Finding out US R1 & R2 Universities in Close Proximity

Mount Drives

This is done to mount the Google Drive for Google Colab.

```
### Mount google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
## set working directory
import os
os.chdir('/content/drive/MyDrive/Office/Dropbox/RMS/Research/Grad-Applicat
```

Import Modules and files

```
! pip install openpyxl # for xlsx
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.12/dist-
packages (3.1.5)
Requirement already satisfied: et-xmlfile in
/usr/local/lib/python3.12/dist-packages (from openpyxl) (2.0.0)
```

```
!pip install folium # for map
```

```
Requirement already satisfied: folium in /usr/local/lib/python3.12/dist-
packages (0.20.0)
Requirement already satisfied: branca>=0.6.0 in
/usr/local/lib/python3.12/dist-packages (from folium) (0.8.2)
Requirement already satisfied: jinja2>=2.9 in
/usr/local/lib/python3.12/dist-packages (from folium) (3.1.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-
packages (from folium) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-
packages (from folium) (2.32.4)
Requirement already satisfied: xyzservices in
/usr/local/lib/python3.12/dist-packages (from folium) (2025.10.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2>=2.9->folium) (3.0.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->folium) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->folium) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->folium) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->folium)
(2025.10.5)
```

```
import pandas as pd
import numpy as np
```

R1 and R2 Universities

The Carnegie Classfication Database is downloaded from https://carnegieclassifications.acenet.edu/institutions/?inst=&research2025%5B%5D=1&research2025%5B%5D=2.

I have selected only the R1 and R2 universities.

```
# Load Excel file of the carnegie classification file
## downloaded from https://carnegieclassifications.acenet.edu/institutions
#r1r2_info = pd.read_excel("R1R2-info.xlsx")
r1r2_info = pd.read_csv("ace-institutional-classifications.csv", low_memor
```

```
df = r1r2_info

# Show all column names
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326 entries, 0 to 325
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   unitid                                  326 non-null    int64  
 1   name                                    326 non-null    object 
 2   city                                    326 non-null    object 
 3   state                                   326 non-null    object 
 4   control                                 326 non-null    object 
 5   Institutional Classification            326 non-null    object 
 6   Student Access and Earnings Classification  326 non-null    object 
 7   Research Activity Designation           326 non-null    object 
 8   Award Level Focus                       326 non-null    object 
 9   Academic Mix                            326 non-null    object 
 10  Graduate Academic Program Mix           326 non-null    object 
 11  Size                                    326 non-null    object 
 12  Campus Setting                          326 non-null    object 
 13  Highest Degree Awarded                  326 non-null    object 
 14  Community Engagement                    154 non-null    object 
 15  Leadership for Public Practice          11 non-null     object 
 16  326 results for  all categories         0 non-null      float64
dtypes: float64(1), int64(1), object(15)
memory usage: 43.4+ KB
```

```
df.head()
```

```
<div>
```

	unitid	name	city	state	control	Institutional Classification
0	222178	Abilene Christian University	Abilene	TX	Private not-for-profit	Professions-focused Undergraduate/Graduate-Doc…
1	200697	Air Force Institute of Technology-Graduate Sch…	Wright-Patterson AFB	OH	Public	Special Focus: Technology, Engineering, and Sc…
2	385415	Albert Einstein College of Medicine	Bronx	NY	Private not-for-profit	Special Focus: Medical Schools and Centers
3	131159	American University	Washington	DC	Private not-for-profit	Mixed Undergraduate/Graduate-Doctorate Medium
4	197869	Appalachian State University	Boone	NC	Public	Professions-focused Undergraduate/Graduate-Mas…

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
161b57c2-79e4-4d1d-bce9-25778b1cfca0')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">
```

```
<script>
  const buttonEl =
    document.querySelector('#df-161b57c2-79e4-4d1d-bce9-25778b1cfca0
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-161b57c2-79e4-4d1d-bce9-
25778b1cfca0');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                                [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-ef4c0f1e-d4d4-43cb-be59-eb23e0711c28">
  <button class="colab-df-quickchart" onclick="quickchart('df-ef4c0f1e-
d4d4-43cb-be59-eb23e0711c28')"
            title="Suggest charts"
            style="display:none;">



  </button>

  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-ef4c0f1e-d4d4-43cb-be59-eb23e0711c28
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>
```

Convert research2025name into only “R1” or “R2”

