vasculature

June 15, 2023

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

[2]:
```
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
import numpy as np
import nibabel as nib
import pandas as pd

from surfplot import Plot
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

from brainspace.utils.parcellation import reduce_by_labels

import utilities

import pingouin as pg
from scipy.stats import linregress, spearmanr
from statannotations.Annotator import Annotator
```

[4]:
```
def reduce_data(data, subfields):
    data_reshape = data.reshape(
        data.shape[0],data.shape[1]*data.shape[2]
    )

    subfields_mean = reduce_by_labels(data_reshape,subfields,axis=1)
    subfields_mean_reshape = subfields_mean.reshape(
        subfields_mean.shape[0],data.shape[1],data.shape[2]
    )

    return data_reshape, subfields_mean_reshape
```

[5]:
```
def reshape_to_2d(data, shape=(126,254), order='C'):
    return np.flipud(data.reshape(shape, order=order))
```

[6]:
```
# Color 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')
magma = matplotlib.cm.get_cmap('magma')
```

[23]:
```
save_fig = True
```

Subjects

[9]:
```
df = pd.read_csv('../config/participants.txt', dtype=str)
subjects = df.participant_id.to_list()
subjects = [ s for s in subjects if s != '09' ]
```

Hippocampal

[10]:
```
# Surface
unfolded  = '../resources/midthickness.L.unfolded.surf.gii'
coords    = nib.load(unfolded).
 ↪get_arrays_from_intent('NIFTI_INTENT_POINTSET')[0].data
nvertices = len(coords)
```

[11]:
```
# 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))
```

[12]:
```
subfields_2d  = subfields.reshape((126,254), order='C')
```

[13]:
```
# Hemispheres
hemis = ['Lflip','R']
```

[14]:
```
# Runs
runs = [str(r) for r in range(1,9)]
```

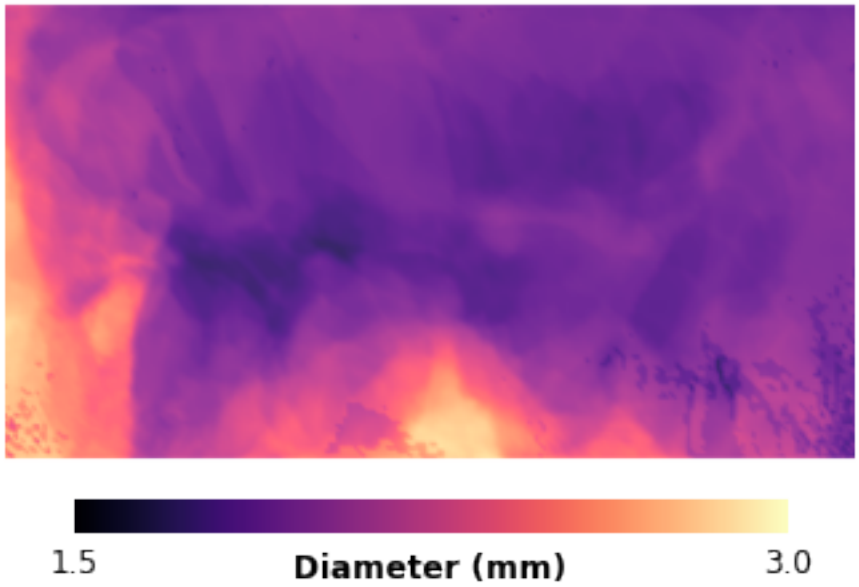
[15]:
```
# Hemispheres
hemis = ['Lflip','R']
```

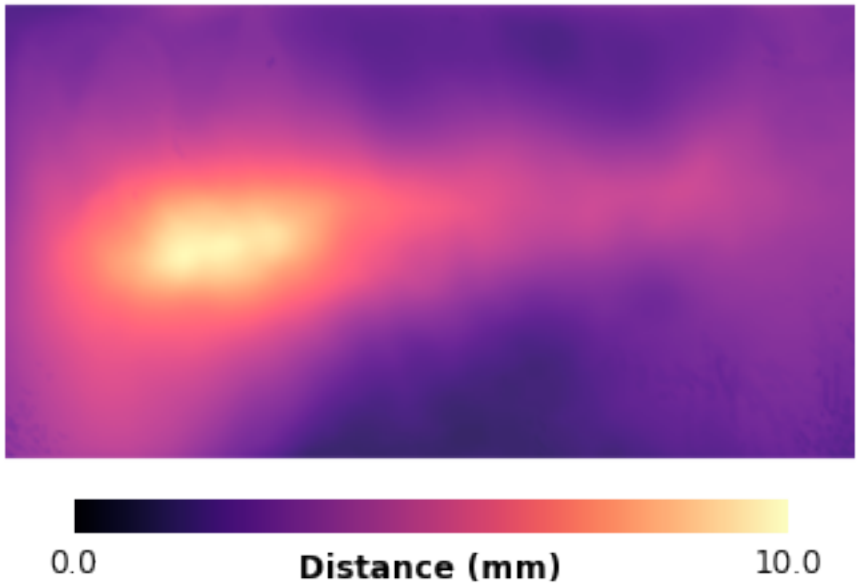
[16]:
```
# Maps
maps_dict = {
    1: ['vesseldiameter', 'magma', (1.5,3), 'Diameter','(mm)', (1.25,3),␣
 ↪(-2,2)],
    2: ['vesseldistance', 'magma', (0,10), 'Distance','(mm)', (0,10), (-2,2)]
}
```

