gexp :
유전자 마커 발굴 소프트웨어
(machine learning 기반
gene marker 통합 분석 library)

팀명 : 센서쟁이
팀원 : 김예지, 한채은, 강서연, 이선우

목차

폐암 아종 소개

폐암의 아종(subtype)은 형태 및 위치에 따라 소세포암, 비소세포암으로 나뉩니다. 또, 비소세포암은 폐 선암(LUAD), 폐 편평상피세포암(LUSC), 큰 세포암으로 나뉩니다.

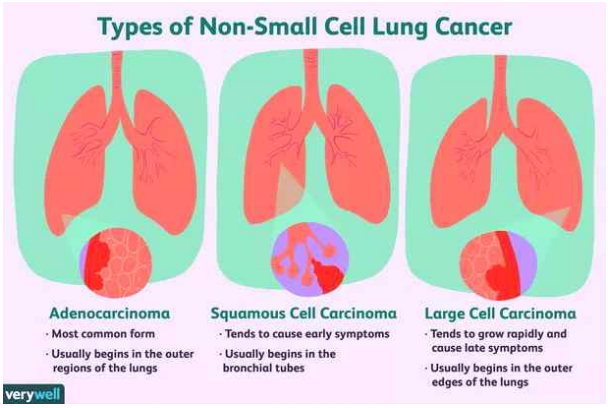
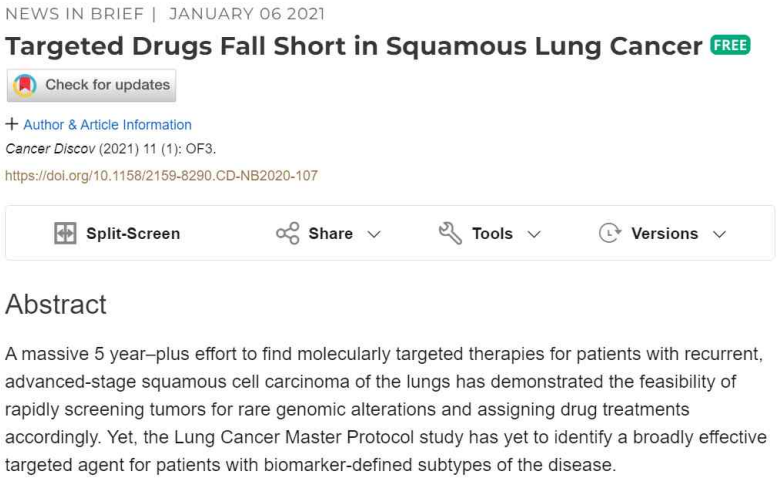


그림 1 각 폐 암 종류의 따른 형태

특히, 이 중에서 폐 선암과 폐 편평상피세포암의 비중이 높습니다.
그런데 폐 선암의 표적 치료제가 많이 개발되어 있는 것에 비해 폐 편평피상피세포암의 표적치료제의 개발은 아직 부족합니다.


NEWS IN BRIEF | JANUARY 06 2021

Targeted Drugs Fall Short in Squamous Lung Cancer FREE

Check for updates

+ Author & Article Information

Cancer Discov (2021) 11 (1): OF3.

https://doi.org/10.1158/2159-8290.CD-NB2020-107

Split-Screen Share Tools Versions

Abstract

A massive 5 year–plus effort to find molecularly targeted therapies for patients with recurrent, advanced-stage squamous cell carcinoma of the lungs has demonstrated the feasibility of rapidly screening tumors for rare genomic alterations and assigning drug treatments accordingly. Yet, the Lung Cancer Master Protocol study has yet to identify a broadly effective targeted agent for patients with biomarker-defined subtypes of the disease.

그림 2. 폐 선암의 표적치료제는 많이 개발되어 있다.

이러한 폐 편평상피세포암의 특징을 이해하는데 바이오 마커 분석이 유의한 결과를 얻을 수 있을 것으로 기대합니다.
본 시연 시나리오는 두 종류의 폐암 아종(폐 선암, 폐 편평상피세포암)에 대한 바이오 마커 유전자를 gexp library로 분석하는 과정입니다.

시연 시나리오 1) 폐암 아종 유전자 마커 분석

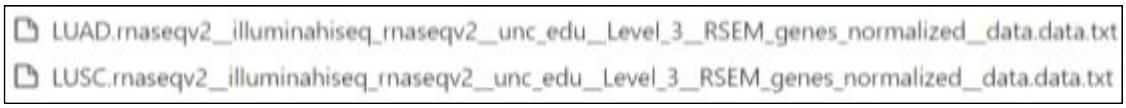
1. Data download

사용자가 입력한 암(LUAD(폐 선암), LUSC(폐 편평상피세포암))에 대한 원시 데이터를 data_source를 참고하여 data_dir에 다운로드 합니다.

```
>>> from gexp import download_data
>>> download_data(cancer_list=['LUAD','LUSC'], data_dir ='./Data')

The LUAD is downloaded successfully
The LUSC is downloaded successfully
```

<result>



2. Data load

다운로드 받은 데이터에서 정상세포 데이터를 제외하고 Target 변수를 추가하여 pandas 데이터 프레임으로 데이터를 로드합니다.

```
>>> from gexp import load_labeled_data
>>> binary_data=load_labeled_data(data_dir = './Data', label_list=['LUAD','LUSC'])
>>> binary_data
```

Hybridization REF	?|100130426	?|100133144	?|100134869	?|10357	?|10431	?|136542	?|155060	?|26823	?|280660	?|317712	...	psiTPTE22|387590	tAKR|389932	Target
TCGA-05-4244-01A-01R-1107-07	0.0	10.0113	11.2820	49.5994	848.9397	0.0	345.2308	1.0472	0.0000	0.0	...	6.6323	0.0000	LUAD
TCGA-05-4249-01A-01R-1107-07	0.0	7.1957	12.4436	90.5117	924.0158	0.0	145.2025	1.6098	0.0000	0.0	...	179.9738	0.0000	LUAD
TCGA-05-4250-01A-01R-1107-07	0.0	7.2453	6.0184	49.5366	1140.6781	0.0	51.7284	0.0000	0.0000	0.0	...	6.3003	0.0000	LUAD
TCGA-05-4382-01A-01R-1206-07	0.0	11.3311	7.5740	82.8303	807.1729	0.0	240.0221	0.4786	0.2393	0.0	...	35.1777	0.0000	LUAD
TCGA-05-4384-01A-01R-1755-07	0.0	3.2254	3.4942	72.5351	562.0037	0.0	274.2822	0.6109	0.0000	0.0	...	378.1307	0.0000	LUAD
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TCGA-O2-A52S-01A-11R-A262-07	0.0	19.9503	47.1026	176.7177	1188.3278	0.0	226.8212	1.6556	0.0000	0.0	...	5.3808	1.2417	LUSC
TCGA-O2-A52V-01A-31R-A262-07	0.0	30.0872	15.2957	188.7215	1248.0303	0.0	147.4945	0.4202	0.0000	0.0	...	2.1011	0.0000	LUSC
TCGA-O2-A52W-01A-11R-A26W-07	0.0	53.6593	33.3907	260.3332	789.3606	0.0	854.3794	0.0000	0.0000	0.0	...	12.3589	0.0000	LUSC
TCGA-O2-A5IB-01A-11R-A27Q-07	0.0	72.5666	42.1832	160.1624	460.8626	0.0	569.2226	1.3312	0.0000	0.0	...	1.0650	0.0000	LUSC
TCGA-XC-AA0X-01A-32R-A405-07	0.0	44.7501	23.6304	76.6488	805.9507	0.0	311.6273	0.5220	0.0000	0.0	...	460.3941	0.0000	LUSC

