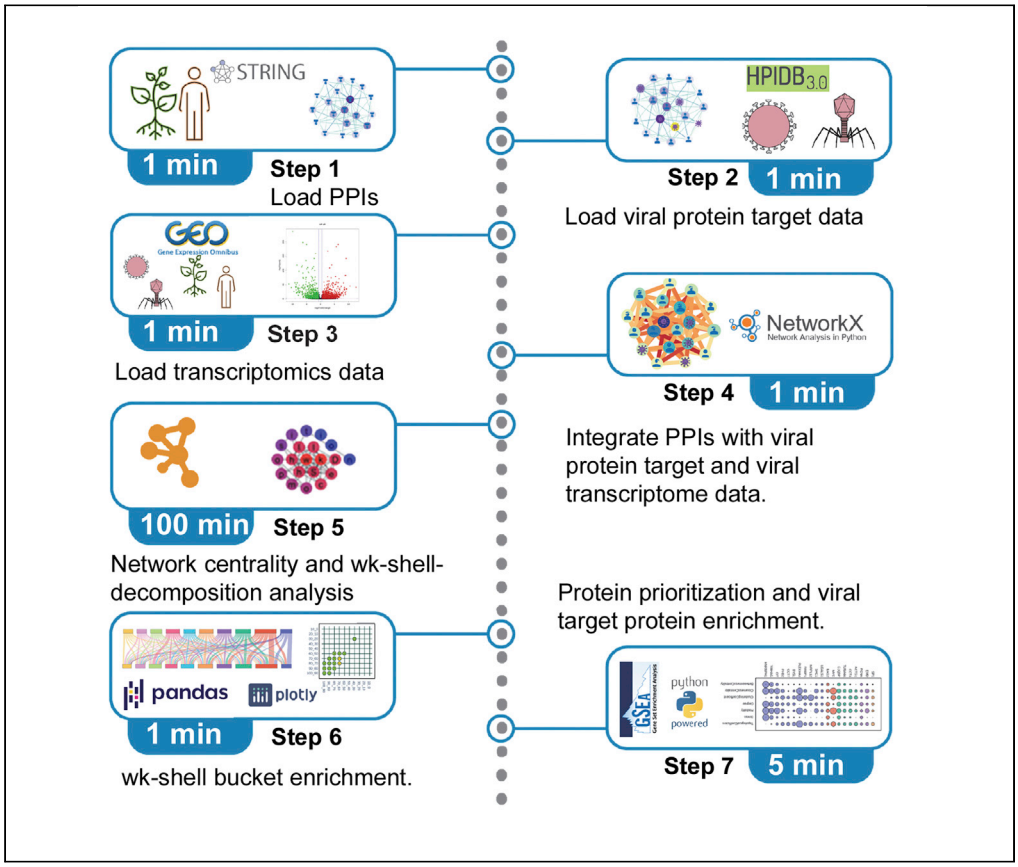
STAR Protocols

CellPress
OPEN ACCESS

Protocol

A pipeline of integrating transcriptome and interactome to elucidate central nodes in host-pathogens interactions



Investigating the complexity of host-pathogen interactions is challenging. Here, we outline a pipeline to identify important proteins and signaling molecules in human-viral interactomes. Firstly, we curate a comprehensive human interactome. Subsequently, we infer viral targets and transcriptome-specific human interactomes (VTTSHI) for papillomavirus and herpes viruses by integrating viral targets and transcriptome data. Finally, we reveal the common and shared nodes and pathways in viral pathogenesis following network topology and pathway enrichment analyses.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Nilesh Kumar, Bharat Mishra, M. Shahid Mukhtar

smukhtar@uab.edu

Highlights

Protocol for integrative analysis of transcriptome and proteome network data

Network subgroup enrichment of two host-pathogen interaction networks

Preprocessing of data and heterogeneous network integration

Kumar et al., STAR Protocols 3, 101608
September 16, 2022
https://doi.org/10.1016/j.xpro.2022.101608

STAR Protocols

CellPress
OPEN ACCESS

Protocol

A pipeline of integrating transcriptome and interactome to elucidate central nodes in host-pathogens interactions

Nilesh Kumar,[1,2] Bharat Mishra,[1] and M. Shahid Mukhtar[1,3,*]

[1]Department of Biology, University of Alabama at Birmingham, Birmingham, AL 35294, USA

[2]Technical contact: nileshkr@uab.edu

[3]Lead contact

*Correspondence: nileshkr@uab.edu (N.K.), smukhtar@uab.edu (M.S.M.)
https://doi.org/10.1016/j.xpro.2022.101608

SUMMARY

Investigating the complexity of host-pathogen interactions is challenging. Here, we outline a pipeline to identify important proteins and signaling molecules in human-viral interactomes. Firstly, we curate a comprehensive human interactome. Subsequently, we infer viral targets and transcriptome-specific human interactomes (VTTSHI) for papillomavirus and herpes viruses by integrating viral targets and transcriptome data. Finally, we reveal the common and shared nodes and pathways in viral pathogenesis following network topology and pathway enrichment analyses.

For complete details on the use and execution of this protocol, please refer to Kumar et al. (2020).

BEFORE YOU BEGIN

Host-pathogen interactions result from an intricate interplay between diverse pathogens such as bacteria, fungi, virus, and their respective hosts including humans, mice, and plants (Garbutt et al., 2014; González-Fuente et al., 2020; Kumar et al., 2020; Liu et al., 2022; Lopez and Mukhtar, 2017; McCormack et al., 2016; Mishra et al., 2019, 2021a, 2021b; Mukhtar et al., 2016; Spears et al., 2019; Vidal et al., 2011). Such complex interactions between pathogens and their hosts are mediated, at least in part, by secreted microbial proteins (also known as effectors) and other molecules that are capable of manipulating host cells (Barabasi et al., 2011; Casadevall and Pirofski, 2000; Feaugas and Sauvonnet, 2021; Long et al., 2022; Mishra et al., 2017; Mukhtar et al., 2016; Naqvi et al., 2019; Schneweis et al., 1984; Washington et al., 2016; Yang et al., 2011; Zaidi et al., 2020; Zhang et al., 2016; Zhu and Viejo-Borbolla, 2021). Over the last two decades, research in both plant and animal fields has reported that these diverse pathogens target host proteins that are well-positioned in the biological system for efficient pathogenesis as well as host survival (Ahmed et al., 2018; Arabidopsis Interactome Mapping, 2011; Barabasi et al., 2011; Gordon et al., 2020; Klopffleisch et al., 2011; Kumar et al., 2020; Mishra et al., 2018; Mott et al., 2019; Mukhtar et al., 2011; Smakowska-Luzan et al., 2018; Vidal et al., 2011; Watkins et al., 2021; Wessling et al., 2014; Zhou et al., 2020). Therefore, network biology is at the forefront of discovering novel players, modules, and pathways associated with host-pathogen interaction (Bosl et al., 2019; Casadevall and Pirofski, 2000; Garbutt et al., 2014; Mishra et al., 2019, 2021b, 2022; Vidal et al., 2011). Previously, we have constructed COVID19 transcriptome-driven human-viral interactome to identify significant proteins involved in the pathogenesis of SARS-CoV-2 (Kumar et al., 2020).

This protocol facilitates the integration of the human protein-protein interaction (PPI) network, the viral protein targets, and the viral pathogenesis transcriptome data (Kumar et al., 2020).

Apart from conventional network analysis, the protocol implements a method to uncover key players in pathogenesis (Mishra et al., 2019, 2021b). Python scripts and Jupyter notebooks are used to build the protocol. A Jupyter notebook is a scalable, cross-platform web-based interactive computing platform that can produce executable Python scripts for HPC. Further, significantly expressed genes are also integrated after the DESeq2 (Love et al., 2014) analysis of the RNA-Seq data (GSE124118 and GSE74927 (Qin et al., 2020; Qin et al., 2018; Zhang et al., 2016)) of respective viral pathogens. This method is scalable and can also be applied to other hosts including plants (Arabidopsis, rice, cotton, etc.), animals (mouse, rat, etc.), and their respective pathogenic interactions.

All the analysis steps described in this protocol are performed using Python, a high-level, interpreted, general-purpose programming language, which can be run on most operating systems including UNIX, Windows, and macOS. The current protocol was developed using Python version 3.10.4 running on a Linus system (Red Hat Enterprise Linux Server release 7.9 (Maipo)). Additionally, to Python, a list of Linux packages and tools is required since they provide the necessary or optional functions for the data processing, analysis, or visualization steps covered in this protocol. Linux tools and packages used in this protocol are open source and come pre-built or are available to be installed. Even though the protocol is compatible with most UNIX and Linux distributions, Ubuntu 22 and Fedora 36, Red Hat 7, as well as macOS Monterey, are recommended. As this protocol uses few Linux tools, Windows users will need to install Windows Subsystem for Linux 2 (WSL) in order to run it.

Besides HPC devices, this protocol has been tested on other devices, with the following specifications:

OS: Windows 11 (5.10.102.1-microsoft-standard-WSL2), and Fedora 36.

RAM: 16 GB.

SDD: 256 GB.

CPU: Intel i7.

conda 4.12.0.

⚠ CRITICAL: Italicized texts in the box are the output of the preceding Jupyter notebook cell and are not executable as Python or bash scripts.

Clone the protocol directory

Timing: 5 min

The Python notebook and most of the required data sets for running this protocol can be found on the GitHub repository viral targets and transcriptome-specific human interactomes (VTTSHI).

https://github.com/nilesh-iiita/VTTSHI

1. Clone the protocol directory from the GitHub repository by using the command below from the Linux command-line interface:

```
$git clone https://github.com/nilesh-iiita/VTTSHI
```

Alternatives: Go to https://github.com/nilesh-iiita/VTTSHI and select 'Download Zip' under the 'Code' button to download the pipeline.

