NIH Dashabord

AUTHOR
Marlene Lin, Skylar Lyu, Haofei Sun, Qin Lin

PUBLISHED
March 23, 2022

NIH Grant & Funding Data Dashboard

Winter 2023 PIC16B Final Project

finalpost_files/img/nih.png

Background

Begun as a one-room Laboratory of Hygiene in 1887, the National Institutes of Health (NIH) today is one of the world's foremost medical research centers. It is also the largest public funder of biomedical research in the world, investing more than $32 billion a year to enhance life, and reduce illness and disability.

Yet, the devil is in the details. The painstaking process of grant application and review has long been questioned by many researchers that spent more time writing up proposals than doing actual research. With a blogpost on 'the inequality in NIH funding distribution' published last year by the NIH office, more questions regarding how the money was spent were brought up by the researchers. In this project, we attempt to answer some of their key questions through data analysis and visualization using a web-based dashboard.

Git: https://github.com/linnilinnil/NIH-Fundings-Dashboard/tree/main

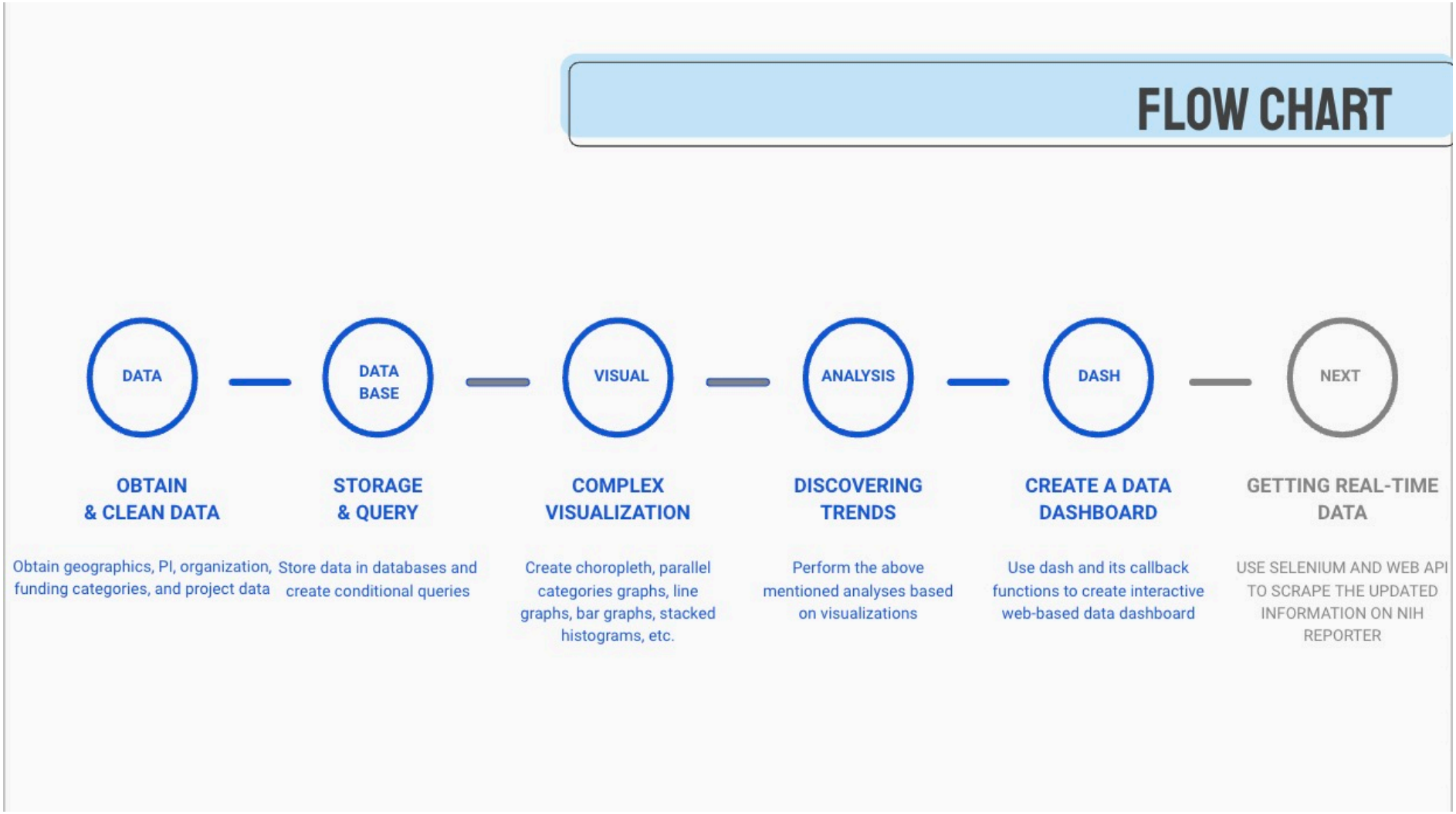
whatknow.png

Overview

Our analysis contains four aspects:

four.png

The technical components involved are: - SQL Database interaction - Complex visualization with Plotly - Web-based application with flask + Dash - Data collection and cleaning (some through API request)



flow.png

We would be using the following packages and libraries:

```
# importing dependencies
import os
import glob
import regex as re
import json
from urllib.request import urlopen
```

```
import requests
import time
from geopy import Nominatim
import sqlite3
# dash
import dash
from dash import Dash, dcc, html
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output, State

# plotting and df
import plotly
print(plotly.__version__)
import plotly.graph_objs as go
import plotly.express as px
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

import io
import PIL
r = np.random.RandomState(42)
```

```
ModuleNotFoundError: No module named 'geopy'
```

```
import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

Data cleaning (TL;DR)

We first take a look at this data:
NIH Awards: by project names, prpoject number (Activity code included) Location, Organization, admin IC, funding mechanisms, direct costs, and indirect costs per fiscal year and project number https://report.nih.gov/award/index.cfm?ot=&fy=2023&state=&ic=&fm=&orgid=&distr=&rfa=&om=n&pid=#tab5

To create a map visualization, we would be needing the relevant FIPS code for states and counties.
We import a city FIPS table for geospatial data display. The following dataset maps county (all upper cases) to their FIPs, because county FIPS are needed to visualize the county map in plotly.

```
cfips = pd.read_csv("county-fips.csv")
cfips["full_name"] = (cfips["county_name"]+" "+cfips["state_name"]).str.upper()
fip_dict = dict(zip(list(cfips["full_name"]),list(cfips["fips"])))
list(fip_dict.items())[1:5]
```

```
[('BALDWIN COUNTY ALABAMA', 1003),
 ('BARBOUR COUNTY ALABAMA', 1005),
 ('BIBB COUNTY ALABAMA', 1007),
 ('BLOUNT COUNTY ALABAMA', 1009)]
```

And the following table is capable of mapping city to their respective county.

```
# just downloaded the county and city fips code from us census data and mapped
city = pd.read_csv("uscities.csv")
city.city = city.city.str.upper()
city.county_name = city.county_name.str.upper()
city.state_name = city.state_name.str.upper()
city = city.dropna()
# store data in database data.db table name "decade"
try:
    conn = sqlite3.connect("data.db")
    city.to_sql("city",conn,index=False)
    conn.close()
except:
    print("The city table already existed.")
```

```
The city table already existed.
```

However, since this table does not contain information on all cities, we make further use of the geolocator module to fill in the missing information.

```
# retrieving location: longitude and latitude based on city and state
geolocator = Nominatim(user_agent="app")
```

The following would be a dict to map state full name to state two-letter code. Plotly has an option to display state map by state code (two letters).

```
anym = pd.read_csv("acronynm.csv")
c_dict = dict(zip(list(anym.state.str.upper()),list(anym.code)))
```

Now we fill in some additional information to make mapping geo codes and future visualization easier: - MONTH: derive from award notice date - YEAR: derive from award notice data also - CITY, STATE: convert to upper - FULL_LOC: combine city and state, make searching with geo_locator easier - CODE: state code

```
data = pd.DataFrame()
path = "/Users/linlin/Desktop/2023/16b/dash/awardsbyloc"
xls_list = os.listdir(path)
xlsx = glob.glob(os.path.join(path, "*.xls"))
for x in xlsx:
    if '.DS_Store' not in x:
        print(x)
        df = pd.read_excel(x)
        df.columns = [s.upper() for s in list(df.columns)]
        df["MONTH"] = pd.to_datetime(df["AWARD NOTICE DATE"]).dt.month
        df["YEAR"] = pd.to_datetime(df["AWARD NOTICE DATE"]).dt.year
        df["CITY"] = df["CITY"].str.upper()
        df["STATE OR COUNTRY NAME"] = df["STATE OR COUNTRY NAME"].str.upper()
        df["FULL_LOC"] = df["CITY"] + " " + df["STATE OR COUNTRY NAME"]
        df['CODE'] = df["STATE OR COUNTRY NAME"].map(c_dict)
        if len(data) == 0:
            data = df
        else:
            data = pd.concat([data, df],    # Combine vertically
                             ignore_index = False,
                             sort = False)
```

We retain only the relevant columns in the dataset.

```
data = data[['ORGANIZATION NAME', 'ORGANIZATION ID (IPF)', 'PROJECT NUMBER',
       'FUNDING MECHANISM', 'PI NAME', 'PI PERSON ID',
       'PROJECT TITLE', 'DIRECT COST', 'INDIRECT COST', 'FUNDING',
       'CITY', 'STATE OR COUNTRY NAME',
       'INSTITUTION TYPE', 'AWARD NOTICE DATE','MONTH', 'YEAR',
       'FULL_LOC', 'CODE']]
data = data[data.YEAR.notna()]
data.shape
```

```
(611898, 18)
```

The different funding mechanisms and institution types of NIH Project and Funding Grants are:

```
data['INSTITUTION TYPE'] = data['INSTITUTION TYPE'].fillna('None')
data['INSTITUTION TYPE'].unique()
```

```
array(['None', 'Research Institutes', 'Domestic Higher Education',
       'Independent Hospitals'], dtype=object)
```

```
data['FUNDING MECHANISM'].unique()
```

```
array(['RPGs - SBIR/STTR', 'RPGs - Non SBIR/STTR',
       'Other Research-Related', 'Training - Individual',
       'Training - Institutional', 'Research Centers', 'Other',
       'Construction'], dtype=object)
```

About funding mechanisms: (this block is written by Qin Lin)

The National Institutes of Health (NIH) uses a variety of funding mechanisms to support medical research and advance public health. These funding mechanisms are designed to provide support for researchers at all stages of their careers and for a wide range of research topics.

One common type of funding mechanism is the research project grant (R01), which provides funding for a specific research project for up to five years. These grants are awarded based on the scientific merit of the proposed research, as determined by peer review.

Other types of funding mechanisms include program project grants (P01), which provide support for a group of related research projects; center grants (P50), which support multidisciplinary research centers; and training grants (T32), which provide funding for graduate and postdoctoral training programs.

The NIH also offers small business innovation research (SBIR) and small business technology transfer (STTR) grants, which provide funding to small businesses for research and development of new technologies and therapies.

Additionally, the NIH provides funding for research infrastructure through facilities and equipment grants (S10), which support the purchase of large, expensive equipment or the construction or renovation of research facilities.

Through these different funding mechanisms, the NIH supports a broad range of research and researchers, from basic science to clinical trials, with the goal of advancing medical knowledge and improving public health.

Since we have a huge amount of data —— over 600k entries, it would be wise to store and query the data using databases.

```
# store data in database data.db table name "decade"
conn = sqlite3.connect("data.db")
cur = conn.cursor()
cur.execute("DROP TABLE IF EXISTS decade")
conn.commit()
data.to_sql("decade",conn,index=False)
```

```
611898
```

Now, it's time to write a query function to select data by fiscal year, funding mechanism, and insitution type. Additionally, we would want to map the data to corresponding geocodes, and fill in anything that is missing with the geolocator module. (The functions are saved in separate python files as required.)

Let's write a small helper function to print all tables in a given database, too.

```
def print_table(db = 'data.db'):
    conn = sqlite3.connect(db)
    alltbl = """SELECT name FROM sqlite_master
    WHERE type='table';"""
    cur = conn.cursor()
    print(cur.execute(alltbl).fetchall())
```

```
    conn.close()
print_table()
```

```
[('decade',), ('city',)]
```

```
FME = ['RPGs - SBIR/STTR','RPGs - Non SBIR/STTR','Other Research-Related','Training - Individual',
       'Training - Institutional','Research Centers','Other','Construction']
INST = ['None','Research Institutes', 'Domestic Higher Education','Independent Hospitals']
col = ['ORGANIZATION NAME','ORGANIZATION ID (IPF)','PROJECT NUMBER','FUNDING MECHANISM','PI NAME','PI PERSON ID',
 'PROJECT TITLE','DIRECT COST','INDIRECT COST','FUNDING', 'CITY', 'STATE OR COUNTRY NAME', 'INSTITUTION TYPE',
 'AWARD NOTICE DATE', 'MONTH', 'YEAR', 'FULL_LOC', 'CODE']
def map_que(year=list(np.arange(2012,2022)),month=list(np.arange(1,12)),
            fme = FME,
            inst = INST):
        cmd = '''
        SELECT D.*,C.COUNTY_FIPS,C.COUNTY_NAME,C.LAT,C.LNG
        FROM
                (SELECT LNG,LAT,COUNTY_NAME,CITY,COUNTY_FIPS,STATE_ID
                FROM city) AS C
        LEFT JOIN decade as D
        ON D.CITY = C.CITY
        WHERE D.YEAR IN ''' + str(tuple(year))\
        + ''' AND MONTH IN ''' + str(tuple(month))\
        + ''' AND D.[FUNDING MECHANISM] IN ''' + str(tuple(fme))\
        + ''' AND D.[INSTITUTION TYPE] IN ''' + str(tuple(inst))#+'''LIMIT 1'''
        conn = sqlite3.connect("data.db")
        cursor = conn.cursor()
        cursor.execute(cmd)
        df = pd.DataFrame(cursor.fetchall(),columns=col+["COUNTY_FIPS","COUNTY_NAME","LAT","LNG"])
        print(df.head(n=1))
        conn.close()
        return df
df = map_que()
```

```
  ORGANIZATION NAME  ORGANIZATION ID (IPF)   PROJECT NUMBER FUNDING MECHANISM  \
0       5 POINT APP             10052988.0  1R43CA261446-01  RPGs - SBIR/STTR
```

```
         PI NAME  PI PERSON ID  \
0  BECK, SUSAN L     6170838.0

                                      PROJECT TITLE  DIRECT COST  \
0  A New Multi-functional and Connected Mobile Ap...          NaN

   INDIRECT COST  FUNDING  ... INSTITUTION TYPE    AWARD NOTICE DATE MONTH  \
0            NaN   397367  ...             None  2021-09-16 00:00:00   9.0

     YEAR           FULL_LOC  CODE COUNTY_FIPS COUNTY_NAME      LAT      LNG
0  2021.0  NEW YORK NEW YORK    NY       36081      QUEENS  40.6943 -73.9249

[1 rows x 22 columns]
```

The query took quite long to run on my MAC, some where between 1 to 3 minutes, so ideally we might want to query all of them when we open the web app and subset for different filters. Or use cache to store some of the query results for a real website.

Then, we fill in the missing locations with the following function:

```
def geo_encode(dat,fip_dict):
    proj = dat.copy()
    #BECAUSE THERE ARE DUPLICATED VALUES AND USUALLY THE FIRST OCCURENCES ARE RIGHT
    locs = list(dat.FULL_LOC)
    lngs =  list(dat.LNG)
    lats = list(dat.LAT)
    fips = list(dat.COUNTY_FIPS)
    countys = list(dat.COUNTY_NAME)

    lon_dict = dict(zip(list(locs),list(lngs)))
    lat_dict = dict(zip(list(locs),list(lats)))
    ct_dict = dict(zip(list(locs),list(countys)))
    fip_dict = dict(zip(list(locs),list(fips)))
    notfound = (proj[proj["LAT"].isna()]["FULL_LOC"]).unique()
    #print(notfound)

    geo = [geolocator.geocode(i) for i in notfound]
    lat = [i.latitude if i is not None else np.nan for i in geo]
    lon = [i.longitude if i is not None else np.nan for i in geo]
```

```
    county = [i.address.upper().split(", ")[1] if i is not None else np.nan for i in geo]
    fip = [fip_dict.get(i.address.upper().split(", ")[1] + " " +i.address.upper().split(", ")[2]) \
           if i is not None else np.nan
           for i in geo]

    notfoundl = list(notfound)
    lat_add = dict(zip(notfoundl,lat))
    lon_add = dict(zip(notfoundl,lon))
    ct_add = dict(zip(notfoundl,county))
    fip_add = dict(zip(notfoundl,fip))

    ct_dict.update(ct_add)
    fip_dict.update(fip_add)
    lat_dict.update(lat_add)
    lon_dict.update(lon_add)

    proj["COUNTY"] = proj["FULL_LOC"].map(ct_dict)
    proj["FIPS"] = proj["FULL_LOC"].map(fip_dict)
    proj["LNG"] = proj["FULL_LOC"].map(lon_dict)
    proj["LAT"] = proj["FULL_LOC"].map(lat_dict)
    return proj
```

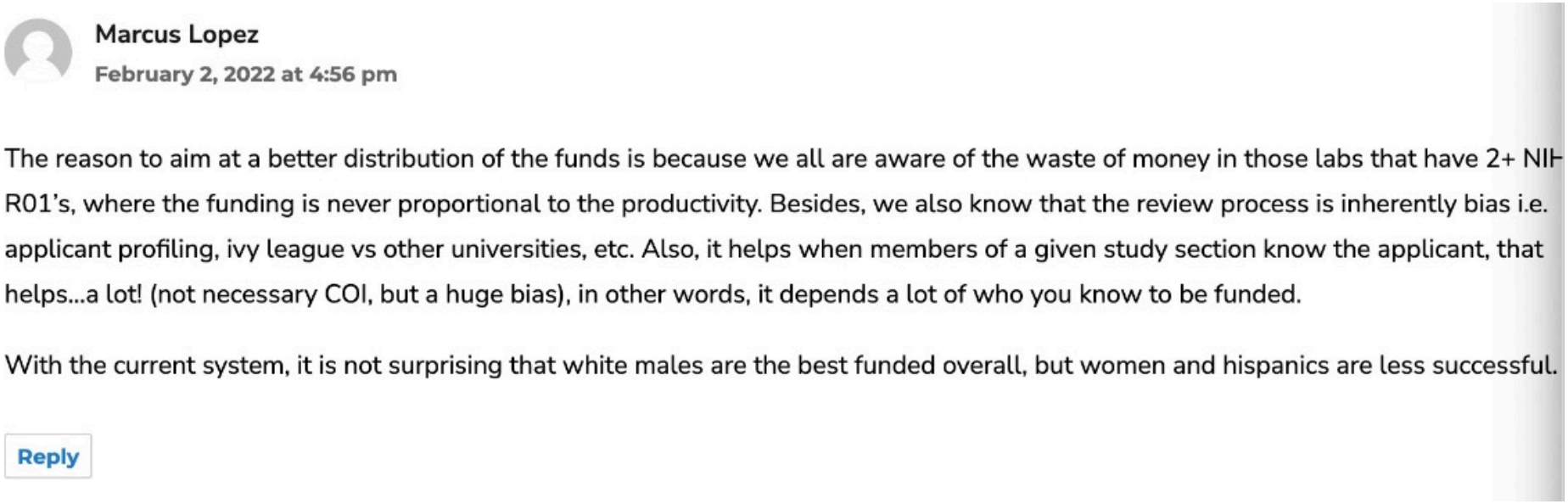
As you would see, the geolocator runs on API request so it is going to take a long time as well.

