```
# specify directory where bamcmp output txt files are located
directory = "/path/to/directory"
```

```
import pandas as pd
import glob
import os

# Get a list of all .txt files in the directory
txt_files = glob.glob(os.path.join(directory, "*.txt"))

# Create an empty dictionary to store the DataFrames
dfs = {}

# Loop through each .txt file
for file in txt_files:
    # Extract the DataFrame name from the file name
    df_name = os.path.basename(file).split('_reads_counts.txt')[0]

    # Read the file into a DataFrame
    df = pd.read_csv(file, delimiter='\t', header=None, names=['Data'])

    # Split the first column into two columns at the space
    df[['Sample', 'Count']] = df['Data'].str.split(' ', n=1, expand=True)

    # Extract relevant information
    sample_name = df.iloc[0, 0].split('_')[0]  # Extract 'IU112_S101'

    # Create a new DataFrame
    new_df = pd.DataFrame(index=[sample_name])

    # Iterate through rows and populate the new DataFrame
    for index, row in df.iterrows():
        column_name = row['Sample'].split('_')[-1].replace('.bam', '')  # Extract column name
        value = row['Count']
        new_df.loc[sample_name, column_name] = value

    # Remove the original column if needed
    df = df.drop(columns=['Data']) # Dropping the original column named 'Data'
    df = df.iloc[1:]

    # Extract "mouseBetter" using string manipulation
    df['Sample'] = df['Sample'].str.split('_').str[-1].str.split('.').str[0]

    # Store the DataFrame in the dictionary
    dfs[df_name] = df

# Access the DataFrames using their names
# For example, to access the DataFrame for "IU112_S101_read_counts.txt":
# df_IU112_S101 = dfs["IU112_S101"]
```

```
dfs['IU113_S102_read_counts.txt']
```

	Sample	Count
1	humanLoss	11409891
2	mouseLoss	115296065
3	mouseOnly	0
4	mouseBetter	6412452
5	humanOnly	0
6	humanBetter	60105366

```
# List to hold individual DataFrames with 'Category' as index
dfs_with_index = []

for sample, df in dfs.items():
    # Set 'Sample' as index and rename the 'Count' column to the sample name
    df = df.set_index('Sample')
    df = df.rename(columns={'Count': sample})
    dfs_with_index.append(df)
```

```
# Merge all dataframes on the 'Category' index
merged_df = pd.concat(dfs_with_index, axis=1)

merged_df = merged_df.rename(columns=lambda x: x.replace('_read_counts.txt', '') if '_read_counts.txt' in x else x)

df=merged_df.T
df
```

Sample	humanLoss	mouseLoss	mouseOnly	mouseBetter	humanOnly	humanBetter
IU113_S102	11409891	115296065	0	6412452	0	60105366
IU120_S105	524987	547041106	0	514009	0	261161465
IU119_S104	3561594	385814413	0	1967452	0	184217657
IU112_S101	5544210	366022277	0	2984962	0	175087025
IU118_S103	2940956	375186183	0	1680995	0	183266301
IU121_S106	2656235	493331288	0	1678899	0	236439899

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Assuming df is your DataFrame
df = df.drop(columns=['mouseLoss', 'humanLoss'])
df
```

Sample	mouseOnly	mouseBetter	humanOnly	humanBetter
IU113_S102	0	6412452	0	60105366
IU120_S105	0	514009	0	261161465
IU119_S104	0	1967452	0	184217657
IU112_S101	0	2984962	0	175087025
IU118_S103	0	1680995	0	183266301
IU121_S106	0	1678899	0	236439899

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Assuming df is your DataFrame
df['humanBetter'] = df['humanBetter'].astype(int)
df['mouseBetter'] = df['mouseBetter'].astype(int)
df['humanOnly'] = df['humanOnly'].astype(int)
df['mouseOnly'] = df['mouseOnly'].astype(int)
df['human'] = df['humanBetter'] + df['humanOnly']
df['mouse'] = df['mouseBetter'] + df['mouseOnly']

# Optionally, you can remove the original columns if you no longer need them:
df = df.drop(columns=['humanBetter', 'humanOnly', 'mouseBetter', 'mouseOnly'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, IU113_S102 to IU121_S106
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   human   6 non-null      int64
 1   mouse   6 non-null      int64
dtypes: int64(2)
memory usage: 316.0+ bytes
```

```
df
```

Sample	human	mouse
IU113_S102	60105366	6412452
IU120_S105	261161465	514009
IU119_S104	184217657	1967452
IU112_S101	175087025	2984962
IU118_S103	183266301	1680995
IU121_S106	236439899	1678899