[30]:
```
# 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

avg_data = np.nanmean(run_data, axis=(3,2,1))
cov_data = np.nanstd(np.nanmean(run_data, axis=3).reshape(32004,-1), axis=1)/np.
 ↪nanmean(run_data, axis=(3,2,1))
```

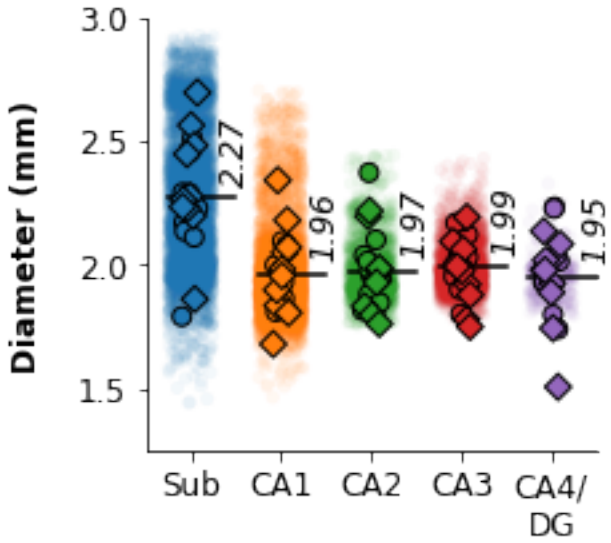
[32]:
```
# Load input, per subject
vessel_data = np.zeros((
    nvertices,
    len(subjects),
    len(hemis),
    len(maps_dict)
))

for s, subject in enumerate(subjects):
    for h, hemi in enumerate(hemis):
        for i, idx in enumerate(maps_dict.keys()):
            vessel_data[:,s,h,i] = nib.load(
                '../results/surface_maps/sub-{0}/sub-{0}_{1}_{2}.native.shape.
 ↪gii'.format(
                    subject, maps_dict[idx][0], hemi)
            ).darrays[0].data

# Average across hemispheres, subjects
vessel_avg_data = np.nanmean(vessel_data, axis=(2,1))
```

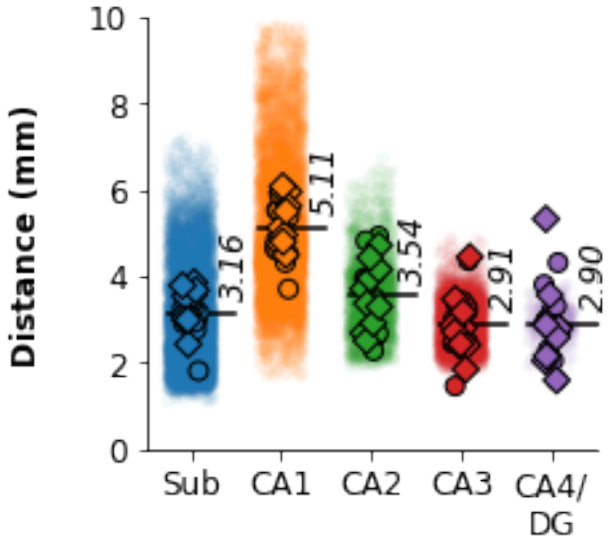
[25]:
```
# Iterate through maps and plot unfolded
for i, idx in enumerate(maps_dict.keys()):
    p = Plot(unfolded, layout='row', views=['dorsal'], zoom=2, size=(500, 300))
    p.add_layer(vessel_avg_data[:,i], color_range=maps_dict[idx][2],␣
 ↪cmap=maps_dict[idx][1])

    fig = p.build(cbar_kws=cbar_unfolded_kws)
    fig.axes[1].set_xlabel(f'{maps_dict[idx][3]} (mm)', labelpad=-11,␣
 ↪fontsize=12, fontweight='bold')
    if save_fig:
        fig.savefig(
            '../visualization/unfolded/sub-group_{}_unfolded.png'.format(
                maps_dict[idx][0]), dpi=600, bbox_inches='tight',␣
 ↪transparent=True
        )
    fig.show()
```





