# perfusion_stability

June 15, 2023

[1]:
```
%load_ext autoreload
%autoreload 2
```

[2]:
```
import os
import time
import h5py
import numpy as np
import pandas as pd
import nibabel as nib
import seaborn as sns

from itertools import product
from joblib import Parallel, delayed, dump, load
from brainspace.utils.parcellation import reduce_by_labels

import matplotlib
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from surfplot import Plot

import utilities
```

[3]:
```
def define_ci(data, alpha):
    import numpy as np

    nboot = len(data)
    data_sorted = np.sort(data)

    ci_high = data_sorted[int(np.floor((1-alpha/2)*nboot+.5))]
    ci_low = data_sorted[int(np.floor((alpha/2)*nboot+.5))]

    return ci_high, ci_low
```

[4]:
```
def get_lineplot_data(data, i, red_op='median', as_percentage=True):
    # Average across vertices
    if len(data.shape) > 3:
        if i == 1:
```

```
            data = np.nanstd(data, axis=2)/np.nanmean(data, axis=2)
        else:
            data = np.nanmean(data, axis=2)

    # Change to percentage of all subjects, all runs
    if as_percentage:
        for j in range(data.shape[-1]):
            data[:,:,j] = ((data[:,:,j]-data[-1,-1,j])/data[-1,-1,j])*100

    # Get mean of median value across bootstrap samples
    if red_op == 'mean':
        bootstrap_avg = np.nanmean(data, axis=2)
    elif red_op == 'median':
        bootstrap_avg = np.nanmedian(data, axis=2)

    return bootstrap_avg
```

[5]:
```
def add_contours(ax, data, levels=[0.001, 0.005, 0.01, 0.05], c='white',␣
↪fmt='%1.3f'):
    from scipy.interpolate import griddata

    grid_x, grid_y = np.meshgrid(
        np.linspace(0, data.shape[0]-1, 200, endpoint = True),
        np.linspace(0, data.shape[1]-1, 160, endpoint = True)
    )

    grid_z = griddata(
        np.argwhere(~np.isnan(data)), data.flatten(), (grid_x, grid_y),
        method='cubic'
    )

    cp = ax.contour(
        grid_y, grid_x, grid_z,
        levels,
        colors=c
    )
    ax.clabel(cp, inline=1, fontsize=8, fmt=fmt)

    return ax
```

[6]:
```
# Colorbar specs
cbar_unfolded_kws = dict(
    outer_labels_only=True,
    fontsize=12,
    pad=.02,
    n_ticks=2,
    decimals=1,
```

```
    shrink=.8,
    fraction=.1,
    draw_border=False
)
```

[7]:
```
cmap = matplotlib.cm.get_cmap('tab10')
```

[8]:
```
# Surface
unfolded  = '../resources/midthickness.L.unfolded.surf.gii'
coords    = nib.load(unfolded).
 ↪get_arrays_from_intent('NIFTI_INTENT_POINTSET')[0].data
nvertices = len(coords)
```

[9]:
```
# Atlas
atlas      = '../resources/BigBrain_ManualSubfieldsUnfolded_254x126.shape.gii'
subfields  = nib.load(atlas).darrays[0].data
labels     = ['Sub', 'CA1', 'CA2', 'CA3', 'CA4/DG']
nsubfields = len(np.unique(subfields))
```

[10]:
```
# Subjects
df = pd.read_csv('../config/participants.txt', dtype=str)
subjects = df.participant_id.to_list()
subjects = [ s for s in subjects if s != '09' ]
```