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Calculate total counts for each row
df['total'] = df['human'] + df['mouse']

# Calculate percentages for each column
df['human_pct'] = round((df['human'] / df['total']) * 100,2)
df['mouse_pct'] = round((df['mouse'] / df['total']) * 100,2)

# Create the new dataframe with only percentages
percentage_df = df[['human_pct', 'mouse_pct']]

# Display the new dataframe
percentage_df
```

Sample	human_pct	mouse_pct
IU113_S102	90.36	9.64
IU120_S105	99.80	0.20
IU119_S104	98.94	1.06
IU112_S101	98.32	1.68
IU118_S103	99.09	0.91
IU121_S106	99.29	0.71

Next steps: Generate code with percentage_df View recommended plots New interactive sheet

```
df = df[['human','mouse']]
df
```

Sample	human	mouse
IU113_S102	60105366	6412452
IU120_S105	261161465	514009
IU119_S104	184217657	1967452
IU112_S101	175087025	2984962
IU118_S103	183266301	1680995
IU121_S106	236439899	1678899

Next steps: Generate code with df View recommended plots New interactive sheet

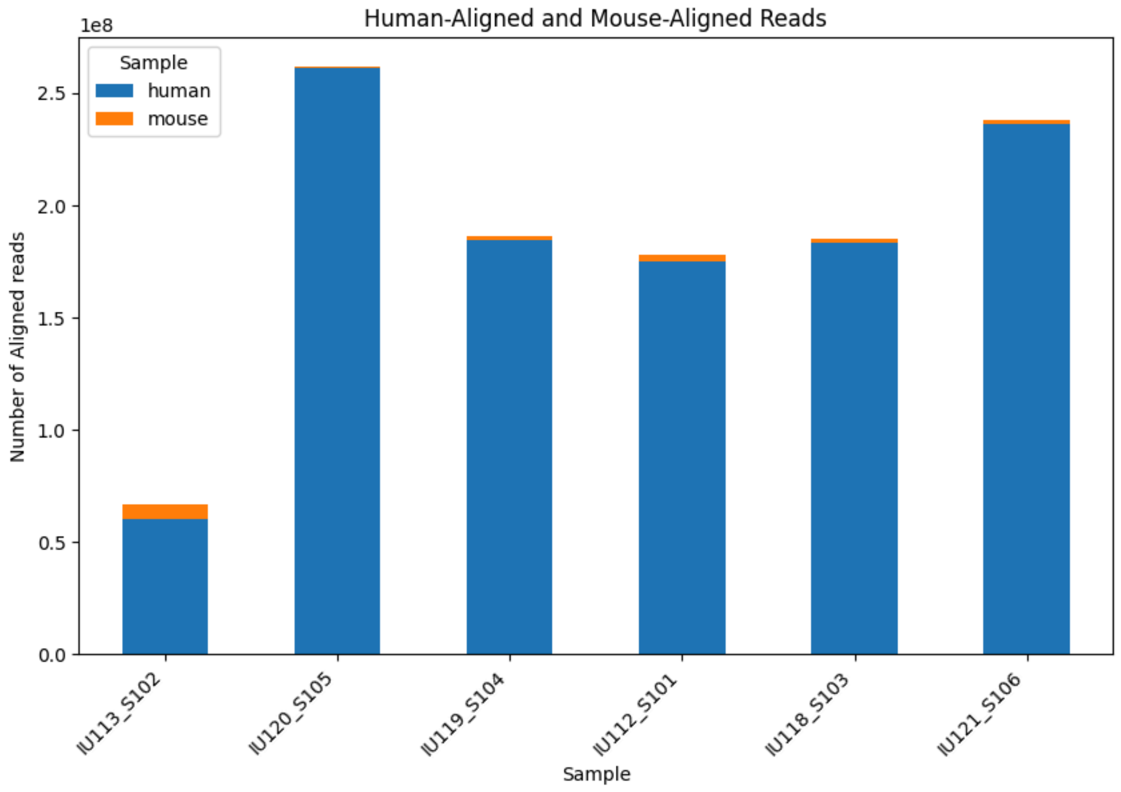
```
import matplotlib.pyplot as plt
import pandas as pd

# Create the stacked bar plot
ax = df.plot(kind='bar', stacked=True, figsize=(10, 6))

# Set plot labels and title
plt.xlabel('Sample')
plt.ylabel('Number of Aligned reads')
plt.title('Human-Aligned and Mouse-Aligned Reads')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')
```

```
# Display the plot
plt.show()
```



```
!jupyter nbconvert --to html /content/your_notebook.ipynb
```