```
proj = geo_encode(df,fip_dict)
```

```
# Let's resave the data, store data in database data.db table name "decade"
conn = sqlite3.connect("data.db")
cur = conn.cursor()
cur.execute("DROP TABLE IF EXISTS decade")
conn.commit()
proj.to_sql("decade",conn,index=False)
conn.close()
```

Geospatial Analysis

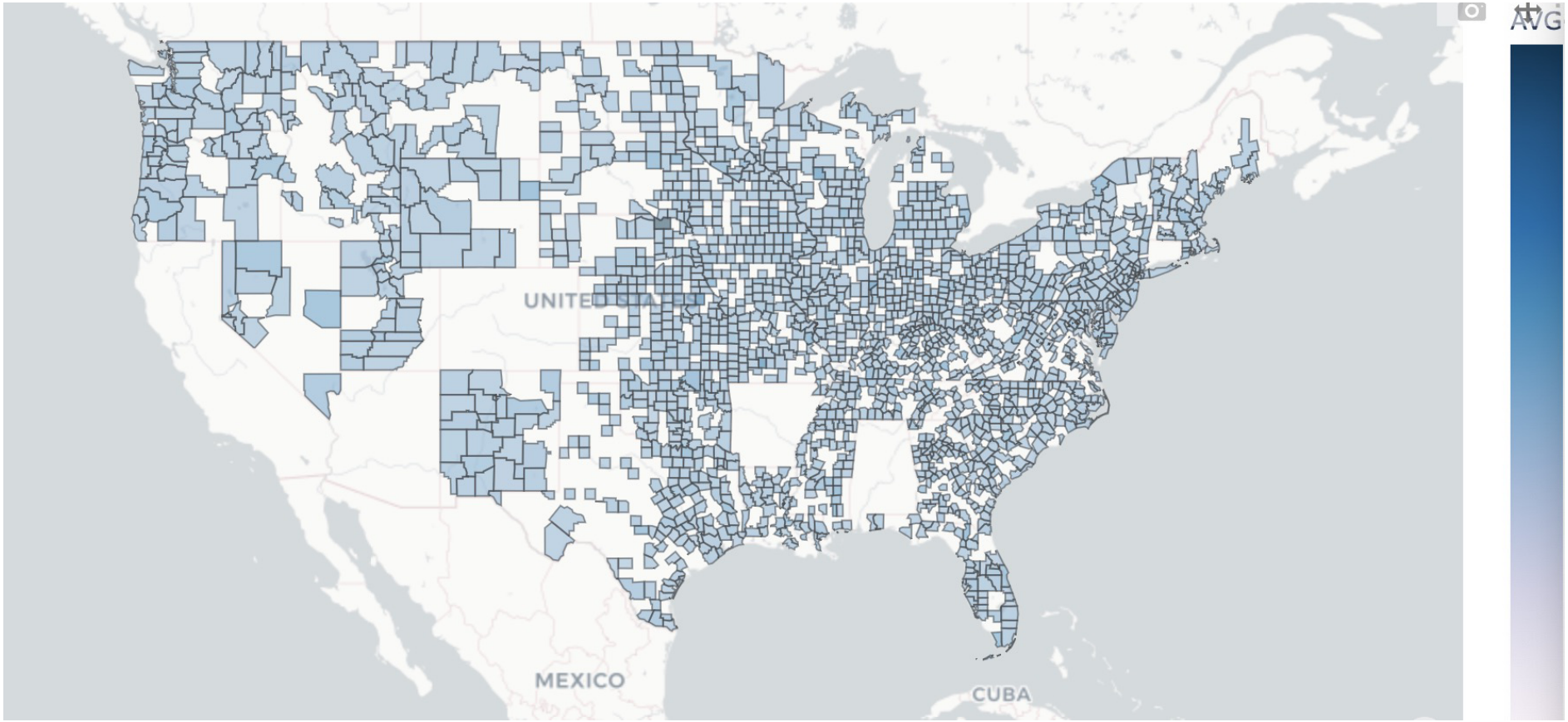
comment excerpt



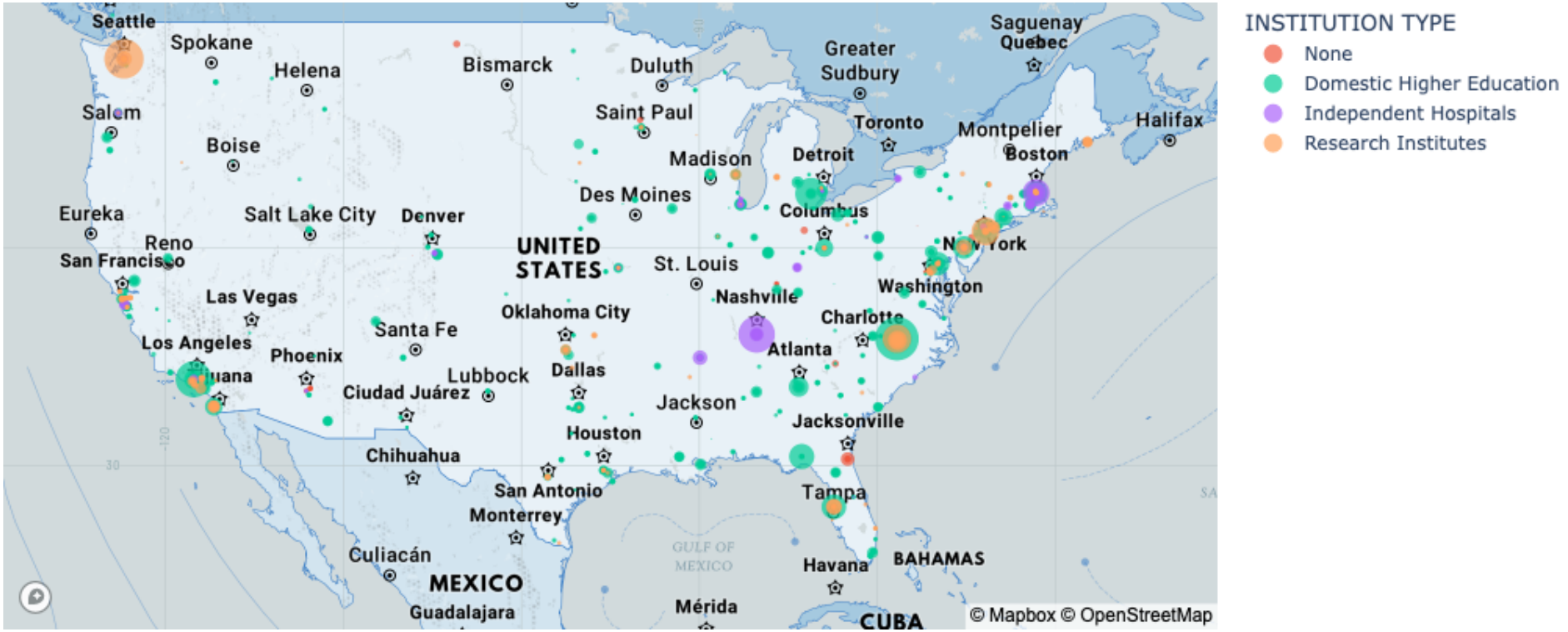
geor.png

The rationale for conducting this analysis is that we want to examine whether there are cliques that consist of institutions locating within a certain geographical region that share resources and 'help' each other out during the grant application/review process. Arguably, visualizing by couty is a btter option, but the by-county map does not look so great, so we opted for the by-state map instead for demo purposes.

(Below are examples of by-county maps, scatter and choropleth.)



county1.png



scatter1.png

Now, let's plot a state choropleth map to visualize the funding distributions. For the sake of time, I will just keep working with the dataframe in the workspace. The choropleth map would be colored by one of the following metrics: - Total fundings/number of organizations - Total fundings/number of projects - Total fundings/number of PI IDs

//another way of doing this is to further normalize, if we were to look at the data irregardless of the funding/institute type, we create some sort of basket norm, like calculating CPI from a counsumer basket

```
proj.columns
```

```
Index(['ORGANIZATION NAME', 'ORGANIZATION ID (IPF)', 'PROJECT NUMBER',
       'FUNDING MECHANISM', 'PI NAME', 'PI PERSON ID', 'PROJECT TITLE',
       'DIRECT COST', 'INDIRECT COST', 'FUNDING', 'CITY',
       'STATE OR COUNTRY NAME', 'INSTITUTION TYPE', 'AWARD NOTICE DATE',
       'MONTH', 'YEAR', 'FULL_LOC', 'CODE', 'COUNTY_FIPS', 'COUNTY_NAME',
```

```
       'LAT', 'LNG', 'COUNTY', 'FIPS'],
      dtype='object')
```

```
# Average funding per organization
fund_org_avg = proj.groupby(
    ['CODE','YEAR','FUNDING MECHANISM','INSTITUTION TYPE','ORGANIZATION NAME']
    )['FUNDING'].agg('mean')
fund_org_avg = fund_org_avg.reset_index(0).reset_index(0).reset_index(0).reset_index(0).reset_index(0)
fund_org_avg = proj.groupby(
    ['CODE','YEAR','FUNDING MECHANISM','INSTITUTION TYPE']
    )['FUNDING'].agg('mean')
fund_org_avg = fund_org_avg.reset_index(0).reset_index(0).reset_index(0).reset_index(0)
#for demo
fund_org_avg.to_csv('fund_org_avg.csv')
fund_org_avg.head()
```

	INSTITUTION TYPE	FUNDING MECHANISM	YEAR	CODE	FUNDING
0	Domestic Higher Education	Other Research-Related	2012.0	AK	2.869343e+05
1	Domestic Higher Education	RPGs - Non SBIR/STTR	2012.0	AK	3.684612e+05
2	None	RPGs - Non SBIR/STTR	2012.0	AK	3.097755e+05
3	Domestic Higher Education	Research Centers	2012.0	AK	2.311864e+06
4	Domestic Higher Education	Other Research-Related	2013.0	AK	2.007843e+05

```
# Average funding per pi ~ to a team/lab
fund_pi_avg = proj.groupby(
    ['CODE','YEAR','FUNDING MECHANISM','INSTITUTION TYPE','PI PERSON ID']
    )['FUNDING'].agg('mean')
fund_pi_avg = fund_pi_avg.reset_index(0).reset_index(0).reset_index(0).reset_index(0).reset_index(0)
fund_pi_avg = proj.groupby(
    ['CODE','YEAR','FUNDING MECHANISM','INSTITUTION TYPE']
    )['FUNDING'].agg('mean')
fund_pi_avg = fund_pi_avg.reset_index(0).reset_index(0).reset_index(0).reset_index(0)
```

```
fund_pi_avg.to_csv('fund_pi_avg.csv')
fund_pi_avg.head()
```

	INSTITUTION TYPE	FUNDING MECHANISM	YEAR	CODE	FUNDING
0	Domestic Higher Education	Other Research-Related	2012.0	AK	2.869343e+05
1	Domestic Higher Education	RPGs - Non SBIR/STTR	2012.0	AK	3.684612e+05
2	None	RPGs - Non SBIR/STTR	2012.0	AK	3.097755e+05
3	Domestic Higher Education	Research Centers	2012.0	AK	2.311864e+06
4	Domestic Higher Education	Other Research-Related	2013.0	AK	2.007843e+05

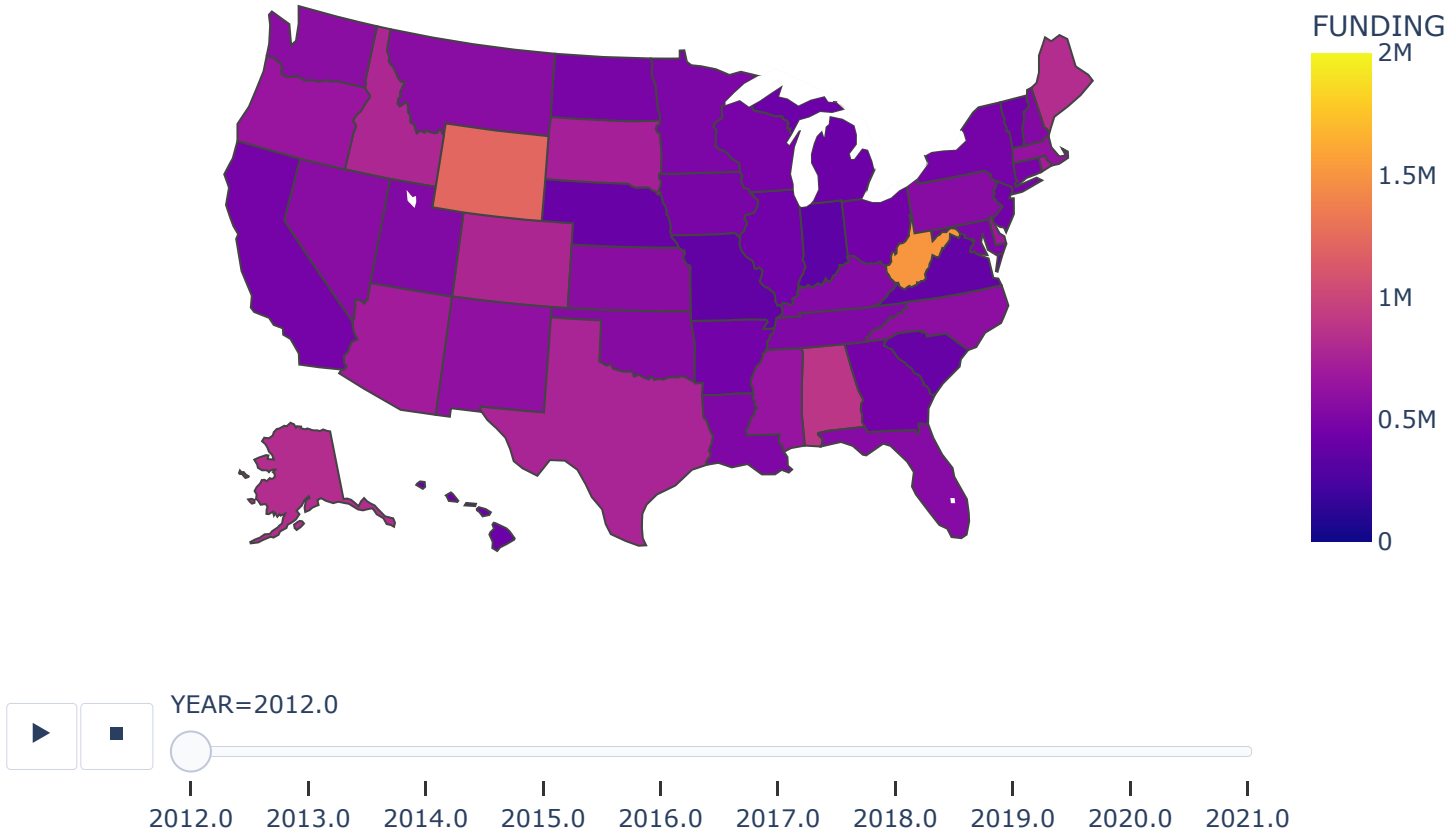
```
# average funding per project
fund_proj_avg = proj.groupby(
    ['CODE','YEAR','FUNDING MECHANISM','INSTITUTION TYPE']
    )['FUNDING'].agg('mean')
fund_proj_avg = fund_proj_avg.reset_index(0).reset_index(0).reset_index(0).reset_index(0)

fund_proj_avg.to_csv('fund_proj_avg.csv')
fund_proj_avg.head()
```

	INSTITUTION TYPE	FUNDING MECHANISM	YEAR	CODE	FUNDING
0	Domestic Higher Education	Other Research-Related	2012.0	AK	2.869343e+05
1	Domestic Higher Education	RPGs - Non SBIR/STTR	2012.0	AK	3.684612e+05
2	None	RPGs - Non SBIR/STTR	2012.0	AK	3.097755e+05
3	Domestic Higher Education	Research Centers	2012.0	AK	2.311864e+06
4	Domestic Higher Education	Other Research-Related	2013.0	AK	2.007843e+05

```
fund_proj_avg = pd.read_csv(path+'fund_proj_avg.csv')
```

The following is an example animated graph of the average funding per project by state:

```
by_year = fund_proj_avg.groupby(
    ['CODE','YEAR']
    )['FUNDING'].agg('mean').reset_index(0).reset_index(0)
fig = px.choropleth(by_year,locations="CODE", animation_frame = 'YEAR',locationmode="USA-states", color="FUNDING", scope
fig.show()
```



Research Trend / Demand

In this part, we would be examining a funding history of different categories generated by RCDC —— computerized process the NIH uses to categorize and report the amount it funded in each of the more than 280 reported categories of disease, condition, or research area.

The table contains estaimted fundings for 2022,2023, historical fundings from 2008 to 2021, and the disease prevalence/death counts totaled in 2019. In 2008, 2009, extra funding and grants were allocated to NIH projects as a result of the American Recovery and Reinvestment Act, a.k.a. Obama Stimulus. We have combined them with the 2008 and 2009 funding columns.

```
rcdc = pd.read_excel(path+"RCDCFundingSummary.xlsx",)
rcdc.columns = rcdc.loc[1,:]
rcdc = rcdc.loc[2:310,:]
```

The following function is written to clean the RCDC summary data.

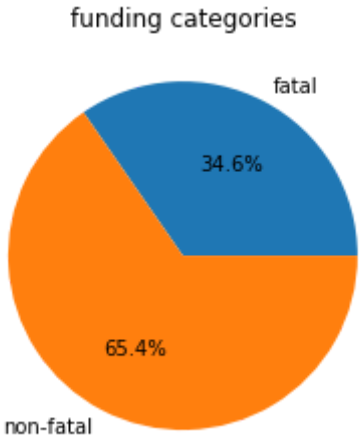
```
def RCDC_clean(rcdc):
    selected = rcdc.copy()
    #- indicates no death from this category
    selected['2019 US Mortality 19'] \
        = selected['2019 US Mortality 19'].apply(lambda x:str(x).replace('-','0'))
    #+ indicates the category did not exist in that specific year.
    # encode to a numeric indicator
    selected = selected.applymap(lambda x:str(x).replace('+','-1'))
    #get rid of the dollar signs so as to convert to numeric
    selected = selected.applymap(lambda x:str(x).replace('$',''))

    #combining ARRA
    selected[2009]=selected[2009].astype(float) + selected['2009 ARRA'].astype(float)
    selected[2010]=selected[2010].astype(float) + selected['2010 ARRA'].astype(float)
    selected.drop(['2009 ARRA','2010 ARRA'],axis=1,inplace=True)

    #number of years the category has existed
    selected['exist'] = selected.apply((lambda row: sum([1 if (row[x] != '-1') else 0 for x in list(selected.columns)[1:
    #average fundings over the year
    selected['avg'] = selected.apply((lambda row: np.mean([int(row[x]) if (row[x] != '-1') else 0 for x in list(selected
    #estimated increase from 2022 to 2023
    selected['estimate'] = selected['2023 Estimated'].astype(float) - selected['2022 Estimated'].astype(float)
    #type conversion
    selected['2019 US Mortality 19'] = selected['2019 US Mortality 19'].astype('float')
```

```
    #percentage of increase
    selected["pct2023"] = selected.loc[:,['2022 Estimated','2023 Estimated']].astype(float).pct_change(axis=1)['2023 Est
    #sorting
    selected =selected.sort_values('2019 US Mortality 19')
    print(selected.head(n=1))
    #We divide the categories into lethal conditions, and non lethal
    fatal = selected[selected['2019 US Mortality 19'] > 0]
    nonfatal = selected[selected['2019 US Mortality 19'] == 0]
    return fatal, nonfatal
```

```
fatal, nonfatal = RCDC_clean(rcdc)
```

```
1 Research/Disease Areas \n (Dollars in millions and rounded) 2008  2009  \
2                     Acquired Cognitive Impairment             -1  -2.0

1  2010 2011 2012 2013 2014 2015  2016  ...  2020  2021 2022 Estimated  \
2  -2.0   -1   -1   -1   -1  798  1132  ...  2897  3259           3555

1 2023 Estimated 2019 US Mortality 19 2019 US Prevalence SE 19 exist     avg  \
2           3354                  0.0                         -    10  1309.5

