AVERON Notebook:
Computational platform to predict actionable vulnerabilities enabled by rewired oncogenic networks

TUTORIAL

Ivanov lab
Department of Pharmacology & Chemical Biology
Emory University School of Medicine
Emory University
Atlanta GA, United States
2023

Contents

1. Introduction

Oncogenic mutations in tumor-driver genes are among the key events in cancer initiation and progression. Such mutations can change protein structure and functions, ultimately rewiring the protein-protein interaction (PPI) networks and dysregulating signaling and metabolic pathways. Discovery of the most clinically and biologically significant mutant-directed neomorph PPIs (neoPPIs) that drive cancer is vital to developing new personalized clinical strategies. However, experimental interrogation of neoPPI functions and therapeutic potential is highly challenging even in cell-based models and currently is not feasible in cancer patients. To address this critical challenge, we developed a computational platform, termed AVERON Notebook, to discover Actionable Vulnerabilities Enabled by Rewired Oncogenic Networks. Implemented in a widely used Jupyter Notebook format, AVERON Notebook enables systematic profiling of the association between decreased clinical outcomes and neoPPI levels, rather than the status of individual genes, uncovering molecular mechanisms of neoPPI-driven tumorigenesis, and identification of druggable and therapeutically significant neoPPI-regulated genes to inform new biological models and personalized therapeutic strategies in mutant-driven cancers.

This document provides comprehensive and in-depth guidance on the features and capabilities of the AVERON Notebook. It is designed to empower users with a thorough understanding of the Jupyter notebook's functions and operations, offering step-by-step instructions that are tailored to specific needs and requirements.

- To complete this tutorial and use the AVERON Notebook, basic knowledge of the Jupyter Notebook environment and the coding principles are preferable but not required.

- To run the AVERON Notebook Jupyter Notebook or Jupyter Lab is needed. We recommend installing Jupyter Lab as part of ANACONDA: https://www.anaconda.com/download. To execute the notebook, navigate to the /AVERON/notebook folder and open averon_notebook.ipynb file with Jupyter Lab.

- Detailed documentation on the Jupyter Lab and Notebook is available elsewhere:
- https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/jupyter/index.html
- https://docs.jupyter.org/en/latest/

2. Requirements

2.1. The python packages needed

The following python pages are use by the Averon functions:

glob	importlib	ipycytoscape	ipywidgets	Jinja2	json
lifelines	math	matplotlib	Numpy	os	pandas
py4cytoscape	pygtop	Requests	scikit_posthocs	scipy.stats	seaborn
statsmodels	api	sys	tabulate	tkinter	warnings
statsmodel.sandbox.stats.multicomp					

Please refer to the Jupyter Lab, conda, and pip documentation for the package installation instructions:

- https://jupyterlab.readthedocs.io/en/stable/user/index.html
- https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html
- https://packaging.python.org/en/latest/tutorials/installing-packages/#ensure-you-can-run-pip-from-the-command-line

Note: The Averon Notebook was tested with Jupyter Lab 3.6.3 installed with Anaconda 2.5.0. Upon the fresh installation, the following packages had to be installed: ipycytoscape, lifelines, scikit_posthocs, py4cytoscape, and pygtop==2.1.4

2.2. The software required

The Cytoscape software is needed for network visualization. The Cytoscape can be downloaded here: https://cytoscape.org/download-platforms.html

2.3. The external datasets used by AVERON

The AVERON uses the data pubclicly available through the following databases and servers:
PanCancer Atlas data, including:
Mutation data: https://gdc.cancer.gov/about-data/publications/pancanatlas
mRNA expression: https://gdc.cancer.gov/about-data/publications/pancanatlas
Clinical data: https://gdc.cancer.gov/about-data/publications/pancanatlas

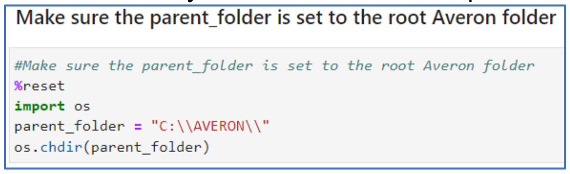
HGNC gene annotations: https://www.genenames.org/
Molecular Signature Database (MSigDB): https://www.gsea-msigdb.org/gsea/msigdb
The International Union of Basic and Clinical Pharmacology (IUPHAR): https://iuphar.org/

Except mutation data, the datasets are provided in the Averon/input folder or accessed on-the-fly.
For this tutorial, the braf_v600e.maf file that contains information about BRAF V600E mutation across different cancer type samples is used and provided. The complete PanCancer Atlas mutation dataser is available here: mc3.v0.2.8.PUBLIC.maf

3. Preparation

3.1. Import essential dependencies and setup general environment.

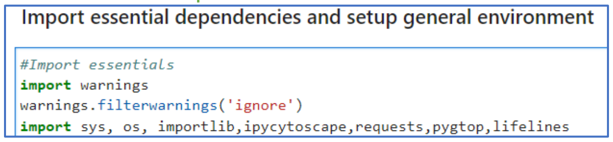
i. Make sure that you run Averon from its parent folder by executing the first cell of the notebook:



Make sure the parent_folder is set to the root Averon folder

```
#Make sure the parent_folder is set to the root Averon folder
%reset
import os
parent_folder = "C:\\AVERON\\"
os.chdir(parent_folder)
```

ii. Execute the #Import essentials cell:



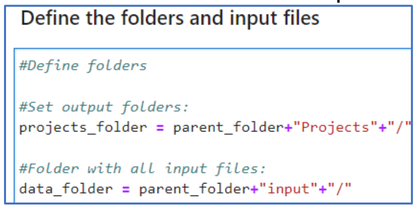
Import essential dependencies and setup general environment

```
#Import essentials
import warnings
warnings.filterwarnings('ignore')
import sys, os, importlib,ipycytoscape,requests,pygtop,lifelines
```

Proceed (y/[n])?", type "y" in the text box to confirm the reset of all variables.

3.2. Define the folders and the input files

Define the folders and the input files by executing the #Define folders cell:

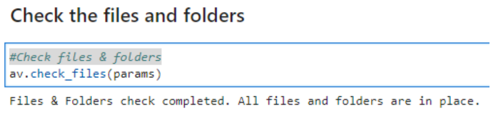


Define the folders and input files

```
#Define folders

#Set output folders:
projects_folder = parent_folder+"Projects"+"/"

#Folder with all input files:
data_folder = parent_folder+"input"+"/"
```

3.3. Check the files and folders.

Execute the #Check files & folders cell to make sure that all required files and folders are in place.

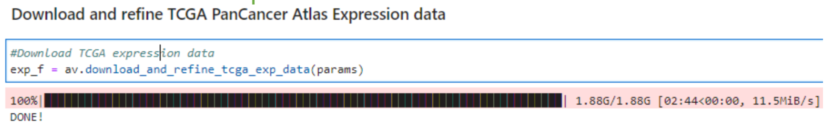


Check the files and folders

```
#Check files & folders
av.check_files(params)
```

```
Files & Folders check completed. All files and folders are in place.
```

The mRNA expression file “EBPlusPlusAdjustPANCAN_IlluminaHiSeq_RNASeqV2.geneExp.tsv” is not provided as part of AVERON Notebook. It should be downloaded and pre-processed by executing #Download TCGA expression data cell:



Download and refine TCGA PanCancer Atlas Expression data

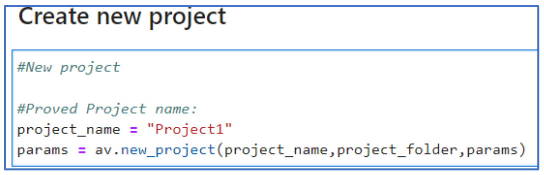
```
#Download TCGA expression data
exp_f = av.download_and_refine_tcga_exp_data(params)
```

```
100%|██████████████████████████████████████████████████| 1.88G/1.88G [02:44<00:00, 11.5MiB/s]
DONE!
```

This step should be done just once and can be skipped during the next AVERON Notebook executions.

3.4. Create new project

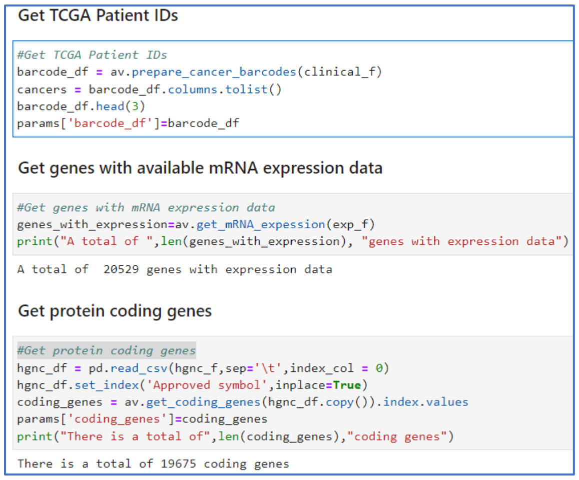
In the #New project cell provide the project name by defining *project_name* variable”
Execute the cell.



Create new project

```
#New project

#Proved Project name:
project_name = "Project1"
params = av.new_project(project_name,project_folder,params)
```

3.5. Consequently, execute the #Get TCGA Patient IDs, #Get genes with mRNA expression data, and #Get protein coding genes cells to load the corresponding datasets:



Get TCGA Patient IDs

```
#Get TCGA Patient IDs
barcode_df = av.prepare_cancer_barcodes(clinical_f)
cancers = barcode_df.columns.tolist()
barcode_df.head(3)
params['barcode_df']=barcode_df
```

Get genes with available mRNA expression data

```
#Get genes with mRNA expression data
genes_with_expression=av.get_mRNA_expession(exp_f)
print("A total of ",len(genes_with_expression), "genes with expression data")
```

```
A total of  20529 genes with expression data
```

Get protein coding genes

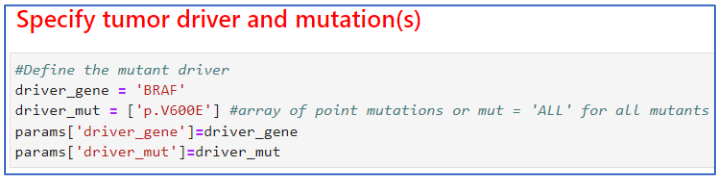
```
#Get protein coding genes
hgnc_df = pd.read_csv(hgnc_f,sep='\t',index_col = 0)
hgnc_df.set_index('Approved symbol',inplace=True)
coding_genes = av.get_coding_genes(hgnc_df.copy()).index.values
params['coding_genes']=coding_genes
print("There is a total of",len(coding_genes),"coding genes")
```

```
There is a total of 19675 coding genes
```

4. Conduct AVERON Analysis

4.1. Define the mutant tumor driver gene

In the #Define the mutant driver cell, provide the name of tumor driver gene of interest in *driver_gene* variable. Define a single mutation or list several mutations in *driver_mut* variable. Use 'ALL' to consider all driver mutations. In this tutorial we will use BRAF V600E mutant as an example:



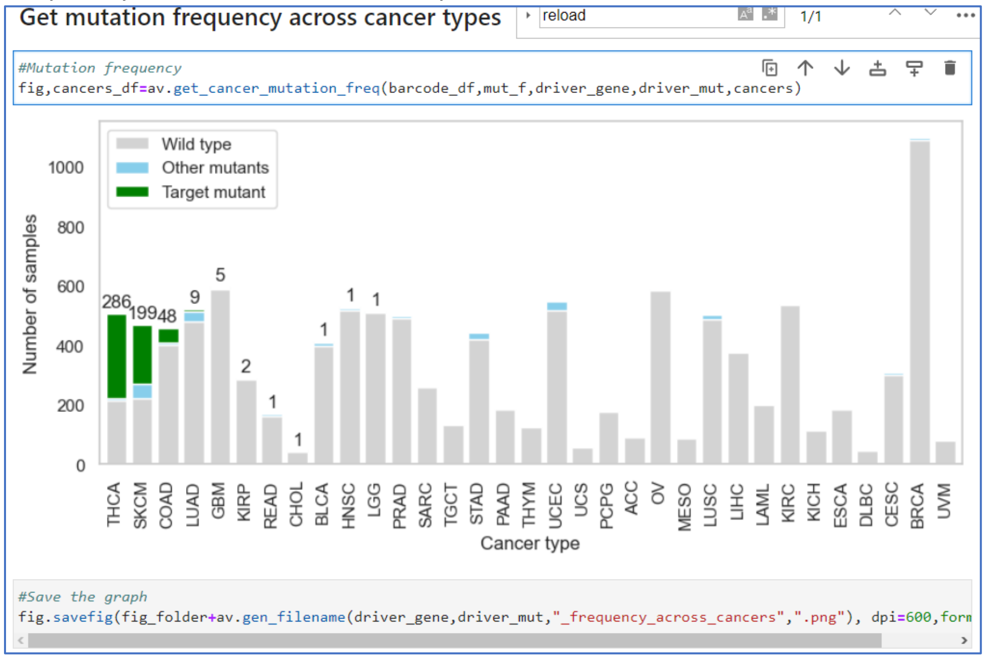
Specify tumor driver and mutation(s)

```
#Define the mutant driver
driver_gene = 'BRAF'
driver_mut = ['p.V600E'] #array of point mutations or mut = 'ALL' for all mutants
params['driver_gene']=driver_gene
params['driver_mut']=driver_mut
```

Execute the cell.

4.2. Determine mutation frequency across cancer types

4.2.1. Execute the #Mutation frequency cell and the next cell to get and save the graph with the frequency of the specified mutation across cancer types:

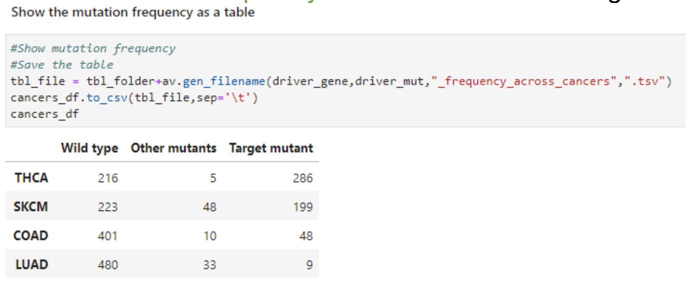


Get mutation frequency across cancer types

```
#Mutation frequency
fig,cancers_df=av.get_cancer_mutation_freq(barcode_df,mut_f,driver_gene,driver_mut,cancers)
```

```
#Save the graph
fig.savefig(fig_folder+av.gen_filename(driver_gene,driver_mut,"_frequency_across_cancers",".png"), dpi=600,form
```

4.2.2. #Show mutation frequency cell can be executed to get the mutation frequency in a tabular format:



Show the mutation frequency as a table

```
#Show mutation frequency
#Save the table
tbl_file = tbl_folder+av.gen_filename(driver_gene,driver_mut,"_frequency_across_cancers",".tsv")
cancers_df.to_csv(tbl_file,sep='\t')
cancers_df
```

	Wild type	Other mutants	Target mutant
THCA	216	5	286
SKCM	223	48	199
COAD	401	10	48
LUAD	480	33	9

It will also save the data to the Project/Tables folder

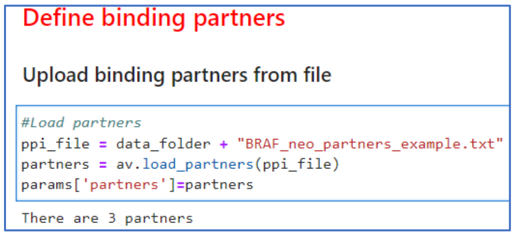
4.3. Define binding partners.

Two options are available to provide the binding partners.

i) Execute the #Load partners cell to upload all binding partners from an external file located in the AVERONE/input folder. In this tutorial, we use BRAF_neo_partners_example.txt file that contains three BRAF V600E neoPPI partners:

AURKA
RAB25
CDK4

Note: To reproduce the manuscript data, BRAF_neo_partners.txt file with a list of 130 BRAF V600E neoPPI partners should be provided in #Load partners cell:



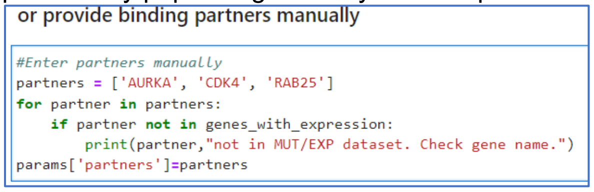
Define binding partners

Upload binding partners from file

```
#Load partners
ppi_file = data_folder + "BRAF_neo_partners_example.txt"
partners = av.load_partners(ppi_file)
params['partners']=partners
```

```
There are 3 partners
```

ii) Alternatively, move on to the #Enter partners manually cell to manually upload specific binding partners by populating an array named "partners":

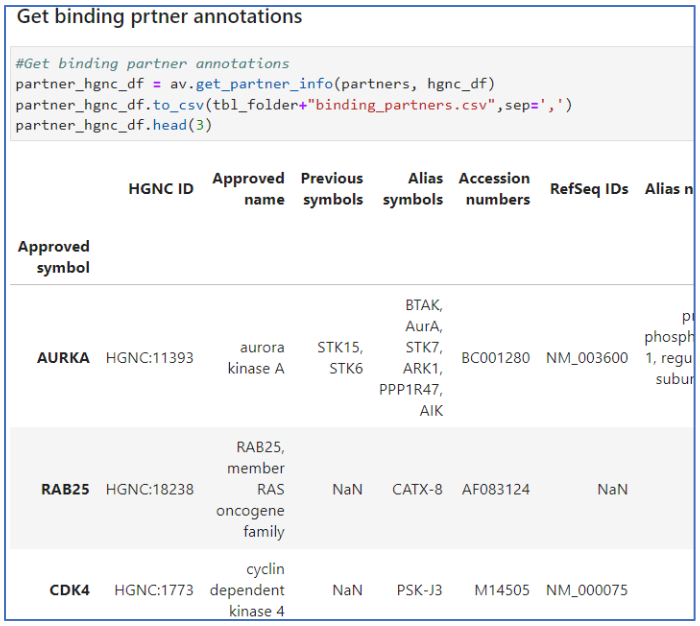


or provide binding partners manually

```
#Enter partners manually
partners = ['AURKA', 'CDK4', 'RAB25']
for partner in partners:
    if partner not in genes_with_expression:
        print(partner,"not in MUT/EXP dataset. Check gene name.")
params['partners']=partners
```

If a binding partner lacks mRNA expression data in the database, a message stating "The gene is not in the MUT/EXP dataset. Check gene name" will be displayed. Proceed to execute the subsequent cell to view detailed information on the first five rows of the binding partners.

4.3.1. The execution of #Get binding partner annotation cell will automatically annotate the proteins based on the information available from the HGNC database. These data will be saved to the Project/Tables folder:

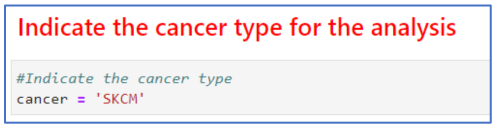


Get binding prtner annotations

```
#Get binding partner annotations
partner_hgnc_df = av.get_partner_info(partners, hgnc_df)
partner_hgnc_df.to_csv(tbl_folder+"binding_partners.csv",sep=',')
partner_hgnc_df.head(3)
```

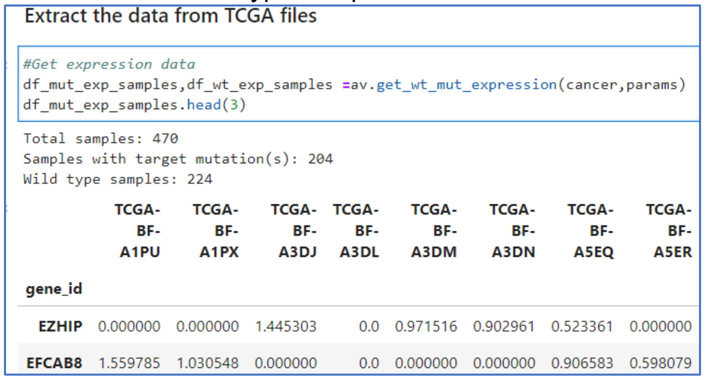
	HGNC ID	Approved name	Previous symbols	Alias symbols	Accession numbers	RefSeq IDs	Alias n
Approved symbol							
AURKA	HGNC:11393	aurora kinase A	STK15, STK6	BTAK, AurA, STK7, ARK1, PPP1R47, AIK	BC001280	NM_003600	p phosph 1, regu subur
RAB25	HGNC:18238	RAB25, member RAS oncogene family	NaN	CATX-8	AF083124	NaN	
CDK4	HGNC:1773	cyclin dependent kinase 4	NaN	PSK-J3	M14505	NM_000075	

4.3.2. Execute the #Indicate the cancer type cell to define the cancer type for the analysis:


Indicate the cancer type for the analysis

```
#Indicate the cancer type
cancer = 'SKCM'
```

4.3.3. The execution of the #Get expression data cell will extract and prepare mRNA expression data for the mutant and the wild-type samples:


Extract the data from TCGA files

```
#Get expression data
df_mut_exp_samples,df_wt_exp_samples =av.get_wt_mut_expression(cancer,params)
df_mut_exp_samples.head(3)
```

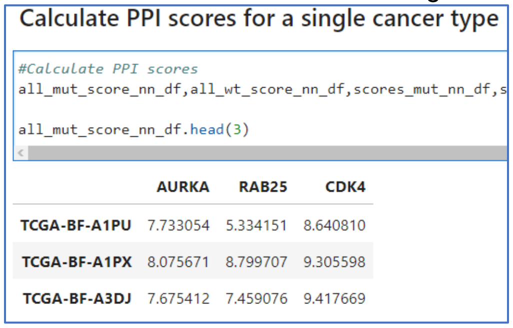
```
Total samples: 470
Samples with target mutation(s): 204
Wild type samples: 224
```

	TCGA-BF-A1PU	TCGA-BF-A1PX	TCGA-BF-A3DJ	TCGA-BF-A3DL	TCGA-BF-A3DM	TCGA-BF-A3DN	TCGA-BF-A5EQ	TCGA-BF-A5ER
gene_id								
EZHIP	0.000000	0.000000	1.445303	0.0	0.971516	0.902961	0.523361	0.000000
EFCAB8	1.559785	1.030548	0.000000	0.0	0.000000	0.000000	0.906583	0.598079

4.4. Evaluation of neoPPI levels

4.4.1. Calculate PPI scores for a single cancer type

To calculate the PPI scores for a single cancer type, execute #Calculate PPI scores cell:

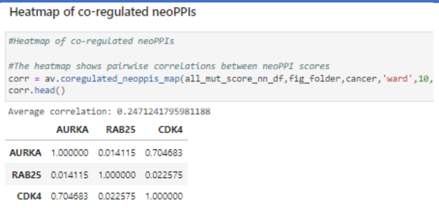

Calculate PPI scores for a single cancer type

```
#Calculate PPI scores
all_mut_score_nn_df,all_wt_score_nn_df,scores_mut_nn_df,s

all_mut_score_nn_df.head(3)
```

