Traveling Subject法を用いた複数プロトコルデータセットのHarmonize法

Yinghan Zhu

Multi-site data

- 多施設多疾患データプロジェクトの増加(HCP, ADNI, SRPB, CRHD, DecNef)

- コホートデータ(思春期においてcortical thicknessとageに負の相関が見られる)

メーカー、スキャナー、撮像プロトコル、ヘッドコイル、位相方向など

疾患による違い、発達による変化などは機種間差と交絡し、検出しにくくなる

Traveling Subjects

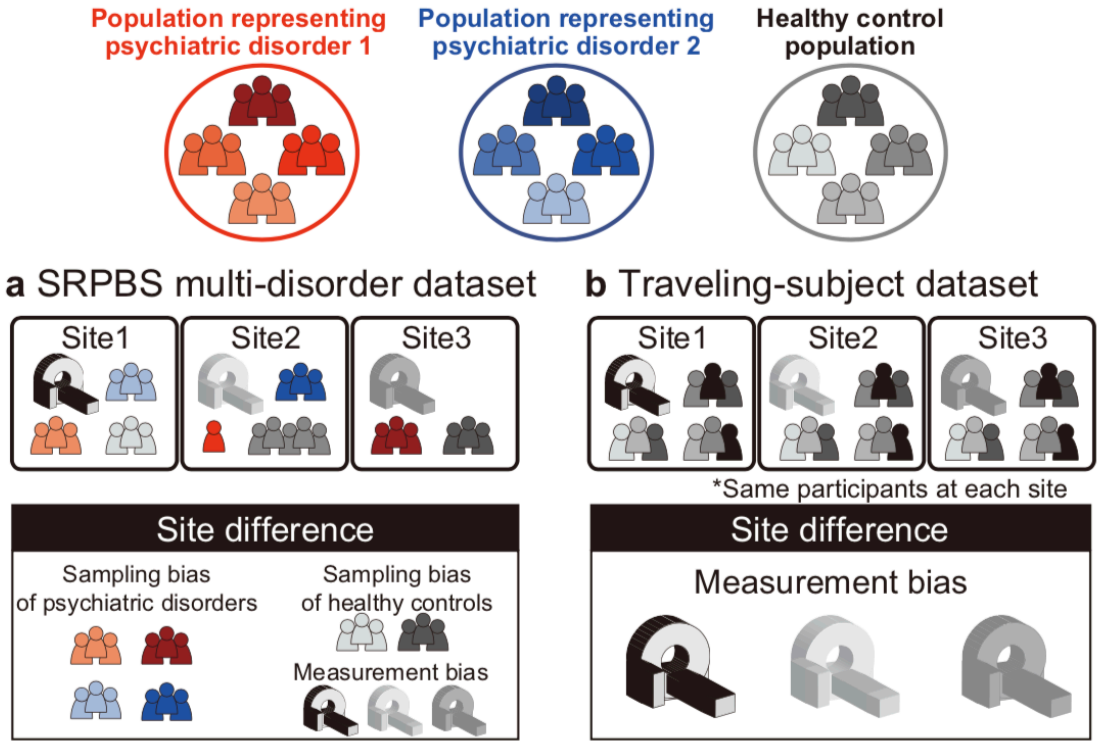
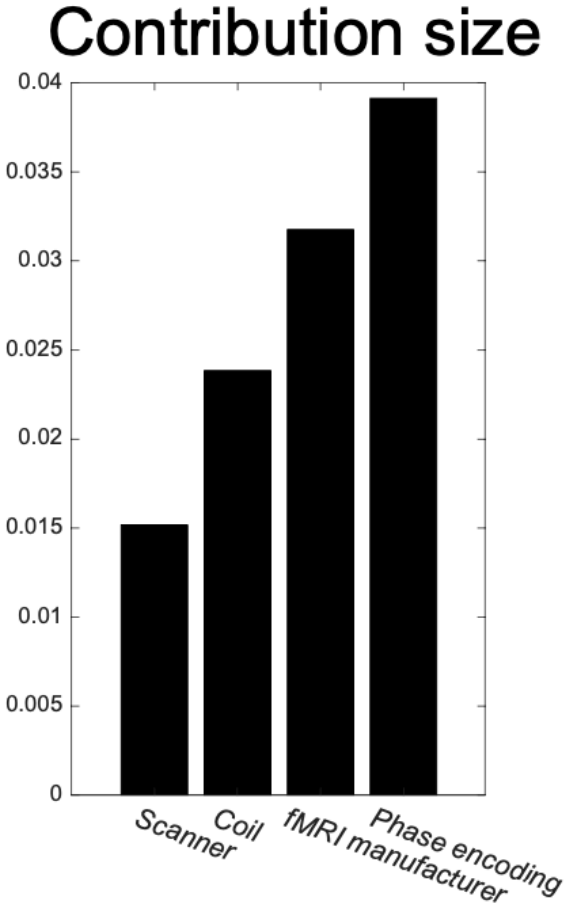


