

```
In [48]: def install_reqs():
#This defines a function named install_reqs.

!pip install pandas
#Installs the pandas library, used for reading and manipulating tabular data (like .csv, .tsv).

!pip install "matplotlib>=3.4"
#Installs matplotlib version 3.4 or higher, used for plotting graphs and visualizations.

!pip install numpy
#Installs numpy, a fundamental package for numerical computations and arrays.

!pip install statsmodels
#Installs statsmodels, a library used for statistical tests (e.g., t-tests, Wilcoxon), regression, and model fitting.

!pip install scipy
#Installs scipy, which contains scientific computing tools, including statistical and optimization functions.

install_reqs()
# to install all the above dependencies uncomment the above line.
```

```
Requirement already satisfied: pandas in c:\users\user\anaconda3\lib\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2.1.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\anaconda3\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib>=3.4 in c:\users\user\anaconda3\lib\site-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (1.4.8)
Requirement already satisfied: numpy>=1.23 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (2.1.3)
Requirement already satisfied: packaging>=20.0 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\anaconda3\lib\site-packages (from matplotlib>=3.4) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.4) (1.17.0)
Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-packages (2.1.3)
Requirement already satisfied: statsmodels in c:\users\user\anaconda3\lib\site-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in c:\users\user\anaconda3\lib\site-packages (from statsmodels) (2.1.3)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in c:\users\user\anaconda3\lib\site-packages (from statsmodels) (1.15.3)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in c:\users\user\anaconda3\lib\site-packages (from statsmodels) (2.2.3)
Requirement already satisfied: patsy>=0.5.6 in c:\users\user\anaconda3\lib\site-packages (from statsmodels) (1.0.1)
Requirement already satisfied: packaging>=21.3 in c:\users\user\anaconda3\lib\site-packages (from statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.17.0)
Requirement already satisfied: scipy in c:\users\user\anaconda3\lib\site-packages (1.15.3)
Requirement already satisfied: numpy<2.5,>=1.23.5 in c:\users\user\anaconda3\lib\site-packages (from scipy) (2.1.3)
```

```
In [49]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [50]: from compass_analysis import cohens_d, wilcoxon_test, get_reaction_consistencies, get_metareactions, labeled_reactions, amino_acid_metab
```

```
In [51]: from matplotlib import __version__ as matplotlibversion
if matplotlibversion < "3.4":
    print("Matplotlib versions older than 3.4 may not be able to generate figure 2E, as they do not support alpha arrays")
```

Matplotlib versions older than 3.4 may not be able to generate figure 2E, as they do not support alpha arrays

```
In [52]: #import os
#os.system(
#    "compass --data expression.tsv "
#    "--model RECON2_mat --species homo_sapiens --media default-media --lambda 0.25 "
#    "--and-function mean --output-dir extdata/APOB --penalty-diffusion knn --num-neighbors 10 "
#    "--isoform-summing legacy --num-processes 50"
#)
```

```
In [53]: import os
print(os.getcwd())
```

C:\Users\user\Desktop\Harshini\Human dataset_APOB_AIM\3_C vs D

```
In [54]: reaction_penalties = pd.read_csv("reactions.tsv", sep="\t", index_col=0)
```

```
In [55]: import pandas as pd

# Load reactions.tsv
reaction_penalties = pd.read_csv("reactions.tsv", sep="\t", index_col=0)

# Extract cell IDs from the columns
cell_ids = reaction_penalties.columns

# --- NEW FUNCTION: infer group based on suffix after "_" ---
def infer_group(cell_id):
    """
    Extracts the group label from a cell name like '1_A' or '15_C'.
    """
    if "_" in cell_id:
        return cell_id.split("_")[-1]  # A, B, C, or D
    else:
        return "Unknown"

# Create list of inferred groups for each cell
cell_groups = [infer_group(cell) for cell in cell_ids]

# Create metadata DataFrame
cell_metadata = pd.DataFrame({
    "cell_id": cell_ids,
    "group": cell_groups
}).set_index("cell_id")

# Save file
cell_metadata.to_csv("cell_metadata.csv")

print("Created cell_metadata.csv with the following group counts:")
print(cell_metadata["group"].value_counts())
```

Created cell_metadata.csv with the following group counts:

```
group
D    38
A    37
B    35
C    34
Name: count, dtype: int64
```

```
In [56]: cell_metadata = pd.read_csv("cell_metadata.csv", sep=",")
cell_metadata.set_index("cell_id", inplace=True)
```

```
In [57]: print(cell_metadata.columns.tolist())
print(cell_metadata.head())
```

```
['group']
      group
cell_id
1_A      A
2_A      A
3_A      A
4_A      A
5_A      A
```

```
In [58]: # Select cells by group
A_cells = cell_metadata.index[cell_metadata["group"] == "A"]
B_cells = cell_metadata.index[cell_metadata["group"] == "B"]
C_cells = cell_metadata.index[cell_metadata["group"] == "C"]
D_cells = cell_metadata.index[cell_metadata["group"] == "D"]

# Optional: print counts
print("Number of cells per group:")
print("A:", len(A_cells))
print("B:", len(B_cells))
print("C:", len(C_cells))
print("D:", len(D_cells))
```

```
Number of cells per group:
A: 37
B: 35
C: 34
D: 38
```

```
In [59]: print(cell_metadata.columns.tolist())
```

```
['group']
```

```
In [60]: print(cell_metadata.head())
print(cell_metadata.columns.tolist())
```

```
      group
cell_id
1_A      A
2_A      A
3_A      A
4_A      A
5_A      A
['group']
```

```
In [61]: reaction_metadata = pd.read_csv("reaction_metadata.csv", index_col = 0)
```

```
In [62]: reaction_metadata.loc[['r0281']]
```

