02_eda_pubmed

March 16, 2025

[13]:
```
import pandas as pd
pubmed = pd.read_csv('data/all_pubmed.csv') # make sure pubmed has a column␣
  ↪named 'category'
print(pubmed.columns)
# pubmed.head()
```

```
Index(['pubmed_id', 'title', 'keywords', 'journal', 'abstract', 'methods',
       'results', 'conclusions', 'publication_date', 'category'],
      dtype='object')
```

[14]:
```
pubmed['category'] = (
    pubmed['category']
    .str.replace("Pubmed_", "", regex=False)
    .str.replace(".csv", "", regex=False)
    .str.replace("_", " ", regex=False)
    .str.replace("-", " ")  # Optional: Replace hyphens with spaces if needed
    .str.title()  # Capitalize each word
)
pubmed['category'].value_counts()
```

[14]:
```
category
Pancreatic Cancer              9891
Influenza                      8905
Hepatitis                      7087
Malaria                        6855
Endometriosis                  2839
Duchenne Muscular Dystrophy    1423
Drug Resistant Tuberculosis    1274
Chagas Disease                  740
Breast Cancer                   171
Alzheimer                        60
Name: count, dtype: int64
```

for rare diseases

[15]:
```
pubmed = pubmed[pubmed['category'].isin(['Pancreatic Cancer', 'Endometriosis',␣
  ↪'Chagas Disease','Drug Resistant Tuberculosis', 'Duchenne Muscular␣
  ↪Dystrophy'])]
```

[18]:

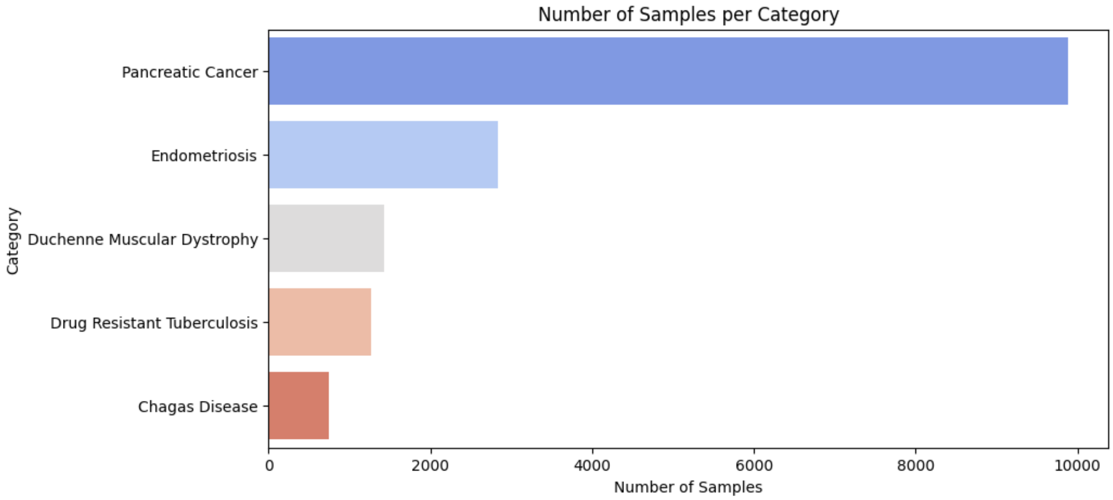
```
import pandas as pd


# Convert publication_date to datetime
pubmed['publication_date'] = pd.to_datetime(pubmed['publication_date'],␣
 ↪errors="coerce")

### 1) Number of samples per category
category_counts = pubmed['category'].value_counts()
# print("Number of samples per category:")
# print(category_counts)
print("\n")

### 2) Top 10 journal venues per category
top_journals = pubmed.groupby('category')['journal'].value_counts().
 ↪groupby(level=0).head(10)
# print("Top 10 journal venues per category:")
# print(top_journals)
print("\n")

### 3) Earliest, latest publication date, and elapsed days per category
date_range = pubmed.groupby('category')['publication_date'].agg(['min', 'max'])
date_range['elapsed_days'] = (date_range['max'] - date_range['min']).dt.days  #␣
 ↪Compute elapsed days
# print("Earliest, latest publication date, and elapsed days per category:")
# print(date_range)
```

[17]:

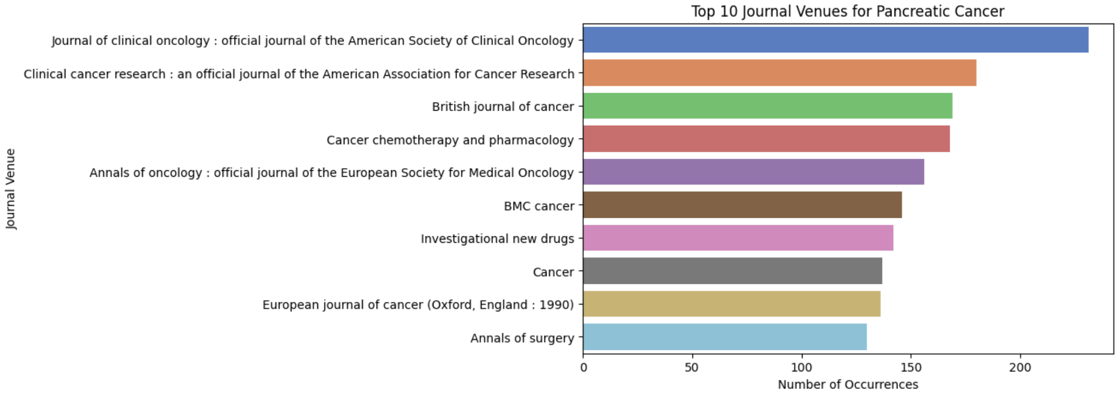
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd


# Convert publication_date to datetime
pubmed['publication_date'] = pd.to_datetime(pubmed['publication_date'])

# 1) Number of Samples per Category
plt.figure(figsize=(10, 5))
sns.countplot(y=pubmed['category'], palette="coolwarm",␣
 ↪order=pubmed['category'].value_counts().index)
plt.xlabel("Number of Samples")
plt.ylabel("Category")
plt.title("Number of Samples per Category")
```

```python
plt.show()

categories = pubmed['category'].unique()

for category in categories:
    plt.figure(figsize=(8, 5))
    category_data = pubmed[pubmed['category'] == category]['journal'].value_counts().head(10)

    sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
    plt.xlabel("Number of Occurrences")
    plt.ylabel("Journal Venue")
    plt.title(f"Top 10 Journal Venues for {category}")
    plt.show()

# 3) Earliest, Latest Publication Date, and Elapsed Days per Category
date_range = pubmed.groupby('category')['publication_date'].agg(['min', 'max']).reset_index()
date_range['elapsed_days'] = (date_range['max'] - date_range['min']).dt.days