1018 rows × 20532 columns

3. Rank biomarker gene

라벨 인코딩을 수행하고, 지정한 머신러닝 모델(예시에서는 RF(RandomForest), XGB(XGBoost), Ada(AdaBoost))을 사용하여 유전자에 대한 중요도 순위를 계산합니다.

```
>>> from gexp import biomarker_rank
>>> data = binary_data.copy()
>>> data['Target'] = data['Target'].map({'LUAD':0, 'LUSC':1})  #함수 사용전 라벨인코딩을 한다.
>>> rank, importances = biomarker_rank(data, models=[['RF', 'default'],['XGB' , {'colsample_bytree': 0.5, 'n_estimators': 200, 'subsample': 0.75}],['Ada','recommended']])
```

<rank>

Hybridization REF	RF	XGB	Ada
KRT74|121391	1	172	156
FAM181B|220382	2	172	156
DSC3|1825	3	3	9
LASS3|204219	4	1	110
BNC1|646	5	148	156
...	...	...	...

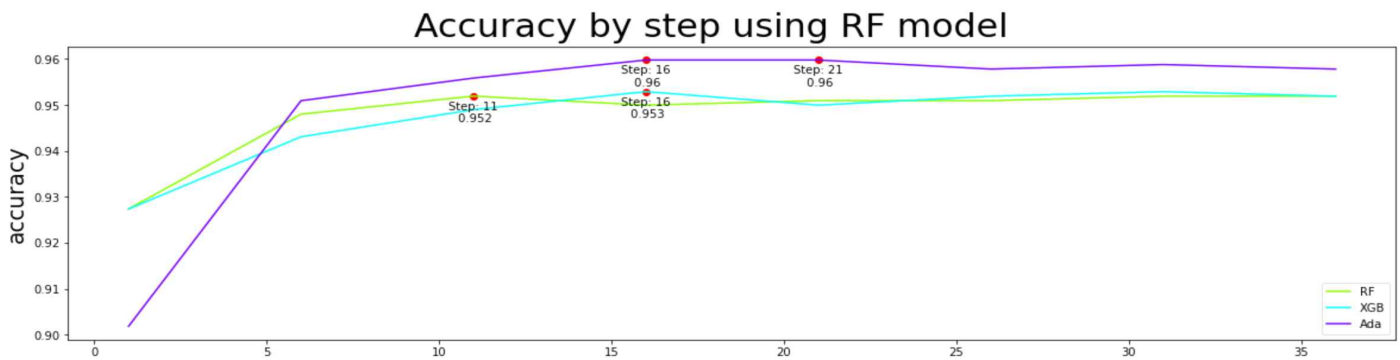
<importances>

Hybridization REF	RF	XGB	Ada
KRT74|121391	0.025327	0.000000	0.000000
FAM181B|220382	0.018782	0.000000	0.000000
DSC3|1825	0.017703	0.060717	0.013089
LASS3|204219	0.015998	0.343053	0.004774
BNC1|646	0.013963	0.000199	0.000000
...	...	...	...

4. Visualization stepwise accuracy

계산한 유전자 순위에 대해서 단계별로(예시에서는 1부터 40까지 5step씩 증가) 상위 N개를 바이오 마커로 가정했을 때의 정확도 성능을 평가하고, 시각화합니다.

```
>>> from gexp import plot_stepwise_accuracy
>>> acc = plot_stepwise_accuracy(data, rank, list(np.arange(1,41,5), model = ['RF', "recommended"], accuracy_metric=['accuracy'], multi_class=False)
>>> acc
```



	RF_accuracy	XGB_accuracy	Ada_accuracy
1	0.907718	0.926355	0.896899
6	0.940124	0.944040	0.947972
11	0.944040	0.949937	0.955839
16	0.948947	0.949937	0.958766
21	0.951898	0.949937	0.958775
26	0.947972	0.954844	0.957790
31	0.947967	0.953859	0.958775
36	0.950918	0.953859	0.958775

5. About gene list

성능 평가 결과를 가지고 가장 성능이 좋은 model의 유전자 수만큼 유전자의 설명을 보여준다.

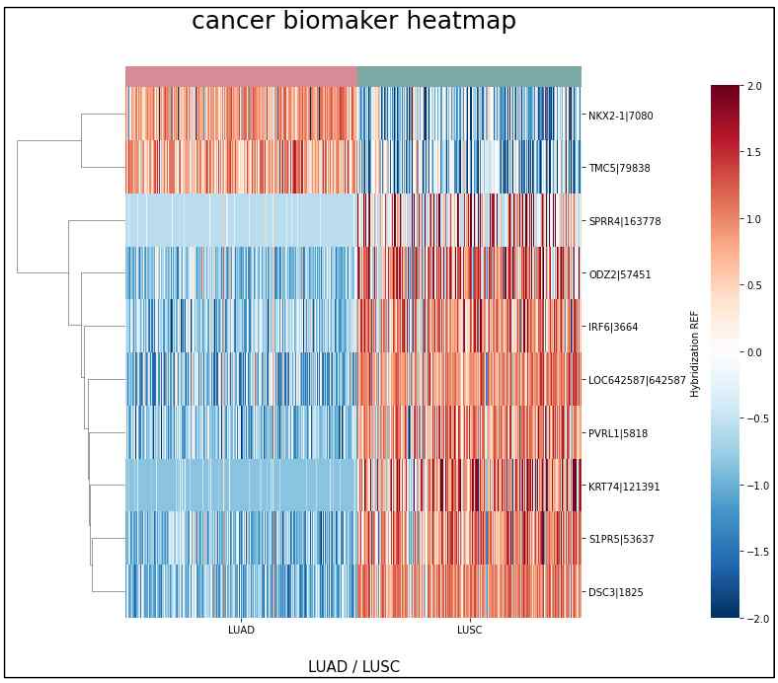
```
>>> from gexp import describe_genes
>>> mx = acc.max()
>>> mx_where = mx.argmax()
>>> gene_list = rank.sort_values(by='RF').iloc[:acc.iloc[:,mx_where].argmax()].index
>>> describe_genes(gene_list)
```

