

GUI code:

```
import tkinter as tk
from tkinter import filedialog, messagebox, Toplevel
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
from statsmodels.stats.multitest import multipletests
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

class DEGApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Differential Expression Analyzer")
        self.root.geometry("500x650")
        self.root.configure(bg="#e6f2ff")

        self.file_path = None
        self.df = None

        tk.Label(root, text="Differential Expression Analyzer", font=("Arial", 16, "bold"),
bg="#e6f2ff").pack(pady=15)
        tk.Button(root, text="Upload Gene Expression File", font=("Arial", 10),
command=self.upload_file).pack(pady=10)

        tk.Label(root, text="Data Type:", bg="#e6f2ff").pack()
        self.data_type_var = tk.StringVar(value="TPM")
        tk.Radiobutton(root, text="TPM / Normalized", variable=self.data_type_var,
value="TPM", bg="#e6f2ff").pack()
        tk.Radiobutton(root, text="Raw Count (DESeq2-like)", variable=self.data_type_var,
value="Counts", bg="#e6f2ff").pack()

        tk.Label(root, text="Group 1 Samples (comma-separated):", bg="#e6f2ff").pack()
        self.group1_entry = tk.Entry(root, width=50)
        self.group1_entry.pack(pady=5)

        tk.Label(root, text="Group 2 Samples (comma-separated):", bg="#e6f2ff").pack()
        self.group2_entry = tk.Entry(root, width=50)
        self.group2_entry.pack(pady=5)

        tk.Label(root, text="P-value Threshold:", bg="#e6f2ff").pack()
        self.pvalue_entry = tk.Entry(root, width=10)
        self.pvalue_entry.insert(0, "0.05")
        self.pvalue_entry.pack(pady=5)
```

```

tk.Label(root, text="Min abs(log2FC):", bg="#e6f2ff").pack()
self.logfc_entry = tk.Entry(root, width=10)
self.logfc_entry.insert(0, "1")
self.logfc_entry.pack(pady=5)

tk.Button(root, text="Find DEGs", font=("Arial", 10), bg="#4CAF50", fg="white",
command=self.find_degs).pack(pady=20)

def upload_file(self):
    self.file_path = filedialog.askopenfilename(
        initialdir=os.path.expanduser("~"),
        title="Select Gene Expression File",
        filetypes=[
            ("All Supported", "*.csv *.CSV *.tsv *.TSV *.txt *.TXT"),
            ("CSV files", "*.csv *.CSV"),
            ("TSV files", "*.tsv *.TSV"),
            ("Text files", "*.txt *.TXT"),
            ("All files", "*.*")
        ]
    )

    if self.file_path:
        try:
            if self.file_path.endswith(('.tsv', '.txt', '.TSV', '.TXT')):
                self.df = pd.read_csv(self.file_path, sep='\t')
            else:
                self.df = pd.read_csv(self.file_path)

            first_col = self.df.columns[0]
            if first_col != "Gene":
                self.df.rename(columns={first_col: "Gene"}, inplace=True)

            if "Gene" not in self.df.columns:
                messagebox.showerror("Error", "First column must contain gene names.")
                return

            tk.Label(self.root, text=f"Loaded: {os.path.basename(self.file_path)}",
bg="#e6f2ff", font=("Arial", 9, "italic")).pack()
            messagebox.showinfo("Success", f"File '{os.path.basename(self.file_path)}' loaded
successfully.")

        except Exception as e:
            messagebox.showerror("File Error", str(e))

def show_plot_popup(self, fig, title="Plot"):
    popup = Toplevel()
    popup.title(title)

    canvas = FigureCanvasTkAgg(fig, master=popup)
    canvas.draw()

