In [1]:

```
# import python libraries
import os
import numpy as np
import pandas as pd
from IPython.display import clear_output
```

In [2]:

```
cwd=os.getcwd() # update location if all files from github not stored in same location as this
cwd_data=os.path.join(cwd,'seq_studies_data')
cwd_interim=os.path.join(cwd,'interim_files/')
cwd_results_RNA=os.path.join(cwd,'Results') # where results will be stored
```

In [3]:

```
# Define Useful Functions
setlen=lambda x:len(set(x)) # Calculate length of set of a list.

# Read in genes of interest in complex 2
with open(os.path.join(cwd,'genelists','C2_genes.txt')) as fc:
    list_C2=fc.read()
    list_C2=list_C2.split(',')
with open(os.path.join(cwd,'genelists','MHC1_antigen_genelist.txt')) as fc:
    list_MHC=fc.read()
    list_MHC=list_MHC.split(',')
list_outputGenes=list_MHC # update as necessary for a different gene list

def FindDataFile(ftype,prefix=None,inppath=None,secondary=None):
    """
    This function inputs a keyword such as 'sample'/ 'patient'/'mutations' as input and returns
    \nReturns False if no such file is preset in current working folder.
    \nThis function exists because studies have two kinds of naming conventions for data files
    """
    if inppath==None:
        list_all_files=os.listdir()
    else:
        list_all_files=os.listdir(inppath)
    if prefix == None:
        prefix='data'
#     list_all_files=os.listdir()
    fname=[]
    if secondary is None:
        for row in list_all_files:
            if (ftype.lower() in row.lower()) and (prefix in row.lower()):
                fname=fname+[row]
        if len(fname)>1:
            if 'data_mutations.txt' in fname:
                return 'data_mutations.txt' # kind of poor man's file name handling due to cons
            elif 'data_mutations_extended.txt' in fname:
                return 'data_mutations_extended.txt'
            else:
                return fname
        if len(fname)==1:
            return fname[0]
        else:
            return False
    else:
        for row in list_all_files:
            keywords=[prefix,ftype]+([secondary] if type(secondary)==str else secondary)
            keyword_logic=[istr.lower() in row.lower()  for istr in keywords]
            if all(keyword_logic):
                fname=fname+[row]
        if len(fname)>1:
            return fname
        if len(fname)==1:
            return fname[0]
        else:
            return False
```