Out[62]:

	reaction_name	formula	associated_genes	subsystem	EC_number	confidence
reaction_no_direction						
r0281	Putrescine:oxygen oxidoreductase (deaminating)...	1.00 * Water [e] + 1.00 * O2 [e] + 1.00 * Putr...	AOC1	Methionine and cysteine metabolism	1.4.3.6	0.0

```
In [63]: #This function is repeated here for clarity
def get_reaction_consistencies(compass_reaction_penalties, min_range=1e-3):
    """
    Converts the raw penalties outputs of compass into scores per reactions where higher numbers indicate more activity
    """
    df = -np.log(compass_reaction_penalties + 1)
    df = df[df.max(axis=1) - df.min(axis=1) >= min_range]
    df = df - df.min().min()
    return df
```

```
In [64]: reaction_consistencies = get_reaction_consistencies(reaction_penalties)
```

```
In [65]: common_cells = list(set(reaction_consistencies.columns).intersection(set(cell_metadata.index)))
reaction_consistencies = reaction_consistencies[common_cells]
```

```
In [66]: # Select cells for groups C and D
C_cells = cell_metadata.index[cell_metadata["group"] == "C"]
D_cells = cell_metadata.index[cell_metadata["group"] == "D"]

# Run Wilcoxon test between C and D
wilcox_results = wilcoxon_test(reaction_consistencies, C_cells, D_cells)

# Add metadata reaction ID column
wilcox_results['metadata_r_id'] = ""

for r in wilcox_results.index:
    if r in reaction_metadata.index:
        wilcox_results.loc[r, 'metadata_r_id'] = r
    elif r[:-4] in reaction_metadata.index: # handles possible suffixes
        wilcox_results.loc[r, 'metadata_r_id'] = r[:-4]
    else:
        print("Should not occur:", r)

# Optional: check the results
print(wilcox_results.head())
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval	\
10FTHF5GLUtl_pos	845.0	0.025154	0.523758	0.108648	
10FTHF5GLUtm_pos	749.0	0.247612	0.150682	0.375004	
10FTHF6GLUtl_pos	844.0	0.025898	0.523491	0.110399	
10FTHF6GLUtm_pos	758.0	0.208505	0.198577	0.340815	
10FTHF7GLUtl_pos	835.0	0.033485	0.505302	0.128149	
	metadata_r_id				
10FTHF5GLUtl_pos	10FTHF5GLUtl				
10FTHF5GLUtm_pos	10FTHF5GLUtm				
10FTHF6GLUtl_pos	10FTHF6GLUtl				
10FTHF6GLUtm_pos	10FTHF6GLUtm				
10FTHF7GLUtl_pos	10FTHF7GLUtl				

```
In [67]: print(reaction_consistencies.columns.tolist()[:10])
```

```
['40_A', '6_C', '5_C', '8_D', '30_B', '9_D', '33_A', '19_C', '40_D', '28_D']
```

```
In [68]: print(C_cells[:10])
```

```
Index(['5_C', '6_C', '7_C', '9_C', '10_C', '11_C', '12_C', '13_C', '14_C',
      '15_C'],
      dtype='object', name='cell_id')
```

```
In [69]: # Select cells for groups C and D
C_cells = cell_metadata.index[cell_metadata["group"] == "C"]
D_cells = cell_metadata.index[cell_metadata["group"] == "D"]

# Run the Wilcoxon test between C and D
wilcox_results = wilcoxon_test(reaction_consistencies, C_cells, D_cells)

# Annotate metadata reaction IDs
wilcox_results['metadata_r_id'] = ""

for r in wilcox_results.index:
    if r in reaction_metadata.index:
        wilcox_results.loc[r, 'metadata_r_id'] = r
    elif r[:-4] in reaction_metadata.index: # handles possible suffixes like '_x'
        wilcox_results.loc[r, 'metadata_r_id'] = r[:-4]
    else:
        print(f"Should not occur → {r}")

# Optional: check the results
print(wilcox_results.head())
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval	\
10FTHF5GLUt1_pos	845.0	0.025154	0.523758	0.108648	
10FTHF5GLUt1_pos	749.0	0.247612	0.150682	0.375004	
10FTHF6GLUt1_pos	844.0	0.025898	0.523491	0.110399	
10FTHF6GLUt1_pos	758.0	0.208505	0.198577	0.340815	
10FTHF7GLUt1_pos	835.0	0.033485	0.505302	0.128149	