```
# Clean and normalize the text
df["research_clean"] = (
    df["Research Activity Designation"]
        .astype(str)
        .str.strip()                # remove leading/trailing spaces
        .str.normalize('NFKD')      # remove hidden unicode variations
        .str.replace(r'\s+', ' ', regex=True)    # force single spaces
)

print(df["research_clean"].unique())
```

```
['Research 2: High Research Spending and Doctorate Production'
 'Research 1: Very High Research Spending and Doctorate Production']
```

```
# Convert research2025name into only "R1" or "R2"

# Option 1: overwrite the existing column
df["Research Activity Designation"] = np.where(
    df["Research Activity Designation"].str.contains("Research 1", na=Fals
    "R1",
    np.where(
        df["Research Activity Designation"].str.contains("Research 2", na=
        "R2",
        None
    )
)

print(df["Research Activity Designation"].value_counts(dropna=False))
```

```
Research Activity Designation
R1    187
R2    139
Name: count, dtype: int64
```

```
# Clean and normalize the text
df["instnm_clean"] = (
    df["name"]
        .astype(str)
        .str.strip()                # remove leading/trailing spaces
        .str.normalize('NFKD')      # remove hidden unicode variations
        .str.replace(r'\s+', ' ', regex=True)    # force single spaces
)

df["name"] = df["instnm_clean"]
df = df.drop(columns=["instnm_clean"])    # optional: remove helper column

df.head()
```

<div>

	unitid	name	city	state	control	Institutional Classification
0	222178	Abilene Christian University	Abilene	TX	Private not-for-profit	Professions-focused Undergraduate/Graduate-Doc…
1	200697	Air Force Institute of Technology-Graduate Sch…	Wright-Patterson AFB	OH	Public	Special Focus: Technology, Engineering, and Sc…
2	385415	Albert Einstein College of Medicine	Bronx	NY	Private not-for-profit	Special Focus: Medical Schools and Centers
3	131159	American University	Washington	DC	Private not-for-profit	Mixed Undergraduate/Graduate-Doctorate Medium
4	197869	Appalachian State University	Boone	NC	Public	Professions-focused Undergraduate/Graduate-Mas…

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
934a1e33-077d-4b53-81a8-4e653fdde6ad')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">
```

```
<script>
  const buttonEl =
    document.querySelector('#df-934a1e33-077d-4b53-81a8-4e653fdde6ad
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-934a1e33-077d-4b53-81a8-
4e653fdde6ad');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                                [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-6a5bf71f-0f32-4bc5-96ef-381e663ee9bf">
  <button class="colab-df-quickchart" onclick="quickchart('df-6a5bf71f-
0f32-4bc5-96ef-381e663ee9bf')"
            title="Suggest charts"
            style="display:none;">



  </button>

  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-6a5bf71f-0f32-4bc5-96ef-381e663ee9bf
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>
```

```
r1r2 = df[['unitid','name', 'city', 'Research Activity Designation', 'stat
r1r2.head()
```

```
<div>
```

	unitid	name	city	Research Activity Designation	state
0	222178	Abilene Christian University	Abilene	R2	TX
1	200697	Air Force Institute of Technology-Graduate Sch...	Wright-Patterson AFB	R2	OH
2	385415	Albert Einstein College of Medicine	Bronx	R2	NY
3	131159	American University	Washington	R1	DC
4	197869	Appalachian State University	Boone	R2	NC

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
147efb3d-6be4-41ee-9295-668c4482a1fa')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">



<script>
  const buttonEl =
    document.querySelector('#df-147efb3d-6be4-41ee-9295-668c4482a1fa
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-147efb3d-6be4-41ee-9295-
668c4482a1fa');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                                [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-61628211-0dad-4428-81c1-6cbec1e84a95">
  <button class="colab-df-quickchart" onclick="quickchart('df-61628211-
0dad-4428-81c1-6cbec1e84a95')"
            title="Suggest charts"
            style="display:none;">



  </button>
  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-61628211-0dad-4428-81c1-6cbec1e84a95
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>
```