Create a conda environment, then install all the packages at once with the following command.

```
conda env create -file VTTSHI.yml
```

Download datasets and environment setup

Timing: 5 min

This protocol requires a wide range of graph/network data, transcriptomics data, and a few annotation datasets. The first step is to build a PPI network. PPIs for humans have been collected from five databases, STRING (Szklarczyk et al., 2019), HI-union (Luck et al., 2020), BioPlex 3.0 (Huttlin et al., 2015), CoFrac (Wan et al., 2015) and QUBIC (Hein et al., 2015) (key resources table). This protocol is designed with the objective of avoiding unvalidated interactions using only experimental STRING interactions. In addition to PPIs, we need annotation files as well. The human proteins that viral proteins target are also required. HPIDP 3.0 (Ammari et al., 2016; Kumar and Nanduri, 2010) is used to determine viral protein targets. It contains several datasets of host protein targets. In particular, the herpes simplex virus (HSV) and the human papillomavirus (HPV) were chosen, but this protocol is applicable to other organisms as well. Since PPIs come from different sources, they are likely to have different formats for protein identifiers (such as UniProt-KB, gene names, protein names, string IDs, etc.). We must convert all these varying protein identifiers into a single common identifier. To integrate the networks, we will use the UniProt-KB protein identifier. Since most enrichment analysis tools do not support UniProt-KB IDs, UniProt-KB IDs need to be converted to gene names subsequently as required. All steps pertaining to the downloading, interconversion, and preprocessing of protein gene IDs can be found in one Jupyter notebook, "0_Data_prep.ipynb".

2. Execute all cells of "0_Data_prep.ipynb" in the Jupyter notebook.

⚠ CRITICAL: If the protocol is going to be applied to different datasets, then it is imperative that the raw datasets are carefully processed before applying them to the analysis.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
STRING	(Szklarczyk et al., 2019)	https://stringdb-static.org/download/protein.links.full.v11.5/9606.protein.links.full.v11.5.txt.gz
HI-union	(Luck et al., 2020)	http://www.interactome-atlas.org/data/HI-union.tsv
BioPlex 3.0	(Huttlin et al., 2015)	http://wren.hms.harvard.edu/bioplex/data/BioPlex_293T_Network_10K_Dec_2019.tsv
CoFrac	(Wan et al., 2015)	https://static-content.springer.com/esm/art%3A10.1038%2Fnature14877/MediaObjects/41586_2015_BFnature14877_MOESM13_ESM.zip
QUBIC	(Hein et al., 2015)	https://ars.els-cdn.com/content/image/1-s2.0-S0092867415012702-mmc3.xlsx
HPIDB 3.0	(Ammari et al., 2016; Kumar and Nanduri, 2010)	https://hpidb.igbb.msstate.edu/
Herpes simplex virus type I (HSV-1) RNA-Seq	N/A	Gene Expression Omnibus (GEO): GSE124118
Human papillomavirus (HPV) RNA-Seq	(Qin et al., 2018, 2020; Zhang et al., 2016)	Gene Expression Omnibus (GEO): GSE74927

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
glob2 (0.7)	N/A	pip install glob2
GSEApy (0.10.8)	(Mootha et al., 2003; Subramanian et al., 2005)	pip install gseapy
Matplotlib (3.5.1)	(Hunter, 2007)	pip install matplotlib
NetworkX (2.6.3)	(Hagberg et al., 2008)	pip install networkx
NumPy (1.22.1)	(Harris et al., 2020)	pip install numpy
pandas (1.3.5)	(McKinney, 2010)	pip install pandas
Plotly (5.5.0)	(Inc., 2022)	pip install plotly
kaleido (0.2.1)	N/A	pip install kaleido
UpSetplot (0.6.0)	(Lex et al., 2014)	pip install UpSetPlot
poppler (0.3.0)		pip install python-poppler
Cytoscape (3.9.1)	(Shannon et al., 2003)	https://cytoscape.org/download.html
Wk-shell-decomposition	N/A	https://apps.cytoscape.org/apps/wkshelldecomposition

MATERIALS AND EQUIPMENT

The protocol describes an integrative computational approach that is primarily implemented using Python 3.10 and preferably Jupyter Notebook (an interactive computing platform on the web). Additional Python library is mentioned in the key resources table*, glob2, GSEApy* (Mootha et al., 2003; Subramanian et al., 2005), Matplotlib (Hunter, 2007), NetworkX (Hagberg et al., 2008), NumPy (Harris et al., 2020), pandas (McKinney, 2010), Plotly (Inc., 2022), kaleido, UpSetplot (Lex et al., 2014), etc. *For network analysis, visualization, and interpretation, Cytoscape* (Shannon et al., 2003) *is required. For transcriptome analysis, R is also necessary but already analyzed transcriptome data is available in the GitHub folder "VTTSHI". Check out the "*key resources table*" and click the provided link to learn how to install software/packages. The Conda package and environment management system is recommended for ease of managing packages, dependencies, and environments.*

STEP-BY-STEP METHOD DETAILS

The protocol includes seven parts, from PPIs to network comparisons to getting pathogenic target proteins, and everything in between. Each of these sections is described in more detail below. Create a Python notebook (e.g.,: "VTTSHI_STAR_Protocol.ipynb") to execute all Python scripts and Shell commands step-by-step. In the Linux terminal using the following command open a new Jupyter notebook.

```
$ jupyter notebook
```

Load protein-protein interaction networks (PPIs)

Timing: 1 min

1. Make sure that the Python libraries are loaded before you proceed.
2. Load and prepare PPIs with the NetworkX and pandas Python package.
3. Run the following command in the Jupyter notebook.

```
import pandas as pd

from glob2 import glob

from collections import defaultdict

import networkx as nx
```

```
from pathlib import Path
from upsetplot import UpSet, from_contents
from copy import copy
import urllib.parse
import urllib.request
import gseapy as gp
from matplotlib import pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
```

a. Make sure all network files are in the path (same environment path).

⚠ CRITICAL: Every network is stored as a tab-separated edge list (*_net.tsv).

Note: Use the following command to check if all "*_net.tsv" are in the same path (Optional step). The tree is a linux tool and can be installed using linux package maneger such as "dnf" and "apt" on the Fedora/centos and Ubuntu respectively.

```
!tree D1_Network_data/ -P *_net.tsv
```

The output should look something like this:

```
D1_Network_data/
├── BioPlex_3
│   └── BioPlex_net.tsv
├── CoFrac

│   ├── CoFrac_net.tsv
│   ├── __MACOSX
│   │   └── nature14871-s2
│   └── nature14871-s2
├── HuRI_db
│   └── HuRI_Union_net.tsv
├── QUBIC
│   └── QUBIC_net.tsv
└── STRING_db
    └── STRING_exp_net.tsv

8 directories, 5 files
```

b. In the Jupyter notebook, run the following command to read network data as a pandas data-frame.

```
Network_files = glob('D1_Network_data//*/*_net.tsv')
Network_dfs = defaultdict(dict)
Network_Graphs = defaultdict(dict)
Network_Nodes = defaultdict(dict)
for Network_file in Network_files:
  Net_name = Network_file.split('/')[-1].replace('_net.tsv', '')
  df = pd.read_csv(Network_file, sep=''\t'')
  Network_dfs[Net_name] = df
  G = nx.from_pandas_edgelist(df, 'IDa', 'IDb')
  G.remove_edges_from(nx.selfloop_edges(G))
  Network_Graphs[Net_name] = G
  nodes = set(df.IDa.unique()).union(set(df.IDb.unique()))
  Network_Nodes[Net_name] = nodes
print(f''List of Networks as dataframe {list(Network_dfs)}'')
print(f''List of Networks as Graph object{list(Network_Graphs)}'')
print(f''List of Networks Nodes list {list(Network_Nodes)}'')
```

4. Use the following Python script to merge networks.

Note: Our protein networks include 'BioPlex', 'CoFrac', 'HuRI _Union', 'QUBIC', and 'STRING experimental' at this stage.

```
List_of_dfs = list(Network_dfs.values())
df_network = pd.concat(List_of_dfs)
df_network.drop_duplicates(inplace=True)
```

Load viral target data

Timing: 1 min

Optional: Ensure that all viral target datasets are in the path (same environment path) by using the following command (optional step).

```
!tree HPIDB_data/**/*.tsv
```

5. Run the following command to load the viral data.

```
Viral_target = defaultdict(dict)
Viral_files = glob('HPIDB_data/*/*.mitab_plus.tsv')
for Viral_file in Viral_files:
  V_name = Viral_file.split('/')[-1].split('_')[0].title()
```

```
  df = pd.read_csv(Viral_file, sep=''\t'')

  df_arrt = df.copy()

  df_arrt[list(df_arrt)[-1]] = V_name + ''_target''

  df_arrt.to_csv(Viral_file.replace('mitab_plus.tsv', ''mitab_plus_attr.txt''), index=-
False, sep='' '')

  Human_proteins = set(df.Human.unique())

  Viral_target[V_name] = Human_proteins

print(list(Viral_target))
```

6. The following Python function to check for overlap between viral targets and individual PPI networks.

```
def Upset_protein(Target):

  Uset_data = defaultdict(dict)

  print('>', list(Network_Nodes))

  Uset_data = copy(Network_Nodes)

  Uset_data[Target] = Viral_target[Target]

  print(list(Uset_data))

  return Uset_data
```

a. To make a plot of HVP overlaps, follow the script below (Figure 2B).

```
Data = Upset_protein('Papillomaviruses')

Data = from_contents(Data)

Data

upset = UpSet(Data, show_counts='%d', show_percentages=True, shading_color=.0, other_
dots_color=.1, orientation='vertical')

upset.style_subsets(present=''Papillomaviruses'', edgecolor=''blue'', linewidth=1)

Path(''Images'').mkdir(parents=True, exist_ok=True)

upset.plot()

fig = plt.gcf()

fig.savefig(''Images/Upset_HPV.png'', dpi=300)

fig.savefig(''Images/Upset_HPV.pdf'')

del(Data)
```

b. To make a plot of HSP overlaps, follow the script below (Figure 2C).

