```
!python --version
```

```
Python 3.10.12
```

```
! pip install -U kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 5.6 MB/s eta 0:00:00
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
```

```
! pip install ONE-api ibllib
```

```
Collecting jmespath<2.0.0,>=0.7.1 (from boto3->ONE-api)
  Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.11.0,>=0.10.0 (from boto3->ONE-api)
  Downloading s3transfer-0.10.4-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: pyjwt<3.0.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pyjwt[crypto]<3.0.0,>=2.0.0->globu
Requirement already satisfied: cryptography!=3.4.0,>=3.3.1 in /usr/local/lib/python3.10/dist-packages (from globus-sdk->ibllib) (43.0
Collecting PyQt5-sip<13,>=12.15 (from pyqt5->ibllib)
  Downloading PyQt5_sip-12.15.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.whl.metadata (421 bytes)
Collecting PyQt5-Qt5<5.16.0,>=5.15.2 (from pyqt5->ibllib)
  Downloading PyQt5_Qt5-5.15.15-py3-none-manylinux2014_x86_64.whl.metadata (536 bytes)
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest->ibllib) (2.0.0)
Requirement already satisfied: pluggy<2,>=1.5 in /usr/local/lib/python3.10/dist-packages (from pytest->ibllib) (1.5.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/python3.10/dist-packages (from pytest->ibllib) (1.2.2)
Requirement already satisfied: tomli>=1 in /usr/local/lib/python3.10/dist-packages (from pytest->ibllib) (2.1.0)
Requirement already satisfied: networkx>=2.8 in /usr/local/lib/python3.10/dist-packages (from scikit-image->ibllib) (3.4.2)
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-packages (from scikit-image->ibllib) (2.36.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.10/dist-packages (from scikit-image->ibllib) (2024.9.20)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.10/dist-packages (from scikit-image->ibllib) (0.4)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography!=3.4.0,>=3.3.1->globus-sdk->ib
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.3->ibll
Requirement already satisfied: cloudpickle>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from dask->phylib>=2.6.0->ibllib) (3.1.0
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dist-packages (from dask->phylib>=2.6.0->ibllib) (2024.1
Requirement already satisfied: partd>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from dask->phylib>=2.6.0->ibllib) (1.4.2)
Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/python3.10/dist-packages (from dask->phylib>=2.6.0->ibllib
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels->slidingRP>=1.1.1->ibllib) (1
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography!=3.4.0,>=3.3.1->glo
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata>=4.13.0->dask->phylib>=2
Requirement already satisfied: locket in /usr/local/lib/python3.10/dist-packages (from partd>=1.4.0->dask->phylib>=2.6.0->ibllib) (1.0
Downloading ONE_api-2.11.1-py3-none-any.whl (119 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 119.6/119.6 kB 10.6 MB/s eta 0:00:00
Downloading ibllib-3.0.0-py3-none-any.whl (8.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 92.6 MB/s eta 0:00:00
Downloading colorlog-6.9.0-py3-none-any.whl (11 kB)
Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.7/57.7 kB 6.3 MB/s eta 0:00:00
Downloading ibl_neuropixel-1.5.0-py3-none-any.whl (98 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.8/98.8 kB 11.5 MB/s eta 0:00:00
Downloading iblatlas-0.5.5-py3-none-any.whl (174 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 174.5/174.5 kB 18.4 MB/s eta 0:00:00
Downloading iblutil-1.14.0-py3-none-any.whl (49 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.8/49.8 kB 5.5 MB/s eta 0:00:00
Downloading mtscomp-1.0.2-py2.py3-none-any.whl (16 kB)
Downloading phylib-2.6.0-py2.py3-none-any.whl (80 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.4/80.4 kB 9.3 MB/s eta 0:00:00
Downloading pynrrd-1.1.1-py3-none-any.whl (23 kB)
Downloading slidingRP-1.1.1-py3-none-any.whl (40 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.2/40.2 kB 4.4 MB/s eta 0:00:00
Downloading boto3-1.35.75-py3-none-any.whl (139 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.2/139.2 kB 16.1 MB/s eta 0:00:00
Downloading globus_sdk-3.49.0-py3-none-any.whl (392 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 392.1/392.1 kB 35.4 MB/s eta 0:00:00
Downloading imagecodecs-2024.9.22-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (43.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.3/43.3 MB 13.0 MB/s eta 0:00:00
Downloading Psychofit-1.0.0.post0-py3-none-any.whl (5.7 kB)
Downloading PyQt5-5.15.11-cp38-abi3-manylinux_2_17_x86_64.whl (8.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 86.0 MB/s eta 0:00:00
Downloading sparse-0.15.4-py2.py3-none-any.whl (237 kB)
```

```
# Instanciate the ONE cache
from pathlib import Path
ibl_cache = Path.home() / 'Downloads' / 'IBL_Cache'