plt.figure(figsize=(12, 6))
sns.barplot(data=date_range, y="category", x="elapsed_days", palette="viridis")
plt.xlabel("Elapsed Days (Difference Between Earliest & Latest Publication)")
plt.ylabel("Category")
plt.title("Publication Date Range & Elapsed Days per Category")
plt.show()
```

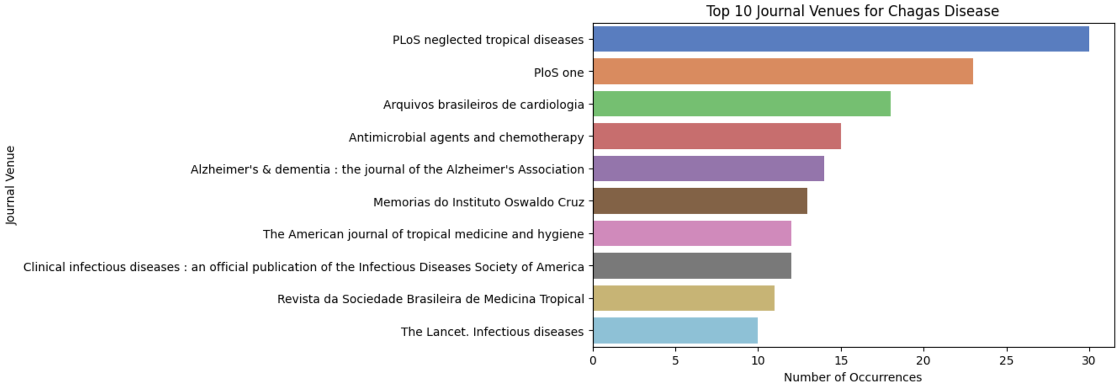
```
/tmp/ipykernel_16005/2142337901.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(y=pubmed['category'], palette="coolwarm",
order=pubmed['category'].value_counts().index)
```



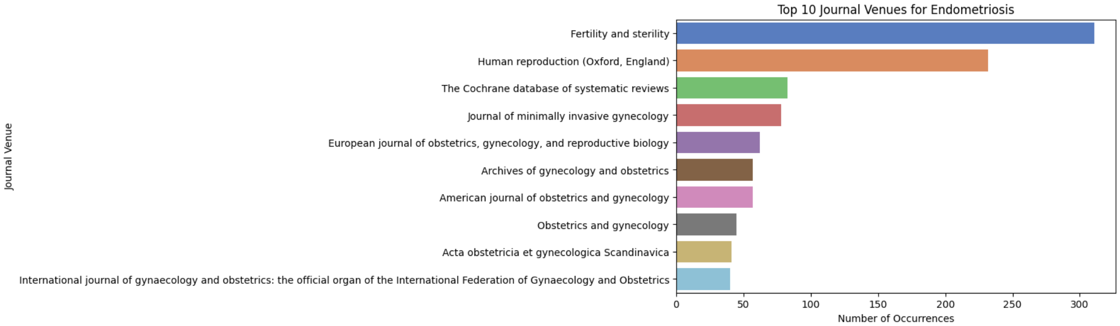
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



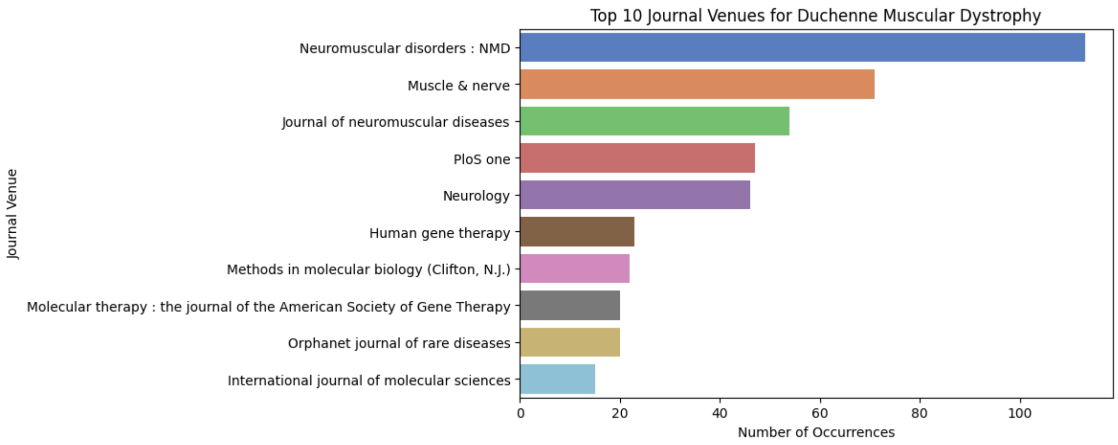
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



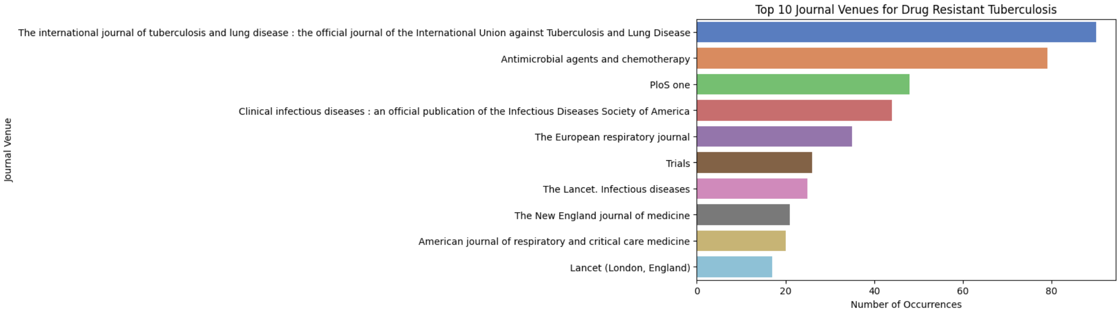
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



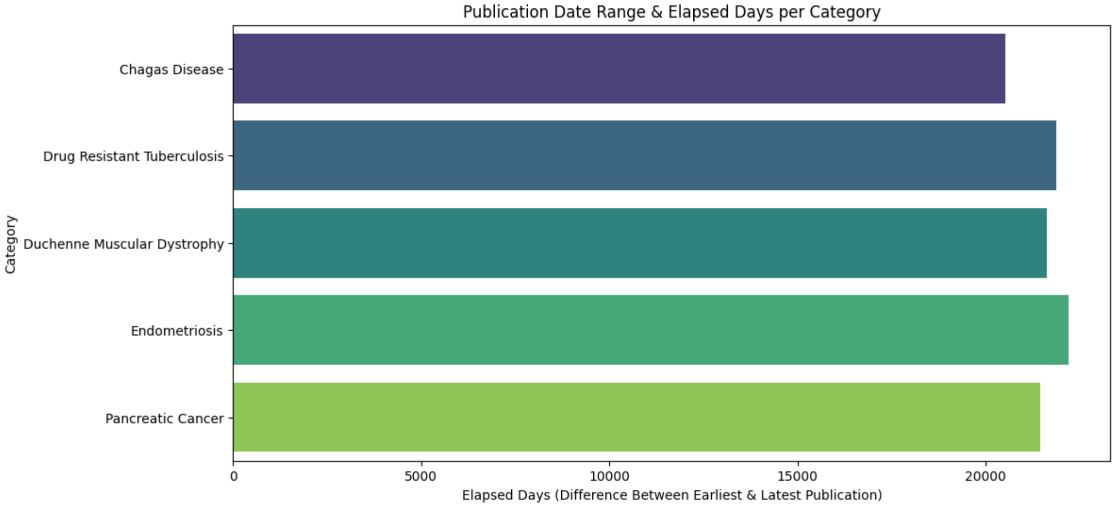
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