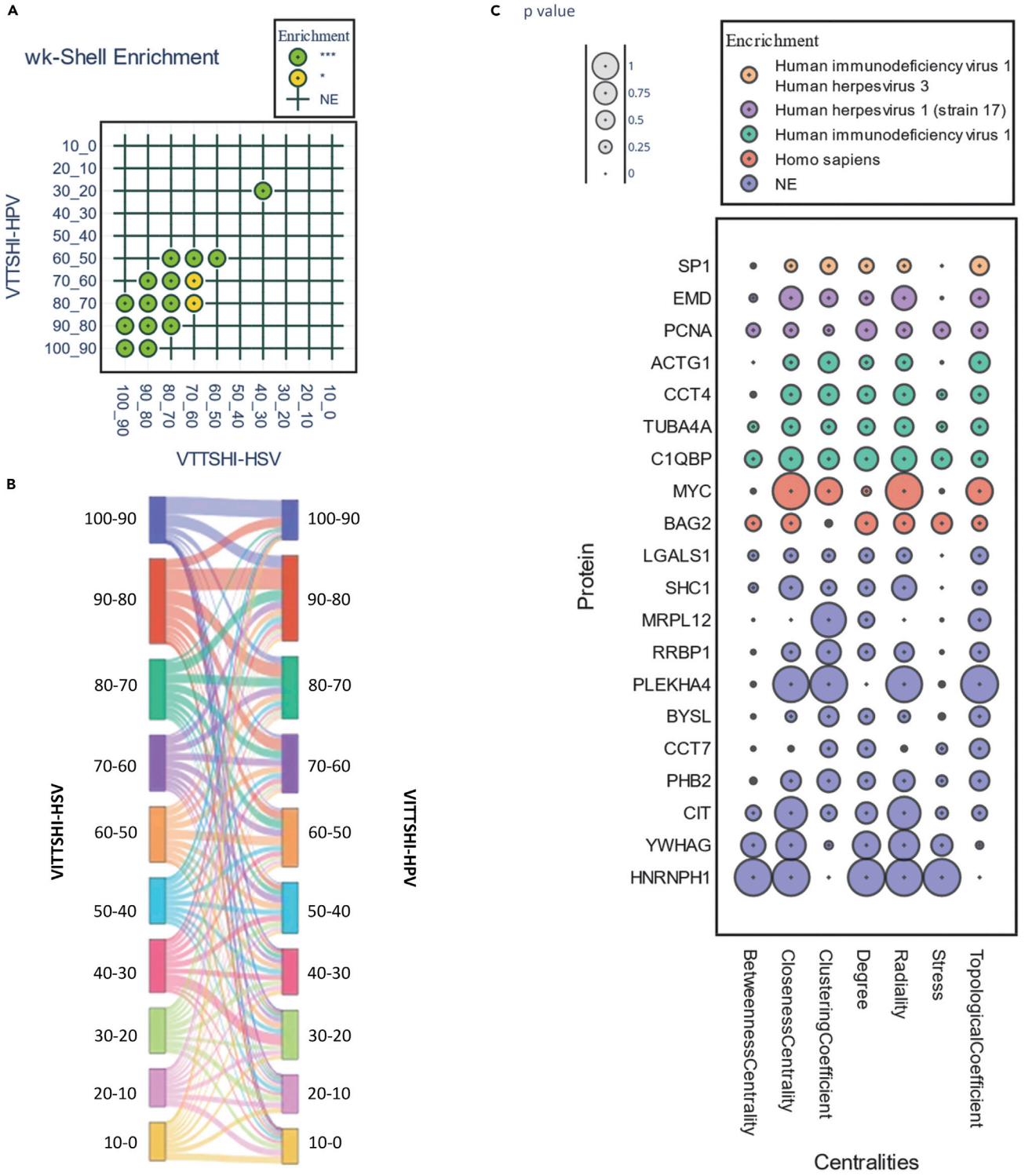


Figure 1. Enrichment analysis and comparison of VITTSHI-HPV and VITTSHI-HSV

(A) Shell-to-shell enrichment analysis of VITTSHI-HSP and VITTSHI-HSP wk-shell decomposition buckets.

(B) The Sankey plot of overlapping wk-shell decomposition buckets.

(C) Prioritization of proteins based on VITTSHI-HPV centrality and their functional enrichment.

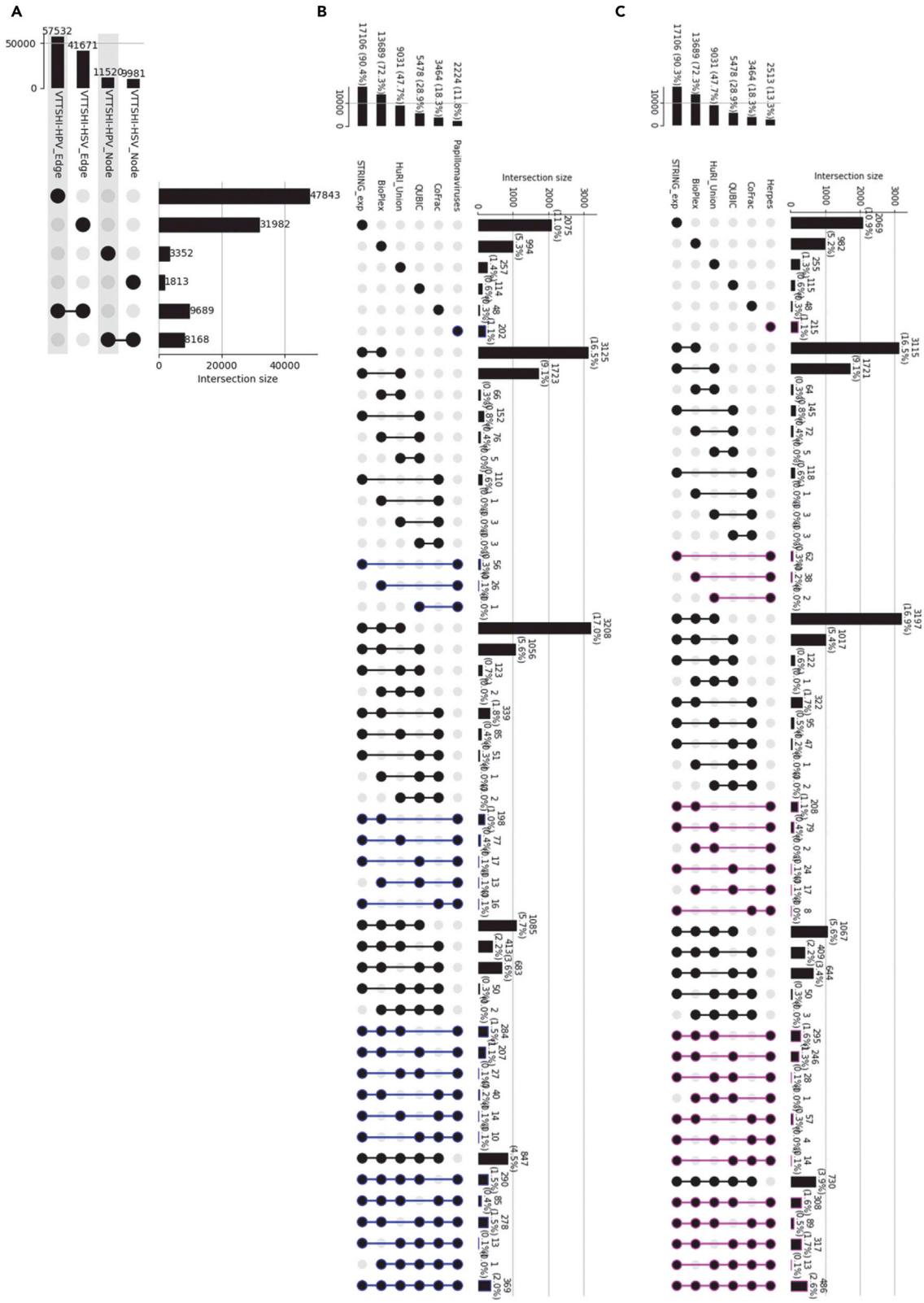

A
B
C
Intersection size
VTTSHI-HPV_Edge
VTTSHI-HSV_Edge
VTTSHI-HPV_Node
VTTSHI-HSV_Node
STRING_exp
BioPlex
HuRI_Union
QUBIC
CoFrac
Papillomaviruses
Herpes

Figure 2. Overlapping UpSet plots of nodes and edges
(A) The intersection between the VITTSHI-HPV network and the VITTSHI-HSV network.
(B) Couplings between HPV target proteins and PPIs.
(C) A variety of overlapping combinations of HSV target proteins and different PPIs.

```
Data = Upset_protein('Herpes')

Data = from_contents(Data)

Data

upset = UpSet(Data, show_counts='%d', show_percentages=True, shading_color=.0, other_
dots_color=.1, orientation='vertical')

upset.style_subsets(present=''Herpes'', edgecolor=''magenta'', linewidth=1)

Path(''Images'').mkdir(parents=True, exist_ok=True)

upset.plot()

fig = plt.gcf()

fig.savefig(''Images/Upset_HSV.png'', dpi=300)

fig.savefig(''Images/Upset_HSV.pdf'')

del(Data)
```

⚠ CRITICAL: Only about 1 percent of viral protein targets do not overlap with PPI datasets in both cases. It is imperative to minimize non-overlapping protein targets.

Load transcriptomics data

Timing: 1 min

The RNA-Seq analysis has been done using DESeq2 using GSE124118 and GSE74927 (Qin et al., 2018, 2020; Zhang et al., 2016) datasets.

7. Ensure that all transcriptomics files are in the path (same environment path).

⚠ CRITICAL: All networks should be in a comma-separated edge list. (*_net.csv) format. Check that all "*.csv" files are in the path (same environment path) using the following command.

```
!tree Transcriptome/**/*00_DEDeq2_Data**/*.csv
```

8. Load transcriptomics data using the following script.

```
Herpes_df = pd.read_csv(''Transcriptome/Herpes_GSE124118/00_DEDeq2_Data_HSV/hsv_cells_
Skin_vs_lung.csv'')

Herpes_exp = set(Herpes_df[Herpes_df.padj <= 0.05].UNIPROT.dropna())

Papillomavirus_df = pd.read_csv(''Transcriptome/Papillomavirus_GSE74927/00_DEDeq2_Da-
ta_HPV/hpv_status_HPV_vs_Neg.csv'')

Papillomavirus_exp = set(Papillomavirus_df[Papillomavirus_df.padj <= 0.05].UNIPROT.
dropna())

print(f''Number of expressed genes in HSV:{len(Herpes_exp)}'')

print(f''Number of expressed genes in HPV:{len(Papillomavirus_exp)}'')
```

⚠ CRITICAL: To eliminate genes that are not significantly expressed, only a q-value (padj *= 0.05) filter is used. However, other filters can be applied, depending on the circumstances. P-value can be used in place of q-value, for example:

```
Herpes_exp = set(Herpes_df[Herpes_df.pvalue <= 0.05].UNIPROT.dropna())

Papillomavirus_exp = set(Papillomavirus_df[Papillomavirus_df.pvalue <= 0.05].UNIPROT.
dropna())
```

Integrate PPIs with viral protein target and viral transcriptome data

Timing: 1 min

9. Using the following Python function Integrate PPIs with Viral protein target and Viral transcriptome data.

Note: This Python function that takes a protein list and a network as inputs, and outputs two networks, an open network, and a closed network. From the network, it extracts the partners of the proteins or nodes list provided. When the extracted network contains only proteins from the provided list, it is a closed network; otherwise, if there are additional proteins not listed in the list, it is an open network. In technical terms, a network consists of edges. Each edge in an open target network contains at least one target node (protein target). In contrast, in a close target network, both nodes of an edge list should be protein' (Viral) targets.

```
Network_Dir = ''Network_data/''

def df_nx(df, Targets):

  LOL = df.values.tolist() #[:10]

  Open = []

  Close = []

  for a,b in LOL:

    # print(a,b)

    if a in Targets and b in Targets:

      Close.append([a,b])

    if a in Targets or b in Targets:

      Open.append([a,b])

  O = nx.Graph()

  O.add_edges_from(Open)

  O.remove_edges_from(nx.selfloop_edges(O))

  C = nx.Graph()

  C.add_edges_from(Close)

  C.remove_edges_from(nx.selfloop_edges(C))

  print(O)

  print(C)
```

```
  return O, C

List_of_dfs = list(Network_dfs.values())

df_network = pd.concat(List_of_dfs)

df_network.drop_duplicates(inplace=True)

df_network.head()
```

10. Integrate PPIs with viral (HSV) targets and transcriptome data using the following Python script and make an open target and close target network.

