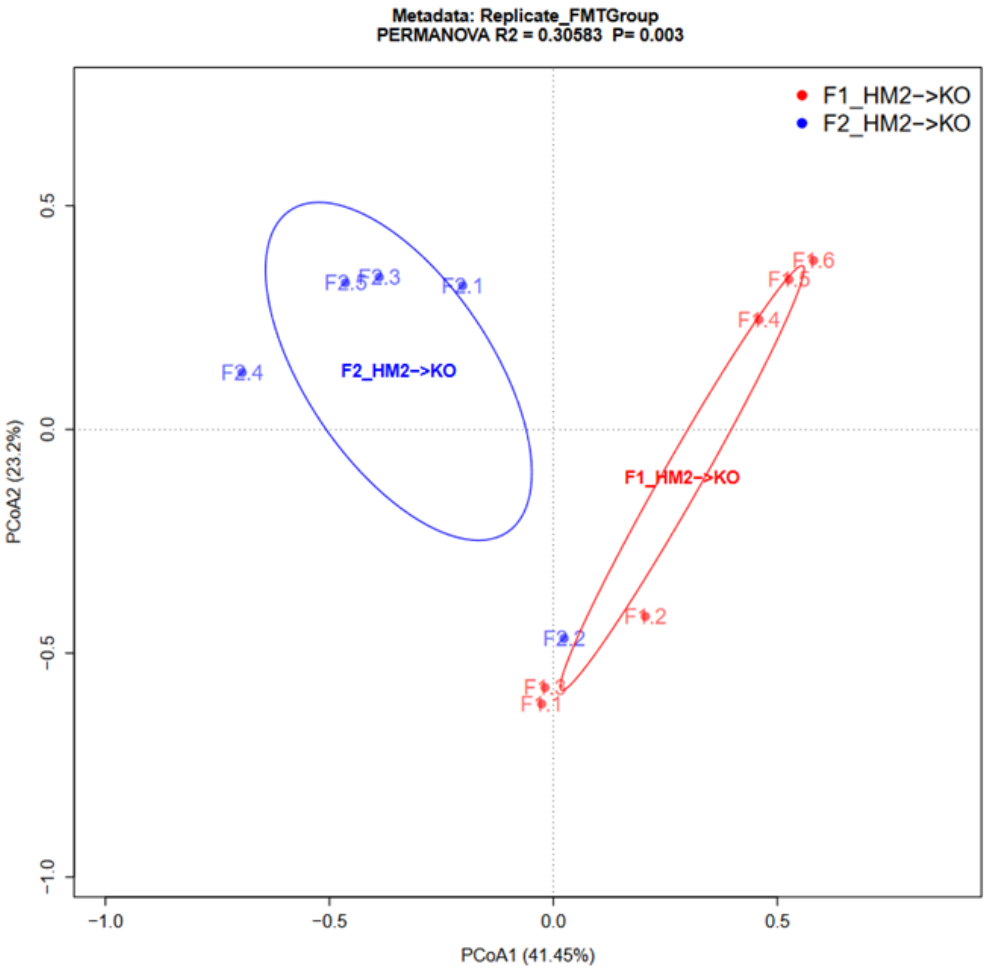
Contents

Aim: Replicate figure



Main Python Methods

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import os
import skbio
from scipy import stats
from scipy.spatial.distance import pdist, squareform
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms

# ****************************** MAIN METHODS ******************************
# Retrieve tab delimited file
def read_csv_file(file_path, skiprows=None, header = 0, sep = '\t', index_col=
False):
    df = pd.read_csv(file_path, skiprows=skiprows, sep=sep, header=header, ind
ex_col=index_col)

    return df

# Reference:  Function "get_cov_ellipse" was extracted from the following URL:

# https://scipython.com/book/chapter-7-matplotlib/examples/bmi-data-with-confi
dence-ellipses/
# (Learning Scientific Programming with Python by Christian Hill)
def get_cov_ellipse(cov, centre, nstd, **kwargs):
    """
    Return a matplotlib Ellipse patch representing the covariance matrix
    cov centred at centre and scaled by the factor nstd.
    """
    # Find and sort eigenvalues and eigenvectors into descending order
    eigvals, eigvecs = np.linalg.eigh(cov)
    order = eigvals.argsort()[::-1]
    eigvals, eigvecs = eigvals[order], eigvecs[:, order]

