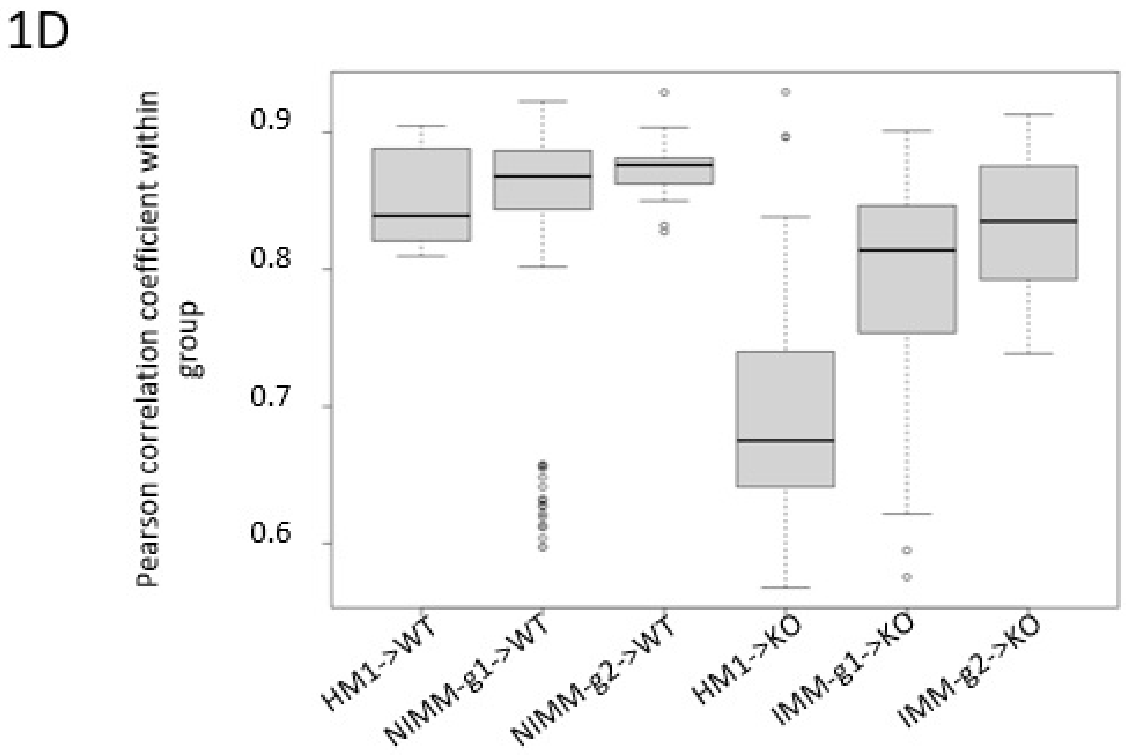
Contents

Aim: Replicate Figure 1D



Main Python Methods

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import re
from scipy import stats
from scipy.special import comb

# ******************************* 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

# Create dictionary from metadata table
def get_dict_from_metadata(input_df):
    mydict = {}
    for row in input_df.iterrows():
        obj = row[1]
        sample_id = obj['SampleID']
        key = obj['FMTGroupFMTsourcegtRecipientbackground']
        if key not in mydict:
            mydict[key] = [sample_id]
        else:
            mydict[key].append(sample_id)
    return mydict

# Calculate Pearson correlation coefficients from input set of samples
def get_pearson_corr_values(input_df, ser_sample_count_sums):
    overall_mean_count = ser_sample_count_sums.mean()
    col_names = input_df.columns
    df_input_norm_logged = _get_norm_counts(input_df, col_names, ser_sample_co
unt_sums, overall_mean_count)
    final_corr_result = _get_corr_values(col_names, df_input_norm_logged)
    return final_corr_result

