```python
# -*- coding: utf-8 -*-
"""
Example Python script:

Multiple imputation and formatting ML predictions with data

v2:
    Add 23-category occ and immigration status into training
    Remove redundant package for other models
v3:
    Finetune workflow


@author: Xingyun Wu

Initial: 3/17/2025
Latest: 3/17/2025
"""

# data processing
import os
import pandas as pd
import numpy as np
import random
import pickle
import sklearn.metrics as metrics
# summary
from scipy import stats
import matplotlib.pyplot as plt


# work directory
os.chdir('/Users/xywu/OneDrive - Johns Hopkins/personal/dissertation')
os.chdir('C:/Users/xwu70/OneDrive - Johns Hopkins/personal/dissertation')


# version control
tmp_version = 'v3'


# load data
cdt = pd.read_csv('paper2/data/couple_data_for_dyads.csv', low_memory = False)
list(cdt.columns)

# load estimated models
with open('paper2/results/ml_pred_replicates_{}.pkl'.format(tmp_version), 'rb') a
    gbr_d = pickle.load(file)
# check replicates
len(set([v['seed'] for k, v in gbr_d['female'].items() if k != 'time']))
```

```
"""
data formatting
"""

# data by sex
dt = dict()
dt['all'] = dict()
dt['all']['data'] = cdt.copy(deep = True)
dt['female'] = dict()
dt['female']['data'] = cdt[cdt['female'] == 1].copy(deep = True)
dt['male'] = dict()
dt['male']['data'] = cdt[cdt['female'] == 0].copy(deep = True)


## X
for k, v in dt.items():
    tdt = v['data']
    X = tdt[['m_age', 'm_ba', 'f_age', 'f_ba',
             'm_earn', 'f_earn', 'm_wkhrs', 'f_wkhrs',
             'h_otheradult', 'h_ageyc', 'urban', 'hhowner', 'fambus',
             'm_fmrsn', 'm_wkrsn', 'f_fmrsn', 'f_wkrsn',
             'pcp_family_home', 'pcp_family_out']].copy(deep = True)
    # reth dummies
    m_reth = pd.get_dummies(data = tdt['m_reth'], drop_first = True, dtype = 'int
    m_reth.columns
    m_reth.columns = ['m_black', 'm_hisp', 'm_asian', 'm_retho']
    f_reth = pd.get_dummies(data = tdt['f_reth'], drop_first = True, dtype = 'int
    f_reth.columns = ['f_black', 'f_hisp', 'f_asian', 'f_retho']
    # child dummies
    hhkid = pd.get_dummies(data = tdt['h_numhhchd'], drop_first = True, dtype = '
    hhkid.columns
    hhkid.columns = ['hhkid1', 'hhkid2']
    # year dummies
    tyear = pd.get_dummies(data = tdt['year'], drop_first = True, dtype = 'int')
    tyear.columns = ['yr' + str(x) for x in list(tyear.columns)]
    # day dummies
    tday = pd.get_dummies(data = tdt['day'], drop_first = True, dtype = 'int')
    tday.columns = ['day' + str(x) for x in list(tday.columns)]
    # month dummies
    tmonth = pd.get_dummies(data = tdt['month'], drop_first = True, dtype = 'int'
    tmonth.columns = ['month' + str(x) for x in list(tmonth.columns)]

    # emp dummies
    m_emp = pd.get_dummies(data = tdt['m_emp'], drop_first = True, dtype = 'int')
    m_emp.columns
    m_emp.columns = ['m_ptnp', 'm_ptpr', 'm_ftnp', 'm_ftpr']
    f_emp = pd.get_dummies(data = tdt['f_emp'], drop_first = True, dtype = 'int')
    f_emp.columns = ['f_ptnp', 'f_ptpr', 'f_ftnp', 'f_ftpr']
    # merge prof and non-prof in emp status
    m_emp['m_pt'] = m_emp['m_ptnp'] + m_emp['m_ptpr']
    m_emp['m_ft'] = m_emp['m_ftnp'] + m_emp['m_ftpr']
```

```
        f_emp['f_pt'] = f_emp['f_ptnp'] + f_emp['f_ptpr']
        f_emp['f_ft'] = f_emp['f_ftnp'] + f_emp['f_ftpr']
        # drop redundant columns
        m_emp.drop(columns = ['m_ptnp', 'm_ptpr', 'm_ftnp', 'm_ftpr'],
                   axis = 1, inplace = True)
        f_emp.drop(columns = ['f_ptnp', 'f_ptpr', 'f_ftnp', 'f_ftpr'],
                   axis = 1, inplace = True)


        # add 23-category occ