Note: We will refer to the integrated network as VTTSHI-HSV.

```
Count_info = defaultdict(lambda: defaultdict(lambda: defaultdict(dict)))

O, C = df_nx(df_network, Viral_target['Herpes'])

Out_Dir = Network_Dir + ''Herpes/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(O, Out_Dir + 'Herpes' + ''_Open_edgelist.nx'', delimiter=' ', data=False)

nx.write_edgelist(C, Out_Dir + 'Herpes' + ''_Close_edgelist.nx'', delimiter=' ', data=False)

Count_info['Herpes']['Viral_target']['Open']['Node'] = O.number_of_nodes()

Count_info['Herpes']['Viral_target']['Open']['Edges'] = O.number_of_edges()

Count_info['Herpes']['Viral_target']['Close']['Node'] = C.number_of_nodes()

Count_info['Herpes']['Viral_target']['Close']['Edges'] = C.number_of_edges()

O_exp_O, O_exp_C = df_nx(nx.to_pandas_edgelist(O), Herpes_exp)

Out_Dir = Network_Dir + ''Herpes/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(O_exp_O, Out_Dir + 'Herpes' + ''_Open_exp_Open_edgelist.nx'', delimiter=' ',
data=False)

nx.write_edgelist(O_exp_C, Out_Dir + 'Herpes' + ''_Open_exp_Close_edgelist.nx'', delimiter=' ',
data=False)

Count_info['Herpes']['Viral_expressed_open']['Open']['Node'] = O_exp_O.number_of_nodes()

Count_info['Herpes']['Viral_expressed_open']['Open']['Edges']  =  O_exp_O.number_of_
edges()

Count_info['Herpes']['Viral_expressed_open']['Close']['Node']  =  O_exp_C.number_of_
nodes()

Count_info['Herpes']['Viral_expressed_open']['Close']['Edges']  =  O_exp_C.number_of_
edges()

C_exp_O, C_exp_C = df_nx(nx.to_pandas_edgelist(C), Herpes_exp)

Out_Dir = Network_Dir + ''Herpes/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(C_exp_O, Out_Dir + 'Herpes' + ''_Close_exp_Open_edgelist.nx'', delimiter=' ',
data=False)
```

```
nx.write_edgelist(C_exp_C, Out_Dir + 'Herpes' + ''_Close_exp_Close_edgelist.nx'', delimiter=' ', data=False)

Count_info['Herpes']['Viral_expressed_close']['Open']['Node'] = C_exp_O.number_of_nodes()

Count_info['Herpes']['Viral_expressed_close']['Open']['Edges'] = C_exp_O.number_of_edges()

Count_info['Herpes']['Viral_expressed_close']['Close']['Node'] = C_exp_C.number_of_nodes()

Count_info['Herpes']['Viral_expressed_close']['Close']['Edges'] = C_exp_C.number_of_edges()
```

11. Integrate PPIs with viral (HPV) targets with transcriptome data using the following Python script and make an open target and close target network.

Note: We will refer to the integrated network as VTTSHI-HPV.

```
O, C = df_nx(df_network, Viral_target['Papillomaviruses'])

Out_Dir = Network_Dir + ''Papillomaviruses/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(O, Out_Dir + 'Papillomaviruses' + ''_Open_edgelist.nx'', delimiter=' ', data=False)

nx.write_edgelist(C, Out_Dir + 'Papillomaviruses' + ''_Close_edgelist.nx'', delimiter=' ', data=False)

Count_info['Papillomaviruses']['Viral_target']['Open']['Node'] = O.number_of_nodes()

Count_info['Papillomaviruses']['Viral_target']['Open']['Edges'] = O.number_of_edges()

Count_info['Papillomaviruses']['Viral_target']['Close']['Node'] = C.number_of_nodes()

Count_info['Papillomaviruses']['Viral_target']['Close']['Edges'] = C.number_of_edges()

O_exp_O, O_exp_C = df_nx(nx.to_pandas_edgelist(O), Papillomavirus_exp)

Out_Dir = Network_Dir + ''Papillomaviruses/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(O_exp_O, Out_Dir + 'Papillomaviruses' + ''_Open_exp_Open_edgelist.nx'', delimiter=' ', data=False)

nx.write_edgelist(O_exp_C, Out_Dir + 'Papillomaviruses' + ''_Open_exp_Close_edgelist.nx'', delimiter=' ', data=False)

Count_info['Papillomaviruses']['Viral_expressed_open']['Open']['Node'] = O_exp_O.number_of_nodes()

Count_info['Papillomaviruses']['Viral_expressed_open']['Open']['Edges'] = O_exp_O.number_of_edges()

Count_info['Papillomaviruses']['Viral_expressed_open']['Close']['Node'] = O_exp_C.number_of_nodes()

Count_info['Papillomaviruses']['Viral_expressed_open']['Close']['Edges'] = O_exp_C.number_of_edges()

C_exp_O, C_exp_C = df_nx(nx.to_pandas_edgelist(C), Papillomavirus_exp)
```

```
Out_Dir = Network_Dir + ''Papillomaviruses/''

Path(Out_Dir).mkdir(parents=True, exist_ok=True)

nx.write_edgelist(C_exp_O, Out_Dir + 'Papillomaviruses' + ''_Close_exp_Open_edgelist.nx'', delimiter=' ', data=False)

nx.write_edgelist(C_exp_C, Out_Dir + 'Papillomaviruses' + ''_Close_exp_Close_edgelist.nx'', delimiter=' ', data=False)

Count_info['Papillomaviruses']['Viral_expressed_close']['Open']['Node'] = C_exp_O.number_of_nodes()

Count_info['Papillomaviruses']['Viral_expressed_close']['Open']['Edges'] = C_exp_O.number_of_edges()

Count_info['Papillomaviruses']['Viral_expressed_close']['Close']['Node'] = C_exp_C.number_of_nodes()

Count_info['Papillomaviruses']['Viral_expressed_close']['Close']['Edges'] = C_exp_C.number_of_edges()
```

12. Examine the overlap statistics for VTTSHI-HSV and VTTSHI-HPV.

```
pd.concat({k: pd.DataFrame(v).T for k, v in Count_info.items()}, axis=0)
```

⚠ CRITICAL: The overlap statistics for all integration steps should be examined for each viral dataset to be able to choose an appropriate overlapping network. In this protocol, the "open-expressed-open" dataset is used to analyze both viral data. This step is critical for different datasets, hosts, and pathogens. Make sure the chosen network is not too sparse for further processing.

The output of the overlap statistics:

		Open	*Close*
Herpes	*Viral_target*	*{'Node': 16557, 'Edges': 247923}*	*{'Node': 2263, 'Edges': 50368}*
	Viral_expressed_open	*{'Node': 9981, 'Edges': 41671}*	*{'Node': 861, 'Edges': 2023}*
	Viral_expressed_close	*{'Node': 1845, 'Edges': 8624}*	*{'Node': 184, 'Edges': 443}*
Papilloma viruses	*Viral_target*	*{'Node': 16227, 'Edges': 207850}*	*{'Node': 1977, 'Edges': 32875}*
	Viral_expressed_open	*{'Node': 11521, 'Edges': 57537}*	*{'Node': 1690, 'Edges': 4824}*
	Viral_expressed_close	*{'Node': 1718, 'Edges': 8660}*	*{'Node': 270, 'Edges': 697}*

Network centrality and weighted *k*-shell decomposition analysis

Timing: 100 min

By analyzing network centrality, key characteristics of the network structure can be uncovered, and complex patterns of relationships can be estimated. The importance of protein nodes can also be determined using indicators of centrality like 'Betweenness Centrality', 'Closeness Centrality', 'Clustering Coefficient', 'Degree', etc. The iterative refinement centrality of a given node, such as that of wk-shell decomposition, also informs us of the mutual enhancement effect. Shells are assigned to nodes in the wk-shell decomposition. Networks have varying numbers of shells. Shells are normalized to percentiles, and further percentiles are grouped into buckets of ten so that buckets of each network can be fairly compared.

⚠ CRITICAL: Please install the Cytoscape software as indicated in the "key resources table". Using the Cytoscape app manager, install the wk-shell-decomposition app as well.

13. Use the following command to ensure that network datasets are in the path (same environment path, optional step).

```
!tree Network_data/**/*Open_exp_Open_edgelist.nx
```

14. Perform network centrality and weighted *k*-shell (wk-shell) decomposition analysis, following the steps below using Cytoscape.
    a. Open Cytoscape.
    b. Load network - Click => File -> Import -> From File -> (Choose network file, separated by spaces and without headers (*.nx)).
    c. Click => Tools -> Analyze Networks -> (Uncheck Directed Graph option) > OK.
    d. Select Apps -> wk-shell-decomposition.
    e. In the parent directory, make a "Cytoscape_network_analysis" folder if it doesn't exist.

```
!mkdir Cytoscape_network_analysis
```

    f. Select File -> Export -> Table To File… -> Save to "Cytoscape_network_analysis" ('Herpes_Open_exp_Open_edgelist.nx default node.csv' and 'Papillomaviruses_Open_exp_Open_edgelist.nx default node.csv')
    g. Make sure all files are in the path (same environment path) using the following command (optional step).

```
!tree Cytoscape_network_analysis/*.csv
```

wk-shell enrichment

Timing: 1 min

Comparing VTTSHI-HSV and VTTSHI-HPV networks using the wk-shell-decomposition network analysis (from the previous step).