	AURKA	RAB25	CDK4
TCGA-BF-A1PU	7.733054	5.334151	8.640810
TCGA-BF-A1PX	8.075671	8.799707	9.305598
TCGA-BF-A3DJ	7.675412	7.459076	9.417669

4.4.2. Determine co-regulated neoPPIs

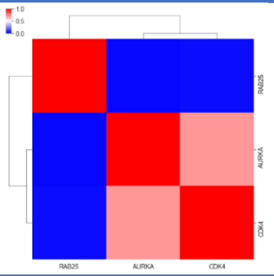
The next cell will conduct the analysis of co-regulated neoPPIs, with significant correlation between their PPI scores in mutant samples:


Heatmap of co-regulated neoPPIs

```
#Heatmap of co-regulated neoPPIs

#The heatmap shows pairwise correlations between neoPPI scores
corr = av.coregulated_neoppis_map(all_mut_score_nn_df,fig_folder,cancer,'ward',10,
corr.head()
```

```
Average correlation: 0.2471241795981188
```

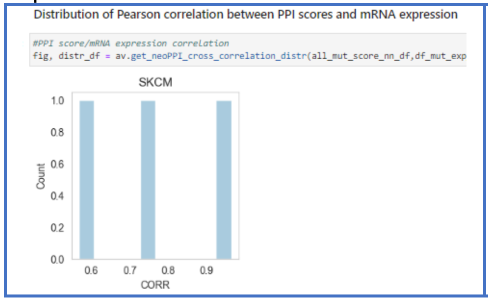
	AURKA	RAB25	CDK4
AURKA	1.000000	0.014115	0.704683
RAB25	0.014115	1.000000	0.022575
CDK4	0.704683	0.022575	1.000000



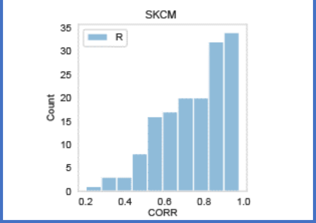
The heatmap shows the lack of correlation between the PPI scores obtained for BRAF V600E neoPPIs with RAB25 and AURKA as well as RAB25 and CDK5. Meanwhile, there is more prominent correlation of 0.7 between BRAF V600E neoPPI with AURKA and CDK4.

4.4.3. Determine PPI score/mRNA expression correlation

In some cases, it can be informative to determine the overall correlation between PPI scores and expression of the binding partners. The #PPI score/mRNA expression correlation cell shows the distribution of Pearson correlation between PPI scores and mRNA expression. The examples below show the histograms obtained for BRAF neoPPIs with AURKA, CDK4, and RAB25 (left) and an expanded set of 130 BRAF V600E neoPPIs (right).

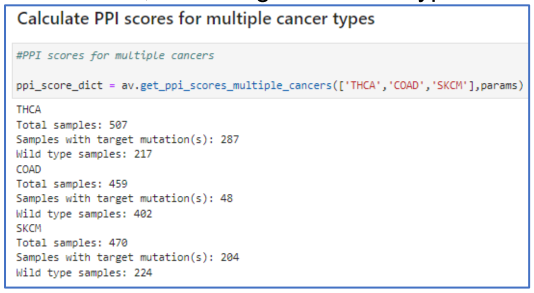

Distribution of Pearson correlation between PPI scores and mRNA expression

```
#PPI score/mRNA expression correlation
fig, distr_df = av.get_neoPPI_cross_correlation_distr(all_mut_score_nn_df,df_mut_exp
```



4.4.4. Calculation of PPI scores across multiple cancer types

To calculate the PPI scores simultaneously in multiple cancer type, execute the #PPI scores for multiple cancers cell, indicating the cancer types as the standard TCGA four-letter abbreviations:

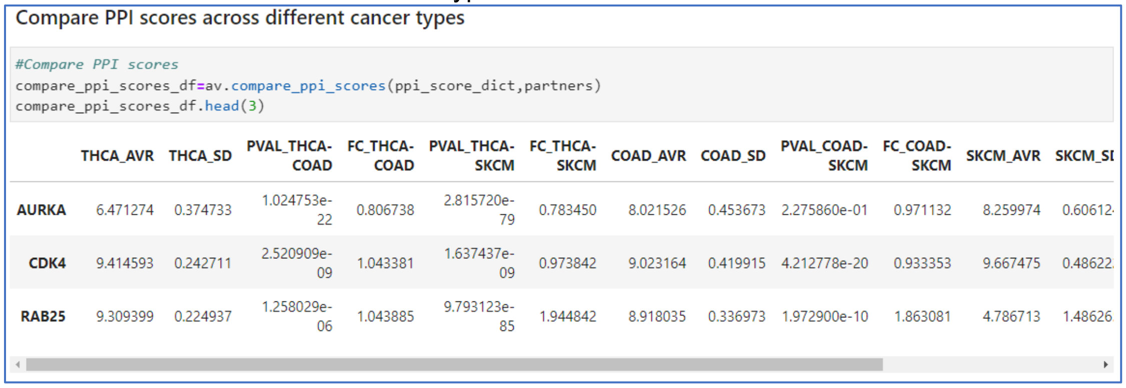

Calculate PPI scores for multiple cancer types

```
#PPI scores for multiple cancers

ppi_score_dict = av.get_ppi_scores_multiple_cancers(['THCA','COAD','SKCM'],params)
```

```
THCA
Total samples: 507
Samples with target mutation(s): 287
Wild type samples: 217
COAD
Total samples: 459
Samples with target mutation(s): 48
Wild type samples: 402
SKCM
Total samples: 470
Samples with target mutation(s): 204
Wild type samples: 224
```

4.4.5. Comparison of neoPPI levels

i. A subsequent execution of #Compare PPI scores will provide a table with statistical difference in PPI scores calculated for different cancer types.


Compare PPI scores across different cancer types

```
#Compare PPI scores
compare_ppi_scores_df=av.compare_ppi_scores(ppi_score_dict,partners)
compare_ppi_scores_df.head(3)
```

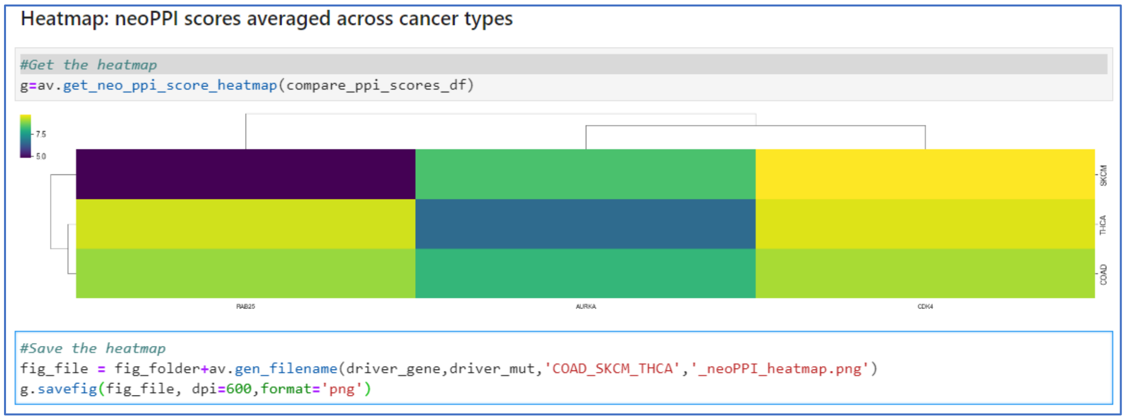
	THCA_AVR	THCA_SD	PVAL_THCA-COAD	FC_THCA-COAD	PVAL_THCA-SKCM	FC_THCA-SKCM	COAD_AVR	COAD_SD	PVAL_COAD-SKCM	FC_COAD-SKCM	SKCM_AVR	SKCM_SI
AURKA	6.471274	0.374733	1.024753e-22	0.806738	2.815720e-79	0.783450	8.021526	0.453673	2.275860e-01	0.971132	8.259974	0.60612
CDK4	9.414593	0.242711	2.520909e-09	1.043381	1.637437e-09	0.973842	9.023164	0.419915	4.212778e-20	0.933353	9.667475	0.48622
RAB25	9.309399	0.224937	1.258029e-06	1.043885	9.793123e-85	1.944842	8.918035	0.336973	1.972900e-10	1.863081	4.786713	1.48626

Below is the description of the columns in the table:

Column	Description
THCA_AVR, COAD_AVR, SKCM_AVR	Average PPI scores calculated for thyroid, colon, and skin cancer samples
THCA_SD, COAD_SD, SKCM_SD	Standard deviation of the PPI scores calculated for thyroid, colon, and skin cancer samples
FC_THCA-COAD, FC_THCA-SKCM, FC_COAD-SKCM	Fold change values calculated as the ratio of mean PPI scores
PVAL_THCA-COAD, PVAL_THCA-SKCM, PVAL_COAD-SKCM	Post hoc pairwise Dunn's test p-values

Column	Description
KW_H_Test_PVAL	Kruskal-Wallis test p-value
KW_H_Test_QVAL	The FDR-adjusted Kruskal-Wallis test p-value (q-value)

ii. Execute #Save it! cell to save the table to the Project/Tables folder.
iii. The #Get the heatmap cell will deliver a heatmap where a brighter color signifies a higher average neoPPI score. #Save the heatmap cell will save the heatmap image to the Project/Figures folder.

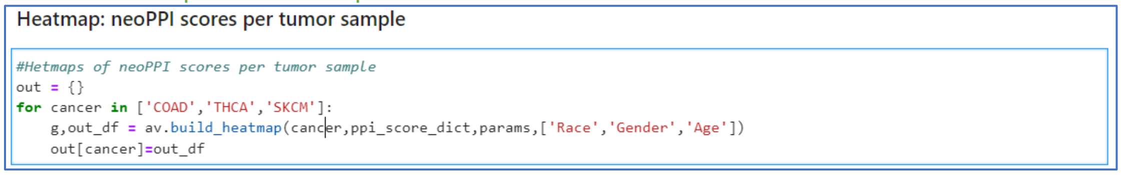


Heatmap: neoPPI scores averaged across cancer types

```
#Get the heatmap
g=av.get_neo_ppi_score_heatmap(compare_ppi_scores_df)
```

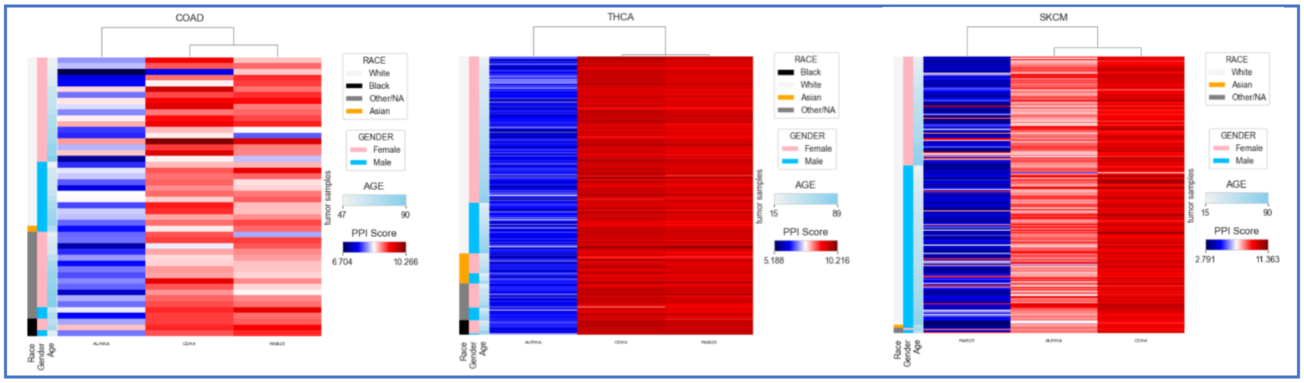
```
#Save the heatmap
fig_file = fig_folder+av.gen_filename(driver_gene,driver_mut,'COAD_SKCM_THCA','_neoPPI_heatmap.png')
g.savefig(fig_file, dpi=600,format='png')
```

iv. The AVERON also allows the analysis of PPI scores in individual tumor samples, showcasing the average neoPPI scores, and annotate this data with patients' clinical information, including race, gender, age, and mutant status, in the form of heatmaps. To conduct the analysis, execute the #Hetmaps of neoPPI scores per tumor sample cell.

