Sucrose Only in LHA

Mike Schaid

```
%load_ext autoreload
%autoreload 2
```

```
import datetime
import numpy as np
import pandas as pd
import polars as pl
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from pathlib import Path

pl.enable_string_cache()
```

functions to draw cue box and assign sex

```
from coral.config_loader import ConfigLoader
from coral.experimental_metadata import ExperimentMetaData
from coral.data_preprocessor import BehaviorDataPreprocessor, PhotometryDataPreprocessor
config_path = '/Volumes/fsmresfiles/Basic_Sciences/Phys/Lerner_Lab_tnl2633/Mike/LHA_dopami
configs = ConfigLoader(config_path)
```

```
# Your date is in MM-DD-YYYY format, so use:
sucralose_date = datetime.datetime.strptime(configs.config_data['reward_dates']['sucralose
def draw_cue_box(ax, color, alpha):
    #  draw box on plot for cue
    y_lower = ax.get_ylim()[0]
    y_ags_sum = sum(np.abs(ax.get_ylim()))
    rect = patches.Rectangle(
        (0, y_lower), width=5, height=y_ags_sum, alpha=alpha, facecolor=color)
    ax.add_patch(rect)
    return rect
```

paths to data

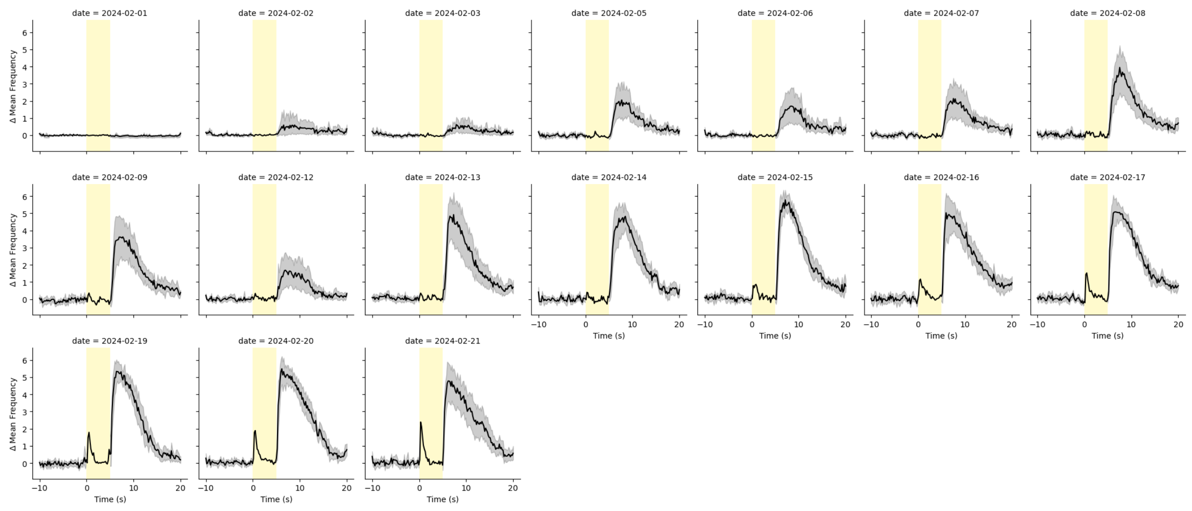
```
behavior_path = Path(
    '/Volumes/fsmresfiles/Basic_Sciences/Phys/Lerner_Lab_tnl2633/Mike/LHA_dopamine/LH_NAC_
fp_path = Path('/Volumes/fsmresfiles/Basic_Sciences/Phys/Lerner_Lab_tnl2633/Mike/LHA_dopam
filter_date = datetime.date(2024, 2, 9)
```

```
bh_df = pl.read_parquet(behavior_path).drop("__index_level_0__").sort('date').sort('subjec


baselines = (bh_df
             .filter(
                 (pl.col('time') <0)
                 &
                 (pl.col('time') > -3))
             .group_by(['subject', 'date']).mean()
             .drop(['time', 'time_recorded', 'user'])
             .sort('date', descending=True)
             )
baseline_corrected_df = (bh_df
                         .filter(pl.col('date') < sucralose_date)
             .join(baselines, on = ['subject', 'date'], how = 'left', suffix = '_baseline'
             .with_columns(
                 (pl.col('encoder_aligned_to_cue') - pl.col('encoder_aligned_to_cue_baseli
                 (pl.col('lick_aligned_to_cue') - pl.col('lick_aligned_to_cue_baseline')).
                 (pl.col('lick_aligned_to_reward') - pl.col('lick_aligned_to_reward_baseli
                 (pl.col('encoder_aligned_to_reward') - pl.col('encoder_aligned_to_reward_
                 (pl.col('subject').cast(pl.Int32)),
                 (pl.col('date').cast(pl.Date))
             )
             .drop(['sucralose_baseline', 'male_baseline','lick_aligned_to_cue', 'encoder_
)
# every_other_day = baseline_corrected_df['date'].unique()[::2]
```

