pyTCR

pyTCR is a comprehensive Jupyter notebook-based platform for T-Cell receptor (TCR) data analysis to facilitate reproducibility and rigor of immunogenomics research

User Manual

Introduction

T cell receptor (TCR) studies have grown substantially with the advancement in the sequencing techniques of T cell receptor repertoire sequencing (TCR-Seq). The analysis of the TCR-Seq data requires computational skills to run the computational analysis of TCR repertoire tools. However biomedical researchers with limited computational backgrounds face numerous obstacles to properly and efficiently utilizing bioinformatics tools for analyzing TCR-Seq data. Here we report pyTCR, a computational notebook-based solution for comprehensive and scalable TCR-Seq data analysis. Computational notebooks, which combine code, calculations, and visualization, are able to provide users with a high level of flexibility and transparency for the analysis. Additionally, computational notebooks are demonstrated to be user-friendly and suitable for researchers with limited computational skills. Our platform has a rich set of functionalities including various TCR metrics, statistical analysis, and customizable visualizations. The application of pyTCR on large and diverse TCR-Seq datasets will enable the effective analysis of large-scale TCR-Seq data with flexibility, and eventually facilitate new discoveries.

Workflow

pyTCR utilizes interactive computational Jupyter notebooks to facilitate reproducibility and rigor of performed TCR-Seq analysis. You may use Jupyter Notebook on your local computer or use these notebooks on Google Colab.

Jupyter Notebook

Step 1 - Install Jupyter Notebook

In order to use Jupyter Notebook locally, you need to install the required software first. The easiest way to get started is to install Anaconda, a distribution of Python libraries and useful tools. You may follow the instructions to install Anaconda (https://www.anaconda.com/products/distribution). After getting Anaconda, open your terminal and use pip to install Jupyter Notebook (https://jupyter.org/install).

```
pip3 install jupyter
```

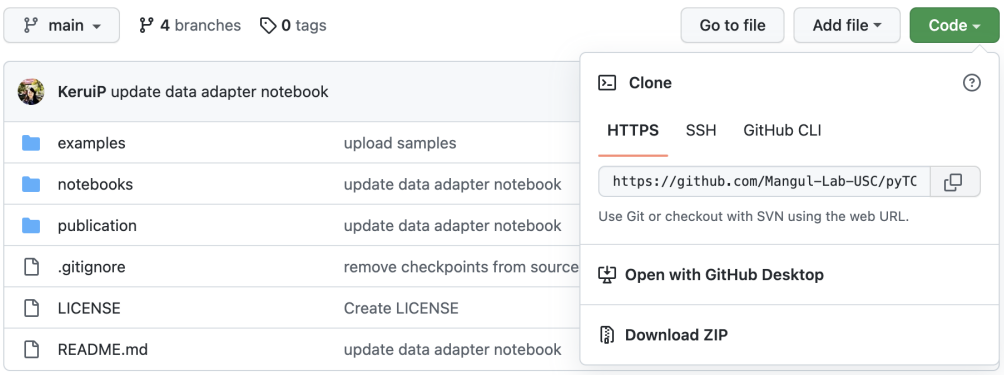
To run Jupyter Notebook, you may type the following command.

```
Jupyter notebook
```

Step 2 - Clone pyTCR repository

Now, the quickest way to use pyTCR is to clone the pyTCR repository from GitHub. Here is the guide provided by GitHub for cloning a repository (https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository).

For example, you may clone pyTCR repository to your desktop using HTTPS. Go to https://github.com/Mangul-Lab-USC/pyTCR, then copy the HTTPS link to the terminal to get pyTCR



```
~ :> cd desktop
desktop :> git clone https://github.com/Mangul-Lab-USC/pyTCR.git
```

After cloning the repository, use the following command to open the Jupyter notebooks

```
desktop :> cd pytcr
pytcr [main] :> cd notebooks/
notebooks [main] :> jupyter notebook
```

From here, you will be able to select the analysis that you want to run. You may want to run `data_adapter.ipynb` notebook first. Please refer to the following sections for step by step guide.

Google Colab Usage

pyTCR provides explicit support for use in Google Colab environments which is free to use for Google users.