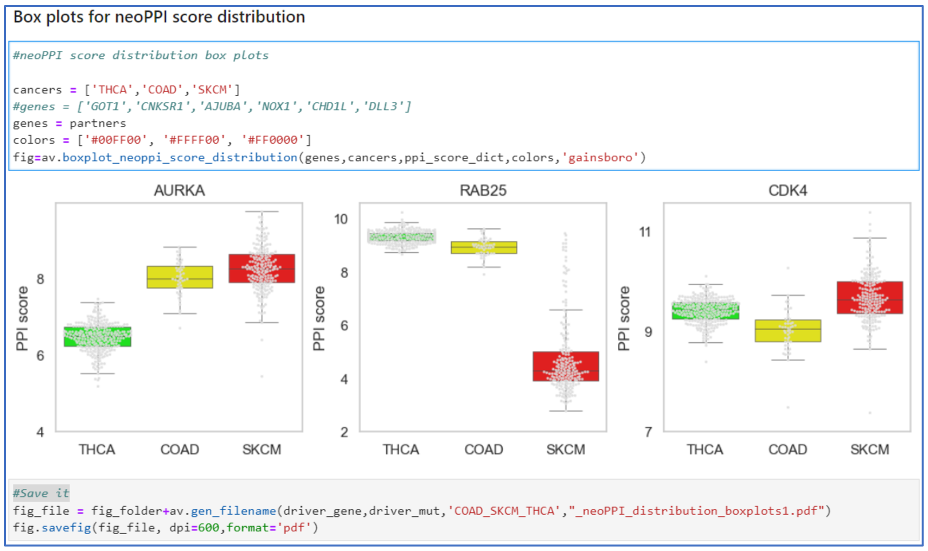


Heatmap: neoPPI scores per tumor sample

```
#Hetmaps of neoPPI scores per tumor sample
out = {}
for cancer in ['COAD','THCA','SKCM']:
    g,out_df = av.build_heatmap(cancer,ppi_score_dict,params,['Race','Gender','Age'])
    out[cancer]=out_df
```



v. Execute the #neoPPI score distribution box plots cell to visualize the average neoPPI scores across various cancer types for multiple genes through box plots. These plots provide insights into the distribution and variation of neoPPI scores within and between different cancers. The colors of the box plots can be specified in the colors list. To retain these box plots for future reference, proceed to run the following #Save it cell for saving.

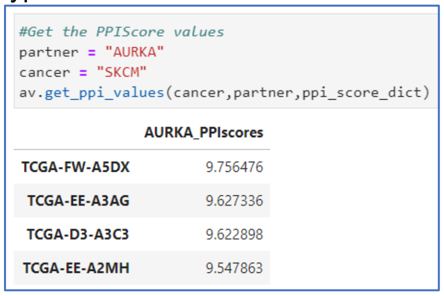


Box plots for neoPPI score distribution

```
#neoPPI score distribution box plots

cancers = ['THCA','COAD','SKCM']
#genes = ['GOT1','CNKSR1','AJUBA','NOX1','CHD1L','DLL3']
genes = partners
colors = ['#00FF00', '#FFFF00', '#FF0000']
fig=av.boxplot_neoppi_score_distribution(genes,cancers,ppi_score_dict,colors,'gainsboro')
```

```
#Save it
fig_file = fig_folder+av.gen_filename(driver_gene,driver_mut,'COAD_SKCM_THCA',"_neoPPI_distribution_boxplots1.pdf")
fig.savefig(fig_file, dpi=600,format='pdf')
```

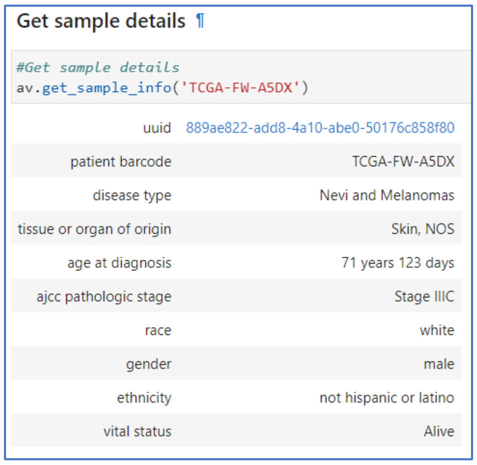
vi. The exact PPI score values per tumor sample can be extracted by running #Get the PPIScore values cell. Use *partner* variable to indicate a particular ne binding partner, and *cancer* to indicate the cancer type:



```
#Get the PPIScore values
partner = "AURKA"
cancer = "SKCM"
av.get_ppi_values(cancer,partner,ppi_score_dict)
```

	AURKA_PPIscores
TCGA-FW-A5DX	9.756476
TCGA-EE-A3AG	9.627336
TCGA-D3-A3C3	9.622898
TCGA-EE-A2MH	9.547863

vii. #Get sample details cell will provide the detailed metadata associated with individual sample and a direct link to the GDC DataPortal for further exploration:



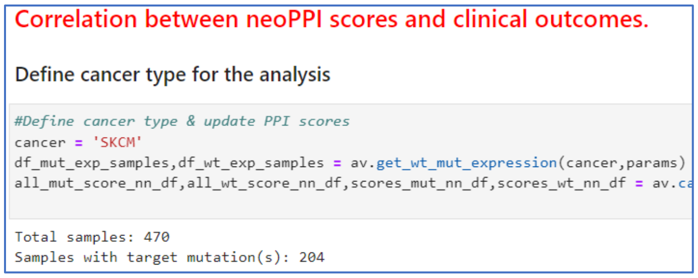
Get sample details ¶

```
#Get sample details
av.get_sample_info('TCGA-FW-A5DX')
```

uuid	889ae822-add8-4a10-abe0-50176c858f80
patient barcode	TCGA-FW-A5DX
disease type	Nevi and Melanomas
tissue or organ of origin	Skin, NOS
age at diagnosis	71 years 123 days
ajcc pathologic stage	Stage IIIC
race	white
gender	male
ethnicity	not hispanic or latino
vital status	Alive

4.5. Identification of clinically significant neoPPIs

4.5.1. The next section allows us to determine the correlations between neoPPI levels and clinical outcomes of cancer patients. It is recommended to reset the cancer type and update the PPI scores prior to the survival analysis by executing the #Define cancer type & update PPI scores cell.



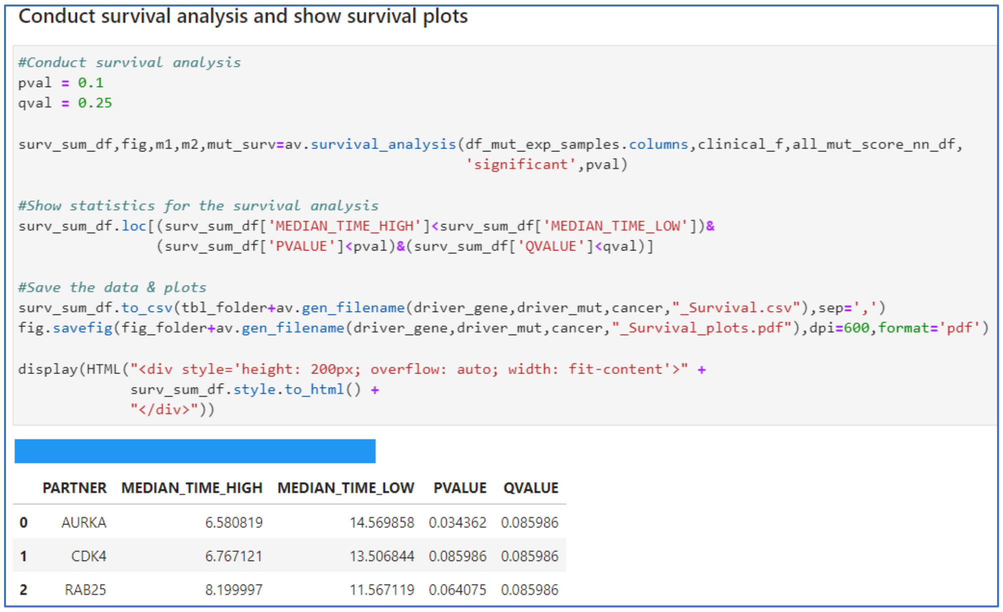
Correlation between neoPPI scores and clinical outcomes.

Define cancer type for the analysis

```
#Define cancer type & update PPI scores
cancer = 'SKCM'
df_mut_exp_samples,df_wt_exp_samples = av.get_wt_mut_expression(cancer,params)
all_mut_score_nn_df,all_wt_score_nn_df,scores_mut_nn_df,scores_wt_nn_df = av.ca
```

```
Total samples: 470
Samples with target mutation(s): 204
```

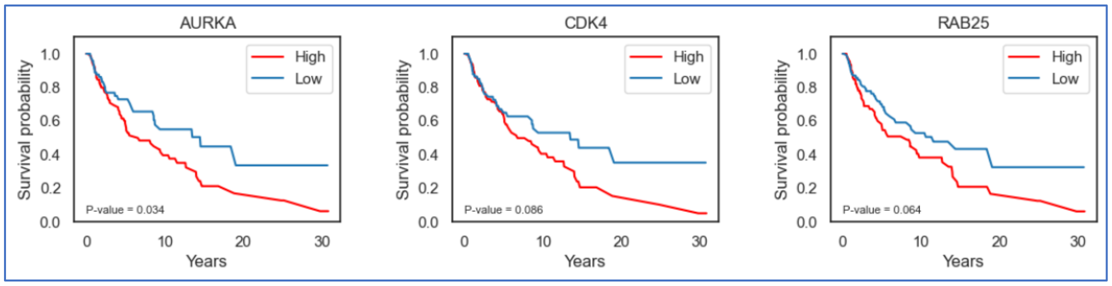
4.5.2. A subsequent execution of #Conduct survival analysis cell will perform a survival analysis for neoPPIs to find out which neoPPI can contribute to decreased clinical outcomes.