SKIN

```
In [4]:
studies_skin=['mel_dfci_2019','mel_tsam_liang_2017', 'mel_ucla_2016', 'skcm_tcga', 'skcm_tcga_p
```

```
In [22]:
%%time
#ImportRNAseq data
geneslist=list_C2 # genes, at least one among which must be sequenced in a study to be included
nohugosymbol=[] # count files without gene labels - there is a code to map ensemble codes, if p
nodata=[] # count files with RNAseq data but no modifications in list of included genes.
iter1=0
list_RNAdata_skin=[]
for idir in studies_skin:
    iter1+=1 # count study numbers
    iurl=os.path.join(cwd_data,idir)
    fnames=FindDataFile(ftype='RNA',inppath=iurl,secondary=['all_sample','zscore'])
    if (type(fnames) is not str) and (type(fnames) is not bool):
        fnames=[row for row in fnames if 'v2' in row.lower()] # if both RNA seq and seq v2 pres
        if len(fnames)>1:
            fnames=[row for row in fnames if 'normal' in row.lower()] # if data with ref sample
        fname=fnames[0] if (len(fnames) <= 1) else fnames # this will create an error when ther
    else:
        fname=fnames
    # Import RNAseq files when present:
    if fname:
        df_RNAseqdata=pd.read_csv(os.path.join(iurl,fname),sep='\t',dtype=str)
        # Extract only the lines which contain an expression level change in the genes of inter
        if 'Hugo_Symbol' in df_RNAseqdata.columns:
            df_RNAseqdata['Study_ID']=idir
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Hugo_Symbol!=''] # remove unlabelled
            list_RNAdata_skin=list_RNAdata_skin+[df_RNAseqdata]#.loc[df_RNAseqdata.Hugo_Symbol.
            print('Study Number ',iter1,' done:', idir,'Nsamples:',len(df_RNAseqdata.columns)-1
        elif 'Entrez_Gene_Id' in df_RNAseqdata.columns:
            # Convert From Entrez gene ID to Hugo_Symbol from same folder
            fname_CNA=FindDataFile(ftype='data_CNA.txt',inppath=iurl)
            df_Hug2Entrz=pd.read_csv(os.path.join(iurl,fname_CNA),sep='\t',dtype=str)
            df_Hug2Entrz=df_Hug2Entrz.loc[df_Hug2Entrz.Hugo_Symbol!='']
            Entrz2Hugo=lambda entr:df_Hug2Entrz[df_Hug2Entrz.Entrez_Gene_Id==entr].Hugo_Symbol.
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Entrez_Gene_Id!=''] # remove unlabell
            df_RNAseqdata['Hugo_Symbol']=[Entrz2Hugo(irow) for irow in df_RNAseqdata.Entrez_Ger
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Hugo_Symbol!=''] # remove unlabelled
            df_RNAseqdata['Study_ID']=idir
            list_RNAdata_skin=list_RNAdata_skin+[df_RNAseqdata]#.loc[df_RNAseqdata.Hugo_Symbol.
            print('Study Number ',iter1,' done:', idir,'Nsamples:',len(df_RNAseqdata.columns)-2
#          del df_RNAseqdata
#          clear_output()
    else:
        print('No seq file',iter1,idir,fnames)
```

```
Study Number  1  done: mel_dfci_2019 Nsamples: 123 RNAseq file data_RNA_Seq_expression_tpm_all_
sample_Zscores.txt.gz
Study Number  2  done: mel_tsam_liang_2017 Nsamples: 37
RNAseq file data_RNA_Seq_mRNA_median_all_sample_Zscores.txt.gz
Study Number  3  done: mel_ucla_2016 Nsamples: 28
RNAseq file data_RNA_Seq_mRNA_median_all_sample_Zscores.txt.gz
Study Number  4  done: skcm_tcga Nsamples: 474 RNAseq file data_RNA_Seq_v2_mRNA_median_all_samp
le_Zscores.txt.gz
Study Number  5  done: skcm_tcga_pan_can_atlas_2018 Nsamples: 445 RNAseq file data_RNA_Seq_v2_m
RNA_median_all_sample_Zscores.txt.gz
Study Number  6  done: skcm_dfci_2015 Nsamples: 41
RNAseq file data_RNA_Seq_mRNA_median_all_sample_Zscores.txt.gz
Study Number  7  done: skcm_mskcc_2014 Nsamples: 22
RNAseq file data_RNA_Seq_mRNA_median_all_sample_Zscores.txt.gz
Wall time: 3min 32s
```

```
In [26]:
geneslist=list_C2+list_MHC
df_RNA_skin=pd.DataFrame(columns=geneslist)
dic_RNA_skin=dict()
```

```
    for idf in list_RNAdata_skin:
        if 'Entrez_Gene_Id' in idf.columns:
            idf1=idf.drop(columns=['Entrez_Gene_Id','Study_ID']).set_index('Hugo_Symbol').astype(fl
        else:
            idf1=idf.drop(columns=['Study_ID']).set_index('Hugo_Symbol').astype(float).groupby(by='
        idf1=idf1.dropna(how='all',axis='columns')
        dic_RNA_skin[idf.Study_ID.iloc[0]]=idf1
        geneslist=[igene for igene in geneslist if igene in idf1.columns]
        df_RNA_skin=pd.concat([df_RNA_skin,idf1[geneslist]])
```