```
# keep only these columns
r1r2 = r1r2.rename(columns={"Research Activity Designation": "R1/R2", "nam
r1r2.head()
```

<div>

	unitid	institutes	city	R1/R2	states
0	222178	Abilene Christian University	Abilene	R2	TX
1	200697	Air Force Institute of Technology-Graduate Sch…	Wright-Patterson AFB	R2	OH
2	385415	Albert Einstein College of Medicine	Bronx	R2	NY
3	131159	American University	Washington	R1	DC
4	197869	Appalachian State University	Boone	R2	NC

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
2bae6009-b804-4d13-9af4-56a96009028b')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">
```

```
<script>
  const buttonEl =
    document.querySelector('#df-2bae6009-b804-4d13-9af4-56a96009028b
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-2bae6009-b804-4d13-9af4-
56a96009028b');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                                [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-34637f36-30d2-473b-897b-c19565c5a7fb">
  <button class="colab-df-quickchart" onclick="quickchart('df-34637f36-
30d2-473b-897b-c19565c5a7fb')"
            title="Suggest charts"
            style="display:none;">



  </button>

  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-34637f36-30d2-473b-897b-c19565c5a7fb
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>
```

US Institutes Info

The database of US universities with geo information is downloaded from US Dept. of

Education - https://ed-public-download.scorecard.network/downloads/Most-Recent-Cohorts-Institution_05192025.zip

```
df2 = pd.read_csv("inst-cohort.csv", low_memory=False)
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6429 entries, 0 to 6428
Columns: 3306 entries, UNITID to SCORECARD_SECTOR
dtypes: float64(920), int64(14), object(2372)
memory usage: 162.2+ MB
```

```
print(df2.columns)
```

```
Index(['UNITID', 'OPEID', 'OPEID6', 'INSTNM', 'CITY', 'STABBR', 'ZIP',
       'ACCREDAGENCY', 'INSTURL', 'NPCURL',
       ...
       'COUNT_WNE_MALE1_P11', 'GT_THRESHOLD_P11', 'MD_EARN_WNE_INC1_P11',
       'MD_EARN_WNE_INC2_P11', 'MD_EARN_WNE_INC3_P11',
       'MD_EARN_WNE_INDEP0_P11', 'MD_EARN_WNE_INDEP1_P11',
       'MD_EARN_WNE_MALE0_P11', 'MD_EARN_WNE_MALE1_P11',
'SCORECARD_SECTOR'],
      dtype='object', length=3306)
```

```
##### Find out relevant columns

# show columns that has the text private

cols_with_private = []

for col in df2.select_dtypes(include="object").columns:
    mask = df2[col].str.contains("private", case=False, na=False)
    if mask.any():
        cols_with_private.append(col)
        sample_value = df2.loc[mask, col].iloc[0]
        print(f"{col}: {sample_value}")

cols_with_private
```

```
NPCURL: https://www.sscc.edu/_private/npcalc.htm
CONTROL_PEPS: Private Nonprofit
```

```
['NPCURL', 'CONTROL_PEPS']
```

```
print(df2["CONTROL_PEPS"].head())
```

```
0               Public
1               Public
2    Private Nonprofit
3               Public
4               Public
Name: CONTROL_PEPS, dtype: object
```

```
# find any column containing lat
[col for col in df2.columns if "LAT" in col.upper()]
```

```
['LATITUDE']
```

```
df2['STABBR'].head()
```

	STABBR
0	AL
1	AL
2	AL
3	AL
4	AL

dtype: object

```
# find any column containing lon
[col for col in df2.columns if "LON" in col.upper()]
```

```
['LONGITUDE']
```

```
# keep only these columns
inst = df2[['UNITID','INSTNM', 'CITY', 'STABBR','LONGITUDE', 'LATITUDE', '
inst.head()
```

```
<div>
```

	UNITID	INSTNM	CITY	STABBR	LONGITUDE	LATITUDE	C
0	100654	Alabama A & M University	Normal	AL	-86.568502	34.783368	P
1	100663	University of Alabama at Birmingham	Birmingham	AL	-86.799345	33.505697	P
2	100690	Amridge University	Montgomery	AL	-86.174010	32.362609	P
3	100706	University of Alabama in Huntsville	Huntsville	AL	-86.640449	34.724557	P
4	100724	Alabama State University	Montgomery	AL	-86.295677	32.364317	P

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
a9dfcb62-f35f-495c-bd80-d02c4523069a')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">
```

```
<script>
  const buttonEl =
    document.querySelector('#df-a9dfcb62-f35f-495c-bd80-d02c4523069a
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-a9dfcb62-f35f-495c-bd80-
d02c4523069a');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                               [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-332a979a-66af-4a5f-98a8-b5d1109d50ae">
  <button class="colab-df-quickchart" onclick="quickchart('df-332a979a-
66af-4a5f-98a8-b5d1109d50ae')"
            title="Suggest charts"
            style="display:none;">



  </button>

  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-332a979a-66af-4a5f-98a8-b5d1109d50ae
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>