1 estimate   pct2023
2   -201.0 -5.654008

[1 rows x 23 columns]
```

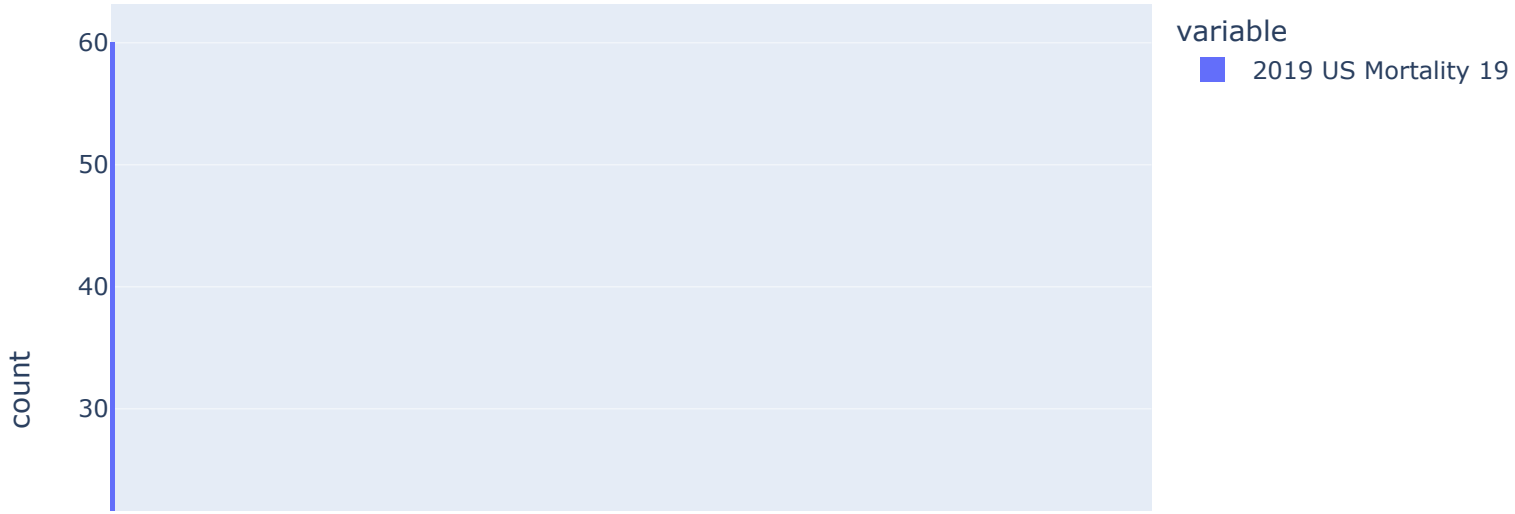
```
plt.pie([len(fatal),len(nonfatal)],labels=['fatal','non-fatal'],autopct='%1.1f%%')
plt.title('funding categories')
plt.show()
```

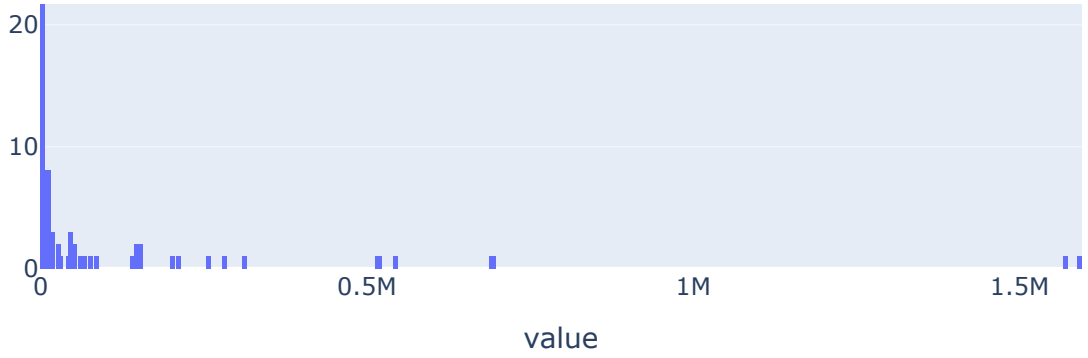


Our first attempt in investigating the data is to figure out whether there is any association among fatality/prevalence, category existed time, average fundings in the past, and the estimated increase in funding. Since quite a lot of variables are involved, without doing any regression analysis, an (perhaps) intuitive way to show the association is through the parallele category plots.

But before that, we should divide the funding and mortality counts into different bins based on their distributions.

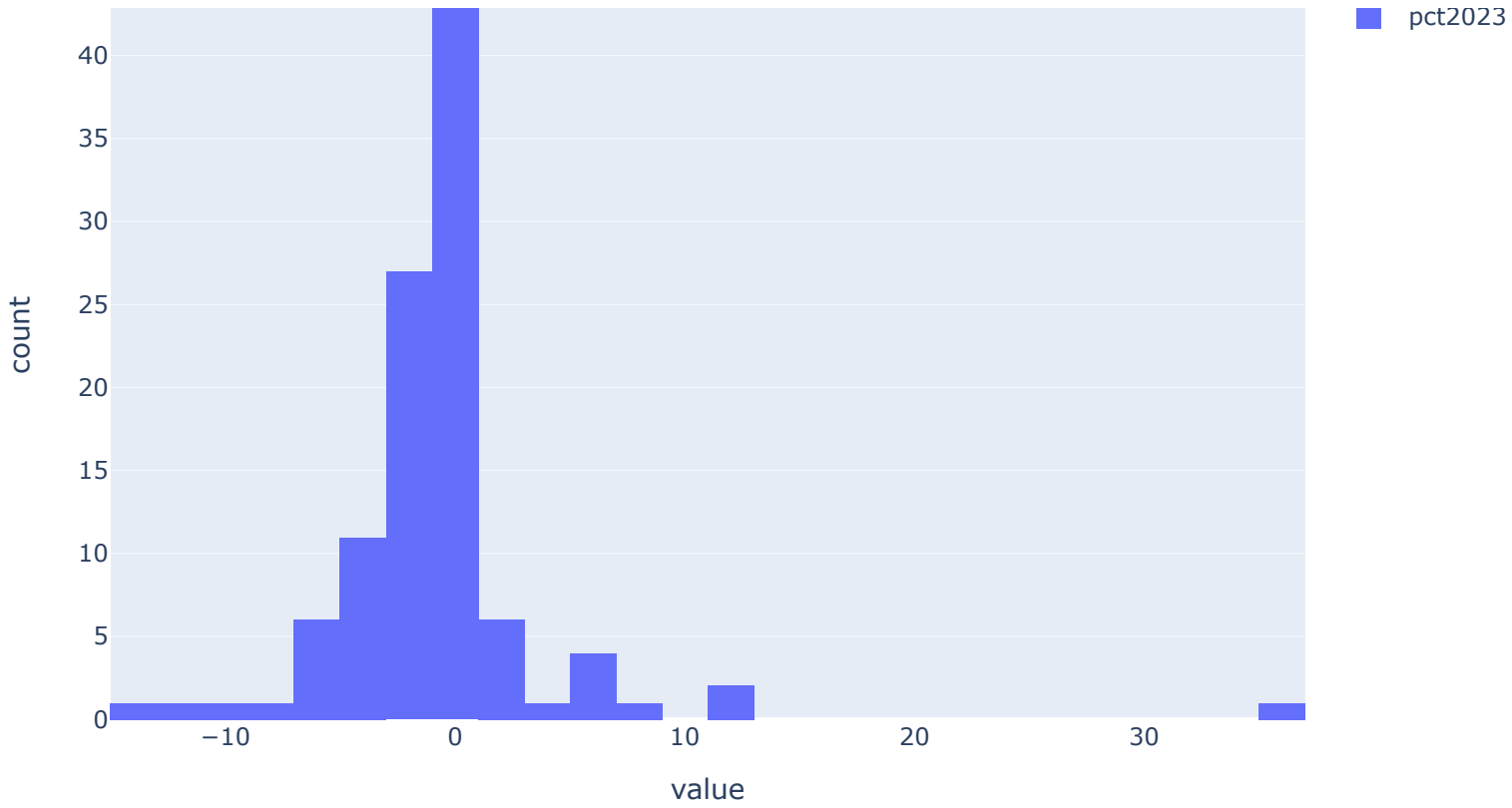
```
px.histogram(fatal['2019 US Mortality 19'],nbins=500)
```






```
mort_bins = pd.IntervalIndex.from_tuples([(0, 50*10e2),(50*10e2, 99*10e2), (99*10e2, 100*10e2),
                                          (100*10e2, 150*10e2),
                                          (150*10e2, 200*10e2),
                                          (200*10e2, 250*10e2),
                                          (250*10e2, 300*10e2),
                                          (300*10e2, 350*10e2),
                                          (350*10e2, 400*10e2),
                                          (400*10e2, 1*10e6),
                                          (10e6, 2*10e6)])
mort_dict = dict(zip([str(i) for i in list(mort_bins)],['<5k','5-10k','10-100k',
                                    '100-150k',
                                    '150-200k',
                                    '200-250k',
                                    '250-300k',
                                    '300-350k',
                                    '350-400k',
                                    '400k - 1M',
                                    '> 1M']))
```

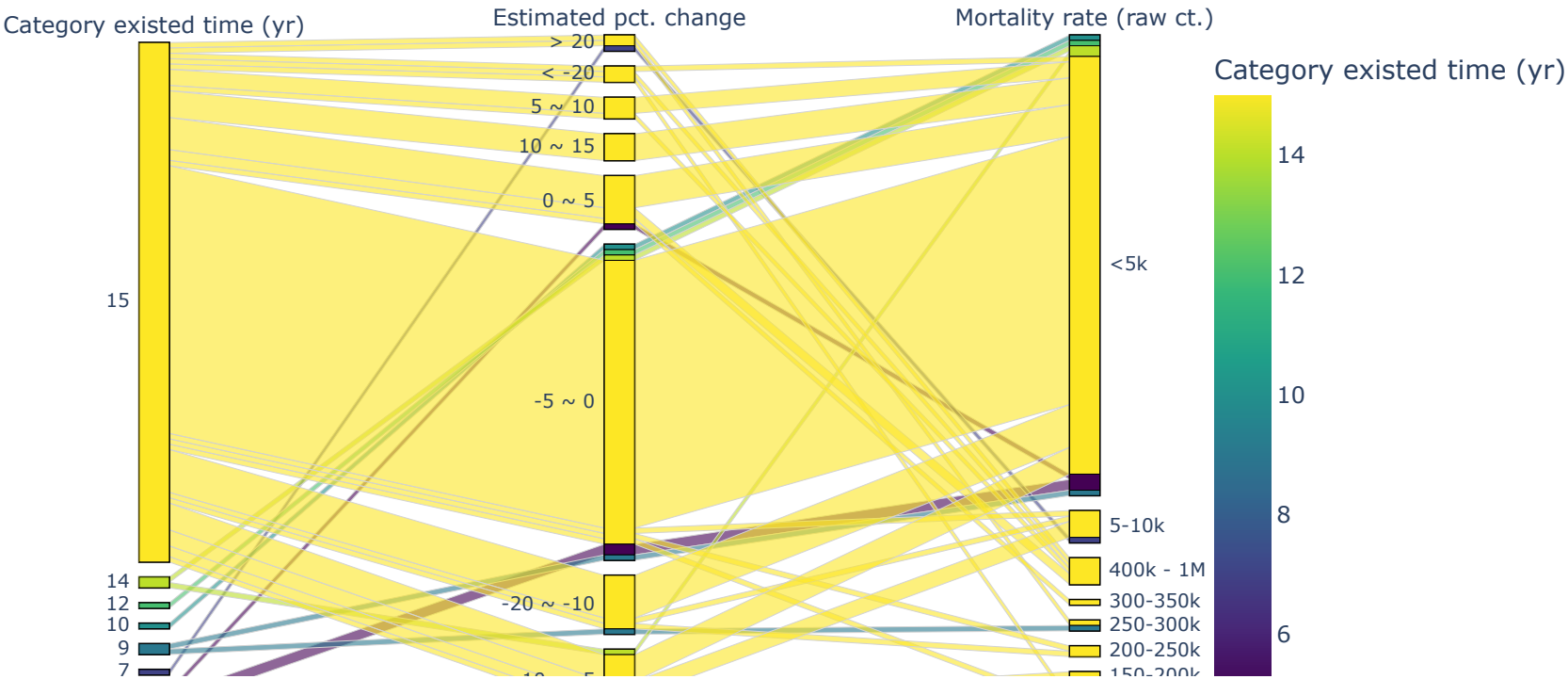
```
px.histogram(fatal['pct2023'])
```




```
pct_bins = pd.IntervalIndex.from_tuples([(-300,-20),(-20, -10),(-10,-5), (-5,0),
                                         (0,5),
                                         (5,10),
                                         (10,15),
                                         (15,20),
                                         (20,300)])
pct_dict = dict(zip([str(i) for i in list(pct_bins)],['< -20','-20 ~ -10','-10 ~ -5', '-5 ~ 0',
                                                      '0 ~ 5',
                                                      '5 ~ 10',
                                                      '10 ~ 15',
                                                      '15 ~ 20',
                                                      '> 20']))
fatal['est_bin'] = pd.cut(fatal['estimate'], pct_bins).astype(str)
fatal['mort_bin'] = pd.cut(fatal['2019 US Mortality 19'], mort_bins).astype(str)
```

```
fatal['mort_str'] = fatal['mort_bin'].astype(str).map(mort_dict)
fatal['est_str'] = fatal['est_bin'].astype(str).map(pct_dict)
```

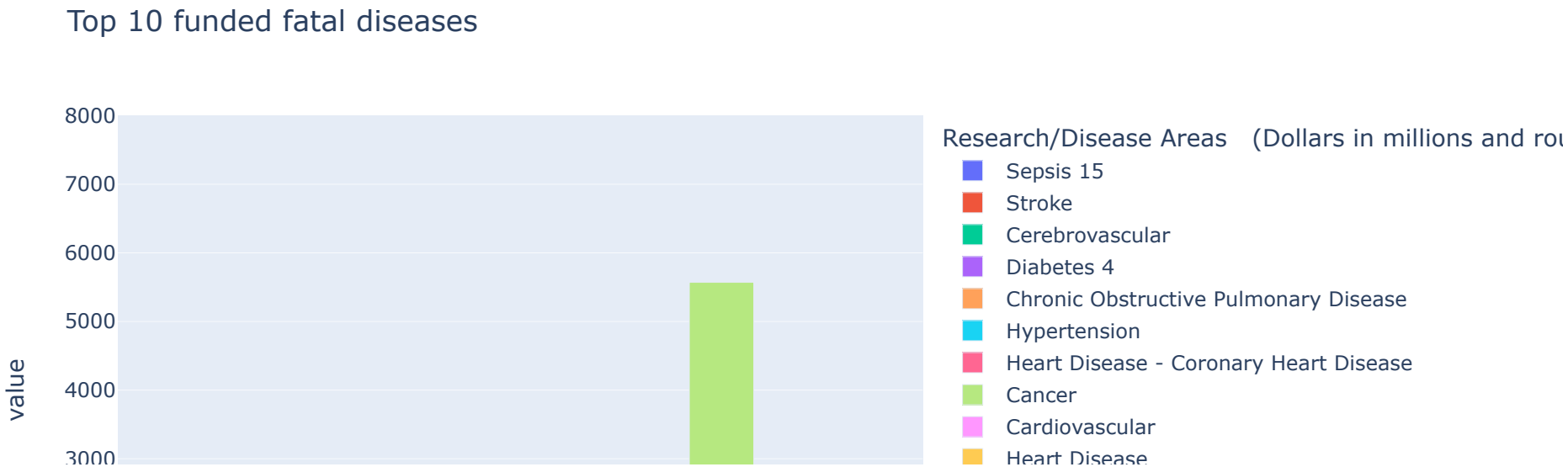
```
#df = px.data.tips()
fig = px.parallel_categories(fatal, dimensions=['exist','est_str','mort_str'],
                color="exist", color_continuous_scale="Viridis",
                labels={'exist':'Category existed time (yr)',
                        'est_str':'Estimated pct. change',
                        'mort_str':'Mortality rate (raw ct.)'},
                height = 500,
                width = 900
                )
dimensions=['exist','est_str','mort_bin']
fig.update_traces(dimensions=[{"categoryorder": "category descending"} for _ in dimensions])
plt.tight_layout()
fig.show()
```

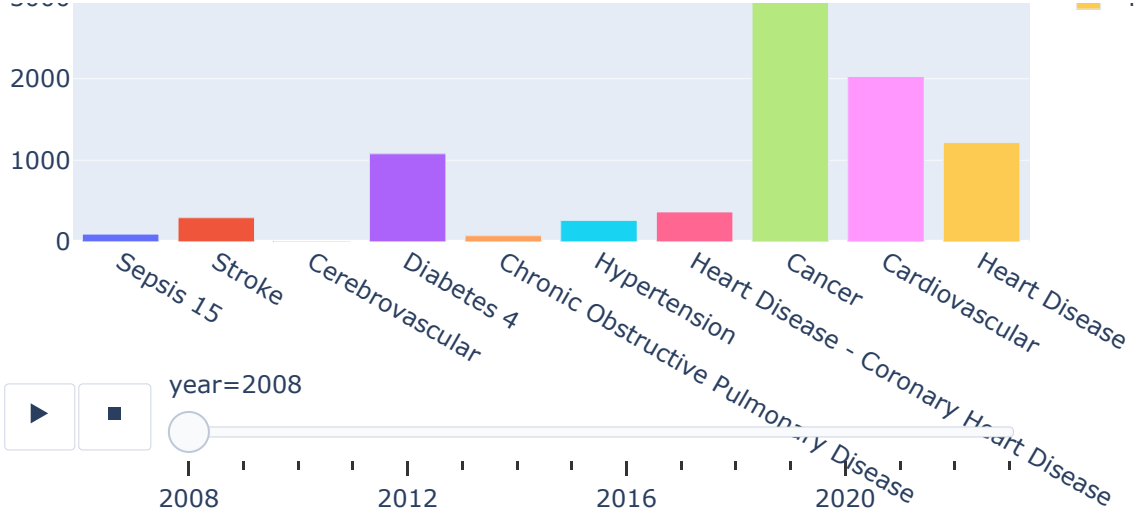



```
<Figure size 432x288 with 0 Axes>
```

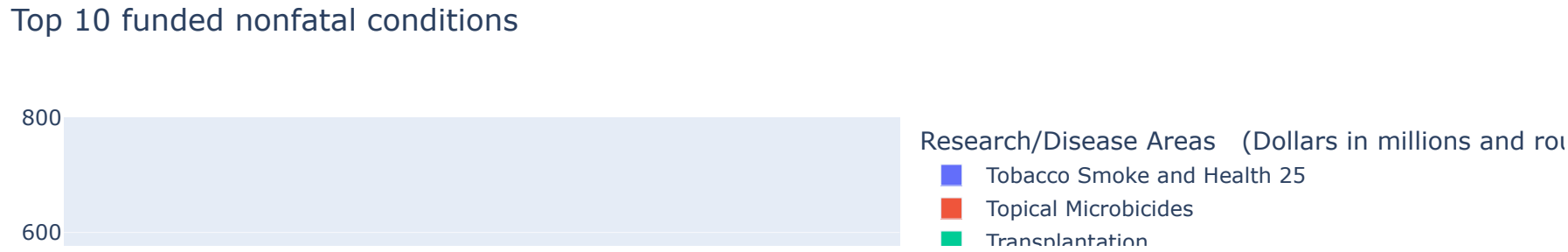
Let's also take a look at how the top-funded categories have changed in the past decade.

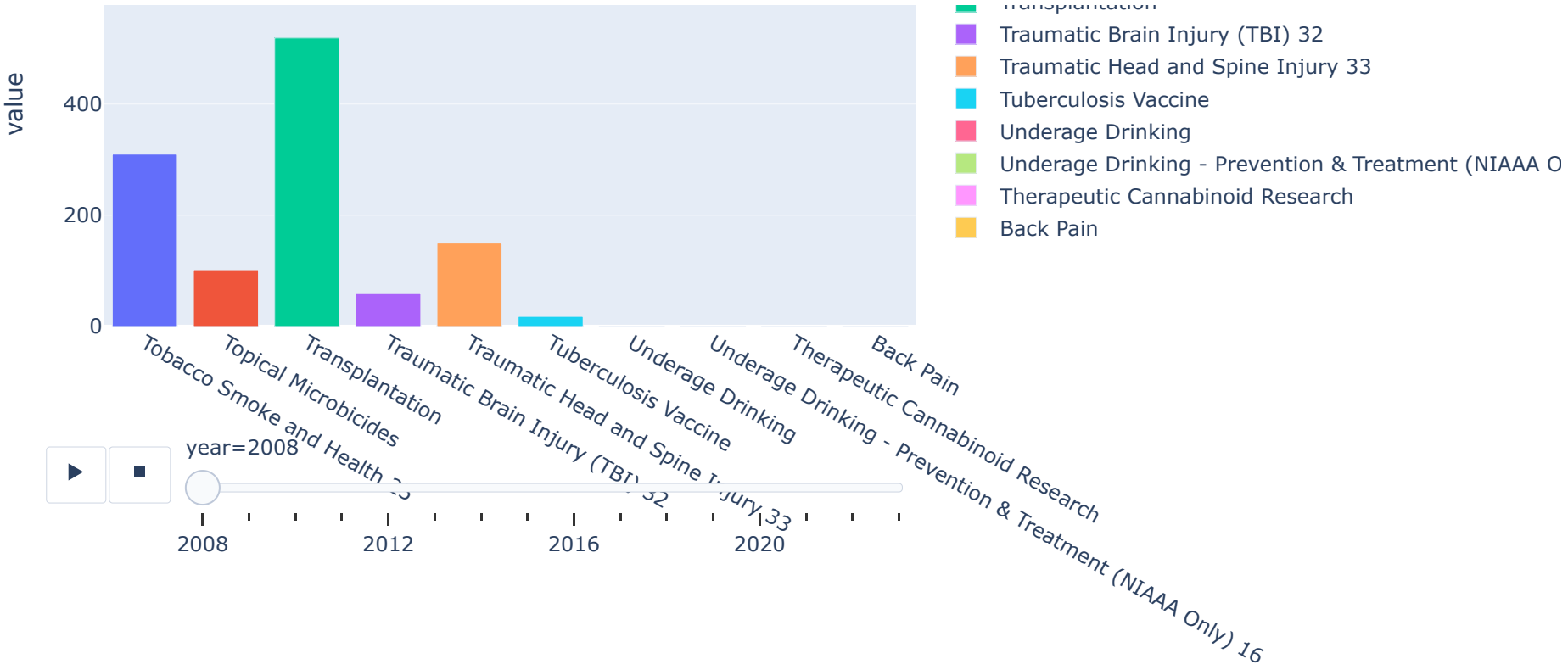
```
fatal10 = fatal.sort_values('2019 US Mortality 19').iloc[-10:,]
sem = pd.melt(fatal10, id_vars=['Research/Disease Areas \n (Dollars in millions and rounded)'],
        value_vars=fatal.columns[1:17],
        var_name='year', value_name='value')
sem = sem.rename(columns={'2022 Estimated': '2022','2023 Estimated': '2023'})
sem['value'] = sem['value'].astype(int)
fig = px.bar(sem, x='Research/Disease Areas \n (Dollars in millions and rounded)',
             y="value", color='Research/Disease Areas \n (Dollars in millions and rounded)',
             title='Top 10 funded fatal diseases',
  animation_frame="year",width = 1000,height = 600,range_y=[0,8000])