	metadata_r_id
10FTHF5GLUt1_pos	10FTHF5GLUt1
10FTHF5GLUt1_pos	10FTHF5GLUt1
10FTHF6GLUt1_pos	10FTHF6GLUt1
10FTHF6GLUt1_pos	10FTHF6GLUt1
10FTHF7GLUt1_pos	10FTHF7GLUt1

```
In [70]: # Select cells for groups C and D
C_cells = cell_metadata.index[cell_metadata["group"] == "C"]
D_cells = cell_metadata.index[cell_metadata["group"] == "D"]

# Run the Wilcoxon test between C and D
wilcox_results = wilcoxon_test(reaction_consistencies, C_cells, D_cells)

# Annotate metadata reaction IDs
wilcox_results['metadata_r_id'] = ""

for r in wilcox_results.index:
    if r in reaction_metadata.index:
        wilcox_results.loc[r, 'metadata_r_id'] = r
    elif r[:-4] in reaction_metadata.index: # handles possible suffixes like '_x'
        wilcox_results.loc[r, 'metadata_r_id'] = r[:-4]
    else:
        print(f"Should not occur → {r}")
```

```
# Optional: view top results
print(wilcox_results.head())
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval	\
10FTHF5GLUt1_pos	845.0	0.025154	0.523758	0.108648	
10FTHF5GLUtm_pos	749.0	0.247612	0.150682	0.375004	
10FTHF6GLUt1_pos	844.0	0.025898	0.523491	0.110399	
10FTHF6GLUtm_pos	758.0	0.208505	0.198577	0.340815	
10FTHF7GLUt1_pos	835.0	0.033485	0.505302	0.128149	

	metadata_r_id
10FTHF5GLUt1_pos	10FTHF5GLUt1
10FTHF5GLUtm_pos	10FTHF5GLUtm
10FTHF6GLUt1_pos	10FTHF6GLUt1
10FTHF6GLUtm_pos	10FTHF6GLUtm
10FTHF7GLUt1_pos	10FTHF7GLUt1

```
In [71]: W = wilcox_results.merge(reaction_metadata, how='left',
                                left_on='metadata_r_id', right_index=True, validate='m:1')
W = W[W['confidence'].isin([0,4])]
W = W[~W['EC_number'].isna()]
W.loc[(W['formula'].map(lambda x: '[m]' not in x)) & (W['subsystem'] == "Citric acid cycle"), 'subsystem'] = 'Other'
```

```
In [72]: wilcox_results.loc[['r0281_pos']]
```

```
Out[72]:
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval	metadata_r_id
r0281_pos	670.0	0.790953	-0.034986	0.835206	r0281

```
In [73]: reaction_metadata.loc['r0281']['formula']
```

```
Out[73]: '1.00 * Water [e] + 1.00 * O2 [e] + 1.00 * Putrescine [e] --> 1.00 * Ammonium [e] + 1.00 * Hydrogen peroxide [e] + 1.00 * 4-Aminobutanal [e]\nAOC1'
```

```
In [74]: def plot_differential_scores(data, title, c):
    plt.figure(figsize=(10,10))
    axs = plt.gca()
    axs.scatter(data['cohens_d'], -np.log10(data['adjusted_pval']), c=c)
    axs.set_xlabel("Cohen's d", fontsize=16)
    axs.set_ylabel("-log10 (Wilcoxon-adjusted p)", fontsize=16)

    # Visual markers and title
    axs.set_xlim(-8, 8)
    axs.set_ylim(0, 6)
    axs.axvline(0, dashes=(3,3), c='black')
    axs.axhline(1, dashes=(3,3), c='black')
    axs.set_title(title, fontdict={'fontsize':20})

    # Arrows for group identity (C vs D)
    axs.annotate('', xy=(0.5, -0.08), xycoords='axes fraction', xytext=(0, -0.08),
                arrowprops=dict(arrowstyle="<-", color='#348C73', linewidth=4))
    axs.annotate('Group C', xy=(0.75, -0.12), xycoords='axes fraction', fontsize=16)
    axs.annotate('', xy=(0.5, -0.08), xycoords='axes fraction', xytext=(1, -0.08),
                arrowprops=dict(arrowstyle="<-", color='#E92E87', linewidth=4))
    axs.annotate('Group D', xy=(0.25, -0.12), xycoords='axes fraction', fontsize=16)

    # Dynamic annotation loop with variable offsets and labels
    for i, r in enumerate(data.index):
        if r in labeled_reactions:
            x = data.loc[r, 'cohens_d']
```

```

y = -np.log10(data.loc[r, 'adjusted_pval'])

dx = 30 if x >= 0 else -120 # horizontal offset
dy = (i % 6) * 12 - 30     # vertical offset varies to reduce stacking

axs.annotate(
    labeled_reactions[r],
    (x, y),
    xytext=(dx, dy),
    textcoords='offset pixels',
    arrowprops=dict(arrowstyle="-", shrinkA=6, shrinkB=4),
    fontsize=10,
    zorder=5, # make sure text is on top
    bbox=dict(boxstyle="round,pad=0.3", fc="white", ec="gray", alpha=0.7)
)