In [29]:

```
# output individual study data for C2 and MHC genes
if not os.path.isdir(os.path.join(cwd_results_RNA,'Skin')):
    os.mkdir(os.path.join(cwd_results_RNA,'Skin'))
for istudy in dic_RNA_skin.keys():
    for igene in list_C2:
        if igene in dic_RNA_skin[istudy].columns.values:
            genelist=[igene]+[g1 for g1 in list_outputGenes if g1 in dic_RNA_skin[istudy].colum
            df_istudy=dic_RNA_skin[istudy][genelist]
            df_RNA_istudy_describe=df_istudy[igene].describe()
            def quarter1(x):
                if x<=df_RNA_istudy_describe.loc['25%']:
                    return('Low_quart')
                if x>=df_RNA_istudy_describe.loc['75%']:
                    return('Up_quart')
                else:
                    return(None)
            df_istudy['quart']=df_istudy[igene].apply(quarter1)
            fname_file='ByStudy_RNA_Zscores_SKIN_'+istudy+'_quartile_'+igene+'.xlsx'
            (df_istudy.sort_values(by=['quart'])).to_excel(os.path.join(cwd_results_RNA,'Skin',
```

```
C:\Users\GM\anaconda3\lib\site-packages\pandas\core\frame.py:3607: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy
  self._set_item(key, value)
```

In [49]:

```
# output individual study data for ALL genes
if not os.path.isdir(os.path.join(cwd_results_RNA,'Skin_GSEA')):
    os.mkdir(os.path.join(cwd_results_RNA,'Skin_GSEA'))
for istudy in dic_RNA_skin.keys():
    for igene in list_C2:
        if igene in dic_RNA_skin[istudy].columns.values:
            genelist=dic_RNA_skin[istudy].columns.values.tolist()
            df_istudy=dic_RNA_skin[istudy][genelist]
            df_RNA_istudy_describe=df_istudy[igene].describe()
            def quarter1(x):
                if x<=df_RNA_istudy_describe.loc['25%']:
                    return('Low_quart')
                if x>=df_RNA_istudy_describe.loc['75%']:
                    return('Up_quart')
                else:
                    return(None)
            df_istudy['quart']=df_istudy[igene].apply(quarter1)
            fname_file='ByStudy_RNA_Zscores_SKIN_'+istudy+'_quartile_'+igene+'.csv'
            columnsreset=['quart',igene]+[ig for ig in df_istudy.columns if (ig!='quart' and ig
            (df_istudy[columnsreset].sort_values(by=['quart'])).to_csv(os.path.join(cwd_results
```

In [50]:

```
# specialized handling of redundant values: remove any lines of redundancies where one of the r
# Then take the first one available. <<This ensures only synchronized and simultaneous sample m
redundantPIDlist=[pid for pid in set(df_RNA_skin.index) if sum(df_RNA_skin.index==pid)>1]
df_RNA_skin1=df_RNA_skin[list_C2+list_outputGenes].loc[[pid for pid in set(df_RNA_skin.index) i
for pid in redundantPIDlist:
    dfpid=df_RNA_skin[list_C2+list_outputGenes].loc[pid]
    dfpid.dropna(inplace=True)
```

```
        if len(dfpid)>=1:
            df_RNA_skin1=df_RNA_skin1.append(dfpid.iloc[[0]])
```

In [51]:

```
df_RNA_skin1.to_excel(os.path.join(cwd_results_RNA,'RNA_Zscores_SKIN_7studies.xlsx'))
```

BREAST

In [4]:

```
 # manually curated to remove cell lines and any non-applicable studies
studies_Breast=['brca_tcga_pan_can_atlas_2018','brca_tcga_pub2015', 'brca_metabric', 'brca_cpta
```