ibl_cache.mkdir(exist_ok=True, parents=True)
```

```
# Setup the ONE API
from one.api import ONE
ONE.setup()
```

```
Param ALYX_URL, current value is ["https://openalyx.internationalbrainlab.org"]:
Param ALYX_LOGIN, current value is ["intbrainlab"]:
Param HTTP_DATA_SERVER, current value is ["https://ibl.flatironinstitute.org/public"]:
Param HTTP_DATA_SERVER_LOGIN, current value is ["None"]:
Enter the FlatIron HTTP password for None (leave empty to keep current): ··········
Enter the location of the download cache, current value is ["/root/Downloads/ONE/openalyx.internationalbrainlab.org"]:
Would you like to set this URL as the default one? [Y/n]Y
Are the above settings correct? [Y/n]Y
ONE Parameter files location: /root/.one
IBLParams(CLIENT_MAP={'openalyx.internationalbrainlab.org': '/root/Downloads/ONE/openalyx.internationalbrainlab.org'},
DEFAULT='openalyx.internationalbrainlab.org')
```

```
from brainbox.io.one import SpikeSortingLoader, SessionLoader
from brainbox.ephys_plots import plot_brain_regions
from brainbox.task.trials import get_event_aligned_raster, get_psth
#from ibllib.atlas import AllenAtlas <-- deprecated
from iblatlas.atlas import AllenAtlas
import numpy as np
import pandas as pd
from brainbox.task.trials import find_trial_ids
import plotly.express as px
```

```
one = ONE(base_url='https://openalyx.internationalbrainlab.org', \
          password='international', silent=True, cache_dir=ibl_cache)
```

```
ba = AllenAtlas()
```

```
Downloading: /root/Downloads/ONE/openalyx.internationalbrainlab.org/histology/ATLAS/Needles/Allen/average_template_25.nrrd Bytes: 329989
100%|██████████| 31.470260620117188/31.470260620117188 [00:01<00:00, 16.40it/s]
Downloading: /root/Downloads/ONE/openalyx.internationalbrainlab.org/histology/ATLAS/Needles/Allen/annotation_25.nrrd Bytes: 4035363
100%|██████████| 3.848422050476074/3.848422050476074 [00:00<00:00,  4.91it/s]
```

```
insertions = one.search_insertions(atlas_acronym='SCdg', query_type='remote')
```

```
pid = insertions[32]
```

```
[eid, pname] = one.pid2eid(pid)
```

```
ssl = SpikeSortingLoader(pid=pid, one=one, atlas=ba)
spikes, clusters, channels = ssl.load_spike_sorting()
clusters = ssl.merge_clusters(spikes, clusters, channels)
```

```
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/spikes.amps.npy: 100%|██████████| 36.7M/36
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/spikes.clusters.npy: 100%|██████████| 18.4
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/spikes.depths.npy: 100%|██████████| 36.7M/
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/spikes.times.npy: 100%|██████████| 36.7M/3
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/clusters.channels.npy: 100%|██████████| 12
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/clusters.depths.npy: 100%|██████████| 6.16
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/clusters.metrics.pqt: 100%|██████████| 108
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/clusters.uuids.csv: 100%|██████████| 55.8k
/usr/local/lib/python3.10/dist-packages/one/util.py:543: ALFWarning: Multiple revisions: "", "2024-05-06"
  warnings.warn(f'Multiple revisions: {rev_list}', alferr.ALFWarning)
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/channels.brainLocationIds_ccf_2017.npy: 10
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/#2024-05-06#/channels.labels.npy: 100%|
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/channels.localCoordinates.npy: 100%|
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/channels.mlapdv.npy: 100%|██████████| 4.74
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/pykilosort/channels.rawInd.npy: 100%|██████████| 3.20
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/electrodeSites.brainLocationIds_ccf_2017.npy: 100%|
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/electrodeSites.mlapdv.npy: 100%|██████████| 4.74k/4.7
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/probe00/electrodeSites.localCoordinates.npy: 100%|
/usr/local/lib/python3.10/dist-packages/one/util.py:543: ALFWarning: Multiple revisions: "", "2024-05-06"
  warnings.warn(f'Multiple revisions: {rev_list}', alferr.ALFWarning)
```

```
good_cluster_idx = clusters['label'] == 1
good_cluster_IDs = clusters['cluster_id'][good_cluster_idx]
```

```
clusters_g = {key: val[good_cluster_idx] for key, val in clusters.items()}
good_spk_indx = np.where(np.isin(spikes['clusters'], good_cluster_IDs))
```

```
spikes_g = {key: val[good_spk_indx] for key, val in spikes.items()}


sl = SessionLoader(eid=eid, one=one)
sl.load_trials()
events = sl.trials['firstMovement_times']
```

```
/usr/local/lib/python3.10/dist-packages/one/util.py:543: ALFWarning: Multiple revisions: "", "2024-07-15"
  warnings.warn(f'Multiple revisions: {rev_list}', alferr.ALFWarning)
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/_ibl_trials.goCueTrigger_times.npy: 100%|██████████| 6.74k/6.