        m_occ = pd.get_dummies(data = tdt['m_occ2'], drop_first = True,
                               dummy_na = False, dtype = 'int', prefix = 'm_occ')
        m_occ.columns = ['m_occ' + str(x) for x in range(2, 23)]
        f_occ = pd.get_dummies(data = tdt['f_occ2'], drop_first = True,
                               dummy_na = False, dtype = 'int', prefix = 'f_occ')
        f_occ.columns = ['f_occ' + str(x) for x in range(2, 23)]


        # add immigration status
        m_im = pd.get_dummies(data = tdt['m_imstat'], drop_first = True,
                              dummy_na = False, dtype = 'int', prefix = 'm_im')
        m_im.columns = ['m_im1', 'm_im2']
        f_im = pd.get_dummies(data = tdt['f_imstat'], drop_first = True,
                              dummy_na = False, dtype = 'int', prefix = 'm_im')
        f_im.columns = ['f_im1', 'f_im2']

        # add marital status
        h_cohab = pd.get_dummies(data = tdt['spousepres'], drop_first = True,
                                 dummy_na = False, dtype = 'int', prefix = 'h_cohab')
        h_cohab.columns = ['h_cohab']

        # concat
        v['X'] = pd.concat(
            [X, m_reth, f_reth, m_im, f_im, m_emp, f_emp, m_occ, f_occ,
             hhkid, h_cohab, tyear, tday, tmonth], axis = 1).copy(deep = True)

        # interactions
        v['X']['hhkid1_ageyc'] = v['X']['hhkid1'] * v['X']['h_ageyc']
        v['X']['hhkid2_ageyc'] = v['X']['hhkid2'] * v['X']['h_ageyc']

        # remove intermediate objects
        del [tdt, X, m_reth, f_reth, m_im, f_im, m_emp, f_emp, m_occ, f_occ,
             hhkid, h_cohab, tyear, tday, tmonth]

# drop indices
del [k, v]
# check dimension
dt['female'].keys()
dt['female']['X'].shape
dt['male'].keys()
dt['male']['X'].shape
```

```
## y
for k, v in dt.items():
    print(k)
    tdt = v['data']
    tdt['fmtotal'].hist(bins = 50)
    plt.title(k + ' outcome: original')
    plt.show()
    v['y'] = np.log(tdt['fmtotal'] + 1)
    v['z'] = tdt['fmtotal']
    # inspect distribution after transformation
    v['y'].hist(bins = 50)
    plt.title(k + ' outcome: log-transformed')
    plt.show()
# drop intermediate items
del [k, v, tdt]
# check items
dt.keys()
dt['female'].keys()
# check type
type(dt['female']['y'])
type(dt['male']['y'])



"""
MI on unobserved partners
"""

'''
auxiliary functions
'''

def impute_by_sex(feature, outcome, model, est_female = True, seed = 42):
    '''
    Parameters
    ----------
    feature: pandas df of features for prediction
    outcome: numpy array of observed outcome
    model: tratined predictive ML model
        male model on female-observed subsample
        female model on male-observed subsample
    est_female: whether female was the observed person in couple
    seed: an integer for random seed if seed not provided for noise

    Returns
    -------
    rv: pandas df combining features and prediction
    '''
```

```
    # predict
    pred = model.predict(feature)

    # diff random seed for male
    if not est_female:
        random.seed(seed)
        seed = random.randint(1, 10000)
    # random noise
    random.seed(seed)
    noise = np.random.uniform(low = -2.0, high = 2.0, size = feature.shape[0])

    # combine features with outcomes
    rv = feature.copy(deep = True)
    rv.reset_index(drop = False, inplace = True)
    if est_female:
        rv['f_lnfm'] = np.array(pred) + np.array(noise)
        rv['m_lnfm'] = np.array(outcome)
    else:
        rv['f_lnfm'] = np.array(outcome)
        rv['m_lnfm'] = np.array(pred) + np.array(noise)

    return rv



def impute_sample(data_d, model_d, model_id):
    '''
    Parameters
    ----------
