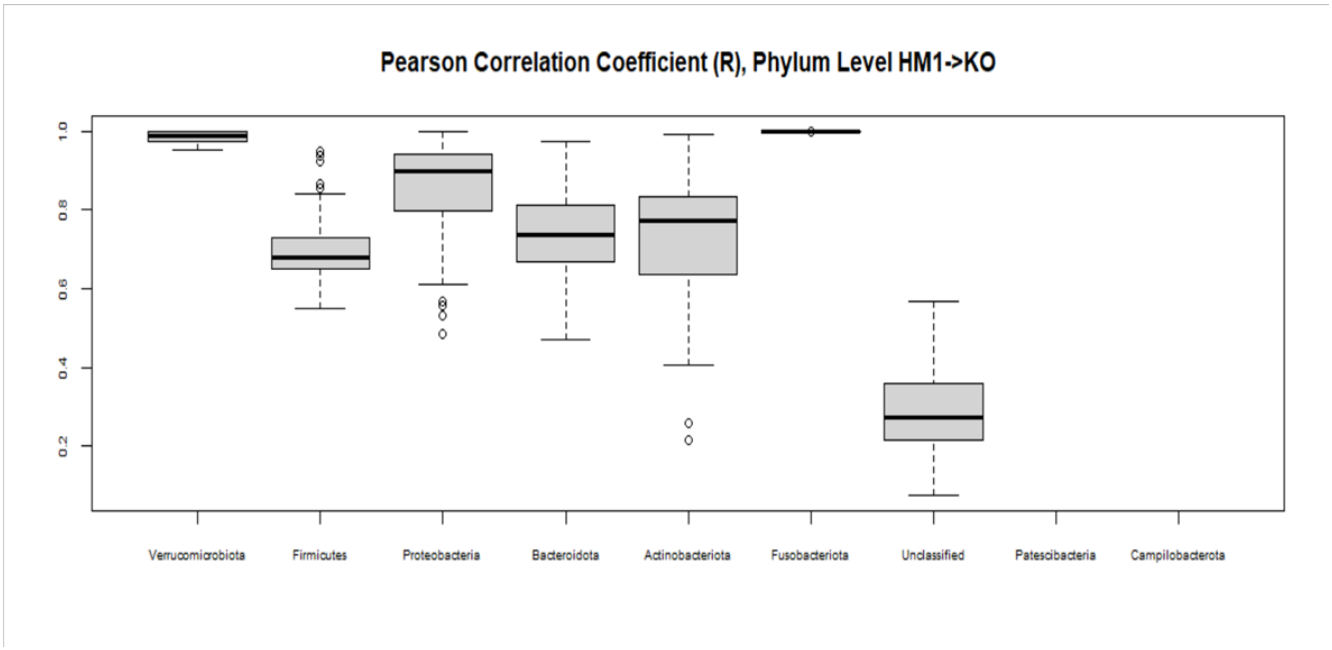
Contents

Aim: Replicate Figure 5D



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 ******************************
# Filter rows for target taxa
def filter_rows(taxa_query, taxa_target):
    final_result = False
```

```
    search_result = re.search(taxa_target, taxa_query)
    if search_result:
        final_result = True

    return final_result
# Calculate Pearson correlation coefficients from input set of samples
def get_inter_corr_values(input_df):
    tup_list_column_names = [(col1, col2) for col1 in input_df.columns for col
2 in input_df.columns if input_df.columns.get_loc(col2) > input_df.columns.get
_loc(col1)]
    corr_values = [stats.pearsonr(input_df.loc[:, tup[0]], input_df.loc[:, tup
[1]]).statistic \
            for tup in tup_list_column_names]
    # Remove nan values
    corr_values = [val for val in corr_values if not np.isnan(val)]
    return corr_values

# 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

# ****************************** HELPER METHODS ****************************
**
# Normalize sample count values
def _get_norm_counts(input_df, ser_sample_count_sums):
    overall_mean_count = ser_sample_count_sums.mean()
    df_norm_logged =    pd.DataFrame()
    for col_name in input_df.columns:
        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

