```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving stepparent_pre_0603.csv to stepparent_pre_0603.csv
```

```
!pip install pingouin
import pingouin as pg
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.anova import AnovaRM
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.formula.api import ols
```

```
df = pd.read_csv('stepparent_pre_0603.csv')

df.head()
```

```
# Display basic info about the dataset
print("Dataset shape:", df.shape)
print("\nColumn names:")
print(df.columns.tolist())
print("\nFirst few rows:")
print(df.head())
print("\nData types:")
print(df.dtypes)

# Map 'gender_pre' values to 'male' and 'female'
df['gender_pre'] = df['gender_pre'].map({1: 'male', 2: 'female'}).astype('category')
```

```
# Define the variables for analysis
variables = [
    ('PS_M_bio_pre', 'PS_M_step_pre'),
    ('PSlaxness_M_bio_pre', 'PSlaxness_M_step_pre'),
    ('PSoverreactivity_M_bio_pre', 'PSoverreactivity_M_step_pre'),
    ('PSverbosity_M_bio_pre', 'PSverbosity_M_step_pre'),
    ('PSnotOnFactor_M_bio_pre', 'PSnotOnFactor_M_step_pre')
]

# Define additional variables for analysis
additional_variables = [
    ('PAS1to23_M_bio_pre', 'PAS1to23_M_step_pre'),
    ('PAS_bio_experience', 'PAS_step_experience'),
    ('PAS_bio_expression', 'PAS_step_expression'),
    ('love_M_bio_pre', 'love_M_step_pre')
]

# Combine with existing variables
all_variables = variables + additional_variables
```

```
# Loop through each pair of variables
for bio_var, step_var in all_variables:
    # Reshape the data for repeated measures analysis
  df_long = pd.melt(df,
                        value_vars=[bio_var, step_var],
                    var_name='ChildStatus',
                    value_name='PS_Score',
                        id_vars=[col for col in df.columns if col not in [bio_var, step_var]])
```

```
# Clean the ChildStatus variable
df_long['ChildStatus'] = df_long['ChildStatus'].str.replace('PS_M_', '').str.replace('_pre', '')
df_long['ChildStatus'] = df_long['ChildStatus'].str.replace('PSnotOnFactor', 'indifference')

# Use "phone" as the unique identifier for each case
if 'phone' in df.columns:
    df_long['subject_id'] = df_long['phone']
else:
    print("Warning: 'phone' column not found, creating sequential subject IDs")
    df_long['subject_id'] = df_long.groupby(['ChildStatus']).cumcount()

# Descriptive statistics
desc_stats = df_long.groupby('ChildStatus')['PS_Score'].agg(['count', 'mean', 'std', 'sem']).round(4)
print(f"\nDESCRIPTIVE STATISTICS for {bio_var} vs {step_var}")
print(desc_stats)
```

```
DESCRIPTIVE STATISTICS for love_M_bio_pre vs love_M_step_pre
                 count    mean     std     sem
ChildStatus
love_M_bio     174  4.3956  0.5887  0.0446
love_M_step    174  3.8994  0.7607  0.0577
```

```
# Repeated measures ANOVA
rm_anova = pg.rm_anova(data=df_long,
                       dv='PS_Score',
                       within='ChildStatus',
                       subject='subject_id',
                       detailed=True)
print(f"\nREPEATED MEASURES ANOVA for {bio_var} vs {step_var}")
print(rm_anova)

    # Check available columns in the ANOVA results
print("\nAvailable columns in ANOVA results:")
print(rm_anova.columns)

    # Display Partial Eta Squared (ηp²)
if 'ng2' in rm_anova.columns:
    eta_squared = rm_anova['ng2'].iloc[0]
    print(f"Partial Eta Squared (η²p): {eta_squared:.4f}")
else:
    print("Effect size column 'ng2' not found in ANOVA results.")