# rename columns
inst = inst.rename(columns ={"INSTNM": "INSTITUTES", "STABBR": "STATES_ABB
```

Merge the Institute Info and R1/R2 Data

```
inst.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6429 entries, 0 to 6428
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   UNITID          6429 non-null   int64
 1   INSTITUTES      6429 non-null   object
 2   CITY            6429 non-null   object
 3   STATES_ABB      6429 non-null   object
 4   LONGITUDE       5924 non-null   float64
 5   LATITUDE        5924 non-null   float64
 6   PUBLIC/PRIVATE  6405 non-null   object
dtypes: float64(2), int64(1), object(4)
memory usage: 351.7+ KB
```

```
r1r2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326 entries, 0 to 325
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   unitid      326 non-null    int64
 1   institutes  326 non-null    object
 2   city        326 non-null    object
 3   R1/R2       326 non-null    object
 4   states      326 non-null    object
dtypes: int64(1), object(4)
memory usage: 12.9+ KB
```

```
# Select required columns from inst
inst_sel = inst[[
    "UNITID", "INSTITUTES", "CITY", "STATES_ABB",
    "LONGITUDE", "LATITUDE", "PUBLIC/PRIVATE"
]].drop_duplicates(subset="UNITID")

# Merge using UNITID
merged = r1r2.merge(
    inst_sel,
    left_on="unitid",
    right_on="UNITID",
    how="left"
)

# Keep only the needed columns, in the required order

merged = merged[[
    "UNITID",        # from inst
    "INSTITUTES",
    "CITY",
    "states",        # from r1r2
    "STATES_ABB",
    "R1/R2",         # from r1r2
    "PUBLIC/PRIVATE",
    "LONGITUDE",
    "LATITUDE"
]]

# remove "the" from the university names

merged["INSTITUTES"] = (
    merged["INSTITUTES"]
        .str.replace(r"^the\s+", "", case=False, regex=True)
        .str.strip()
)
```

CS Ranking Data

Computer Science open rankings compiled by Brown University is collected in a csv file.

https://drafty.cs.brown.edu/csopenrankings/

```
# Load the file
cs = pd.read_csv("csbrownrank.csv", low_memory=False)

cs["university"].head()
```

	university
0	Carnegie Mellon University+
1	Massachusetts Institute of Technology+
2	University of California, Berkeley+
3	Stanford University+
4	University of Illinois at Urbana-Champaign+

dtype: object

```
cs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230 entries, 0 to 229
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   #                  230 non-null    int64  
 1   university         230 non-null    object 
 2   size               124 non-null    float64
 3   U.S. News          230 non-null    int64  
 4   csrankings.org     201 non-null    float64
 5   placement rank     191 non-null    float64
 6   best paper awards  121 non-null    float64
 7   total              230 non-null    int64  
dtypes: float64(4), int64(3), object(1)
memory usage: 14.5+ KB
```

```
# Rename "#" to "rank"
cs = cs.rename(columns={"#": "rank"})

# Clean the university names (strip spaces, quotes, special chars)
cs["university"] = (
    cs["university"]
    .astype(str)
    .str.strip()  # # remove leading/trailing spaces
    .str.replace(r"[^\w\s.&-]", "", regex=True)   # remove strange charact
    .str.replace(r"^the\s+", "", case=False, regex=True)
)
```

Merge those matched

```
# Check how many names match
matches = cs["university"].isin(merged["INSTITUTES"]).sum()

print("Total CS universities:", len(cs))
print("Total matches with R1/R2 list:", matches)
```

```
Total CS universities: 230
Total matches with R1/R2 list: 138
```

```
## Merge 'merged' (R1/R2 table) with CS rank

merged_cs = merged.merge(
    cs[["university", "rank"]],
    left_on="INSTITUTES",
    right_on="university",
    how="left"
)
```

```
merged_cs["rank"].notna().sum()
```

```
np.int64(138)
```

```
merged_cs.to_excel("final-merged.xlsx", index=False) ## match this with th
```

Check unmatched data

```
# Identify unmatched rows from merged_cs

non_matched = cs[~cs["university"].isin(merged["INSTITUTES"])]

print("Total universities in CS ranking:", len(cs))
print("Matched:", cs["university"].isin(merged["INSTITUTES"]).sum())
print("Not matched:", len(non_matched))
```

```
Total universities in CS ranking: 230
Matched: 138
Not matched: 92
```

```
non_matched.to_excel("cs_non_matched.xlsx", index=False)
```

Merge the fixed data

The cs_non_matched.xlsx file has been updated with corrected university names that match those in final-merged.xlsx. To be noted that some universities in the CS ranking are not really R1 or R2 institutions.