# ******************************* HELPER METHODS *****************************
**
# Normalize counts table
def _get_norm_counts(input_df, col_names, ser_sample_count_sums, overall_mean_
count):
    df_norm_logged =    pd.DataFrame()
    for col_name in col_names:
        df_norm_logged.loc[:, col_name] = \
        np.log10(((input_df.loc[:, col_name] / ser_sample_count_sums[col_name
]) * overall_mean_count) + 1)

    return df_norm_logged

# Calculate Pearson Correlation Coefficients
def _get_corr_values(col_names, input_norm_logged_df):
    corr_result = []
    tup_list = _get_tup_list(col_names)

    for tup in tup_list:
        x_col_name = col_names[tup[0]]
        y_col_name = col_names[tup[1]]
        rho_value = \
        stats.pearsonr(input_norm_logged_df.loc[:, x_col_name], input_norm_log
ged_df.loc[:, y_col_name]).statistic
        corr_result.append(rho_value)

    # Remove nan values
    corr_result = [val for val in corr_result if not np.isnan(val)]

    return corr_result

# Generate tuple list of sample column names
```

```
def _get_tup_list(col_names):
    tup_list = []
    corr_result = []
    col_name_length = len(col_names)
    for i in range(col_name_length):
        for j in range(i+1, col_name_length):
            tup_list.append((i, j))

    return tup_list

# ******************************* INPUT FILE PATHS ******************************
# Get current working directory
current_working_dir = os.getcwd()
# original counts table
asv_tbl_file_path = os.path.join(current_working_dir, 'asv_biom-with-taxonomy.txt')
# original metadata table
metadata_file_path = os.path.join(current_working_dir, 'mappingMetadata.txt')
```

Data Processing Steps

Step 1: Retrieve original counts table

In [2]:

```
df_asv = read_csv_file(asv_tbl_file_path, 1)
df_asv = df_asv.astype({col:'int32' for col in df_asv.columns[1:-1] }, copy=False)
df_asv
```

Out[2]:

	#OTU ID	1gKO.1	1gKO.2	1gKO.3	1gWT.1	1gWT.2	1gWT.3	2gKO.1
0	1ba8c796d07406783c96d016a6a5cace	13615	16637	17148	20227	23630	25656	14832
1	a6c38249aff7768283faf6cfbdeb05a8	26439	30129	19743	8955	10759	7074	18489
2	062f38ff92cfaee0654200b6f5be5ddf	7451	8774	8754	174	214	148	21958
3	1183cc23f552d81e63c93ca9fcba2f2c	225	223	184	13762	16856	18692	269
4	5e15ecfb579e72bf87c0bea3920bbf42	10108	12117	8633	10027	11910	7424	5979
...	...	...	...	...	...	...	...	...
4070	92bb8f4683ef5c8651e7d34dbb37ab2e	0	0	0	0	0	0	0
4071	92f09070a4fd5786bb34e756217e6ee1	0	0	0	0	0	0	0
4072	919b82324c41ed0046323c63aa1550da	0	0	0	0	0	0	0
4073	dbc0dad15ec1c8ad9d826cab94e18696	0	0	0	0	0	0	0

4074	1ff2d07d10264c23dc43e08d3097cd7c	0	0	0	0	0	0	0

4075 rows × 112 columns

Step 2: Retrieve original metadata table

In [3]:

```
df_metadata = read_csv_file(metadata_file_path)
df_metadata
```

Out[3]:

	SampleID	UniversalCageNumber	Background	FMTGroupFMTsourcegtRecipientbackground	Passage
0	F8-1	F8-cage-1	129.IL10KO	1gKOgtKO	8
1	F8-2	F8-cage-1	129.IL10KO	1gKOgtKO	8
2	F8-3	F8-cage-2	129.IL10KO	1gKOgtKO	8
3	F8-4	F8-cage-2	129.IL10KO	1gKOgtKO	8
4	F8-5	F8-cage-3	129.IL10KO	1gKOgtKO	8
...	...	...	...	...	...
105	1gWT.2	NaN	NaN	1gWTinput	1gWT
106	1gWT.3	NaN	NaN	1gWTinput	1gWT
107	2gWT.1	NaN	NaN	2gWTinput	2gWT
108	2gWT.2	NaN	NaN	2gWTinput	2gWT
109	2gWT.3	NaN	NaN	2gWTinput	2gWT

110 rows × 8 columns

Step 3: Create dictionary from metadata table

In [4]:

```
dict_metadata = get_dict_from_metadata(df_metadata)
dict_metadata.keys()
```

Out[4]:

```
dict_keys(['1gKOgtKO', '2gKOgtKO', '1gWTgtKO', '1gWTgtWT', '2gWTgtWT', 'hFM
T.1.2.3.gtKO', 'hFMT.3.4.5.gtKO', 'hFMT.1.2.3.gtWT', 'hFMT.1.2.3.input', 'hF
MT.3.4.5.input', '1gKOinput', '2gKOinput', '1gWTinput', '2gWTinput'])
```

Step 4: Calculate total read counts per sample

In [5]:

```
sample_count_sums = df_asv.iloc[:, 1:-1].sum(axis=0)
sample_count_sums
```

Out[5]:

```
1gKO.1       118256
1gKO.2       141891
1gKO.3       123292
1gWT.1       119717
1gWT.2       146158
              ...
h1-2-3.2     135326
h1-2-3.3     133745
```

```
h3-4-5.1     129613
h3-4-5.2     140316
h3-4-5.3     132984
Length: 110, dtype: int64
```

Step 5: Extract counts for all six sample groups

In [6]:

```
# 'hFMT.1.2.3.gtWT' -->  'HM1->WT'
key_name = 'hFMT.1.2.3.gtWT'
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_met
adata[key_name]), 2))))
HM1_WT = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
HM1_WT
```

```
Grouped columns: ['F1-16', 'F1-17', 'F1-18', 'F1-19', 'F1-20', 'F1-21', 'F1-
22']
Number of expected correlation values: 21
```

Out[6]:

	F1-16	F1-17	F1-18	F1-19	F1-20	F1-21	F1-22
0	19186	10727	29599	32588	32950	18853	5199
1	2938	6816	3233	1118	1710	1047	2202
2	9072	10348	5289	176	196	207	269
3	247	258	275	18297	22544	19687	21014
4	2389	3858	1904	3904	2928	5286	4884
...	...	...	...	...	...	...	...
4070	0	0	0	0	0	0	0
4071	0	0	0	0	0	0	0
4072	0	0	0	0	0	0	0
4073	0	0	0	0	0	0	0
4074	0	0	0	0	0	0	0

4075 rows × 7 columns

In [7]:

```
# '1gWTgtWT' --> 'NIMM-g1->WT'
key_name = '1gWTgtWT'
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_met
adata[key_name]), 2))))

NIMM_g1_WT = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
NIMM_g1_WT
```

```
Grouped columns: ['F8-27', 'F8-28', 'F8-29', 'F8-30', 'F8-31', 'F8-32', 'F8-
33', 'F11-1', 'F11-2', 'F11-3', 'F11-4', 'F11-5', 'F11-6', 'F11-7', 'F11-8',
'F11-9', 'F11-10', 'F11-11']
Number of expected correlation values: 153
```

Out[7]:

	F8-27	F8-28	F8-29	F8-30	F8-31	F8-32	F8-33	F11-1	F11-2	F11-3	F11-4	F11-5	F11-6	F
0	21313	6113	24573	19347	18028	10482	11761	34742	24360	31173	45582	18929	22349	1

1	300	972	397	332	573	825	1995	410	434	376	376	370	337	
2	135	243	197	193	99	207	237	129	291	167	282	412	190	
3	21071	16272	27247	21302	26135	23762	22465	25848	26231	19490	22181	18524	12915	2
4	2245	5227	5760	1189	2327	14279	7911	1416	2031	1899	2725	2679	1920	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4070	0	0	0	0	0	0	0	0	0	0	0	0	0	
4071	0	0	0	0	0	0	0	0	0	0	0	0	0	
4072	0	0	0	0	0	0	0	0	0	0	0	0	0	
4073	0	0	0	0	0	0	0	0	0	0	0	0	0	
4074	0	0	0	0	0	0	0	0	0	0	0	0	0	

4075 rows × 18 columns

In [8]:

```
# '2gWTgtWT' --> 'NIMM-g2->WT'
key_name = '2gWTgtWT'
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_metadata[key_name]), 2))))
NIMM_g2_WT = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
NIMM_g2_WT
```