```

```

canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

def save_figure():
    file = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG
Image", "*.png"), ("All Files", "*.*)])
    if file:
        fig.savefig(file)
        messagebox.showinfo("Saved", f"Plot saved to: {file}")

tk.Button(popup, text="Save Plot", command=save_figure).pack(pady=5)

def find_degs(self):
    if self.df is None:
        messagebox.showerror("Error", "Please upload a dataset first.")
        return

    try:
        group1 = [col.strip() for col in self.group1_entry.get().split(",")]
        group2 = [col.strip() for col in self.group2_entry.get().split(",")]
        p_thresh = float(self.pvalue_entry.get())
        logfc_thresh = float(self.logfc_entry.get())
        data_type = self.data_type_var.get()

        gene_names = []
        p_values = []
        log2fcs = []

        data_subset = self.df["Gene"].to_frame().copy()
        data_subset[group1 + group2] = self.df[group1 + group2]

        if data_type == "TPM":
            data_subset = data_subset[(data_subset[group1 + group2].mean(axis=1)) > 1]
        elif data_type == "Counts":
            data_subset = data_subset[(data_subset[group1 + group2].sum(axis=1)) > 10]

        for idx, row in data_subset.iterrows():
            try:
                g1_vals = row[group1].astype(float)
                g2_vals = row[group2].astype(float)
                stat, pval = ttest_ind(g1_vals, g2_vals, equal_var=False)

                g1_mean = np.mean(g1_vals)
                g2_mean = np.mean(g2_vals)
                log2fc = np.log2(g2_mean / g1_mean) if g1_mean > 0 and g2_mean > 0 else
np.nan

                gene_names.append(row['Gene'])
                p_values.append(pval)
                log2fcs.append(log2fc)
            except Exception:

```

```

        continue

    adj_pvals = multipletests(p_values, alpha=p_thresh, method='fdr_bh')[1]

    degs = pd.DataFrame({
        "Gene": gene_names,
        "log2FoldChange": log2fcs,
        "P-Value": p_values,
        "Adjusted P-Value": adj_pvals
    })

    degs_filtered = degs[(degs["Adjusted P-Value"] <= p_thresh) &
        (degs["log2FoldChange"].abs() >= logfc_thresh)]

    out_path = os.path.splitext(self.file_path)[0] + f"_DEGs_{data_type}.csv"
    degs_filtered.to_csv(out_path, index=False)
    messagebox.showinfo("DEGs Saved", f"{len(degs_filtered)} significant DEGs saved
to {out_path}")

    # Volcano Plot in Popup
    fig1, ax1 = plt.subplots(figsize=(6, 5))
    ax1.scatter(degs['log2FoldChange'], -np.log10(degs['P-Value']), alpha=0.5)
    ax1.set_xlabel("log2(Fold Change)")
    ax1.set_ylabel("-log10(P-value)")
    ax1.set_title("Volcano Plot")
    ax1.grid(True)
    self.show_plot_popup(fig1, title="Volcano Plot")

    # Heatmap in Popup
    top_genes = degs_filtered.sort_values("Adjusted P-Value").head(20)["Gene"]
    heatmap_data = self.df.set_index("Gene").loc[top_genes, group1 + group2]
    cg = sns.clustermap(heatmap_data, cmap="vlag", z_score=0, figsize=(8, 6))
    cg.fig.suptitle("Top 20 DEGs Heatmap")
    self.show_plot_popup(cg.fig, title="Heatmap")

except Exception as e:
    messagebox.showerror("Error", str(e))


if __name__ == "__main__":
    root = tk.Tk()
    app = DEGApp(root)
    root.mainloop()

```

Snaps of how the GUI works:

For raw count:

Step1: Fig 1: shows the interface and a Raw count file being uploaded with samples as test cases to be observed and the p-value is set to 0.05 and lof2Fc set to min 1.



The screenshot shows a web-based application window titled "Differential Expression Analyzer". The interface is light blue with a white title bar. At the top, there's a header "Differential Expression Analyzer" with a feather icon on the left and standard window controls (minimize, maximize, close) on the right. Below the header, the title "Differential Expression Analyzer" is repeated in a large, bold, black font. A button labeled "Upload Gene Expression File" is centered. Below this, the "Data Type:" section has two radio buttons: "TPM / Normalized" (unselected) and "Raw Count (DESeq2-like)" (selected). The "Group 1 Samples (comma-separated):" field contains the text "-3,HFD-NC-4,HFD-SH-1,HFD-SH-2,HFD-SH-4,HFD-SH-5". The "Group 2 Samples (comma-separated):" field contains the text "HT1,HT2,HT3,HT4,HT5,LT1,LT2,LT3,LT4,LT5". The "P-value Threshold:" field is set to "0.05". The "Min abs(log2FC):" field is set to "1". A green button labeled "Find DEGs" is centered below these fields. At the bottom, the text "Loaded: combined_gene_expression_data.csv" is displayed.

Differential Expression Analyzer

Differential Expression Analyzer

Upload Gene Expression File

Data Type:

☐ TPM / Normalized

☒ Raw Count (DESeq2-like)

Group 1 Samples (comma-separated):

-3,HFD-NC-4,HFD-SH-1,HFD-SH-2,HFD-SH-4,HFD-SH-5

Group 2 Samples (comma-separated):

HT1,HT2,HT3,HT4,HT5,LT1,LT2,LT3,LT4,LT5

P-value Threshold:

0.05

Min abs(log2FC):

1

Find DEGs

Loaded: combined_gene_expression_data.csv

Figure 1: GUI interface and Raw count file loaded

Step 2: Fig.2 shows the output from the file i.e the DEGs.