	GeneID	Symbol	type_of_gene	map_location	description	links
0	54848	ARHGEF38	protein-coding	4q24	Rho guanine nucleotide exchange factor 38	https://www.genecards.org/cgi-bin/carddisp.pl?gene=54848
1	408	ARRB1	protein-coding	11q13.4	arrestin beta 1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=408
2	53637	S1PR5	protein-coding	19p13.2	sphingosine-1-phosphate receptor 5	https://www.genecards.org/cgi-bin/carddisp.pl?gene=53637
3	121391	KRT74	protein-coding	12q13.13	keratin 74	https://www.genecards.org/cgi-bin/carddisp.pl?gene=121391
4	646	BNC1	protein-coding	15q25.2	basonuclin 1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=646
5	339967	TMPRSS11A	protein-coding	4q13.2	transmembrane serine protease 11A	https://www.genecards.org/cgi-bin/carddisp.pl?gene=339967
6	348825	TPRXL	pseudo	3p25.1	tetrapeptide repeat homeobox like (pseudogene)	https://www.genecards.org/cgi-bin/carddisp.pl?gene=348825
7	4680	CEACAM6	protein-coding	19q13.2	CEA cell adhesion molecule 6	https://www.genecards.org/cgi-bin/carddisp.pl?gene=4680
8	221	ALDH3B1	protein-coding	11q13.2	aldehyde dehydrogenase 3 family member B1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=221

6. Heatmap Visualization

상위 10개 바이오 마커 유전자에 대해서 정규화된 데이터를 사용하여 히트맵(heatmap)을 그려 시각화합니다.

```
>>> from gexp import normalize, plot_heatmap
>>> log1p_zscore_df = normalize(binary_data, methods = ['log1p', 'z_score'], exclude='Target')
>>> plot_heatmap(log1p_zscore_df, gene_list)
```



유방암 아종에 대한 바이오 마커 분석

유방암 아종(subtype)은 분자적 분류에 따라 LuminalA, LuminalB, Her2, Basal(triple negative)로 나뉩니다.

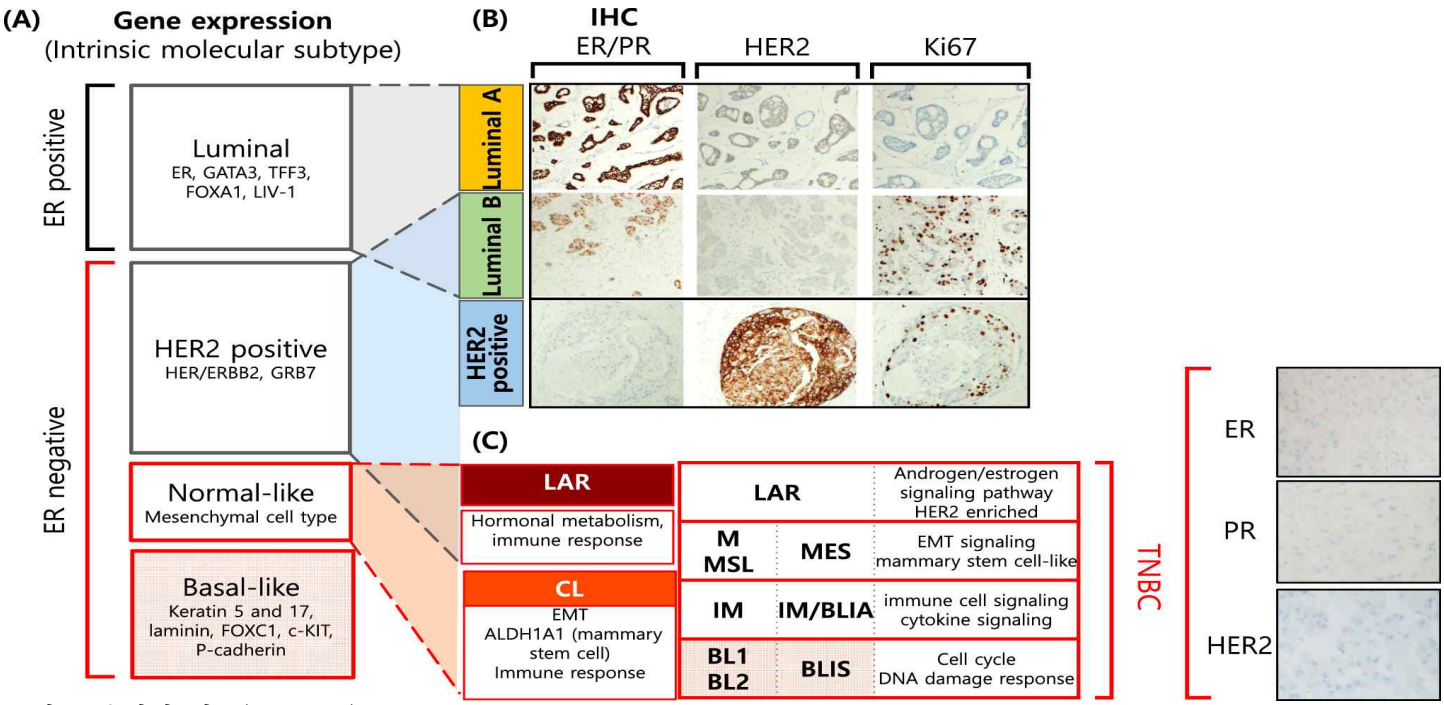


그림 3 유방암 아종(subtype)

이러한 아종 분류에 따라 예후(prognosis)와 치료법을 다르게 적용합니다. LuminalA<LuminalB<Her2<Basal 순서로 예후가 나쁘고 치료가 어렵습니다. 유방암에 아종을 분류하는 유전자에 대한 연구가 지속적으로 진행되어 왔습니다. 본 시연 시나리오는 네 종류의 유방암 아종에 대한 바이오 마커 유전자를 gexp library로 분석하는 과정입니다.

Journal List > Am J Cancer Res > v.5(10); 2015 > PMC4656721

American Journal of Cancer Research
ISSN: 2156-6976
Home Editorial Board Contents Submission

Am J Cancer Res. 2015; 5(10): 2929–2943.
Published online 2015 Sep 15.
PMCID: PMC4656721
PMID: 26693050

Breast cancer intrinsic subtype classification, clinical use and future trends

Xiaofeng Dai,[1] Ting Li,[1] Zhonghu Bai,[1] Yankun Yang,[1] Xiuxia Liu,[1] Jinling Zhan,[1] and Bozhi Shi[2]

▸ Author information ▸ Article notes ▸ Copyright and License information Disclaimer

그림 4 유방암 아종(subtype)을 분류하는 유전자에 대한 연구는 지속적으로 진행되고 있다.