```
cs_fixed = pd.read_excel("cs_unmatched.xlsx")
```

```
# Merge only to pull the corrected ranks

temp = merged_cs.merge(
    cs_fixed[["university", "rank"]],
    left_on="INSTITUTES",
    right_on="university",
    how="left"
)

# Update the existing rank column
temp["rank"] = temp["rank_x"].fillna(temp["rank_y"])

# remove helper columns
final = temp.drop(columns=["rank_x", "rank_y", "university_x", "university
```

```
# write the NaN as None
final["rank"] = final["rank"].fillna("N/A")


final.head()
```

```
<div>
```

	UNITID	INSTITUTES	CITY	states	STATES_ABB	R1/R2	PUBLIC
0	222178	Abilene Christian University	Abilene	TX	TX	R2	Private N
1	200697	Air Force Institute of Technology-Graduate Sch...	Wright-Patterson AFB	OH	OH	R2	Public
2	385415	Albert Einstein College of Medicine	Bronx	NY	NY	R2	Private N
3	131159	American University	Washington	DC	DC	R1	Private N
4	197869	Appalachian State University	Boone	NC	NC	R2	Public

```
<div class="colab-df-buttons">

<button class="colab-df-convert" onclick="convertToInteractive('df-
8e72f19a-7d16-4a89-ba42-300cceeaa677')"
        title="Convert this dataframe to an interactive table."
        style="display:none;">



<script>
  const buttonEl =
    document.querySelector('#df-8e72f19a-7d16-4a89-ba42-300cceeaa677
button.colab-df-convert');
  buttonEl.style.display =
    google.colab.kernel.accessAllowed ? 'block' : 'none';

  async function convertToInteractive(key) {
    const element = document.querySelector('#df-8e72f19a-7d16-4a89-ba42-
300cceeaa677');
    const dataTable =
      await google.colab.kernel.invokeFunction('convertToInteractive',
                                                [key], {});
    if (!dataTable) return;

    const docLinkHtml = 'Like what you see? Visit the ' +
      '<a target="_blank"
href=https://colab.research.google.com/notebooks/data_table.ipynb>data
table notebook</a>'
      + ' to learn more about interactive tables.';
    element.innerHTML = '';
    dataTable['output_type'] = 'display_data';
    await google.colab.output.renderOutput(dataTable, element);
    const docLink = document.createElement('div');
    docLink.innerHTML = docLinkHtml;
    element.appendChild(docLink);
  }
</script>

<div id="df-b876f208-e341-4377-aec8-ed7c25f75d63">
  <button class="colab-df-quickchart" onclick="quickchart('df-b876f208-
e341-4377-aec8-ed7c25f75d63')"
            title="Suggest charts"
            style="display:none;">
  </button>

  <script>
    async function quickchart(key) {
      const quickchartButtonEl =
        document.querySelector('#' + key + ' button');
      quickchartButtonEl.disabled = true;  // To prevent multiple clicks.
      quickchartButtonEl.classList.add('colab-df-spinner');
      try {
        const charts = await google.colab.kernel.invokeFunction(
            'suggestCharts', [key], {});
      } catch (error) {
        console.error('Error during call to suggestCharts:', error);
      }
      quickchartButtonEl.classList.remove('colab-df-spinner');
      quickchartButtonEl.classList.add('colab-df-quickchart-complete');
    }
    (() => {
      let quickchartButtonEl =
        document.querySelector('#df-b876f208-e341-4377-aec8-ed7c25f75d63
button');
      quickchartButtonEl.style.display =
        google.colab.kernel.accessAllowed ? 'block' : 'none';
    })();
  </script>
</div>

</div>
```

```python
# check how many updates

before = merged_cs["rank"].notna().sum()
after  = final["rank"].notna().sum()

print("Before:", before)
print("After :", after)
print("Newly filled ranks:", after - before)
```

```
Before: 138
After : 326
Newly filled ranks: 188
```

```python
# final.to_excel("final_with_fixed_ranks.xlsx", index=False)
```

still missing rank

```python
# Filter rows where rank is missing
missing_rank = final[final["rank"].isna()].copy()

# Count how many R1 and R2 have no ranking
missing_counts = missing_rank["R1/R2"].value_counts()

print("Missing R1 rankings:", missing_counts.get("R1", 0))
print("Missing R2 rankings:", missing_counts.get("R2", 0))
```