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/_ibl_trials.stimOff_times.npy: 100%|██████████| 6.74k/6.74k [
(S3) /root/Downloads/IBL_Cache/cortexlab/Subjects/KS022/2019-12-09/001/alf/#2024-07-15#/_ibl_trials.table.pqt: 100%|██████████| 64.8k/64
```

```
nan_index = np.where(np.isnan(events))[0]
events = events.drop(index=nan_index).to_numpy()
choice = sl.trials.choice.drop(index=nan_index).to_numpy()


good_cluster_df = pd.DataFrame(clusters_g)
SCdg_df = good_cluster_df[good_cluster_df['acronym'] == 'SCdg']
SCiw_df = good_cluster_df[good_cluster_df['acronym'] == 'SCiw']


order = 'trial num'
trial_idx, dividers = find_trial_ids(sl.trials, sort='choice', order=order)


selected_cluster_id = SCdg_df['cluster_id'].iloc[0]
spikes_idx = spikes['clusters'] == selected_cluster_id


pre_time = -0.5
post_time = 3
raster_bin = 0.07


import os

if not os.path.exists("images"):
    os.mkdir("images")
```

SCDG

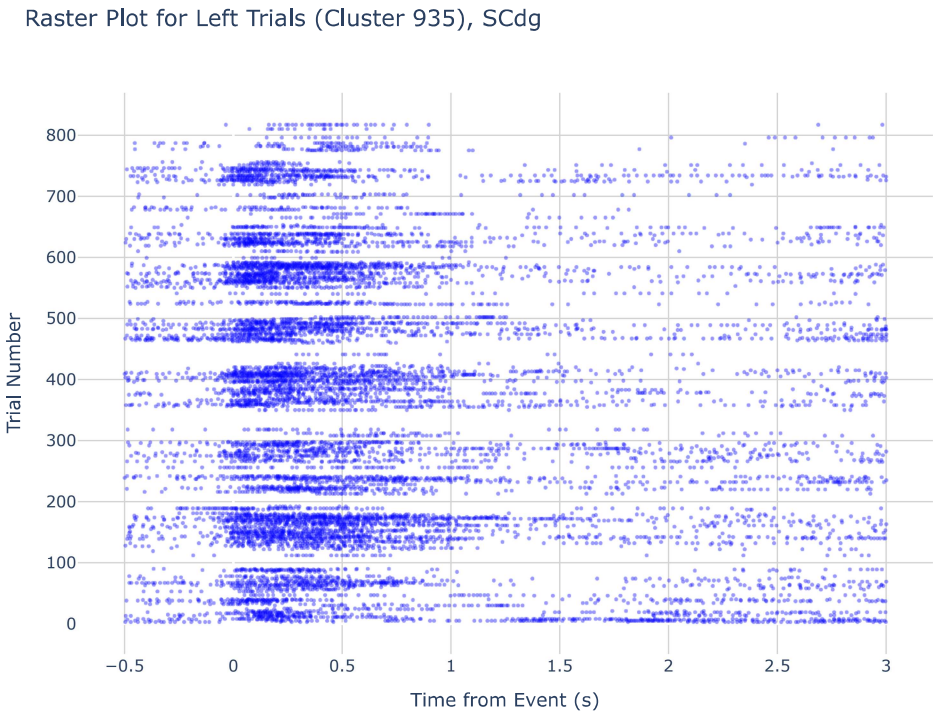
```
aligned_spikes = []
for t in trial_idx:
    # Ensure the index 't' is within the valid range of 'events'
    if 0 <= t < len(events):
        trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time)]
        aligned_spikes.extend([(spike - events[t], t) for spike in trial_spikes])
    else:
        continue

raster_df = pd.DataFrame(aligned_spikes, columns=["Time (s)", "Trial"])


raster_fig = px.scatter(
    raster_df,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Cluster {selected_cluster_id}, SCdg",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["black"],
    opacity=0.2,
    width=1080,
    height=800
)

raster_fig.update_traces(marker=dict(size=0.75)) # Adjust marker size
raster_fig.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray') # Customize x-axis grid
raster_fig.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray') # Customize y-axis grid
raster_fig.update_layout(plot_bgcolor='white')  # Set plot background color to white
raster_fig.show()

raster_fig.write_image(f"images/Raster Plot for Cluster {selected_cluster_id}, SCdg.png")
```

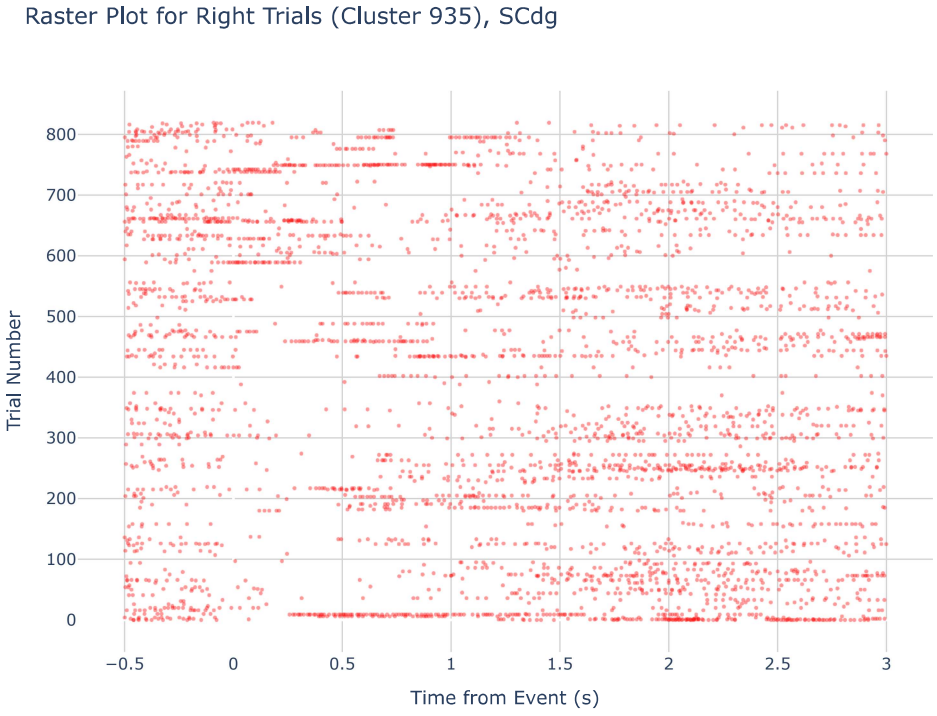
```
Buffered data was truncated after reaching the output size limit.