시연 시나리오 2) 유방암 아종 유전자 마커 분석

1. Data download

사용자가 입력한 암 데이터들을 다운로드 해줍니다.

```
>>> from gexp import download_data
>>> multi_data=download_data(data_dir = './Data', cancer_list=['BRCA'])
>>> multi_data
The BRCA is downloaded successfully
```

<result>

BRCA.rnaseqv2__illuminahiseq_rnaseqv2__unc_edu__Level_3__RSEM_genes_normalized__data.data.txt

2. Data load

암 이름으로 라벨링을 하여 하나의 데이터 프레임으로 합칩니다.

```
>>> from gexp import load_labeled_data
>>> # 유방암 옵션은 ['LumA', 'Her2', 'LumB', 'Basal']
>>> multi_data = load_labeled_data(data_dir = './Data', label_list=['BRCA_LUMA', 'BRCA_LUMB',
'BRCA_HER2', 'BRCA_BASAL'], patient_type = './BRCApatients_type.csv')
>>> multi_data
```

	?|100130426	?|100133144	?|100134869	?|10357	?|10431	?|136542	?|155060	?|26823	?|280660	?|317712	...	ZXDB|158586	psiTPTE22|387590	tAKR|389932	Target
TCGA-3C-AAAU-01A-11R-A41B-07	0.0000	16.3644	12.9316	52.1503	408.0760	0.0	1187.0050	0.0000	0.0	0.0	...	1007.7824	1.7233	0.0000	BRCA_LUMA
TCGA-3C-AALK-01A-11R-A41B-07	0.0000	12.0894	11.0799	143.8643	865.5358	0.0	552.7513	0.4137	0.0	0.0	...	437.7327	66.6115	0.0000	BRCA_LUMA
TCGA-4H-AAAK-01A-12R-A41B-07	0.0000	6.8468	14.4298	84.2128	766.3830	0.0	260.8511	0.4255	0.0	0.0	...	424.2553	187.2340	0.0000	BRCA_LUMA
TCGA-5L-AAT0-01A-12R-A41B-07	0.0000	3.9889	13.6090	114.2572	807.7431	0.0	276.2868	0.0000	0.0	0.0	...	643.4961	85.0565	0.0000	BRCA_LUMA
TCGA-5L-AAT1-01A-12R-A41B-07	0.0000	0.0000	10.5949	115.9984	1108.3945	0.0	208.6390	0.0000	0.0	0.0	...	568.0522	57.8647	0.0000	BRCA_LUMA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TCGA-PL-A8LV-01A-21R-A41B-07	0.4618	20.0732	33.8820	72.5789	964.3347	0.0	545.9534	0.0000	0.0	0.0	...	302.6978	372.6566	0.0000	BRCA_BASAL

3. Rank biomarker gene

라벨 인코딩을 하여 모델별로 상위 중요도 유전자를 나타냅니다.

```
>>> from gexp import biomarker_rank
>>> data = multi_data.copy()
>>> data['Target'] = data['Target'].map({'BRCA_LUMA':0, 'BRCA_LUMB':1, 'BRCA_HER2':2, 'BRCA_BASAL':3})
>>> rank, importances = biomarker_rank(data, models=[['RF', 'default'],['XGB' , {'colsample_bytree': 0.5, 'n_estimators': 200, 'subsample': 0.75}],['Ada','recommended']])
```

<rank>

	RF	XGB	Ada
TPX2|22974	1	789	108
CCNB2|9133	2	789	108
NCAPH|23397	3	789	108
AURKA|6790	4	27	108
MLPH|79083	5	2	32
...	...	...	...

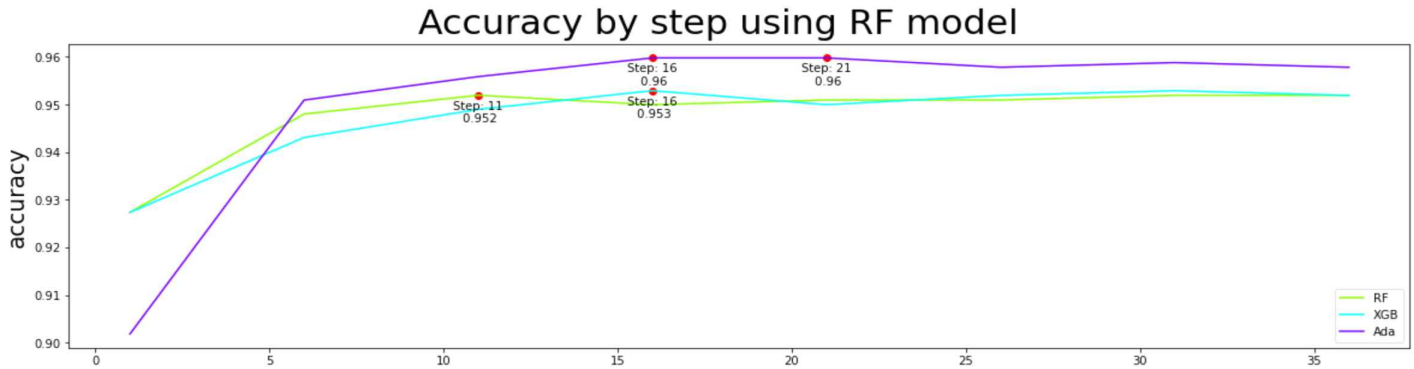
<importances>

	RF	XGB	Ada
TPX2|22974	0.011072	0.000000	0.000000
CCNB2|9133	0.007005	0.000000	0.000000
NCAPH|23397	0.006822	0.000000	0.000000
AURKA|6790	0.006796	0.006119	0.000000
MLPH|79083	0.006722	0.036483	0.008554
...	...	...	...

4. Visualization stepwise accuracy

상위 중요도 유전자 수에 따른 정확도 성능을 평가하여 시각화합니다.

```
>>> from gexp import plot_stepwise_accuracy
>>> acc = plot_stepwise_accuracy(data, rank, list(np.arange(1,41,5)), model = ['MLP', "recommended"], accuracy_metric=['accuracy'], multi_class=True)
>>> acc
```



	RF_accuracy	XGB_accuracy	Ada_accuracy
1	0.629827	0.423160	0.481637
6	0.694071	0.727975	0.666722
11	0.732782	0.720527	0.741250
16	0.777130	0.773415	0.831897
21	0.783774	0.838585	0.723406
26	0.747930	0.858312	0.824349
31	0.767746	0.863073	0.845140
36	0.777193	0.875311	0.812112

5. About gene list

성능 평가 결과를 가지고 가장 성능이 좋은 model의 유전자 수만큼 유전자를 보여주고,
유전자에 대한 정보도 함께 알 수 있다.