```
/tmp/ipykernel_16005/2142337901.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(data=date_range, y="category", x="elapsed_days",
palette="viridis")
```



1 For Common Disease

[8]:
```
import pandas as pd
pubmed = pd.read_csv('data/all_pubmed.csv') # make sure pubmed has a column␣
  ↪named 'category'
print(pubmed.columns)
# pubmed.head()
```

```
Index(['pubmed_id', 'title', 'keywords', 'journal', 'abstract', 'methods',
       'results', 'conclusions', 'publication_date', 'category'],
      dtype='object')
```

[9]:
```
pubmed['category'] = (
    pubmed['category']
    .str.replace("Pubmed_", "", regex=False)
    .str.replace(".csv", "", regex=False)
    .str.replace("_", " ", regex=False)
    .str.replace("-", " ")  # Optional: Replace hyphens with spaces if needed
    .str.title()  # Capitalize each word
)
pubmed['category'].value_counts()
```

[9]:
```
category
Pancreatic Cancer              9891
Influenza                      8905
Hepatitis                      7087
Malaria                        6855
Endometriosis                  2839
Duchenne Muscular Dystrophy    1423
```

```
Drug Resistant Tuberculosis    1274
Chagas Disease                  740
Breast Cancer                   171
Alzheimer                        60
Name: count, dtype: int64
```

[10]:

```
pubmed = pubmed[~pubmed['category'].isin(['Pancreatic Cancer', 'Endometriosis',␣
  ↪'Chagas Disease', 'Drug Resistant Tuberculosis', 'Duchenne Muscular␣
  ↪Dystrophy'])]
```

[11]:

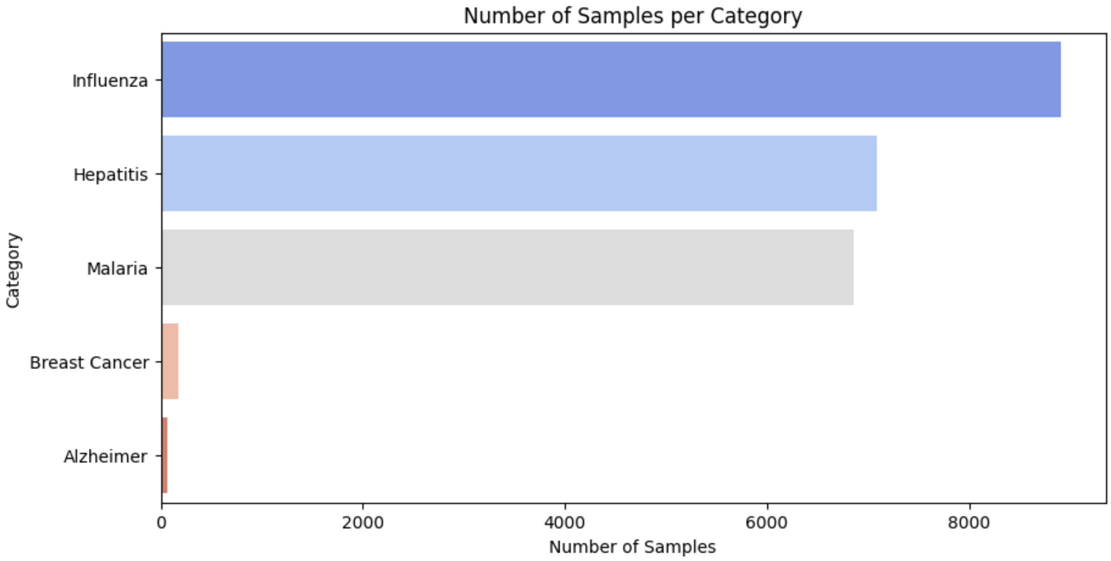
```
import pandas as pd