    # The anti-clockwise angle to rotate our ellipse by
    vx, vy = eigvecs[:,0][0], eigvecs[:,0][1]
    theta = np.arctan2(vy, vx)

    # Width and height of ellipse to draw
    width, height = 2 * nstd * np.sqrt(eigvals)
    return Ellipse(xy=centre, width=width, height=height,
                   angle=np.degrees(theta), fill=False, **kwargs)

# ****************************** INPUT FILE PATHS ***************************
****
# Get current working directory
current_working_dir = os.getcwd()
# original counts table file path
feature_tbl_file_path = os.path.join(current_working_dir, 'feature-table_balfo
ur.txt')
# dictionary for two sample groupings
dict_metadata = {'129.IL10KO_1':['F1-1', 'F1-2', 'F1-3', 'F1-4', 'F1-5', 'F1-
6'],
                 '129.IL10KO_2':['F2-1' ,'F2-2', 'F2-3', 'F2-4', 'F2-5']}
# cross reference for sample grouping names
cross_ref_dict = {'129.IL10KO_1':'F1_HM2->KO', '129.IL10KO_2':'F2_HM2->KO'}
# cross reference for sample names
sample_cross_ref_dict = {'F1-1':'F1.1', 'F1-2':'F1.2', 'F1-3':'F1.3', 'F1-4':
'F1.4', 'F1-5':'F1.5', 'F1-6':'F1.6',
                         'F2-1':'F2.1','F2-2':'F2.2', 'F2-3':'F2.3', 'F2-4':'F
2.4', 'F2-5':'F2.5'}
```

Data Processing Steps

Step 1: Retrieve source feature counts table

In [2]:
```
df_feature_counts = read_csv_file(feature_tbl_file_path, 1)
df_feature_counts = df_feature_counts.astype({col:'int32' for col in df_featur
e_counts.columns[1:]})
df_feature_counts
```

Out[2]:

	#OTU ID	1gKO.1	1gKO.2	1gKO.3	1gWT.1	1gWT.2	1gWT.3
0	d__Bacteria;p__Verrucomicrobiota;c__Verrucomic...	11370	14120	14648	16632	19817	21706
1	d__Bacteria;p__Firmicutes;c__Clostridia;o__Lac...	22545	25836	17048	7547	9272	5996
2	d__Bacteria;p__Proteobacteria;c__Gammaproteoba...	5919	6481	6714	31	31	30
3	d__Bacteria;p__Firmicutes;c__Clostridia;o__Lac...	8134	9827	7090	8264	10100	6324
4	d__Bacteria;p__Firmicutes;c__Clostridia;o__Lac...	6280	7649	5719	9715	11851	8047
...	...	...	...	...	...	...	...
191	d__Bacteria;p__Firmicutes;c__Clostridia;o__Pep...	0	0	0	0	0	0
192	d__Bacteria;p__Firmicutes;c__Clostridia;o__Clo...	0	0	0	0	0	0
193	d__Bacteria;p__Firmicutes;c__Clostridia;o__Chr...	0	0	0	0	0	0
194	d__Bacteria;p__Firmicutes;c__Incertae_Sedis;o_...	0	0	0	0	0	0
195	d__Bacteria;p__Firmicutes;c__Bacilli;o__Lactob...	0	0	0	0	0	0

196 rows × 111 columns

Step 2: Filter feature counts table with target sample columns

In [3]:
```
target_samples = [ sample for sample_group in list(dict_metadata.values()) for
 sample in sample_group ]
df_feature_counts_filtered = df_feature_counts[target_samples]
df_feature_counts_filtered
```

Out[3]:

	F1-1	F1-2	F1-3	F1-4	F1-5	F1-6	F2-1	F2-2	F2-3	F2-4	F2-5
0	10401	2753	7914	2398	6595	2313	8957	10961	344	460	63
1	7133	7051	3523	26054	524	2105	10749	5732	14621	319	9372
2	2132	5867	4116	54	87	284	2194	1425	3296	1071	13246
3	1965	394	779	828	929	1973	4084	39	2827	17547	2891
4	4785	1982	3119	2274	41521	49604	3306	3350	3278	2247	1094
...	...	...	...	...	...	...	...	...	...	...	...
191	0	0	0	0	0	0	0	0	0	0	0
192	0	0	0	0	0	0	0	0	0	0	0
193	0	0	0	0	0	0	0	0	0	0	0
194	0	0	0	0	0	0	0	0	0	0	0

195	0	0	0	0	0	0	0	0	0	0

196 rows × 11 columns

Step 3: Calculate total read counts per sample

In [4]:
```
sample_count_sums = df_feature_counts_filtered.sum(axis=0)
sample_count_sums
```

Out[4]:
```
F1-1    120620
F1-2     89533
F1-3    103243
F1-4     92643
F1-5    120934
F1-6     98057
F2-1     98214
F2-2     90013
F2-3     95932
F2-4    111659
F2-5     87600
dtype: int64
```

Step 4: Generate normalized counts feature table

In [5]:
```
df_feature_counts_filtered_norm = np.log10((df_feature_counts_filtered/sample_
count_sums)*sample_count_sums.mean() + 1)
df_feature_counts_filtered_norm
```

Out[5]:

	F1-1	F1-2	F1-3	F1-4	F1-5	F1-6	F2-1	F2-2	F2-3	F2-
0	3.939028	3.491286	3.887914	3.416526	3.740068	3.376202	3.963359	4.088903	2.559118	2.61923
1	3.775248	3.899645	3.536501	4.452400	2.641099	3.335297	4.042557	3.807391	4.186364	2.46072
2	3.250934	3.819821	3.604045	1.776236	1.866243	2.466648	3.352579	3.203104	3.539472	2.98566
3	3.215530	2.647814	2.881570	2.955022	2.889351	3.307185	3.622339	1.649917	3.472831	4.19965
4	3.601894	3.348638	3.483620	3.393476	4.539055	4.707369	3.530580	3.574178	3.537094	3.30724
...	...	...	...	...	...	...	...	...	...	.
191	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
192	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
193	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
194	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
195	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

196 rows × 11 columns

Step 5: Generate Bray-Curtis dissimilarity distance matrix

In [6]:
```
condensed_arr = pdist(df_feature_counts_filtered_norm.T, metric='braycurtis')
```