fig.show()
```






```
nfatal10 = nonfatal.sort_values('2019 US Mortality 19').iloc[-10:,]
sem = pd.melt(nfatal10, id_vars=['Research/Disease Areas \n (Dollars in millions and rounded)'],
        value_vars=fatal.columns[1:17],
        var_name='year', value_name='value')
sem = sem.rename(columns={'2022 Estimated': '2022','2023 Estimated': '2023'})
sem['value'] = sem['value'].astype(int)
fig = px.bar(sem, x='Research/Disease Areas \n (Dollars in millions and rounded)',
             y="value", color='Research/Disease Areas \n (Dollars in millions and rounded)',
             title='Top 10 funded nonfatal conditions',
  animation_frame="year",width = 1000,height = 600,range_y=[0,800])

fig.show()
```



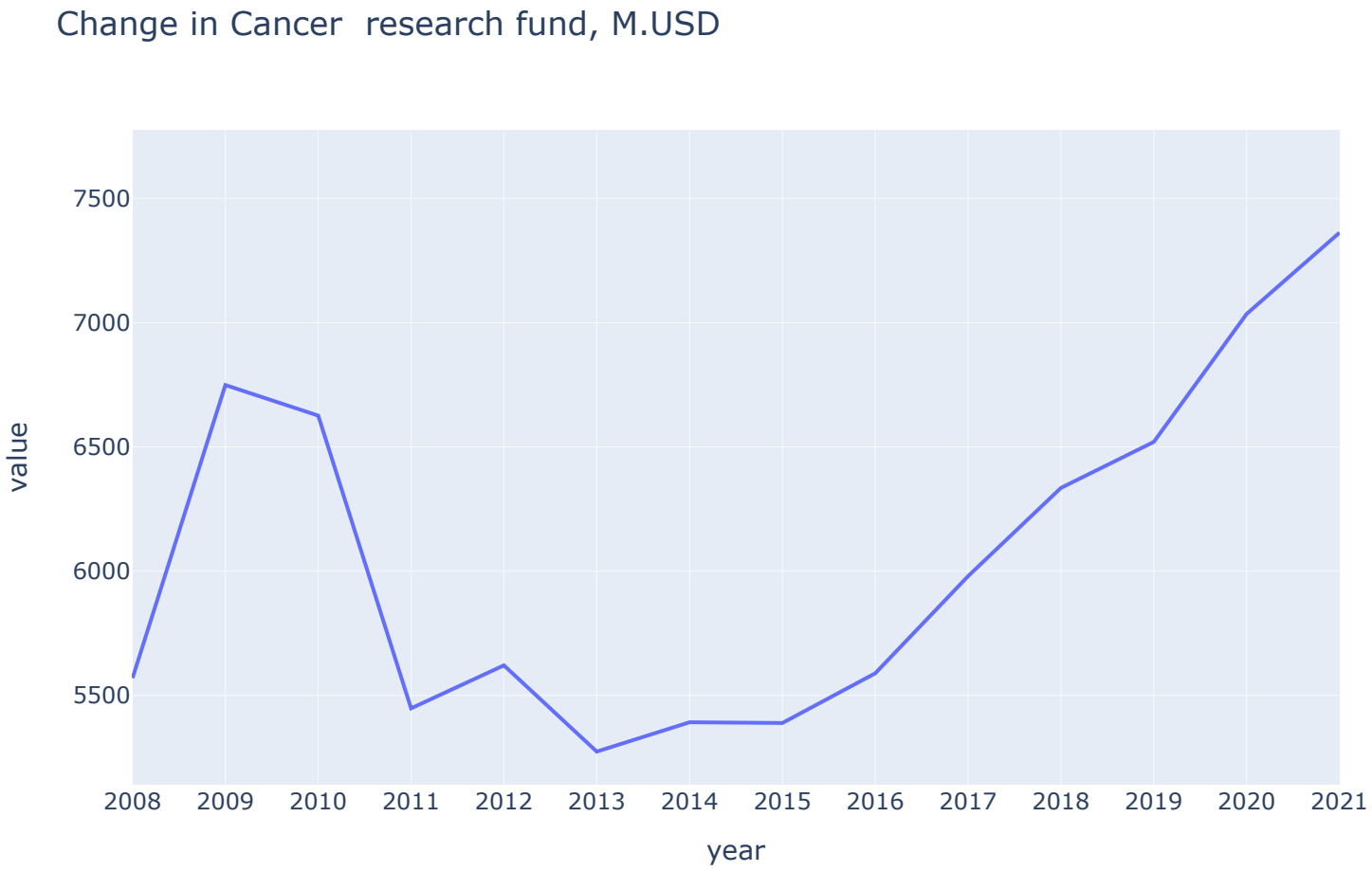


Research/Disease Areas (Dollars in millions and rounded)

We would be adding more [interactive visualizations[(https://dash.plotly.com/interactive-graphing)] for this part later in the dash app. Mainly, we would be displaying the time-series data for each category based on user selection. Once user hovers over a category, we would display the change in the funding for that category over the past decade on the side:

```
def draw_line(df, selected):
    area = "Research/Disease Areas \n (Dollars in millions and rounded)"
    sub = df[df[area]== selected]
    sub.rename(columns={'2022 Estimated': '2022','2023 Estimated':'2023'})
    melted = pd.melt(sub, id_vars=[area],
            value_vars=fatal.columns[1:17],
            var_name='year', value_name='value')
    melted['value'] = melted['value'].astype(int)

    fig = px.line(melted,x='year',y='value',title="Change in "+ selected+'\n'+"research fund, M.USD")
```

```
    fig.update_layout(
        xaxis = dict(
            tickmode = 'linear',
            tick0 = 1,
            dtick = 1
        )
    )
    return fig
draw_line(fatal,'Cancer ')
```



Project Efficiency (written by Haofei)

There are a lot of demands on analysis of funding efficiency (statistical tests, new indices) to stop waste and encourage fairness.In this part, we need to analyze the funding efficieny between different organizations, and use some coefficients and graphs to measure their inequality level.

To show and measure the efficiency, we divide the analysis into two parts: Regression analysis and Coefficients Visualization. - Regression analysis among variables measruing - PRODUCTIVITY - (Publication / Patent per Project) - (Publication / Patent per Organization

- COST - (Variable Cost: Total Funding)

- Visualization with - GINI COEFFICIENTS - LORENZ CURVE - THEIL INDEX

Part 1: Regression Analysis Among Variables

To prepare for the regression anslysis,we need to processing data in tables first. Here we need to process and merge four tables： - annual project - publication - patent - publink

The data range we used is from 2000 to 2022.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from plotly import express as px
import os
import glob
import sqlite3
import numpy as np
```

```
#read data from csv and excel file
def read_csv_file(path):
    data = pd.DataFrame()
    csv_list = os.listdir(path)
    csv = glob.glob(os.path.join(path, "*.csv"))
```

```
    for x in csv:
        if '.DS_Store' not in x:
            print(x)
            df = pd.read_csv(x,encoding='latin-1')
            df.columns = [s.upper() for s in list(df.columns.str.upper())]
        if len(data) == 0:
            data = df
        else:
            data = pd.concat([data, df],
                             ignore_index = False,
                             sort = False)
    return data

def read_excel_file(path):
    data = pd.DataFrame()
    xls_list = os.listdir(path)
    xlsx = glob.glob(os.path.join(path, "*.xls"))
    for x in xlsx:
        if '.DS_Store' not in x:
            print(x)
            df = pd.read_excel(x)
            df.columns = [s.upper() for s in list(df.columns.str.upper())]
        if len(data) == 0:
            data = df
        else:
            data = pd.concat([data, df],
                             ignore_index = False,
                             sort = False)
    return data
```

```
# paths for four tables
project_link='/Users/a10033/Desktop/pic16B project/annual 2'
publication_link="/Users/a10033/Desktop/pic16B project/PUB_C"
publink_link="/Users/a10033/Desktop/pic16B project/PUBLNK"
patent_link="/Users/a10033/Desktop/pic16B project/Patents.csv"
```

```
data=read_excel_file(project_link)
pub=read_csv_file(publication_link)
publink=read_csv_file(publink_link)
pat=pd.read_csv(patent_link)
```

```
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2018.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2019.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2009.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2021.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2020.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2008.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2022.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2006.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2012.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2013.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2007.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2011.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2005.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2004.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2010.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2014.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2000.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2001.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2015.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2003.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2017.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2016.xls
/Users/a10033/Desktop/pic16B project/annual 2/Worldwide2002.xls
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2013.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2007.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2006.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2012.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2004.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2010.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2011.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2005.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2001.csv
```

```
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2015.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2014.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2000.csv
/Users/a10033/Desktop/pic16B project/PUB_C/REPORTER_PUB_C_2016.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2002.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2003.csv
/Users/a10033/Desktop/pic16B project/PUB_C/REPORTER_PUB_C_2017.csv
/Users/a10033/Desktop/pic16B project/PUB_C/REPORTER_PUB_C_2019.csv
/Users/a10033/Desktop/pic16B project/PUB_C/REPORTER_PUB_C_2018.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2008.csv
/Users/a10033/Desktop/pic16B project/PUB_C/REPORTER_PUB_C_2020.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2021.csv
/Users/a10033/Desktop/pic16B project/PUB_C/RePORTER_PUB_C_2009.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/REPORTER_PUBLNK_C_2016.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2002.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2003.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/REPORTER_PUBLNK_C_2017.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2001.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2015.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2014.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2000.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2004.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2010.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2011.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2005.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2013.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2007.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2006.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2012.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2008.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/REPORTER_PUBLNK_C_2020.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2021.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/RePORTER_PUBLNK_C_2009.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/REPORTER_PUBLNK_C_2019.csv
/Users/a10033/Desktop/pic16B project/PUBLNK/REPORTER_PUBLNK_C_2018.csv
```

```
#set the index to be unqiue
data=data.reset_index().drop(columns="index")#set index to be unique
```

```
pub=pub.reset_index().drop(columns="index")
publink=publink.reset_index().drop(columns="index")
pat=pat.reset_index().drop(columns="index")
```

Funding and Project

To get total funding and project numbers for each organization,we need to choose columns:“ORGANIZATION NAME”,“PROJECT NUMBER” and “FUNDING”.By observing the original data table, we can find that the percentage of NAN values is close to 50%, so we need to delete the rows containing NAN values， because we need to count the fundings,and project number.

```
data=data.loc[:,["ORGANIZATION NAME","PROJECT NUMBER","FUNDING"]]
data=data.dropna(axis=0,subset="FUNDING")
data=data.dropna(axis=0,how="any",subset="ORGANIZATION NAME")
data=data.reset_index().drop(columns="index")#set index to be unique
data.isnull().any() #check if null in each column
```

```
ORGANIZATION NAME    False
PROJECT NUMBER       False
FUNDING              False
dtype: bool
```

Next, we will use groupby and transform to count how many projects each organization has and the total number of fundings

```
data["TOTAL FUNDING"]=data.groupby(['ORGANIZATION NAME'])['FUNDING'].transform('sum')
data["TOTAL PROJECT"]=data.groupby(['ORGANIZATION NAME'])['PROJECT NUMBER'].transform("count")
#drop duplicated rows
data1=data.drop_duplicates(subset="ORGANIZATION NAME",keep="first")
#drop unnecessary columns
data1=data1.drop(columns=["PROJECT NUMBER","FUNDING"])
```

Table of Total Funding & Total Project

```
data1
```

	ORGANIZATION NAME	TOTAL FUNDING	TOTAL PROJECT
0	21ST CENTURY MEDICINE, INC.	2242103.0	5
2	21ST CENTURY THERAPEUTICS, INC.	2147493.0	7
3	3-C INSTITUTE FOR SOCIAL DEVELOPMENT	20665498.0	50
8	3DT HOLDINGS, LLC	14089745.0	26
12	3P BIOTECHNOLOGIES, INC.	2507358.0	6
...	...	...	...
622729	WI-SENSE, LLC	190000.0	2
622815	WOODS & POOLE ECONOMICS I	16500.0	1
622827	WORLD 2 SYSTEMS	99926.0	1
622853	XANDEM TECHNOLOGY, LLC	225000.0	1
623818	ZITEO, INC.	224922.0	1

7965 rows × 3 columns

Patent

We need to merge the tables of patent and project by organization name, so we need to delete the rows where organization name is NAN

```
pat=pat.dropna(axis=0,how="any",subset=["PROJECT_ID","PATENT_ORG_NAME"])
pat.rename(columns={"PATENT_ORG_NAME":"ORGANIZATION NAME"},inplace=True)
pat=pat.reset_index().drop(columns="index")
pat["TOTAL PATENT"]=pat.groupby('ORGANIZATION NAME')["PROJECT_ID"].transform("count")
pat=pat.loc[:,["ORGANIZATION NAME","TOTAL PATENT"]]
pat1=pat.drop_duplicates(subset="ORGANIZATION NAME",keep="first")
pat1=pat1.reset_index().drop(columns="index")
```

Table of Patent

```
pat1
```

	ORGANIZATION NAME	TOTAL PATENT
0	BAYLOR RESEARCH INSTITUTE	28
1	OREGON HEALTH & SCIENCE UNIVERSITY	604
2	PENNSYLVANIA STATE UNIVERSITY	338
3	TEXAS BIOMEDICAL RESEARCH INSTITUTE	18
4	THORATEC LABORATORIES CORPORATION	5
...	...	...
1026	ERGON PHARMACEUTICALS, LLC	1
1027	NOMADICS, INC.	2
1028	PRIMITY, INC.	1
1029	TRIA BIOSCIENCE CORPORATION	1
1030	RESOURCE TECHNOLOGIES GROUP, INC.	1

1031 rows × 2 columns

Publication

In order to count the number of publications per organization, we need to use the data of projects, publication links and publications. Each PROJECT_NUMBER in Publink is part of the PROJECT NUMBER in data table. The PMID in Publink also realted to PROJECT NUMBER in data, and the PMID in Publink can also correspond to the PMID in Pub. However, we will find that one PROJECT_NUMBER corresponds to multiple PROJECT NUMBERs, and one PMID may also correspond to multiple PROJECT NUMBERs.

```
publink.rename(columns={"PROJECT_NUMBER":"PROJECT NUMBER"},inplace=True)
#choose part of project number
data["PROJECT NUMBER"]=data["PROJECT NUMBER"].map(lambda x:str(x)[1:])
data['PROJECT NUMBER']=data['PROJECT NUMBER'].apply(lambda x: str(x).split('-')[0])
```

```
data=data.drop_duplicates(subset="PROJECT NUMBER",keep="first")
data=data.reset_index().drop(columns="index")#set index to be unique
```

```
#merge data and publink
joined_1=pd.merge(data,publink,on="PROJECT NUMBER",how="outer")
joined_1=joined_1.dropna(axis=0,subset="ORGANIZATION NAME")
joined_1=joined_1.drop(columns=["FUNDING","TOTAL FUNDING","TOTAL PROJECT"])
```

```
#merge pub & data & publink
pub=pub.loc[:,["PMID"]]
joined_2=pd.merge(joined_1,pub,on="PMID",how="outer")
joined_2=joined_2.dropna(axis=0,subset="ORGANIZATION NAME")
joined_2["TOTAL PUBLICATON"]=joined_2.groupby(['ORGANIZATION NAME'])['PMID'].transform("count")
joined_2=joined_2.drop(columns=["PROJECT NUMBER","PMID"])
joined_2=joined_2.drop_duplicates(subset="ORGANIZATION NAME",keep="first")
joined_2=joined_2.reset_index().drop(columns="index")
```

Table of Publication

```
joined_2
```

	ORGANIZATION NAME	TOTAL PUBLICATON
0	21ST CENTURY MEDICINE, INC.	0
1	21ST CENTURY THERAPEUTICS, INC.	9
2	3-C INSTITUTE FOR SOCIAL DEVELOPMENT	8
3	3DT HOLDINGS, LLC	8
4	3SCAN, INC.	0
...	...	...
7799	O2 REGENTECH, LLC	1
7800	SPEAK MODALITIES, LLC	1

	ORGANIZATION NAME	TOTAL PUBLICATON
7801	STONESTABLE, INC.	1
7802	SYMBIOS TECHNOLOGIES, INC	2
7803	UNIVERSITY OF WISCONSIN PARKSIDE	1

7804 rows × 2 columns

Combine All Tables

```
table1=pd.merge(data1,pat1,how="outer",on="ORGANIZATION NAME")
table1=table1.dropna(axis=0,subset="TOTAL FUNDING")
table1["TOTAL PATENT"]=table1["TOTAL PATENT"].fillna(0)
table2=pd.merge(table1,joined_2,on="ORGANIZATION NAME",how="outer")
table2["TOTAL PUBLICATON"]=table2["TOTAL PUBLICATON"].fillna(0)
table2=table2.sort_values(by="TOTAL FUNDING")
table2=table2.reset_index().drop(columns="index")
table2['PATENTE & PUBLICATION'] = table2.apply(lambda x: x['TOTAL PATENT'] + x['TOTAL PUBLICATON'], axis=1)
table2.isnull().any()# check null
```

```
ORGANIZATION NAME        False
TOTAL FUNDING            False
TOTAL PROJECT            False
TOTAL PATENT             False
TOTAL PUBLICATON         False
PATENTE & PUBLICATION    False
dtype: bool
```

Here, we will get all the data we need to make regression analysis

```
table2
```

	ORGANIZATION NAME	TOTAL FUNDING	TOTAL PROJECT	TOTAL PATENT	TOTAL PUBLICATON	PATENTE & PUBLICATION
0	ICHOR MEDICAL SYSTEMS, INC.	1.000000e+00	1.0	0.0	7.0	7.0
1	EPIDEMIOLOGY RESEARCH AND TRAINING UNIT	1.000000e+00	1.0	0.0	1.0	1.0
2	PARATEK PHARMACEUTICALS	1.000000e+00	1.0	0.0	1.0	1.0
3	ANDROSCIENCE CORPORATION	2.000000e+00	2.0	0.0	1.0	1.0
4	FORS MARSH GROUP LLC	1.300000e+01	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...
7960	DUKE UNIVERSITY	5.350370e+09	9244.0	562.0	40978.0	41540.0
7961	UNIVERSITY OF PITTSBURGH AT PITTSBURGH	5.559754e+09	11899.0	17.0	64404.0	64421.0
7962	UNIVERSITY OF PENNSYLVANIA	5.818834e+09	12902.0	1553.0	66418.0	67971.0
7963	UNIVERSITY OF CALIFORNIA, SAN FRANCISCO	5.843529e+09	11890.0	1527.0	44033.0	45560.0
7964	JOHNS HOPKINS UNIVERSITY	7.591852e+09	14843.0	2907.0	83893.0	86800.0