15. Quantitatively compare node and edge overlap.

Note: The following Python script loads networks as NetworkX Graph objects.

```
G_h = nx.read_edgelist('Network_data/Herpes/Herpes_Open_exp_Open_edgelist.nx')

G_h.name = 'VTTSHI-HSV'

print(G_h)

G_p = nx.read_edgelist('Network_data/Papillomaviruses/Papillomaviruses_Open_exp_Open_
edgelist.nx')

G_p.name = 'VTTSHI-HPV'

print(G_p)
```

16. Create an UpSet plot Python dictionary object of nodes and sets, using the following function.

```
def Upset_Graph_data(Graphs, sort_edges = True):
  Data = defaultdict(list)
  for i in range(len(Graphs)):
    G = Graphs[i]
    if len(G.name) == 0:
      G.name = ''G"+str(i+1)
    # print(G.name)
    Data[G.name + ''_Node''] = set(G.nodes())
    Edges = list(G.edges())
    for j in range(len(Edges)):
      e = list(Edges[j])
      if sort_edges:
        e.sort()
      e = ''_''.join(e)
      Edges[j] = e
    Data[G.name + ''_Edge''] = set(Edges)
  for i in Data:
    print(f''{i} : {len(Data[i])}'')
  return dict(Data)
Upset_data = Upset_Graph_data([G_h, G_p])
Data = from_contents(Upset_data)
Data
```

17. Create an UpSet plot Using the following Python script (Figure 2A).

```
upset = UpSet(Data, show_counts='%d', shading_color=.1, other_dots_color=.1, element_si-
ze=None, orientation='vertical')
upset.plot()
Path(''Images'').mkdir(parents=True, exist_ok=True)
fig = plt.gcf()
fig.set_size_inches(5, 6)
fig.savefig(''Images/Graph_comp.png'', dpi=300)
fig.savefig(''Images/Graph_comp.pdf")
```

18. Make a pandas dataframe object containing the Cytoscape centrality analysis results.

```
Herpes_Cyto_File = ''Cytoscape_network_analysis/Herpes_Open_exp_Open_edgelist.nx default
node.csv''
Papilloma_Cyto_File = ''Cytoscape_network_analysis/Papillomaviruses_Open_exp_Open_
edgelist.nx default node.csv''
```

```
Herpes_df = pd.read_csv(Herpes_Cyto_File)

Papilloma_df = pd.read_csv(Papilloma_Cyto_File)
```

19. Create a Python dictionary object of the wk-shell-decomposition bucket.

Note: Refer to the following Python definition.

```
def Bucket(File, Step=10, Print=False):
  df = pd.read_csv(File)
  df = df.set_index(''name'')
  df = df.dropna()
  df['Percentile'] = df._wkshell.rank(pct = True)
  df = df.sort_values('Percentile', ascending = False)
  df = df[[''Percentile'']]
  List = [i/100 for i in range(0,100,Step)]
  for i in List:
    df.loc[(df['Percentile'] >= i), 'Bucket'] = int(i*100)
  df = df[[''Bucket'']]
  d = df.T.to_dict('list')
  Dic = defaultdict(list)
  for i in d:
    Gene,Bucket = i, str(Step+int(d[i][0])) + ''_'' + str(int(d[i][0]))
    Dic[Bucket].append(Gene)
  for i in Dic:
    Dic[i] = set(Dic[i])
    if Print:
      print(f''Number of gene in {i} bucket : {len(Dic[i])}'')
return Dic
Herpes_wkshell_bukt = Bucket(Herpes_Cyto_File)
Papilloma_wkshell_bukt = Bucket(Papilloma_Cyto_File)
```

20. Use the following Python function, save 'wk-shell-decomposition bucket' as a gmt (Gene Matrix Transposed file format) file.

```
def Dict_to_gmt(Dic, discription="NA'', file_name = ''Wkshell_file''):
  from pathlib import Path
  Genes = []
  GMT_Dir = ''GMT_base/''
  Path(GMT_Dir).mkdir(parents=True, exist_ok=True)
```

```
  file_name = GMT_Dir + file_name + ''.gmt''
  fh = open(file_name, ''w'')
  for i in Dic:
    set_name = i
    Items = '\t'.join(set(Dic[i]))
    for j in Dic[i]:
      Genes.append(j)
  fh.close()
  Genes = list(set(Genes))
  print(file_name, len(Genes))
  return file_name, Genes
GMT_file_HSV, Genes_HSV = Dict_to_gmt(Herpes_wkshell_bukt, discription=''Wk_shell_HSV'', file_name = ''VTTSHI-HSV'')
GMT_file_HPV, Genes_HPV = Dict_to_gmt(Papilloma_wkshell_bukt, discription=''Wk_shell_HPV'', file_name = ''VTTSHI-HPV'')
Genes = list(set(Genes_HSV + Genes_HPV))
print(f"\nTotal Number of genes in {len(Genes)}")
```

21. Perform bucket-to-bucket enrichment analysis Using the GSEApy Python library. Select one VTTSHI gmt file as a target (HSV) and another as a query (HPV).

```
def Module_Enrichment(gmt, Genes, Gene_set, sig = 'Adjusted P-value', col_name = ''Target''):
  enr = gp.enrichr(gene_list=Gene_set,
           gene_sets='GMT_base/HSV.gmt',
           description='test_name',
           outdir='test',
           background=Genes,
           cutoff=1
           )
  enr.results = enr.results.rename(columns={'Term':col_name})
  return enr.results[[col_name, sig, 'Overlap']]
```

22. Perform enrichment of each bucket of the query bucket (HPV) with respect to the target bucket (HSV) Using the following Python script.

Note: A single list of proteins from both networks is used as the background for the enrichment analysis.

```
df_list = []
for S in Papilloma_wkshell_bukt:
  Gene_set = list(Papilloma_wkshell_bukt[S])
```

```
ench_Df = Module_Enrichment('GMT_base/HSV.gmt', Genes, Gene_set).copy()
ench_Df['Source'] = S
ench_Df.loc[ench_Df['Adjusted P-value'] > 0.05, 'Significance'] = 'NE'
ench_Df.loc[ench_Df['Adjusted P-value'] <= 0.05, 'Significance'] = '*'
ench_Df.loc[ench_Df['Adjusted P-value'] <= 0.01,'Significance'] = '**'
ench_Df.loc[ench_Df['Adjusted P-value'] < 0.001, 'Significance'] = '***'
df_list.append(ench_Df)
```

⚠ CRITICAL: Based on enrichment, an 'Adjusted P-value' significance symbol is assigned for further visualization purposes ('***' >> 0.05, > 0.05 '**' <0.01 and > 0.01'*').

23. Combine all enrichment into one pandas dataframe object using The following Python script.

```
df_gseapy = pd.concat(df_list)
df_gseapy.Target.unique()
df_gseapy
```

24. Make a scatter plot of the bucket-to-bucket enrichment analysis using Plotly.

Note: The color represents significance level ('green' >> 0.05, > 0.05, 'gold' <0.01 and > 0.01 'yellow') and a non-circle is not significant (Figure 1A).

```
import plotly.express as px
fig = px.scatter(df_gseapy, x="Target'', y="Source'', color="Significance'', symbol = ''Significance'',
        category_orders={"Target'': ['10_0', '20_10', '30_20', '40_30', '50_40',
                        '60_50', '70_60', '80_70', '90_80', '100_90'][::-1],
                ''Source'': ['10_0', '20_10', '30_20', '40_30', '50_40',
                        '60_50', '70_60', '80_70', '90_80', '100_90'],
                ''Significance'' : ['***', ''**'', ''*'', ''NC''],},
        color_discrete_map={'***' : ''#88C408'', '**' : ''#A69363'', '*': ''#FFD602'', ''NE'': ''#808285''},
        symbol_map ={'***' : 200, '**' : 200, '*': 200, ''NE'': 33},
        template=''plotly_white'',)
fig.update_traces(marker=dict(size=15, line=dict(width=2, color='#144B39')), selector=dict(mode='markers'))
fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)
fig.update_layout(title=f''wk-Shell Enrichment'', autosize=False, width=400, height=430, plot_bgcolor='rgba(0,0,0,0)', xaxis=dict(title=''VTTSHI-HSV''), yaxis=dict(title=''VTTSHI-HPV''), font=dict(family=''Arial'', size=14,))
```

```
Path(''Images'').mkdir(parents=True, exist_ok=True)

fig.write_image(''Images/Enrichment_dot.svg'')

fig.write_image(''Images/Enrichment_dot.png'', scale=2)

fig.write_image(''Images/Enrichment_dot.pdf'')

fig.show()
```