```

```
hist_fig = px.histogram(raster_df, x="Time (s)", nbins=1000, title= f"Spike Time Histogram for Cluster {selected_cluster_id}, SCdg",color_di
hist_fig.update_layout(xaxis_title="Time from Event (s)", yaxis_title="Spike Count")
hist_fig.show()
hist_fig.write_image(f"images/Spike Histogram for Cluster {selected_cluster_id}, SCdg.png")
```

```
Buffered data was truncated after reaching the output size limit.
```

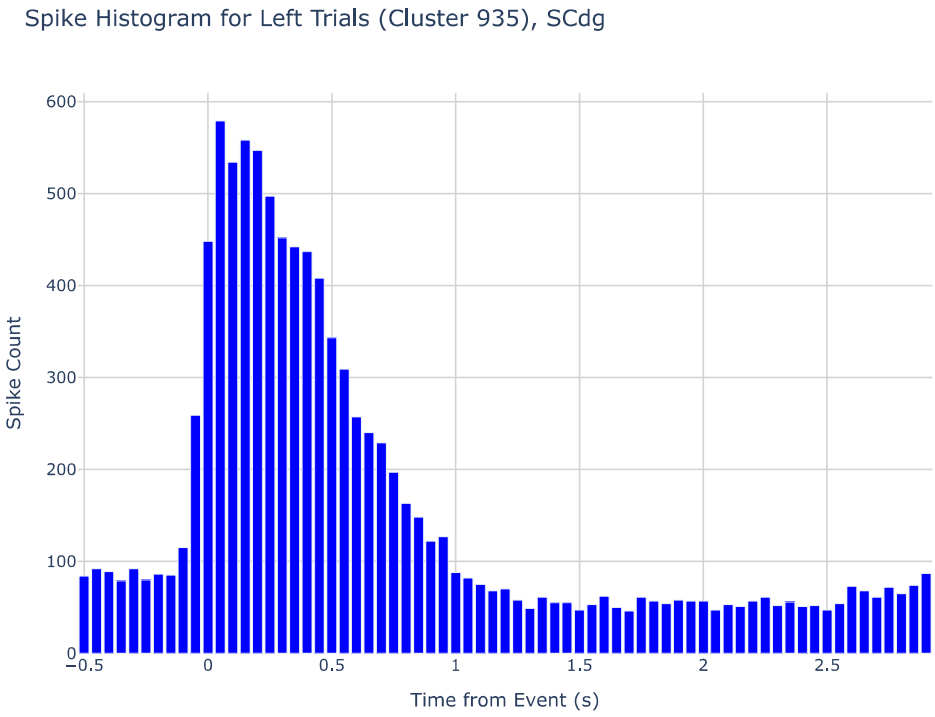
```
pre_time = -0.5
post_time = 3
selected_cluster_id = SCdg_df['cluster_id'].iloc[0]
spikes_idx = spikes['clusters'] == selected_cluster_id

# Separate trials by choice (-1: left, 1: right)
left_trials = np.where(choice == -1)[0]
right_trials = np.where(choice == 1)[0]

# Prepare raster data for left trials
aligned_spikes_left = []
for t in left_trials:
    trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
    aligned_spikes_left.extend([(spike - events[t], t) for spike in trial_spikes])

raster_df_left = pd.DataFrame(aligned_spikes_left, columns=["Time (s)", "Trial"])

# Prepare raster data for right trials
aligned_spikes_right = []
for t in right_trials:
    trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
    aligned_spikes_right.extend([(spike - events[t], t) for spike in trial_spikes])

raster_df_right = pd.DataFrame(aligned_spikes_right, columns=["Time (s)", "Trial"])

# Plot raster for left trials
raster_fig_left = px.scatter(
    raster_df_left,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Left Trials (Cluster {selected_cluster_id}), SCdg",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["blue"],
    opacity=0.4,
    width=800,
    height=600
)
raster_fig_left.update_traces(marker=dict(size=3))  # Adjust marker size
raster_fig_left.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_left.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_left.update_layout(plot_bgcolor='white')
raster_fig_left.show()

title=f"Raster Plot for Left Trials (Cluster {selected_cluster_id}), SCdg"
raster_fig_left.write_image(f"images/{title}.png")

# Plot raster for right trials
raster_fig_right = px.scatter(
    raster_df_right,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Right Trials (Cluster {selected_cluster_id}), SCdg",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["red"],
    opacity=0.4,
    width=800,
    height=600
)
raster_fig_right.update_traces(marker=dict(size=3))  # Adjust marker size