7965 rows × 6 columns

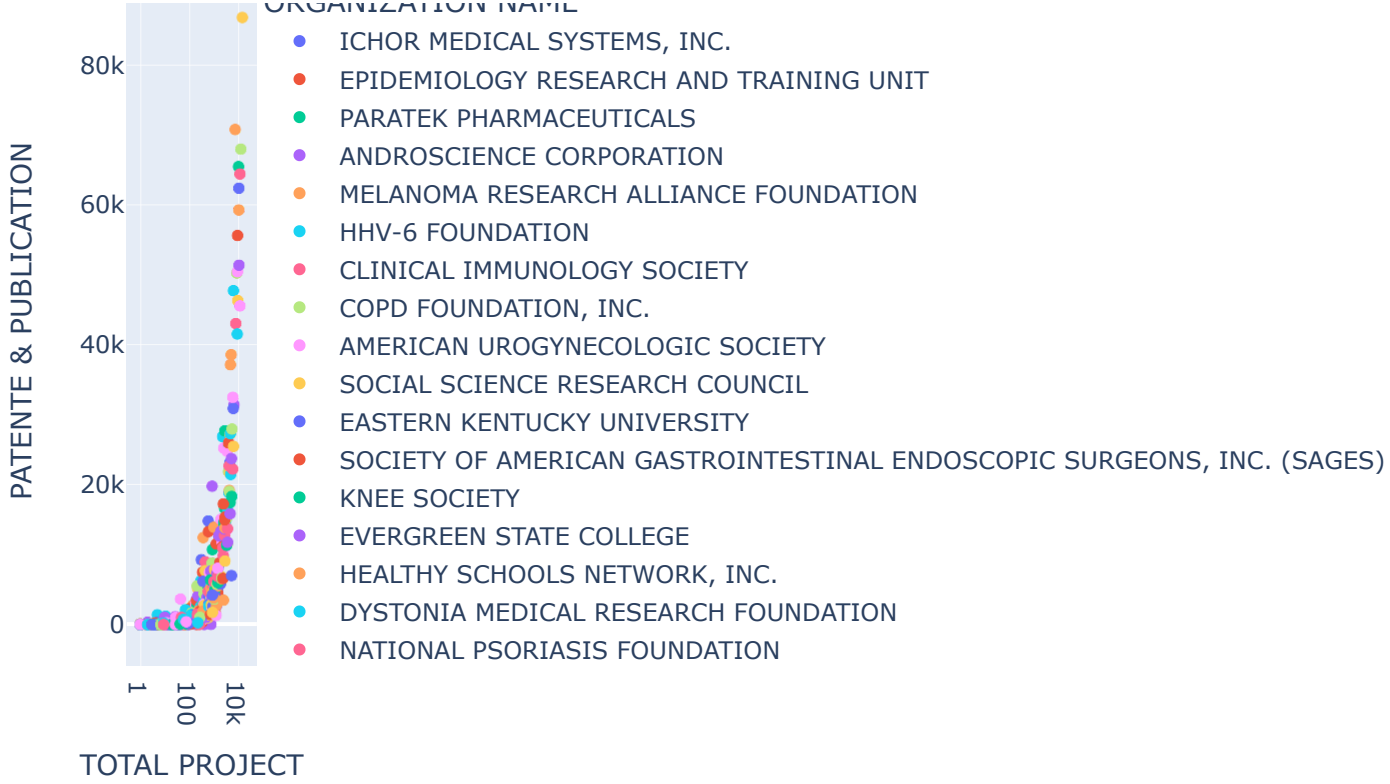
By looking at these three figures, we can find some results of efficiency.The more the project a organization has, the more patents and publications, and the more funding they get.This can be seen on the first two figures. From Figure 1, we can see that the closer to the right side, the larger the size of the points, and the more funding they have. At the same time, we can find that most of the points are concentrated in the lower left corner.Howver,something special is that some organizations with few projects evern have more patents and publications, but thier funding size is so small.We can see that many points are far from the regression line in Figure 3.

```
fig = px.scatter(table2,y="PATENTE & PUBLICATION",x="TOTAL PROJECT",size="TOTAL FUNDING",color="ORGANIZATION NAME"
                 ,title="TOTAl PATENT & Publication per Project")
fig.show()
```

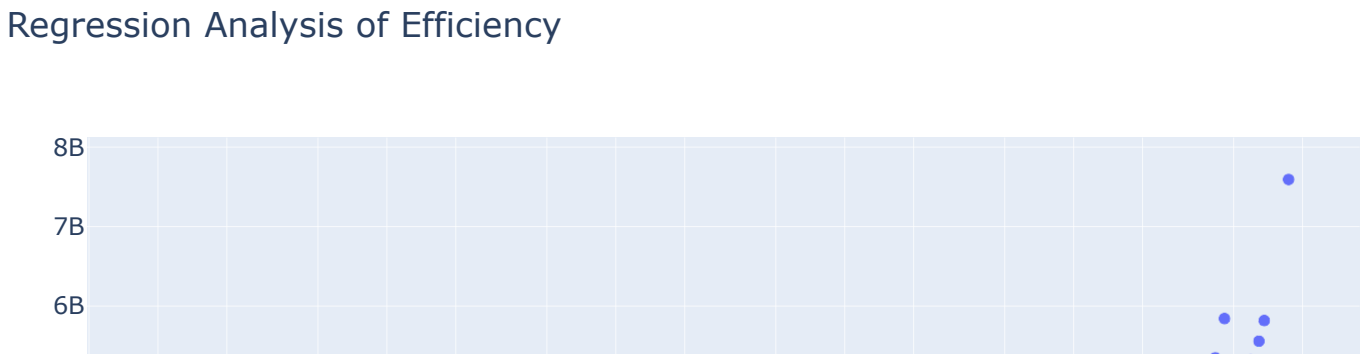
TOTAl PATENT & Publication per Project

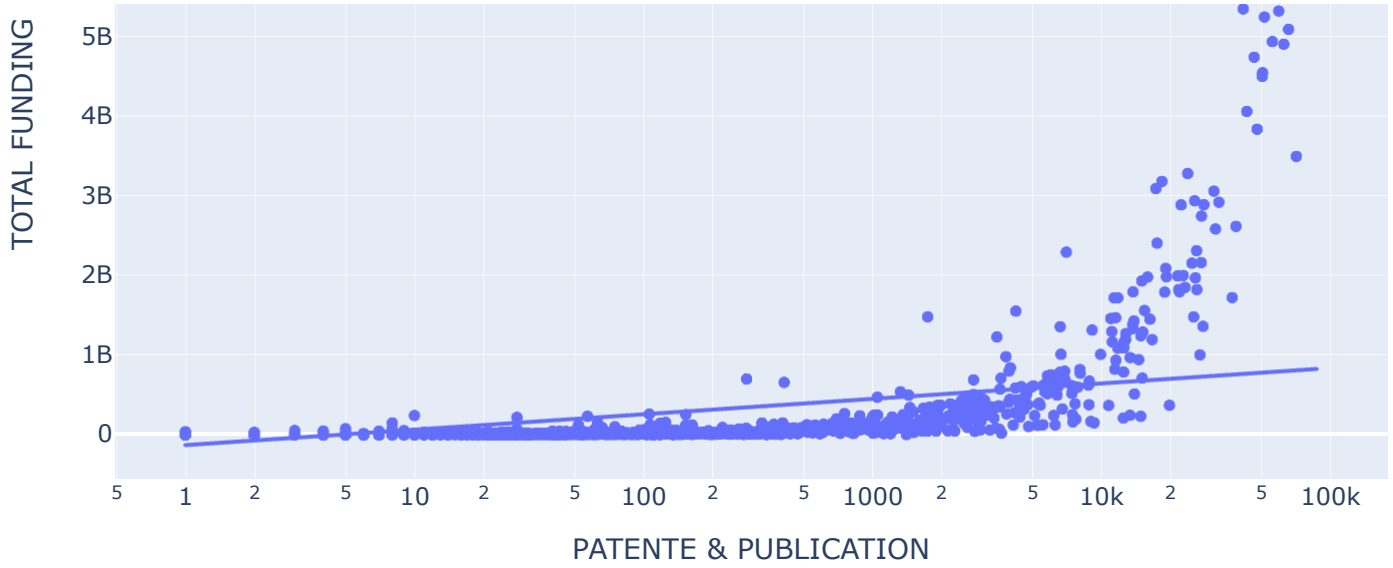

```
fig = px.scatter(table2, x="TOTAL PROJECT", y="PATENTE & PUBLICATION", log_x=True,color="ORGANIZATION NAME" ,
                 trendline="ols", trendline_options=dict(log_x=True),
                 title="Regression Analysis of Efficiency")
fig.show()
```

Regression Analysis of Efficiency




```
fig = px.scatter(table2, x="PATENTE & PUBLICATION", y="TOTAL FUNDING", log_x=True,
                 trendline="ols", trendline_options=dict(log_x=True),
                 title="Regression Analysis of Efficiency")
fig.show()
```





Part 2: Coefficient Visualization

Gini Coefficient

The extremely high value of the Gini coefficient fully reflects the imbalance in the distribution of funds, which can also be observed in the Lorenz curve.

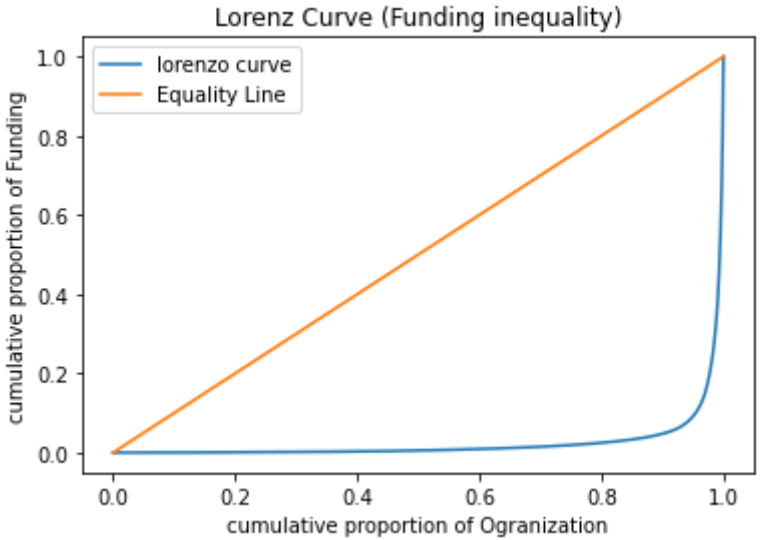
```
import math
#def gini(data_list):
eff_list=table2["TOTAL FUNDING"]
arr=eff_list.values
arr = np.array(sorted(arr))
n = len(arr)
coef_ = 2. / n
const_ = (n + 1.) / n
weighted_sum = sum([(i+1)*yi for i, yi in enumerate(arr)])
gini=coef_*weighted_sum/(arr.sum()) - const_

gini
```

```
0.9517462419972877
```

Lorenz Curve

Looking at the Lorenz curve, 95 percent of organizations account for only 8 percent of the total funding, while the remaining 5 percent accounts for nearly 92 percent, which perfectly reflects the ininequality of funding, as stated in the following comments, rich organizations become richer and poor organizations become poorer.

```
table2.sort_values(by=['TOTAL FUNDING'], ascending=True, inplace=True)
table2['percentile']=table2['TOTAL FUNDING'].cumsum()/table2['TOTAL FUNDING'].sum()
plt.plot(np.linspace(0.,1.,len(table2)), table2['percentile'],label='lorenzo curve')
plt.plot([0,1], [0,1],label="Equality Line")
plt.title('Lorenz Curve (Funding inequality)')
plt.ylabel('cumulative proportion of Funding')
plt.xlabel('cumulative proportion of Ogranization')
plt.legend()
plt.show()
```



Theil Index

```
table2=table2[table2["PATENTE & PUBLICATION"]!=0]
table2=table2[table2["TOTAL FUNDING"]!=0]
p_2=list(table2["PATENTE & PUBLICATION"])
funding=list(table2["TOTAL FUNDING"])
T=0
sum_funding = sum(funding)
sum_2p =sum(p_2)
for i in range(len(p_2)):
    w = funding[i]/sum_funding  #funding weight
    e = p_2[i]/sum_2p   # pat& pub weight
    T += w * math.log(w/e)

T
```

```
0.19422062796817025
```

The value of the Thiel coefficient is low, and we can assume that the outputs of funding and funding are matched, and the larger the lab outputs the more money they get. And we will also concluded that, if small size organizations produce more publications,they will also get more funding, it will affect the funding efficiency.

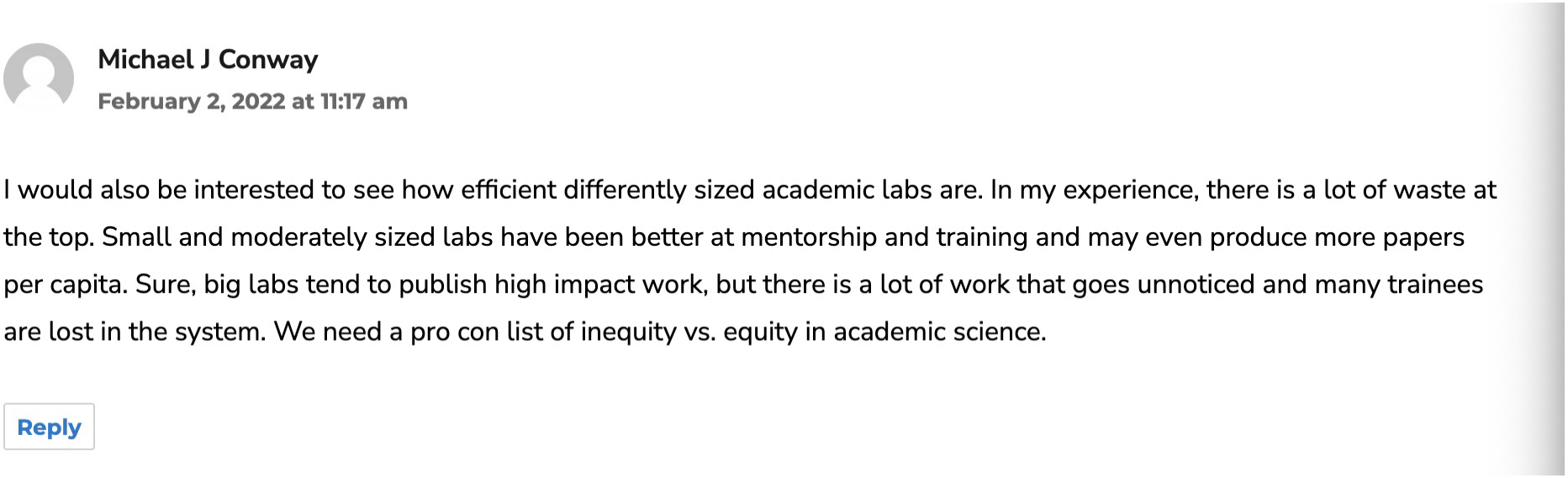
Michael J Conway

February 2, 2022 at 11:17 am

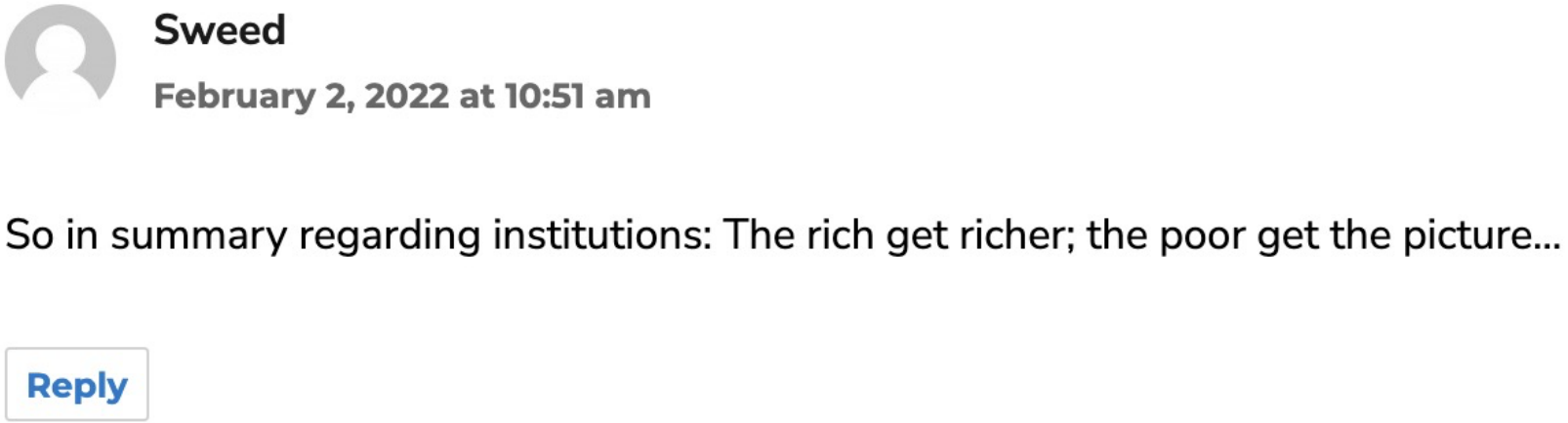
I would also be interested to see how efficient differently sized academic labs are. In my experience, there is a lot of waste at the top. Small and moderately sized labs have been better at mentorship and training and may even produce more papers per capita. Sure, big labs tend to publish high impact work, but there is a lot of work that goes unnoticed and many trainees are lost in the system. We need a pro con list of inequity vs. equity in academic science.

Reply

eff1.png



eff2.png

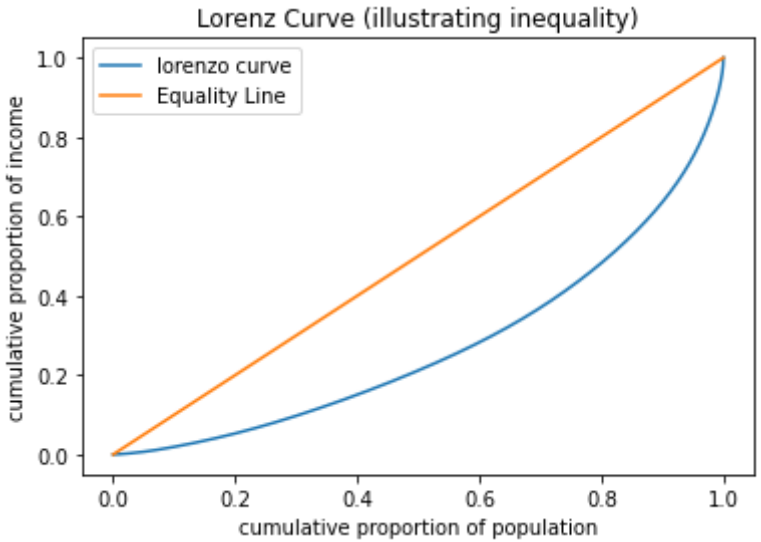


eff3.png

Lorenz curve (written by Skylar)

```
#source from report https://elifesciences.org/articles/71712/figures#content
pi=pd.read_csv(path+'pi.csv')
```

```
pi.sort_values(by=['tot_doll'], ascending=True, inplace=True)
pi['percentile']=pi['tot_doll'].cumsum()/pi['tot_doll'].sum()
plt.plot(np.linspace(0.,1.,len(pi)), pi['percentile'],label='lorenz curve')
plt.plot([0,1], [0,1],label="Equality Line")
plt.title('Lorenz Curve (illustrating inequality)')
plt.ylabel('cumulative proportion of income')
plt.xlabel('cumulative proportion of population')
plt.legend()
plt.show()
```



From the Lorenz Curve, we can observe that the distribution of grants among candidates are not equal, as illustrated as not exactly lying on the Equality Line. For example, the bottom 50% cumulative proportion of population occupies 20% of cumulative proportion of income, while the top 20% total dollars is assigned to nearly 60% of cumulative income.

Diversity & Equity

The complaint about this part is mainly a lack of hypothesis testing results. While we have not properly included this part yet (like astericks in the graph), we had done some preliminary visualizations.

Anonymous
February 2, 2022 at 1:54 pm

The text draws conclusions based on the data in shown in Table 5 and Figure 2, but these don't seem to be results of hypothesis tests. Those boxplots overlap a lot. What is the probability that the observed differences are just due to chance? What is the probability of the data given specified models? It's seems odd that the NIH would put out a report and not present basic statistical tests to support the conclusions it draws. I certainly wouldn't be funded if I turned in a grant like this!

