crosscorrelation

June 15, 2023

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

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

[3]:
```
import os, time, h5py
import numpy as np
import pandas as pd
import nibabel as nib
import seaborn as sns
import pingouin as pg
import scipy.io as spio

from tqdm.notebook import tqdm
from itertools import product
from joblib import Parallel, delayed, dump, load
from IPython.display import clear_output

from brainspace.utils.parcellation import reduce_by_labels

from scipy.stats import zscore, linregress, pearsonr
from sklearn.preprocessing import MinMaxScaler
from pyls import pls_regression
from pingouin import multicomp

import matplotlib
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from mpl_toolkits.axes_grid1 import make_axes_locatable
from surfplot import Plot

import utilities
```

[4]:
```
from skimage.transform import resize
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
```

```
from sklearn.impute import SimpleImputer
from sklearn.cluster import AgglomerativeClustering, KMeans
```

[5]:
```
def load_shape_gii(metric):
    return nib.load(
        f'../results/surface_maps/group/sub-group_{metric}_LR_avg.native.shape.
 ↪gii'
    ).darrays[0].data
```

[6]:
```
def permute_corrmatrix(corrmatrix, df, coords, permsamples, p):
    import numpy as np
    import pandas as pd

    for i in range(len(coords)):
        x, y = coords[i][0], coords[i][1]

        corrmatrix[x,y,p] = np.corrcoef(
            np.hstack((
                df.iloc[permsamples, [x+1]].values,
                df.iloc[:, [y+1]].values
            )).T
        )[0,1]

    return corrmatrix
```

[7]:
```
# 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
)
```

[8]:
```
cmap = matplotlib.cm.get_cmap('tab10')
```

[9]:
```
# Surface
unfolded  = '../resources/midthickness.L.unfolded.surf.gii'
coords    = nib.load(unfolded).
 ↪get_arrays_from_intent('NIFTI_INTENT_POINTSET')[0].data
faces     = nib.load(unfolded).
 ↪get_arrays_from_intent('NIFTI_INTENT_TRIANGLE')[0].data
nvertices = len(coords)
nfaces    = len(faces)
```

[10]:
```
# Atlas
atlas        = '../resources/BigBrain_ManualSubfieldsUnfolded_254x126.shape.gii'
subfields    = nib.load(atlas).darrays[0].data
subfields_2d = subfields.reshape((126,254), order='C')
labels       = ['Sub', 'CA1', 'CA2', 'CA3', 'CA4/DG']
nsubfields   = len(np.unique(subfields))
```

[11]:
```
# 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' ]
```

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

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

[15]:
```
# Category to do analysis
category = 'mri'
save_fig = True
```

[60]:
```
if category == 'mri':
    # MRI quality maps
    maps = [
        'tSNR', 'B1map', 'T1', 'Shiftmap',
        'pveCSF', 'pveGM', 'pveWM',
    ]

    maps_perm = maps + ['CBF']
    columns   = ['Subfield', 'Subject'] + maps + ['Perfusion']
    tick_labels = [
        'tSNR', 'B$_{1}^{+}$', 'T$_1$', 'Distortion',
        'PVE$_{CSF}$', 'PVE$_{GM}$', 'PVE$_{WM}$',
        'Perfusion'
    ]

elif category == 'morph':
    # Tissue maps
    maps = [
        'thickness', 'gyrification', 'curvature',
        'myelin', 'vesseldiameter', 'vesseldistance'
    ]

    maps_perm = maps + ['CBF']
    columns = ['Subfield', 'Subject'] + maps + ['Perfusion']
    tick_labels = [
```

```
        'Thickness', 'Gyrification', 'Curvature', 'Myelination',
        'Vessel diameter', 'Vessel distance', 'Perfusion'
    ]
```

0.0.1 Load data

[17]:
```
subfields_col = []
subjects_col = []

for subfield in range(nsubfields):
    for subject in range(len(subjects)):
        subfields_col.append(subfield)
        subjects_col.append(subject)
```