raster_fig_right.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_right.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_right.update_layout(plot_bgcolor='white')
raster_fig_right.show()

title=f"Raster Plot for Right Trials (Cluster {selected_cluster_id}), SCdg"
raster_fig_right.write_image(f"images/{title}.png")
```

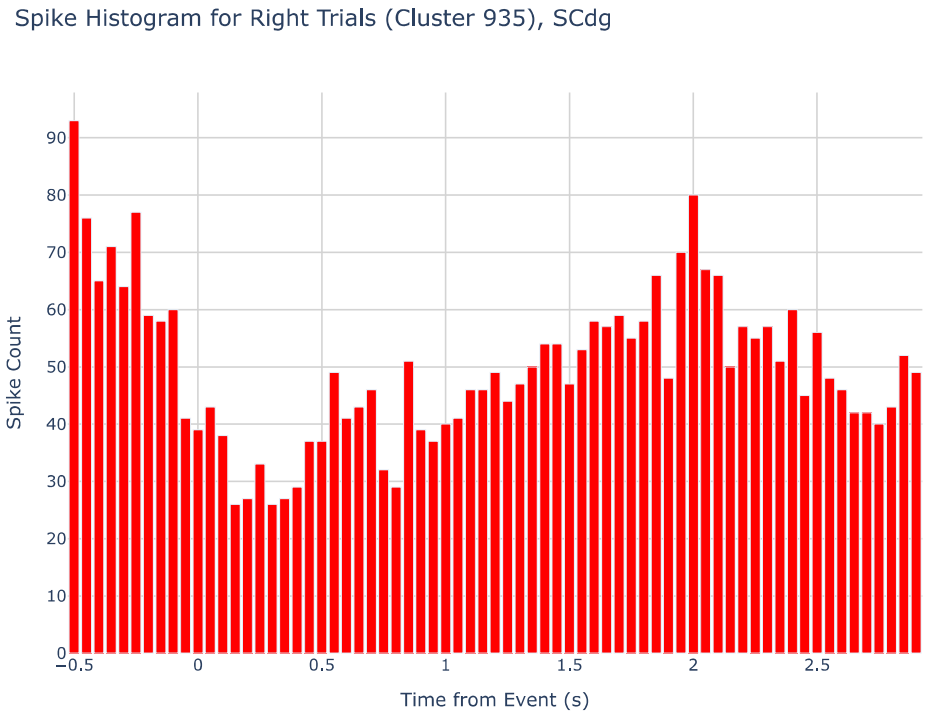





```
# Bin size and time alignment
bin_size = 0.05
time_bins = np.arange(pre_time, post_time, bin_size)

# Histogram data for left trials
spike_times_left = [
    spike - events[t]
    for t in left_trials
    for spike in spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
]
psth_counts_left, _ = np.histogram(spike_times_left, bins=time_bins)
```

```
psth_df_left = pd.DataFrame({
    "Time (s)": time_bins[:-1],
    "Spike Count": psth_counts_left
})

# Histogram data for right trials
spike_times_right = [
    spike - events[t]
    for t in right_trials
    for spike in spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
]
psth_counts_right, _ = np.histogram(spike_times_right, bins=time_bins)

psth_df_right = pd.DataFrame({
    "Time (s)": time_bins[:-1],
    "Spike Count": psth_counts_right
})

# Plot histogram for left trials
hist_fig_left = px.bar(
    psth_df_left,
    x="Time (s)",
    y="Spike Count",
    title=f"Spike Histogram for Left Trials (Cluster {selected_cluster_id}), SCdg",
    labels={"Time (s)": "Time from Event (s)", "Spike Count": "Spike Count"},
    color_discrete_sequence=["blue"],
    width=800,
    height=600
)
hist_fig_left.update_xaxes(showgrid=True, gridcolor='LightGray')
hist_fig_left.update_yaxes(showgrid=True, gridcolor='LightGray')
hist_fig_left.update_layout(plot_bgcolor='white')
hist_fig_left.show()

title=f"Spike Histogram for Left Trials (Cluster {selected_cluster_id}), SCdg"
hist_fig_left.write_image(f"images/{title}.png")


# Plot histogram for right trials
hist_fig_right = px.bar(
    psth_df_right,
    x="Time (s)",
    y="Spike Count",
    title=f"Spike Histogram for Right Trials (Cluster {selected_cluster_id}), SCdg",
    labels={"Time (s)": "Time from Event (s)", "Spike Count": "Spike Count"},
    color_discrete_sequence=["red"],
    width=800,
    height=600
)
hist_fig_right.update_xaxes(showgrid=True, gridcolor='LightGray')
hist_fig_right.update_yaxes(showgrid=True, gridcolor='LightGray')
hist_fig_right.update_layout(plot_bgcolor='white')