Licks aligned to cue

```
every_other_day = baseline_corrected_df['date'].unique()#[2::2]
```

```
plot_df = (baseline_corrected_df
           .filter(
              #  (pl.col('date')>filter_date)
              #      &
        (pl.col('date').is_in(every_other_day))
           )
           .to_pandas(date_as_object=True)
)


facet = sns.FacetGrid(
                      plot_df,
                      col='date',
                      col_wrap=7,
                      height=3,
                      aspect=1,
                      margin_titles=True)
facet.map(sns.lineplot, 'time',
          'lick_aligned_to_cue_baseline_corrected', color = 'black')
for ax in facet.axes.flat:
    draw_cue_box(ax, color='lemonchiffon', alpha=1)
    ax.set_ylabel(r'$\Delta$ Mean Frequency')
    ax.set_xlabel('Time (s)')
```



```
numb_subjects = baseline_corrected_df.select(pl.col(['subject']).n_unique())
```

```
males = baseline_corrected_df.filter(pl.col('male') == True).select(pl.col(['subject']).n_
females = baseline_corrected_df.filter(pl.col('male') == False).select(pl.col(['subject'])
```

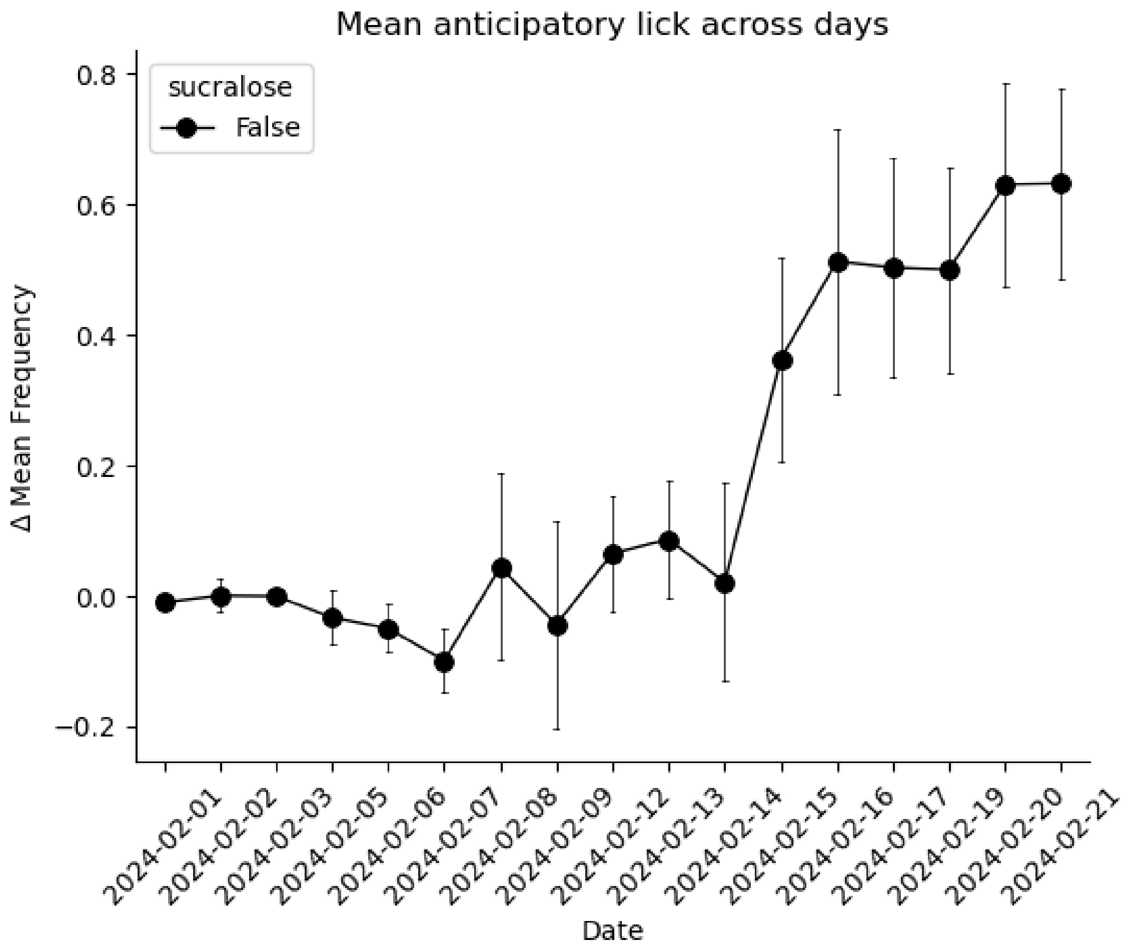
number of subjects in behavior = 10

5 males & 5 females

mean anticipatory licks

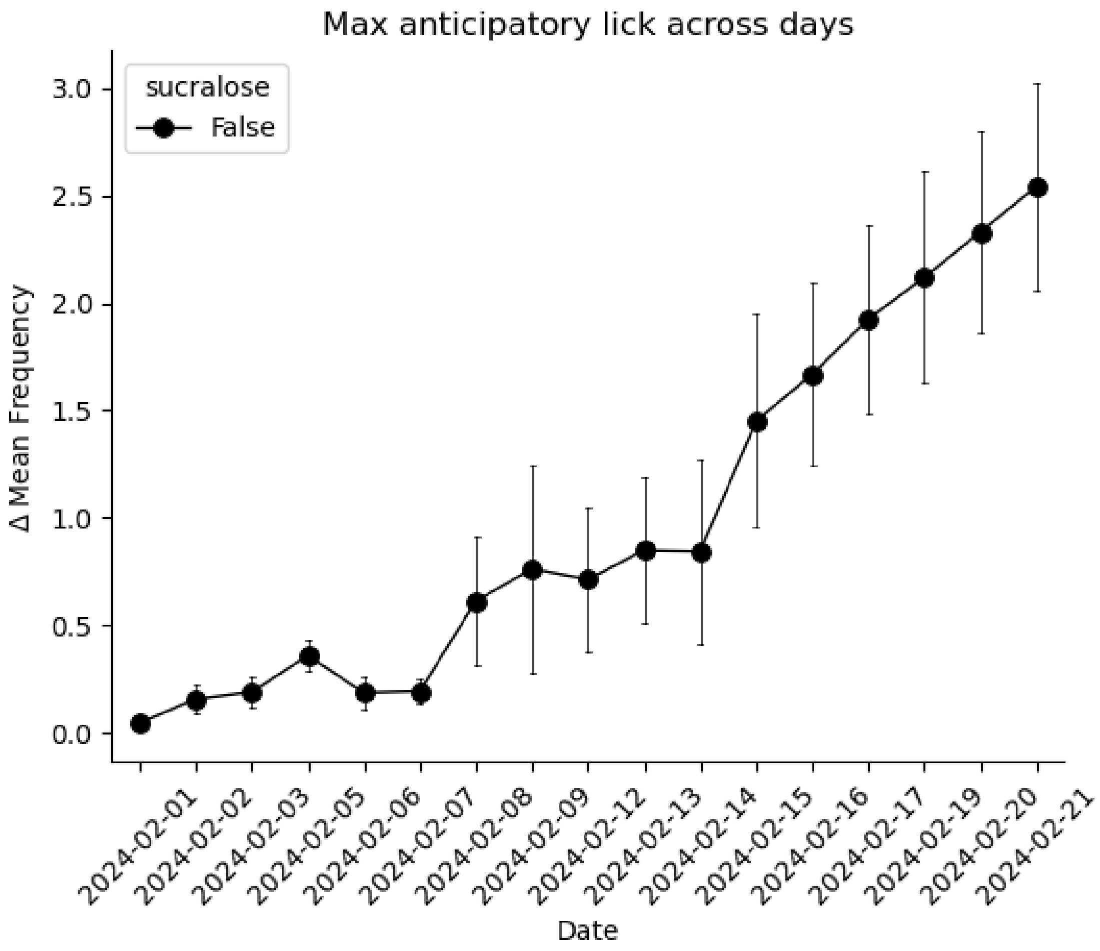
```
grouped_baseline_al =(
    baseline_corrected_df
             .filter(
                 (pl.col('time') >0)
                 &
                 (pl.col('time') <3)
                 &
                 (pl.col('date').is_in(every_other_day))
                )
             .group_by(['subject', 'date', 'sucralose'])