Conduct survival analysis and show survival plots

```
#Conduct survival analysis
pval = 0.1
qval = 0.25

surv_sum_df,fig,m1,m2,mut_surv=av.survival_analysis(df_mut_exp_samples.columns,clinical_f,all_mut_score_nn_df,
                                                    'significant',pval)

#Show statistics for the survival analysis
surv_sum_df.loc[(surv_sum_df['MEDIAN_TIME_HIGH']<surv_sum_df['MEDIAN_TIME_LOW'])&
                (surv_sum_df['PVALUE']<pval)&(surv_sum_df['QVALUE']<qval)]

#Save the data & plots
surv_sum_df.to_csv(tbl_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_Survival.csv"),sep=',')
fig.savefig(fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_Survival_plots.pdf"),dpi=600,format='pdf')

display(HTML("<div style='height: 200px; overflow: auto; width: fit-content'>" +
             surv_sum_df.style.to_html() +
             "</div>"))
```

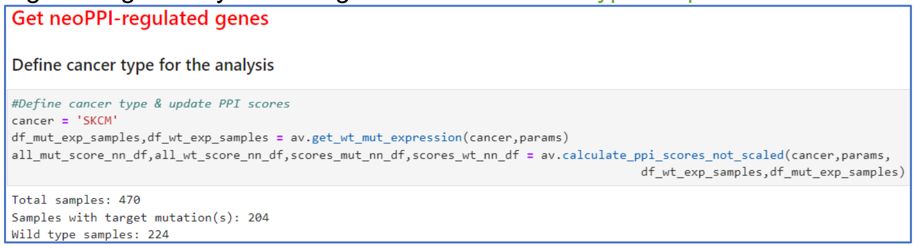
	PARTNER	MEDIAN_TIME_HIGH	MEDIAN_TIME_LOW	PVALUE	QVALUE
0	AURKA	6.580819	14.569858	0.034362	0.085986
1	CDK4	6.767121	13.506844	0.085986	0.085986
2	RAB25	8.199997	11.567119	0.064075	0.085986



The *pval* and *qval* variables control the p-value and the FDR-adjusted p-value (q-value) thresholds for statistical significance.
The plots and statistics will be automatically saved to the Project/Figures and Project/Tables folders, respectively.

4.6. Identification of neoPPI-regulated genes

It is recommended to reset the cancer type and update the PPI scores prior to the analysis of neoPPI-regulated genes by executing the #Define cancer type & update PPI scores cell.



Get neoPPI-regulated genes

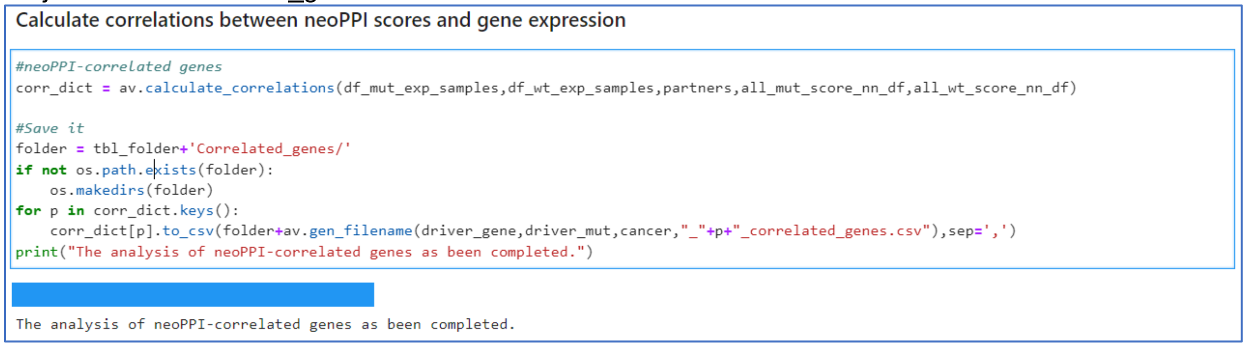
Define cancer type for the analysis

```
#Define cancer type & update PPI scores
cancer = 'SKCM'
df_mut_exp_samples,df_wt_exp_samples = av.get_wt_mut_expression(cancer,params)
all_mut_score_nn_df,all_wt_score_nn_df,scores_mut_nn_df,scores_wt_nn_df = av.calculate_ppi_scores_not_scaled(cancer,params,
                                                                                                               df_wt_exp_samples,df_mut_exp_samples)
```

```
Total samples: 470
Samples with target mutation(s): 204
Wild type samples: 224
```

4.6.1. neoPPI-correlated genes.

NeoPPI-correlated genes are genes, whose expression correlates with the neoPPI scores. Run the #neoPPI-correlated genes cell to determine the correlation between neoPPI scores and gene expression in mutant samples. The results will be automatically saved to the Project/Table/Correlated_genes folder.

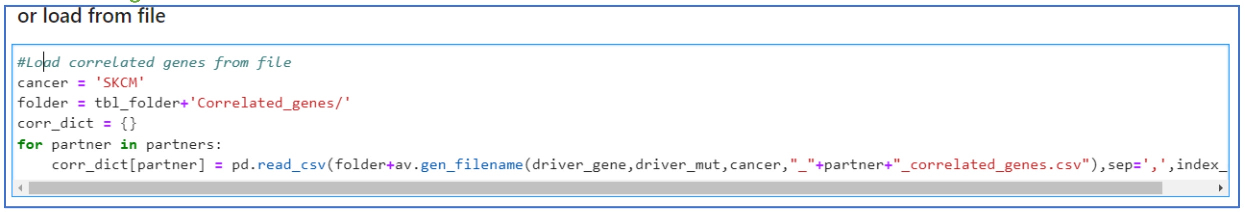


Calculate correlations between neoPPI scores and gene expression

```
#neoPPI-correlated genes
corr_dict = av.calculate_correlations(df_mut_exp_samples,df_wt_exp_samples,partners,all_mut_score_nn_df,all_wt_score_nn_df)

#Save it
folder = tbl_folder+'Correlated_genes/'
if not os.path.exists(folder):
    os.makedirs(folder)
for p in corr_dict.keys():
    corr_dict[p].to_csv(folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+p+"_correlated_genes.csv"),sep=',')
print("The analysis of neoPPI-correlated genes as been completed.")
```

```
The analysis of neoPPI-correlated genes as been completed.
```

Alternatively, previously calculated neoPPI-correlated genes can be loaded by executing the #Load correlated genes from file cell:

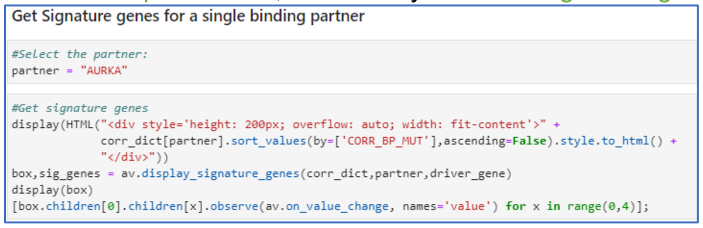


or load from file

```
#Load correlated genes from file
cancer = 'SKCM'
folder = tbl_folder+'Correlated_genes/'
corr_dict = {}
for partner in partners:
    corr_dict[partner] = pd.read_csv(folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+partner+"_correlated_genes.csv"),sep=',',index_
```

4.6.2. Determining signature genes for individual neoPPI.

In general, neoPPI-signature genes (or neoPPI-regulated genes) are genes whose expression correlates with the PPI scores in mutant samples significantly stronger then in wild-type samples. To determine the signature genes, regulated by a neoPPI, first indicate the neoPPI partner by executing #Select the partner cell, followed by the #Get signature genes cell.



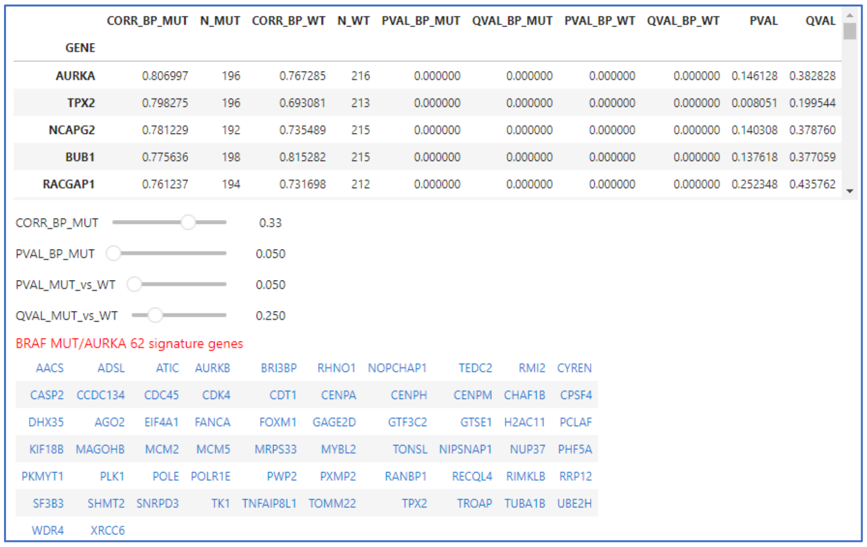
Get Signature genes for a single binding partner

```
#Select the partner:
partner = "AURKA"
```

```
#Get signature genes
display(HTML("<div style='height: 200px; overflow: auto; width: fit-content'>" +
             corr_dict[partner].sort_values(by=['CORR_BP_MUT'],ascending=False).style.to_html() +
             "</div>"))
box,sig_genes = av.display_signature_genes(corr_dict,partner,driver_gene)
display(box)
[box.children[0].children[x].observe(av.on_value_change, names='value') for x in range(0,4)];
```

The resulting table will show all calculated statistics for the correlations between gene expression and PPI scores:

Column	**Description**
GENE:	neoPPI-regulated gene
CORR_BP_MUT:	Correlation coefficient between neoPPI score and expression of the neoPPI-regulated gene in mutant samples
N_MUT:	The number of mutant samples used to calculate CORR_BP_MUT
CORR_BP_WT:	Correlation coefficient between neoPPI score and expression of the neoPPI-regulated gene in the wild type samples
N_WT:	The number of mutant samples used to calculate CORR_BP_WT
PVAL_BP_MUT:	P-value of CORR_BP_MUT correlation
QVAL_BP_MUT:	FDR-adjusted P-value of CORR_BP_MUT correlation
PVAL_BP_WT:	P-value of CORR_BP_WT correlation
QVAL_BP_WT:	FDR-adjusted P-value of CORR_BP_WT correlation
PVAL:	P-value of statistical difference between CORR_BP_MUT and CORR_BP_WT
QVAL:	FDR-adjusted P-value of statistical difference between CORR_BP_MUT and CORR_BP_WT



GENE	CORR_BP_MUT	N_MUT	CORR_BP_WT	N_WT	PVAL_BP_MUT	QVAL_BP_MUT	PVAL_BP_WT	QVAL_BP_WT	PVAL	QVAL
AURKA	0.806997	196	0.767285	216	0.000000	0.000000	0.000000	0.000000	0.146128	0.382828
TPX2	0.798275	196	0.693081	213	0.000000	0.000000	0.000000	0.000000	0.008051	0.199544
NCAPG2	0.781229	192	0.735489	215	0.000000	0.000000	0.000000	0.000000	0.140308	0.378760
BUB1	0.775636	198	0.815282	215	0.000000	0.000000	0.000000	0.000000	0.137618	0.377059
RACGAP1	0.761237	194	0.731698	212	0.000000	0.000000	0.000000	0.000000	0.252348	0.435762

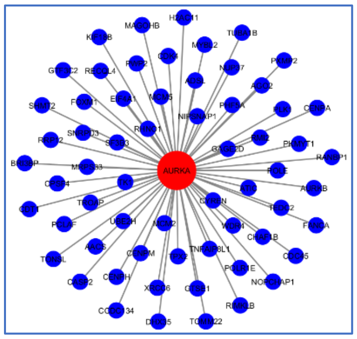
CORR_BP_MUT	0.33
PVAL_BP_MUT	0.050
PVAL_MUT_vs_WT	0.050
QVAL_MUT_vs_WT	0.250

BRAF MUT/AURKA 62 signature genes

AACS	ADSL	ATIC	AURKB	BRI3BP	RHNO1	NOPCHAP1	TEDC2	RMI2	CYREN
CASP2	CCDC134	CDC45	CDK4	CDT1	CENPA	CENPH	CENPM	CHAF1B	CPSF4
DHX35	AGO2	EIF4A1	FANCA	FOXM1	GAGE2D	GTF3C2	GTSE1	H2AC11	PCLAF
KIF18B	MAGOHB	MCM2	MCM5	MRPS33	MYBL2	TONSL	NIPSNAP1	NUP37	PHF5A
PKMYT1	PLK1	POLE	POLR1E	PWP2	PXMP2	RANBP1	RECQL4	RIMKLB	RRP12
SF3B3	SHMT2	SNRPD3	TK1	TNFAIP8L1	TOMM22	TPX2	TROAP	TUBA1B	UBE2H
WDR4	XRCC6								