    data_d: dictionary with female data and male data
    model_d: dictionary of female's and male's trained predictive models
    model_id: id of replicates

    Returns
    -------
    rv: pandas df combining features and prediction
    '''

    # female's unobserved partner
    fdf = impute_by_sex(feature = data_d['female']['X'],
                        outcome = data_d['female']['y'],
                        model = model_d['male'][str(model_id)]['model'],
                        est_female = False,
                        seed = model_d['male'][str(model_id)]['seed'])
    # add female's own predicted value
    ftmp = impute_by_sex(feature = data_d['female']['X'],
                        outcome = data_d['female']['y'],
                        model = model_d['female'][str(model_id)]['model'],
                        est_female = True,
                        seed = model_d['female'][str(model_id)]['seed'])
    ftmp.rename(columns = {'f_lnfm': 'y_pred'}, inplace = True)
```

```
    fdf = fdf.merge(ftmp[['index', 'y_pred']], how = 'left', on = 'index')

    # male's unobserved partner
    mdf = impute_by_sex(feature = data_d['male']['X'],
                        outcome = data_d['male']['y'],
                        model = model_d['female'][str(model_id)]['model'],
                        est_female = True,
                        seed = model_d['female'][str(model_id)]['seed'])
    # add male's own predicted value
    mtmp = impute_by_sex(feature = data_d['male']['X'],
                         outcome = data_d['male']['y'],
                         model = model_d['male'][str(model_id)]['model'],
                         est_female = False,
                         seed = model_d['male'][str(model_id)]['seed'])
    mtmp.rename(columns = {'m_lnfm': 'y_pred'}, inplace = True)
    mdf = mdf.merge(mtmp[['index', 'y_pred']], how = 'left', on = 'index')

    # append
    rv = pd.concat([fdf, mdf], axis = 0, ignore_index = True)

    # merge-in caseid by original index
    adf = data_d['all']['data'].copy(deep = True)
    adf.reset_index(drop = False, inplace = True)
    rv = rv.merge(adf[['index', 'caseid', 'female', 'fmtotal']],
                  how = 'left', on = 'index')

    # output
    return rv


'''
wrapper function
'''


def main(data_d, model_d, path = 'paper2/results/mi_sample', print_iter = True, o
    '''

    Parameters
    ----------
    data_d : dictionary of data frames for female and male data separately
    model_d : dictionary of trained 100 replicates of models for female and male

    Returns
    -------
    rv: a list of data frames

    '''

    rv = list()
    seeds = [int(v['seed']) for k, v in model_d['female'].items() if k != 'time']
```

```
    for i in range(len(seeds)):
        if(print_iter):
            print('Sample {}'.format(i))
        tdf = impute_sample(data_d = data_d, model_d = model_d, model_id = i)
        tdf = tdf[['caseid', 'm_lnfm', 'f_lnfm', 'y_pred']]
        rv.append(tdf)
        if output:
            tdf.to_csv(path + '/sample{}.csv'.format(i), index = False)

    return rv


'''
implement
'''

# one model
t0 = impute_sample(data_d = dt, model_d = gbr_d, model_id = 5)
t0.head()
t0.tail()
print('Any missing? Female = {}, male = {}'.format(
    t0['f_lnfm'].isnull().any(), t0['m_lnfm'].isnull().any()))


# output to data files
mi_lst = main(data_d = dt, model_d = gbr_d, output = True)


# not output to files
mi_lst = main(data_d = dt, model_d = gbr_d, output = False)



"""
evaluation
"""

mi_lst = []
for i in range(100):
    mi_lst.append(pd.read_csv('paper2/results/mi_sample/sample{}.csv'.format(i)))


'''
sample-level
'''