According to some recent limitations of colab (for free space), we just used a light sample, but you can use big data as input for a run on your computer or upgrade the colab to premium to have more space.

Step 1 - Set up a Google Colab environment

You need to log in to your Google account, then click on this link to go to the Google Colab homepage: https://colab.research.google.com/

Step 2 - Upload notebooks

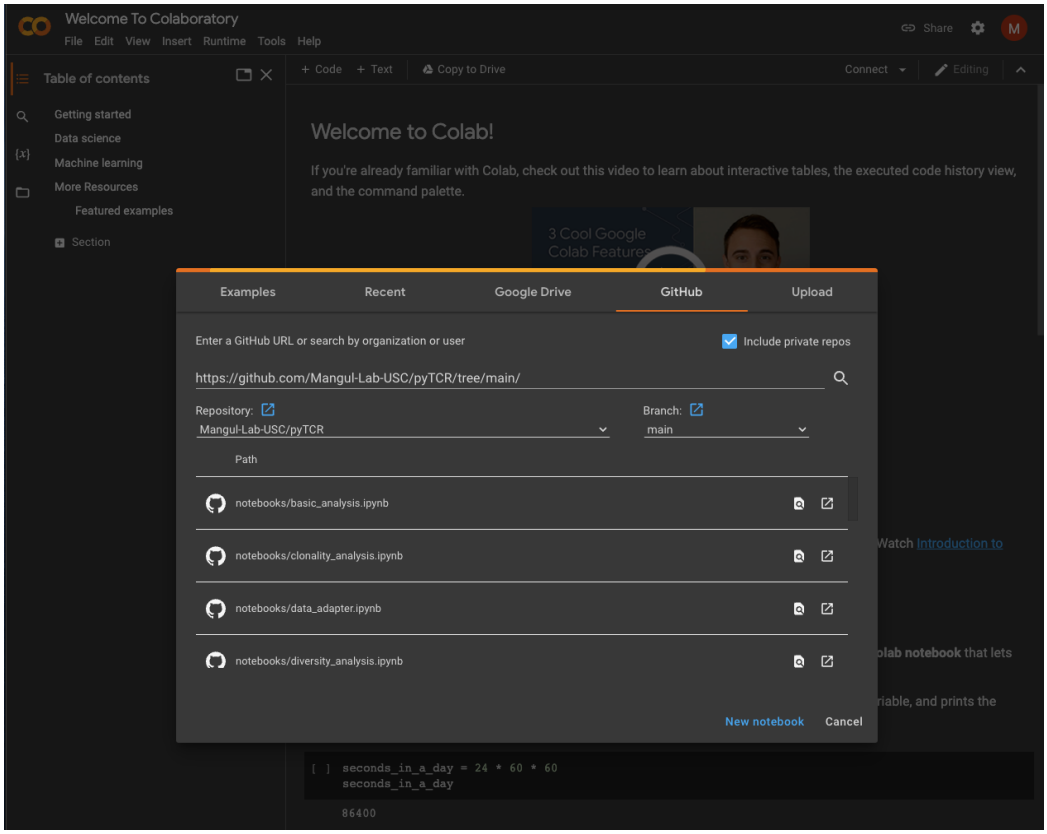
From Google Colab, you can upload pyTCR notebooks by clicking file -> upload notebook. From there, you have multiple different options to upload the notebooks:

- Google Drive: Upload the notebooks directly to your Google Drive, and then select them from the upload prompt

- GitHub: You can use the notebooks without ever downloading them through GitHub. Select the GitHub tab from the upload prompt and paste the following url into the search: https://github.com/Mangul-Lab-USC/pyTCR/tree/main/
    - From there, it will list all available notebooks, and allow you to select which ones you want to use

- Direct upload: This option allows you to upload the notebooks from your local computer into Google Colab

We recommended using GitHub. You can use the notebooks without ever downloading them through GitHub. Select the GitHub tab from the upload prompt and paste the following url into the search: https://github.com/Mangul-Lab-USC/pyTCR/tree/main/



From there, it will list all available notebooks, and allow you to select which ones you want to use.