[11]:
```
# Hemispheres
hemis = ['Lflip','R']
```

[12]:
```
# Runs
runs = [str(r) for r in range(1,9)]
```

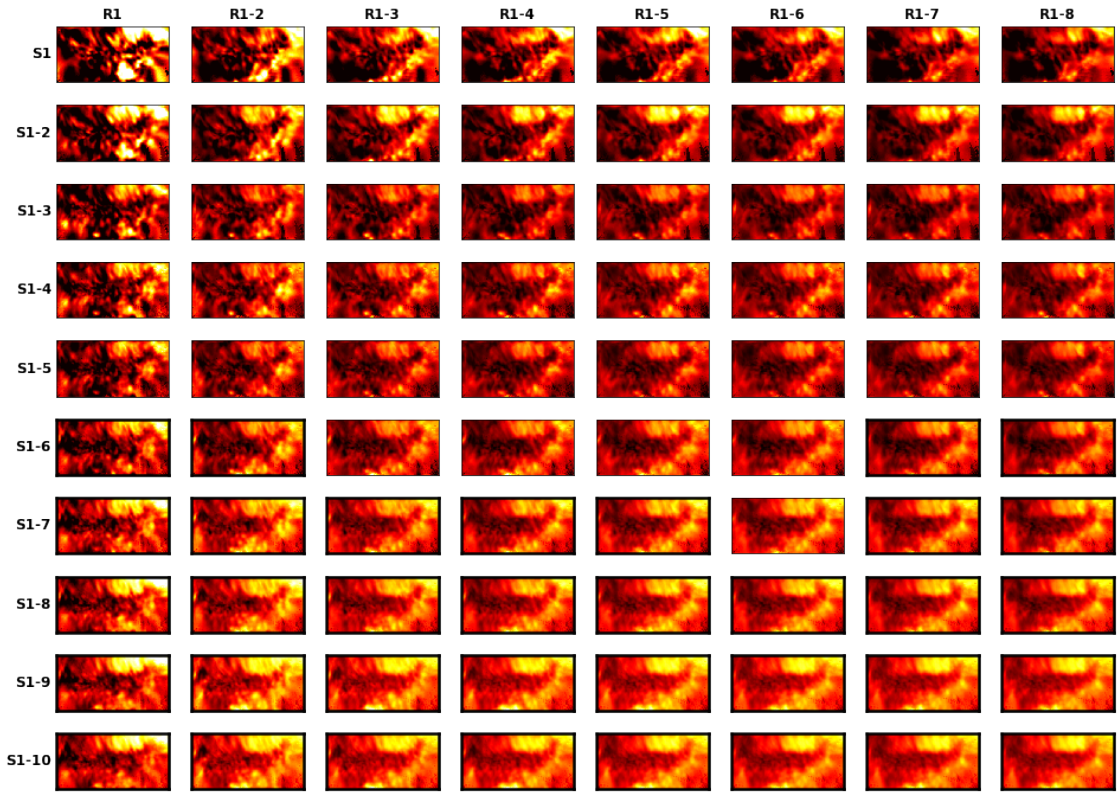
[13]:
```
# Maps
maps_dict = {
    1: ['CBF', 'hot', (20,60), 'Perfusion','(ml/100 g/min)', (10,75), (-10,10)],
    2: ['PWI', 'gray', (6,18), 'Perfusion-weigthed\nsignal','(a.u.)', (0,25),␣
 ↪(-5,5)],
    3: ['tSNR', 'hot', (0,10), 'tSNR', '(a.u.)', (0,8), (-2,2)]
}
```

[14]:
```
# Load CBF maps
fname = '../results/surface_maps/sub-{0}/run-{1}/sub-{0}_run-{1}_CBF_{2}.native.
 ↪shape.gii'

run_data = np.zeros((
    nvertices,
    len(subjects),
    len(runs),
    len(hemis)
```

```
))

for s, subject in enumerate(subjects):
    for r, run in enumerate(runs):
        for h, hemi in enumerate(hemis):
            data = nib.load(fname.format(subject, run, hemi)).darrays[0].data
            run_data[:,s,r,h] = data
```

