - sum(torf)
```

ratio = 1 인 때, 즉, 모든 발현값이 0 인 mRNA 는 2214 개 이다.
ratio = 0.95 인 때, 발현값이 0 인 경우가 95% 인 mRNA 의 개수는 12729 개이다.
ratio = 0.9 인 mRNA 는 16867 개이다.
ratio = 0.8 인 mRNA 는 21526 개이다.
ratio = 0.5 인 mRNA 는 29046 개이다.

위의 결과에서 mRNA 의 발현값이 0 인 경우가 굉장히 많다는 것을 알 수 있었다.
그렇다면 case 마다 발현량이 0 이 아닌 mRNA 의 개수, 즉 발현된 mRNA 의 개수를 새로운 변수로 사용할 수 있을 것이라고 생각하였다. 아래 코드를 이용하여 각각의 case 에 대한 발현된 mRNA 개수를 나타내는 no_zero_sum 변수를 새로 만들

어보기도 하였다.

```
no_zero_num <- vector()
for(i in 1:1195){
  tf <- mRNA[i,]>0
  no_zero_num <- append(no_zero_num, sum(tf))
}
no_zero_num
```

다음으로 변수간의 상관관계를 알아보기 위하여 WGCNA 의 cor 함수를 사용하였다. valid_mRNA 는 모든 수치가 0 인 변수는 제거한 mRNA 자료이다.

```
torf <- vector()

for(i in big_data[,-1]){
  if(sum(i)==0){
    torf <- append(torf, FALSE)
    } else {
      torf <- append(torf, TRUE)
    }
}

valid_mRNA <- mRNA[,torf]

library("WGCNA")
corr <- WGCNA::cor(valid_mRNA)
sum(corr>0.5)
```

결과값은 140577029 이다.

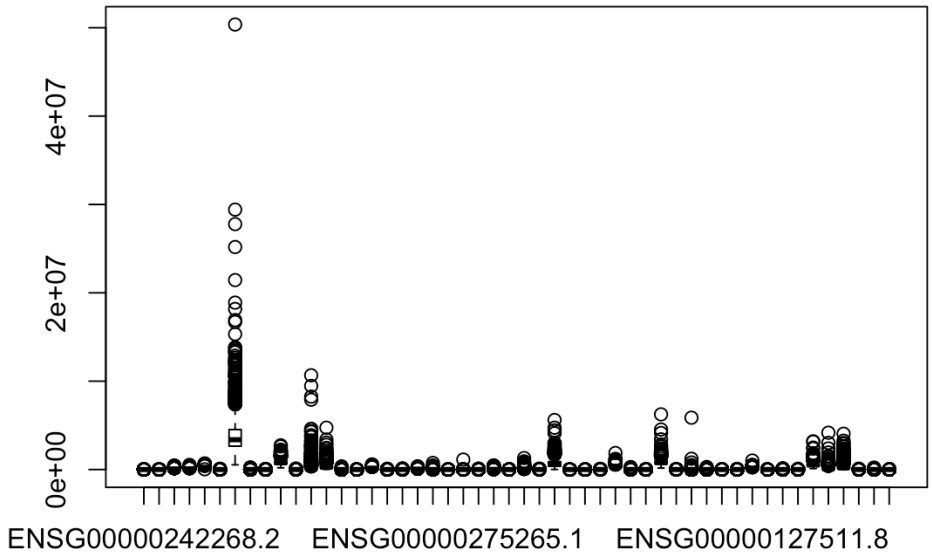
위에서 corr 변수는 발현량이 모두 0 은 아닌 mRNA 변수들(약 58000 개)의 상관계수를 담고 있는 변수로, 58000*58000 행렬이다.

따라서 58,000*58,000 행렬로 나타난 상관계수 중 14,577,029 개의 상관계수가 0.5 를 넘어가는 것을 알 수 있다.

이는 약 0.4 퍼센트에 해당하는 값으로, 상관 관계가 높은 유전자 수가 아주 많은 것은 아니라는 것을 알 수 있다.

다음으로는 발현값들의 분포를 살펴보았다. 다음 그림은 50 개 mRNA 의 발현값에 대한 box plot 이다.

```
boxplot( mRNA[,1:50] )
```



앞서 보았듯, 0 에 분포하는 경우가 매우 많고, 그림만으로는 어떤 분포를 따르는지 정확히 알기 어려웠다.

혹시 발현값이 정규분포를 따르는지 확인하기 위하여, 각각의 mRNA 에 대하여 shapiro-wilk test 를 실행해보았다. (모든 값이 0 인 mRNA 는 제외)

```
t.f <- vector()
for(i in 1:60483){
   exp_vals <- valid_mRNA[,i]
   tf <- exp_vals >0
   exp_vals <- exp_vals[tf]
   result <- shapiro.test(exp_vals)
   t.f <- append(t.f, result$p.value)
}
sum( t.f > 0.05 )
```

결과값은 0 이다.

6 만개의 변수들 중에서 shapiro-wilk test 에서 p-value 가 0.05 를 넘는 경우는 하나도 없는 것이다. p-value 가 0.05 이하이면 정규분포를 따른다는 귀무가설을 기각하게 되므로, 모든 발현량은 정규분포를 따르지 않는 것으로 생각된다.

기준값을 0.01 로 하여도 결과값은 0 이며, 0.001 로 할 때에는 1 이 이었다.

이상으로 탐색적 분석을 마친다.