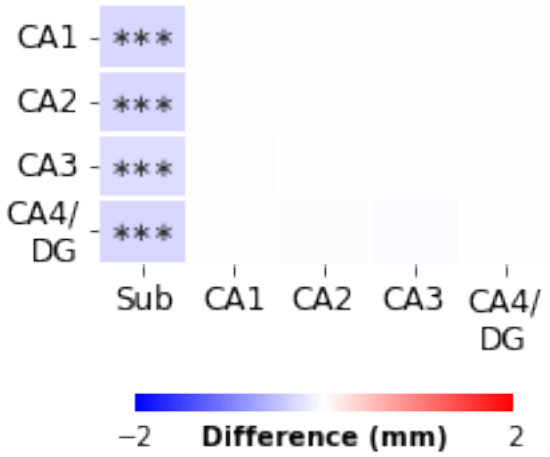
[26]:
```
for i, idx in enumerate(maps_dict.keys()):
    filename = '../visualization/subfield-averages/
 ↪sub-group_{}_subfields_averages.png'.format(maps_dict[idx][0])
    utilities.plot_subfield_data(
        vessel_data[:,:,:,i], subfields, maps_dict[idx], filename, scale=.7,␣
 ↪stats=False, save_fig=save_fig
    )
```

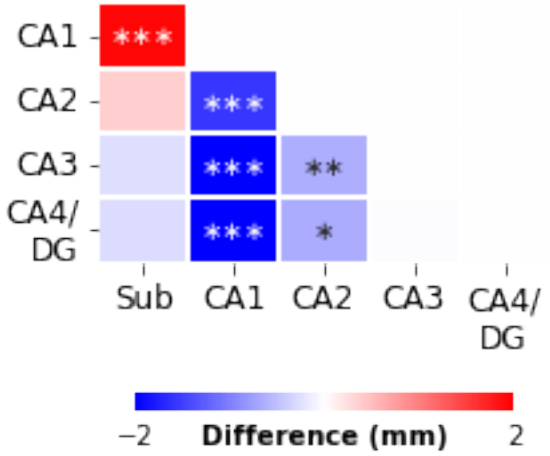




[27]:
```
for i, idx in enumerate(maps_dict.keys()):
    df_stats, rm, pw = utilities.subfield_stats(vessel_data, i, 1, subjects,␣
 ↪subfields)
    print(rm)

    # Heatmap
    filename = '../visualization/subfield-averages/
 ↪sub-group_{}_subfields_stat_differences.png'.format(maps_dict[idx][0])
    utilities.plot_subfield_pairs(df_stats, rm, pw, i, maps_dict[idx],␣
 ↪filename, scale=.7, save_fig=save_fig)
```

```
            Source      W  ddof1      Q     p-unc
Friedman  Subfield  0.536      4  21.44  0.000259
            Source     W  ddof1     Q      p-unc
Friedman  Subfield  0.63      4  25.2  0.000046
```





0.0.1 Perfusion vs vasculature

[23]:
```
# Correlation between maps
for i,ii in enumerate(['Diameter','Distance']):
    r, p = spearmanr(avg_data, vessel_avg_data[:,i])
    pperm  = utilities.correlation_between_maps(
        avg_data, vessel_avg_data[:,i], 5000
    )

    print(f'Mean perfusion vs. {ii}: {r}, {p}, {pperm}')
```

```
Mean perfusion vs. Diameter: -0.055842621053231956, 1.563449211957007e-23,
0.8086
Mean perfusion vs. Distance: -0.5919560244715735, 0.0, 0.0152
```

[24]:
```
for i,ii in enumerate(['Diameter','Distance']):
    r, p = spearmanr(cov_data, vessel_avg_data[:,i])
    pperm  = utilities.correlation_between_maps(
        cov_data, vessel_avg_data[:,i], 5000
    )

    print(f'Perfusion variability vs. {ii}: {r}, {p}, {pperm}')
```

```
Perfusion variability vs. Diameter: 0.38251769079472353, 0.0, 0.0053
Perfusion variability vs. Distance: -0.08030998290481516, 5.990874978729929e-47,
0.3373
```