25. Make a Sankey plot that can be used for quantitative analysis, using the following Python script (Figure 1B).

```
def Sankey_Plot_Wkshell(D1,D2, lab1=''A'', lab2=''B'', method=''Wkshell'', spaceing = 0.05):

  def tone_color(H, percent = 50):

    h = H.lstrip('#')

    r,b,g = tuple(int(h[i:i+2], 16) for i in (0, 2, 4))

    a = round(percent/100, 2)

    r,b,g,a = map(str, (r,b,g,a))

    rgba = ''rgba('' + '', ''.join((r,b,g,a))+ '')''

    return(rgba)

  print(f''Length of D1:{len(D1)}, Length of D2:{len(D2)}'')

  if len(D1) != len(D1):

    print(''Not same size!! Killed'')

    return

  if len(set(D1)-set(D1)):

    print(''Dataset doesn't have similar bins!!! Killed'')

    return

  D1 = {lab1 + '' '' +k:v for k,v in D1.items()}

  D2 = {lab2 + '' '' +k:v for k,v in D2.items()}

  print(len(D1[lab1 + '' '' +'100_90']))

  print(f''Generating {2*len(D1)} lables... '')

  label = []

  D1_lab = list(D1)

  if method != ''Wkshell'':

    D1_lab.sort()

  D2_lab = list(D2)

  if method != ''Wkshell'':

    D2_lab.sort()

  Colors = []

  for i in range(len(D1_lab)):

    l1, l2 = D1_lab[i], D2_lab[i]
```

```
    Colors.append(px.colors.qualitative.Plotly[i])
    Colors.append(px.colors.qualitative.Plotly[i])
    label.append(l1)
    label.append(l2)
  print(f''Generating Y {len(D1)} Co-ordinates... '')
  y = [(i+1)/10 for i in range(len(D1))]
  y = y + y
  y.sort()
  print(f''Generating X {len(D1)} Co-ordinates... '')
  x = []
  for i in range(len(D1)):
    x1 = 0.1
    x2 = x1 + spaceing
    x.append(x1)
    x.append(x2)
  print(f''Generating edge data ... '')
  source = []
  target = []
  intersection = []
  Edges_colors = []
  for i in range(len(D1_lab)):
    l1 = D1_lab[i]
    S1 = D1[l1]
    for j in range(len(D2_lab)):
      l2 = D2_lab[j]
      S2 = D2[l2]
      I = (S1.intersection(S2))
      source.append(label.index(l1))
Edges_colors.append(px.colors.qualitative.Plotly[D1_lab.index(l1)])
       target.append(label.index(l2))
       intersection.append(len(I))
  Edges_colors = [tone_color(h) for h in Edges_colors]
  fig = go.Figure(go.Sankey(
  textfont=dict(color=''rgba(0,0,0,0)'', size=1),
  arrangement = ''snap'',
  node = {
     ''label'': label,
     ''x'': x,
```

```
    ''y'': y,
    'pad':10, 'thickness' : 10,
    'color' : Colors,
  },
  link = {
    ''source'': source,
    ''target'': target,
    ''value'': intersection
  }))
  fig.update_traces(orientation='h', selector=dict(type='sankey'))
  fig.update_traces(link_color=Edges_colors, selector=dict(type='sankey'))
  fig.update_layout({
    'plot_bgcolor': 'rgba(0, 0, 0, 0)',
    'paper_bgcolor': 'rgba(0, 0, 0, 0)',})
  fig.update_layout(
  title=f'',
  autosize=False,
  width=1000,
  height=600,
  plot_bgcolor='rgba(0,0,0,0)',
  xaxis=dict(
    title=''Type of Network''),
  yaxis=dict(title=''''))
  fig.update_layout(
  title=f''{lab1} PPI vs {lab2} PPI'',
  font=dict(
    family=''Arial'',
    size=12,
    color=''black''))
  return fig
fig = Sankey_Plot_Wkshell(Herpes_wkshell_bukt, Papilloma_wkshell_bukt, lab1=''HSV'',
lab2=''HPV'', spaceing = 0.1)
fig.write_image(''Images/Sanky_plot.pdf'')
fig.write_image(''Images/Sanky_plot.png'', scale=300)
fig.write_image(''Images/Sanky_count_plot.svg'')
fig.write_html(''Images/Sanky_count_plot.html'')
fig.show()
```

Protein prioritization and viral protein enrichment

Timing: 5 min

To demonstrate the integration of networks centrality and enrichment analysis, we use only VTTSHI-HPV as an example.

```
Papilloma_df = pd.read_csv(Papilloma_Cyto_File)
gene_list = Papilloma_df.name
gene_list
glist = gene_list.squeeze().str.strip().tolist()
print(glist[:10])
```

26. Map UniProt-KB IDs to gene names Using UniProt Python API.

```
url = 'https://www.uniprot.org/uploadlists/'
params = {
'from': 'ACC+ID',
'to': 'GENENAME',
'format': 'tab',
'query': '' ''.join(glist)
}
data = urllib.parse.urlencode(params)
data = data.encode('utf-8')
req = urllib.request.Request(url, data)
with urllib.request.urlopen(req) as f:
  response = f.read()
LOL = []
for i in response.decode('utf-8').splitlines():
  LOL.append(i.split())
df = pd.DataFrame(LOL)
new_header = df.iloc[0]
df = df[1:]
df.columns = new_header
df.head(5)
UniprotKB_to_Genename = dict(zip(df.From, df.To))
Papilloma_df.replace({'shared name': UniprotKB_to_Genename}, inplace=True)
Papilloma_df.rename(columns={''shared name'': ''GeneName''}, inplace=True)
Papilloma_df.rename(columns={''_wks_percentile_bucket'': ''wk-shell''}, inplace=True)
Papilloma_df
```

27. Use the following Python script to keep only the desired columns and discard the rest.

```
Papilloma_df = Papilloma_df[['name','GeneName', 'BetweennessCentrality', 'Closeness Centrality', 'ClusteringCoefficient', 'Degree', 'Radiality', 'Stress', 'Topological Coefficient']]

Papilloma_df.set_index(['name', 'GeneName'], inplace=True)

Papilloma_df.head(5)
```

28. Visualize the centralities selected by to rescaling the values and selecting the top 20 genes from the list.

```
Papilloma_df['Sum'] = Papilloma_df.loc[:,:].sum(axis=1)

Papilloma_df.sort_values(by=['Sum'], ascending=False, inplace=True)

list(Papilloma_df)

Top_20 = Papilloma_df[['BetweennessCentrality', 'ClosenessCentrality', 'ClusteringCoefficient', 'Degree', 'Radiality', 'Stress', 'TopologicalCoefficient']].head(20)

Top_20 -= Top_20.min()

Top_20 /= Top_20.max()

Top_20 = Top_20.reset_index()

Top_20.index += 1

df = Top_20[list(Top_20)[2:]]

Cols = list(df)

Col_dict = {Cols[i]:i+1 for i in range(len(Cols))}

df.index = list(Top_20.GeneName)

df.head(5)

glist = list(df.index)

print(len(glist))

print(*glist, sep='\n')
```

29. Select the enrichment databases that you want to use using the Enrichr databases.

Note: Python script queries all databases with the "Virus" keyword and then selects the "VirusMINT" database. It's possible to use multiple databases or different databases for enrichment analysis depending on the requirements. The documentation and list of all supported databases can be found at ('https://maayanlab.cloud/Enrichr/#libraries').

```
names = gp.get_library_name(organism='Human') # default: Human

for db in names:

  if 'VIRUS' in db.upper():

    print(db)

gene_sets=['VirusMINT']

gene_sets
```

30. Perform enrichment analysis using The Enrichr Python API as in the following Python script.

```
enr = gp.enrichr(gene_list=glist, gene_sets=gene_sets, organism='Human', description=
'VirusMINT', outdir='VirusMINT', cutoff=1)

enr.results.head(5)

encrihment_df = enr.results[enr.results['Adjusted P-value'] <= 0.005]

Encrihment_dict = defaultdict(list)

for Term, Genes in encrihment_df[['Term', 'Genes']].values.tolist():

  for gene in Genes.split(';'):

    Encrihment_dict[gene].append(Term)

for i in Encrihment_dict:

  Encrihment_dict[i] = ''<br>''.join(Encrihment_dict[i])

Encrihment_dict = dict(Encrihment_dict)
```

31. Preproces the Enrichr results to create a dot plot with the following Python script.

```
x = []

y = []

size = []

text = []

groups = []

for c in Col_dict:

  I = list(df[c].index)

  for i in I:

    y.append(i)

    x.append(c)

    E = 'NE'

    if i in Encrihment_dict:

      E = Encrihment_dict[i]

    groups.append(E)

  for s in df[c].values:

    size.append(s)

    text.append(''<br>''.join([str(round(s, 2)), c]))

Dot_plot_df = pd.DataFrame({''Centrality'':x,

                ''Protein'':y,

                ''Value'':size,

                ''Info'':text,

                ''Encrichment'':groups})

Dot_plot_df.head(5)
```

32. Make a dot plot using the Plotly Python library (Figure 1C).

```
fig = px.scatter(Dot_plot_df, x=''Centrality'', y=''Protein'', size="Value'', color=
''Encrichment'', hover_data=['Info'])

fig.update_traces(mode='markers', marker_symbol = 200, marker_line_width=2, marker_line_
color='rgba(0, 0, 0, 1)')

fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)

fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)

fig.update_layout(title=None, autosize=False, width=420, height=910, plot_bgcolor=
'rgba(0,0,0,0)', xaxis=dict(

   title=''Centralities''),

  yaxis=dict(title=''Protein''),

  font=dict(family=''Arial'', size=14, color=''black''))

fig.update_layout(

  legend=dict(orientation=''v'', yanchor=''bottom'', y=1.02, xanchor=''right'', x=1,
traceorder=''reversed'',

   title_font_family=''Times New Roman'',

   font=dict(family=''Arial'', size=12, color=''black''), bgcolor=''rgba(0,0,0,0)'',
bordercolor=''Black'', borderwidth=2)

)

fig.write_image(''Images/Dot_plot.png'', scale=2)

fig.write_image(''Images/Dot_plot.pdf'')

fig_dots = fig

fig_dots.show()
```

a. generate a relative dot size legend using the following Python script.

```
fig = px.scatter(x=['', '', '', '', ''], y=[0, 0.25, 0.5, 0.75, 1], size = [0, .25, .50, .75, 1])

fig.update_traces(mode='markers', marker_symbol = 200, marker_line_width=2, marker_line_
color='rgba(0, 0, 0, 1)')

fig.update_traces(marker=dict(color='lightgray'))

fig.update_xaxes(visible=False)

fig.update_yaxes(ticklabelposition="outside right'', side= 'right')

fig.update_yaxes(tick0=0, dtick=0.25)

fig.update_xaxes(tick0=0, dtick=0)

fig.update_xaxes(showline=True, linewidth=2, linecolor='black', mirror=True)

fig.update_yaxes(showline=True, linewidth=2, linecolor='black', mirror=True)

fig.update_layout(title=f'Values', autosize=False, width=200, height=290, plot_bgco-
lor='rgba(0,0,0,0)', yaxis=dict(title=None))

fig.write_image(''Images/Dot_legend_plot.png'', scale=300)

fig_radius = fig

fig_radius.show()
```

⚠ CRITICAL: VTTSHI-HPV serves as an example of the integration of networks centrality and enrichment analysis. However, the same steps can be applied to any data set, VTTSHI-HPV, or other appropriate datasets.