Reply

stat1.png

PI demographics

```
import pandas as pd
path = 'https://raw.githubusercontent.com/linnilinnil/NIH-Fundings-Dashboard/main/data/'

pi=pd.read_csv(path+'pi-organization/pi.csv')
```

```
pi=pi[pi['ETHNICITY2']!="Unknown"]
pi=pi[pi['ETHNICITY2']!="Withheld"]

pi=pi[pi['gender']!="Unknown"]
pi=pi[pi['gender']!="Withheld"]

pi=pi[pi['race']!="Unknown"]
pi=pi[pi['race']!="Withheld"]
pi=pi[pi['age']!="Unknown"]

pi['tot_doll']=round(pi['tot_doll'],2)
```

```
cols=["FY","ETHNICITY2","gender","DEGREE","tot_doll","race","age"]
pi=pi[cols]
pi = pi.rename(columns={'ETHNICITY2':'hispanic','DEGREE':'degree'})
pi.head()
```

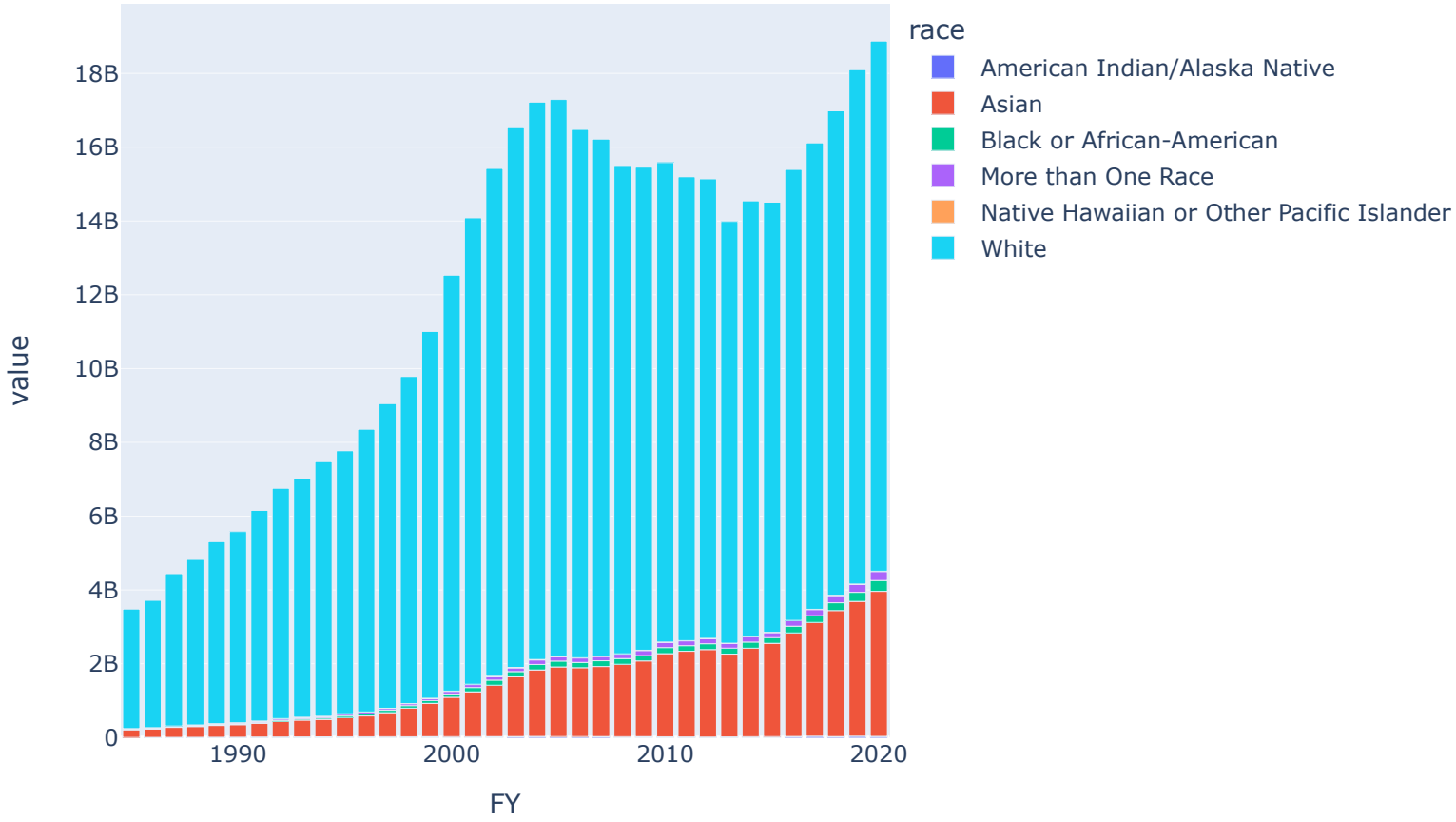
	FY	hispanic	gender	degree	tot_doll	race	age
3	1985	Not Hispanic	Male	MD	7244536.02	White	Mid
4	1985	Not Hispanic	Male	PhD	161216.56	White	Early
6	1985	Not Hispanic	Male	PhD	849001.63	White	Early
7	1985	Not Hispanic	Male	PhD	1623625.99	White	Early
8	1985	Not Hispanic	Male	MD-PhD	368401.10	White	Early

```
agg = pi.groupby(["race","FY","gender","degree","age"]).agg(sum)
divsum = agg.reset_index(0).reset_index(0).reset_index(0).reset_index(0).reset_index(0)
```

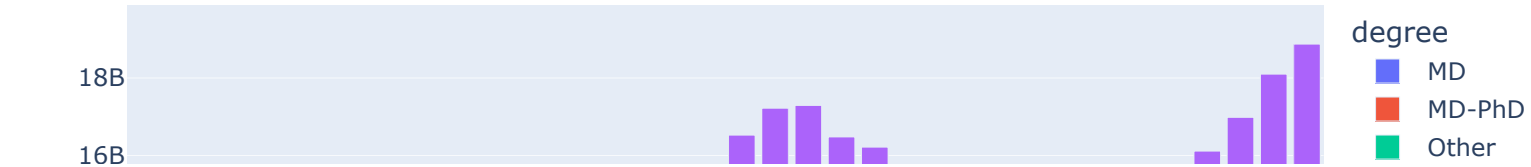
```
FutureWarning: The operation <built-in function sum> failed on a column. If any error is raised, this will raise an
exception in a future version of pandas. Drop these columns to avoid this warning.
  agg = pi.groupby(["race","FY","gender","degree","age"]).agg(sum)
```

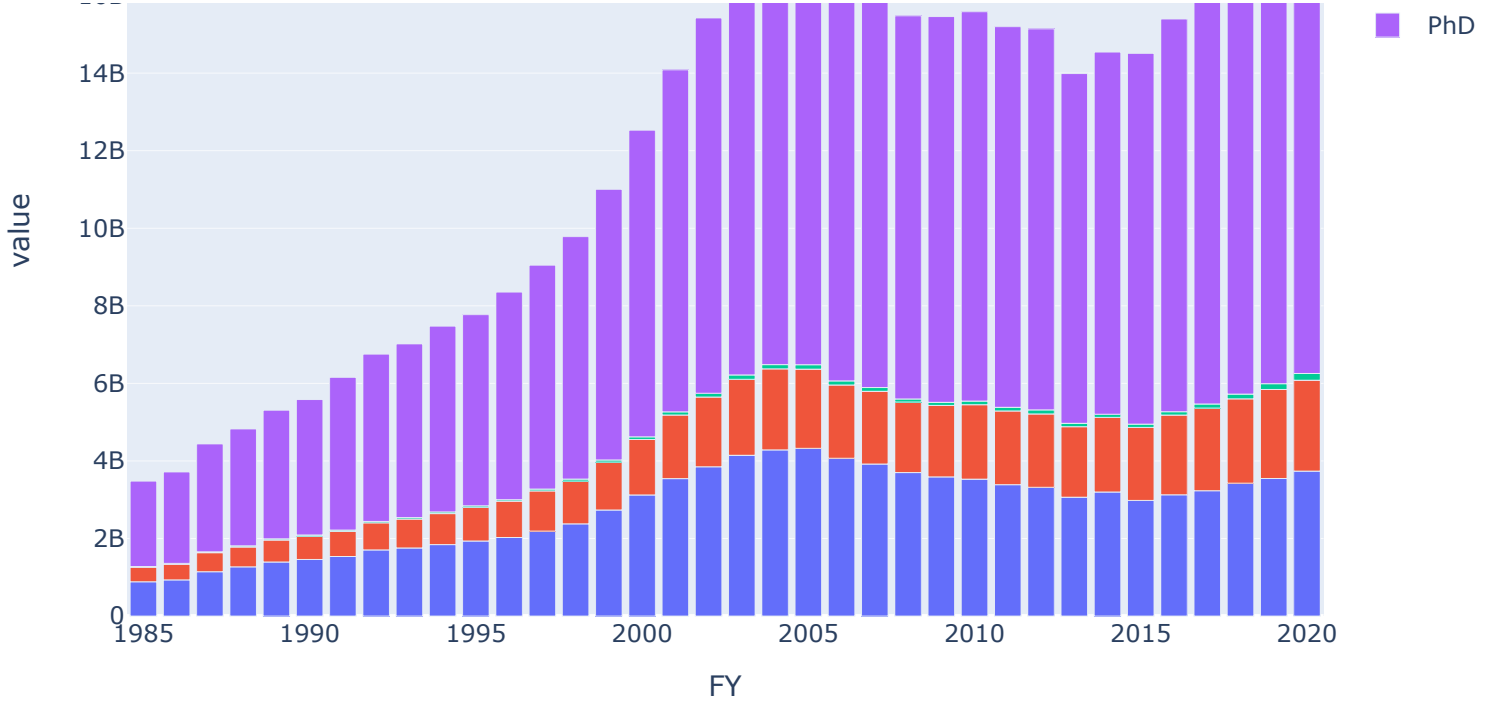
```
def stacked_bar(divsum,par):
    subdiv = divsum.groupby([par,'FY'])['tot_doll'].agg(sum)
    subdiv = subdiv.reset_index(0).reset_index(0)
    subdiv = pd.pivot(subdiv, index='FY', columns=par, values='tot_doll')
    fig = px.bar(subdiv)
    return fig
```

```
fig = stacked_bar(divsum,'race')
fig.show()
# SAME FOR DEGREE AND AGE
```

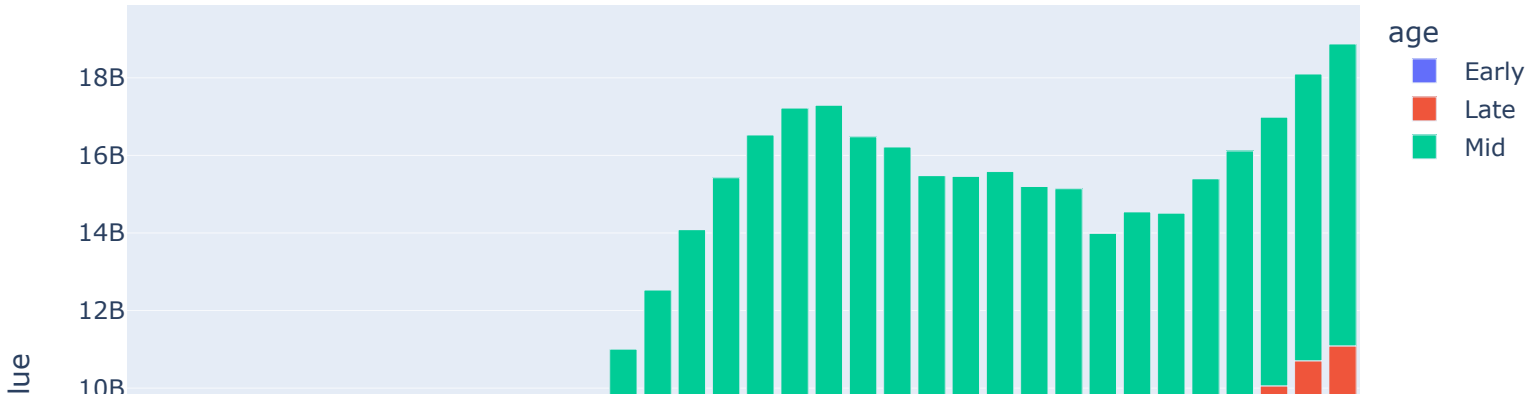


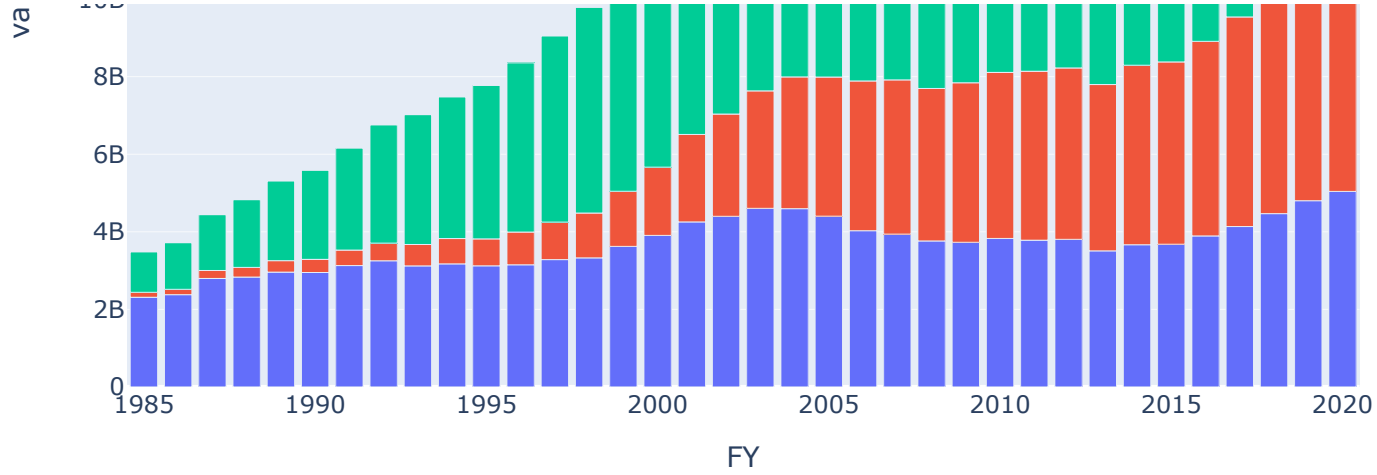

```
fig = stacked_bar(divsum,'degree')
fig.show()
# SAME FOR DEGREE AND AGE
```





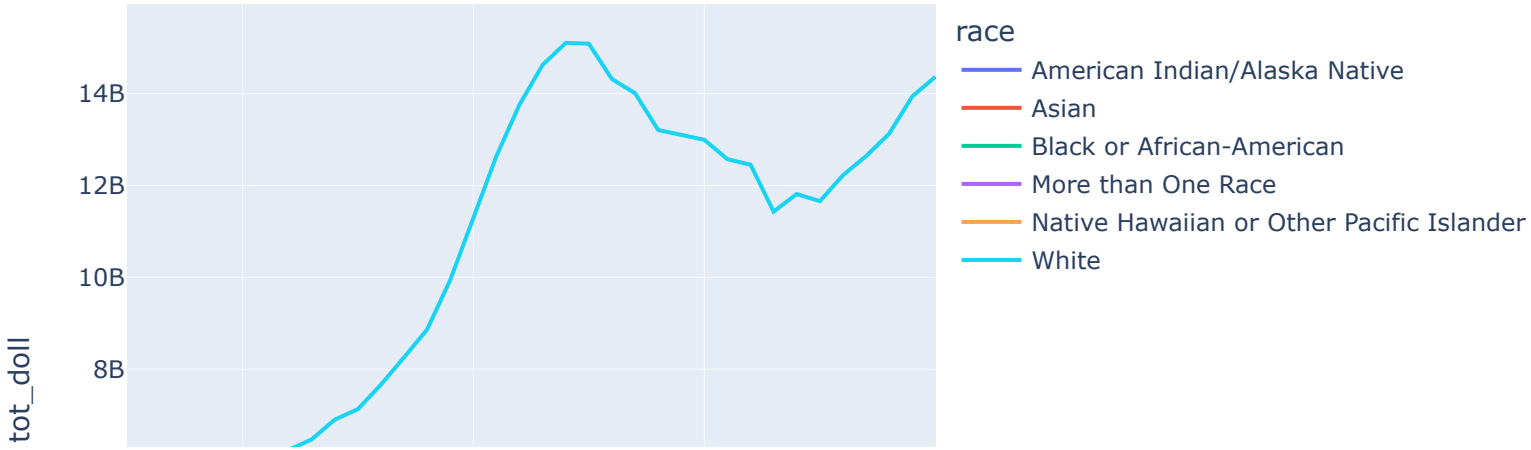

```
fig = stacked_bar(divsum,'age')
fig.show()
# SAME FOR DEGREE AND AGE
```

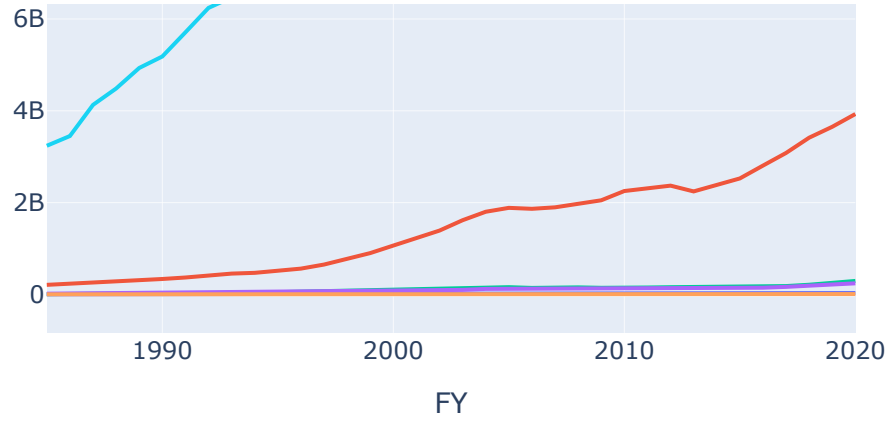




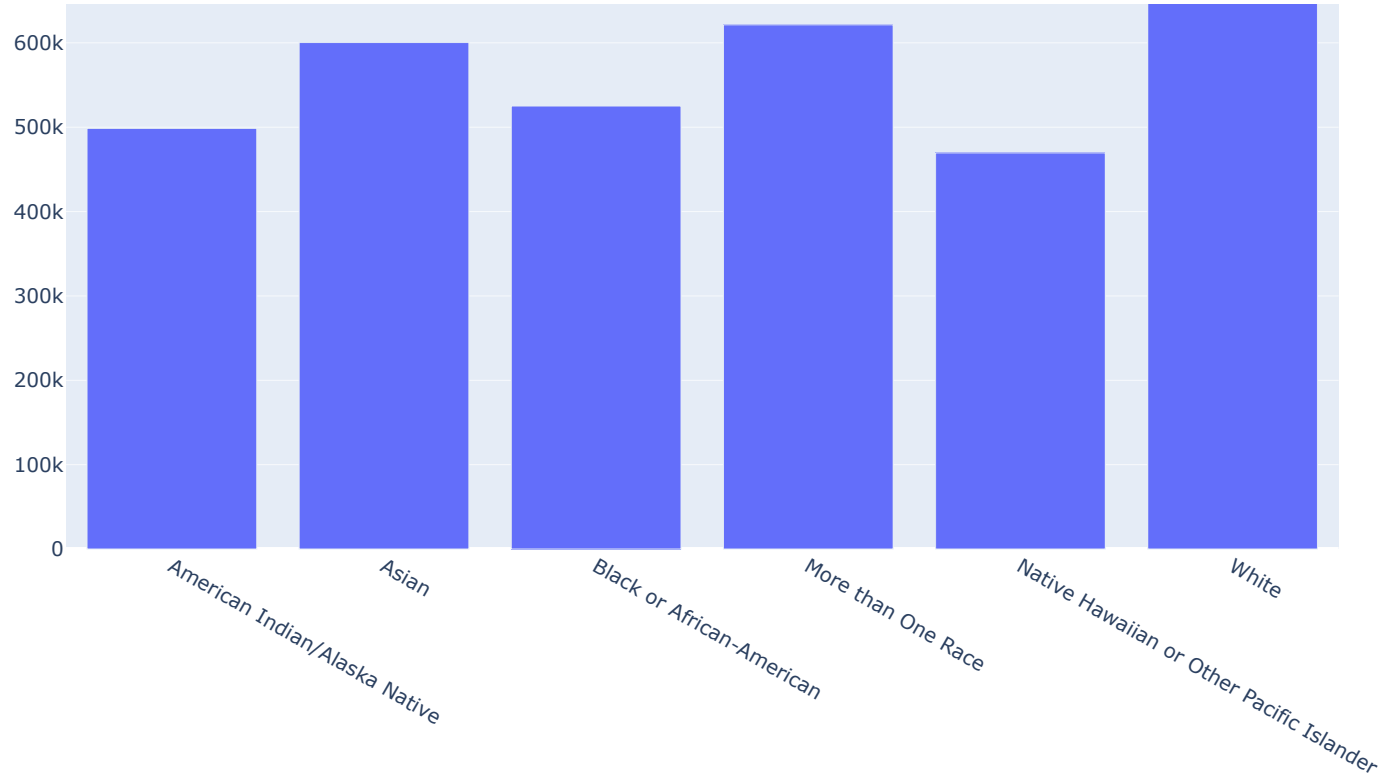
Alternative visual

```
para = 'race'
subdiv = divsum.groupby([para,'FY'])['tot_doll'].agg(sum).reset_index(0).reset_index(0)
linefig = px.line(subdiv,x='FY',y='tot_doll',color=para,hover_name = 'FY',
                  hover_data = ['FY'])
linefig.show()
```






```
# when it's on dash it would be a function.
pag = pd.read_csv(path + 'pag.csv')
ag = pag[pag['FY']==2020].loc[:,[para,'mean','count']]
ag['prod'] = ag['mean'] * ag['count']
ag = ag.groupby([para]).agg('sum').apply(lambda x:x['prod'] / x['count'],axis=1)

barfig = go.Figure()
barfig.add_trace(go.Bar(
x=list(ag.index), y=list(ag.values),
#error_y=dict(type='data', array=ag['std'])
))
barfig.update_layout(
font=dict(
        size=10,
    ),
title = 'average funding per PI in '+str(2020),
showlegend=False)
```



Diversity within cancer research (by Skylar)

National Cancer Institute Triennnial Inclusion Reports, Appendix B: Enrollment for All NIH-Defined Clinical Research
Data scraped and processed with Power Query

https://report.nih.gov/sites/report/files/docs/2019-NCI-Triennial-Inclusion-Report.pdf
https://report.nih.gov/sites/report/files/docs/NCITriennialInclusionReportFY2019_FY2021Final.pdf

(In future, we will try making category-specific analysis avaible for all areas. But for now, this is an example in the cancer research. (Not displayed on dashboard))