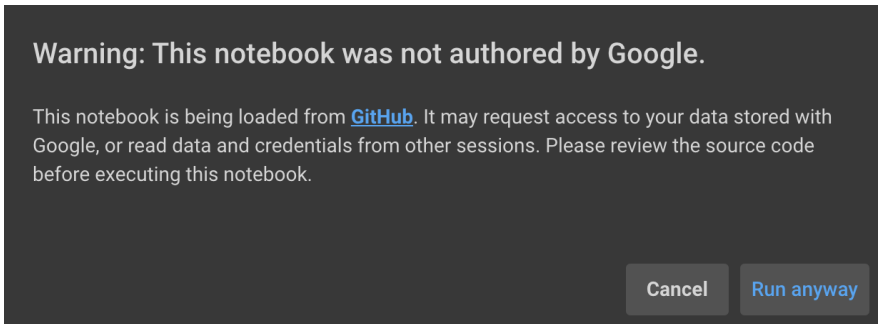
By nature, all notebooks can be modified to fit any data format. However, pyTCR includes a default format. Most likely you will have one TCR-Sequencing file for each sample, pyTCR provides a notebook "`data_adapter.ipynb`", which can be used to convert existing data formats such as MiXCR outputs or any other tabular format, to the pyTCR format and combine individual sample files into one file to further analysis. This is generally faster than modifying each notebook individually to fit your data setup. pyTCR expects the following columns in your data file and is required for the notebooks to work out of the box.

Here we show the steps to run `data_adapter.ipynb` notebook.

Step 1 - Import the required python libraries

```
[ ]  %pip install IPython
     %pip install pandas

     from IPython import get_ipython
     import pandas as pd
     import glob
     import os
```

When you first-time run and import “python libraries”, will get this message:



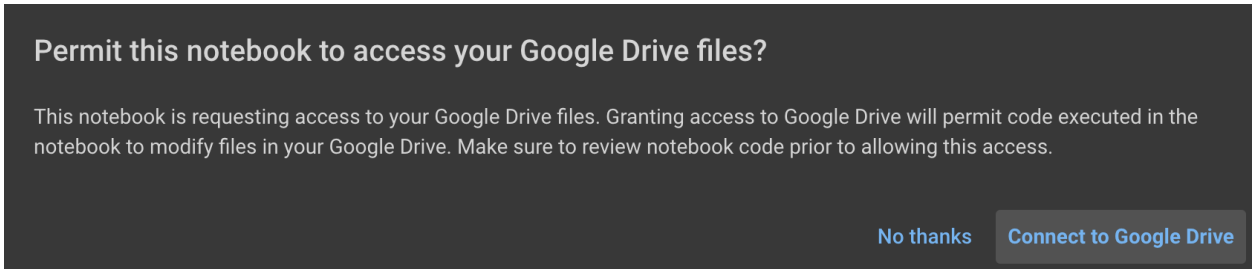
Click on “Run anyway”

Step 2 - Load and read your data file

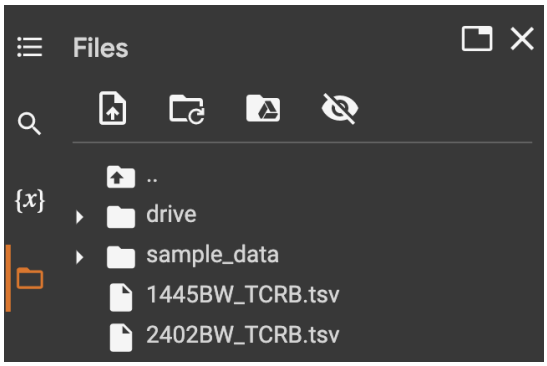
First, you need to run the following cell to mount Google Colab.

```
[ ]  # Mount Google Colab
     isInGoogle = 'google.colab' in str(get_ipython())

     if isInGoogle:
         from google.colab import drive
         drive.mount('/content/drive')
```

After running, follow the instructions to connect to your Google Drive.



Assume that you have stored your data on your Google Drive. If not, you may use the file upload option at the sidebar to upload data files to Google Drive.