[33]:
```
## Downsample maps to reduce number of points
downsample_factor = .5

# Perfusion
y1_data = utilities.downsample_data(
    reshape_to_2d(avg_data), downsample_factor, True, False
)

y2_data = utilities.downsample_data(
    reshape_to_2d(cov_data), downsample_factor, True, False
)

# Vasculature
x_data = utilities.downsample_data(
    reshape_to_2d(vessel_avg_data[:,1]), downsample_factor, True, False
)

c_data = utilities.downsample_data(
    reshape_to_2d(vessel_avg_data[:,0]), downsample_factor, True, False
)
```

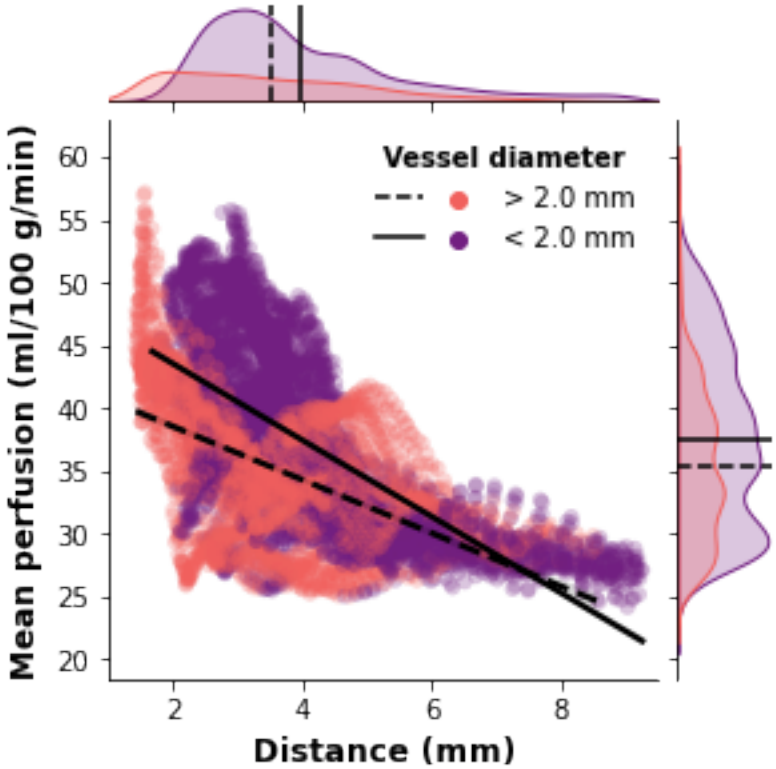
[42]:
```
# Categorize diameter data
c_data_cut  = pd.cut(
    c_data, bins=[0, 2, np.max(c_data)],
    precision=1, labels=['< 2.0 mm', '> 2.0 mm']
)
```

[43]:
```
# Combine data
stacked = np.vstack((
    x_data, y1_data, y2_data, c_data, c_data_cut
))
stacked = np.transpose(stacked)

df = pd.DataFrame(
    data = stacked,
```

```
    columns = [
        "Distance", "Mean perfusion", "Perfusion variability",
        "Diameter", "Diameter category"
    ]
)

df['Distance']              = df['Distance'].astype(float)
df['Diameter']              = df['Diameter'].astype(float)
df['Mean perfusion']        = df['Mean perfusion'].astype(float)
df['Perfusion variability'] = df['Perfusion variability'].astype(float)

df_grouped = df.groupby(['Diameter category']).mean().reset_index()
```