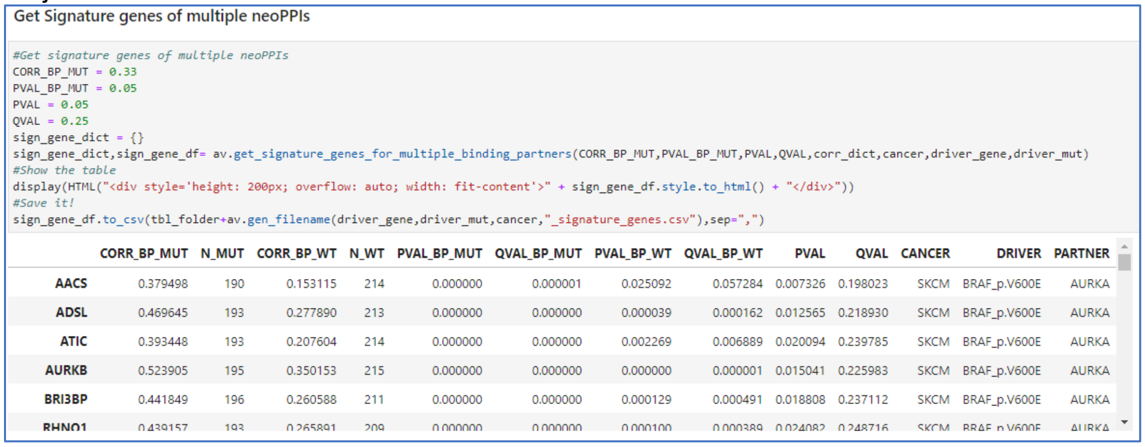
The interaction scrollbars allow us to adjust the statistical thresholds. Each signature gene within the display is linked to the HGNC portal (https://www.genenames.org) for comprehensive gene information.

To visualize the signature gene network, run the Cytoscape application and then execute the #Network of signature genes. The network will appear in the Notebook as an image and in the Cytoscape app as an interactive network.



4.6.3. Determine neoPPI-regulated genes for multiple neoPPIs

i. The #Get signature genes of multiple neoPPIs cells allows us to adjust the statistical thresholds and run the identification of signature genes for all defined binding partners used at step 3.12. The resulting table with identified genes and associated statistical parameters will be shown and saved to the Project/Tables folder:



Get Signature genes of multiple neoPPIs

```
#Get signature genes of multiple neoPPIs
CORR_BP_MUT = 0.33
PVAL_BP_MUT = 0.05
PVAL = 0.05
QVAL = 0.25
sign_gene_dict = {}
sign_gene_dict,sign_gene_df= av.get_signature_genes_for_multiple_binding_partners(CORR_BP_MUT,PVAL_BP_MUT,PVAL,QVAL,corr_dict,cancer,driver_gene,driver_mut)
#Show the table
display(HTML("<div style='height: 200px; overflow: auto; width: fit-content'>" + sign_gene_df.style.to_html() + "</div>"))
#Save it!
sign_gene_df.to_csv(tbl_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_signature_genes.csv"),sep=",")
```

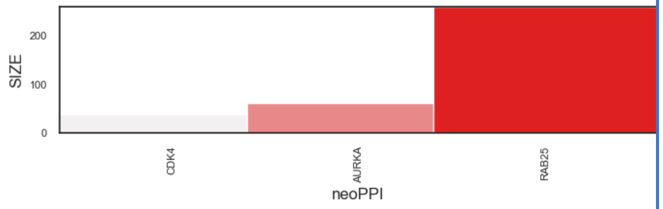
	CORR_BP_MUT	N_MUT	CORR_BP_WT	N_WT	PVAL_BP_MUT	QVAL_BP_MUT	PVAL_BP_WT	QVAL_BP_WT	PVAL	QVAL	CANCER	DRIVER	PARTNER
AACS	0.379498	190	0.153115	214	0.000000	0.000001	0.025092	0.057284	0.007326	0.198023	SKCM	BRAF_p.V600E	AURKA
ADSL	0.469645	193	0.277890	213	0.000000	0.000000	0.000039	0.000162	0.012565	0.218930	SKCM	BRAF_p.V600E	AURKA
ATIC	0.393448	193	0.207604	214	0.000000	0.000000	0.002269	0.006889	0.020094	0.239785	SKCM	BRAF_p.V600E	AURKA
AURKB	0.523905	195	0.350153	215	0.000000	0.000000	0.000000	0.000001	0.015041	0.225983	SKCM	BRAF_p.V600E	AURKA
BRI3BP	0.441849	196	0.260588	211	0.000000	0.000000	0.000129	0.000491	0.018808	0.237112	SKCM	BRAF_p.V600E	AURKA
RHNO1	0.439157	193	0.265891	209	0.000000	0.000000	0.000100	0.000389	0.024082	0.248716	SKCM	BRAF_p.V600E	AURKA

ii. The comparison of the size and overlap between the signature genes determined for different neoPPIs is provided by #Size distribution of neoPPI-regulated gene sets and #neoPPI-signature gene set overlap, JACCARD score cells, that determine and visualize the size of different neoPPI-signature gene sets, shows the similarity heatmap, and provide the Jaccard-score, as a metric of overall similarity between the signature gene sets.

Distribution of the size of neoPPI-regulated gene sets

```
#Size distribution of neoPPI-regulated gene sets
fig,stat_df = av.neoPPI_genes_distr(sign_gene_dict,'red',8,8)
fig.savefig(fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,
                    "_signature_genes_hist.pdf"),dpi=600,format="pdf")
stat_df.sort_values(by='SIZE',ascending=False).head(10)
```

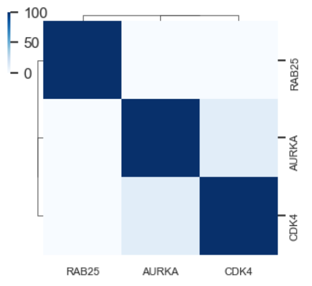
	neoPPI	SIZE
2	RAB25	260
1	AURKA	62
0	CDK4	37



Heatmap: neoPPI-regulated gene set similarity; Jaccard score

```
#neoPPI-signature gene set overlap, JACCARD score
g=av.jaccard(sign_gene_dict,fig_folder,'Blues',8,8)
g.savefig(fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,
                    "_jaccard_heatmap1.pdf"),dpi=600,format='pdf')
```

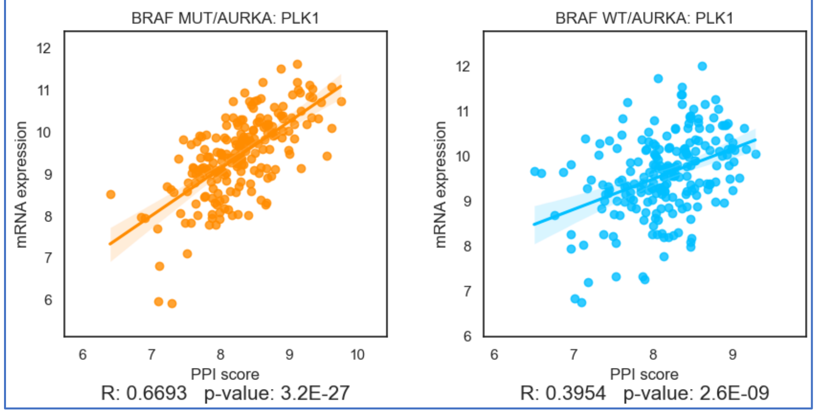
```
Average Jaccard Index:  3.7 %
```



iii. The correlation between PPI-scores and mRNA expression of a signature gene in mutant and the wild-type samples can be visualized by executing #PPI-score/mRNA expression correlation cell. The *partner* and *gene* variables define the neoPPI binding partner and the signature gene, respectively. The correlation plots will be automatically saved to the Project/Figures folder.

Regression plots for a specific signature gene in Mut and WT samples

```
#PPI-score/mRNA expression correaltion
partner='AURKA'
gene = 'PLK1'
fig = av.ppi_score_mrna_corr(gene,driver_gene,partner,all_mut_score_nn_df,all_wt_score_nn_df,
                             df_mut_exp_samples,df_wt_exp_samples,'darkorange','deepskyblue')
#Save it
fig_file = fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+partner+"_"+gene+"_correlattion_plot.png")
fig.savefig(fig_file, dpi=600,format='png')
```



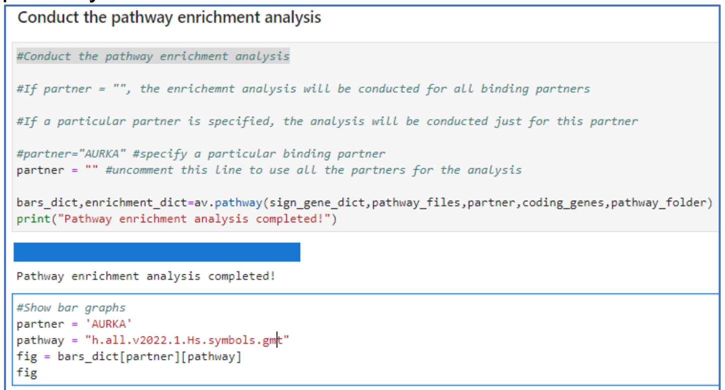
4.6.4. Enrichment analysis.

i. The AVERON enables pathway enrichment analysis for the neoPPI signature genes. The following steps should be done to conduct the analysis. First, the reference gene sets should be defined in the #Define the pathway gene sets to analyze cell. In this example, we use "h.all.v2022.1.Hs.symbols.gmt", "c2.cp.kegg.v2022.1.Hs.symbols.gmt", and "c2.cp.reactome.v2022.1.Hs.symbols.gmt" sets defined in the Molecular Signature Database (MSigDB). The corresponding reference GMT dataset files are located in the AVERON\input\pathway_genesets\MSigDB folder, defined by *pathway_folder = data_folder+"pathway_genesets/MSigDB/"* variable.