    # Estimated Marginal Means
overall_mean = df_long['PS_Score'].mean()
overall_se = stats.sem(df_long['PS_Score'].dropna())
print(f"\nOverall Marginal Mean: {overall_mean:.4f} (SE = {overall_se:.4f})")

    # Marginal means by ChildStatus
group_stats = df_long.groupby('ChildStatus')['PS_Score'].agg(['mean', 'sem', 'count']).round(4)
print("\nMarginal Means by ChildStatus:")
```

```
for group in group_stats.index:
    mean = group_stats.loc[group, 'mean']
    se = group_stats.loc[group, 'sem']
    n = group_stats.loc[group, 'count']
    print(f"{group.capitalize()}: Mean = {mean:.4f}, SE = {se:.4f}, N = {int(n)}")
```

```
REPEATED MEASURES ANOVA for love_M_bio_pre vs love_M_step_pre
        Source         SS   DF         MS          F         p-unc       ng2  \
0  ChildStatus  21.417944    1  21.417944  72.575784  7.606700e-15  0.118016
1        Error  51.054278  173   0.295111        NaN           NaN       NaN

   eps
0  1.0
1  NaN

Available columns in ANOVA results:
Index(['Source', 'SS', 'DF', 'MS', 'F', 'p-unc', 'ng2', 'eps'], dtype='object')
Partial Eta Squared (η²p): 0.1180

Overall Marginal Mean: 4.1475 (SE = 0.0388)

Marginal Means by ChildStatus:
Love_m_bio: Mean = 4.3956, SE = 0.0446, N = 174
Love_m_step: Mean = 3.8994, SE = 0.0577, N = 174
```

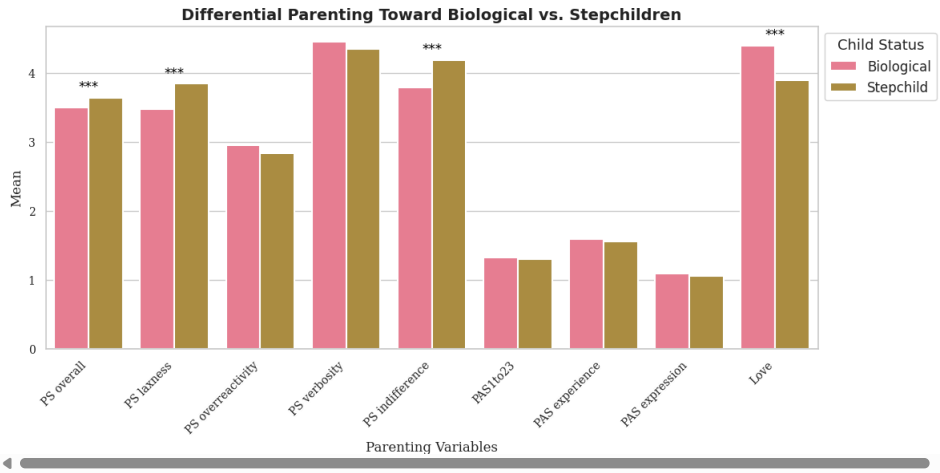
```
# Pairwise comparisons
bio_scores = df[bio_var].dropna()
step_scores = df[step_var].dropna()

    # Ensure same length for paired comparison
min_len = min(len(bio_scores), len(step_scores))
bio_scores = bio_scores.iloc[:min_len]
step_scores = step_scores.iloc[:min_len]

    # Paired t-test
t_stat, p_value = stats.ttest_rel(bio_scores, step_scores)

    # Calculate Cohen's d for paired samples
cohen_d = (bio_scores.mean() - step_scores.mean()) / np.sqrt(((bio_scores.std()**2 + step_scores.std()**2) / 2))

    # Calculate 95% confidence interval for the difference
diff_scores = bio_scores - step_scores
diff_mean = diff_scores.mean()
diff_se = stats.sem(diff_scores)
ci_lower = diff_mean - 1.96 * diff_se
ci_upper = diff_mean + 1.96 * diff_se

print(f"\nPAIRWISE COMPARISONS (LSD method) for {bio_var} vs {step_var}")
print(f"Mean difference (Bio - Step): {diff_mean:.4f}")
print(f"95% CI: [{ci_lower:.4f}, {ci_upper:.4f}]")
print(f"t({min_len-1}) = {t_stat:.4f}, p = {p_value:.4f}")
print(f"Cohen's d = {cohen_d:.4f}")

    # Interpretation
if p_value < 0.001:
    sig_level = "p < .001"
elif p_value < 0.01:
    sig_level = "p < .01"
elif p_value < 0.05:
    sig_level = "p < .05"
else:
    sig_level = f"p = {p_value:.3f} (not significant)"

print(f"Significance: {sig_level}")
```

```
PAIRWISE COMPARISONS (LSD method) for love_M_bio_pre vs love_M_step_pre
Mean difference (Bio - Step): 0.4962
95% CI: [0.3820, 0.6103]
t(173) = 8.5191, p = 0.0000
Cohen's d = 0.7295
Significance: p < .001
```

```