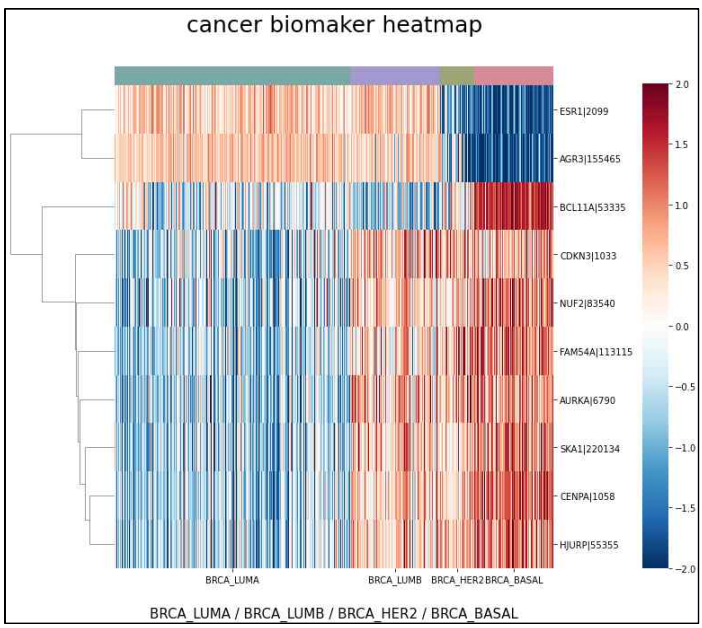
```
>>> from gexp import describe_genes
>>> mx = acc.max()
>>> mx_where = mn.argmax()
>>> gene_list = rank.sort_values(by='RF').iloc[:acc.iloc[:,mx_where].argmax()].index
>>> describe_genes(gene_list)
```

31	123099	DEGS2	protein-coding	14q32.2	delta 4-desaturase, sphingolipid 2	https://www.genecards.org/cgi-bin/carddisp.pl?gene=123099
32	51678	PALS2	protein-coding	7p15.3	protein associated with LIN7 2, MAGUK p55 family member	https://www.genecards.org/cgi-bin/carddisp.pl?gene=51678
33	10551	AGR2	protein-coding	7p21.1	anterior gradient 2, protein disulphide isomerase family member	https://www.genecards.org/cgi-bin/carddisp.pl?gene=10551
34	4001	LMNB1	protein-coding	5q23.2	lamin B1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=4001
35	83540	NUF2	protein-coding	1q23.3	NUF2 component of NDC80 kinetochore complex	https://www.genecards.org/cgi-bin/carddisp.pl?gene=83540
36	81930	KIF18A	protein-coding	11p14.1	kinesin family member 18A	https://www.genecards.org/cgi-bin/carddisp.pl?gene=81930
37	983	CDK1	protein-coding	10q21.2	cyclin dependent kinase 1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=983
38	79000	AUNIP	protein-coding	1p36.11	aurora kinase A and ninein interacting protein	https://www.genecards.org/cgi-bin/carddisp.pl?gene=79000

6. Normalize & Heatmap

상위 10개 바이오 마커 유전자에 대해서 정규화된 데이터를 사용하여 히트맵(heatmap)을 그려 시각화합니다.

```
>>> from gexp import normalize, plot_heatmap
>>> log1p_zscore_df = normalize(multi_data, methods = ['log1p', 'z_score'], exclude='Target')
>>> plot_heatmap(log1p_zscore_df, gene_list)
```



<부록>

함수 매뉴얼

gexp.download_data

def download_data(cancer_list, data_dir='./Data', data_source='./Metadata/cancer_link.csv')

- **Description**

The Cancer Genome Atlas(TCGA)의 cancer_list 암에 대한 암 유전체 mRNAseq2 데이터를 data_source의 주소를 참고하여 data_dir에 다운받는다. (필요한 메타 데이터는 Metadata파일 안에 자동으로 다운 받는다.)

parameter::	**cancer_list : list , len(list) <= 37** ["ACC", "BLCA", "BRCA", "CESE", "CHOL", "COAD", "COADREAD", "DLBC", "ESCA", "GBM", "GBMLGG", "HNSC", "KICH", "KIPAN", "KIRC", "KIRP", "LAML", "LGG", "LIHC", "LUAD", "LUSC", "MESO", "OV", "PAAD", "PCPG", "PRAD", "READ", "SARC", "SKCM", "STAD", "STES", "TGCT", "THCA", "THYM", "UCEC", "UCS", "UVM"] 원하는 TCGA data 암 종류를 리스트 형식의 Argument로 입력 **data_source : default =** './Metadata/cancer_link.csv' package 사용 시 제공되는 cancer_link 파일 ("firebrowse" 사이트 내에 존재하는 cancer별 TCGA 데이터의 다운로드 경로 파일) **data_dir : default =** './Data' 데이터 다운로드 경로 지정 path 경로 설정 시 : 데이터가 다운 받아지는 경로 조정 default = './Data' : 현재 위치에 Data 파일을 만들고 다운로드

- **Required Libraries**

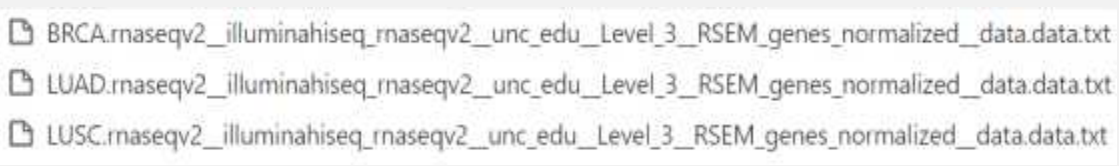
os, pandas, requests, shutil, targile

- **Examples**

```
>>> from gexp import download_data
>>> download_data(cancer_list=['LUAD', 'LUSC', 'BRCA'], data_source='./Metadata/cancer_link.csv',
    data_dir ='Data')

The LUAD already exists.
The LUSC already exists.
The BRCA is downloaded successfully
```

<result>



gexp.load_labeled_data

def load_labeled_data(data_dir, label_list, patient_type='./Metadata/BRCApatients_type.csv')

- **Description**

label_list에 들어있는 암에 대해서 pandas 데이터 프레임 형식으로 데이터를 불러온다

parameter::	
	data_dir : path, default = ‘./Data’ TCGA 데이터를 압축 해제한 txt파일들이 들어 있는 경로 **data_source : list** binary-class dataframe : [“ACC”, “BLCA”] mulit-class dataframe : [“UCEC”, “UCS”, “UVM”] BRCA subtype dataframe : ['BRCA_LUMA','BRCA_LUMB', 'BRCA_HER2'] - subtype = ['LumA, 'Her2', 'LumB', 'Basal'] **patient_type : default =** './Metadata/BRCApatients_type.csv', BRCA의 Subtype 분류를 위한 csv 파일 ('./BRCApatients_type.csv') 출처 : dataon의 ~~을 사용해 일부 subtype으로 사용함