```
dist_mat_sym = squareform(condensed_arr)
print('Dimensions of symmetrical Bray-Curtris distance matrix: {}'.format(dist
_mat_sym.shape))
dist_mat_sym
```

```
Dimensions of symmetrical Bray-Curtris distance matrix: (11, 11)
```

Out[6]:
```
array([[0.        , 0.17543969, 0.1210743 , 0.24413798, 0.26614686,
        0.27932716, 0.24563653, 0.16897739, 0.25286129, 0.25787212,
        0.26180328],
       [0.17543969, 0.        , 0.12735125, 0.21321068, 0.23873452,
        0.23562064, 0.27308808, 0.22960201, 0.24593477, 0.32563064,
        0.25044773],
       [0.1210743 , 0.12735125, 0.        , 0.25242907, 0.24876063,
        0.27042458, 0.25521475, 0.17566823, 0.23123946, 0.2648364 ,
        0.23475823],
       [0.24413798, 0.21321068, 0.25242907, 0.        , 0.15078165,
        0.16505002, 0.24546878, 0.23891727, 0.2609215 , 0.34959776,
        0.27547553],
       [0.26614686, 0.23873452, 0.24876063, 0.15078165, 0.        ,
        0.12648274, 0.24629414, 0.26384811, 0.27647145, 0.34676066,
        0.31165956],
       [0.27932716, 0.23562064, 0.27042458, 0.16505002, 0.12648274,
        0.        , 0.29074424, 0.29024396, 0.29012887, 0.36578334,
        0.30479539],
       [0.24563653, 0.27308808, 0.25521475, 0.24546878, 0.24629414,