```
Grouped columns: ['F8-34', 'F8-35', 'F8-36', 'F8-37', 'F8-38', 'F8-39', 'F8-40', 'F8-41']
Number of expected correlation values: 28
```

Out[8]:

	F8-34	F8-35	F8-36	F8-37	F8-38	F8-39	F8-40	F8-41
0	23089	23716	26435	13911	32356	18934	23958	18944
1	329	537	320	362	473	1848	258	282
2	172	254	238	146	225	135	189	66
3	11455	20177	11798	13910	13422	24886	20107	18889
4	4360	10659	3839	3613	5501	13735	2324	12215
...	...	...	...	...	...	...	...	...
4070	0	0	0	0	0	0	0	0
4071	0	0	0	0	0	0	0	0
4072	0	0	0	0	0	0	0	0
4073	0	0	0	0	0	0	0	0
4074	0	0	0	0	0	0	0	0

4075 rows × 8 columns

In [9]:

```
# 'hFMT.1.2.3.gtKO' --> 'HM1->KO'
key_name = 'hFMT.1.2.3.gtKO'
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_metadata[key_name]), 2))))
HM1_KO = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
HM1_KO
```

```
Grouped columns: ['F3-7', 'F3-8', 'F3-9', 'F3-10', 'F3-11', 'F3-12', 'F1-7',
'F1-8', 'F1-9', 'F1-10', 'F1-11', 'F1-12', 'F1-13', 'F1-14', 'F1-15']
```

```
Number of expected correlation values: 105
```

Out[9]:

	F3-7	F3-8	F3-9	F3-10	F3-11	F3-12	F1-7	F1-8	F1-9	F1-10	F1-11	F1-12	F1-13	F
0	10097	574	12959	18737	17857	12650	10738	9576	7304	14409	14382	8787	7694	
1	35102	51633	31840	22887	20446	33188	23565	21707	12802	26719	5701	15715	25125	
2	1328	91	2203	189	182	244	33299	13739	53077	17570	34094	18922	19204	4
3	137	315	305	134	221	290	203	348	363	238	219	205	255	
4	34960	13573	4109	3811	1155	28938	1144	6973	666	8230	1287	11346	13195	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4070	0	0	0	0	0	0	0	0	0	0	0	0	0	
4071	0	0	0	0	0	0	0	0	0	0	0	0	0	
4072	0	0	0	0	0	0	0	0	0	0	0	0	0	
4073	0	0	0	0	0	0	0	0	0	0	0	0	0	
4074	0	0	0	0	0	0	0	0	0	0	0	0	0	

4075 rows × 15 columns

In [10]:

```python
# '1gKOgtKO' --> 'IMM-g1->KO'
key_name = '1gKOgtKO'
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_metadata[key_name]), 2))))
IMM_g1_KO = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
IMM_g1_KO
```

```
Grouped columns: ['F8-1', 'F8-2', 'F8-3', 'F8-4', 'F8-5', 'F8-6', 'F8-7', 'F
8-8', 'F4-1', 'F4-2', 'F4-3', 'F4-4', 'F4-5', 'F3-1', 'F3-2', 'F3-4', 'F3-
5', 'F3-6']
Number of expected correlation values: 153
```

Out[10]:

	F8-1	F8-2	F8-3	F8-4	F8-5	F8-6	F8-7	F8-8	F4-1	F4-2	F4-3	F4-4	F4-5	
0	3339	15223	11423	18205	12486	10761	9339	466	29629	20134	27588	26741	25644	2
1	10283	6602	9314	13651	8789	16118	12045	13841	3687	6998	6904	7796	4011	
2	41532	40602	52337	11038	16781	8926	25758	31166	1627	37042	23825	26508	30376	
3	240	276	331	302	341	231	226	306	275	198	213	271	172	2
4	1806	6778	6725	12734	3901	10948	10347	1234	8891	3718	9188	4419	8315	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4070	0	0	0	0	0	0	0	0	0	0	0	0	0	
4071	0	0	0	0	0	0	0	0	0	0	0	0	0	
4072	0	0	0	0	0	0	0	0	0	0	0	0	0	
4073	0	0	0	0	0	0	0	0	0	0	0	0	0	
4074	0	0	0	0	0	0	0	0	0	0	0	0	0	

4075 rows × 18 columns

In [11]:

```python
# '2gKOgtKO' --> 'IMM-g2->KO'
key_name = '2gKOgtKO'
```

```
print('Grouped columns: {}'.format(dict_metadata[key_name]))
print('Number of expected correlation values: {}'.format(int(comb(len(dict_met
adata[key_name]), 2))))
IMM_g2_KO = df_asv[[col_name for col_name in dict_metadata[key_name] ]]
IMM_g2_KO
```