             .agg([
                 pl.col('lick_aligned_to_cue_baseline_corrected').mean().alias(
                     'mean_cue_lick'),
                 pl.col('lick_aligned_to_cue_baseline_corrected').max().alias(
                     'max_cue_lick')
             ])
             .sort('date')
             .with_columns(
                 (pl.col('subject').cast(pl.Int64))
             )
)
grp_bs_al_pd = grouped_baseline_al.to_pandas(date_as_object=True)
def plot_anticipatory_licks(col, title):

    sns.pointplot(data = grp_bs_al_pd,
                x = 'date',
                y=col,
                hue = 'sucralose',
                palette=['black', 'darkred'],
                errorbar='se',
                linewidth=1,
                color = 'black',
                markersize = 7,
                capsize=0.1,
                err_kws = {"color":'black',
                            "linewidth": .5},

                )

    plt.xticks(rotation=45)
    plt.ylabel(r'$\Delta$ Mean Frequency')
    plt.xlabel('Date')
    plt.title(title)
    sns.despine()
    plt.show()
plot_anticipatory_licks('mean_cue_lick', 'Mean anticipatory lick across days')
```

```
/var/folders/_3/4x4mtlsd3n37vfrcjmsz1vcd8clmkl/T/ipykernel_9605/1618349483.py:25: UserWarning
  sns.pointplot(data = grp_bs_al_pd,
```



```
plot_anticipatory_licks('max_cue_lick', 'Max anticipatory lick across days')
```

```
/var/folders/_3/4x4mtlsd3n37vfrcjmsz1vcd8clmkl/T/ipykernel_9605/1618349483.py:25: UserWarning
  sns.pointplot(data = grp_bs_al_pd,
```