[17]:
```
# Montage
fig, ax = plt.subplots(len(subjects), len(runs), figsize=(16,12))

for s in range(len(subjects)):
    for r in range(len(runs)):
        stats_data = np.nanmean(run_data[:,:s+1,:r+1,:], axis=2)[:,:,:,np.
 ↪newaxis]
        _, rm, _ = utilities.subfield_stats(stats_data, 0, 1, subjects[:s+1],␣
 ↪subfields)
        qval = rm['Q'].values[0]
        pval = rm['p-unc'].values[0]

        avg    = np.nanmean(run_data[:,:s+1,:r+1,:], axis=(3,2,1))
        avg_2d = avg.reshape((126,254), order='C')
        avg_2d = np.flipud(avg_2d)

        ax[s,r].imshow(avg_2d, interpolation='none', aspect='equal',␣
 ↪cmap='hot', vmin=20, vmax=60)
        ax[s,r].set_xticks([])
        ax[s,r].set_yticks([])

        if pval < .05:
            for spine in ['bottom','left','top','right']:
                ax[s,r].spines[spine].set_linewidth(2.5)

        if s == 0:
            ax[s,r].set_title(
                'R1{0}'.format('' if r == 0 else f'-{r+1}'),
                weight='bold', fontsize=12
            )

        if r == 0:
            ax[s,r].set_ylabel(
                'S1{0}'.format('' if s == 0 else f'-{s+1}'),
                weight='bold', fontsize=12, rotation=0,
                ha='right', va='center'
            )

plt.savefig(
```

```
    f'../visualization/unfolded/sub-group_CBF_stability_montage_cbf.png',
    bbox_inches='tight', dpi=600, transparent=True)
plt.show()
```




[30]:
```
# Reshape data
dims          = run_data.shape
number_maps   = dims[1]*dims[2]*dims[3]
run_data_long = np.zeros((nvertices, number_maps, 2))

# Make data long using hemisphere > run > subject
i = 0
for s in range(len(subjects)):
    for r in range(len(runs)):
        for h in range(2):
            run_data_long[:,i,0] = run_data[:,s,r,h]
            i += 1

# Make data long using hemisphere > subject > run
i = 0
for r in range(len(runs)):
    for s in range(len(subjects)):
        for h in range(2):
```

```
            run_data_long[:,i,1] = run_data[:,s,r,h]
            i += 1
```

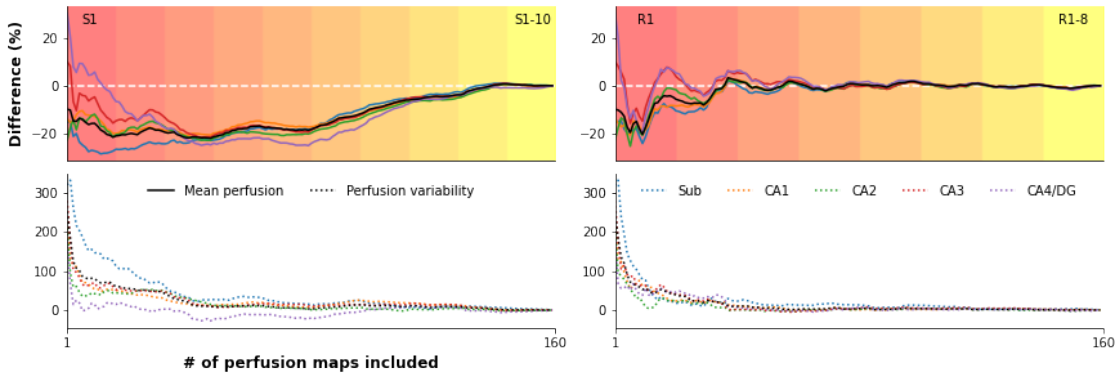
[31]:
```
# Calculate perfusion averages and variability as function of included maps
cbf_avg_evolution = np.zeros((2,6,run_data_long.shape[1]))
cbf_std_evolution = np.zeros((2,6,run_data_long.shape[1]))

for i in range(2):
    for j in range(run_data_long.shape[1]):
        data = np.nanmean(run_data_long[:,:j+1,i], axis=1)

        cbf_avg_evolution[i,0,j] = np.nanmean(data)
        cbf_std_evolution[i,0,j] = np.nanstd(data)/cbf_avg_evolution[i,0,j]

        cbf_avg_evolution[i,1:,j] = reduce_by_labels(data, subfields)
        cbf_std_evolution[i,1:,j] = [ np.nanstd(data[subfields==s]) for s in np.
 ↪unique(subfields) ]/cbf_avg_evolution[i,1:,j]

cbf_avg_evolution = (cbf_avg_evolution-cbf_avg_evolution[:,:,-1][:,:,None])/
 ↪cbf_avg_evolution[:,:,-1][:,:,None]*100
cbf_std_evolution = (cbf_std_evolution-cbf_std_evolution[:,:,-1][:,:,None])/
 ↪cbf_std_evolution[:,:,-1][:,:,None]*100
```