```
Missing R1 rankings: 21
Missing R2 rankings: 106
```

```python
# # 3. Save to Excel
# missing_rank.to_excel("missing_rank_universities.xlsx", index=False)
```

Find Closest Universities

```python
from geopy.distance import geodesic
import itertools
import pandas as pd
```

```python
# Keep valid coordinates
df_coords = final.dropna(subset=["LATITUDE", "LONGITUDE"]).reset_index(dro

pairs = []

for (i1, row1), (i2, row2) in itertools.combinations(df_coords.iterrows()

    dist_km = geodesic(
        (row1["LATITUDE"], row1["LONGITUDE"]),
        (row2["LATITUDE"], row2["LONGITUDE"])
    ).km

    pairs.append([
        row1["UNITID"], row1["INSTITUTES"], row1["PUBLIC/PRIVATE"], row1["
        row2["UNITID"], row2["INSTITUTES"], row2["PUBLIC/PRIVATE"], row2["
        dist_km
    ])

# Create dataframe with proper column names
distance_df = pd.DataFrame(
    pairs,
    columns=[
        "UNITID_1", "University_1", "Type_1", "City_1", "State_1", "R_Type
        "UNITID_2", "University_2", "Type_2", "City_2", "State_2", "R_Type
        "Distance_km"
    ]
)

# Sort and filter
dist_50 = (
    distance_df
        .sort_values(by="Distance_km")
        .query("Distance_km <= 50")
        .reset_index(drop=True)
)
```

```python
# Save output
dist_50.to_csv("close-universities.csv", index=False)

dist_50.head()
```

Cluster Map

```python
from folium.plugins import MarkerCluster, HeatMap
import folium
import numpy as np

# Prepare data

df_geo = final.dropna(subset=["LATITUDE", "LONGITUDE"]).copy()

# For color gradient: normalize rank (lower = better rank)
# Missing rank □ mid-gray
ranks = df_geo["rank"]
rank_norm = (ranks.max() - ranks) / (ranks.max() - ranks.min())
df_geo["rank_norm"] = rank_norm.fillna(0.5)   # midpoint color

# convert normalized rank to color

def rank_to_color(x):
    # green □ yellow □ red
    r = int(255 * x)
    g = int(255 * (1 - x))
    b = 60
    return f"#{r:02x}{g:02x}{b:02x}"
```

```python
# create map

m = folium.Map(location=[39.5, -98.35], zoom_start=4)

cluster = MarkerCluster().add_to(m)

# color for R1 and R2

def rtype_color(rtype):
    if rtype == "R1":
        return "#1f77b4"   # blue
    elif rtype == "R2":
        return "#d62728"   # red
    else:
        return "#888888"   # fallback

final.info()

# plot markers

for _, row in df_geo.iterrows():
    color = rtype_color(row["R1/R2"])

    folium.CircleMarker(
        location=[row["LATITUDE"], row["LONGITUDE"]],
        radius=6,
        color=color,
        fill=True,
        fill_opacity=0.9,
        popup=(
            f"<b>{row['INSTITUTES']}</b><br>"
            f"State: {row['states']}<br>"
            f"Type: {row['PUBLIC/PRIVATE']}<br>"
            f"R-Type: {row['R1/R2']}<br>"
            f"CS Rank: {row['rank']}"
        )
    ).add_to(cluster)

# Add Heatmap (for density)

heat_data = df_geo[["LATITUDE", "LONGITUDE"]].values.tolist()
HeatMap(heat_data, radius=18, blur=12).add_to(m)

coord_lookup = final.set_index("UNITID")[["LATITUDE", "LONGITUDE"]].to_dic
```

```python
## show distances

for _, row in dist_50.iterrows():
    lat1 = coord_lookup[row["UNITID_1"]]["LATITUDE"]
    lon1 = coord_lookup[row["UNITID_1"]]["LONGITUDE"]
    lat2 = coord_lookup[row["UNITID_2"]]["LATITUDE"]
    lon2 = coord_lookup[row["UNITID_2"]]["LONGITUDE"]

    folium.PolyLine(
        [(lat1, lon1), (lat2, lon2)],
        color="blue",
        weight=2,
        tooltip=f"{row['Distance_km']:.1f} km"
    ).add_to(m)

# show map

m
```

Layering R1 and R2

```python
# Create layer groups
layer_r1 = folium.FeatureGroup(name="R1 Universities")
layer_r2 = folium.FeatureGroup(name="R2 Universities")
# layer_other = folium.FeatureGroup(name="Others (No R-Type)")