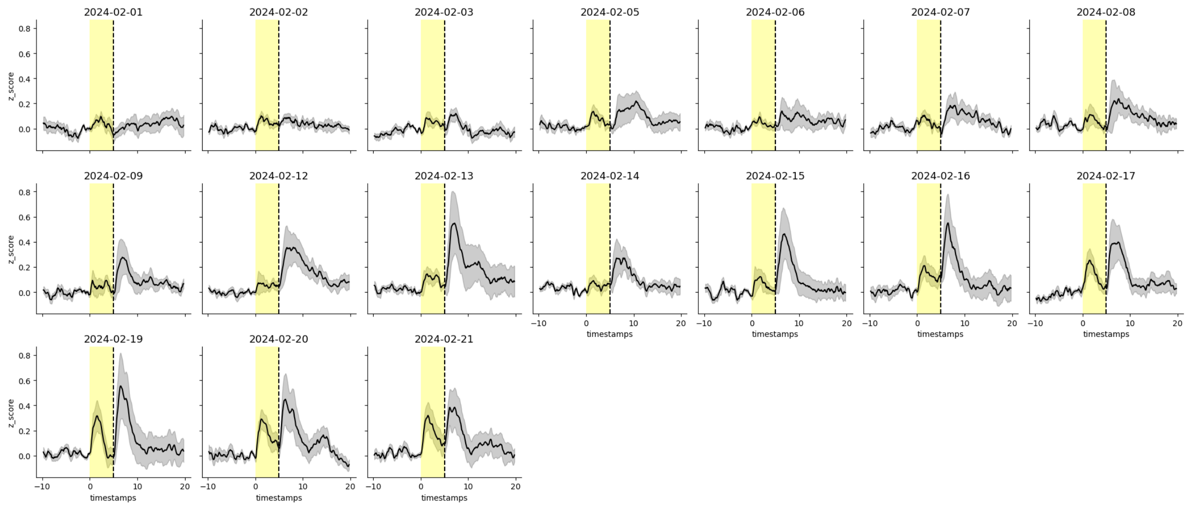


read fp data and calculate group average

```
#processing is dropping males coloumns
fp_df = pl.read_parquet(fp_path).sort('date').sort('subject').filter((pl.col('date') < suc
grouped_fp = (fp_df
              .drop('time', 'trial', 'user')
              .group_by(*[c for c in fp_df.columns if c not in ['z_score', 'time', 'trial'
              .mean()
              .with_columns(pl.col('date').cast(pl.Date))
             #   .to_pandas(date_as_object=True)
)
```