[22]:
```
run_ind     = np.array([ np.repeat(x, 22) for x in range(len(runs)) ]).
 ↪flatten()[np.newaxis,:]
subject_ind = np.array([ np.repeat(x, 16) for x in range(len(subjects)) ]).
 ↪flatten()[np.newaxis,:]
```

[35]:
```
# Plot
fig = plt.figure(constrained_layout=True, figsize=(12,4))
gs = GridSpec(2, 2, figure=fig)

xdata = np.arange(0,run_data_long.shape[1])

ax1, ax3 = fig.add_subplot(gs[0,0]), fig.add_subplot(gs[1,0])
ax2, ax4 = fig.add_subplot(gs[0,1], sharey=ax1), fig.add_subplot(gs[1,1],␣
 ↪sharey=ax3)

# Add subfield-specific plots
for i in range(1,6):
    sns.lineplot(x=xdata, y=cbf_avg_evolution[0,i,:], ax=ax1, color=cmap(i-1))
    sns.lineplot(x=xdata, y=cbf_avg_evolution[1,i,:], ax=ax2, color=cmap(i-1))

    sns.lineplot(x=xdata, y=cbf_std_evolution[0,i,:], ax=ax3, color=cmap(i-1),␣
 ↪linestyle=':')
```

```
    sns.lineplot(x=xdata, y=cbf_std_evolution[1,i,:], ax=ax4, color=cmap(i-1),␣
 ↪linestyle=':', label=labels[i-1])

# Mean across all vertices, left y-axis
sns.lineplot(x=xdata, y=cbf_avg_evolution[0,0,:], ax=ax1, color='black',␣
 ↪label='Mean perfusion')
sns.lineplot(x=xdata, y=cbf_avg_evolution[1,0,:], ax=ax2, color='black')

# CoV across all vertices, right y-axis
sns.lineplot(x=xdata, y=cbf_std_evolution[0,0,:], ax=ax3, color='black',␣
 ↪linestyle=':', label='Perfusion variability')
sns.lineplot(x=xdata, y=cbf_std_evolution[1,0,:], ax=ax4, color='black',␣
 ↪linestyle=':')

# Center axes at 0 and some other things
for ax in fig.get_axes():
    ax.axhline(y=0, color='white', linestyle='--', zorder=-5)
    ax.set_xlim(0,number_maps)

# Add background colors
ax1.imshow(
    subject_ind, aspect='auto', origin='lower', alpha=.5,
    extent=(0,number_maps,ax1.get_ylim()[0],ax1.get_ylim()[1]),
    cmap='autumn', zorder=-10
)
ax2.imshow(
    run_ind, aspect='auto', origin='lower', alpha=.5,
    extent=(0,number_maps,ax2.get_ylim()[0],ax2.get_ylim()[1]),
    cmap='autumn', zorder=-10
)

# Add text annotations
ax1.annotate('S1', ((1/11)/2,.95), xycoords='axes fraction', va='top',␣
 ↪ha='center')
ax1.annotate('S1-10', (1-(1/11)/2,.95), xycoords='axes fraction', va='top',␣
 ↪ha='center')

ax2.annotate('R1', ((1/8)/2,.95), xycoords='axes fraction', va='top',␣
 ↪ha='center')
ax2.annotate('R1-8', (1-(1/8)/2,.95), xycoords='axes fraction', va='top',␣
 ↪ha='center')

# Add/remove labels
ax1.set_ylabel(f'Difference (%)', fontsize=12, fontweight='bold')
ax1.set_xticks([])
ax2.set_xticks([])
```

```
for ax in [ax3, ax4]:
    ax.set_xticks([0,number_maps])
    ax.set_xticklabels([1,number_maps])
ax3.set_xlabel(f'# of perfusion maps included', fontsize=12, fontweight='bold')

# Legends
l1, ll1 = ax1.get_legend_handles_labels()
l2, ll2 = ax3.get_legend_handles_labels()
ax3.legend(l1 + l2, ll1 + ll2, frameon=False, ncol=2, loc='upper center')
ax4.legend(frameon=False, ncol=5, loc='upper center')
ax1.get_legend().remove()

sns.despine()
fig.savefig(
    f'../visualization/unfolded/sub-group_CBF_stability_averages.png',
    dpi=600, bbox_inches='tight', transparent=True
)
plt.show()
```