        0.29074424, 0.        , 0.20846872, 0.17002751, 0.26274988,
        0.1835378 ],
       [0.16897739, 0.22960201, 0.17566823, 0.23891727, 0.26384811,
        0.29024396, 0.20846872, 0.        , 0.24951014, 0.30229098,
        0.27901923],
       [0.25286129, 0.24593477, 0.23123946, 0.2609215 , 0.27647145,
        0.29012887, 0.17002751, 0.24951014, 0.        , 0.18683878,
        0.11773076],
       [0.25787212, 0.32563064, 0.2648364 , 0.34959776, 0.34676066,
        0.36578334, 0.26274988, 0.30229098, 0.18683878, 0.        ,
        0.1907071 ],
       [0.26180328, 0.25044773, 0.23475823, 0.27547553, 0.31165956,
        0.30479539, 0.1835378 , 0.27901923, 0.11773076, 0.1907071 ,
        0.        ]])
```

Step 6: Perform principal coordinate analysis (PCoA)

In [7]:
```
import warnings
warnings.filterwarnings('ignore')
my_pcoa = skbio.stats.ordination.pcoa(dist_mat_sym)
df_pcoa = my_pcoa.samples[['PC1', 'PC2']]
# Normalize PC1 and PC2 into unit vectors
df_pcoa = pd.DataFrame(df_pcoa.to_numpy()/np.linalg.norm(df_pcoa.to_numpy(), a
xis=0))
print('PCoA proportion explained:')
my_pcoa.proportion_explained
```

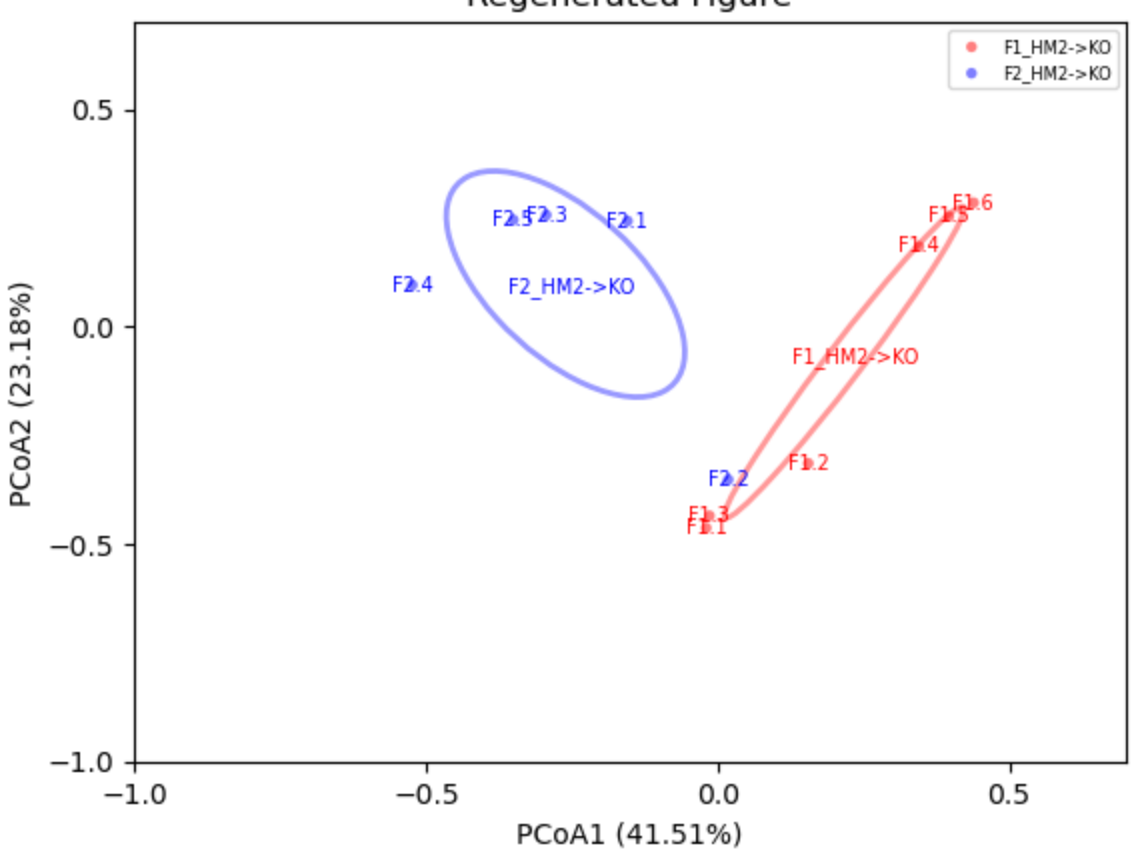
```
PCoA proportion explained:
```

Out[7]:
```
PC1     0.415128
PC2     0.231760
PC3     0.129317
PC4     0.093920
```