Sex & Ethnicity vs Enrollment for NIH-Defined Clinical Research

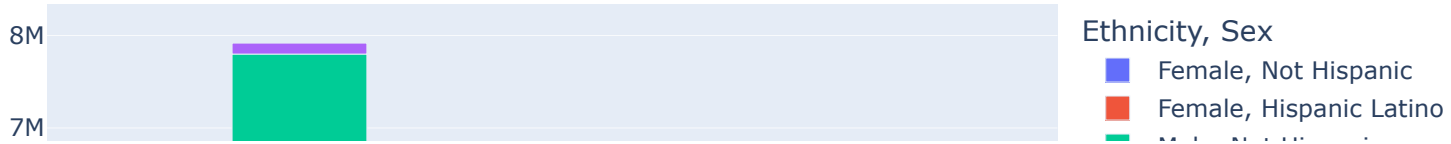
```
enroll_ethn=pd.read_csv(path+'Enroll_ethnicity.csv')
enroll_ethn.head()
```

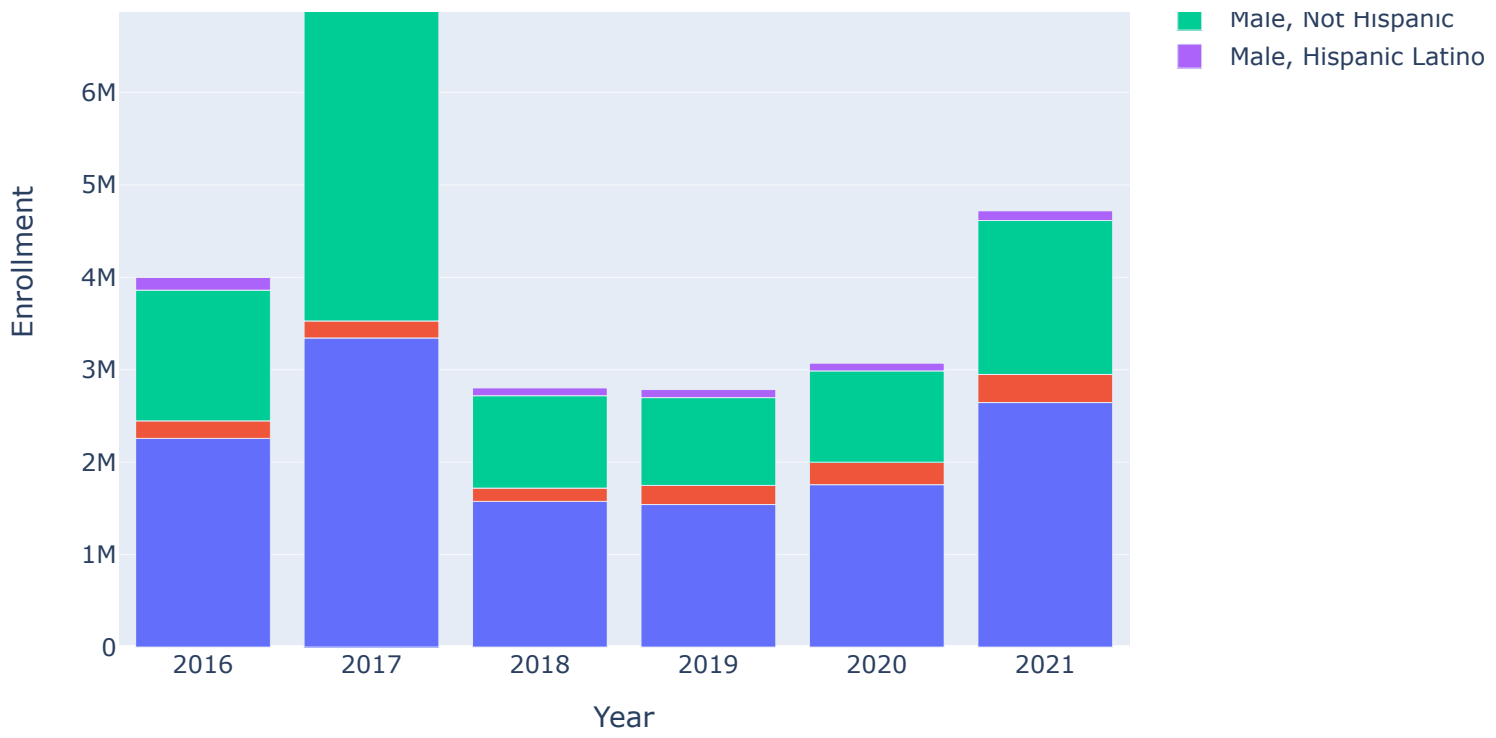
	Year	Sex	Ethnicity	Enrollment
0	2016	Female	Not Hispanic	2257542
1	2016	Female	Hispanic Latino	189508
2	2016	Male	Not Hispanic	1414034
3	2016	Male	Hispanic Latino	136656
4	2017	Female	Not Hispanic	3343498

```
enroll_ethn['Ethnicity, Sex'] = enroll_ethn[enroll_ethn.columns[1:3]].apply(
    lambda x: ', '.join(x.dropna().astype(str)), axis=1)
enroll_ethn.head()
```

	Year	Sex	Ethnicity	Enrollment	Ethnicity, Sex
0	2016	Female	Not Hispanic	2257542	Female, Not Hispanic
1	2016	Female	Hispanic Latino	189508	Female, Hispanic Latino
2	2016	Male	Not Hispanic	1414034	Male, Not Hispanic
3	2016	Male	Hispanic Latino	136656	Male, Hispanic Latino
4	2017	Female	Not Hispanic	3343498	Female, Not Hispanic

```
fig = px.bar(enroll_ethn, x="Year", y="Enrollment",
             color='Ethnicity, Sex', hover_name="Ethnicity, Sex")
fig.show()
```





Sex & Race vs Enrollment for NIH-Defined Clinical Research

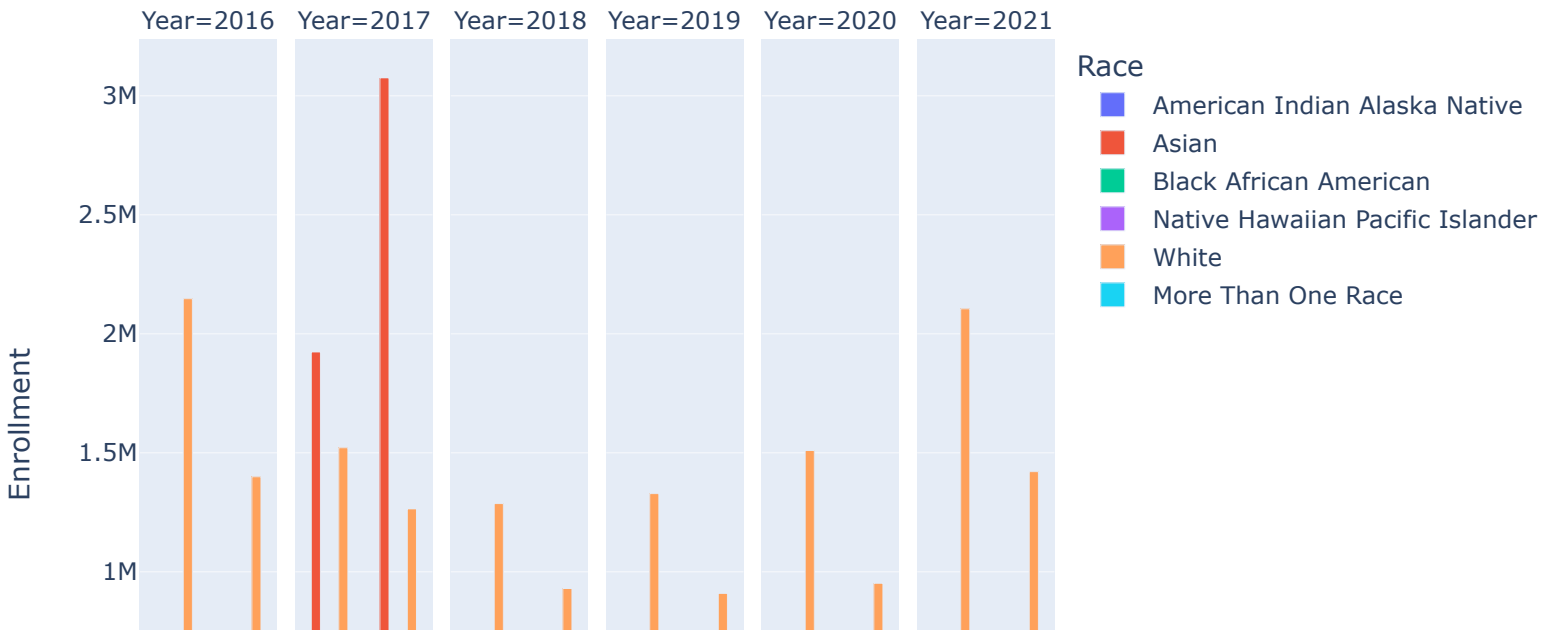
```
enroll_race=pd.read_csv('https://raw.githubusercontent.com/linnilinnil/NIH-Fundings-Dashboard/main/Enroll_Race.csv')
enroll_race
```

	Year	Sex	Race	Enrollment
0	2016	Female	American Indian Alaska Native	8289
1	2016	Female	Asian	185301
2	2016	Female	Black African American	162469
3	2016	Female	Native Hawaiian Pacific Islander	3832
4	2016	Female	White	2147704
...	...	...	...	...

	Year	Sex	Race	Enrollment
67	2021	Male	Asian	137938
68	2021	Male	Black African American	175627
69	2021	Male	Native Hawaiian Pacific Islander	29970
70	2021	Male	White	1420952
71	2021	Male	More Than One Race	13741

72 rows × 4 columns

```
enroll_race['Race, Sex'] = enroll_race[enroll_race.columns[1:3]].apply(
    lambda x: ', '.join(x.dropna().astype(str)), axis=1)

fig = px.bar(enroll_race, x="Sex", y="Enrollment", facet_col="Year",
             color='Race', hover_name="Race, Sex", barmode='group')
fig.show()
```



Data from NIH RCDC Inclusion Statistics Report for Diversity in Different Cancer Research Areas

- https://report.nih.gov/RISR/#/home?single=All%20Studies&ic=NCI&rcdcFilter=cancer&facet=Race&fiscalYear=2021

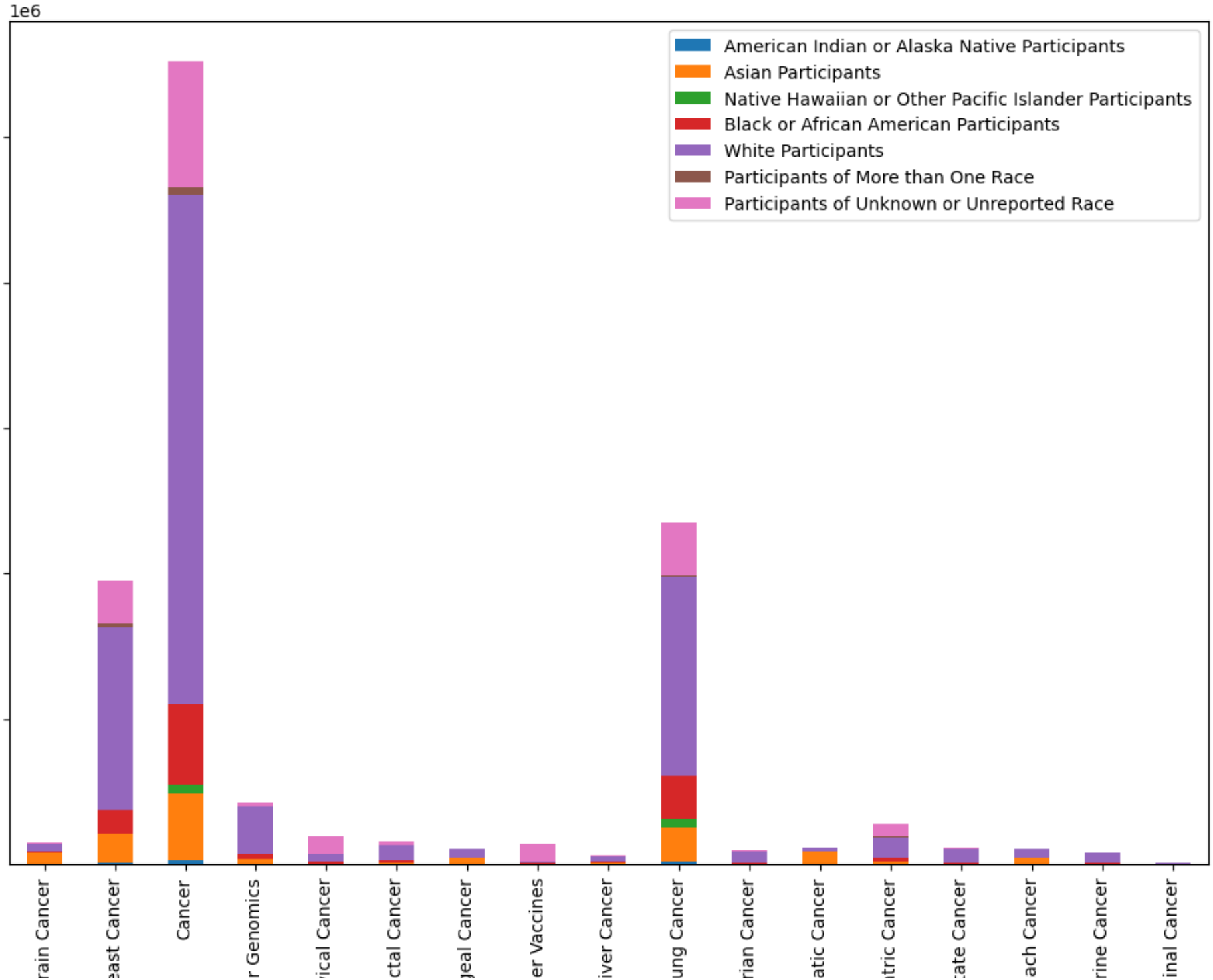
```
rcdc=pd.read_csv(path+'2021-Race-Exc-Single.csv')
cols=['RCDC Category',
      'American Indian or Alaska Native Participants','Asian Participants',
      'Native Hawaiian or Other Pacific Islander Participants','Black or African American Participants',
      'White Participants','Participants of More than One Race','Participants of Unknown or Unreported Race']
rcdc=rcdc[cols]
rcdc=pd.DataFrame(rcdc.replace('<12',0))

for col in cols[1:]:
    rcdc[col]=rcdc[col].astype(int)

rcdc.head(3)
```

	RCDC Category	American Indian or Alaska Native Participants	Asian Participants	Native Hawaiian or Other Pacific Islander Participants	Black or African American Participants	White Participants	Participants of More than One Race	Participants of Unknown or Unreported Race
0	Brain Cancer	187	81047	50	7894	45793	190	8202

	RCDC Category	American Indian or Alaska Native Participants	Asian Participants	Native Hawaiian or Other Pacific Islander Participants	Black or African American Participants	White Participants	Participants of More than One Race	Participants of Unknown or Unreported Race
1	Breast Cancer	9142	200317	2394	161097	1256260	23005	299274
2	Cancer	29466	456813	62967	552598	3502761	55345	863694

```
rcdc.set_index('RCDC Category').plot(kind='bar', stacked=True,figsize=(10,10))
plt.tight_layout()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9509adad90>
```


1e6
5
4
3
2
1
American Indian or Alaska Native Participants
Asian Participants
Native Hawaiian or Other Pacific Islander Participants
Black or African American Participants
White Participants
Participants of More than One Race
Participants of Unknown or Unreported Race
rain Cancer
ast Cancer
Cancer
r Genomics
ical Cancer
ctal Cancer
eal Cancer
er Vaccines
iver Cancer
ung Cancer
rian Cancer
atic Cancer
tric Cancer
ate Cancer
ach Cancer
rine Cancer
inal Cancer

We can observe a recurring pattern that White researchers dominate in all cancer research areas, followed by Asian researchers and/or Unknown or Unreported Race. This is not surprising given the overall dominance of white candidates in NIH research grants.

Dashboard component

Example: Map + Slider

Dash is a built upon the plotly library and well-integrated with flask for web app deplotment. From their product description: "Dash apps give a point-&-click interface to models written in Python, vastly expanding the notion of what's possible in a traditional 'dashboard.'". Below is a demostration of how to transfer the choropleth maps we've plotted above to a web-based interactive dashboard.

To get started, create a python file name **app.py**, where you would write the html components and callbacks (mainly for updating graphs based on user inputs); also make a new folder under the same directory name **assests**, where you will store your images and CSS (in **style.css**)

In **app.py**, the code would look something like this:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.

from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd

# Initialize app
app = Dash(__name__)

# Define, read, or pull your data
df = ...

fig = px... # Define the graph you want to put on the dashboard

app.layout = html.Div(children=[
    html.H1(children='Hello Dash'),

    html.Div(children='''
```

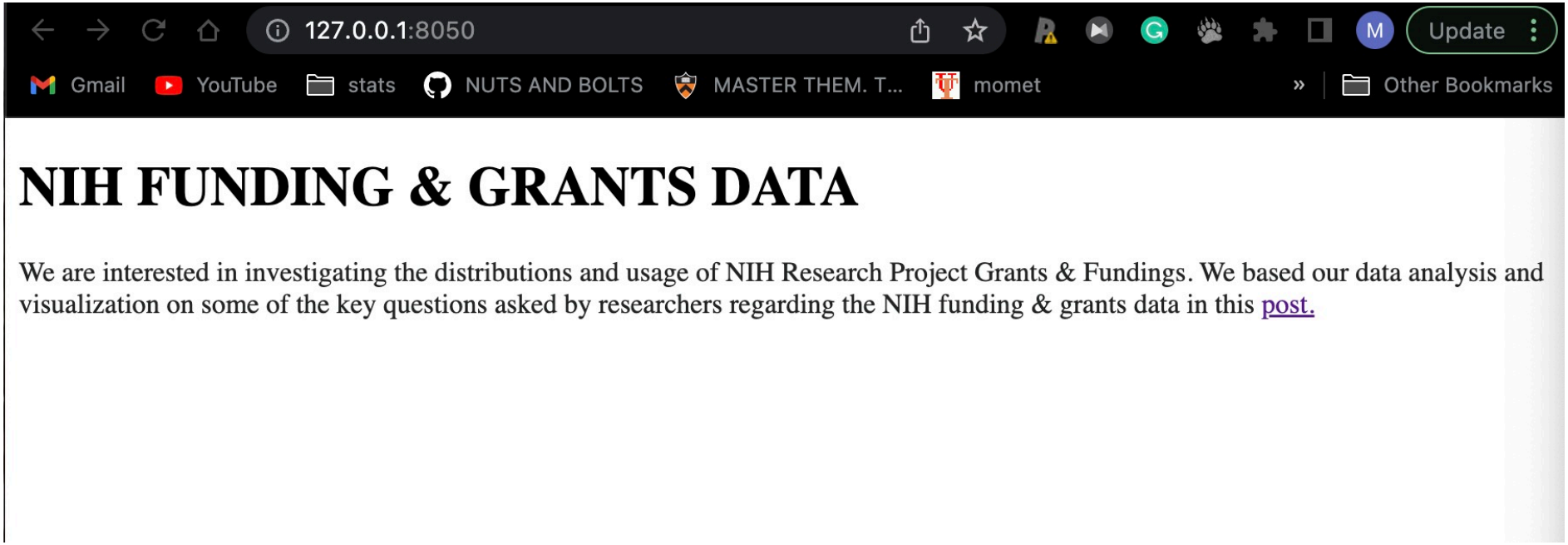
```
        Dash: A web application framework for your df.
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
]) ## Define HTML layout

## Just like flask
if __name__ == '__main__':
    app.run_server(debug=True) # Setting this to true will allow browser                    auto-refresh on code change
```

If you want to read more about how this becomes what you will see in the borwser, you could read more here.

```
#importing the necessary components
import dash
from dash import html, dcc
from dash.dependencies import Input, Output, State
import dash_daq as daq
import dash_bootstrap_components as dbc
import numpy as np
import pandas as pd
import plotly.graph_objs as go
import plotly.express as px

#host files on github
path = 'https://raw.githubusercontent.com/linnilinnil/NIH-Fundings-Dashboard/main/'

#read in file, for demo purpose they all come from pre-saved csv.
fund_org = pd.read_csv(path + "fund_org_avg.csv")
fund_pi = pd.read_csv(path + "fund_pi_avg.csv")
fund_proj = pd.read_csv(path + "fund_proj_avg.csv")



#------------------------------------------------- APP ----------------------------------------------------
```

```
app = dash.Dash(__name__)

server = app.server

app.layout = html.Div([

    html.Div([
        html.H1(children='NIH FUNDING\n& GRANTS DATA'),
        html.Label(['We are interested in investigating the distributions and usage of NIH Research Project Grants & Fun
                    html.A('post.',href='https://nexus.od.nih.gov/all/2022/01/18/inequalities-in-the-distribution-of-nat
                    style={'color':'rgb(33 36 35)'})
    ], className='side_bar'),
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

Which will show you something like this:



blank.jpeg

Then, we move on to add the choropleth to the page, and a slider option to choose which year's data to display.