# Add markers to the correct layer

for _, row in df_geo.iterrows():
    color = rtype_color(row["R1/R2"])
    popup_html = (
        f"<b>{row['INSTITUTES']}</b><br>"
        f"State: {row['states']}<br>"
        f"Type: {row['PUBLIC/PRIVATE']}<br>"
        f"R-Type: {row['R1/R2']}<br>"
        f"CS Rank: {row['rank']}"
    )

    marker = folium.CircleMarker(
        location=[row["LATITUDE"], row["LONGITUDE"]],
        radius=6,
        color=color,
        fill=True,
        fill_opacity=0.9,
        popup=popup_html
    )

    # assign markers to layers
    if row["R1/R2"] == "R1":
        marker.add_to(layer_r1)
    elif row["R1/R2"] == "R2":
        marker.add_to(layer_r2)
    # else:
    #     marker.add_to(layer_other)

# Add layers to the map
layer_r1.add_to(m)
layer_r2.add_to(m)
# layer_other.add_to(m)

# Add layer control
folium.LayerControl().add_to(m)

m.save("university_map.html")
```

Streamlit App for the Map

```python
# Save output
final.to_csv("final-with-ranks.csv", index=False)

!pip install streamlit pydeck geopy
```

```
Collecting streamlit
  Using cached streamlit-1.51.0-py3-none-any.whl.metadata (9.5 kB)
Collecting pydeck
  Using cached pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: geopy in /usr/local/lib/python3.12/dist-
packages (2.4.1)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<6,>=4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (5.5.2)
Requirement already satisfied: click<9,>=7.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (8.3.0)
Requirement already satisfied: numpy<3,>=1.23 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.0.2)
Requirement already satisfied: packaging<26,>=20 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (25.0)
```

```
/usr/local/lib/python3.12/dist-packages (from streamlit) (25.0)
Requirement already satisfied: pandas<3,>=1.4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<13,>=7.1.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (11.3.0)
Requirement already satisfied: protobuf<7,>=3.20 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (5.29.5)
Requirement already satisfied: pyarrow<22,>=7.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (18.1.0)
Requirement already satisfied: requests<3,>=2.27 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (2.32.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (8.5.0)
Requirement already satisfied: toml<2,>=0.10.1 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (4.15.0)
Requirement already satisfied: watchdog<7,>=2.1.5 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (6.0.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (3.1.45)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in
/usr/local/lib/python3.12/dist-packages (from streamlit) (6.5.1)
Requirement already satisfied: jinja2>=2.10.1 in
/usr/local/lib/python3.12/dist-packages (from pydeck) (3.1.6)
Requirement already satisfied: geographiclib<3,>=1.52 in
/usr/local/lib/python3.12/dist-packages (from geopy) (2.1)
Requirement already satisfied: jsonschema>=3.0 in
/usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,
<6,>=4.0->streamlit) (4.25.1)
Requirement already satisfied: narwhals>=1.14.2 in
/usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,
<6,>=4.0->streamlit) (2.11.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.12/dist-packages (from gitpython!=3.1.19,
<4,>=3.0.7->streamlit) (4.0.12)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2>=2.10.1->pydeck)
(3.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->streamlit)
(2025.2)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27-
>streamlit) (2025.10.5)
Requirement already satisfied: smmap<6,>=3.0.1 in
/usr/local/lib/python3.12/dist-packages (from gitdb<5,>=4.0.1-
>gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.2)
Requirement already satisfied: attrs>=22.2.0 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (25.4.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (2025.9.1)
Requirement already satisfied: referencing>=0.28.4 in
```

```
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (0.37.0)
Requirement already satisfied: rpds-py>=0.7.1 in
/usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0-
>altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (0.28.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-
packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit)
(1.17.0)
Downloading streamlit-1.51.0-py3-none-any.whl (10.2 MB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m10.2/10.2 MB[0m
[31m111.4 MB/s[0m eta [36m0:00:00[0m
[?25hDownloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m6.9/6.9 MB[0m
[31m113.8 MB/s[0m eta [36m0:00:00[0m
[?25hInstalling collected packages: pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.51.0
```

```python
# %%writefile app.py
# import streamlit as st
# import pandas as pd
# import pydeck as pdk
# from geopy.distance import geodesic
# import itertools

# # -------------------------------------------------
# # Load your final dataset
# # -------------------------------------------------
# final = pd.read_csv("final_with_fixed_ranks.csv")