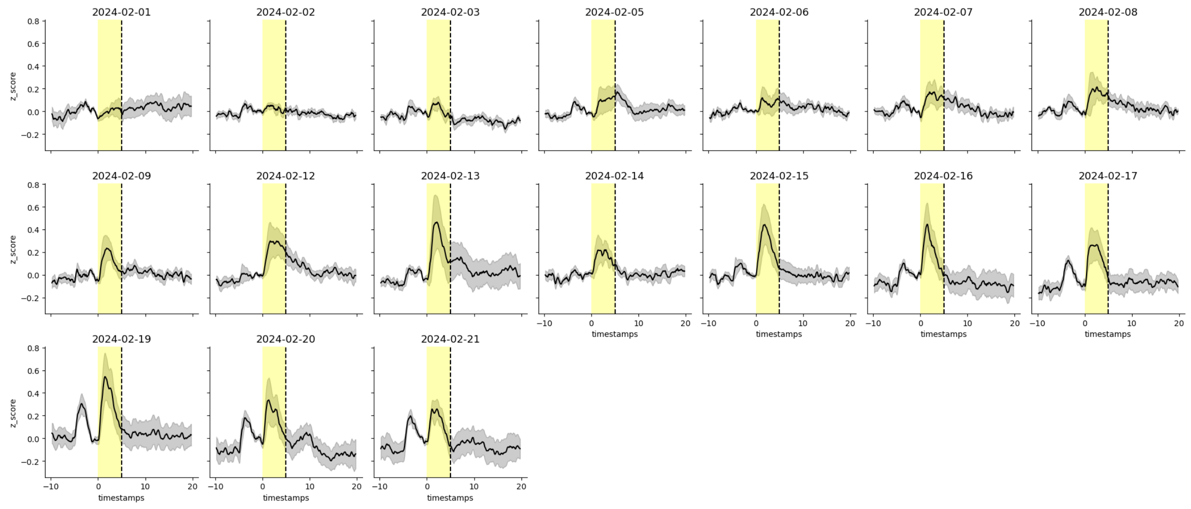
Cue Response

```


def plot_fp_response(event):
    sucralose_start = datetime.datetime.strptime(
        configs.config_data['reward_dates']['sucralose'], '%m-%d-%Y').date()

    filtered_data = (grouped_fp
                     .filter(
                         (pl.col('behavioral_events') == event)
                         &
                         (pl.col('date').is_in(every_other_day))
                         )
                         .sort('date')
                     )



    facet = sns.FacetGrid(filtered_data.to_pandas(date_as_object=True),
                          col='date',
                          col_wrap=7,
                        #   color = 'black',
                          height=3,
                          aspect=1,
                          margin_titles=True)
    facet.map(sns.lineplot, 'timestamps', 'z_score', color = 'black', errorbar='se')
    for ax in facet.axes.flat:
        ax.axvline(5, color='black', linestyle='--')
        draw_cue_box(ax, color='yellow', alpha=0.3)

    for ax in facet.axes.flat:
        label = ax.title.get_text()
        string_date = label.replace('date = ', '').replace(" 00:00:00", '')
        if len(string_date)<1:
            pass
        else:
            date = datetime.datetime.strptime(string_date,"%Y-%m-%d").date()
            if date < sucralose_start:
                ax.set_title(string_date)
                ax.title.set_color('black')
                ax.title.set_fontsize(13)
            else:
                ax.title.set_color('darkred')
                ax.set_title(string_date)
                ax.title.set_fontsize(13)

    plt.tight_layout()
plot_fp_response('cue')
```