[18]:
```
# Load CBF maps
fname = '../results/surface_maps/sub-{0}/sub-{0}_{1}_{2}.native.shape.gii'

perf_data = np.zeros((
    nsubfields,
    len(subjects),
    len(hemis)
))

for s, subject in enumerate(subjects):
    for h, hemi in enumerate(hemis):
        data = nib.load(fname.format(subject, 'CBF', hemi)).darrays[0].data
        perf_data[:,s,h] = reduce_by_labels(data, subfields)

# Average across hemispheres and runs
y_data = np.nanmean(perf_data, axis=2)
Y = y_data.flatten()[:,np.newaxis]
```

[22]:
```
# Load other maps
fname = '../results/surface_maps/sub-{0}/sub-{0}_{1}_{2}.native.shape.gii'

other_data = np.zeros((
    nsubfields,
    len(subjects),
    len(hemis),
    len(maps)
))

for m, metric in enumerate(maps):
    for s, subject in enumerate(subjects):
        for h, hemi in enumerate(hemis):
            # data = nib.load(fname.format(subject, metric, hemi)).darrays[0].
 ↪data
```

```
            data = nib.load(
                    fname.format(
                        subject,
                        f"{'tpl-hires_' if metric == 'Shiftmap' else␣
 ↪''}{metric}",
                        hemi
                    )
            ).darrays[0].data

            other_data[:,s,h,m] = reduce_by_labels(data, subfields)

# Average across hemispheres and runs
X_data = np.nanmean(other_data, axis=2)
X = np.zeros((len(Y), X_data.shape[2]))
for m in range(X_data.shape[2]):
    X[:,m] = X_data[:,:,m].flatten(order='C')
```

[23]:
```
# Combine into DataFrame
df = pd.DataFrame(
    data=np.hstack((
        np.array(subfields_col)[:,np.newaxis],
        np.array(subjects_col)[:,np.newaxis],
        X, Y
    )),
    columns = columns
)
```

Normalize

[44]:
```
# Z-score within subjects
Y_scaled = Y.copy()
for s in range(len(subjects)):
    mask = np.isin(subjects_col, s)
    Y_scaled[mask,0] = zscore(Y[mask,0])
```

[45]:
```
# Z-score within subjects
X_scaled = X.copy()
for m in range(X_scaled.shape[1]):
    for s in range(len(subjects)):
        mask = np.isin(subjects_col, s)
        X_scaled[mask,m] = zscore(X[mask,m])
```

[46]:
```
# Combine into DataFrame
df_scaled = pd.DataFrame(
    data=np.hstack((
        np.array(subfields_col)[:,np.newaxis],
        np.array(subjects_col)[:,np.newaxis],
```

```
        X_scaled, Y_scaled
    )),
    columns = columns
)
```

0.0.2 Correlation analyses

[47]:
```
df_scaled['Subfield'].replace(
    to_replace={
        0: 'Sub', 1: 'CA1', 2: 'CA2',
        3: 'CA3', 4: 'CA4/DG'
    }, inplace=True
)

df['Subfield'].replace(
    to_replace={
        0: 'Sub', 1: 'CA1', 2: 'CA2',
        3: 'CA3', 4: 'CA4/DG'
    }, inplace=True
)
```

[48]:
```
df_scaled = df_scaled.astype(
    {'Subfield':'category',
     'Subject':'category'}
)

df = df.astype(
    {'Subfield':'category',
     'Subject':'category'}
)
```