```

```

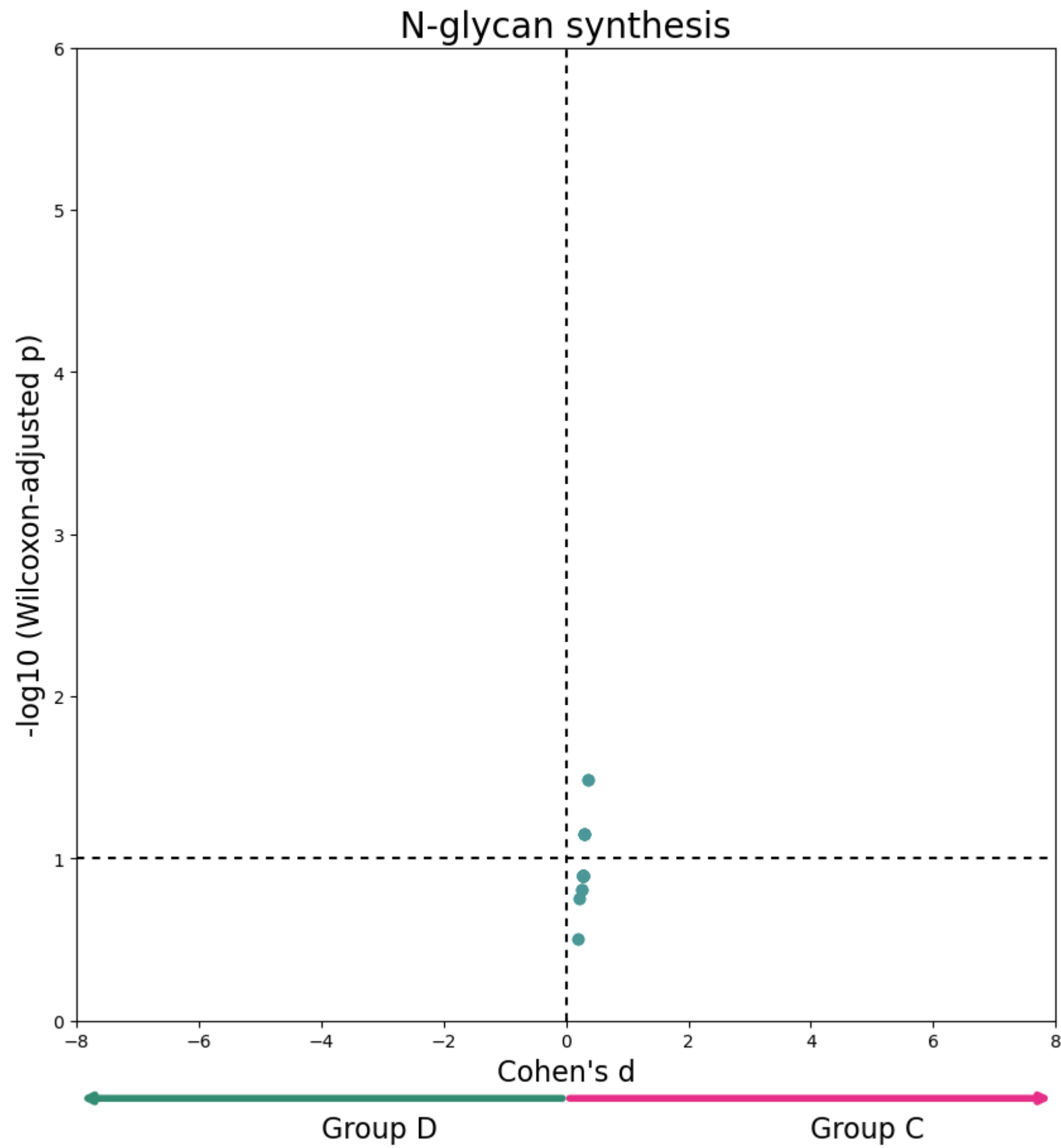
In [75]: filtered_data = pd.concat([
    W[W['subsystem'] == "Glycolysis/gluconeogenesis"],
    W[W['subsystem'] == "Citric acid cycle"],
    W[W['subsystem'].isin(amino_acid_metab)],
    W[W['subsystem'] == "N-glycan synthesis"],
    W[W['subsystem'] == "Fatty acid oxidation"]
])