Figure 1: Schematic examples illustrating the two main datasets.
(a) The SRPBS multi-disorder dataset includes patients with psychiatric disorders and healthy controls. The number of patients and scanner types differed among sites. Thus, site differences consist of sampling bias and measurement bias. (b) The traveling-subject dataset includes only healthy controls, and the participants were the same across all sites. Thus, site differences consist of measurement bias only. SRPBS: Strategic Research Program for Brain Sciences.



Yamashita et al., 2019

GLM

1-of-K binary coding scheme

$$Connectivity = \mathbf{x}_m{}^{\mathrm{T}}\boldsymbol{m} + \mathbf{x}_{s_{hc}}{}^{\mathrm{T}}\boldsymbol{s_{hc}} + \mathbf{x}_{s_{mdd}}{}^{\mathrm{T}}\boldsymbol{s_{mdd}} + \mathbf{x}_{s_{scz}}{}^{\mathrm{T}}\boldsymbol{s_{scz}} + \mathbf{x}_d{}^{\mathrm{T}}\boldsymbol{d} + \mathbf{x}_p{}^{\mathrm{T}}\boldsymbol{p} + const + e,$$

$$\text{such that } \sum_j^9 p_j = 0, \sum_k^{12} m_k = 0, \sum_k^6 s_{hc_k} = 0, \sum_k^3 s_{mdd_k} = 0, \sum_k^3 s_{scz_k} = 0,\ d_1(\mathrm{HC}) = 0,$$

m : measurement bias
S_{hc}: Sampling bias for HC
S_{mdd}: Sampling bias for MDD
S_{sch} : Sampling bias for Scz
d:disorder factor
p: participant factor
Const : mean activities across sites across participants
e:e~N(0,γ^{-1}) noise

Yamashita et al., 2019

$$\text{ROI thickness} = X_m{}^{\mathrm{T}}m + X_p{}^{\mathrm{T}}p + \text{const} + e$$