Define the pathways gene sets to analyze

```
#Define the pathway gene sets to analyze

#Currently Averon uses genesets defined in MSigDB: https://www.gsea-msigdb.org/gsea/msigdb
pathway_files = ["h.all.v2022.1.Hs.symbols.gmt", "c2.cp.kegg.v2022.1.Hs.symbols.gmt", "c2.cp.reactome.v2022.1.Hs.symbols.gmt"]
params['pathway_files']=pathway_files
```

ii. A consequent execution of #Conduct the pathway enrichment analysis and #Show bar graphs cells will conduct the analysis and visualize the results as bar graphs for specified neo-binding partner and the pathway set:

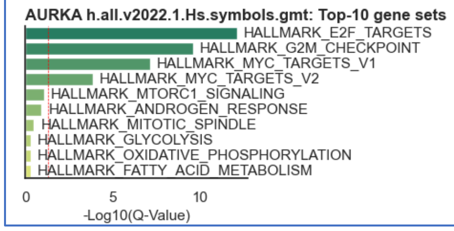


Conduct the pathway enrichment analysis

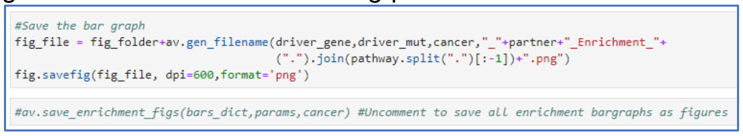
```
#Conduct the pathway enrichment analysis

#If partner = "", the enrichemnt analysis will be conducted for all binding partners

#If a particular partner is specified, the analysis will be conducted just for this partner

#partner="AURKA" #specify a particular binding partner
partner = "" #uncomment this line to use all the partners for the analysis

bars_dict,enrichment_dict=av.pathway(sign_gene_dict,pathway_files,partner,coding_genes,pathway_folder)
print("Pathway enrichment analysis completed!")
```

```
Pathway enrichment analysis completed!
```

```
#Show bar graphs
partner = 'AURKA'
pathway = "h.all.v2022.1.Hs.symbols.gmt"
fig = bars_dict[partner][pathway]
fig
```



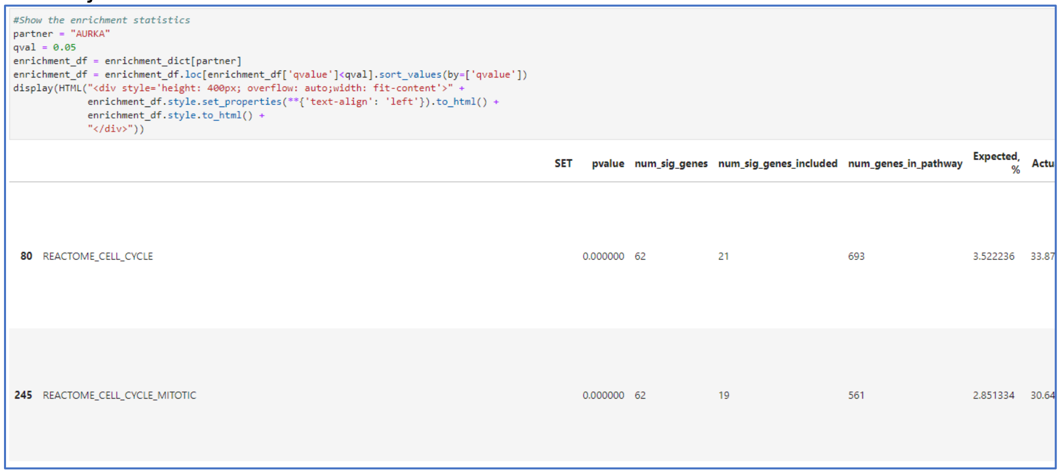
iii. The individual enrichment plots can be save by executing the #Save the bar graph cell. The uncommenting and execution of the next cell will enable an automated saving of all enrichment plots generated for all defined binding partners:



```
#Save the bar graph
fig_file = fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+partner+"_Enrichment_"+
                                       (".").join(pathway.split(".")[:-1])+".png")
fig.savefig(fig_file, dpi=600,format='png')
```

```
#av.save_enrichment_figs(bars_dict,params,cancer) #Uncomment to save all enrichment bargraphs as figures
```

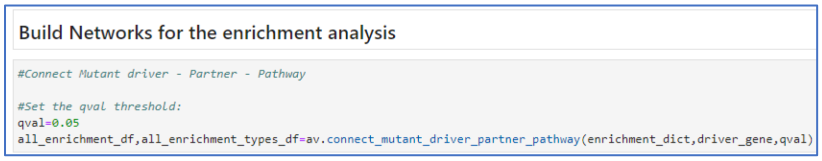
iv. The resulting statistics of the enrichment analysis can be obtained and saved by running the #Show the enrichment statistics cell. The *qval* variable defines the threshold for statistical significance, and *partner* variable indicates for which neo-binding partner the results will be shown. Note that the enrichment analysis results obtained for signature genes of all neoPPIs will be automatically saved to the Project/Tables folder.



```
#Show the enrichment statistics
partner = "AURKA"
qval = 0.05
enrichment_df = enrichment_dict[partner]
enrichment_df = enrichment_df.loc[enrichment_df['qvalue']<qval].sort_values(by=['qvalue'])
display(HTML("<div style='height: 400px; overflow: auto;width: fit-content'>" +
             enrichment_df.style.set_properties(**{'text-align': 'left'}).to_html() +
             enrichment_df.style.to_html() +
             "</div>"))
```

	SET	pvalue	num_sig_genes	num_sig_genes_included	num_genes_in_pathway	Expected, %	Actu
80	REACTOME_CELL_CYCLE	0.000000	62	21	693	3.522236	33.87
245	REACTOME_CELL_CYCLE_MITOTIC	0.000000	62	19	561	2.851334	30.64

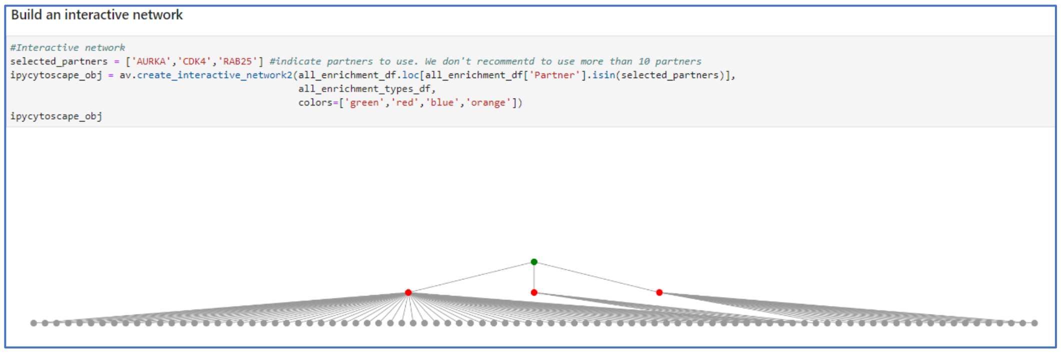
v. It is also possible to visualize and export the enrichment analysis as a network. For this purpose, first the #Connect Mutant driver - Partner – Pathway cell should be executed to prepare the network.



Build Networks for the enrichment analysis

```
#Connect Mutant driver - Partner - Pathway

#Set the qval threshold:
qval=0.05
all_enrichment_df,all_enrichment_types_df=av.connect_mutant_driver_partner_pathway(enrichment_dict,driver_gene,qval)
```

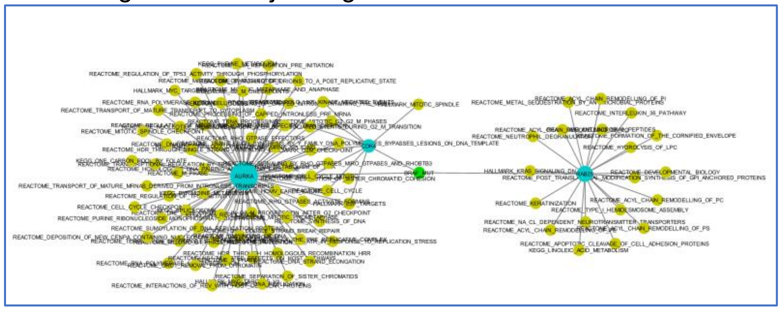
Then either an interactive network within the Notebook can be generated by the #Interactive network cell:



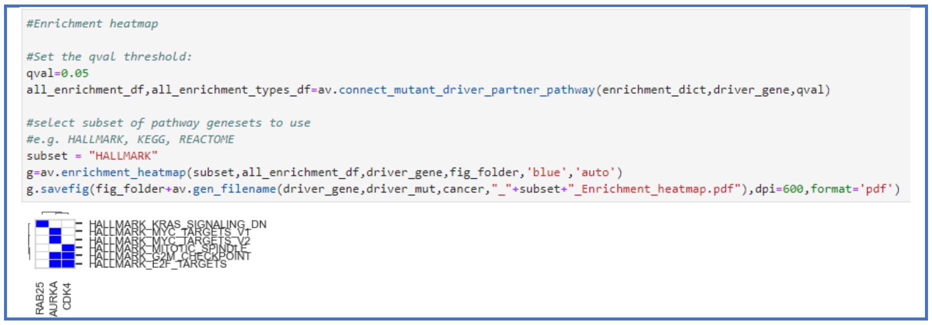
Build an interactive network

```
#Interactive network
selected_partners = ['AURKA','CDK4','RAB25'] #indicate partners to use. We don't recommentd to use more than 10 partners
ipycytoscape_obj = av.create_interactive_network2(all_enrichment_df.loc[all_enrichment_df['Partner'].isin(selected_partners)],
                                                  all_enrichment_types_df,
                                                  colors=['green','red','blue','orange'])
ipycytoscape_obj
```

or a Cytoscape network can be built by executing #Cytoscape network cell. Note that the Cytoscape application should be run prior the Cytoscape network generation. The Cytoscape network will be save as an image to the Project/Figures folder and as a SIF file to the Project/Networks folder.

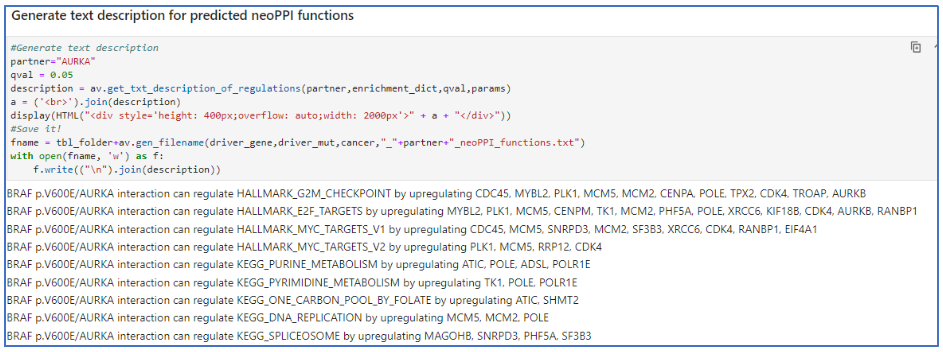


vi. The enrichment analysis can be also represented as a heatmap by running the #Enrichment heatmap cell:



```
#Enrichment heatmap

#Set the qval threshold:
qval=0.05
all_enrichment_df,all_enrichment_types_df=av.connect_mutant_driver_partner_pathway(enrichment_dict,driver_gene,qval)

#select subset of pathway genesets to use
#e.g. HALLMARK, KEGG, REACTOME
subset = "HALLMARK"
g=av.enrichment_heatmap(subset,all_enrichment_df,driver_gene,fig_folder,'blue','auto')
g.savefig(fig_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+subset+"_Enrichment_heatmap.pdf"),dpi=600,format='pdf')
```

vii. The "Generate text description for predicted neoPPI functions" option allows a user to generate a descriptive summary of how different neoPPIs can regulate different biological pathways through their signature genes. To generate the description, execute the #Generate text description cell:


Generate text description for predicted neoPPI functions

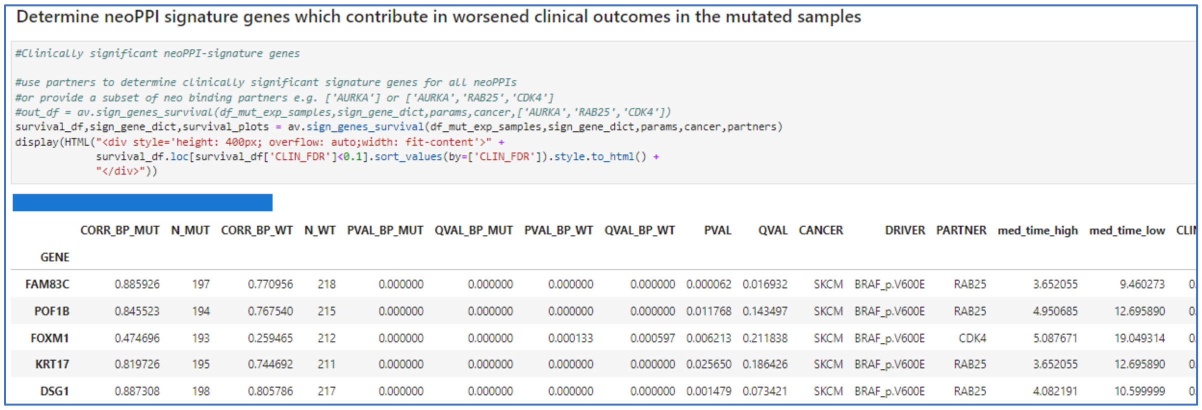
```
#Generate text description
partner="AURKA"
qval = 0.05
description = av.get_txt_description_of_regulations(partner,enrichment_dict,qval,params)
a = ('<br>').join(description)
display(HTML("<div style='height: 400px;overflow: auto;width: 2000px'>" + a + "</div>"))
#Save it!
fname = tbl_folder+av.gen_filename(driver_gene,driver_mut,cancer,"_"+partner+"_neoPPI_functions.txt")
with open(fname, 'w') as f:
    f.write(("\n").join(description))
```

```
BRAF p.V600E/AURKA interaction can regulate HALLMARK_G2M_CHECKPOINT by upregulating CDC45, MYBL2, PLK1, MCM5, MCM2, CENPA, POLE, TPX2, CDK4, TROAP, AURKB
BRAF p.V600E/AURKA interaction can regulate HALLMARK_E2F_TARGETS by upregulating MYBL2, PLK1, MCM5, CENPM, TK1, MCM2, PHF5A, POLE, XRCC6, KIF18B, CDK4, AURKB, RANBP1
BRAF p.V600E/AURKA interaction can regulate HALLMARK_MYC_TARGETS_V1 by upregulating CDC45, MCM5, SNRPD3, MCM2, SF3B3, XRCC6, CDK4, RANBP1, EIF4A1
BRAF p.V600E/AURKA interaction can regulate HALLMARK_MYC_TARGETS_V2 by upregulating PLK1, MCM5, RRP12, CDK4
BRAF p.V600E/AURKA interaction can regulate KEGG_PURINE_METABOLISM by upregulating ATIC, POLE, ADSL, POLR1E
BRAF p.V600E/AURKA interaction can regulate KEGG_PYRIMIDINE_METABOLISM by upregulating TK1, POLE, POLR1E
BRAF p.V600E/AURKA interaction can regulate KEGG_ONE_CARBON_POOL_BY_FOLATE by upregulating ATIC, SHMT2
BRAF p.V600E/AURKA interaction can regulate KEGG_DNA_REPLICATION by upregulating MCM5, MCM2, POLE
BRAF p.V600E/AURKA interaction can regulate KEGG_SPLICEOSOME by upregulating MAGOHB, SNRPD3, PHF5A, SF3B3
```

The results will be saved as a text file to the Project/Table folder.

4.6.5. Determine clinically significant neoPPI-regulated genes

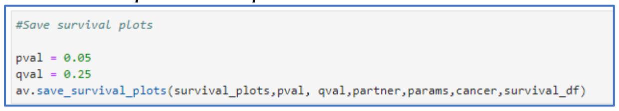
i. The next question that the Averon helps to answer is which of the neoPPI-signature genes may represent clinically important and druggable targets? The execution of #Clinically significant neoPPI-signature genes cell will determine neoPPI signature genes whose high expression in mutant samples correlates with worsened clinical outcomes:


Determine neoPPI signature genes which contribute in worsened clinical outcomes in the mutated samples

```
#Clinically significant neoPPI-signature genes

#use partners to determine clinically significant signature genes for all neoPPIs
#or provide a subset of neo binding partners e.g. ['AURKA'] or ['AURKA','RAB25','CDK4']
#out_df = av.sign_genes_survival(df_mut_exp_samples,sign_gene_dict,params,cancer,['AURKA','RAB25','CDK4'])
survival_df,sign_gene_dict,survival_plots = av.sign_genes_survival(df_mut_exp_samples,sign_gene_dict,params,cancer,partners)
display(HTML("<div style='height: 400px; overflow: auto;width: fit-content'>" +
             survival_df.loc[survival_df['CLIN_FDR']<0.1].sort_values(by=['CLIN_FDR']).style.to_html() +
             "</div>"))
```

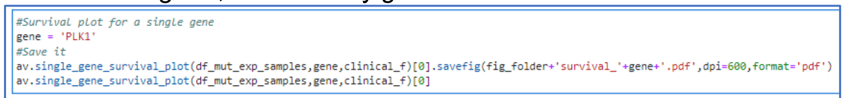
	CORR_BP_MUT	N_MUT	CORR_BP_WT	N_WT	PVAL_BP_MUT	QVAL_BP_MUT	PVAL_BP_WT	QVAL_BP_WT	PVAL	QVAL	CANCER	DRIVER	PARTNER	med_time_high	med_time_low	CLI
GENE																
FAM83C	0.885926	197	0.770956	218	0.000000	0.000000	0.000000	0.000000	0.000062	0.016932	SKCM	BRAF_p.V600E	RAB25	3.652055	9.460273	0
POF1B	0.845523	194	0.767540	215	0.000000	0.000000	0.000000	0.000000	0.011768	0.143497	SKCM	BRAF_p.V600E	RAB25	4.950685	12.695890	0
FOXM1	0.474696	193	0.259465	212	0.000000	0.000000	0.000133	0.000597	0.006213	0.211838	SKCM	BRAF_p.V600E	CDK4	5.087671	19.049314	0
KRT17	0.819726	195	0.744692	211	0.000000	0.000000	0.000000	0.000000	0.025650	0.186426	SKCM	BRAF_p.V600E	RAB25	3.652055	12.695890	0
DSG1	0.887308	198	0.805786	217	0.000000	0.000000	0.000000	0.000000	0.001479	0.073421	SKCM	BRAF_p.V600E	RAB25	4.082191	10.599999	0

ii. The next cell, #Save survival plots will automatically save the survival plot images to the Project/Images folder. The *pval* and *qval* variables will define the statistical thresholds to save the plots.

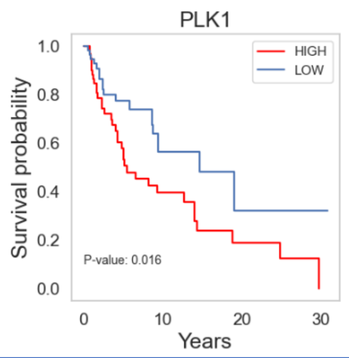


```
#Save survival plots

pval = 0.05
qval = 0.25
av.save_survival_plots(survival_plots,pval, qval,partner,params,cancer,survival_df)
```

iii. By the execution #Survival plot for a single gene cell, a user can save and visualize the survival plot for an individual gene, indicated by *gene* variable.

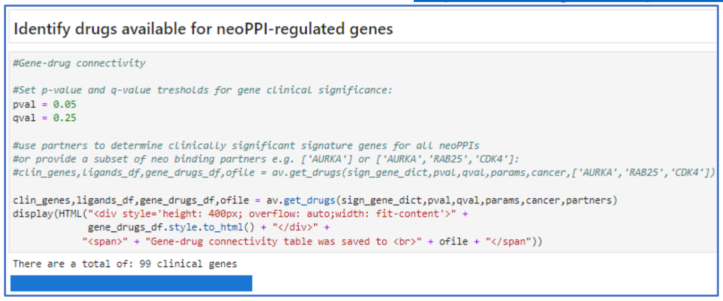


```
#Survival plot for a single gene
gene = 'PLK1'
#Save it
av.single_gene_survival_plot(df_mut_exp_samples,gene,clinical_f)[0].savefig(fig_folder+'survival_'+gene+'.pdf',dpi=600,format='pdf')
av.single_gene_survival_plot(df_mut_exp_samples,gene,clinical_f)[0]
```



4.6.6. Identify drugs available for neoPPI-regulated genes

i. The approved drugs and general inhibitors available for neoPPI-regulated genes can be identified by executing #Gene-drug connectivity cell. The pval and qval variables can be used to set statistical thresholds and conduct the analysis for a subset of clinically significant genes. To perform the analysis for all neoPPI signature genes, set pval=1 and qval=1. The information about the available drugs is extracted from IUPHAR database: https://www.guidetopharmacology.org

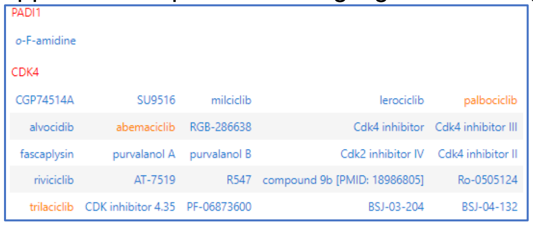


Identify drugs available for neoPPI-regulated genes

```
#Gene-drug connectivity

#Set p-value and q-value tresholds for gene clinical significance:
pval = 0.05
qval = 0.25

#use partners to determine clinically significant signature genes for all neoPPIs
#or provide a subset of neo binding partners e.g. ['AURKA'] or ['AURKA','RAB25','CDK4']:
#clin_genes,ligands_df,gene_drugs_df,ofile = av.get_drugs(sign_gene_dict,pval,qval,params,cancer,['AURKA','RAB25','CDK4'])

clin_genes,ligands_df,gene_drugs_df,ofile = av.get_drugs(sign_gene_dict,pval,qval,params,cancer,partners)
display(HTML("<div style='height: 400px; overflow: auto;width: fit-content'>" +
             gene_drugs_df.style.to_html() + "</div>" +
            "<span>" + "Gene-drug connectivity table was saved to <br>" + ofile + "</span>"))
```

```
There are a total of: 99 clinical genes
```

ii. The identified compounds will shown in an interactive table, where each compound is directly linked to its page on the https://www.guidetopharmacology.org website for detailed exploration. The FDA-approved compounds are highlighted in orange:



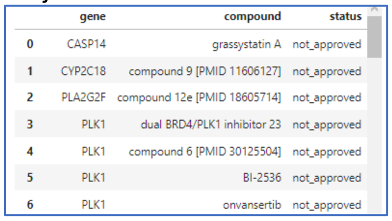
PADI1

o-F-amidine				

CDK4

CGP74514A	SU9516	milciclib	lerociclib	palbociclib
alvocidib	abemaciclib	RGB-286638	Cdk4 inhibitor	Cdk4 inhibitor III
fascaplysin	purvalanol A	purvalanol B	Cdk2 inhibitor IV	Cdk4 inhibitor II
riviciclib	AT-7519	R547	compound 9b [PMID: 18986805]	Ro-0505124
trilaciclib	CDK inhibitor 4.35	PF-06873600	BSJ-03-204	BSJ-04-132

iii. For convenience, the data is also shown in a scrollable table, which is automatically saved to the Project/Tables folder:



	gene	compound	status
0	CASP14	grassystatin A	not_approved
1	CYP2C18	compound 9 [PMID 11606127]	not_approved
2	PLA2G2F	compound 12e [PMID 18605714]	not_approved
3	PLK1	dual BRD4/PLK1 inhibitor 23	not_approved
4	PLK1	compound 6 [PMID 30125504]	not_approved
5	PLK1	BI-2536	not_approved
6	PLK1	onvansertib	not_approved

Together, these examples demonstrate how to use the Averon Notebook to estimate and compare the levels of mutant-enabled neoPPIs across different cancer types, prioritize the most clinically significant neoPPIs whose high levels may contribute to decreased patient survival, determine the sets of neoPPI-regulated genes and the associated oncogenic pathways, identify the most clinically relevant signature genes, and further determine neoPPI-regulated genes with available inhibitors and approved drugs.