[14]:
```
# Load tag-control difference maps
fname = '../results/surface_maps/sub-{0}/full_fit/sub-{0}_DIFF_{1}.native.shape.
  ↪gii'
nvolumes = len(nib.load(fname.format(subjects[0], hemis[0])).darrays)

full_fit_data = np.zeros((
    nvertices,
    len(subjects),
    len(hemis),
    nvolumes
))

for s, subject in enumerate(subjects):
    for h, hemi in enumerate(hemis):
```

```
        full_fit = nib.load(fname.format(subject, hemi))
        full_fit = np.array([ full_fit.darrays[x].data for x in␣
 ↪range(0,len(full_fit.darrays)) ]).T
        full_fit_data[:,s,h,:] = full_fit
```

[15]:
```
# Separation of individual runs
nvolumes_per_run = nvolumes/len(runs)
run_index        = np.zeros((nvolumes))

for i in range(0,len(runs)):
    start = int(i*nvolumes_per_run)
    end   = int(start+nvolumes_per_run)
    run_index[start:end] = i
```

[22]:
```
# Load bootstrapping results (run separately using run_bootstrapping.job)
nboot = 1000
data = h5py.
 ↪File(f'bootstrap_matrix_full-fit_Nboot-{nboot}_batch-avg_new_vertices.hdf5',␣
 ↪'r')
```