# Convert publication_date to datetime
pubmed['publication_date'] = pd.to_datetime(pubmed['publication_date'],␣
  ↪errors="coerce")

### 1) Number of samples per category
category_counts = pubmed['category'].value_counts()
# print("Number of samples per category:")
# print(category_counts)
print("\n")

### 2) Top 10 journal venues per category
top_journals = pubmed.groupby('category')['journal'].value_counts().
  ↪groupby(level=0).head(10)
# print("Top 10 journal venues per category:")
# print(top_journals)
print("\n")

### 3) Earliest, latest publication date, and elapsed days per category
date_range = pubmed.groupby('category')['publication_date'].agg(['min', 'max'])
date_range['elapsed_days'] = (date_range['max'] - date_range['min']).dt.days  #␣
  ↪Compute elapsed days
# print("Earliest, latest publication date, and elapsed days per category:")
# print(date_range)
```

[12]:

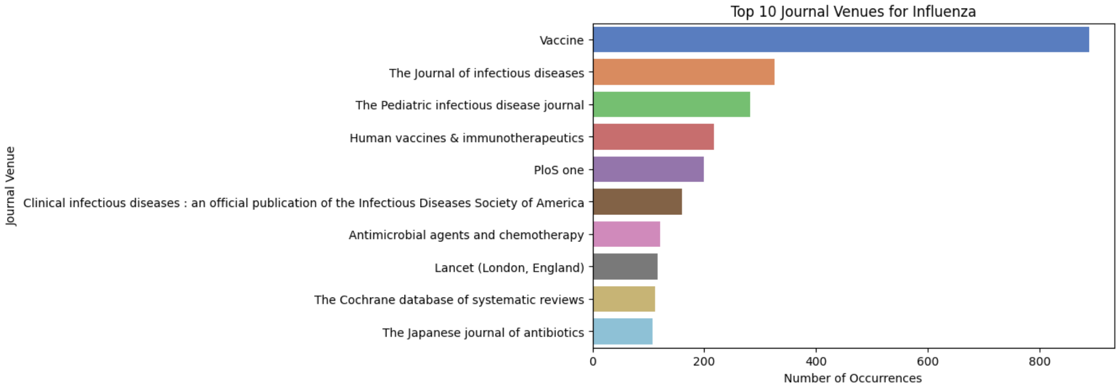
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```python
# Convert publication_date to datetime
pubmed['publication_date'] = pd.to_datetime(pubmed['publication_date'])

# 1) Number of Samples per Category
plt.figure(figsize=(10, 5))
sns.countplot(y=pubmed['category'], palette="coolwarm",␣
 ↪order=pubmed['category'].value_counts().index)
plt.xlabel("Number of Samples")
plt.ylabel("Category")
plt.title("Number of Samples per Category")
plt.show()

categories = pubmed['category'].unique()

for category in categories:
    plt.figure(figsize=(8, 5))
    category_data = pubmed[pubmed['category'] == category]['journal'].
 ↪value_counts().head(10)

    sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
    plt.xlabel("Number of Occurrences")
    plt.ylabel("Journal Venue")
    plt.title(f"Top 10 Journal Venues for {category}")
    plt.show()

# 3) Earliest, Latest Publication Date, and Elapsed Days per Category
date_range = pubmed.groupby('category')['publication_date'].agg(['min', 'max']).
 ↪reset_index()
date_range['elapsed_days'] = (date_range['max'] - date_range['min']).dt.days

plt.figure(figsize=(12, 6))
sns.barplot(data=date_range, y="category", x="elapsed_days", palette="viridis")
plt.xlabel("Elapsed Days (Difference Between Earliest & Latest Publication)")
plt.ylabel("Category")
plt.title("Publication Date Range & Elapsed Days per Category")
plt.show()
```