m_spd = list(zip(
    # range(len(mi_lst)),
    [metrics.mean_squared_error(np.log(x.loc[x['female'] == 0, 'fmtotal'] + 1),
                                x.loc[x['female'] == 0, 'y_pred']) for x in mi_ls
```

```
    [metrics.mean_absolute_error(np.log(x.loc[x['female'] == 0, 'fmtotal'] + 1),
                                 x.loc[x['female'] == 0, 'y_pred']) for x in mi_l
m_spd[:5]
m_spd = pd.DataFrame(m_spd, columns = ['mse', 'mae']) # 'sample',

f_spd = list(zip(
    # range(len(mi_lst)),
    [metrics.mean_squared_error(np.log(x.loc[x['female'] == 1, 'fmtotal'] + 1),
                                x.loc[x['female'] == 1, 'y_pred']) for x in mi_ls
    [metrics.mean_absolute_error(np.log(x.loc[x['female'] == 1, 'fmtotal'] + 1),
                                 x.loc[x['female'] == 1, 'y_pred']) for x in mi_l
f_spd[:5]
f_spd = pd.DataFrame(f_spd, columns = ['mse', 'mae']) # 'sample',


## combine
spdf = pd.concat([m_spd, f_spd], axis = 0, keys = ['male', 'female'])
spdf.reset_index(drop = False, inplace = True)
spdf.rename(columns = {'level_0': 'gender', 'level_1': 'sample'}, inplace = True)
spdf.to_csv('paper2/results/mi_desc/sample_mse_mae_{}.csv'.format(
    tmp_version), index = False)
# summarize
spsum = pd.DataFrame(spdf[['gender', 'mse', 'mae']].groupby('gender').describe())
spsum.to_csv('paper2/results/mi_desc/sample_mse_mae_summary_{}.csv'.format(
    tmp_version), index = True)


'''
observation-level
'''

obsd = list(zip(mi_lst[0].index, mi_lst[0].female, np.log(mi_lst[0].fmtotal + 1))
obsd = [[x[0], x[1], x[2]] for x in obsd]
for i in range(len(obsd)):
    t_out = [x.loc[i, 'y_pred'] for x in mi_lst]
    obsd[i] += [np.mean(t_out), np.std(t_out), np.min(t_out),
                np.quantile(t_out, q = .25), np.median(t_out),
                np.quantile(t_out, q = .75), np.max(t_out)]
obsd = pd.DataFrame(obsd, columns = ['index', 'female', 'y', 'mean', 'std',
                                     'min', 'q25', 'mid', 'q75', 'max'])

## coverage rate by sex
obsd['obs_within'] = np.where(
    obsd['y'] >= obsd['min'], np.where(obsd['y'] <= obsd['max'], 1, 0), 0)
print(obsd.groupby(by = 'obs_within')['index'].count()) # 1733, 22961
print(obsd.groupby(by = ['female', 'obs_within'])['index'].count())
print(f'Coverage rate: female = {12363/(382 + 12363): .3f}, male = {10598/(1351 +
# Coverage rate: female =  0.970, male =  0.887


## MAE from median
```

```
# all
obsd['mae_mid'] = obsd['y'] - obsd['mid']
stats.describe(obsd['mae_mid'])
plt.hist(obsd['mae_mid'], bins = 100)
plt.show()

# summary by sex
print('MAE from median: \nMale:\n{}\nMedian = {}\nFemale:\n{}\nMedian = {}'.forma
    stats.describe(obsd.loc[obsd['female'] == 0, 'mae_mid']),
    np.median(obsd.loc[obsd['female'] == 0, 'mae_mid']),
    stats.describe(obsd.loc[obsd['female'] == 1, 'mae_mid']),
    np.median(obsd.loc[obsd['female'] == 1, 'mae_mid'])))
'''
MAE from median:

Male:
DescribeResult(nobs=11949, minmax=(-4.359422078100892, 4.144447407133434),
               mean=-0.0013882960783622462, variance=1.9536890205393431,
               skewness=-0.5002787822144392, kurtosis=0.4333386842045859)
Median = 0.07618714275956862

Female:
DescribeResult(nobs=12745, minmax=(-5.912734035442815, 3.098560907478685),
               mean=-0.0005932950793437807, variance=0.964766082975963, s
               kewness=-1.741453929977417, kurtosis=5.872231345989219)
Median = 0.09793823222269982
'''

# plot
fig, axs = plt.subplots(2, 1, sharex = True, sharey = True)
axs[0].hist(obsd.loc[obsd['female'] == 0, 'mae_mid'], bins = 60, edgecolor = 'whi
axs[0].set_xlabel('MAE')
axs[0].set_ylabel('Observed male\nFreq.')