- **Required Libraries**

os, pandas, numpy

- **Examples**

```
>>> from gexp import load_labeled_data
>>> Binary_df = load_labeled_data(data_dir = './Data', label_list=['LUAD', 'LUSC'])

load :LUAD.rnaseqv2__illuminahiseq_rnaseqv2__unc_edu__Level_3__RSEM_genes_normalized__data.data.txt
load :LUSC.rnaseqv2__illuminahiseq_rnaseqv2__unc_edu__Level_3__RSEM_genes_normalized__data.data.txt
```

Hybridization REF	?|100130426	?|100133144	?|100134869	?|10357	?|10431	?|136542	?|155060	?|26823	?|280660	?|317712	...	ZYG11B|79699	ZYX|7791	ZZEF1|23140	ZZZ3|26009	psiTPTE22|387590	tAKR|389932	Target
TCGA-05-4244-01A-01R-1107-07	0.0	10.0113	11.2820	49.5994	848.9397	0.0	345.2308	1.0472	0.0000	0.0	...	1088.0531	2837.9440	871.2802	575.2683	6.6323	0.0000	LUAD
TCGA-05-4249-01A-01R-1107-07	0.0	7.1957	12.4436	90.5117	924.0158	0.0	145.2025	1.6098	0.0000	0.0	...	787.5061	2351.2500	1138.1170	690.2752	179.9738	0.0000	LUAD
TCGA-05-4250-01A-01R-1107-07	0.0	7.2453	6.0184	49.5366	1140.6781	0.0	51.7284	0.0000	0.0000	0.0	...	475.1720	5437.4534	1170.5214	532.8691	6.3003	0.0000	LUAD
TCGA-05-4382-01A-01R-1206-07	0.0	11.3311	7.5740	82.8303	807.1729	0.0	240.0221	0.4786	0.2393	0.0	...	908.1593	6770.1537	1169.2401	663.8297	35.1777	0.0000	LUAD
TCGA-05-4384-01A-01R-1755-07	0.0	3.2254	3.4942	72.5351	562.0037	0.0	274.2822	0.6109	0.0000	0.0	...	778.8638	3341.4783	1737.3244	723.2743	378.1307	0.0000	LUAD
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TCGA-O2-A52S-01A-11R-A262-07	0.0	19.9503	47.1026	176.7177	1188.3278	0.0	226.8212	1.6556	0.0000	0.0	...	580.2980	2681.7053	1069.5364	783.1126	5.3808	1.2417	LUSC
TCGA-O2-A52V-01A-31R-A262-07	0.0	30.0872	15.2957	188.7215	1248.0303	0.0	147.4945	0.4202	0.0000	0.0	...	845.8872	6094.3376	663.9353	778.6532	2.1011	0.0000	LUSC

gexp.biomarker_rank

```
def biomarker_rank(df, models, test_size=0.2)
```

- **Description**

데이터 안에 약 2만 개의 유전자에 대해서 models의 머신러닝 모델들을 사용하여 중요도와 순위를 계산한다.

parameter::	
	df : dataframe gexp.load_labeled_data에서 cancer명이 라벨링 된 dataframe **model : list [Model , Hyperparameter], default =** ['RF', "recommended"] 정확도 성능을 계산할 sklearn model 과 파라미터 지정 **Model** : {'RF, 'Ada', 'EXtra', 'DT', 'XGB'} sklearn.ensemble.RandomForestClassifier sklearn.ensemble.AdaBoostClassifier sklearn.ensemble.ExtraTreesClassifier sklearn.tree.DecisionTreeClassifier xgboost.XGBClassifier **Hyperparameter** : {'recommended', 'default', { } } 'recommended'는 최적화를 통해 얻은 추천 하이퍼 파라미터 RF : {'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_split': 16, 'n_estimators': 200} Ada : {'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (400,), 'learning_rate': 'invscaling', 'max_iter': 3000, 'solver': 'adam'} EXtra : {'criterion': 'gini', 'max_features': 'log2', 'min_samples_split': 4, 'n_estimators': 500} DT : {'max_depth': 1, 'min_samples_leaf': 1, 'min_weight_fraction_leaf': 0.0} XGB : {'colsample_bytree': 0.5, 'n_estimators': 200, 'subsample': 0.75} 'default'는 sklearn model의 기본 default 설정 하이퍼 파라미터 { }는 직접 하이퍼 파라미터 옵션과 값 지정 **test_size : default = 0.2** train/test의 test의 비율을 설정하면 자동으로 train의 비율이 설정됨

- **Required Libraries**

pandas, sklearn, ensemble

- **Examples**

```
>>> from gexp import biomarker_rank
>>> rank, importance = biomarker_rank(data, models=[['RF', 'default'],['XGB' , {'colsample_bytree': 0.5,
                                                    'n_estimators': 200, 'subsample': 0.75}],['Ada','recommended']])
>>> rank
```

	RF	XGB	Ada
?|100130426	2629	789	108
?|100133144	4487	583	108
?|100134869	4487	307	108
?|10357	4487	353	108
?|10431	4487	789	108
...	...	...	...

```
>>> importance
```

	RF	XGB	Ada
?|100130426	0.000000	0.000000	0.0
?|100133144	0.000000	0.000066	0.0
?|100134869	0.000000	0.000528	0.0
?|10357	0.000000	0.000382	0.0
?|10431	0.000000	0.000000	0.0
...	...	...	...

gexp.plot_stepwise_accuracy

def plot_stepwise_accuracy(df, ranking_df, step_num, model=['RF', 'recommended'], accuracy_metric=['accuracy'], multi_class=None)

- **Description**

step_num의 값에 따라서 단계별로 상위 N개 유전자를 바이오 마커로 선택했을 때, 정확도 성능을 시각화한다.