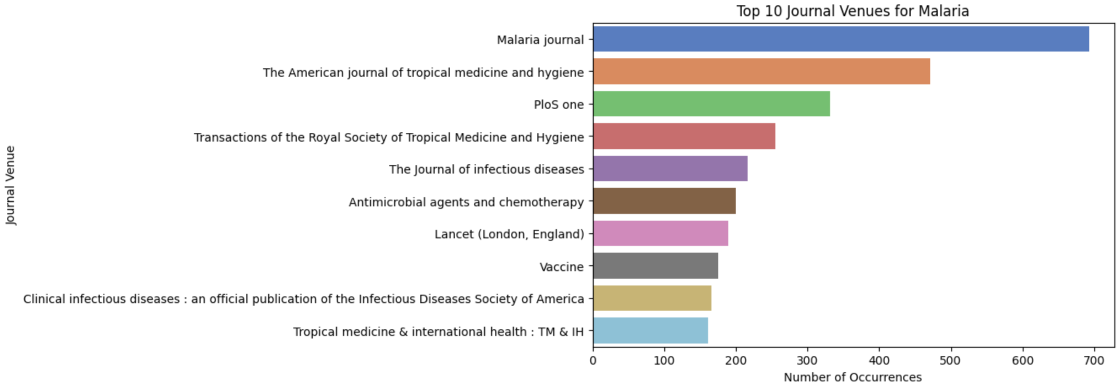
```
/tmp/ipykernel_16005/2142337901.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(y=pubmed['category'], palette="coolwarm",
order=pubmed['category'].value_counts().index)
```



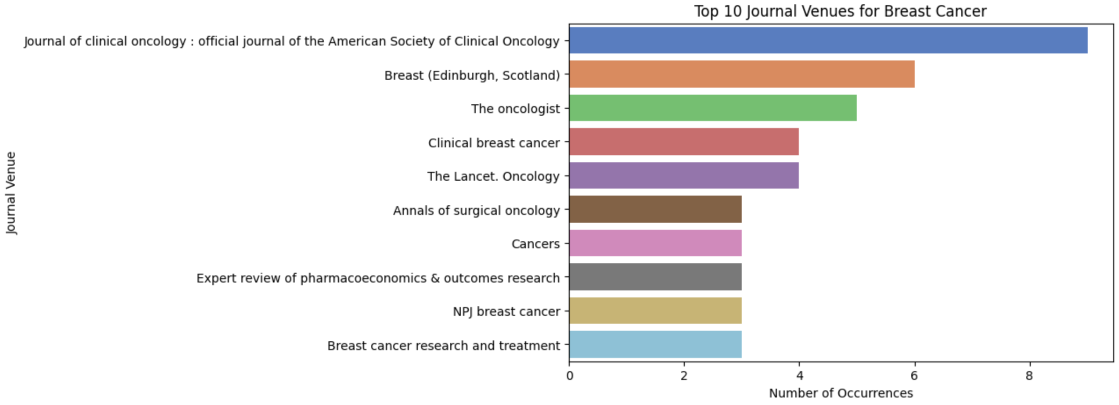
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



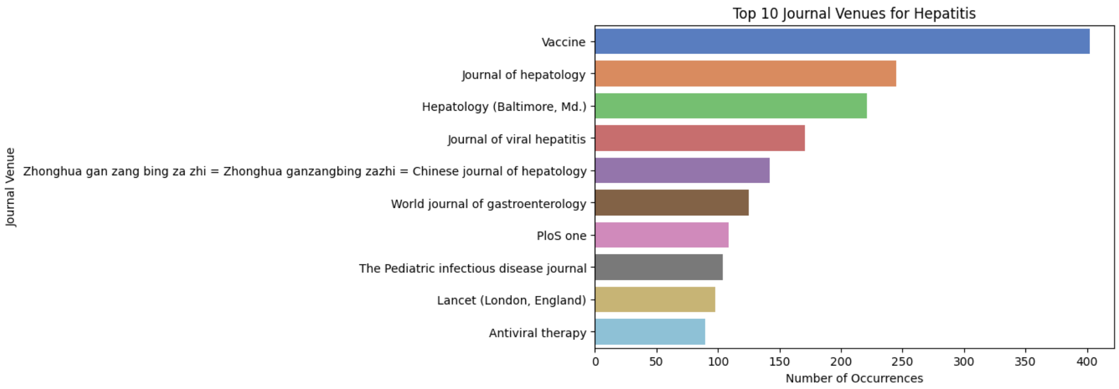
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