EXPECTED OUTCOMES

Successful execution of the protocol generates plots by using the Python library. In addition, the pipeline is flexible enough to allow users to export desired datasets to the disk. Wk-shell bucket enrichment scatter plots show significant enrichment diagonally as a rule of thumb, but in general, they should show significant enrichment in the lower-left corner. It would be a clear sign that the two networks are completely different if enrichment is observed at different wk-shell buckets compared to the corresponding wk-shell buckets of the two networks. Analysis of network centralities and enrichment has mainly two components: (a) analysis of network centrality and (b) enrichment. Both of these components are flexible and scalable. These are some of the most important centralities in the Cytoscape app that are included in this pipeline. It is expected that the central nodes will be extracted effectively using these centralities. Furthermore, in addition to these centralities, more complicated centralities such as PageRank might improve the results, but those are not available in Cytoscape, so users may use other tools to calculate PageRank and other centralities, such as Gephi, NetworkX, etc. Later in this pipeline, we used enrichment analysis as the second component, and the GSEApy Python library was used for the enrichment. Our purpose requires a database of viral-human protein interactions, soVirusMINT is suitably chosen. However, users can choose other databases that suit specific viruses or pathogens.

LIMITATIONS

Computational framework

For this pipeline to work, a robust and significantly overlapping set of PPI networks is essential. It would result in a huge network once all of them are combined if the PPIs are not adequately overlapping. In addition, it may result in several small, non-connected subnetworks within the merged PPIs. The network centrality analysis would be affected as a result. Specifically, the wk-shell analysis, which assumes that the most essential nodes are at the center and only consider the largest connected component, assigns the most outer shell to smaller, unconnected network nodes. Choosing appropriate enrichment databases or libraries is the next aspect of this framework. GSEApy is a Python package that supports multiple enrichment libraries simultaneously as well as a custom enrichment library using a custom *.gmt file. In this pipeline, a .gmt file is generated to each wk-shell percentile bucket for two networks using Python. A .gmt file can also be created using MS Excel or LibreOffice Calc. You can learn more about the .gmt file by reading the broadinstitute format description (https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats#:~:text=The%20GMT%20file%20format%20is,genes%20in%20the%20gene%20set.)

Computational resources

An extensive amount of graph and network integration, annotation, and enrichment analysis is involved in this protocol. Our recommendation is to run the protocol on a high-performance cluster as the calculations are computationally intensive. Depending on the number of PPI networks, the size of the PPI network, the complexity of the network, and the number of viral target proteins, memory and CPU resources may be required for each step (See step-by-step method details). Using this method on smaller datasets will require lower memory requirements, so that it can be run on a PC with limited computational resources, for example. When utilizing the HPC cluster, the user is required to modify the job script accordingly when it comes to the amount of memory and processors needed for each step.

Data access

Almost all the datasets used in this protocol are open source and don't require special permission, however, they do have copy-write restrictions. Please read the instructions on the original download page and cite databases properly before downloading.

Computational time

There are several factors that determine how much time and computational resources are required for downloading and processing network and viral target datasets. Our recommendation is to process a small number of network dataset files in order to guide the computational resources required for large-scale analyses. Using this protocol, the user will have to adjust the time needed for each step in the job script accordingly if using HPC clusters.

Basic programming knowledge

The protocol is based on a Python-Jupyter notebook schema and requires knowledge of the Python programming language especially practical knowledge of pandas and NetworkX Python libraries. In addition, knowledge of the R scripting language is also required for DESeq2 analysis. The first-time users will need to learn some basic commands such as ls, echo, cd, rm, mv, mkdir, tree, sed, grep, pwd, etc.

TROUBLESHOOTING

Problem 1

A common warning may be displayed for Python pandas dataframe mathematical oppressions.

```
/scratch/local/ipykernel_223459/3924348524.py:1: 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
```

Potential solution

It can be ignored by suppressing all warnings in Python using the following Python command.

```
import warnings

warnings.filterwarnings('ignore')

# or

np.warnings.filterwarnings('ignore')
```

Problem 2

Efficiencies in annotation are crucial to the whole pipeline. During the DESeq2 analysis, we used the AnnotationDbi R library to retrieve the UniProt annotation. Unfortunately, there were a lot of missing annotations. This might affect network integration.

Potential solution

To map the desired id type, UniProt "id mapping API" (https://www.uniprot.org/help/api_idmapping) may be used. APIs are currently available in Python 2/3, Java, Ruby, and Perl.

Problem 3

It is common for network data from different sources to use a variety of identifiers and, as a result, be ambiguous and redundant at the same time. Two databases may refer to the same gene but use two different synonyms. Additionally, some identifiers, such as 'Gene symbols,' have some gene names that differ according to some software (for example, SEP1 is interpreted as September 1st in Microsoft Excel).

Potential solution

It is necessary to convert identifiers from different sources into identifiers that have limited or no synonyms in order to eliminate ambiguity and reduce redundancy. The identifiers are all converted to

UniProt-KB IDs within this pipeline. You should also avoid using the gene symbol for any integration and use text editors like Geany and Notepad++ as opposed to spreadsheet software programs like Microsoft Excel. Python pandas and the R dataframe or a similar approach are recommended for manipulating and processing data.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, M. Shahid Mukhtar (smukhtar@uab.edu).

Materials availability

This study did not generate new or unique reagents.

Data and code availability

The data/analyses presented in the current protocol have been deposited in GitHub (https://github.com/nilesh-iiita/VTTSHI) and Zenodo (https://doi.org/10.5281/zenodo.6800259).

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (IOS-1557796 and IOS-2038872) to M.S.M.

AUTHOR CONTRIBUTIONS

N.K. assembled the pipeline and wrote the manuscript. B.M., N.K., and M.S.M. originally devised and refined the protocol. N.K. and B.M. performed the first analyses. M.S.M. and N.K. conceived the project. M.S.M. supervised the analysis. All authors read and edited the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Ahmed, H., Howton, T.C., Sun, Y., Weinberger, N., Belkhadir, Y., and Mukhtar, M.S. (2018). Network biology discovers pathogen contact points in host protein-protein interactomes. Nat. Commun. *9*, 2312–2313.

Ammari, M.G., Gresham, C.R., McCarthy, F.M., and Nanduri, B. (2016). HPIDB 2.0: a curated database for host-pathogen interactions. Database *2016*, :baw103. https://doi.org/10.1093/database/baw103.

Arabidopsis Interactome Mapping, C. (2011). Evidence for network evolution in an Arabidopsis interactome map. Science *333*, 601–607. https://doi.org/10.1126/science.1203877.

Barabasi, A.L., Gulbahce, N., and Loscalzo, J. (2011). Network medicine: a network-based approach to human disease. Nat. Rev. Genet. *12*, 56–68. https://doi.org/10.1038/nrg2918.

Bosl, K., Ianevski, A., Than, T.T., Andersen, P.I., Kuivanen, S., Teppor, M., Zusinaite, E., Dumpis, U., Vitkauskiene, A., Cox, R.J., et al. (2019). Common nodes of virus-host interaction revealed through an integrated network analysis. Front. Immunol. *10*, 2186. https://doi.org/10.3389/fimmu.2019.02186.

Casadevall, A., and Pirofski, L.A. (2000). Host-pathogen interactions: basic concepts of microbial commensalism, colonization, infection, and disease. Infect. Immun. *68*, 6511–6518. https://doi.org/10.1128/IAI.68.12.6511-6518.2000.

Feaugas, T., and Sauvonnet, N. (2021). Organ-on-chip to investigate host-pathogens interactions. Cell Microbiol. *23*, e13336. https://doi.org/10.1111/cmi.13336.

Garbutt, C.C., Bangalore, P.V., Kannar, P., and Mukhtar, M.S. (2014). Getting to the edge: protein dynamical networks as a new frontier in plant–microbe interactions. Front. Plant Sci. *5*, 312.

González-Fuente, M., Carrère, S., Monachello, D., Marsella, B.G., Cazalé, A.C., Zischek, C., Mitra, R.M., Rezé, N., Cottret, L., Mukhtar, M.S., et al. (2020). EffectorK, a comprehensive resource to mine for Ralstonia, Xanthomonas, and other published effector interactors in the Arabidopsis proteome. Mol. Plant Pathol. *21*, 1257–1270.

Gordon, D.E., Jang, G.M., Bouhaddou, M., Xu, J., Obernier, K., White, K.M., O'Meara, M.J., Rezelj, V.V., Guo, J.Z., Swaney, D.L., et al. (2020). A SARS-CoV-2 protein interaction map reveals targets for drug repurposing. Nature *583*, 459–468. https://doi.org/10.1038/s41586-020-2286-9.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX (Los Alamos National Lab (LANL)).

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (2020). Array programming with NumPy. Nature *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Hein, M.Y., Hubner, N.C., Poser, I., Cox, J., Nagaraj, N., Toyoda, Y., Gak, I.A., Weisswange, I., Mansfeld, J., Buchholz, F., et al. (2015). A human interactome in three quantitative dimensions organized by stoichiometries and abundances. Cell *163*, 712–723. https://doi.org/10.1016/j.cell.2015.09.053.

Hunter, J.D. (2007). Matplotlib: a 2D graphics environment. Comput. Sci. Eng. *9*, 90–95. https://doi.org/10.1109/Mcse.2007.55.

Huttlin, E.L., Ting, L., Bruckner, R.J., Gebreab, F., Gygi, M.P., Szpyt, J., Tam, S., Zarraga, G., Colby, G., Baltier, K., et al. (2015). The BioPlex network: a systematic exploration of the human interactome. Cell *162*, 425–440. https://doi.org/10.1016/j.cell.2015.06.043.

Inc., P.T. (2022). Collaborative data science. https://plot.ly.

Klopffleisch, K., Phan, N., Augustin, K., Bayne, R.S., Booker, K.S., Botella, J.R., Carpita, N.C., Carr, T., Chen, J.G., Cooke, T.R., et al. (2011). Arabidopsis G-protein interactome reveals connections to cell

wall carbohydrates and morphogenesis. Mol. Syst. Biol. *7*, 532.

Kumar, N., Mishra, B., Mehmood, A., Mohammad, A., and Mukhtar, M.S. (2020). Integrative network biology framework elucidates molecular mechanisms of SARS-CoV-2 pathogenesis. iScience *23*, 101526. https://doi.org/10.1016/j.isci.2020.101526.

Kumar, R., and Nanduri, B. (2010). HPIDB–a unified resource for host-pathogen interactions. BMC Bioinf. *11*, S16. https://doi.org/10.1186/1471-2105-11-S6-S16.

Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., and Pfister, H. (2014). UpSet: visualization of intersecting sets. IEEE Trans. Vis. Comput. Graph. *20*, 1983–1992. https://doi.org/10.1109/TVCG.2014.2346248.

Liu, J., Fakhar, A.Z., Pajerowska-Mukhtar, K.M., and Mukhtar, M.S. (2022). A TIReless battle: TIR domains in plant-pathogen interactions. Trends Plant Sci. *27*, 426–429. https://doi.org/10.1016/j.tplants.2022.01.011.