Subfield-wise correlation

Pearson's correlation

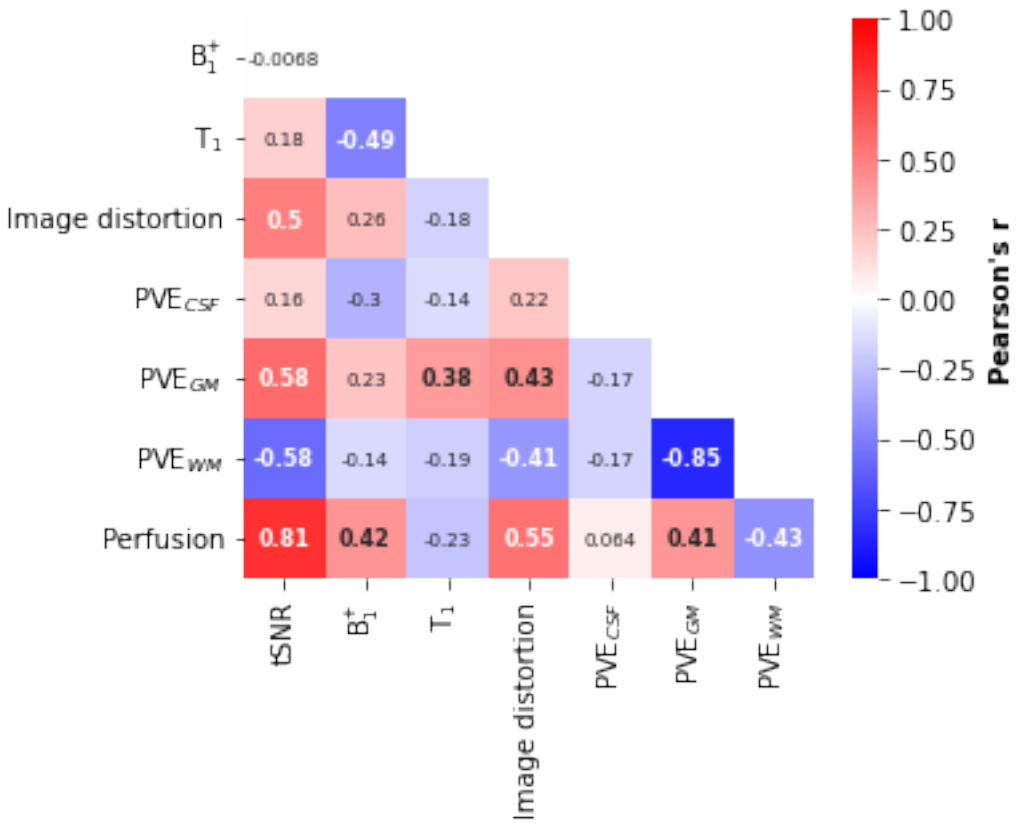
[ ]:
```
df_corr = df_scaled.corr()
rho_pvals = df_scaled.corr(
    method=lambda x, y: pearsonr(x, y)[1]
) - np.eye(*df_corr.shape)
```

[ ]:
```
lt_mask = np.tril(
    np.ones(df_corr.shape),
    k=-1
).astype(bool)
```

[ ]:
```
df_corr_lt = df_corr.where(lt_mask)
```

[ ]:
```
# Multiple comparison correction for lower triangle
rho_pvals_lt       = rho_pvals.values[lt_mask].flatten()
reject, pvals_corr = multicomp(rho_pvals_lt, method='fdr_bh')
rho_pvals_mcc      = np.ones_like(rho_pvals)
rho_pvals_mcc[lt_mask] = pvals_corr
```

[ ]:
```
fig, ax = plt.subplots(1,1, figsize=(5.5,4.5))

g = sns.heatmap(
    df_corr_lt.iloc[1:,:-1],
    mask=np.where(rho_pvals_mcc<=.05, True, False)[1:,:-1],
    cmap="bwr", vmin=-1, vmax=1,
    annot=True, fmt='.2g', annot_kws={'size': 7},
    xticklabels=tick_labels[:-1], yticklabels=tick_labels[1:],
    ax=ax
)

g.collections[0].colorbar.set_label("Pearson's r", fontweight='bold')

sns.heatmap(
    df_corr_lt.iloc[1:,:-1],
    mask = np.where(rho_pvals_mcc>=.05, True, False)[1:,:-1],
    cmap="bwr", vmin=-1, vmax=1, cbar=False,
    annot=True, fmt='.2g', annot_kws={'size': 8, 'weight': 'bold'},
    xticklabels=tick_labels[:-1], yticklabels=tick_labels[1:],
    ax=ax
)

fig.tight_layout()
if save_fig:
    filename = f'../visualization/unfolded/
 ↪sub-group_{category}_feature_subfield_pearson.png'
    plt.savefig(filename, dpi=600, bbox_inches='tight', transparent=True)
plt.show()
```