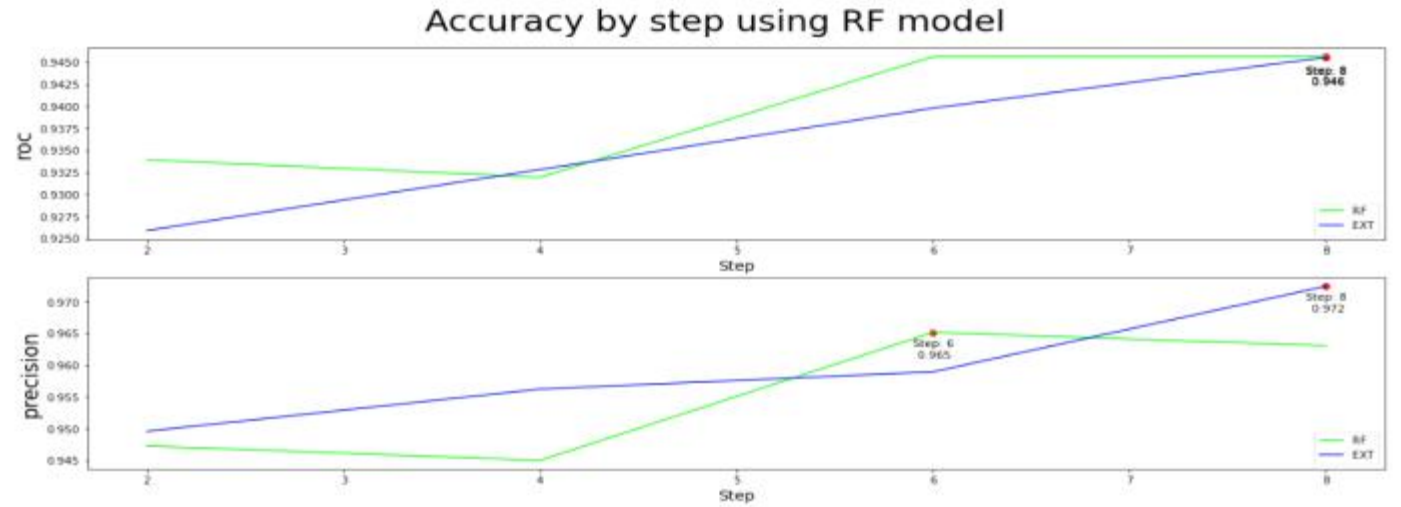
parameter::	**df : dataframe** gexp.load_labeled_data에서 cancer명이 라벨링 된 dataframe **ranking_df : dataframe,** gexp.biomarker_rank에서 얻은 method별 유전자 중요도 랭킹 순위 dataframe **step_num : list** , [1,2,3,4, ...100] 모델별 성능을 확인할 상위 유전자 개수가 담긴 리스트 **model : list [Model , Hyperparameter], default =** ['RF', "recommended"] 정확도 성능을 계산할 sklearn model 과 파라미터 지정 **Model** : {'RF, 'MLP'} sklearn.ensemble.RandomForestClassifier sklearn.neural_network.MLPClassifier **Hyperparameter** : {'recommended', 'default', { } } 'recommended'는 최적화를 통해 얻은 추천 하이퍼 파라미터 RF : {'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_split': 16, 'n_estimators': 200} MLP : {'activation': 'identity', 'alpha': 0.001, 'hidden_layer_sizes': (400,), 'learning_rate': 'invscaling', 'max_iter': 3000, 'solver': 'adam'} 'default'는 sklearn model의 기본 default 설정 하이퍼 파라미터 { }는 직접 하이퍼 파라미터 옵션과 값 지정 **accuracy_metric : list , default =** ['accuracy'] ['f1', 'accuracy', 'precision', 'recall', 'roc', 'aic', 'bic'] 성능 평가의 지표 (sklearn.metrics) multi_class = True ['f1', 'accuracy', 'precision', 'recall', 'roc'], average='macro' 사용 multi_class = None ['f1', 'accuracy', 'precision', 'recall', 'roc', 'aic', 'bic'], average='binary'을 사용 **multi_class** : {True, None} **default =** None Multi-Class Classification 시 True

- **Required Libraries**

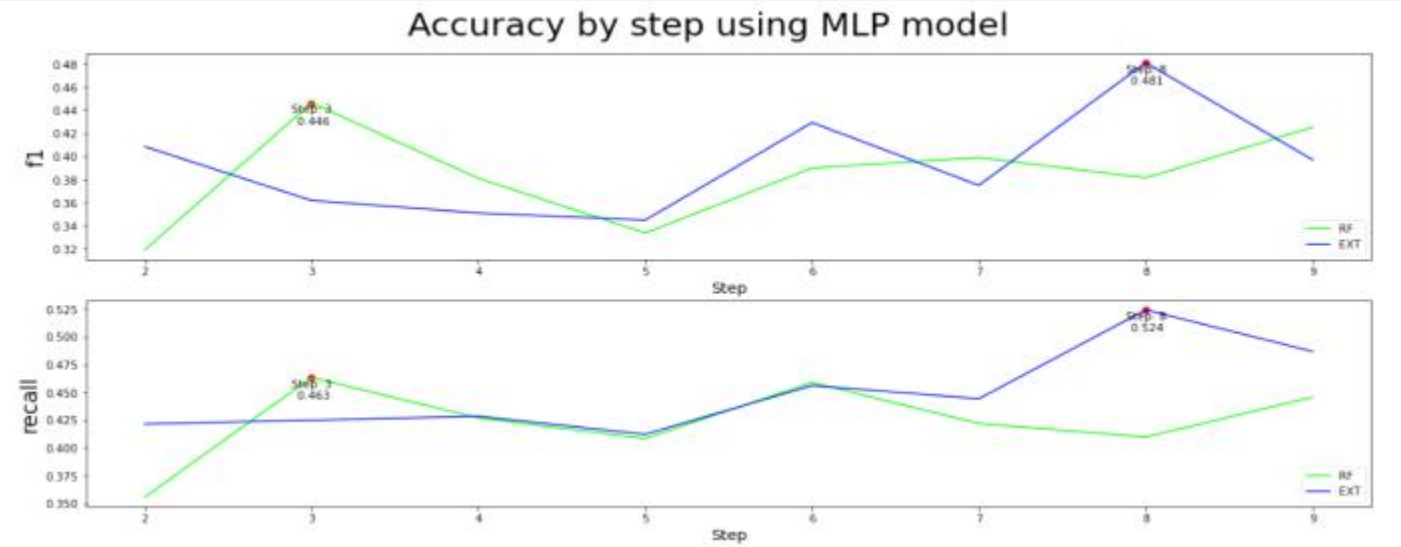
pandas, numpy, seaborn, matplotlib.pyplot, sklearn

- **Examples**

```
>>> from gexp import plot_stepwise_accuracy
>>> score_df = plot_stepwise_accuracy(Binary_df, ranking_df[['RF','EXT']], step_num=list(np.arange(2,10,2))
                                      ,model = ['RF', "recommended"], accuracy_metric=['roc', 'precision'])
```




```
>>> score_df = plot_stepwise_accuracy(Multi_df, ranking_df[['RF','EXT']], step_num=list(np.arange(2,10,1)),
                                      model = ['MLP', "recommended"], accuracy_metric=['f1', 'recall'], multi_class=True)
```




```
>>> score_df
```

	RF_f1	RF_recall	EXT_f1	EXT_recall
2	0.318883	0.355490	0.408494	0.421401
3	0.446273	0.463224	0.361813	0.424650
4	0.381067	0.426907	0.350946	0.428431
5	0.333610	0.408322	0.344819	0.412090
6	0.389932	0.458843	0.429295	0.456068
7	0.398973	0.421852	0.374931	0.444177
8	0.381369	0.409663	0.481126	0.524211
9	0.425320	0.445697	0.397084	0.486767

gexp.describe_genes

def describe_genes(gene_list, gene_description='./Metadata/Homo_sapiens.gene_info')