```
PC5     0.044440
PC6     0.033396
PC7     0.028290
PC8     0.017374
PC9     0.006376
PC10    0.000000
PC11    0.000000
dtype: float64
```

Step 7: Plot PCoA results

In [8]:
```
colors = ['red', 'blue']
text_tup = [(-0.09, 0.00),
            (-0.1, -0.02)]

fig, ax = plt.subplots()
ax.set_xlim((-1.0, 0.70))
ax.set_ylim((-1.0, 0.70))
ax.set_yticks([-1.0, -0.5, 0.0, 0.5])
ax.set_xticks([-1.0, -0.5, 0.0, 0.5])
ax.set_xlabel('PCoA1 ({}%)'.format(np.round(my_pcoa.proportion_explained.PC1 *
 100, 2)))
ax.set_ylabel('PCoA2 ({}%)'.format(np.round(my_pcoa.proportion_explained.PC2 *
 100, 2)))
ax.set_title('Regenerated Figure')

end = 0
for idx, group_name in enumerate(dict_metadata):
    sample_names = dict_metadata[group_name]
    group_name = cross_ref_dict[group_name]
    group_count = len(sample_names)
    start = end
    end = start + group_count

    # multiply by -1 to rotate vector 180 degrees in order to match target fig
ure
    pc1 = df_pcoa.iloc[start:end, 0] * -1.0
    pc2 = df_pcoa.iloc[start:end, 1]

    # plot points
    ax.scatter(pc1, pc2, s=15, c=colors[idx],  label=group_name, alpha=0.5, ed
gecolors='none')
    cov = np.cov(pc1,pc2)
    x_mean = pc1.mean()
    y_mean = pc2.mean()
    e = get_cov_ellipse(cov, (x_mean, y_mean), 1,
                        ec=colors[idx], linewidth=2.0, alpha=0.4)
    ax.text(x_mean + text_tup[idx][0], y_mean + text_tup[idx][1], group_name,
            fontsize='x-small', c=colors[idx])
    ax.add_artist(e)

    for i, sample_name in enumerate(sample_names):
        ax.text(pc1.iloc[i], pc2.iloc[i], sample_cross_ref_dict[sample_name],
                fontsize='x-small', c=colors[idx], horizontalalignment='cente
r',
               verticalalignment='center')
ax.legend(fontsize=6, loc='upper right')
plt.show()
```



Step 8: Compare original and regenerated figures