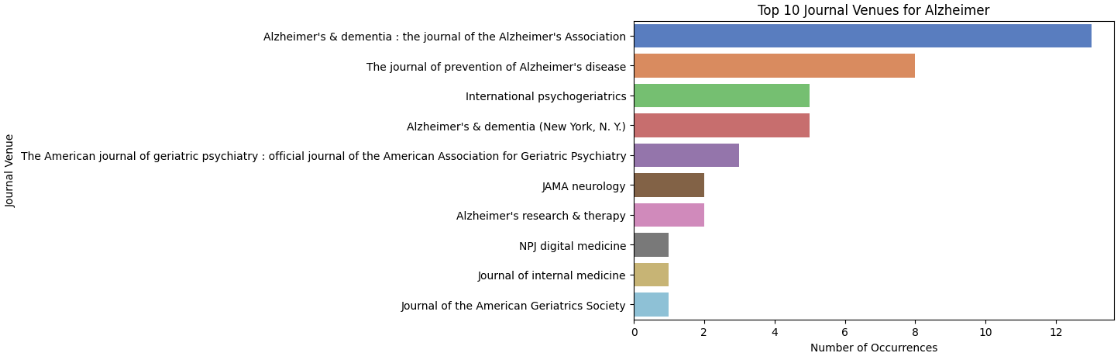
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



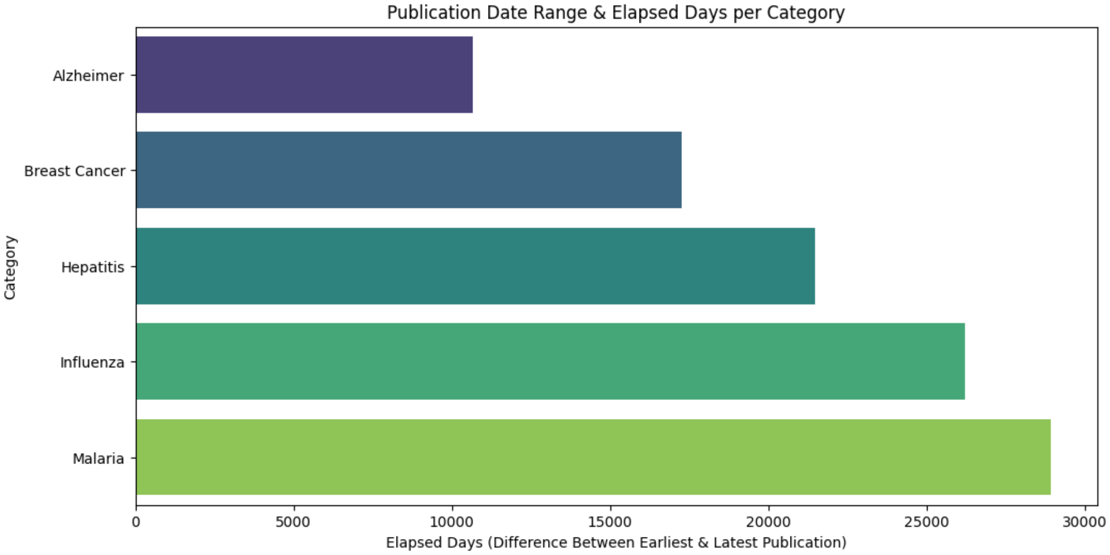
```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



```
/tmp/ipykernel_16005/2142337901.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=category_data.index, x=category_data.values, palette="muted")
```



```
/tmp/ipykernel_16005/2142337901.py:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(data=date_range, y="category", x="elapsed_days",
palette="viridis")
```


Publication Date Range & Elapsed Days per Category
Category
Alzheimer
Breast Cancer
Hepatitis
Influenza
Malaria
0
5000
10000
15000
20000
25000
30000
Elapsed Days (Difference Between Earliest & Latest Publication)