To write a slider, we first define it before the app.layout, and create another html.div to store it.

```
#define a slider
year_slider = daq.Slider(
        id = 'year_slider',
        handleLabel={"showCurrentValue": True,"label": "Year"},
        marks = {str(i):str(i) for i in years},
        min = min(years),
        max = max(years),
        size = 450,
        color='0c4db4'
    )
#layout >>
    #htmlDiv: for title and descripton
    #htmlDiv: for year slider
html.Div([year_slider],
        style={'margin-left': '15%', 'position':'relative', 'top':'100px'}),
```

slider.jpeg

We now move on to the graph component, which would be hosted under dcc.Graph:

```
##after the slider div
 dcc.Graph(id='map',
              figure = dict(
              ),
              style={'position':'relative', 'top':'200px'})
```

Don't worry about how empty it looks at the moment, we would add a **callback** function to update the graph content based on the slider selection. Callbacks are functions that are automatically called by Dash whenever an input component's property changes, in order to update some property in another component (the output).

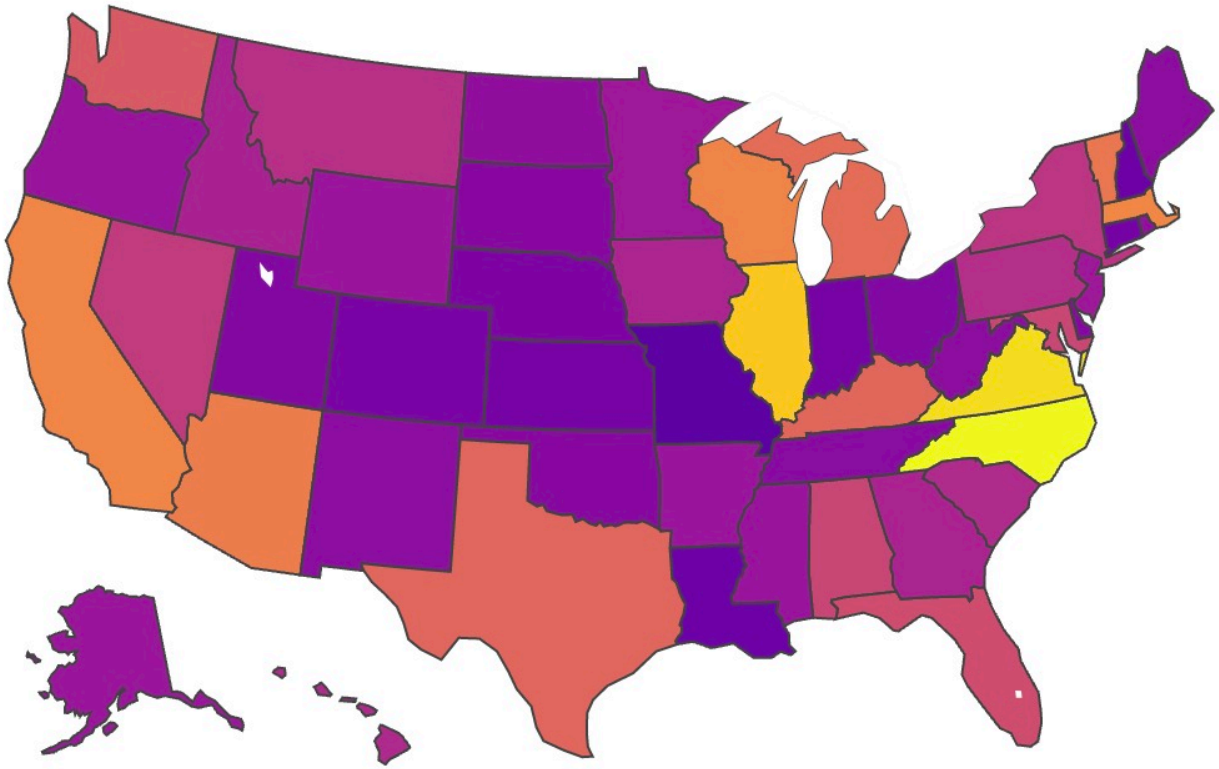
(To include other existing user selected components as inputs, that might not necessarily be changed along the specified input, use [State])

In this callback, we return a plotly figure (which consists of the data and layout components) based on the user-selected year and the dataframes that already exsit in the environment.

```
@app.callback(
    Output(component_id='map', component_property='figure'),
    Input(component_id='year_slider', component_property='value')
)
def update_map(yr):
    by_year = fund_proj[fund_proj['YEAR']==int(yr)]
    by_year = by_year.groupby(
    ['CODE']
    )['FUNDING'].agg('mean').reset_index(0)

    fig = px.choropleth(by_year,color='FUNDING',
                        locations="CODE",
                        locationmode="USA-states",
                        scope="usa",
                        range_color=(0, 2*10e5),
                        width=800,
                        height=500)

    return fig
```

Year
2020

2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022



FUNDING
2M
1.5M
1M
0.5M
0

Example: Bar + Hover callback

mapslider.jpeg

To read more on how this work, refer to [Click and hover callbacks]

HTML Part: we structure the css with the row, box, and (relative) width property

```
html.Div([
                html.Div([
                            html.Div([
                                        fatal_radio,
                            ]),
                            dcc.Graph(  id='histo',
                                        figure = dict(),
                                        hoverData={'points': [{'hovertext': 'Cancer '}]},
                                        )],
                            style={'margin-left':'20%','width': '55%'},
                            className='box'),
                html.Div([
                            dcc.Graph(id='hoverline',
                                    figure = dict(),)],
                            style={'width': '45%',},
                            className='box'),],
                className='row'),
```

CSS:

```
.box {
    border-radius: 20px;
    background-color: #F9F9F8;
    margin: 10px;
    padding: 25px;
    box-shadow: 2px 2px 2px lightgrey;
  }
  .row{
    display: flex;
  }
```

Callback:

```
@app.callback(
    [Output('hoverline', 'figure'),
     Output('histo', 'figure'),], #update histograms and line graph for a category
    [Input('histo', 'hoverData'),
     Input('fatal_radio', 'value') #change based on radio button and hover
     ])
def update_line(hoverData,val):
    if val == 0:
        df = fatal10
        type = 'fatal'
    else:
        type = 'nonfatal'
        df = nonfatal10
    selected_area = hoverData['points'][0]['hovertext']
    # if you are unsure about the structure of the hoverdata
    # you could always set the output to 'children'
    # and display the hover content, or whatever you want to print
    return draw_line(df, selected_area),get_histo(type,df)
    # functions pre-defined to create line plots and histograms
    # graph handles are returned and displayed in the corresponding divs
```

Final Dashboard

Because there are quite a lot of components on this dashboard, I wouldn't go through the nitty-gritty of integrating every single one of them. Instead, here are some screenshots of the dashboard components:

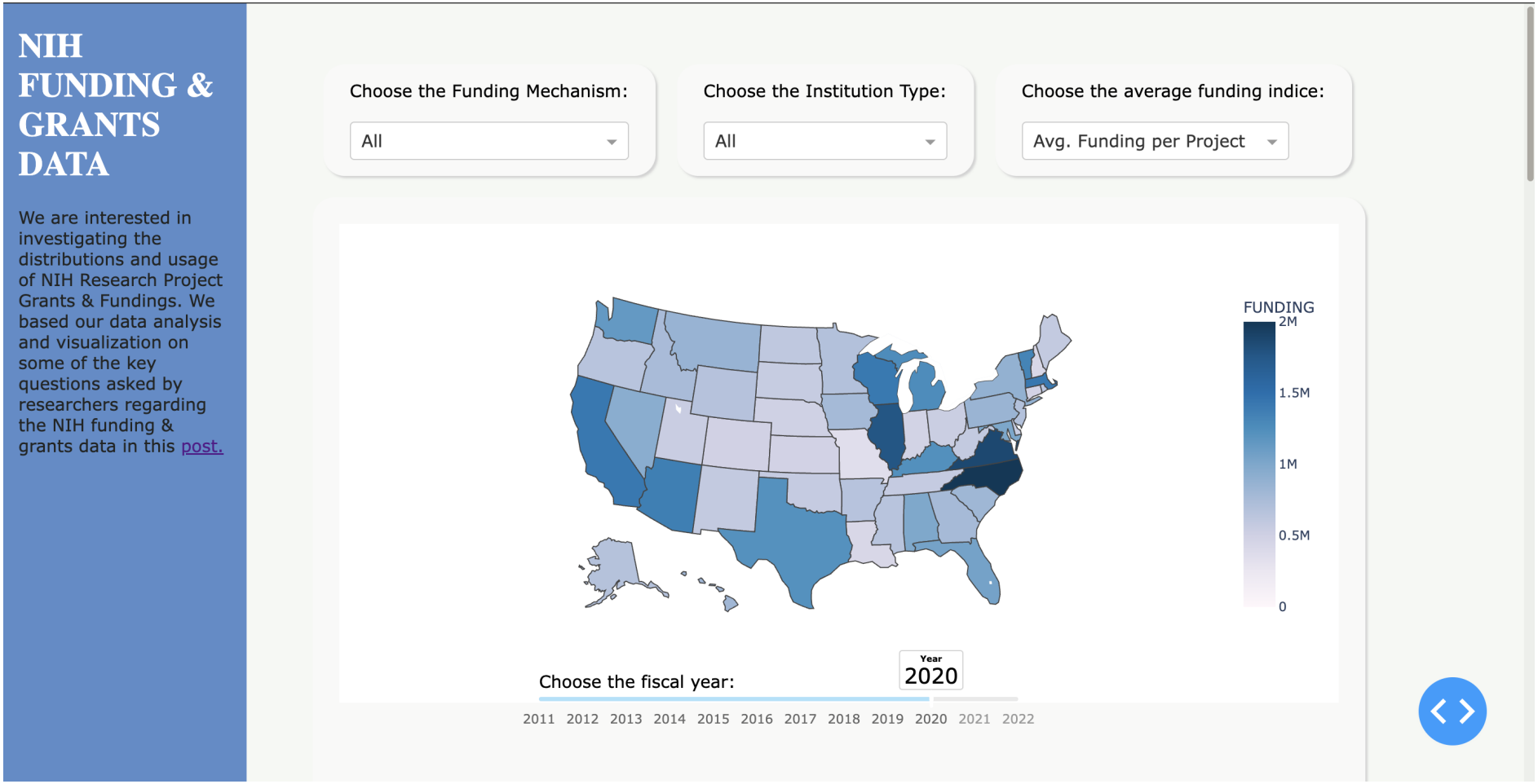
1. Funding distribution choropleth map that could be filtered based on institution types, funding mechanisms, years, and metrics.
2. Parallele categories diagrams that examine the relationship among years a category has existed, annual mortality rate (raw counts in U.S.), and estimated percentage change in funding this year.
3. Top ten invested fatal/non-fatal research areas/conditions, once hover would display the change in funding over time as a line plot.

4. Stacked histograms that visualize change in funding disparity among different populations (by race, by age - stage in academic career, and degree type)

5. Line plot of the above, but once hover would display the average funding per PI for different group in a certain year.

(If you want to run the demo locally, follow the instruction in git to clone the repo.)

The functions and demo data are stored in separate folders then imported when needed.

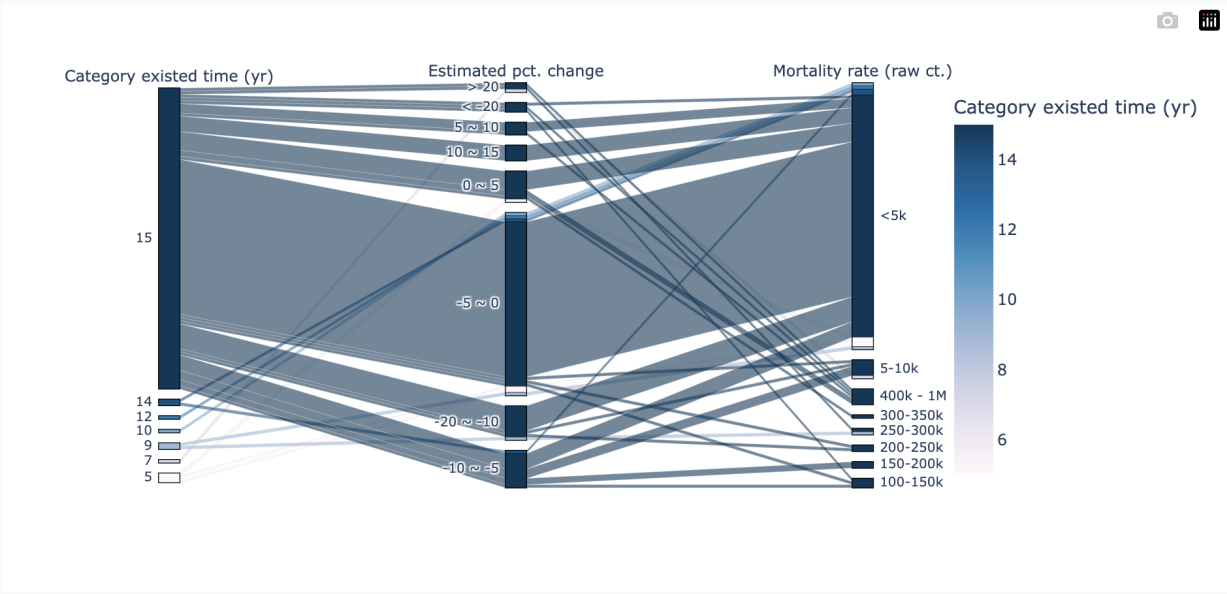


map.png

NIH FUNDING & GRANTS DATA

We are interested in investigating the distributions and usage of NIH Research Project Grants & Fundings. We based our data analysis and visualization on some of the key questions asked by researchers regarding the NIH funding & grants data in this post.

2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022



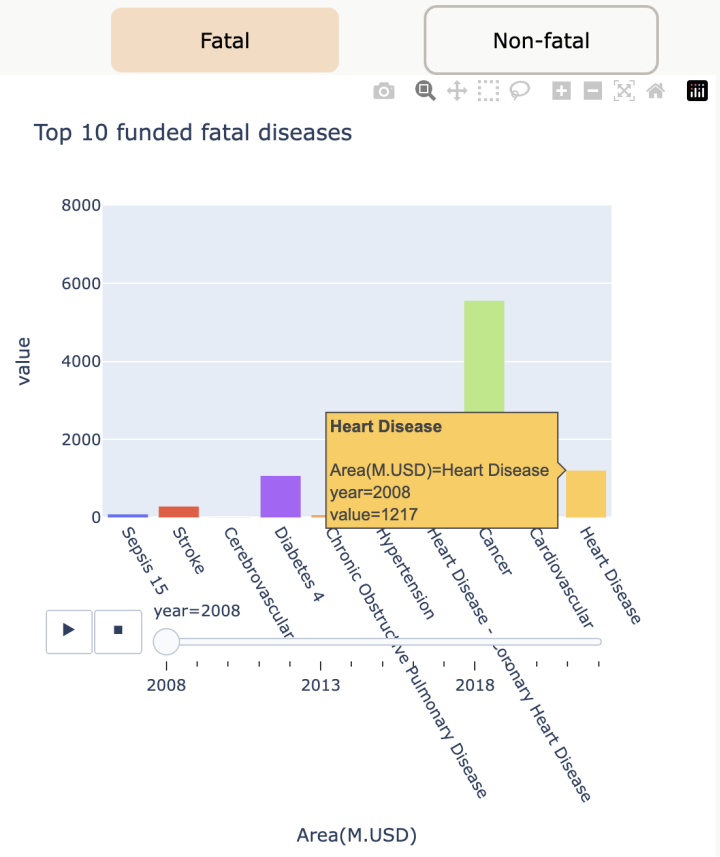
Fatal Non-fatal

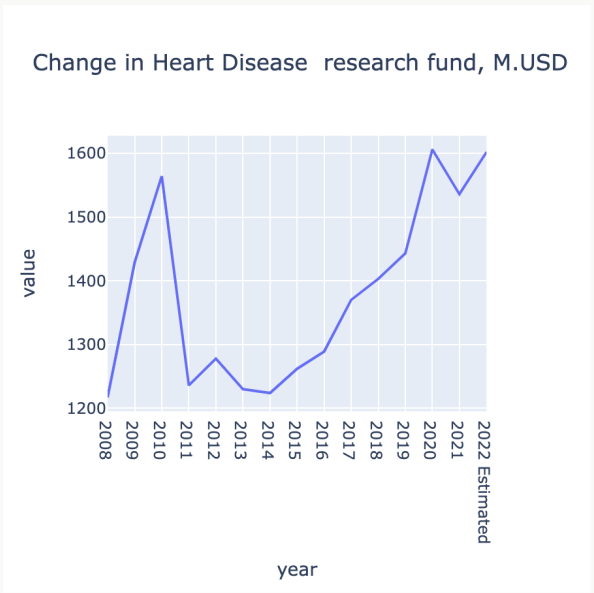
Change in Cancer research fund, M.USD

cate.png

NIH FUNDING & GRANTS DATA

We are interested in investigating the distributions and usage of NIH Research Project Grants & Fundings. We based our data analysis and visualization on some of the key questions asked by researchers regarding the NIH funding & grants data in this post.





10.png

NIH FUNDING & GRANTS DATA

We are interested in investigating the distributions and usage of NIH Research Project Grants & Fundings. We based our data analysis and visualization on some of the key questions asked by researchers regarding the NIH funding & grants data in this post.

stack.png

line.png

Discussion

(This part is written by Qin Lin)

There are various factors that could potentially hinder us from achieving our goal of creating a real-time, interactive data dashboard that monitors and showcases trends in funding distributed by the NIH. Two of these factors are:

1. Availability and quality of data: One significant challenge we may face is the availability and quality of data. Although the NIH provides detailed information on grants awarded, the data may not be comprehensive, accurate, or up-to-date. The quality of data can

significantly impact the accuracy of our analysis, and if we do not have access to the right data, we may not be able to produce meaningful insights.

2. Technical constraints: Developing a real-time, interactive data dashboard requires expertise in data visualization, web development, and computational power. Technical issues such as system crashes, slow loading times, and data storage can also be a challenge. Inadequate computational resources may also limit the amount of data we can analyze, or slow down the process, making it difficult to produce the dashboard in a timely and efficient manner.

Some of the risks that may be applicable for our project include: 1) Data privacy and security: As we are dealing with sensitive data, there is a risk of data breaches or unauthorized access to the information we collect. Ensuring data privacy and security will be critical to maintaining the trust of our users and protecting the confidentiality of grant applicants and recipients. 2) Bias in analysis: Data analysis is subject to bias, and there is a risk that our analysis may be biased, either intentionally or unintentionally. To mitigate this risk, we will need to ensure that our data sources are diverse and representative of different perspectives and that we use rigorous statistical methods to analyze the data. 3) Limited user adoption: The success of our dashboard will depend on user adoption, and there is a risk that our target audience may not find the dashboard useful or engaging. To address this risk, we will need to involve users in the design process, gather feedback, and iterate on the design to ensure that it meets the needs of our users.

Ethics

(Also written by Qin Lin)

As we develop the real-time, interactive data dashboard to monitor and showcase trends in funding distributed by the NIH, it is crucial to consider the potential impacts of our product on its users and the broader world.

Our dashboard has the potential to be a powerful tool for promoting transparency in funding distribution, and we need to ensure that it is not biased or harmful to any group.

One potential bias is the representation of different research areas. We must ensure that the dashboard's data is representative of the different research areas and accurately reflects their funding trends. We will mitigate this bias by using data sources from multiple institutions, verifying the data's accuracy, and using rigorous statistical methods to analyze the data.

Another potential bias is the representation of different groups of researchers. We must ensure that the dashboard's data accurately represents the funding distribution to different groups of researchers, including those from underrepresented communities. We will mitigate this bias by using data sources from diverse institutions and using statistical methods that are sensitive to underrepresented groups' needs. We also need to consider the potential impacts of our dashboard on its users and the broader world. Our dashboard has the potential to

change the way researchers approach grant applications, reviews, and usage, and we need to ensure that it is not harmful to any group. To mitigate this risk, we will:

1. Engage with diverse stakeholders: We will engage with a diverse group of stakeholders, including researchers, institutions, and community groups, to gather feedback on our dashboard's potential impacts and identify any potential biases.

2. Regularly update and improve the dashboard: We will regularly update and improve the dashboard based on user feedback to ensure that it meets the needs of its users and that it is not harmful to any group.

3. Ensure data privacy and security: We will ensure data privacy and security to maintain the trust of our users and protect the confidentiality of grant applicants and recipients. Overall, we will be transparent about our methods and analysis, involve diverse stakeholders in the development process, and regularly evaluate and improve our dashboard to ensure that it is not biased or harmful to any group.