# Prepare data for plotting
plot_data = []
for bio_var, step_var in all_variables:
    bio_mean = df[bio_var].mean()
    step_mean = df[step_var].mean()
    plot_data.append({'Variable': bio_var.replace('_bio_pre', ''), 'ChildStatus': 'Biological', 'Mean': bio_mean})
    plot_data.append({'Variable': step_var.replace('_step_pre', ''), 'ChildStatus': 'Stepchild', 'Mean': step_mean})

# Update variable names for comparison
comparison_variables = [
    ('PAS_bio_experience', 'PAS_step_experience'),
    ('PAS_bio_expression', 'PAS_step_expression')
]

# Update the plot data preparation to include these comparisons
for bio_var, step_var in comparison_variables:
    bio_mean = df[bio_var].mean()
    step_mean = df[step_var].mean()
    plot_data.append({'Variable': bio_var.replace('PAS_bio_', '').replace('_', ' '), 'ChildStatus': 'Biological', 'Mean': bio_mean})
    plot_data.append({'Variable': step_var.replace('PAS_step_', '').replace('_', ' '), 'ChildStatus': 'Stepchild', 'Mean': step_mean})

plot_df = pd.DataFrame(plot_data)

# Correct grouping for experience and expression
plot_df['Variable'] = plot_df['Variable'].replace({
    'PAS_bio_experience': 'PAS_experience',
    'PAS_step_experience': 'PAS_experience',
    'PAS_bio_expression': 'PAS_expression',
    'PAS_step_expression': 'PAS_expression',
    'notonfactor': 'indifference'
})

# Correct variable names for the plot and remove duplicates
plot_df['Variable'] = plot_df['Variable'].replace({
    'PS_M': 'PS overall',
    'PSlaxness_M': 'PS laxness',
    'PSoverreactivity_M': 'PS overreactivity',
    'PSverbosity_M': 'PS verbosity',
    'PSnotOnFactor_M': 'PS indifference',
    'PAS1to23_M': 'PAS1to23',
    'PAS_experience': 'PAS experience',
    'PAS_expression': 'PAS expression',
    'love_M': 'Love',
    'experience': 'PAS experience',
    'expression': 'PAS expression'
})

# Ensure error bars are visible
plt.figure(figsize=(12, 6))
sns.set(style="whitegrid")
sns.set_palette("husl")  # Nature recommended color palette
sns.set_context("notebook", font_scale=1.2)

# Plot without error bars
ax = sns.barplot(data=plot_df, x='Variable', y='Mean', hue='ChildStatus', ci=None,
                 linewidth=1.5)
```

```
# Customize plot
plt.title('Differential Parenting Toward Biological vs. Stepchildren', fontsize=14, fontweight='bold')
plt.xlabel('Parenting Variables', fontsize=12, fontname='Serif')
plt.ylabel('Mean', fontsize=12, fontname='Serif')
plt.xticks(rotation=45, ha='right', fontsize=10, fontname='Serif')
plt.yticks(fontsize=10, fontname='Serif')
plt.legend(title='Child Status', fontsize=12, title_fontsize='13', loc='upper left', bbox_to_anchor=(1, 1))

# Add significance markers
for i, (bio_var, step_var) in enumerate(all_variables):
    t_stat, p_value = stats.ttest_rel(df[bio_var].dropna(), df[step_var].dropna())
    if p_value < 0.001:
        sig_marker = '***'
    elif p_value < 0.01:
        sig_marker = '**'
    elif p_value < 0.05:
        sig_marker = '*'
    else:
        sig_marker = ''
    if sig_marker:
        ax.text(i, max(df[bio_var].mean(), df[step_var].mean()) + 0.1, sig_marker, ha='center', fontsize=12, color='black')

# Save as high-resolution image
plt.tight_layout()
plt.savefig('differential_parenting_plot.svg', format='svg', dpi=600)
plt.show()
```

```
<ipython-input-37-36c0f339ddfc>:55: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  ax = sns.barplot(data=plot_df, x='Variable', y='Mean', hue='ChildStatus', ci=None,
```



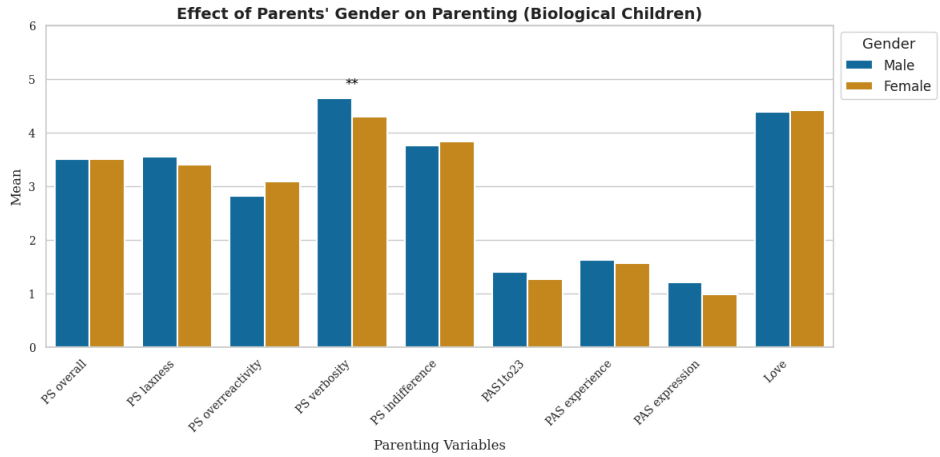

```
# Analyze the impact of gender on specified variables
variables_to_analyze = [
    'PS_M_bio_pre', 'PS_M_step_pre',
    'PSlaxness_M_bio_pre', 'PSlaxness_M_step_pre',
    'PSoverreactivity_M_bio_pre', 'PSoverreactivity_M_step_pre',
    'PSverbosity_M_bio_pre', 'PSverbosity_M_step_pre',
    'PSnotOnFactor_M_bio_pre', 'PSnotOnFactor_M_step_pre',
    'PAS1to23_M_bio_pre', 'PAS1to23_M_step_pre',
    'PAS_bio_experience', 'PAS_step_experience',
    'PAS_bio_expression', 'PAS_step_expression'
]

for var in variables_to_analyze:
    # Perform ANOVA to test the effect of gender
    anova_results = pg.anova(data=df, dv=var, between='gender_pre', detailed=True)
    print(f"\nANOVA for {var} by gender")
    print(anova_results)

    # Check for significance
    p_value = anova_results['p-unc'].iloc[0]
    if p_value < 0.05:
        print(f"Significant effect of gender on {var}, p = {p_value:.4f}")
    else:
        print(f"No significant effect of gender on {var}, p = {p_value:.4f}")


# T-test analysis for the impact of gender on specified variables
for var in variables_to_analyze:
    # Separate data by gender
    male_data = df[df['gender_pre'] == 'male'][var].dropna()
    female_data = df[df['gender_pre'] == 'female'][var].dropna()

    # Perform t-test
    t_stat, p_value = stats.ttest_ind(male_data, female_data)
    print(f"\nT-test for {var} by gender")
    print(f"t-statistic = {t_stat:.4f}, p-value = {p_value:.4f}")

    # Check for significance
    if p_value < 0.05:
        print(f"Significant effect of gender on {var}, p = {p_value:.4f}")
    else:
        print(f"No significant effect of gender on {var}, p = {p_value:.4f}")
```

```
T-test for PSverbosity_M_step_pre by gender
t-statistic = 2.5438, p-value = 0.0118
Significant effect of gender on PSverbosity_M_step_pre, p = 0.0118

T-test for PSnotOnFactor_M_bio_pre by gender
t-statistic = -0.6455, p-value = 0.5195
No significant effect of gender on PSnotOnFactor_M_bio_pre, p = 0.5195

T-test for PSnotOnFactor_M_step_pre by gender
t-statistic = -3.7504, p-value = 0.0002
Significant effect of gender on PSnotOnFactor_M_step_pre, p = 0.0002

T-test for PAS1to23_M_bio_pre by gender