axs[0].text(5.5, 0.95, 'mean = 0.004, mid = 0.107\nskewness = -0.499\nkurtosis = 
            fontsize = 10, bbox = dict(facecolor = 'white', edgecolor = 'black'),
            ha = 'left', va = 'bottom')
axs[1].hist(obsd.loc[obsd['female'] == 1, 'mae_mid'], bins = 60, edgecolor = 'whi
axs[1].set_xlabel('MAE')
axs[1].set_ylabel('Observed female\nFreq.')
axs[1].text(5.5, 0.95, 'mean = -0.000, mid = 0.126\nskewness = -1.856\nkurtosis =
            fontsize = 10, bbox = dict(facecolor = 'white', edgecolor = 'black'),
            ha = 'left', va = 'bottom')
fig.suptitle('Mean absolute error (MAE)\nBetween observed and the median of 100 s
plt.savefig('paper2/results/mi_desc/gb_mae_tomid_by_sample_{}.png'.format(
    tmp_version), dpi = 300, bbox_inches = 'tight')
plt.show()
```

```r
#'Example R script:
#'
#'Run APIM with formal tests
#'  v8: change fanincome + r_cpearn to low-mid-high HH annual income
#'      use construct_var_v5.R instead for variable construction,
#'        for the above or subsequent changes
#'
#'Xingyun Wu
#'
#'Initial: 4/20/2025
#'Latest: 4/20/2025

library(writexl)
library(Hmisc)

options(scipen = 999)

# working directory
setwd('~/OneDrive - Johns Hopkins/personal/dissertation/')

# Run data processing script: my written data pre-processing function
source('paper2/script/construct_var_v5.R')

#'Run apim_function script for my written auxiliary functions:
#'  APIM implementation:
#'      sur_main(), which calls my sur_pool() for Rubin's rules
#'  Estimates visualization:
#'      plot_estimates() for coef plot of specified variables
#'  Standardization of estimates:
#'      std_set() for specified set of variables for coef plot
#'      std_model() for full model for final results table
source('paper2/script/apim_function_v6.R')


############
# synthetic
############

#==============
# SUR 0: base
#==============

## define equations
# male model
eq0_male <- m_lnfm ~ m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# female model
eq0_female <- f_lnfm ~ m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# compile
eq0_cmp <- list(male = eq0_male, female = eq0_female)

# on all samples
sur0 <- sur_main(equations = eq0_cmp, one_sample = t1, observed_data = cdt,
                 n_sample = 100)

# performance
sur0$pooled_sum$cor_pooled
sur0$pooled_sum$metric_mean


#===============
# SUR 1: ba/emp for H_edu and H_emp
#===============
```

```
## define equations
# male model
eq1_male <- m_lnfm ~ m_ba + f_ba + memp2 + femp2 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# female model
eq1_female <- f_lnfm ~ m_ba + f_ba + memp2 + femp2 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# compile
eq1_cmp <- list(male = eq1_male, female = eq1_female)


# implement
sur1 <- sur_main(equations = eq1_cmp, one_sample = t1, observed_data = cdt,
                 n_sample = 100)

# standardize selected coef
sum1st <- std_set(
  df = t1, estimates = sur1$pooled_sum, model = 'sur', types = rep('factor', 4),
  variables = c('m_ba1', 'f_ba1', 'memp2ft', 'femp2ft'))


# coef plot
plot_estimates(estimates = sur1$pooled_sum$est, select_variables = T,
               variables = c('m_ba1', 'f_ba1', 'memp2ft', 'femp2ft'),
               title = 'SUR: Model 1 raw', output = T)
plot_estimates(estimates = sum1st, select_variables = T,
               variables = c('m_ba1', 'f_ba1', 'memp2ft', 'femp2ft'),