```

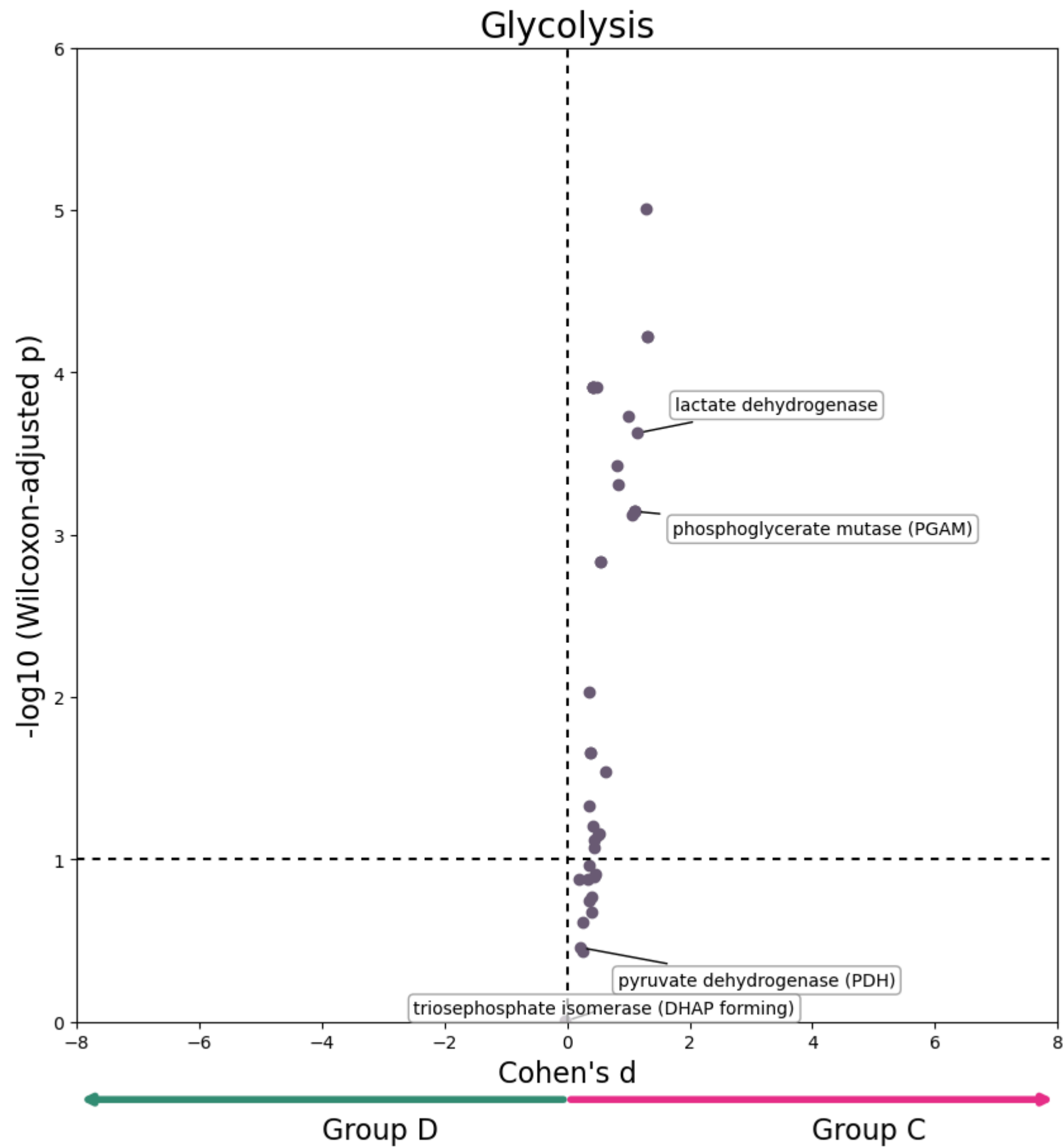
```

In [76]: data = W[W['subsystem'] == "N-glycan synthesis"]
plot_differential_scores(data, title='N-glycan synthesis', c="#4C9A9A")

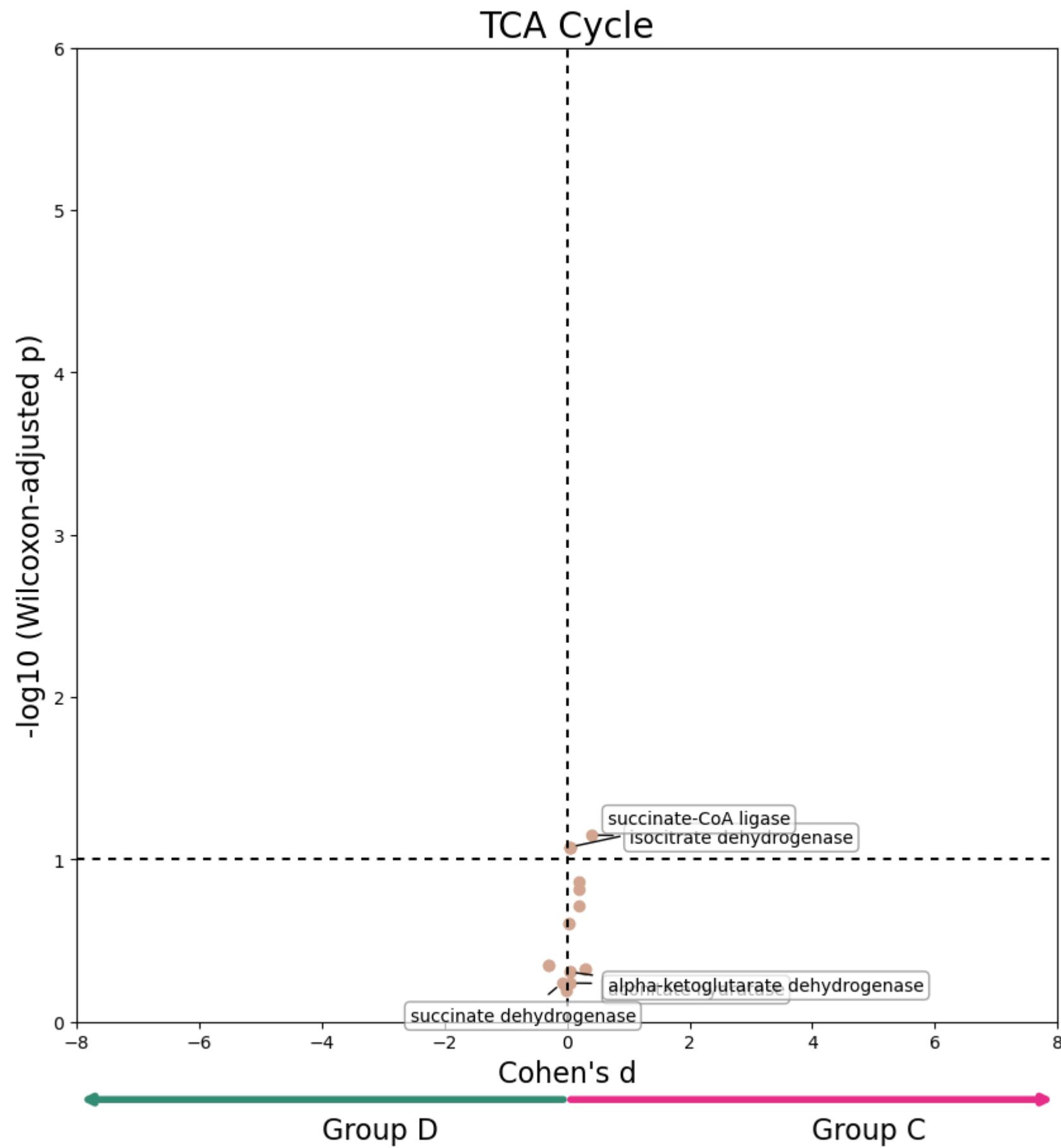
```