t-statistic = 0.8396, p-value = 0.4023
No significant effect of gender on PAS1to23_M_bio_pre, p = 0.4023

T-test for PAS1to23_M_step_pre by gender
t-statistic = 1.2962, p-value = 0.1966
No significant effect of gender on PAS1to23_M_step_pre, p = 0.1966

T-test for PAS_bio_experience by gender
t-statistic = 0.3174, p-value = 0.7514
No significant effect of gender on PAS_bio_experience, p = 0.7514

T-test for PAS_step_experience by gender
t-statistic = 0.3145, p-value = 0.7535
No significant effect of gender on PAS_step_experience, p = 0.7535

T-test for PAS_bio_expression by gender
t-statistic = 1.2817, p-value = 0.2017
No significant effect of gender on PAS_bio_expression, p = 0.2017

T-test for PAS_step_expression by gender
t-statistic = 2.2544, p-value = 0.0254
Significant effect of gender on PAS_step_expression, p = 0.0254
```

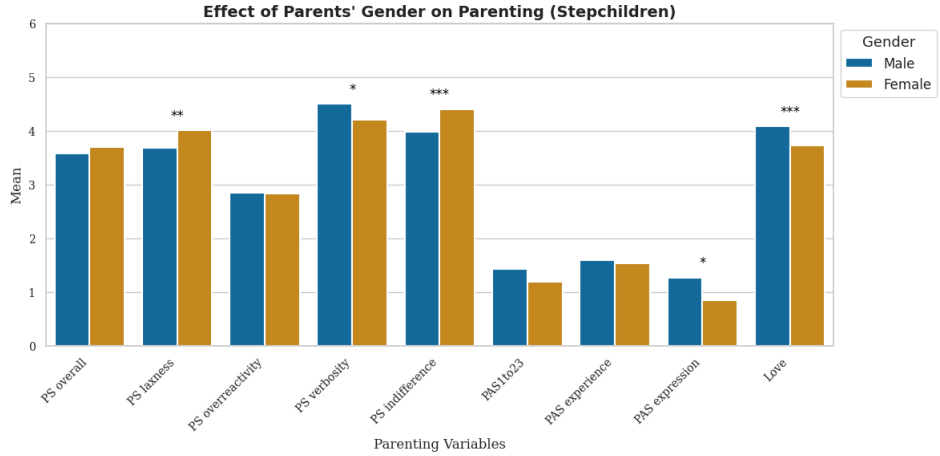
```
# Prepare data for plotting based on t-test results
plot_data_bio = []
plot_data_step = []

# Variables to include in the plot
variables_to_plot = [
    'PS_M_bio_pre', 'PS_M_step_pre',
    'PSlaxness_M_bio_pre', 'PSlaxness_M_step_pre',
    'PSoverreactivity_M_bio_pre', 'PSoverreactivity_M_step_pre',
    'PSverbosity_M_bio_pre', 'PSverbosity_M_step_pre',
    'PSnotOnFactor_M_bio_pre', 'PSnotOnFactor_M_step_pre',
    'PAS1to23_M_bio_pre', 'PAS1to23_M_step_pre',
    'PAS_bio_experience', 'PAS_step_experience',
    'PAS_bio_expression', 'PAS_step_expression',
    'love_M_bio_pre', 'love_M_step_pre'
]

# Extract means for male and female parents
for var in variables_to_plot:
    male_mean = df[df['gender_pre'] == 'male'][var].mean()
    female_mean = df[df['gender_pre'] == 'female'][var].mean()
    if 'bio' in var:
        plot_data_bio.append({'Variable': var.replace('_bio_pre', ''), 'Gender': 'Male', 'Mean': male_mean})
        plot_data_bio.append({'Variable': var.replace('_bio_pre', ''), 'Gender': 'Female', 'Mean': female_mean})
    else:
        plot_data_step.append({'Variable': var.replace('_step_pre', ''), 'Gender': 'Male', 'Mean': male_mean})
        plot_data_step.append({'Variable': var.replace('_step_pre', ''), 'Gender': 'Female', 'Mean': female_mean})

# Convert to DataFrame
plot_df_bio = pd.DataFrame(plot_data_bio)
plot_df_step = pd.DataFrame(plot_data_step)

# Update variable names for the plot
plot_df_bio['Variable'] = plot_df_bio['Variable'].replace({
    'PS_M': 'PS overall',
    'PSlaxness_M': 'PS laxness',
    'PSoverreactivity_M': 'PS overreactivity',
    'PSverbosity_M': 'PS verbosity',
    'PSnotOnFactor_M': 'PS indifference',
    'PAS1to23_M': 'PAS1to23',
    'PAS_bio_experience': 'PAS experience',
    'PAS_bio_expression': 'PAS expression',
    'love_M': 'Love'
})

plot_df_step['Variable'] = plot_df_step['Variable'].replace({
    'PS_M': 'PS overall',
    'PSlaxness_M': 'PS laxness',