Reward Response

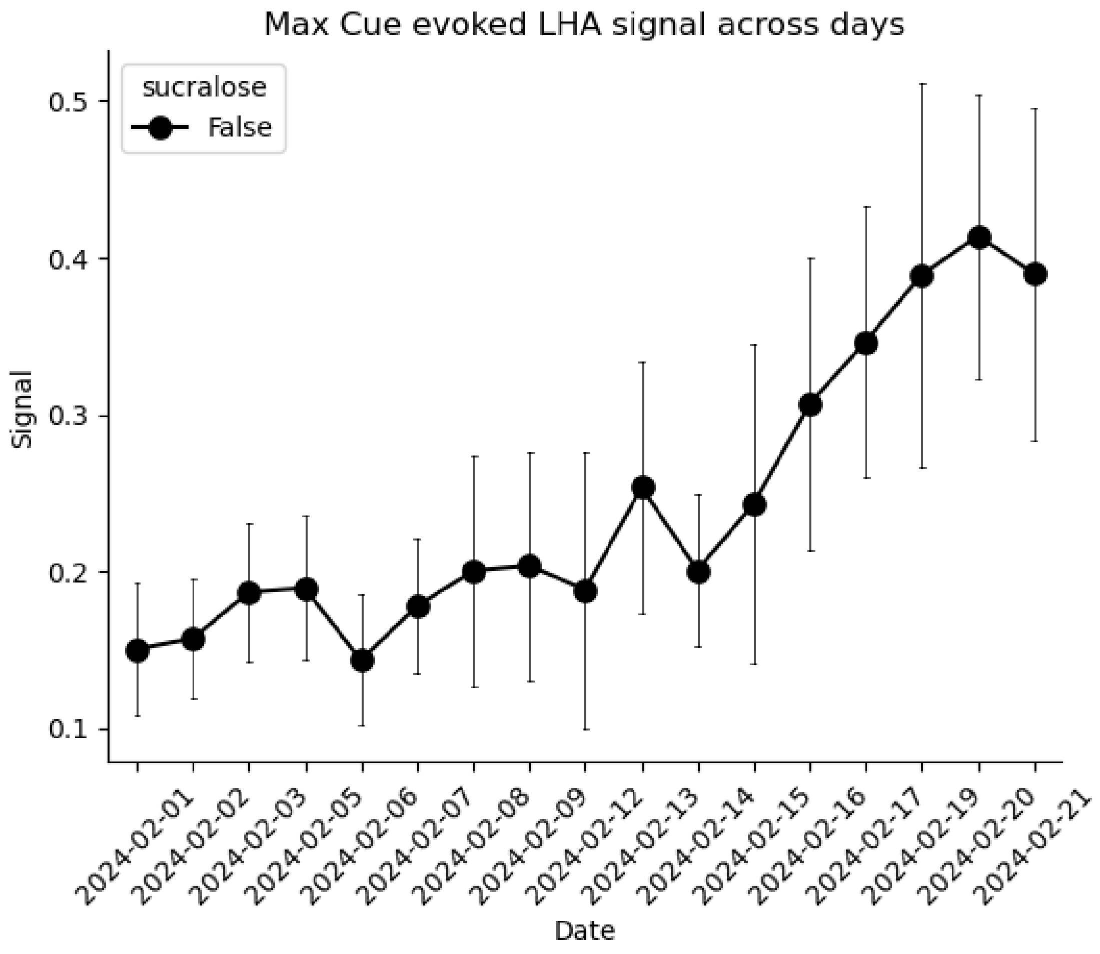
```
plot_fp_response('reward')
```



LHA cue response over days with anticipatory licks

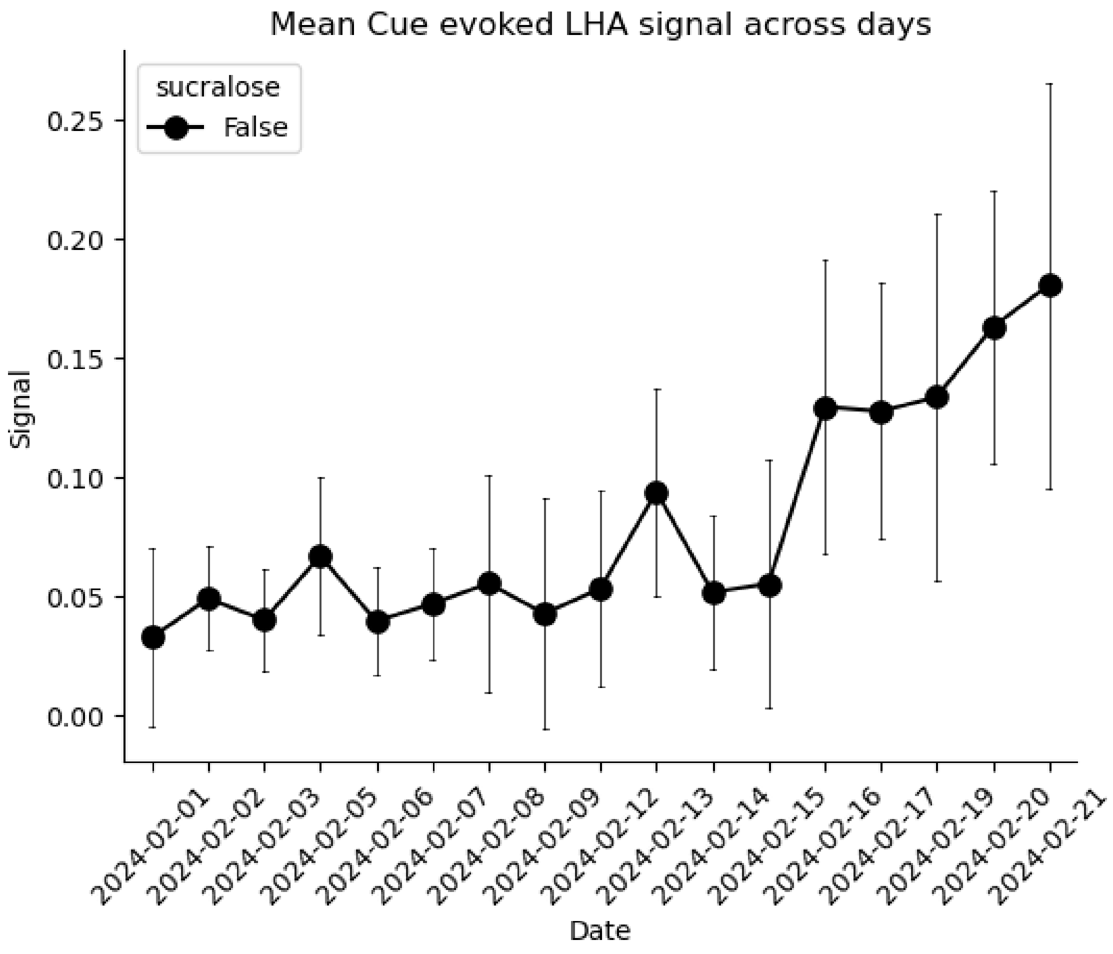
```
fp_anticipatory = (
    grouped_fp
    .rename({'z_score': 'signal'})
    .filter(
        (pl.col('timestamps') > 0)
        &
        (pl.col('timestamps') < 5)
        # &
        # (pl.col('date').is_in(every_other_day)) =
    )
    .group_by(['subject', 'date', 'structures', 'behavioral_events', 'male', 'sucralose'])
    .agg([
        pl.col('signal').max().alias('max_signal'),
        pl.col('signal').mean().alias('mean_signal'),
        pl.col('signal').min().alias('min_signal'),
        pl.col('signal').sum().alias('sum_signal'),
    ])
    .sort('date')
)

def plot_consolidated_fp(y_col):
    sns.pointplot(data=fp_anticipatory.to_pandas().query(
        "structures =='LHA' & behavioral_events == 'cue'"),
                x='date',
                y=y_col,
                errorbar='se',
                linewidth=1.5,
                palette=['black', 'darkred'],
                hue = 'sucralose',
                #   color = 'black',
                capsize=0.1,
                err_kws = {"color":'black',
                            "linewidth": .5},
                **{
                    'marker': 'o',
                    'markersize': 8,
                }
                )
    plt.xticks(rotation=45)
    plt.ylabel('Signal')
    plt.xlabel('Date')
    if y_col == 'max_signal':
        plt.title(f'Max Cue evoked LHA signal across days')
    if y_col == 'mean_signal':
        plt.title(f'Mean Cue evoked LHA signal across days')
    sns.despine()
    plt.show()


plot_consolidated_fp('max_signal')
```

```
/var/folders/_3/4x4mtlsd3n37vfrcjmsz1vcd8clmkl/T/ipykernel_9605/1187630084.py:22: UserWarning
  sns.pointplot(data=fp_anticipatory.to_pandas().query(
```