Long, T., Burk, R.D., Chan, P.K.S., and Chen, Z. (2022). Non-human primate papillomavirus E6-mediated p53 degradation reveals ancient evolutionary adaptation of carcinogenic phenotype to host niche. PLoS Pathog. *18*, e1010444. https://doi.org/10.1371/journal.ppat.1010444.

Lopez, J., and Mukhtar, M.S. (2017). Mapping protein-protein interaction using high-throughput yeast 2-hybrid. In Plant Genomics, Wolfgang Busch, ed. (Humana Press), pp. 217–230.

Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. *15*, 550. https://doi.org/10.1186/s13059-014-0550-8.

Luck, K., Kim, D.K., Lambourne, L., Spirohn, K., Begg, B.E., Bian, W., Brignall, R., Cafarelli, T., Campos-Laborie, F.J., Charloteaux, B., et al. (2020). A reference map of the human binary protein interactome. Nature *580*, 402–408. https://doi.org/10.1038/s41586-020-2188-x.

McCormack, M.E., Lopez, J.A., Crocker, T.H., and Mukhtar, M.S. (2016). Making the right connections: network biology and plant immune system dynamics. Curr. Plant Biol. *5*, 2–12.

McKinney, W. (2010). Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference *445*, 51–56.

Mishra, B., Athar, M., and Mukhtar, M.S. (2021a). Transcriptional circuitry atlas of genetic diverse unstimulated murine and human macrophages define disparity in population-wide innate immunity. Sci. Rep. *11*, 7373–7419.

Mishra, B., Kumar, N., and Mukhtar, M.S. (2019). Systems biology and machine learning in plant-pathogen interactions. Mol. Plant Microbe Interact. *32*, 45–55. https://doi.org/10.1094/MPMI-08-18-0221-FI.

Mishra, B., Kumar, N., and Mukhtar, M.S. (2021b). Network biology to uncover functional and structural properties of the plant immune system. Curr. Opin. Plant Biol. *62*, 102057. https://doi.org/10.1016/j.pbi.2021.102057.

Mishra, B., Kumar, N., and Shahid Mukhtar, M. (2022). A rice protein interaction network reveals high centrality nodes and candidate pathogen effector targets. Comput. Struct. Biotechnol. J. *20*, 2001–2012. https://doi.org/10.1016/j.csbj.2022.04.027.

Mishra, B., Sun, Y., Ahmed, H., Liu, X., and Mukhtar, M.S. (2017). Global temporal dynamic landscape of pathogen-mediated subversion of Arabidopsis innate immunity. Sci. Rep. *7*, 7849–7913.

Mishra, B., Sun, Y., Howton, T.C., Kumar, N., and Mukhtar, M.S. (2018). Dynamic modeling of transcriptional gene regulatory network uncovers distinct pathways during the onset of Arabidopsis leaf senescence. NPJ Syst. Biol. Appl. *4*, 35–44.

Mootha, V.K., Lindgren, C.M., Eriksson, K.F., Subramanian, A., Sihag, S., Lehar, J., Puigserver, P., Carlsson, E., Ridderstråle, M., Laurila, E., et al. (2003). PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat. Genet. *34*, 267–273. https://doi.org/10.1038/ng1180.

Mott, G.A., Smakowska-Luzan, E., Pasha, A., Parys, K., Howton, T.C., Neuhold, J., Lehner, A., Grünwald, K., Stolt-Bergner, P., Provart, N.J., et al. (2019). Map of physical interactions between extracellular domains of Arabidopsis leucine-rich repeat receptor kinases. Sci. Data *6*, 190025–190026.

Mukhtar, M.S., Carvunis, A.R., Dreze, M., Epple, P., Steinbrenner, J., Moore, J., Tasan, M., Galli, M., Hao, T., Nishimura, M.T., et al. (2011). Independently evolved virulence effectors converge onto hubs in a plant immune system network. Science *333*, 596–601. https://doi.org/10.1126/science.1203659.

Mukhtar, M.S., McCormack, M.E., Argueso, C.T., and Pajerowska-Mukhtar, K.M. (2016). Pathogen tactics to manipulate plant cell death. Curr. Biol. *26*, R608–R619.

Naqvi, R.Z., Zaidi, S.S.-e.-A., Mukhtar, M.S., Amin, I., Mishra, B., Strickler, S., Mueller, L.A., Asif, M., and Mansoor, S. (2019). Transcriptomic analysis of cultivated cotton Gossypium hirsutum provides insights into host responses upon whitefly-mediated transmission of cotton leaf curl disease. PLoS One *14*, e0210011. https://doi.org/10.1371/journal.pone.0210011.

Qin, T., Koneva, L.A., Liu, Y., Zhang, Y., Arthur, A.E., Zarins, K.R., Carey, T.E., Chepeha, D., Wolf, G.T., Rozek, L.S., and Sartor, M.A. (2020). Significant association between host transcriptome-derived HPV oncogene E6* influence score and carcinogenic pathways, tumor size, and survival in head and neck cancer. Head Neck *42*, 2375–2389. https://doi.org/10.1002/hed.26244.

Qin, T., Zhang, Y., Zarins, K.R., Jones, T.R., Virani, S., Peterson, L.A., McHugh, J.B., Chepeha, D., Wolf, G.T., Rozek, L.S., and Sartor, M.A. (2018). Expressed HNSCC variants by HPV-status in a well-characterized Michigan cohort. Sci. Rep. *8*, 11458. https://doi.org/10.1038/s41598-018-29599-w.

Schneweis, K.E., Forstbauer, H., Olbrich, M., and Tag, M. (1984). Pathogenesis of genital herpes simplex virus infection in mice. III. Comparison of the virulence of wild and mutant strains. Med. Microbiol. Immunol. *173*, 187–196. https://doi.org/10.1007/BF02122110.

Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. *13*, 2498–2504. https://doi.org/10.1101/gr.1239303.

Smakowska-Luzan, E., Mott, G.A., Parys, K., Stegmann, M., Howton, T.C., Layeghifard, M., Neuhold, J., Lehner, A., Kong, J., Grünwald, K., et al. (2018). An extracellular network of Arabidopsis leucine-rich repeat receptor kinases. Nature *553*, 342–346.

Spears, B.J., Howton, T.C., Gao, F., Garner, C.M., Mukhtar, M.S., and Gassmann, W. (2019). Direct regulation of the EFR-dependent immune response by Arabidopsis TCP transcription factors. Mol. Plant Microbe Interact. *32*, 540–549.

Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., and Mesirov, J.P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc. Natl. Acad. Sci. USA *102*, 15545–15550. https://doi.org/10.1073/pnas.0506580102.

Szklarczyk, D., Gable, A.L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N.T., Morris, J.H., Bork, P., et al. (2019). STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Res. *47*, D607–D613. https://doi.org/10.1093/nar/gky1131.

Vidal, M., Cusick, M.E., and Barabási, A.L. (2011). Interactome networks and human disease. Cell *144*, 986–998. https://doi.org/10.1016/j.cell.2011.02.016.

Wan, C., Borgeson, B., Phanse, S., Tu, F., Drew, K., Clark, G., Xiong, X., Kagan, O., Kwan, J., Bezginov, A., et al. (2015). Panorama of ancient metazoan macromolecular complexes. Nature *525*, 339–344. https://doi.org/10.1038/nature14877.

Washington, E.J., Mukhtar, M.S., Finkel, O.M., Wan, L., Banfield, M.J., Kieber, J.J., and Dangl, J.L. (2016). Pseudomonas syringae type III effector HopAF1 suppresses plant immunity by targeting methionine recycling to block ethylene induction. Proc. Natl. Acad. Sci. USA *113*, E3577–E3586.

Watkins, J.M., Clark, N.M., Song, G., Oliveira, C.C., Mishra, B., Brachova, L., Seifert, C.M., Mitchell, M.S., dos Reis, P.A.B., Urano, D., et al. (2021). Phosphorylation dynamics in a flg22-induced, heterotrimeric G-protein dependent signaling network in Arabidopsis thaliana reveals a candidate PP2A phosphatase involved in AtRGS1 trafficking. Preprint at bioRxiv. https://doi.org/10.1101/2021.12.06.471472.

Wessling, R., Epple, P., Altmann, S., He, Y., Yang, L., Henz, S.R., McDonald, N., Wiley, K., Bader, K.C., Gläßer, C., et al. (2014). Convergent targeting of a common host protein-network by pathogen effectors from three kingdoms of life. Cell Host Microbe *16*, 364–375. https://doi.org/10.1016/j.chom.2014.08.004.

Yang, H., Ke, Y., Wang, J., Tan, Y., Myeni, S.K., Li, D., Shi, Q., Yan, Y., Chen, H., Guo, Z., et al. (2011). Insight into bacterial virulence mechanisms against host immune response via the Yersinia pestis-human protein-protein interaction network. Infect. Immun. *79*, 4413–4424. https://doi.org/10.1128/IAI.05622-11.

Zaidi, S.S., Naqvi, R.Z., Asif, M., Strickler, S., Shakir, S., Shafiq, M., Khan, A.M., Amin, I., Mishra, B.,

Mukhtar, M.S., et al. (2020). Molecular insight into cotton leaf curl geminivirus disease resistance in cultivated cotton (Gossypium hirsutum). Plant Biotechnol. J. *18*, 691–706. https://doi.org/10.1111/pbi.13236.

Zhang, Y., Koneva, L.A., Virani, S., Arthur, A.E., Virani, A., Hall, P.B., Warden, C.D., Carey, T.E., Chepeha, D.B., Prince, M.E., et al. (2016). Subtypes of HPV-positive head and neck cancers are associated with HPV characteristics, copy number alterations, PIK3CA mutation, and pathway signatures. Clin. Cancer Res. *22*, 4735–4745. https://doi.org/10.1158/1078-0432.CCR-16-0323.

Zhou, Y., Hou, Y., Shen, J., Huang, Y., Martin, W., and Cheng, F. (2020). Network-based drug repurposing for novel coronavirus 2019-nCoV/SARS-CoV-2. Cell Discov. *6*, 14. https://doi.org/10.1038/s41421-020-0153-3.

Zhu, S., and Viejo-Borbolla, A. (2021). Pathogenesis and virulence of herpes simplex virus. Virulence *12*, 2670–2702. https://doi.org/10.1080/21505594.2021.1982373.