hist_fig_right.show()

title=f"Spike Histogram for Right Trials (Cluster {selected_cluster_id}), SCdg"
hist_fig_right.write_image(f"images/{title}.png")
```

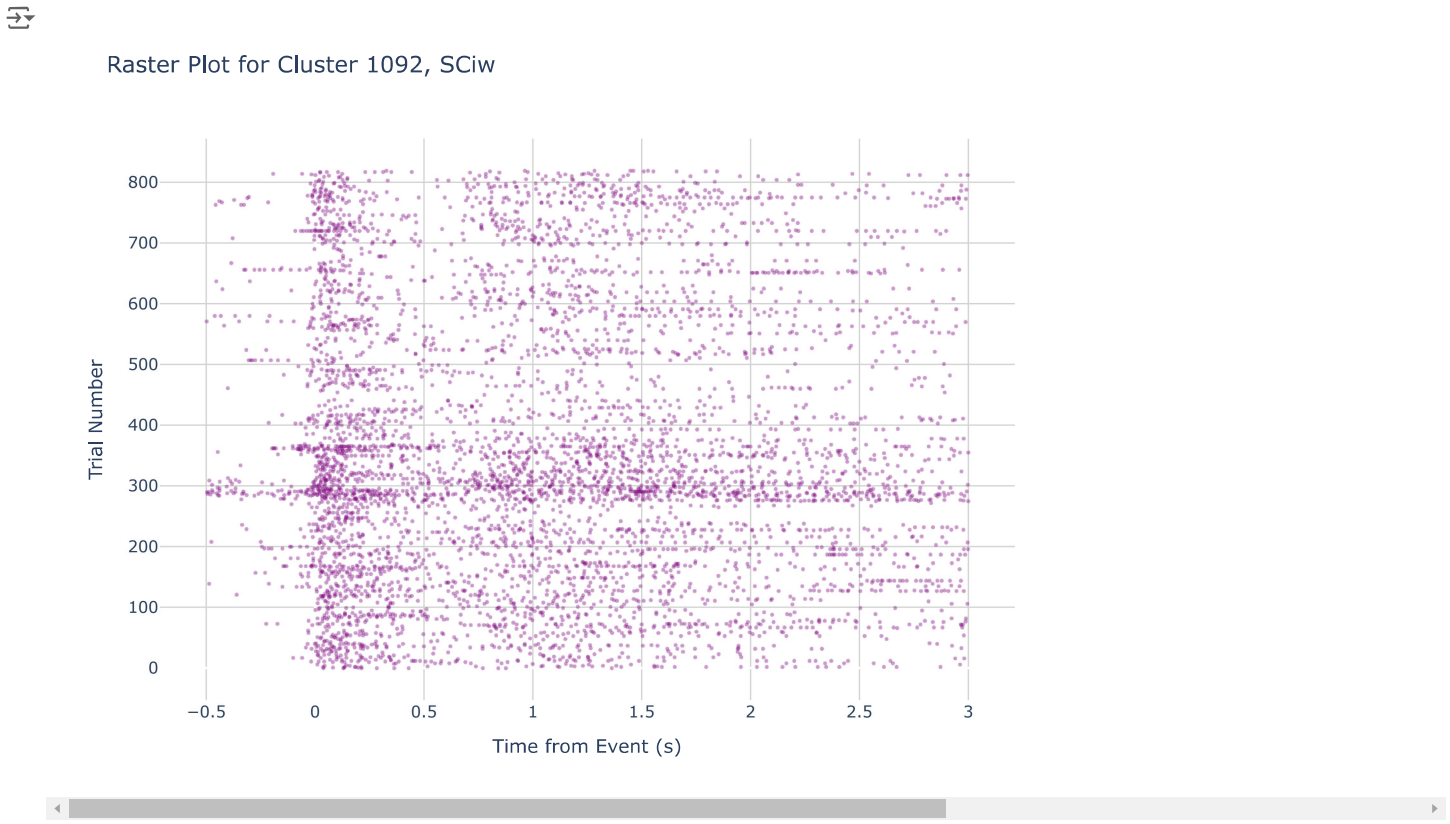




SCIW

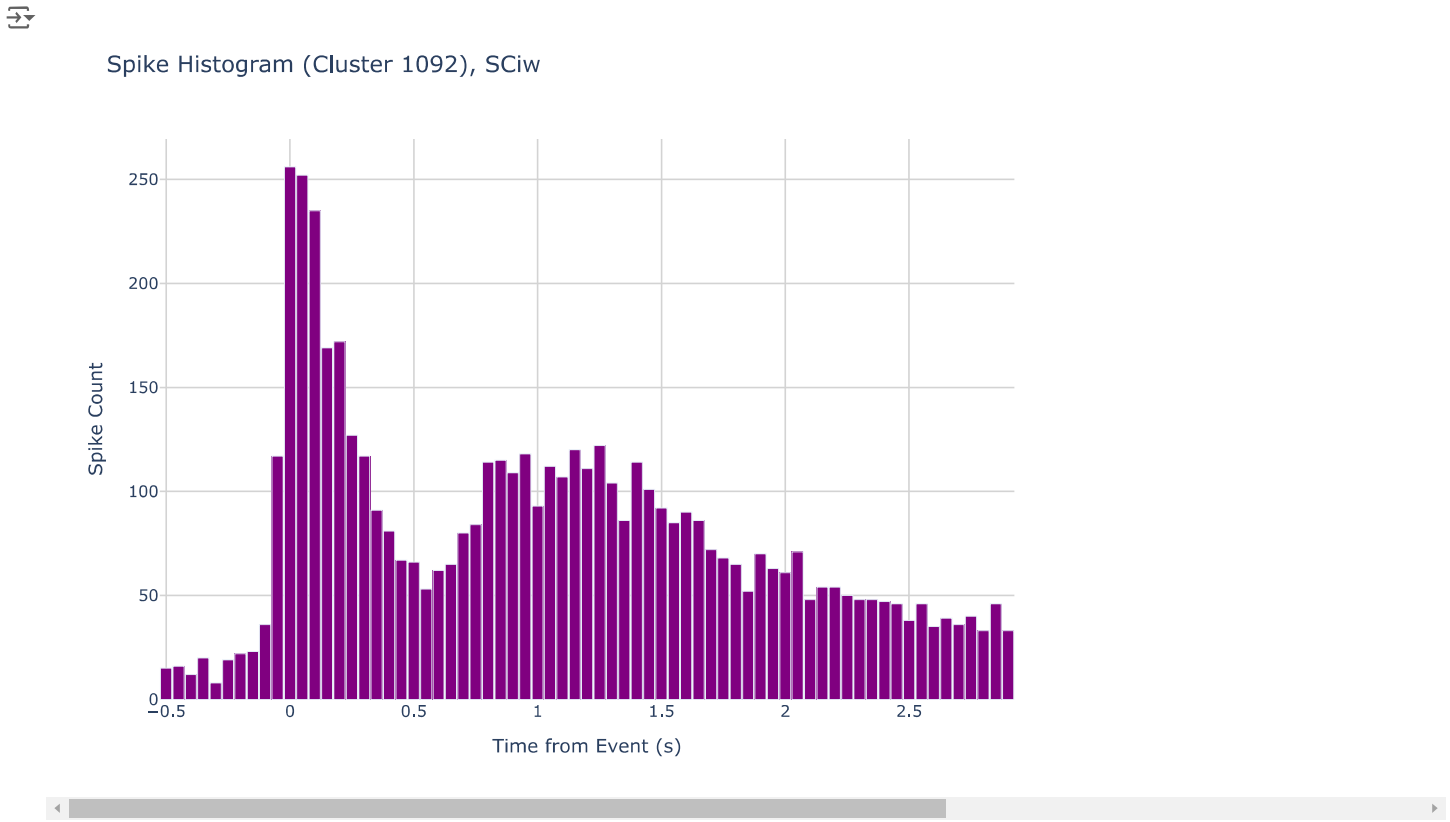
```
selected_cluster_id = SCiw_df['cluster_id'].iloc[0]
spikes_idx = spikes['clusters'] == selected_cluster_id


# Prepare raster data for SCiw
aligned_spikes_sciw = []
for t in trial_idx:
    # Check if t is a valid index for events
```

```
        if 0 <= t < len(events):  # Ensure t is within the valid range
            trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
            aligned_spikes_sciw.extend([(spike - events[t], t) for spike in trial_spikes])
        else:
            continue

raster_df_sciw = pd.DataFrame(aligned_spikes_sciw, columns=["Time (s)", "Trial"])

# Plot raster for SCiw
raster_fig_sciw = px.scatter(
    raster_df_sciw,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Cluster {selected_cluster_id}, SCiw",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["purple"],  # Set marker color to purple
    opacity=0.4,
    width=800,
    height=600
)
raster_fig_sciw.update_traces(marker=dict(size=3))  # Adjust marker size
raster_fig_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_sciw.update_layout(plot_bgcolor='white')
raster_fig_sciw.show()

raster_fig_sciw.write_image(f"images/Raster Plot for Cluster {selected_cluster_id}, SCiw.png")
```




```
# Prepare histogram data for SCiw
spike_times_sciw = [
    spike - events[t]
    for t in trial_idx
    if 0 <= t < len(events) # Check if trial index is within bounds of events
    for spike in spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
]
psth_counts_sciw, _ = np.histogram(spike_times_sciw, bins=time_bins)

psth_df_sciw = pd.DataFrame({
    "Time (s)": time_bins[:-1],
    "Spike Count": psth_counts_sciw
})

# Plot histogram for SCiw
hist_fig_sciw = px.bar(
```

```
    psth_df_sciw,
    x="Time (s)",
    y="Spike Count",
    title=f"Spike Histogram (Cluster {selected_cluster_id}), SCiw",
    labels={"Time (s)": "Time from Event (s)", "Spike Count": "Spike Count"},
    width=800,
    height=600,
    color_discrete_sequence=["purple"]  # Set the color to purple
)
hist_fig_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
hist_fig_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
hist_fig_sciw.update_layout(plot_bgcolor='white', bargap=0.1)