# ****************************** 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
```

Out[2]:

	#OTU ID	1gKO.1	1gKO.2	1gKO.3	1gWT.1	1gWT.2	1gWT.3	2gKO.1
0	1ba8c796d07406783c96d016a6a5cace	13615.0	16637.0	17148.0	20227.0	23630.0	25656.0	14832.0
1	a6c38249aff7768283faf6cfbdeb05a8	26439.0	30129.0	19743.0	8955.0	10759.0	7074.0	18489.0
2	062f38ff92cfaee0654200b6f5be5ddf	7451.0	8774.0	8754.0	174.0	214.0	148.0	21958.0
3	1183cc23f552d81e63c93ca9fcba2f2c	225.0	223.0	184.0	13762.0	16856.0	18692.0	269.0
4	5e15ecfb579e72bf87c0bea3920bbf42	10108.0	12117.0	8633.0	10027.0	11910.0	7424.0	5979.0
...	...	...	...	...	...	...	...	...
4070	92bb8f4683ef5c8651e7d34dbb37ab2e	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4071	92f09070a4fd5786bb34e756217e6ee1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4072	919b82324c41ed0046323c63aa1550da	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4073	dbc0dad15ec1c8ad9d826cab94e18696	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4074	1ff2d07d10264c23dc43e08d3097cd7c	0.0	0.0	0.0	0.0	0.0	0.0	0.0

4075 rows × 112 columns

In [3]:
```
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[3]:

	#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 [4]:
```
df_metadata = read_csv_file(metadata_file_path)
df_metadata
```

Out[4]:

	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 [5]:
```
dict_metadata = get_dict_from_metadata(df_metadata)
dict_metadata.keys()
```

Out[5]:
```
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 [6]:
```
sample_count_sums = df_asv.iloc[:, 1:-1].sum(axis=0)
sample_count_sums
```

Out[6]:
```
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 sample group HM1->KO

In [7]:
```
# 'hFMT.1.2.3.gtKO' -->  'HM1->KO'
key_name = 'hFMT.1.2.3.gtKO'
group_columns = dict_metadata[key_name]
group_columns.extend(['#OTU ID', 'taxonomy'])
key_name = 'hFMT.1.2.3.gtKO'
HM1_KO = df_asv[[col_name for col_name in group_columns ]]
HM1_KO
```

Out[7]:

	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 × 17 columns

In [8]:
```
target_phylum = 'p__Verrucomicrobiota'
df_Verrucomicrobiota = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, t
arget_phylum))]
print('Number of rows: {}'.format(len(df_Verrucomicrobiota.index)))
print('Columns: {}'.format(', '.join(df_Verrucomicrobiota.columns)))
```

```
Number of rows: 24
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, #OTU ID, taxonomy
```

In [9]:
```
target_phylum = 'p__Firmicutes'
df_Firmicutes = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, target_p
hylum))]
print('Number of rows: {}'.format(len(df_Firmicutes.index)))
print('Columns: {}'.format(', '.join(df_Firmicutes.columns)))
```

```
Number of rows: 830
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, #OTU ID, taxonomy
```

In [10]:
```
target_phylum = 'p__Proteobacteria'
df_Proteobacteria = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, targ
et_phylum))]
print('Number of rows: {}'.format(len(df_Proteobacteria.index)))
print('Columns: {}'.format(', '.join(df_Proteobacteria.columns)))
```

```
Number of rows: 51
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, #OTU ID, taxonomy
```

In [11]:
```
target_phylum = 'p__Bacteroidota'
df_Bacteroidota = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, target
_phylum))]
print('Number of rows: {}'.format(len(df_Bacteroidota.index)))
print('Columns: {}'.format(', '.join(df_Bacteroidota.columns)))
```

```
Number of rows: 91
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, #OTU ID, taxonomy
```

In [12]:
```
target_phylum = 'p__Actinobacteriota'
df_Actinobacteriota = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, ta
rget_phylum))]
print('Number of rows: {}'.format(len(df_Actinobacteriota.index)))
print('Columns: {}'.format(', '.join(df_Actinobacteriota.columns)))
```

```
Number of rows: 71
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, #OTU ID, taxonomy
```

In [13]:
```
target_phylum = 'p__Fusobacteriota'
df_Fusobacteriota = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, targ
et_phylum))]
print('Number of rows: {}'.format(len(df_Fusobacteriota.index)))
print('Columns: {}'.format(', '.join(df_Fusobacteriota.columns)))
```

```
Number of rows: 11
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, #OTU ID, taxonomy
```