Then read your data from your Google Drive path. The example files (1445BW_TCRB.tsv and 2402BW_TCRB.tsv) are from Adaptive Biotechnologies.

```
[ ]  # Specify the path to your file in Google Drive or locally
     filePath = "/content/2402BW_TCRB.tsv"

     targetFilename = os.path.basename(filePath)

     df_samples = pd.read_table(filePath, low_memory=False, engine="c")

     df_samples.head()
```

Step 3 - Convert data to the pyTCR standardized format

column	name	description
1	sample	The name of the sample (optional)
2	freq	The share of clonotypes in the sample (required)
3	#count	The number of reads (required)
4	ctr3aa	CDR3 amino acid clonotype (required)

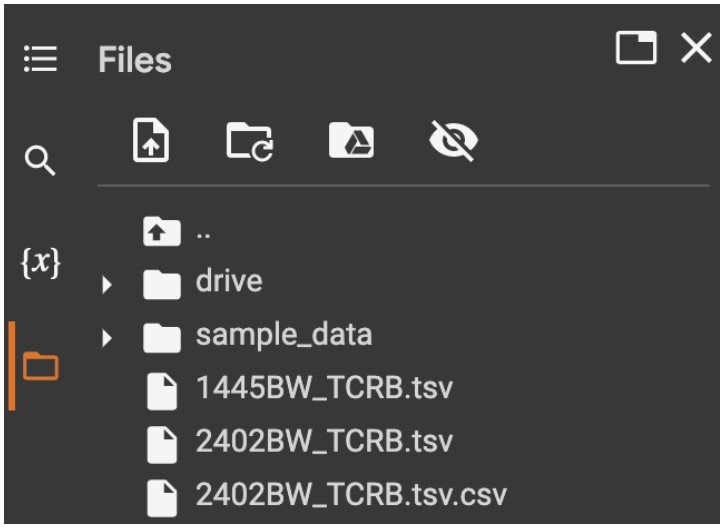
5	ctr3nt	CDR3 nucleotide (required)
6	v	V gene (required)
7	d	D gene (required)
8	j	J gene (required)
…	optional fields	any other fields intended for your use (optional)

Modify the `required_columns` below to match the column names from your data that are equivalent to pyTCR's columns in the same order as described above.

The following code will create a new `.csv` file with the correct pyTCR column names and place it in the current directory.

```
[ ] # Enter the column names from your data that represent the required pyTCR columns
    required_columns = [
    'sample', 'frequency', 'templates',
    'amino_acid', 'rearrangement', 'v_resolved' , 'd_resolved', 'j_resolved'
    ]

    df_new = df_samples.filter(required_columns)

    # Rename the columns to pyTCR standard names
    df_new.columns = [
    'sample', 'freq', '#count', 'cdr3aa',
    'cdr3nt', 'v', 'd', 'j']

    df_new.to_csv(f'./{targetFilename}.csv', na_rep='.', index=False)
```

Then Google Colab makes a new `.csv` file in your sample_data path:



Repeat steps 2 and 3 for all the sample files in order to convert them to pyTCR standardized format.

If you have MiXCR output, run the following cells in step 3

```
[ ]  # If you have MiXCR results, please run the code cell below
     df_samples['allVHitsWithScore']=df_samples['allVHitsWithScore'].str.replace(r"\(.*\)","")
     df_samples['allDHitsWithScore']=df_samples['allDHitsWithScore'].str.replace(r"\(.*\)","")
     df_samples['allJHitsWithScore']=df_samples['allJHitsWithScore'].str.replace(r"\(.*\)","")
```

```
[ ]  # Enter the column names from your data that represent the required pyTCR columns
     required_columns = [
     'cloneFraction', 'cloneCount',
     'aaSeqCDR3', 'nSeqCDR3', 'allVHitsWithScore' , 'allDHitsWithScore', 'allJHitsWithScore']

     df_new = df_samples.filter(required_columns)

     # Rename the columns to pyTCR standard names
     df_new.columns = ['freq', '#count', 'cdr3aa','cdr3nt', 'v', 'd', 'j']

     df_new.to_csv(f'./{targetFilename}.csv', na_rep='.', index=False)
```