In [5]:

```
%%time
#ImportRNAseq data
geneslist=list_C2 # genes, at least one among which must be sequenced in a study to be includec
nohugosymbol=[] # count files without gene labels - there is a code to map ensemble codes, if p
nodata=[] # count files with RNAseq data but no modifications in list of included genes.
iter1=0
list_RNAdata_Breast=[]
for idir in studies_Breast:
    iter1+=1 # count study numbers
    iurl=os.path.join(cwd_data,idir)
    fnames=FindDataFile(ftype='RNA',inppath=iurl,secondary=['all_sample','zscore'])
    if (type(fnames) is not str) and (type(fnames) is not bool):
        fnames=[row for row in fnames if 'v2' in row.lower()] # if both RNA seq and seq v2 pres
        if len(fnames)>1:
            fnames=[row for row in fnames if 'normal' in row.lower()] # if data with ref sample
        fname=fnames[0] if (len(fnames) <= 1) else fnames # this will create an error when ther
    else:
        fname=fnames
    # Import RNAseq files when present:
    if fname:
        df_RNAseqdata=pd.read_csv(os.path.join(iurl,fname),sep='\t',dtype=str)
        # Extract only the lines which contain an expression level change in the genes of inter
        if 'Hugo_Symbol' in df_RNAseqdata.columns:
            df_RNAseqdata['Study_ID']=idir
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Hugo_Symbol!=''] # remove unlabelled
            list_RNAdata_Breast=list_RNAdata_Breast+[df_RNAseqdata]#.loc[df_RNAseqdata.Hugo_Syn
            print('Study Number ',iter1,' done:', idir,'Nsamples:',len(df_RNAseqdata.columns)-1
        elif 'Entrez_Gene_Id' in df_RNAseqdata.columns:
            # Convert From Entrez gene ID to Hugo_Symbol from same folder
            fname_CNA=FindDataFile(ftype='data_CNA.txt',inppath=iurl)
            df_Hug2Entrz=pd.read_csv(os.path.join(iurl,fname_CNA),sep='\t',dtype=str)
            df_Hug2Entrz=df_Hug2Entrz.loc[df_Hug2Entrz.Hugo_Symbol!='']
            Entrz2Hugo=lambda entr:df_Hug2Entrz[df_Hug2Entrz.Entrez_Gene_Id==entr].Hugo_Symbol.
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Entrez_Gene_Id!=''] # remove unlabell
            df_RNAseqdata['Hugo_Symbol']=[Entrz2Hugo(irow) for irow in df_RNAseqdata.Entrez_Ger
            df_RNAseqdata=df_RNAseqdata.loc[df_RNAseqdata.Hugo_Symbol!=''] # remove unlabelled
            df_RNAseqdata['Study_ID']=idir
            list_RNAdata_Breast=list_RNAdata_Breast+[df_RNAseqdata]#.loc[df_RNAseqdata.Hugo_Syn
            print('Study Number ',iter1,' done:', idir,'Nsamples:',len(df_RNAseqdata.columns)-2
#          del df_RNAseqdata
#          clear_output()
    else:
        print('No seq file',iter1,idir,fnames)
```

```
Study Number  1  done: brca_tcga_pan_can_atlas_2018 Nsamples: 1084 RNAseq file ['data_RNA_Seq_v
2_mRNA_median_all_sample_ref_normal_Zscores.txt.gz']
Study Number  2  done: brca_tcga_pub2015 Nsamples: 819 RNAseq file ['data_RNA_Seq_v2_mRNA_media
n_all_sample_Zscores.txt.gz']
Study Number  3  done: brca_metabric Nsamples: 1906 RNAseq file data_mRNA_median_all_sample_Zsc
ores.txt.gz
Study Number  4  done: brca_cptac_2020 Nsamples: 123 RNAseq file data_mrna_seq_fpkm_zscores_ref
_all_samples.txt.gz
Study Number  5  done: brca_tcga_pub Nsamples: 528 RNAseq file data_mRNA_median_all_sample_Zsco
res.txt.gz
Study Number  6  done: brca_tcga Nsamples: 1102 RNAseq file ['data_RNA_Seq_v2_mRNA_median_all_s
ample_Zscores.txt.gz']
Study Number  7  done: brca_smc_2018 Nsamples: 170 RNAseq file data_RNA_Seq_expression_tpm_all_
sample_Zscores.txt.gz
Wall time: 1min 10s
```