# # Keep only valid coordinates
# df_geo = final.dropna(subset=["LATITUDE", "LONGITUDE"]).copy()

# # -------------------------------------------------
# # Streamlit UI
# # -------------------------------------------------
# st.title("R1 & R2 University Map (with Distance Filter)")
# st.write("Interactive map showing R1/R2 universities within selected dis

# # Sidebar controls
# st.sidebar.header("Filters")

# show_r1 = st.sidebar.checkbox("Show R1 Universities", True)
# show_r2 = st.sidebar.checkbox("Show R2 Universities", True)
# show_other = st.sidebar.checkbox("Show Other Universities", True)

# distance_limit = st.sidebar.slider(
#     "Show close university pairs within (km):",
#     min_value=0,
#     max_value=50,
#     value=50,
#     step=5
# )

# # -------------------------------------------------
# # Filter dataset based on user selection
# # -------------------------------------------------
# filtered = pd.DataFrame()

# if show_r1:
#     filtered = pd.concat([filtered, df_geo[df_geo["R1/R2"] == "R1"]])

# if show_r2:
#     filtered = pd.concat([filtered, df_geo[df_geo["R1/R2"] == "R2"]])

# if show_other:
#     filtered = pd.concat([filtered, df_geo[~df_geo["R1/R2"].isin(["R1",

# filtered = filtered.drop_duplicates()

# #
# # --------------------------------------------------
# # Compute distance pairs dynamically
# # --------------------------------------------------
# pairs = []
# for (_, row1), (_, row2) in itertools.combinations(filtered.iterrows(),

#     dist = geodesic(
#         (row1["LATITUDE"], row1["LONGITUDE"]),
#         (row2["LATITUDE"], row2["LONGITUDE"])
#     ).km

#     if dist <= distance_limit:
#         pairs.append({
#             "lat1": row1["LATITUDE"],
#             "lon1": row1["LONGITUDE"],
#             "lat2": row2["LATITUDE"],
#             "lon2": row2["LONGITUDE"],
#             "distance": dist
#         })

# pair_df = pd.DataFrame(pairs)

# # --------------------------------------------------
# # Build PyDeck layers
# # --------------------------------------------------

# # Marker Layer
# marker_layer = pdk.Layer(
#     "ScatterplotLayer",
#     data=filtered,
#     get_position='[LONGITUDE, LATITUDE]',
#     get_radius=6000,
#     get_color="""
#         [
#             R1_R2 == 'R1' ? 30 : R1_R2 == 'R2' ? 200 : 150,
#             R1_R2 == 'R1' ? 90 : R1_R2 == 'R2' ? 30  : 150,
#             200
#         ]
#     """,
#     pickable=True
# )

# # Distance line layer
# line_layer = pdk.Layer(
#     "LineLayer",
#     data=pair_df,
#     get_source_position='[lon1, lat1]',
#     get_target_position='[lon2, lat2]',
#     get_color='[0, 100, 255]',
#     get_width=2,
# )

# # View state
# view_state = pdk.ViewState(
#     latitude=39.5,
#     longitude=-98.35,
#     zoom=4,
#     pitch=0
# )

# # Tooltip
# tooltip = {
#     "html": "<b>{INSTITUTES}</b><br/>"
#             "State: {states}<br/>"
#             "Type: {PUBLIC/PRIVATE}<br/>"
#             "R-Type: {R1/R2}<br/>"
#             "Rank: {rank}",
#     "style": {"backgroundColor": "rgba(255,255,255,0.8)", "font-size": "
# }
# # Render map
# r = pdk.Deck(
#     layers=[marker_layer, line_layer],
#     initial_view_state=view_state,
#     tooltip=tooltip,
#     map_style="mapbox://styles/mapbox/light-v9"
# )

# st.pydeck_chart(r)

# # --------------------------------------------------
# # Show Data Table (optional)
# # --------------------------------------------------
# with st.expander("Show filtered university list"):
#     st.dataframe(filtered)
```

```
Writing app.py
```

```
# ! streamlit run app.py
```

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats
to false.
[0m
[0m
[34m[1m  You can now view your Streamlit app in your browser.[0m
[0m
[34m  Local URL: [0m[1mhttp://localhost:8501[0m
[34m  Network URL: [0m[1mhttp://172.28.0.12:8501[0m
[34m  External URL: [0m[1mhttp://136.107.37.42:8501[0m
[0m
[34m  Stopping...[0m
[34m  Stopping...[0m
```