    'PSoverreactivity_M': 'PS overreactivity',
    'PSverbosity_M': 'PS verbosity',
    'PSnotOnFactor_M': 'PS indifference',
    'PAS1to23_M': 'PAS1to23',
    'PAS_step_experience': 'PAS experience',
    'PAS_step_expression': 'PAS expression',
    'love_M': 'Love'
})

# Change color palette to be colorblind-friendly
sns.set_palette("colorblind")  # Use a colorblind-friendly palette for better contrast

# Plot for biological children
plt.figure(figsize=(12, 6))
ax_bio = sns.barplot(data=plot_df_bio, x='Variable', y='Mean', hue='Gender', ci=None, linewidth=1.5)

# Customize plot
plt.title("Effect of Parents' Gender on Parenting (Biological Children)", fontsize=14, fontweight='bold')
plt.xlabel('Parenting Variables', fontsize=12, fontname='Serif')
plt.ylabel('Mean', fontsize=12, fontname='Serif')
plt.xticks(rotation=45, ha='right', fontsize=10, fontname='Serif')
plt.yticks(fontsize=10, fontname='Serif')
plt.legend(title='Gender', fontsize=12, title_fontsize='13', loc='upper left', bbox_to_anchor=(1, 1))

# Add significance markers for biological children
significance_bio = {
    'PS overall': 0.9630,
    'PS laxness': 0.2045,
    'PS overreactivity': 0.0507,
    'PS verbosity': 0.0045,
    'PS indifference': 0.5195,
    'PAS1to23': 0.4023,
    'PAS experience': 0.7514,
    'PAS expression': 0.2017,
    'Love': 0.9630
}

for i, var in enumerate(plot_df_bio['Variable'].unique()):
    p_value = significance_bio.get(var, 1)
    if p_value < 0.001:
        sig_marker = '***'
    elif p_value < 0.01:
        sig_marker = '**'
    elif p_value < 0.05:
        sig_marker = '*'
    else:
        sig_marker = ''
    if sig_marker:
```

```
        # Reduce the offset for visibility
        max_mean = plot_df_bio[plot_df_bio['Variable'] == var]['Mean'].max()
        ax_bio.text(i, max_mean + 0.2, sig_marker, ha='center', fontsize=12, color='black')

# Extend Y-axis limit for better visibility
ax_bio.set_ylim(0, 6)

# Save plot
plt.tight_layout()
plt.savefig('effect_of_gender_bio.svg', format='svg', dpi=600)
plt.show()

# Plot for stepchildren
plt.figure(figsize=(12, 6))
ax_step = sns.barplot(data=plot_df_step, x='Variable', y='Mean', hue='Gender', ci=None, linewidth=1.5)

# Customize plot
plt.title("Effect of Parents' Gender on Parenting (Stepchildren)", fontsize=14, fontweight='bold')
plt.xlabel('Parenting Variables', fontsize=12, fontname='Serif')
plt.ylabel('Mean', fontsize=12, fontname='Serif')
plt.xticks(rotation=45, ha='right', fontsize=10, fontname='Serif')
plt.yticks(fontsize=10, fontname='Serif')
plt.legend(title='Gender', fontsize=12, title_fontsize='13', loc='upper left', bbox_to_anchor=(1, 1))

# Add significance markers for stepchildren
significance_step = {
    'PS overall': 0.0564,
    'PS laxness': 0.0080,
    'PS overreactivity': 0.8920,
    'PS verbosity': 0.0118,
    'PS indifference': 0.0002,