In [6]:

```
# clean up data to enable quartile generation and comparison of gene expressions
geneslist=list_C2+list_MHC
df_RNA_Breast=pd.DataFrame(columns=geneslist)
dic_RNA_Breast=dict()
for idf in list_RNAdata_Breast:
    if 'Entrez_Gene_Id' in idf.columns:
        idf1=idf.drop(columns=['Entrez_Gene_Id','Study_ID']).set_index('Hugo_Symbol').astype(f
    else:
        idf1=idf.drop(columns=['Study_ID']).set_index('Hugo_Symbol').astype(float).groupby(by='
    idf1=idf1.dropna(how='all',axis='columns')
    dic_RNA_Breast[idf.Study_ID.iloc[0]]=idf1
    geneslist=[igene for igene in geneslist if igene in idf1.columns]
    df_RNA_Breast=pd.concat([df_RNA_Breast,idf1[geneslist]])
```

In [7]:

```
# output individual study data for C2 and MHC genes
if not os.path.isdir(os.path.join(cwd_results_RNA,'Breast')):
    os.mkdir(os.path.join(cwd_results_RNA,'Breast'))
for istudy in dic_RNA_Breast.keys():
    for igene in list_C2:
        if igene in dic_RNA_Breast[istudy].columns.values:
            genelist=[igene]+[g1 for g1 in list_outputGenes if g1 in dic_RNA_Breast[istudy].co
            df_istudy=dic_RNA_Breast[istudy][genelist]
            df_RNA_istudy_describe=df_istudy[igene].describe()
            def quarter1(x):
                if x<=df_RNA_istudy_describe.loc['25%']:
                    return('Low_quart')
                if x>=df_RNA_istudy_describe.loc['75%']:
                    return('Up_quart')
                else:
                    return(None)
            df_istudy['quart']=df_istudy[igene].apply(quarter1)
            fname_file='ByStudy_RNA_Zscores_Breast_'+istudy+'_quartile_'+igene+'.xlsx'
            (df_istudy.sort_values(by=['quart'])).to_excel(os.path.join(cwd_results_RNA,'Breas
```

```
C:\Users\GM\anaconda3\lib\site-packages\pandas\core\frame.py:3607: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
ndexing.html#returning-a-view-versus-a-copy
  self._set_item(key, value)
```

In [8]:

```
# output individual study data for ALL genes
if not os.path.isdir(os.path.join(cwd_results_RNA,'Breast_GSEA')):
    os.mkdir(os.path.join(cwd_results_RNA,'Breast_GSEA'))
for istudy in dic_RNA_Breast.keys():
    for igene in list_C2:
        if igene in dic_RNA_Breast[istudy].columns.values:
            genelist=dic_RNA_Breast[istudy].columns.values.tolist()
            df_istudy=dic_RNA_Breast[istudy][genelist]
            df_RNA_istudy_describe=df_istudy[igene].describe()
            def quarter1(x):
                if x<=df_RNA_istudy_describe.loc['25%']:
                    return('Low_quart')
                if x>=df_RNA_istudy_describe.loc['75%']:
                    return('Up_quart')
                else:
                    return(None)
            df_istudy['quart']=df_istudy[igene].apply(quarter1)
            fname_file='ByStudy_RNA_Zscores_Breast_'+istudy+'_quartile_'+igene+'.csv'
            columnsreset=['quart',igene]+[ig for ig in df_istudy.columns if (ig!='quart' and ig
            (df_istudy[columnsreset].sort_values(by=['quart'])).to_csv(os.path.join(cwd_results
```

In [9]:

```
# specialized handling of redundant values: remove any lines of redundancies where one of the r
# Then take the first one available. <<This ensures only synchronized and simultaneous sample m
redundantPIDlist=[pid for pid in set(df_RNA_Breast.index) if sum(df_RNA_Breast.index==pid)>1]
df_RNA_Breast1=df_RNA_Breast[list_C2+list_outputGenes].loc[[pid for pid in set(df_RNA_Breast.ir
for pid in redundantPIDlist:
    dfpid=df_RNA_Breast[list_C2+list_outputGenes].loc[pid]
    dfpid.dropna(inplace=True)
    if len(dfpid)>=1:
        df_RNA_Breast1=df_RNA_Breast1.append(dfpid.iloc[[0]])
```

In [10]:

```
df_RNA_Breast1.to_excel(os.path.join(cwd_results_RNA,'RNA_Zscores_Breast_7studies.xlsx'))
```