hist_fig_sciw.show()

hist_fig_sciw.write_image(f"images/Spike Histogram for Cluster {selected_cluster_id}, SCiw.png")
```




```
# Prepare raster data for left trials (SCiw)
aligned_spikes_left_sciw = []
for t in left_trials:
    trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
    aligned_spikes_left_sciw.extend([(spike - events[t], t) for spike in trial_spikes])

raster_df_left_sciw = pd.DataFrame(aligned_spikes_left_sciw, columns=["Time (s)", "Trial"])

# Raster plot for left trials (SCiw)
raster_fig_left_sciw = px.scatter(
    raster_df_left_sciw,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Left Trials (Cluster {selected_cluster_id}), SCiw",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["blue"],  # Blue for left trials
    opacity=0.4,
    width=800,
    height=600
)
raster_fig_left_sciw.update_traces(marker=dict(size=3))
raster_fig_left_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_left_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_left_sciw.update_layout(plot_bgcolor='white')
raster_fig_left_sciw.show()

title=f"Raster Plot for Left Trials (Cluster {selected_cluster_id}), SCiw"
raster_fig_left_sciw.write_image(f"images/{title}.png")
```

```
# Prepare histogram data for left trials (SCiw)
spike_times_left_sciw = [
    spike - events[t]
    for t in left_trials
    for spike in spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
]
psth_counts_left_sciw, _ = np.histogram(spike_times_left_sciw, bins=time_bins)

psth_df_left_sciw = pd.DataFrame({
    "Time (s)": time_bins[:-1],
    "Spike Count": psth_counts_left_sciw
})

# Histogram for left trials (SCiw)
hist_fig_left_sciw = px.bar(
    psth_df_left_sciw,
    x="Time (s)",
    y="Spike Count",
    title=f"Spike Histogram for Left Trials (Cluster {selected_cluster_id}), SCiw",
    labels={"Time (s)": "Time from Event (s)", "Spike Count": "Spike Count"},
    width=800,
    height=600,
    color_discrete_sequence=["blue"]  # Blue for left trials
)
hist_fig_left_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
hist_fig_left_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
hist_fig_left_sciw.update_layout(plot_bgcolor='white', bargap=0.1)
hist_fig_left_sciw.show()

title=f"Spike Histogram for Left Trials (Cluster {selected_cluster_id}), SCiw"
hist_fig_left_sciw.write_image(f"images/{title}.png")

# Prepare raster data for right trials (SCiw)
aligned_spikes_right_sciw = []