Vertex-wise correlation

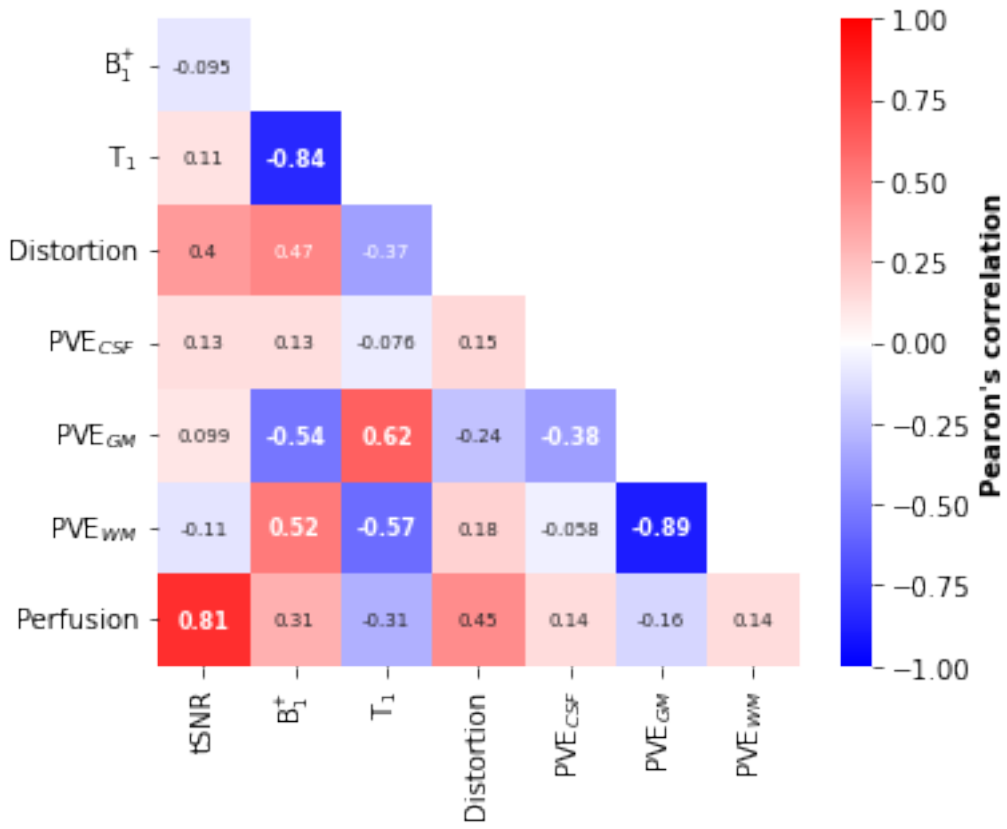
[55]:
```
maps_perm
```

[55]:
```
['tSNR', 'B1map', 'T1', 'Shiftmap', 'pveCSF', 'pveGM', 'pveWM', 'CBF']
```

[56]:
```
vtx_rho   = df_corr.copy()
vtx_pvals = df_corr.copy()

for r, row in enumerate(maps_perm):
    x = load_shape_gii(row)
    for c, col in enumerate(maps_perm):
        if (row != col) & (lt_mask[r,c] == True):
            rho, pval = utilities.correlation_between_maps(
                x, load_shape_gii(col), 5000
            )

            vtx_rho.iloc[r,c]   = rho
            vtx_pvals.iloc[r,c] = pval
```

[57]:
```
# Multiple comparison correction for lower triangle
vtx_pvals_lt       = vtx_pvals.values[lt_mask].flatten()
reject, pvals_corr = multicomp(vtx_pvals_lt, method='fdr_bh')
vtx_pvals_mcc      = np.ones_like(vtx_pvals)
vtx_pvals_mcc[lt_mask] = pvals_corr
```