```
Grouped columns: ['F8-9', 'F8-10', 'F8-11', 'F8-12', 'F8-13', 'F8-14', 'F8-1
5', 'F8-16']
Number of expected correlation values: 28
```

Out[11]:

	F8-9	F8-10	F8-11	F8-12	F8-13	F8-14	F8-15	F8-16
0	23451	24742	31312	13542	4593	10228	14477	23819
1	8030	14203	27872	26539	12663	13720	23316	14430
2	19653	22616	4603	28535	20096	20133	14650	24062
3	266	344	300	269	257	330	249	226
4	6490	6143	12353	2229	6367	11645	9607	10622
...	...	...	...	...	...	...	...	...
4070	0	0	0	0	0	0	0	0
4071	0	0	0	0	0	0	0	0
4072	0	0	0	0	0	0	0	0
4073	0	0	0	0	0	0	0	0
4074	0	0	0	0	0	0	0	0

4075 rows × 8 columns

Step 6: Calculate Spearman Correlation Coefficients for each sample group

In [12]:

```
HM1_WT_data = get_pearson_corr_values(HM1_WT, sample_count_sums)
NIMM_g1_WT_data = get_pearson_corr_values(NIMM_g1_WT, sample_count_sums)
NIMM_g2_WT_data = get_pearson_corr_values(NIMM_g2_WT, sample_count_sums)
HM1_KO_data = get_pearson_corr_values(HM1_KO, sample_count_sums)
IMM_g1_KO_data = get_pearson_corr_values(IMM_g1_KO, sample_count_sums)
IMM_g2_KO_data = get_pearson_corr_values(IMM_g2_KO, sample_count_sums)
```

Step 7: Generate final figure

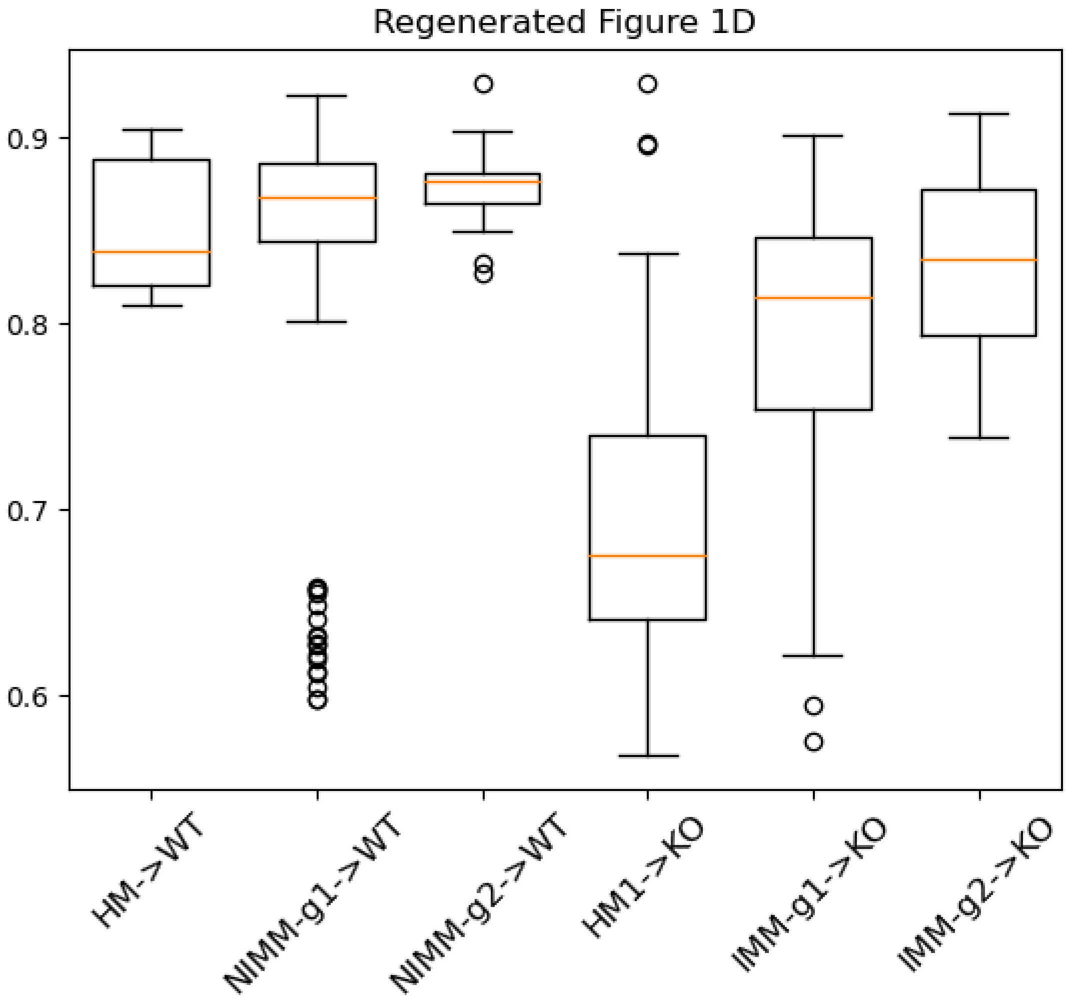
In [13]:

```
data = [HM1_WT_data,
        NIMM_g1_WT_data,
        NIMM_g2_WT_data,
        HM1_KO_data,
        IMM_g1_KO_data,
       IMM_g2_KO_data]

sample_groups = ['HM->WT',
                'NIMM-g1->WT',
                 'NIMM-g2->WT',
                 'HM1->KO',
                 'IMM-g1->KO',
                 'IMM-g2->KO']
```

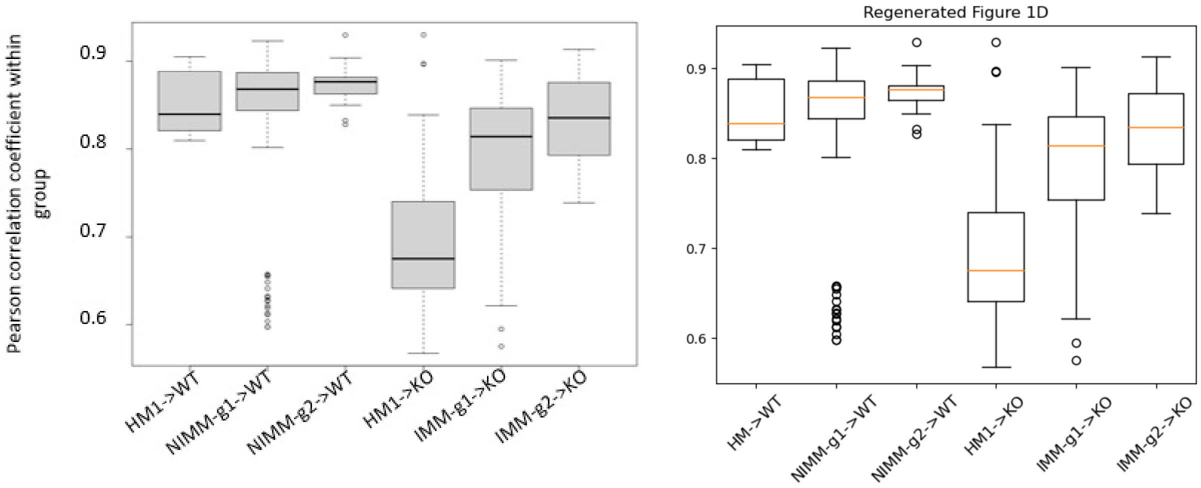
```
# Multiple box plots on one Axes
fig, ax = plt.subplots()
ax.boxplot(data, widths=0.7)
ax.set_title('Regenerated Figure 1D')
ax.set_yticks([0.6, 0.7, 0.8, 0.9])
ax.set_xticklabels(sample_groups,
                   rotation=45, fontsize=12)

plt.show()
```



Step 8: Compare original and regenerated figures

1D


Pearson correlation coefficient within group
HM1->WT
NIMM-g1->WT
NIMM-g2->WT
HM1->KO
IMM-g1->KO
IMM-g2->KO
Regenerated Figure 1D
HM->WT
NIMM-g1->WT
NIMM-g2->WT
HM1->KO
IMM-g1->KO
IMM-g2->KO