for t in right_trials:
    trial_spikes = spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
    aligned_spikes_right_sciw.extend([(spike - events[t], t) for spike in trial_spikes])

raster_df_right_sciw = pd.DataFrame(aligned_spikes_right_sciw, columns=["Time (s)", "Trial"])

# Raster plot for right trials (SCiw)
raster_fig_right_sciw = px.scatter(
    raster_df_right_sciw,
    x="Time (s)",
    y="Trial",
    title=f"Raster Plot for Right Trials (Cluster {selected_cluster_id}), SCiw",
    labels={"Time (s)": "Time from Event (s)", "Trial": "Trial Number"},
    color_discrete_sequence=["red"],  # Red for right trials
    opacity=0.4,
    width=800,
    height=600
)
raster_fig_right_sciw.update_traces(marker=dict(size=3))
raster_fig_right_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_right_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
raster_fig_right_sciw.update_layout(plot_bgcolor='white')
raster_fig_right_sciw.show()

title=f"Raster Plot for Right Trials (Cluster {selected_cluster_id}, SCiw)"
raster_fig_right_sciw.write_image(f"images/{title}.png")

# Prepare histogram data for right trials (SCiw)
spike_times_right_sciw = [
    spike - events[t]
    for t in right_trials
    for spike in spikes['times'][(spikes['times'] >= events[t] + pre_time) & (spikes['times'] <= events[t] + post_time) & spikes_idx]
]
psth_counts_right_sciw, _ = np.histogram(spike_times_right_sciw, bins=time_bins)

psth_df_right_sciw = pd.DataFrame({
    "Time (s)": time_bins[:-1],
    "Spike Count": psth_counts_right_sciw
})

# Histogram for right trials (SCiw)
hist_fig_right_sciw = px.bar(
```

```
        psth_df_right_sciw,
        x="Time (s)",
        y="Spike Count",
        title=f"Spike Histogram for Right Trials (Cluster {selected_cluster_id}), SCiw",
        labels={"Time (s)": "Time from Event (s)", "Spike Count": "Spike Count"},
        width=800,
        height=600,
        color_discrete_sequence=["red"]  # Red for right trials
    )
    hist_fig_right_sciw.update_xaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
    hist_fig_right_sciw.update_yaxes(showgrid=True, gridwidth=1, gridcolor='LightGray')
    hist_fig_right_sciw.update_layout(plot_bgcolor='white', bargap=0.1)
    hist_fig_right_sciw.show()

    hist_fig_right_sciw.write_image(f"images/Spike Histogram for Right Trials Cluster {selected_cluster_id}, SCiw.png")
```