    'PAS1to23': 0.1966,
    'PAS experience': 0.7535,
    'PAS expression': 0.0254,
    'Love': 0.0000
}

for i, var in enumerate(plot_df_step['Variable'].unique()):
    p_value = significance_step.get(var, 1)
    if p_value < 0.001:
        sig_marker = '***'
    elif p_value < 0.01:
        sig_marker = '**'
    elif p_value < 0.05:
        sig_marker = '*'
    else:
        sig_marker = ''
    if sig_marker:
        # Reduce the offset for visibility
        max_mean = plot_df_step[plot_df_step['Variable'] == var]['Mean'].max()
        ax_step.text(i, max_mean + 0.2, sig_marker, ha='center', fontsize=12, color='black')

# Extend Y-axis limit for better visibility
ax_step.set_ylim(0, 6)

# Save plot
plt.tight_layout()
plt.savefig('effect_of_gender_step.svg', format='svg', dpi=600)
plt.show()
```

```
<ipython-input-40-25224dde390e>:85: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  ax_bio = sns.barplot(data=plot_df_bio, x='Variable', y='Mean', hue='Gender', ci=None, linewidth=1.5)
```




```
<ipython-input-40-25224dde390e>:111: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  ax_step = sns.barplot(data=plot_df_step, x='Variable', y='Mean', hue='Gender', ci=None, linewidth=1.5)
```




```
import matplotlib.font_manager as fm

# Get a list of all available font paths
font_paths = fm.findSystemFonts()

# Get the font names from the paths
font_names = [fm.FontProperties(fname=fname).get_name() for fname in font_paths]

# Print the unique font names
print("Available fonts:")
for font_name in sorted(list(set(font_names))):
    print(font_name)
```

```
Available fonts:
Humor Sans
Liberation Mono
Liberation Sans
Liberation Sans Narrow
Liberation Serif
```

```
# Step 1: Data Preparation
# Ensure emotion regulation variables are correctly formatted
emotion_regulation_vars = ['ERQRe_high1Low0', 'ERQSup_high1Low0', 'ERQ_reappraisalUser', 'DERS_high1Low0', 'FLSS_M_pre']

# Calculate difference scores for parenting variables
for bio_var, step_var in variables:
    df[f'diff_{bio_var}'] = df[bio_var] - df[step_var]

# Step 2: RQ3 - Multiple Linear Regressions
# Perform regressions for each parenting variable
for var in variables + additional_variables:
    bio_var, step_var = var
    for outcome in [bio_var, step_var]:
        formula = f'{outcome} ~ ' + ' + '.join(emotion_regulation_vars)
        model = ols(formula, data=df).fit()
        print(f"\nRegression results for {outcome}:")
        print(model.summary())

# Step 3: RQ4 - Moderation Analysis
# Use regression on difference scores to test moderation effects
for bio_var, step_var in variables:
    diff_var = f'diff_{bio_var}'
    formula = f'{diff_var} ~ ' + ' + '.join(emotion_regulation_vars)
    model = ols(formula, data=df).fit()
    print(f"\nModeration analysis for {diff_var}:")
    print(model.summary())
```