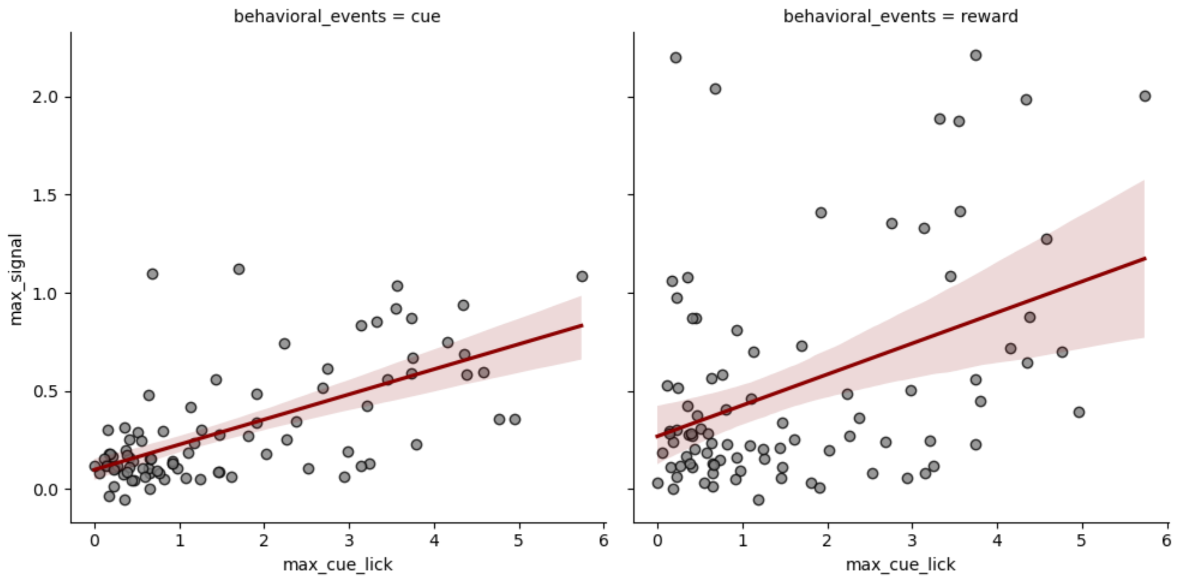
```
plot_consolidated_fp('mean_signal')
```

```
/var/folders/_3/4x4mtlsd3n37vfrcjmsz1vcd8clmkl/T/ipykernel_9605/1187630084.py:22: UserWarning
  sns.pointplot(data=fp_anticipatory.to_pandas().query(
```