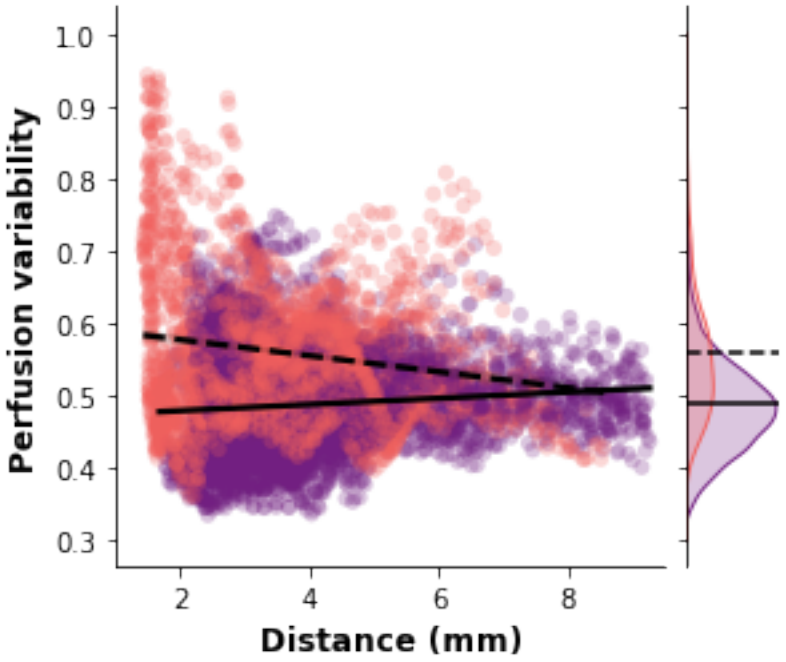
[50]:
```
ydata = "Mean perfusion"
plt.rcParams['legend.frameon'] = False

# Plot scatterplot and marginal distribution plots
g = sns.jointplot(
    data=df, x="Distance", y=ydata, hue="Diameter category",
    marginal_ticks=False, linewidth=0, alpha=0.25, palette='magma_r'
)

# Add mean lines
for i in range(2):
    g.ax_marg_x.axvline(x=df_grouped['Distance'][i], color='black',␣
 ↪linestyle=['-','--'][i])
    g.ax_marg_y.axhline(y=df_grouped[ydata][i], color='black',␣
 ↪linestyle=['-','--'][i])

g.fig.set_size_inches((4, 4))

# Add fit lines
gca = plt.sca(g.ax_joint)
for d, diameter in enumerate(['< 2.0 mm','> 2.0 mm']):
    df_plot = df[df['Diameter category']==diameter]
    sns.regplot(
        data=df_plot, x="Distance", y=ydata,
        scatter=False, color='black', line_kws={'linestyle':['-','--'][d]}
    )

g.set_axis_labels('Distance (mm)', f'{ydata} (ml/100 g/min)', fontsize=12,␣
 ↪weight='bold')
g.ax_joint.set_xlim(1,9.5)

# Add legend
handles, labels = g.ax_joint.get_legend_handles_labels()
line1 = matplotlib.lines.Line2D([0], [0], color='k', linestyle='--')
```

```
line2 = matplotlib.lines.Line2D([0], [0], color='k', linestyle='-')

l = g.ax_joint.legend(
    [(line1, handles[0]), (line2, handles[1])], [labels[0], labels[1]],
    handler_map={tuple: matplotlib.legend_handler.HandlerTuple(ndivide=None)},
    handlelength=4
)

l.set_title('Vessel diameter')
plt.setp(l.get_texts(), fontsize='10')
plt.setp(l.get_title(), fontsize='10', fontweight='bold')

filename = f'../visualization/unfolded/sub-group_CBF_mean_vs_vasculature.png'
if save_fig:
    g.savefig(filename, dpi=600, bbox_inches='tight', transparent=True)
plt.show()
```



[51]:
```
ydata = "Perfusion variability"

# Plot scatterplot and marginal distribution plots
g = sns.jointplot(
    data=df, x="Distance", y=ydata, hue="Diameter category",
```

```
    marginal_ticks=False, linewidth=0, alpha=0.25, palette='magma_r'
)

g.ax_marg_x.remove()
# Add mean lines
for i in range(2):
    # g.ax_marg_x.axvline(x=df_grouped['Distance'][i], color='black',␣
 ↪linestyle=['-','--'][i])
    g.ax_marg_y.axhline(y=df_grouped[ydata][i], color='black',␣
 ↪linestyle=['-','--'][i])

g.ax_joint.set_xlim(1,9.5)
g.fig.set_size_inches((4, 4))

# Add fit lines
gca = plt.sca(g.ax_joint)
for d, diameter in enumerate(['< 2.0 mm','> 2.0 mm']):
    df_plot = df[df['Diameter category']==diameter]
    sns.regplot(
        data=df_plot, x="Distance", y=ydata,
        scatter=False, color='black', line_kws={'linestyle':['-','--'][d]}
    )

g.set_axis_labels('Distance (mm)', f'{ydata}', fontsize=12, weight='bold')
g.ax_joint.legend_.remove()

filename = f'../visualization/unfolded/sub-group_CBF-cov_vs_vasculature.png'
g.savefig(filename, dpi=600, bbox_inches='tight', transparent=True)
plt.show()
```


Perfusion variability
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2
4
6
8
Distance (mm)