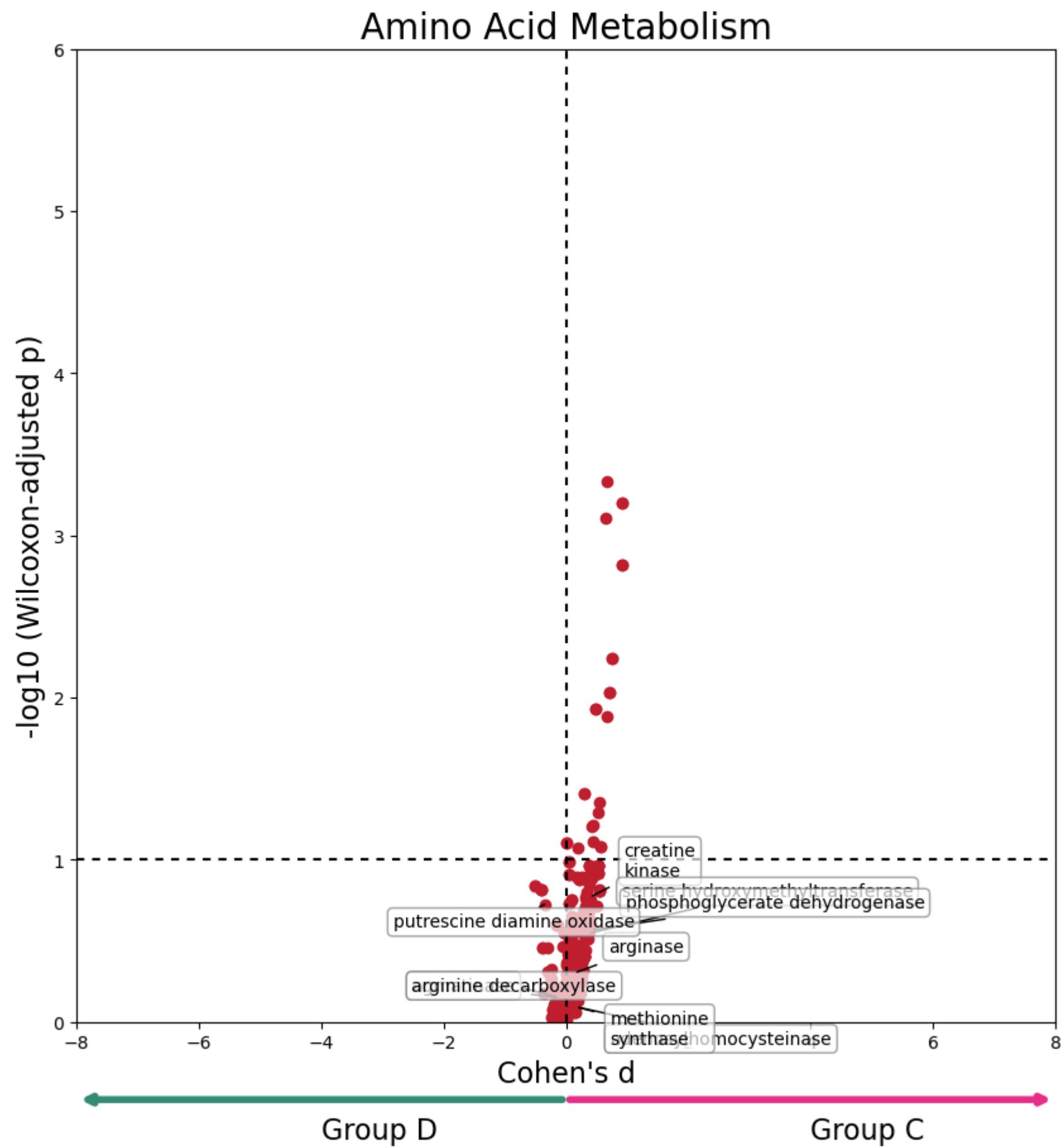
```
In [77]: data = W[W['subsystem'] == "Glycolysis/gluconeogenesis"]  
plot_differential_scores(data, title='Glycolysis', c="#695D73")
```