[62]:
```
fig, ax = plt.subplots(1,1, figsize=(5.5,4.5))

g = sns.heatmap(
    vtx_rho.where(lt_mask).iloc[1:,:-1],
    mask=np.where(vtx_pvals<=.05, True, False)[1:,:-1],
    cmap="bwr", vmin=-1, vmax=1,
    annot=True, fmt='.2g', annot_kws={'size': 7},
    xticklabels=tick_labels[:-1], yticklabels=tick_labels[1:],
    ax=ax
)

g.collections[0].colorbar.set_label("Pearon's correlation", fontweight='bold')

sns.heatmap(
    vtx_rho.where(lt_mask).iloc[1:,:-1],
    mask = np.where(vtx_pvals>.05, True, False)[1:,:-1],
    cmap="bwr", vmin=-1, vmax=1, cbar=False,
    annot=True, fmt='.2g', annot_kws={'size': 8, 'weight': 'bold'},
    xticklabels=tick_labels[:-1], yticklabels=tick_labels[1:],
    ax=ax
)

fig.tight_layout()
if save_fig:
    filename = f'../visualization/unfolded/
 ↪sub-group_{category}_feature_vtx_pearson.png'
    plt.savefig(filename, dpi=600, bbox_inches='tight', transparent=True)
plt.show()
```



0.0.3 BigBrain cell density metrics

[32]:
```
def rugplot(data, i, width=.2, ax=None, **kwargs):
    from matplotlib.collections import LineCollection

    # Specify figure
    ax = ax or plt.gca()

    # Data
    x = data[:,i]
    y_min = np.repeat(i, len(x))-(width/2)
    y_max = np.repeat(i, len(x))+(width/2)

    # Plot properties
    kwargs.setdefault("linewidth", 1)
    kwargs.setdefault("alpha", .05)

    # Specify segments
    segs = np.stack((
            np.c_[y_min, y_max],
```

```
            np.c_[x, x]
        ), axis=-1
    )

    # Show segments in figue
    lc = LineCollection(segs, **kwargs)
    ax.add_collection(lc)
```

[33]:
```
# Load data
bb_left  = spio.loadmat('features_hemi-L.mat')
bb_right = spio.loadmat('features_hemi-R.mat')
```

[34]:
```
# Feature names
bb_features = [ bb_left['FeatureTitles'][0][i][0] for i in range(15) ]
print(bb_features)
```

```
['Thickness', 'Curvature', 'Inner texture', 'Outer texture', 'Gyrification',
'Mean(y)', 'Mean(x)', 'SD(x)', 'Skew(x)', 'Kurt(x)', 'Mean(y.d)', 'Mean(x.d)',
'SD(x.d)', 'Skew(x.d)', 'Kurt(x.d)']
```

[35]:
```
# Load data
bb_data = np.zeros((nvertices,len(bb_features),2))

for i in range(len(bb_features)):
    # Left
    bb_data[:,i,0] = resize(
        bb_left['procFeats'][:,:,i].T,
        (126,254)
    ).flatten(order='C')

    # Right
    bb_data[:,i,1] = resize(
        bb_right['procFeats'][:,:,i].T,
        (126,254)
    ).flatten(order='C')

bb_data_avg = np.nanmean(bb_data, axis=2)
```