この式にTS01.csv, TS02.CSVを代入して計算を行います。

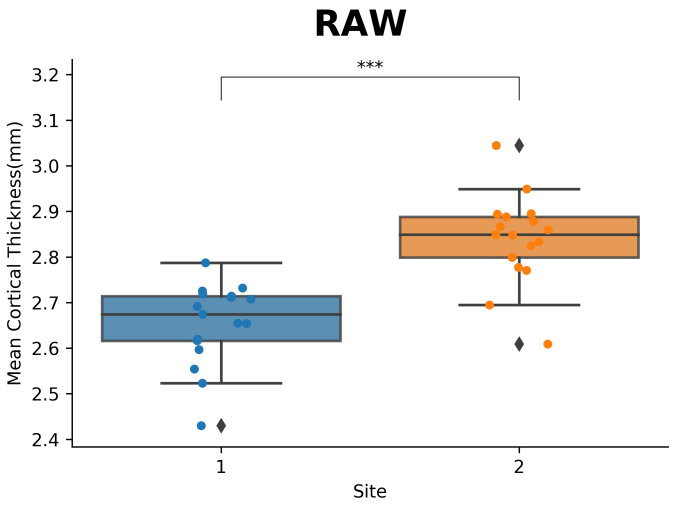
Data

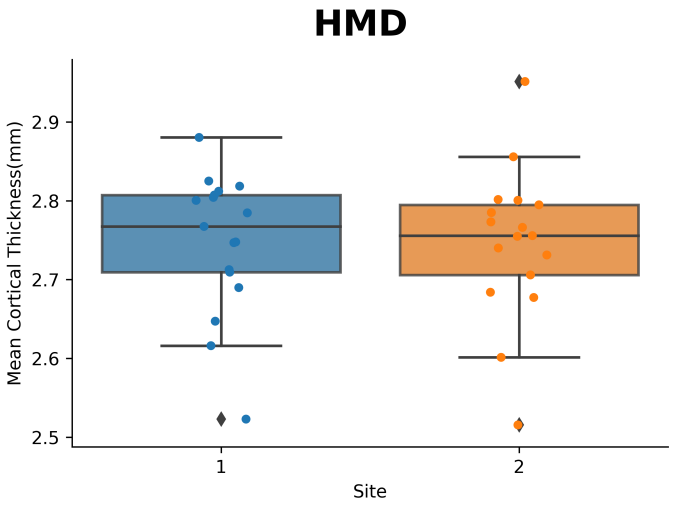
国際脳参画の思春期コホート男女40名

FreeSurfer6.0で前処理したCortical thicknessデータ（実際の数値からランダムに%増減させています）

- 1回目：11歳ごろ(Data01)
- 2回目：16歳ごろ(Data02)

	Manufacture	Scanner	Coil	protocol	TS
Site1(Data01)	Philips	Achieva	SENSE_HEAD_8	SRPB	TS01
Site2(data02)	SIEMENS	Prisma	32Ch Head	CRHD	TS02





Data

TS dataを用いて(TS01.csv, TS02.csv)
サイトごとの係数を求める

GML_HMD-ROI-Tutorial2020 .ipynb

Output files
TS_code.csv
HMD_thickness.csv
site_coef_thickAvg.csv

得られた係数を使って
データ(Data01.csv, Data02.csv)を補正する

APPLY-Tutorial2020 .ipynb

Output files
HMD_Data01.csv
HMD_Data02.csv
thick_mean_raw.png
thick_mean_hmd.png

Python code

GML_HMD-ROI-Tutorial2020 .ipynbを開いてください。

```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
import datetime
from sklearn import metrics as mtr
from sklearn.metrics import mean_squared_error,auc,roc_curve
import itertools
import numpy.matlib
from math import sqrt
```

モジュールを読み込み

Python code

```
# ファイルを読み込む

TS01=pd.read_csv('TS01.csv', skipinitialspace=True)
TS02=pd.read_csv('TS02.csv', skipinitialspace=True)

#SubjIDの列にサフィックスを追加
TS01['SubjID'] = TS01['SubjID'].astype(str)+'-01'
TS02['SubjID'] = TS01['SubjID'].astype(str)+'-02'

# TSデータを縦に連結
TS_data=pd.concat([TS01, TS02], axis=0)
```

```
## 1-of-K binary
# 個人差を推定するための対角行列を作成する
p_code=pd.DataFrame(np.diag(np.ones(17)))

# 列名を作成
p_code.columns=list(range(len(p_code.columns)))
p_code.columns=([('P') + str(i) for i in np.arange(1,18,1)])
```

```
1 p_code.head()
```