[ ]:
```
# Plot
fig = plt.figure(constrained_layout=True, figsize=(10,5))
gs  = GridSpec(2, 6, width_ratios=[1.75,1,1,1.75,1,1], hspace=-.2, figure=fig) #

prop_dict = {
    0: [-1,1, -1,1, 'A Mean perfusion'], 1: [-100,100, -10,100, 'B Perfusion␣
 ↪variability'],
    2: [-3,3, -2,3, 'C tSNR'], 3: [-100,100, -100,10, 'D Between subfield␣
 ↪effect']
}

# How to reduce across bootstrap samples, mean or median
red_op = 'median'

for i, (r, c) in enumerate(zip([0,0,1,1],[0,3,0,3])):
    if i == 0 or i == 2:
        perc = get_lineplot_data(
            data['bootstrap_matrix'][:,:,:,i,:], i, red_op=red_op,␣
 ↪as_percentage=True
        )
    if i == 1:
        perc = get_lineplot_data(
            data['bootstrap_matrix'][:,:,:,0,:], 1, red_op=red_op,␣
 ↪as_percentage=True
        )
    elif i == 3:
```

```
        perc = get_lineplot_data(
            data['bootstrap_matrix'][:,:,0,i,:], i, red_op=red_op,␣
↪as_percentage=True
        )

    vmin, vmax = prop_dict[i][0], prop_dict[i][1]

    # Heatmaps
    ax1 = fig.add_subplot(gs[r,c])
    img = ax1.imshow(perc, origin='lower', cmap='seismic', aspect='auto',␣
↪vmin=vmin, vmax=vmax)
    ax1.yaxis.set_ticks(np.arange(0,10))
    ax1.yaxis.set_ticklabels(np.arange(1,11))
    ax1.xaxis.set_ticks(np.arange(0,8) if i > 1 else [])
    ax1.xaxis.set_ticklabels(np.arange(1,9) if i > 1 else [])

    # Overlay p-value contour plot
    if i == 3:
        pvals = np.nanmedian(data['bootstrap_matrix'][:,:,1,3,:], axis=2)
        ax1 = add_contours(ax1, pvals, c='white')
    else:
        ax1 = add_contours(ax1, perc, levels=0, c='black', fmt='%1.0f %%')

    # Colorbar
    cb = plt.colorbar(img, ax=ax1, orientation='vertical', aspect=16 if i == 0␣
↪else 30 if i in [1,3] else 18.5)
    cb.set_ticks([prop_dict[i][2], 0, prop_dict[i][3]])
    cb.set_ticklabels([prop_dict[i][2], 0, prop_dict[i][3]])
    cb.ax.set_ylim(prop_dict[i][2], prop_dict[i][3])
    cb.ax.set_ylabel('Difference (%)' if i == 0 else ' ', fontweight='bold',␣
↪fontsize=8)
    cb.ax.yaxis.set_label_position('left')
    cb.outline.set_linewidth(0)

    # Line plots
    ax2 = fig.add_subplot(gs[r,c+1])
    ax2.plot(np.arange(1,perc.shape[1]+1), perc[-1,:], c='black', label='Across␣
↪runs', zorder=10)
    ax2.set_xlim(1,perc.shape[1])
    ax2.set_ylim([prop_dict[i][2], prop_dict[i][3]])
    ax2.set_yticks([prop_dict[i][2], prop_dict[i][3]])
    ax2.set_yticklabels([])
    ax2.xaxis.set_ticks([1,8] if r == 1 else [])
    ax2.xaxis.set_ticklabels(['R1','R1-8'] if r == 1 else [])

    ax3 = fig.add_subplot(gs[r,c+2], sharey=ax2)
    ax3.plot(np.arange(1,perc.shape[0]+1), perc[:,-1], c='black', zorder=10)
```

```
    ax3.set_xlim(2, perc.shape[0])
    ax3.xaxis.set_ticks([1,10] if r == 1 else [])
    ax3.xaxis.set_ticklabels(['S1','S1-10'] if r == 1 else [])

    ax2.axhline(y=0, color='lightgray', linestyle='--')
    ax3.axhline(y=0, color='lightgray', linestyle='--')

    if i == 0:
        ax1.set_ylabel('# subjects', weight='bold')
    elif i == 2:
        ax1.set_xlabel('# runs', weight='bold')
        ax2.set_xlabel('# runs', weight='bold')
        ax3.set_xlabel('# subjects', weight='bold')

    sns.despine(ax=ax1, offset=5)
    sns.despine(ax=ax2, offset=5)
    sns.despine(ax=ax3, offset=5)

    # Per subfield
    for j in np.arange(1,6):
        if i == 0 or i == 2:
            perc = get_lineplot_data(
                data['bootstrap_matrix'][:,:,np.argwhere(subfields==j)[:,0],i,:
 ↪],
                i, red_op=red_op, as_percentage=True
            )
        elif i == 1:
            perc = get_lineplot_data(
                data['bootstrap_matrix'][:,:,np.argwhere(subfields==j)[:,0],0,:
 ↪],
                1, red_op=red_op, as_percentage=True
            )
        if i != 3:
            ax2.plot(np.arange(1,perc.shape[1]+1), perc[-1,:],␣
 ↪label=labels[j-1], zorder=-10)
            ax3.plot(np.arange(1,perc.shape[0]+1), perc[:,-1],␣
 ↪label=labels[j-1], zorder=-10)

plt.savefig(f'../visualization/unfolded/sub-group_stability_Nboot-{nboot}.png',␣
 ↪bbox_inches='tight', dpi=600)
plt.show()
```