Step 4 - Combine all sample files

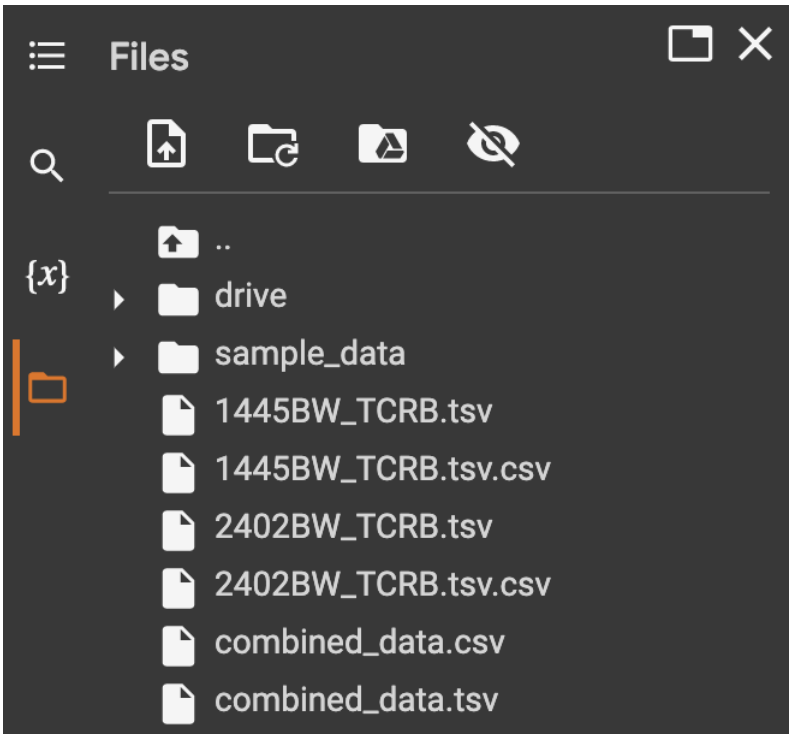
After converting all sample files, you can run the following cell to generate one `.csv` file that combines all sample files. It also adds a new column `sample` to each `.csv` file in the current directory with the filename as the value.

```
[ ]  globbed_files = glob.glob("*.csv")

     data = []

     for csv in globbed_files:
         dataframe = pd.read_csv(csv)
         dataframe['sample'] = os.path.basename(csv.split('.')[0])
         data.append(dataframe)

     combined_data = pd.concat(data)
     combined_data.to_csv("combined_data.csv",index=False)

     df=pd.read_csv("combined_data.csv", index_col=[0])

     df
```

Run the following cell to convert the combined `.csv` file to `.tsv` file for the following analysis

```
[ ]  targetFileExtension = 'tsv'

     df = pd.read_csv("/content/combined_data.csv", low_memory=False, engine="c")

     file = "/content/combined_data.csv".split('.')[0]

     newFile = f'{file}.new.{targetFileExtension}'

     # Save new file to current directory
     df.to_csv(newFile, sep='\t', na_rep='.', index=False)
```

Congratulations! You have successfully combined all samples together. You may use the “combined_data.tsv” for other pyTCR notebooks.



Here we use `basic_analysis.ipynb` notebook as an example to demonstrate how to analyze the data, other notebooks have a similar structure.

Step 1 - Import python libraries

```
%pip install IPython
%pip install matplotlib
%pip install pandas
%pip install seaborn
%pip install scipy

from IPython import get_ipython
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy import stats

pd.options.mode.chained_assignment = None
```

Step 2 - Load and read your data file

Same as the previous step we need to read data:

```
# Specify the path to your data in Google Drive or locally
#filePath = "./complete_COVID_SAMPLES.tsv" # "/content/drive/MyDrive/complete_COVID_samples.tsv"
filePath = "/content/drive/MyDrive/2402BW_TCRB.tsv"

isInGoogle = 'google.colab' in str(get_ipython())

if isInGoogle:
    from google.colab import drive
    drive.mount('/content/drive')

df = pd.read_table(filePath, low_memory=False, engine="c")

optional_fields = ['hospitalized']

df.head()
```