In [14]:
```
target_phylum = 'p__Patescibacteria'
df_Patescibacteria = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, tar
get_phylum))]
print('Number of rows: {}'.format(len(df_Patescibacteria.index)))
print('Columns: {}'.format(', '.join(df_Patescibacteria.columns)))
```

```
Number of rows: 3
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, #OTU ID, taxonomy
```

In [15]:
```
target_phylum = 'p__Campilobacterota'
df_Campilobacterota = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, ta
rget_phylum))]
print('Number of rows: {}'.format(len(df_Campilobacterota.index)))
print('Columns: {}'.format(', '.join(df_Campilobacterota.columns)))
```

```
Number of rows: 3
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, #OTU ID, taxonomy
```

In [16]:
```
target_phylum = 'Unassigned'
df_Unassigned = HM1_KO[HM1_KO.taxonomy.apply(lambda x: filter_rows(x, target_p
hylum))]
print('Number of rows: {:,}'.format(len(df_Unassigned.index)))
print('Columns: {}'.format(', '.join(df_Unassigned.columns)))
```

```
Number of rows: 2,196
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, #OTU ID, taxonomy
```

Step 6: Normalize count values for sample group HM1->KO

In [17]:
```
df_Verrucomicrobiota_norm = _get_norm_counts(df_Verrucomicrobiota.iloc[:, :-2
], sample_count_sums)
print('Number of rows: {}'.format(len(df_Verrucomicrobiota_norm.index)))
print('Columns: {}'.format(', '.join(df_Verrucomicrobiota_norm.columns)))
```

```
Number of rows: 24
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
```

In [18]:
```
df_Firmicutes_norm = _get_norm_counts(df_Firmicutes.iloc[:, :-2], sample_count
_sums)
print('Number of rows: {}'.format(len(df_Firmicutes_norm.index)))
print('Columns: {}'.format(', '.join(df_Firmicutes_norm.columns)))
```

```
Number of rows: 830
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
```

In [19]:
```
df_Proteobacteria_norm = _get_norm_counts(df_Proteobacteria.iloc[:, :-2], samp
le_count_sums)
```

```
print('Number of rows: {}'.format(len(df_Proteobacteria_norm.index)))
print('Columns: {}'.format(', '.join(df_Proteobacteria_norm.columns)))
```

```
Number of rows: 51
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
```

In [20]:
```
df_Bacteroidota_norm = _get_norm_counts(df_Bacteroidota.iloc[:, :-2], sample_c
ount_sums)
print('Number of rows: {}'.format(len(df_Bacteroidota_norm.index)))
print('Columns: {}'.format(', '.join(df_Bacteroidota_norm.columns)))
```

```
Number of rows: 91
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
```

In [21]:
```
df_Actinobacteriota_norm = _get_norm_counts(df_Actinobacteriota.iloc[:, :-2],
sample_count_sums)
print('Number of rows: {}'.format(len(df_Actinobacteriota_norm.index)))
print('Columns: {}'.format(', '.join(df_Actinobacteriota_norm.columns)))
```

```
Number of rows: 71
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
```

In [22]:
```
df_Fusobacteriota_norm = _get_norm_counts(df_Fusobacteriota.iloc[:, :-2], samp
le_count_sums)
print('Number of rows: {}'.format(len(df_Fusobacteriota_norm.index)))
print('Columns: {}'.format(', '.join(df_Fusobacteriota_norm.columns)))
```

```
Number of rows: 11
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
```

In [23]:
```
df_Unassigned_norm = _get_norm_counts(df_Unassigned.iloc[:, :-2], sample_count
_sums)
print('Number of rows: {:,}'.format(len(df_Unassigned_norm.index)))
print('Columns: {}'.format(', '.join(df_Unassigned_norm.columns)))
```

```
Number of rows: 2,196
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
```

In [24]:
```
df_Patescibacteria_norm = _get_norm_counts(df_Patescibacteria.iloc[:, :-2], sa
mple_count_sums)
print('Number of rows: {:,}'.format(len(df_Patescibacteria_norm.index)))
print('Columns: {}'.format(', '.join(df_Patescibacteria_norm.columns)))
```

```
Number of rows: 3
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
```

In [25]:
```
df_Campilobacterota_norm = _get_norm_counts(df_Campilobacterota.iloc[:, :-2],
sample_count_sums)
print('Number of rows: {:,}'.format(len(df_Campilobacterota_norm.index)))
print('Columns: {}'.format(', '.join(df_Campilobacterota_norm.columns)))
```

```
Number of rows: 3

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
```