               title = 'SUR: Model 1 standardized', output = T,
               xlim = c(-0.4, 0.2))
plot_estimates(estimates = sum1st, select_variables = T,
               variables = c('m_ba1', 'f_ba1'),
               title = 'SUR: Model 1 standardized (edu)', output = T,
               xlim = c(-0.1, 0.2), height = 3)
plot_estimates(estimates = sum1st, select_variables = T,
               variables = c('memp2ft', 'femp2ft'),
               title = 'SUR: Model 1 standardized (emp)', output = T,
               xlim = c(-0.4, 0.2), height = 3)


#========
# SUR 2: paired edu, seperated emp for H_edu
#========

## define equations
# male model
eq2_male <- m_lnfm ~ eduprd + memp2 + femp2 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# female model
eq2_female <- f_lnfm ~ eduprd + memp2 + femp2 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# compile
eq2_cmp <- list(male = eq2_male, female = eq2_female)


# implement
sur2 <- sur_main(equations = eq2_cmp, one_sample = t1, observed_data = cdt,
                 n_sample = 100)

# standardize coef of selected variables
sum2st <- std_set(
```

```
  df = t1, estimates = sur2$pooled_sum, model = 'sur', types = rep('factor', 5),
  variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb', 'memp2ft', 'femp2ft'))

# coefplot
plot_estimates(estimates = sur2$pooled_sum$est, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb',
                             'memp2ft', 'femp2ft'),
               title = 'SUR: Model 2 raw', output = T)
plot_estimates(estimates = sum2st, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb',
                             'memp2ft', 'femp2ft'),
               title = 'SUR: Model 2 standardized', output = T , xlim = c(-0.4, 0.3))
plot_estimates(estimates = sum2st, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb'),
               title = 'SUR: Model 2 standardized (edu)', output = T ,
               xlim = c(-0.1, 0.3), height = 4)
plot_estimates(estimates = sum2st, select_variables = T,
               variables = c( 'memp2ft', 'femp2ft'),
               title = 'SUR: Model 2 standardized (emp)', output = T ,
               xlim = c(-0.4, 0.2), height = 3)


#========
# SUR 3: paired edu, paired emp for H_emp
#========

## define equations
# male model
eq3_male <- m_lnfm ~ eduprd + emp4 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# female model
eq3_female <- f_lnfm ~ eduprd + emp4 +
  m_cohort + f_cohort + m_reth + f_reth + m_im + f_im +
  hh_type + h_ageyc + fminc + cohab + urban + weekend + female + year
# compile
eq3_cmp <- list(male = eq3_male, female = eq3_female)


# implement
sur3 <- sur_main(equations = eq3_cmp, one_sample = t1, observed_data = cdt,
                 n_sample = 100)

# standardize selected variables
sum3st <- std_set(
  df = t1, estimates = sur3$pooled_sum, model = 'sur', types = rep('factor', 6),
  variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb',
                'emp4spec', 'emp4reverse', 'emp4other'))

# coefplot
plot_estimates(estimates = sur3$pooled_sum$est, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb',
                             'emp4spec', 'emp4reverse'),
               title = 'SUR: Model 3 raw', output = T)
plot_estimates(estimates = sum3st, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb',
                             'emp4spec', 'emp4reverse'),
               title = 'SUR: Model 3 standardized',