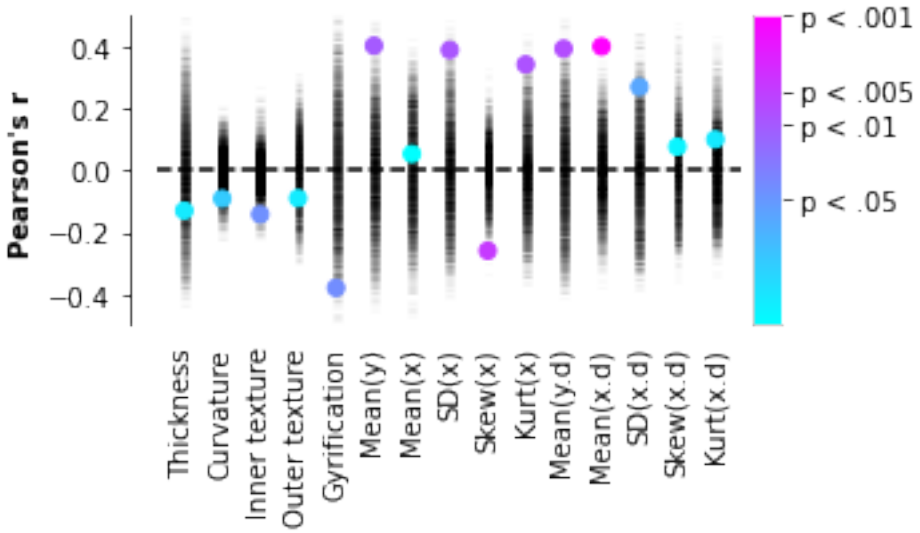
[36]:
```
bb_data_avg = np.hstack((
    bb_data_avg,
    load_shape_gii('CBF').reshape(-1,1)
))
```

[37]:
```
nperms   = 5000
bb_rho   = np.zeros((bb_data_avg.shape[1]-1, nperms+1))
bb_pvals = np.ones((bb_data_avg.shape[1]-1))
```

```
for c in range(bb_data_avg.shape[1]-1):
    rho, pval, perms = utilities.correlation_between_maps(
        bb_data_avg[:,-1:], bb_data_avg[:,c], nperms, return_perms=True
    )

    bb_rho[c, 0]  = rho
    bb_rho[c, 1:] = perms
    bb_pvals[c]   = pval
bb_pvals[bb_pvals==0] = 0.001
```

[38]:
```
fig, ax = plt.subplots(1,1, figsize=(5,3))

xticks_labels = bb_features

sc = ax.scatter(
    x=np.arange(0,len(xticks_labels)),
    y=bb_rho[:, 0],
    c=-np.log10(bb_pvals), cmap='cool',
    vmin=min(-np.log10(bb_pvals)),
    vmax=max(-np.log10(bb_pvals)),
    zorder=10
)

# Add color bar
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cb = plt.colorbar(
    sc, cax=cax,
    ticks=[ -np.log10(i) for i in [.001, .005, .01, .05] ]
)

cax.set_yticklabels(['p < .001','p < .005','p < .01','p < .05',])
cb.outline.set_edgecolor('white')

ax.axhline(y=0, color='black', linestyle='--')

for i in range(len(xticks_labels)):
    rugplot(
        bb_rho[:, 1:].T, i, ax=ax,
        color='black', alpha=.02
    )

ax.set_xticks(np.arange(0,len(xticks_labels)))
ax.set_xticklabels(xticks_labels, rotation=90)
ax.set_ylabel("Pearson's r" , weight='bold')
ax.set_ylim(-.5,.5)
```

```
sns.despine()
ax.spines['left'].set_position(('outward', 10))
ax.spines['bottom'].set_color('none')
ax.xaxis.set_ticks_position('none')

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



[39]:
```
# 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
)
```

[40]:
```
# Iterate through maps and plot unfolded
p = Plot(unfolded, layout='row', views=['dorsal'], zoom=2, size=(500, 300))
p.add_layer(bb_data_avg[:,11], color_range=(4,9)) #,␣
 ↪color_range=maps_dict[idx][2], cmap=maps_dict[idx][1]

fig = p.build(cbar_kws=cbar_unfolded_kws)
```

```
fig.axes[1].set_xlabel(bb_features[11], labelpad=-11, fontsize=12,␣
  ↪fontweight='bold')
fig.savefig('../visualization/unfolded/sub-bb_{}_unfolded.png'.
  ↪format(bb_features[11]), dpi=600, bbox_inches='tight')
fig.show()
```