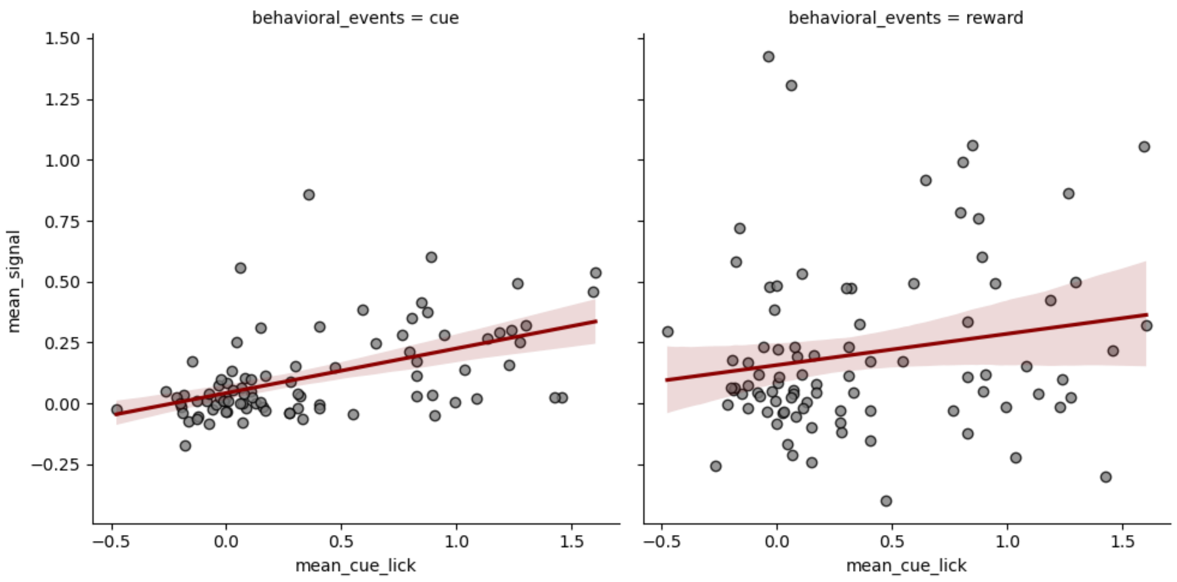
Peak Signal vs Max Anticipatory Licks

```
combined_anticipatory_data = (fp_anticipatory
                 .join(grouped_baseline_al, on = ['subject', 'date']).sort('date')
                 .filter(pl.col('date')>filter_date)
                )

sns.lmplot(data=combined_anticipatory_data.filter(pl.col('date')>filter_date).to_pandas(da
           x='max_cue_lick',
            y = 'max_signal',
            col = 'behavioral_events',
            col_order= ['cue', 'reward'],
            scatter_kws = {'color': 'grey', 'edgecolor': 'black'},
            line_kws = {'color': 'darkred'},
            facet_kws = {
                'sharex':True,
                'sharey':True
            }
            )
sns.despine()
```



Peak Signal vs Max Anticipatory Licks

```
sns.lmplot(data=combined_anticipatory_data.filter(pl.col('date')>filter_date).to_pandas(da
           x='mean_cue_lick',
            y = 'mean_signal',
            col = 'behavioral_events',
            col_order= ['cue', 'reward'],
            scatter_kws = {'color': 'grey', 'edgecolor': 'black'},
            line_kws = {'color': 'darkred'},
            facet_kws = {
                'sharex':True,
                'sharey':True
            }

            )
sns.despine()
```



```
auc = (
    pl.from_pandas(
    grouped_fp
    .filter(
        (pl.col('timestamps') > 0)
        &
        (pl.col('timestamps') < 5)
    )
    .drop('time', 'trial', 'user')
    .sort(['subject', 'behavioral_events', 'structures', 'timestamps', 'date'])
    .to_pandas()
    .groupby(by=['behavioral_events', 'structures', 'subject', 'date'], as_index=False, ob
    .apply(lambda x: np.trapz(x['z_score'], x['timestamps'])).rename({None: 'auc'}, axis=1
    )
    .with_columns(
        (pl.col('date').cast(pl.Date))
        )

)

final_data = combined_anticipatory_data.join(
    auc, on=['subject', 'date', 'structures', 'behavioral_events']).sort('date')


sns.lmplot(data=final_data,
           x='mean_cue_lick',
           y='auc',
           col='structures',
           col_order=['LHA', 'NAC'],
           row='behavioral_events',
           row_order=['cue', 'reward'],
           hue='sucralose',
           palette=['black', 'darkred'],
           facet_kws={
               'sharex': True,
               'sharey': True
           }

           )
plt.xticks(rotation=45)
sns.despine()
```

```
sns.lineplot(final_data.filter(pl.col('behavioral_events')=="cue"),
             x = 'date',
             y = 'auc',
             hue = 'structures',
             palette=['black', 'darkred'],
             linewidth = 1,
             errorbar='se'
             )
```