Step 7: Calculate Spearman Correlation Coefficients for each sample pair

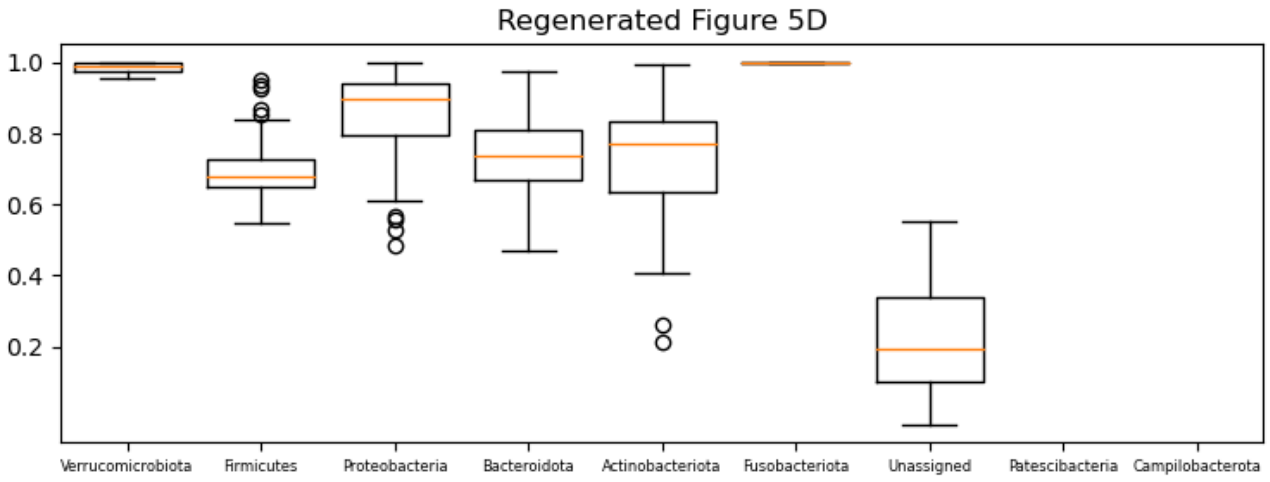
In [26]:
```
corr_Verrucomicrobiota_norm = get_inter_corr_values(df_Verrucomicrobiota_norm)
norm_Firmicutes_norm = get_inter_corr_values(df_Firmicutes_norm)
corr_Proteobacteria_norm = get_inter_corr_values(df_Proteobacteria_norm)
corr_Bacteroidota_norm = get_inter_corr_values(df_Bacteroidota_norm)
corr_Actinobacteriota_norm = get_inter_corr_values(df_Actinobacteriota_norm)
corr_Fusobacteriota_norm = get_inter_corr_values(df_Fusobacteriota_norm)
corr_Unassigned_norm = get_inter_corr_values(df_Unassigned_norm)
corr_Patescibacteria_norm = get_inter_corr_values(df_Patescibacteria_norm)
corr_Campilobacterota_norm = get_inter_corr_values(df_Campilobacterota_norm)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_stats_py.py:4424: Co
nstantInputWarning: An input array is constant; the correlation coefficient
is not defined.
  warnings.warn(stats.ConstantInputWarning(msg))
```

Step 8: Generate final figure

In [27]:
```
data = [corr_Verrucomicrobiota_norm,
            norm_Firmicutes_norm,
            corr_Proteobacteria_norm,
            corr_Bacteroidota_norm,
            corr_Actinobacteriota_norm,
            corr_Fusobacteriota_norm,
            corr_Unassigned_norm,
            corr_Patescibacteria_norm,
            corr_Campilobacterota_norm]

x_tick_labels = ['Verrucomicrobiota',
                 'Firmicutes',
                 'Proteobacteria',
                 'Bacteroidota',
                 'Actinobacteriota',
                 'Fusobacteriota',
                 'Unassigned',
                 'Patescibacteria',
                 'Campilobacterota']
fig, ax = plt.subplots(figsize=(9, 3))
ax.boxplot(data, widths=0.8)
ax.set_title('Regenerated Figure 5D')
ax.set_yticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax.set_xticklabels(x_tick_labels,
                   rotation=0, fontsize=6)
plt.show()
```



Step 9: Compare original and regenerated figures