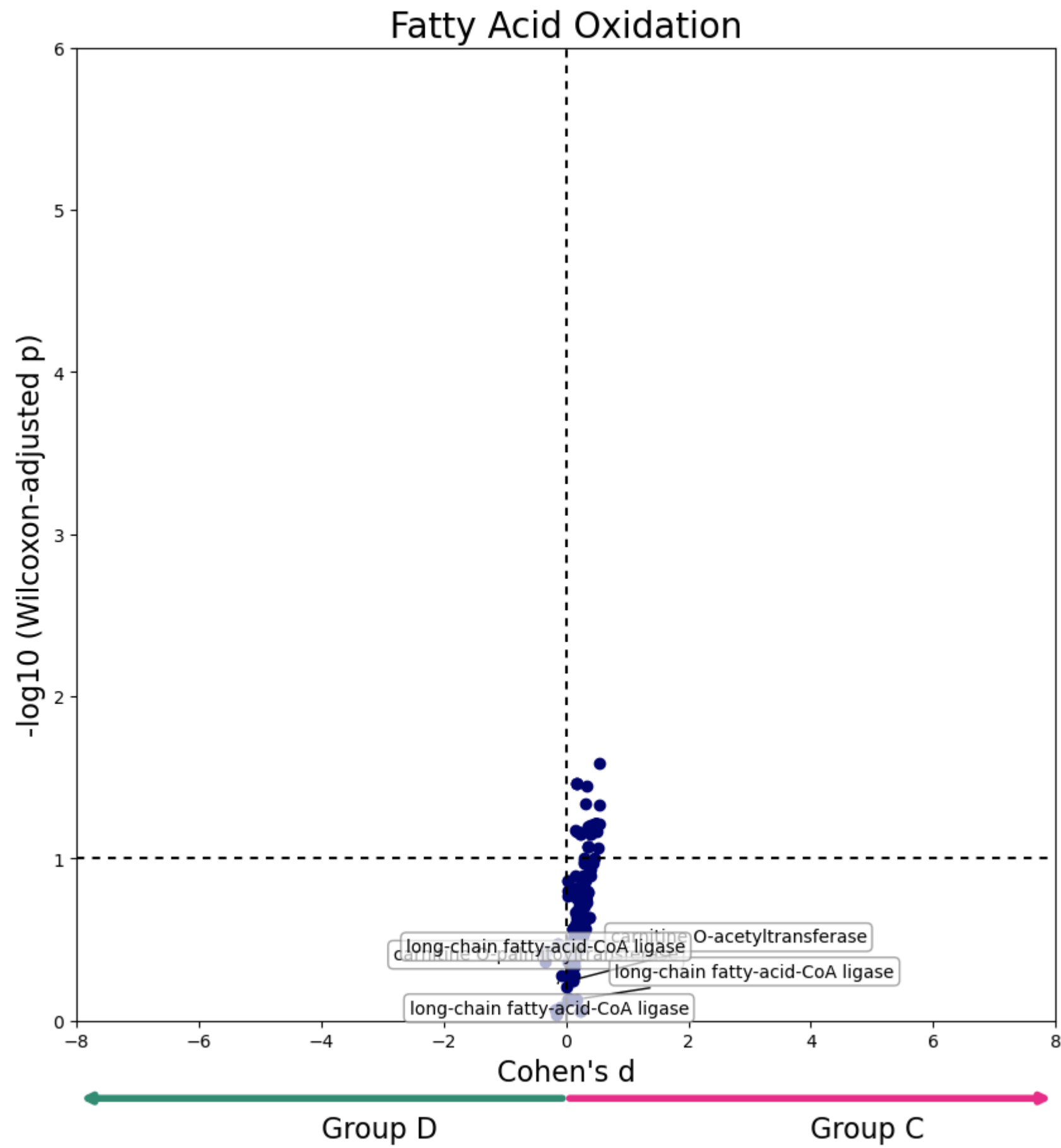
```
In [78]: data = W[W['subsystem'] == "Citric acid cycle"]  
plot_differential_scores(data, title="TCA Cycle", c="#D3A991")
```



```
In [79]: data = W[W['subsystem'].isin(amino_acid_metab)].copy()
data['adjusted_pval'] = data['adjusted_pval'].clip(1e-12)
plot_differential_scores(data, "Amino Acid Metabolism", c="#BF1E2E")
```



```
In [80]: data = W[W['subsystem'] == "Fatty acid oxidation"]
plot_differential_scores(data, "Fatty Acid Oxidation", c="#040772")
```



```
In [81]: data = W[~W['subsystem'].isin(["Miscellaneous", "Unassigned"])]
data = data[~data['subsystem'].map(lambda x: "Transport" in x or "Exchange" in x or x == "Other")]
items, counts = np.unique(data['subsystem'], return_counts=True)
items = [items[i] for i in range(len(items)) if counts[i] > 5] #filter(n() > 5) %>%
data = data[data['subsystem'].isin(items)]
```

```
In [82]: import matplotlib.pyplot as plt

plt.figure(figsize=(18, 18))
axs = plt.gca()

d = data[data['adjusted_pval'] < 0.1].groupby('subsystem')['cohens_d'].median().abs()

d_sorted = d.sort_values()

axs.scatter(d_sorted, d_sorted.index, alpha=0)

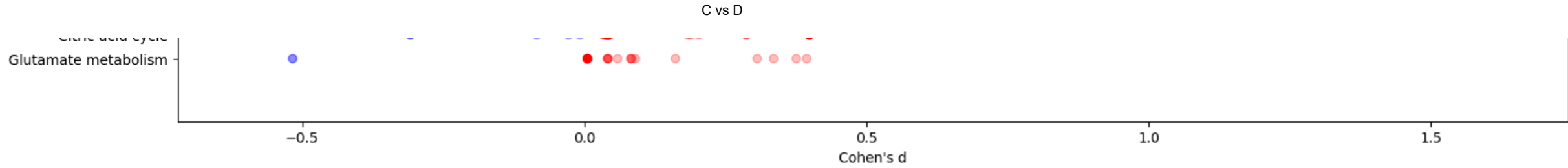
color = data['cohens_d'].map(lambda x: 'r' if x >= 0 else 'b')
alpha = data['adjusted_pval'].map(lambda x: 1.0 if x < 0.1 else 0.25)

axs.scatter(data['cohens_d'], data['subsystem'], c=color, alpha=alpha)

axs.set_xlabel("Cohen's d")
axs.set_ylabel("Pathway (subsystem)")
axs.set_title("Pathway-Level Differential Activity")
```

```
Out[82]: Text(0.5, 1.0, 'Pathway-Level Differential Activity')
```





```
In [83]: reaction_penalties = pd.read_csv("reactions.tsv", sep="\t", index_col = 0)
reaction_penalties[reaction_penalties <= 1e-4] = 0
reaction_penalties = reaction_penalties[np.all(reaction_penalties != 0, axis=1)]
```

```
In [84]: reaction_penalties = reaction_penalties[reaction_penalties.max(axis=1) - reaction_penalties.min(axis=1) != 0]
```

```
In [85]: meta_rxns_map = get_metareactions(reaction_penalties)
meta_rxns = reaction_penalties.join(pd.DataFrame(meta_rxns_map, columns=["meta_rxn_id"], index = reaction_penalties.index)).groupby("meta_rxn_id").mean()
```

```
In [86]: meta_rxn_consistencies = get_reaction_consistencies(meta_rxns)
```

```
In [87]: # Select cells for groups A and D
C_cells = cell_metadata.index[cell_metadata["group"] == "C"]
D_cells = cell_metadata.index[cell_metadata["group"] == "D"]

# Run Wilcoxon test between these two groups
wilcox_meta_rxn_results = wilcoxon_test(meta_rxn_consistencies, C_cells, D_cells)

# Optional: check the top results
print(wilcox_meta_rxn_results.head())
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval
meta_rxn_id				
1	622.0	0.790953	-0.085834	0.850346
2	598.0	0.592106	-0.360002	0.698580
3	547.0	0.266546	-0.332652	0.409833
4	894.0	0.005243	0.214503	0.043962
5	931.0	0.001332	0.584170	0.016708