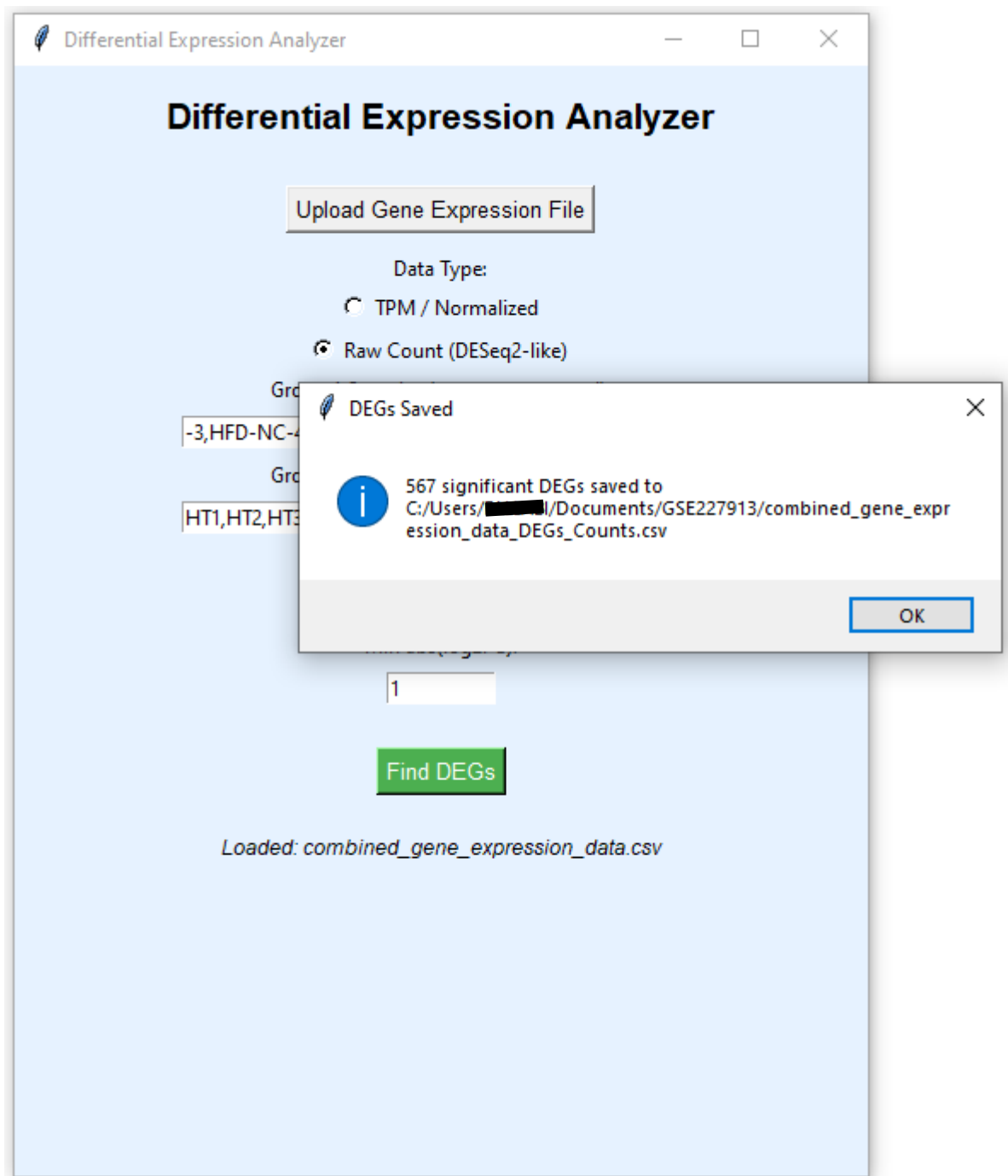


Figure 2: Output

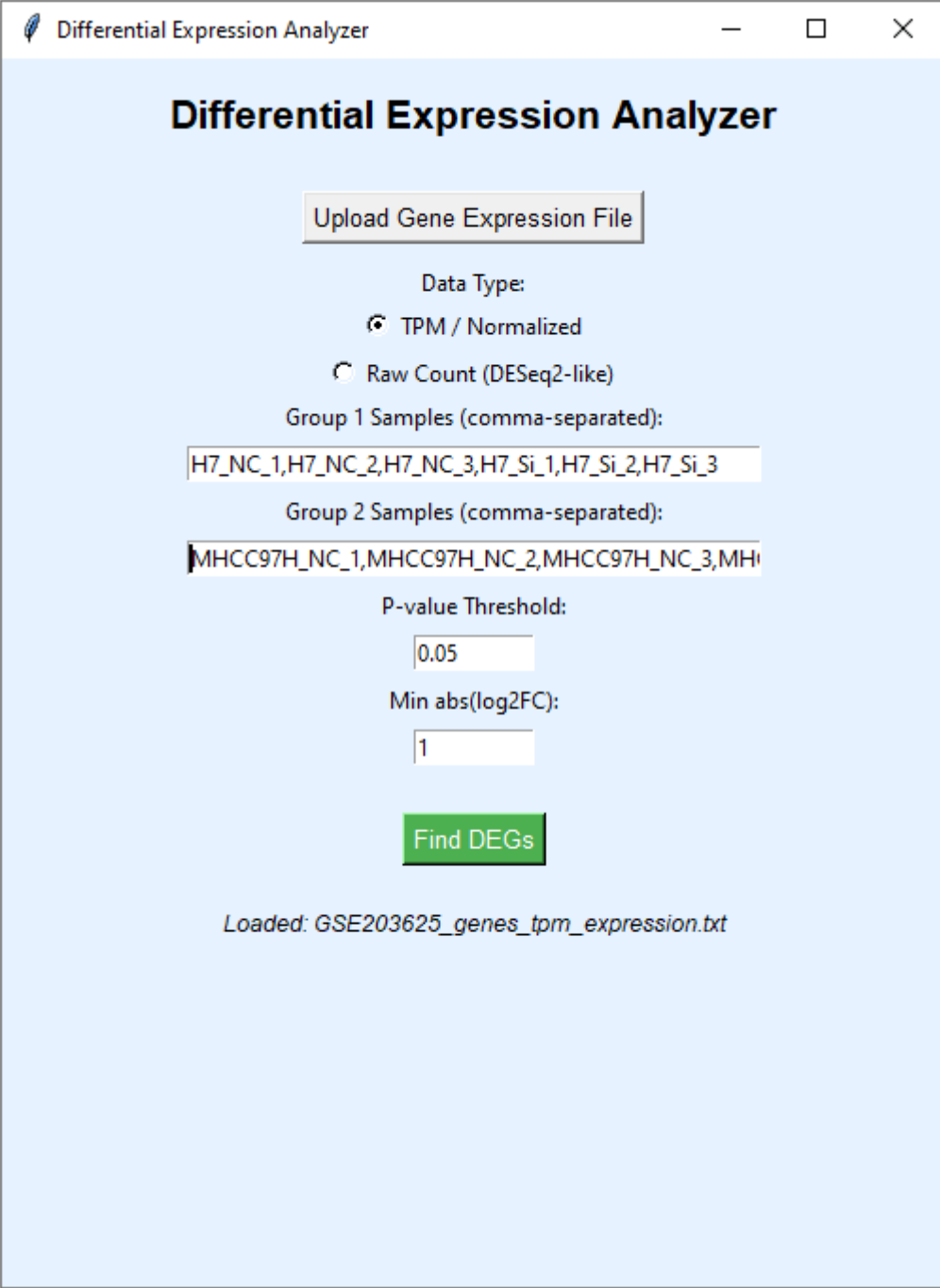
Step 3: Fig.3 shows the volcano plots and heatmaps from the results.



Figure 3: Plots

For TPM/Normalized data:

Step1: Fig 4. shows the interface and a TPM file being uploaded with samples as test cases to be observed and the p-value is set to 0.05 and log2Fc set to min 1.



The screenshot shows a web-based application window titled "Differential Expression Analyzer". The interface is light blue and contains the following elements:

- Upload Gene Expression File**: A button to upload the data file.
- Data Type:** Two radio buttons are present:
 - ☒ TPM / Normalized (selected)
 - ☐ Raw Count (DESeq2-like)
- Group 1 Samples (comma-separated):** A text input field containing "H7_NC_1,H7_NC_2,H7_NC_3,H7_Si_1,H7_Si_2,H7_Si_3".
- Group 2 Samples (comma-separated):** A text input field containing "MHCC97H_NC_1,MHCC97H_NC_2,MHCC97H_NC_3,MHCC97H_Si_1,MHCC97H_Si_2,MHCC97H_Si_3".
- P-value Threshold:** A text input field containing "0.05".
- Min abs(log2FC):** A text input field containing "1".
- Find DEGs**: A green button to execute the analysis.
- Loaded:** A status message at the bottom indicating the file "GSE203625_genes_tpm_expression.txt" has been loaded.