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

	sample	freq	#count	cdr3aa	cdr3nt	v	d	j
0	2402BW_TCRB	71788	0.107454	CASNPRGYEADTQYF	NTGTCGGCTGCTCCCTCCCAAACATCTGTGTACTTCTGTGCCAGCA...	TCRBV06-02*02	TCRBD02-01	TCRBJ02-03*01
1	2402BW_TCRB	18375	0.027504	CASSQSEFYNSPLHF	CAGCCTGCAGAACTGGAGGATTCTGGAGTTTATTTCTGTGCCAGCA...	TCRBV14-01*01	unknown	TCRBJ01-06*02
2	2402BW_TCRB	8226	0.012313	CASSRAHRGQRPTKPQHF	CAGCAGGAGGACTCCGCCGTGTATCTCTGTGCCAGCAGCCGGGCCC...	TCRBV07-08*03	TCRBD01-01*01	TCRBJ01-05*01
3	2402BW_TCRB	6314	0.009451	CASRSIPQRATNEKLFF	CCCAGCCCCAACCAGACCTCTCTGTACTTCTGTGCCAGCAGGTCAA...	TCRBV27-01*01	TCRBD02-01	TCRBJ01-04*01
4	2402BW_TCRB	4052	0.006065	CSPPRNTEAFF	ACTCTGACTGTGAGCAACATGAGCCCTGAAGACAGCAGCATATATC...	TCRBV29-01	TCRBD02-01	TCRBJ01-01*01

Step 3 - Analysis

Basic analysis 1 - Reads count:

If you just use a single file, please do not consider "+ optional_fields"

```
[3]  df_count = df.groupby(['sample'] + optional_fields).agg(
         {'#count': 'sum'}).reset_index().rename(columns={'#count': "reads_count"})
```

Basic analysis 2 - Clonotype count:

```
df_diversity = df.groupby(['sample'] + optional_fields,
                          sort=False).size().reset_index(name='clonotype_count')
```

Basic analysis 3 - Mean frequency:

```
[ ]  df_mean_frequency = df.groupby(['sample'] + optional_fields).agg(
         {'freq': 'mean'}).reset_index().rename(columns={'freq': "mean_frequency"})
```

Basic analysis 4 - Geometric mean of clonotype frequency:

- Create an empty data frame for storing results
- Store the results

```
[ ]  from scipy.stats.mstats import gmean

     samples = df['sample'].unique()

     # Create an empty dataframe for storing results
     df_geomean_frequency = pd.DataFrame(columns=['sample', 'geomean_frequency'])

     for sample in samples:
         tmp = df[df['sample'] == sample]
         geomean_frequency = gmean(tmp['freq'])

         # Store the results
         df_data = pd.DataFrame({'sample': sample, 'geomean_frequency': geomean_frequency}, index=[0])
         df_geomean_frequency = pd.concat([df_geomean_frequency, df_data], copy=False, ignore_index=True)
```

Basic analysis 5 - Mean length of CDR3 nucleotide sequence

```
df['length_weighted'] = df['cdr3nt'].str.len()*df['freq']
df_mean_cdr3nt_length = df.groupby(['sample'] + optional_fields).agg(
    {'length_weighted': 'sum'}).reset_index().rename(columns={'length_weighted': "mean_cdr3nt_length"})
```

Basic analysis 6 - Convergence:

- Count unique CDR3
- Calculate the mean of the unique CDR3 count in each sample

```
[ ]  # Count unique CDR3
     df_unique_CDR3 = df.groupby(['cdr3aa', 'sample'] + optional_fields, as_index=False)[
         'cdr3nt'].agg({'count': 'count'})

     # Calculate the mean of the unique CDR3 count in each sample
     df_unique_CDR3_mean = df_unique_CDR3.groupby(['sample'] + optional_fields).agg(
         {'count': 'mean'}).reset_index().rename(columns={'count': "convergence"})
```

Basic analysis 7:

1 - Spectratype

- CDR3 nucleotide length
- Calculate spectra type

```
# CDR3 nucleotide length
df['nt_length'] = df['cdr3nt'].str.len()

# Calculate spectratype
df_spectratype = df.groupby(['sample', 'nt_length'] + optional_fields).agg(
    {'freq': 'sum'}).reset_index().rename(columns={'freq': "spectratype"})

df_spectratype
```