```
In [88]: wilcox_meta_rxn_results.iloc[0:1]
```

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval
meta_rxn_id				
1	622.0	0.790953	-0.085834	0.850346

```
In [89]: wilcox_meta_rxn_expanded = pd.DataFrame(index=reaction_penalties.index, columns=wilcox_meta_rxn_results.columns)
for i in range(len(wilcox_meta_rxn_expanded.index)):
    if (meta_rxns_map[i] in wilcox_meta_rxn_results.index):
        wilcox_meta_rxn_expanded.loc[wilcox_meta_rxn_expanded.index[i]] = wilcox_meta_rxn_results.loc[meta_rxns_map[i]]
wilcox_meta_rxn_expanded = wilcox_meta_rxn_expanded.dropna().astype('float64')
```

```
In [90]: wilcox_meta_rxn_expanded['metadata_r_id'] = ""
for r in wilcox_meta_rxn_expanded.index:
    if r in reaction_metadata.index:
        wilcox_meta_rxn_expanded.loc[r, 'metadata_r_id'] = r
    elif r[:-4] in reaction_metadata.index:
        wilcox_meta_rxn_expanded.loc[r, 'metadata_r_id'] = r[:-4]
    else:
        print("Should not occur")
```

```
In [91]: wilcox_meta_rxn_expanded.iloc[0:1]
```

Out[91]:

	wilcox_stat	wilcox_pval	cohens_d	adjusted_pval	metadata_r_id
10FTHF5GLUtl_pos	842.0	0.027441	0.518969	0.117051	10FTHF5GLUtl

```
In [92]: outputs = {
    "wilcox_results.csv": wilcox_results,
    "reaction_consistencies.csv": reaction_consistencies,
    "reaction_metadata.csv": reaction_metadata,
    "wilcox_meta_rxn_results.csv": wilcox_meta_rxn_results,
    "wilcox_meta_rxn_expanded.csv": wilcox_meta_rxn_expanded,
    "final_stats_with_metadata.csv": W,
}

for name, df in outputs.items():
    df.to_csv(name)
```

```
In [93]: import zipfile

with zipfile.ZipFile("C vs D.zip", "w") as zipf:
    for filename in outputs:
        zipf.write(filename)
```

```
In [94]: from IPython.display import FileLink
FileLink("C vs D.zip")
```

Out[94]: C vs D.zip

```
In [ ]:
```

```
In [ ]:
```