Figure 4: GUI interface and TPM file loaded

Step 2: Fig.5 shows the output from the file i.e the DEGs.

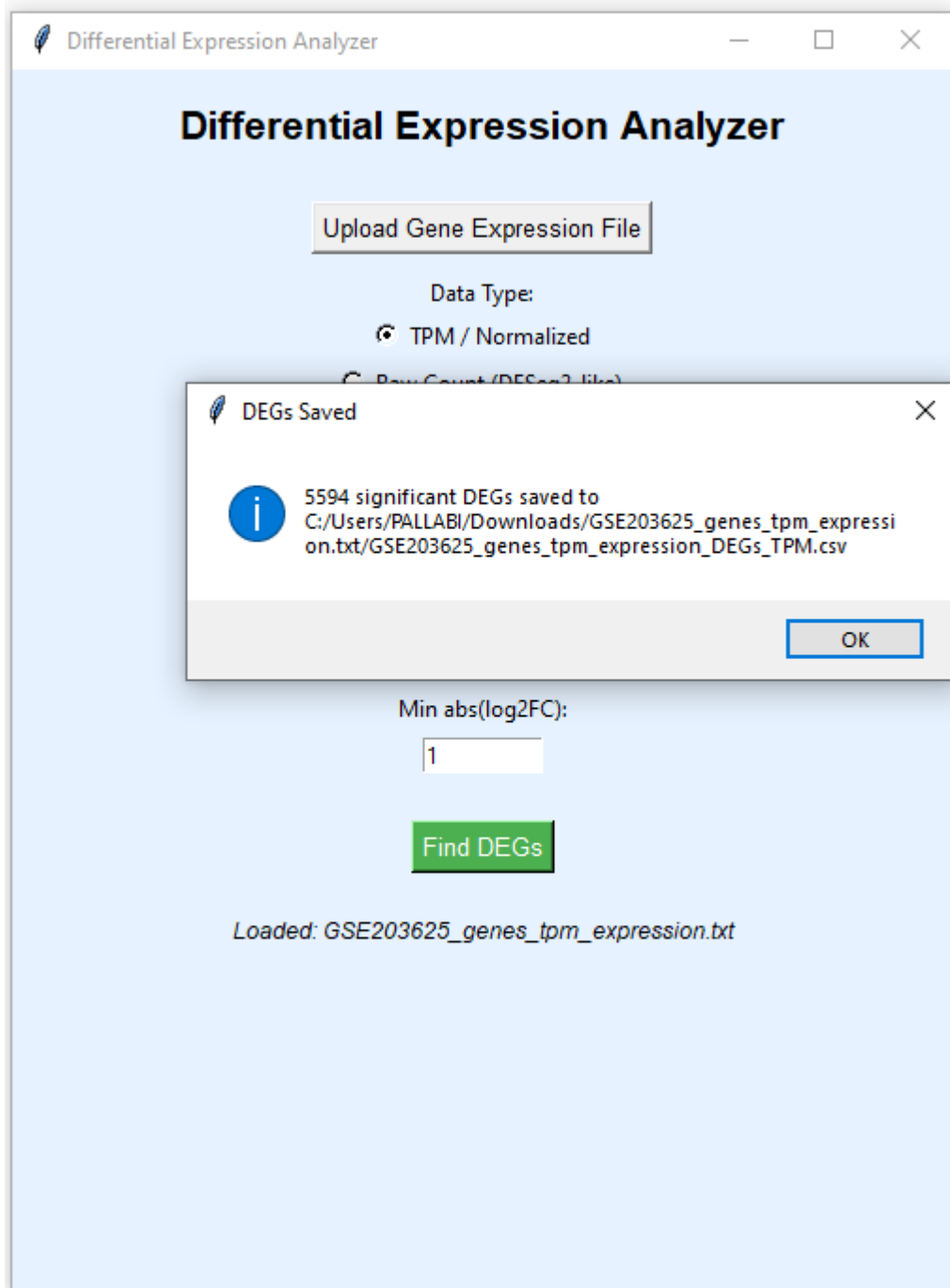


Figure 5: Output

Step 3: Fig.6 shows the volcano plots and heatmaps from the results.



Figure 6: Plots