- ## Description

성능을 평가한 결과를 가지고 선택한 유전자 수만큼 유전자의 설명을 보여준다.

parameter::	**gene_list** : 성능을 평가한 결과를 가지고 가장 성능이 좋은 model의 유전자 수만큼의 유전자 이름 **gene_description** : **default** = './Metadata/Homo_sapiens.gene_info' 유전자의 정보가 담겨있는 메타 데이터

- ## Required Libraries

pandas

- ## Examples

```
>>> from gexp import describe_genes
>>> mx = acc.max()
>>> mx_where = mx.argmax()
>>> gene_list = score_df.iloc[:acc.iloc[:,mx_where].argmax()].index
>>> describe_genes(gene_list)
```

	GeneID	Symbol	type_of_gene	map_location	description	links
0	54848	ARHGEF38	protein-coding	4q24	Rho guanine nucleotide exchange factor 38	https://www.genecards.org/cgi-bin/carddisp.pl?gene=54848
1	408	ARRB1	protein-coding	11q13.4	arrestin beta 1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=408
2	53637	S1PR5	protein-coding	19p13.2	sphingosine-1-phosphate receptor 5	https://www.genecards.org/cgi-bin/carddisp.pl?gene=53637
3	121391	KRT74	protein-coding	12q13.13	keratin 74	https://www.genecards.org/cgi-bin/carddisp.pl?gene=121391
4	646	BNC1	protein-coding	15q25.2	basonuclin 1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=646
5	339967	TMPRSS11A	protein-coding	4q13.2	transmembrane serine protease 11A	https://www.genecards.org/cgi-bin/carddisp.pl?gene=339967
6	348825	TPRXL	pseudo	3p25.1	tetrapeptide repeat homeobox like (pseudogene)	https://www.genecards.org/cgi-bin/carddisp.pl?gene=348825
7	4680	CEACAM6	protein-coding	19q13.2	CEA cell adhesion molecule 6	https://www.genecards.org/cgi-bin/carddisp.pl?gene=4680
8	221	ALDH3B1	protein-coding	11q13.2	aldehyde dehydrogenase 3 family member B1	https://www.genecards.org/cgi-bin/carddisp.pl?gene=221
9	163259	DENND2C	protein-coding	1p13.2	DENN domain containing 2C	https://www.genecards.org/cgi-bin/carddisp.pl?gene=163259
10	23650	TRIM29	protein-coding	11q23.3	tripartite motif containing 29	https://www.genecards.org/cgi-bin/carddisp.pl?gene=23650

gexp.normalize

```
def normalize(df, methods = ['log1p', 'z_xcore'], exclude = 'Target')
```

- **Description**

methods에 포함되어 있는 표준화 방법을 순차적으로 적용하여 데이터를 정규화한다.

parameter::	**df : dataframe, df.shape** = (caner.index의 합, len(gene)) gexp.load_labeled_data에서 cancer 이름으로 라벨링한 dataframe **methods : list , default =** ["log1p", "z-score"] normalization 방법 중 log1p 또는 z-score 사용 **exclude : str, column name, default** = 'Target' labeling한 cancer의 column 이름 설정

- **Required Libraries**

pandas

- **Examples**

```
>>> from gexp import normalize
>>> Double_df = normalize(data_dir = './Data', label_list=['LUAD',  'LUSC'],
    patient_type = './BRCApatients_type.csv')
>>> Double_df
```

Hybridization REF	?|100130426	?|100133144	?|100134869	?|10357	?|10431	?|136542	?|155060	?|26823	?|280660	...	tAKR|389932	Target
TCGA-05-4244-01A-01R-1107-07	0.0	10.0113	11.2820	49.5994	848.9397	0.0	345.2308	1.0472	0.0000	...	0.0000	LUAD
TCGA-05-4249-01A-01R-1107-07	0.0	7.1957	12.4436	90.5117	924.0158	0.0	145.2025	1.6098	0.0000	...	0.0000	LUAD
TCGA-05-4250-01A-01R-1107-07	0.0	7.2453	6.0184	49.5366	1140.6781	0.0	51.7284	0.0000	0.0000	...	0.0000	LUAD

```
>>> log1p_z_score_df = normalize(Double_df, methods = ['log1p', 'z_score'], exclude = 'Target')
>>> log1p_z_score_df
```

Hybridization REF	?|100130426	?|100133144	?|100134869	?|10357	?|10431	?|136542	?|155060	?|26823	...	tAKR|389932	Target
TCGA-05-4244-01A-01R-1107-07	-0.182676	-0.144887	-0.071501	-2.104974	-0.184009	-0.031342	1.125494	1.132595	...	-0.418137	LUAD
TCGA-05-4249-01A-01R-1107-07	-0.182676	-0.466020	0.046895	-0.746134	0.045028	-0.031342	-0.004001	1.825106	...	-0.418137	LUAD
TCGA-05-4250-01A-01R-1107-07	-0.182676	-0.459459	-0.804653	-2.107822	0.614456	-0.031342	-1.340139	-0.910915	...	-0.418137	LUAD
TCGA-05-4382-01A-01R-1206-07	-0.182676	-0.021787	-0.542365	-0.947192	-0.320355	-0.031342	0.650936	0.204560	...	-0.418137	LUAD
TCGA-05-4384-01A-01R-1755-07	-0.182676	-1.186442	-1.388645	-1.247687	-1.298489	-0.031342	0.825065	0.448985	...	-0.418137	LUAD
...	...	...	...	...	...	...	...	...	...	...	...
TCGA-O2-A52S-01A-11R-A262-07	-0.182676	0.554585	1.717102	0.775994	0.725095	-0.031342	0.577139	1.874726	...	1.099508	LUSC

gexp.plot_heatmap

```
def plot_heatmap(df, gene_list, vmin = -2, vmax = 2)
```

- **Description**

gene_list에 포함되어 있는 유전자에 대해서 히트맵(heatmap)을 그려 시각화한다. rank가 높은 순으로 10개의 유전자를 뽑고 환자들의 암종류에 따라 뽑은 유전자를 구분하는 시각화 함수

parameter::	**df** : **dataframe, df.shape** = (cancer.index의 합, len(gene)) 암 label이 들어가 있는 dataframe **gene_list** : **list, len(list) = 10** 상관관계 분석하고 싶은 상위 유전자 인덱스 리스트 **vmin** : **default** = -2 df.values의 최솟값 **vmax** : **default** = 2 df.values의 최댓값

- **Use Packages**

pandas, sklearn, seaborn

- **Examples**

```
>>> from gexp import plot_heatmap
>>> gene_list = rank.sort_values(by='RF').iloc[:10].index
>>> plot_heatmap(log1p_z_score_df, gene_list)
```