2 - Spectratype bar plot for an individual sample

- Define the sample that you would like to plot, replace the "1132289BW_TCRB " with the sample name of interest
- x-axis and y-axis labels, figure size, font size are customizable

```
sample_name = ""

df_sample = df_spectratype.loc[df_spectratype['sample'] == sample_name]

ax = plt.subplots(figsize=(10, 10))
ax = sns.barplot(data=df_sample, x='nt_length', y='spectratype')
ax.set_xlabel('nt_length', fontsize=20)
ax.set_ylabel('frequency', fontsize=20)
plt.xticks(fontsize=10)
plt.yticks(fontsize=20)
sns.despine()

plt.show()
```

Basic analysis 8 - Summary table for basic analysis:

- Merge df_count and df_geomean_frequency first
- Create a data frame that combines all the basic analysis (except for spectratype)

```
# Merge df_count and df_geomean_frequency first
df_geomean_frequency = df_geomean_frequency.merge(
    df_count, on='sample', how='left')

# Create a dataframe that combines all the basic analysis (except for spectratype)
dfs = [df_diversity, df_mean_frequency, df_geomean_frequency,
       df_mean_cdr3nt_length, df_unique_CDR3_mean]

axis = ['sample'] + optional_fields

df_combined = pd.merge(dfs[0], dfs[1], left_on=axis, right_on=axis, how='outer')
for d in dfs[2:]:
    df_combined = pd.merge(df_combined, d, left_on=axis, right_on=axis, how='outer')

df_combined
```

Basic analysis 9 - Statistical analysis

Basic analysis 9.1 - Test if the metric is usually distributed

- The null hypothesis here is normality.
- If the p-value exceeds 0.05, we cannot reject the null hypothesis (it is a normal distribution). If the p-value is smaller than 0.05, we reject the null hypothesis (it is not a normal distribution)
- Change 'clonotype_count' to other metrics that you are interested in

```
x = stats.normaltest(df_combined['clonotype_count'])
x
```

Basic analysis 9.2 - Mean or median of diversity metrics among groups

- If the dataset is normally distributed, calculate mean
- If the dataset is not normally distributed, calculate median
- Change 'clonotype_count' to other metrics that you are interested in

```
# Modify `attribute` to your intended data column
attribute = "hospitalized"

# Calculate the mean among two groups
df_metric_mean = df_combined.groupby(attribute)['clonotype_count'].mean().reset_index()

df_metric_mean
```

```
# Modify `attribute` to your intended data column
attribute = "hospitalized"

# Calculate the median among two groups
df_metric_median = df_combined.groupby(attribute)['clonotype_count'].median().reset_index()

df_metric_median
```

Basic analysis 9.3 - Stats test:

- If the dataset is normally distributed, use t-test (stats.ttest_ind)

  change the group1 and group2 to the groups/samples that you are interested in
- If the dataset is not normally distributed, use Wilcoxon rank-sum test (stats.ranksums)

  change the group1, and group2 to the groups/samples that you are interested in
- Change 'clonotype_count' to other metrics that you are interested in

```
# Modify `attribute` to your intended data column
attribute = "hospitalized"

df1 = df_combined.copy()
df_group1 = df1[df1[attribute] == True]
df_group2 = df1[df1[attribute] == False]
stats.ttest_ind(df_group1['clonotype_count'], df_group2['clonotype_count'])
```

Basic analysis 10 - Visualization (Clonotype count violin plot per group)

- X-axis and y-axis labels, figsize, fontsize are customizable
- Change the violin plot (sns.violinplot) to the plot type that you are interested in, includes strip plot (sns.stripplot), swarm plot (sns.swarmplot), box plot (sns.boxplot), boxen plot (sns.boxenplot), point plot (sns.pointplot), and bar plot (sns.barplot)
- Change 'clonotype_count' to other metrics that you are interested in

```
# Modify `attribute` to your intended data column
attribute = "hospitalized"

ax = plt.subplots(figsize=(10, 10))

ax = sns.violinplot(x=attribute, y='clonotype_count', data=df_combined)

ax.set_xlabel(attribute, fontsize=20)
ax.set_ylabel('number of clonotypes', fontsize=20)
plt.xticks(fontsize=20,rotation=90)
plt.yticks(fontsize=20)
sns.despine()

plt.show()
```

Thank you for reviewing the instructions of pyTCR!

📣 Please use Discussions for questions, suggestions or feature requests 😀

https://github.com/Mangul-Lab-USC/pyTCR/issues