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

pandasの要約を表示する

data.head() データの最初の５行を表示してくれます。

data.tail()　データの最後の５行を表示してくれます。

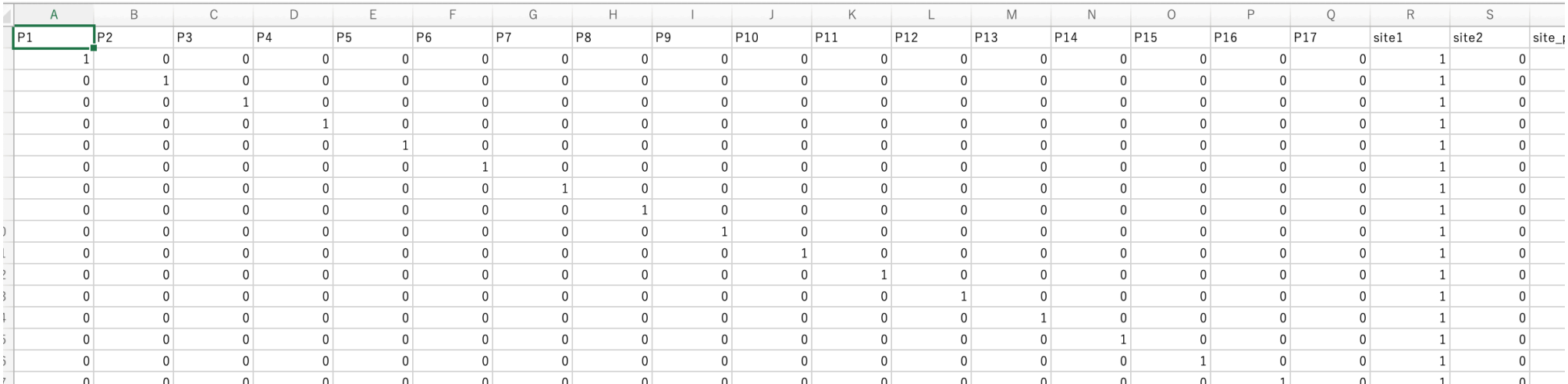
data.corr()　データフレームの列同士の相関を求めてくれます。

Python code

```
#  1回目撮像のコーディングを作成
code01=p_code.copy()
code01['site1']=1
code01['site2']=0
code01['site_plot']=1

# 2回目撮像のコーディングを作成
code02=p_code.copy()
code02['site1']=0
code02['site2']=1
code02['site_plot']=2

TS_code=pd.concat([code01, code02], axis=0)
TS_code.to_csv('TS_code.csv', index=False)#コーティングをCSVに書き出して保存
```



P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	site1	site2	site_
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	

Python code

```
# 上で作成したTSデータとコーディングデータをコピーする
ROI = TS_data.copy()
ROI=ROI.drop(['SubjID'], axis='columns')
site=TS_code.drop(['site_plot'], axis='columns')


n_samples, n_features = ROI.shape
print("%d samples, %d features" % (n_samples, n_features))
print(" ")


# 切片を求める
mean=np.mean(ROI, axis=0)
print(mean.shape)

mean_roi=np.matlib.repmat(mean,n_samples,1)
print(mean.shape,site.shape)

ROI_modified=ROI-mean_roi

#  Ridge回帰を定義
clf=Ridge(alpha=0.01,fit_intercept=False)

clf.fit(site,ROI_modified)
```

Python code

```
# site effectの係数を取り出す
coef_site1 = numpy.matlib.repmat(clf.coef_[:,17],n_samples,1)
coef_site2 = numpy.matlib.repmat(clf.coef_[:,18],n_samples,1)


site1 = numpy.matlib.repmat(site['site1'],n_features,1).T
site2 = numpy.matlib.repmat(site['site2'],n_features,1).T

# site effectを求める
site1_effect=coef_site1*site1
site2_effect=coef_site2*site2


# site effectを取り除く
ROI_HMD=ROI-site1_effect-site2_effect
ROI_HMD=pd.concat([TS_data['SubjID'],ROI_HMD], axis=1)
ROI_HMD.to_csv('HMD_thickness.csv', index=False)

# site effectの係数を書き出して保存する
coef=pd.DataFrame(np.vstack((clf.coef_[:,17],
                             clf.coef_[:,18]
                                  )).T,
                             columns=['site1', 'site2'])
coef.to_csv('site_coef_thickAvg.csv', index=False)
```

Apply to data

APPLY-Tutorial2020 .ipynbを開いてください

```
#site1#
#まずデータ1を補正する

data01=pd.read_csv('Data01.csv', skipinitialspace=True)
coef=pd.read_csv('site_coef_thickAvg.csv', skipinitialspace=True)

#SubjID列を削除し、ROIデータのみのデータフレームを作成する
ROI_data=data01.drop(['SubjID'],axis='columns')
n_samples,n_feature=ROI_data.shape  #新しいデータフレームのサイズを確認

coef_site1 = nb.repmat(coef['site1'],n_samples,1)
print(coef_site1.shape)

#site1によるバイアスを求める
site1 = np.ones_like(ROI_data)
site1_effect=coef_site1*site1

#site1によるバイアスを取り除く
HMD_ROI_data=ROI_data-site1_effect
HMD_ROI=pd.concat([data01['SubjID'],HMD_ROI_data], axis=1)
HMD_ROI.to_csv('HMD_Data01.csv', index=False)
```

site2のデータについて同じ手順で行う

Visualization