               output = T , xlim = c(-0.2, 0.47))
plot_estimates(estimates = sum3st, select_variables = T,
               variables = c('eduprd2ba', 'eduprdmbfn', 'eduprdmnfb'),
               title = 'SUR: Model 3 standardized (edu)', output = T ,
               xlim = c(-0.1, 0.3), height = 4)
plot_estimates(estimates = sum3st, select_variables = T,
               variables = c('emp4spec', 'emp4reverse'),
```

```
               title = 'SUR: Model 3 standardized (emp)', output = T ,
               xlim = c(-0.2, 0.47), height = 3)


#==============
# put together
#==============

## original pooled estimates
colnames(sur1$pooled_sum$est)
scols <- c('variable', 'male_beta', 'male_sig', 'male_se', 'male_pval',
           'female_beta', 'female_sig', 'female_se', 'female_pval')
sur_out <- sur0$pooled_sum$est[, scols] %>%
  full_join(sur1$pooled_sum$est[, scols], by = 'variable') %>%
  full_join(sur2$pooled_sum$est[, scols], by = 'variable') %>%
  full_join(sur3$pooled_sum$est[, scols], by = 'variable')
colnames(sur_out)
# rename columns
num_model <- 4
num_col <- 4
tcols <- as.data.frame(matrix(NA, nrow = num_model * 2 * num_col, ncol = 4))
colnames(tcols) <- c('model', 'sex', 'stat', 'name')
tcols$model <- c(rep('m0', 2 * num_col), rep('m1', 2 * num_col),
                 rep('m2', 2 * num_col), rep('m3', 2 * num_col))
tcols$sex <- rep(c(rep('male', num_col), rep('female', num_col)), num_model)
tcols$stat <- rep(c('beta', 'sig', 'se', 'pval'), 2 * num_model)
tcols$name <- paste(tcols$model, tcols$sex, tcols$stat, sep = '_')
colnames(sur_out) <- c('variable', tcols$name)
colnames(sur_out)
# output together with standardized

# extract estimates of selected variables
sum1st$model <- 1
sum2st$model <- 2
sum3st$model <- 3
sur_select <- rbind(sum1st, sum2st, sum3st)

## standardized coefficients
# standardize
sur0st <- std_model(estimates = sur0$pooled_sum, model = 'sur')
sur1st <- std_model(estimates = sur1$pooled_sum, model = 'sur')
sur2st <- std_model(estimates = sur2$pooled_sum, model = 'sur')
sur3st <- std_model(estimates = sur3$pooled_sum, model = 'sur')
# compile
surst_full <- std_compile(
  models = list(
    'm0' = sur0st, 'm1' = sur1st, 'm2' = sur2st, 'm3' = sur3st),
  model_type = 'sur', sur_metric = 'mean',
  orig_info = list(
    'm0' = sur0$pooled_sum, 'm1' = sur1$pooled_sum,
    'm2' = sur2$pooled_sum, 'm3' = sur3$pooled_sum))
# add raw estimates before standardization
surst_full$raw <- sur_out
surst_full$selected <- sur_select

# output
write_xlsx(surst_full,
           'paper2/results/mi_model/sur_mi_standardized_20250420.xlsx')
#'version:
#'  20250420: replace famincome + r_cpearn with low-mid-high HH income

# save estimated model for hypothesis testing
save(sur1, sur3,
     file = 'paper2/results/mi_model/sur_hypothesis_test_20250420.rda')
```

```
#=====
# example auxiliary function called in sur_main()
#=====
# formal test for significance of residual correlation between equations

myBPtest <- function(data, model, display = F){

  # extract residuals for each equation
  res_male <- residuals(model)$male
  res_female <- residuals(model)$female

  # calculate correlation
  res_corr <- cor(res_male, res_female)

  # calculate the LM statistic
  N <- nrow(data)  # Number of observations
  LM_stat <- N * (res_corr^2)

  # find the p-value using a chi-squared distribution with 1 degree of freedom
  p_value <- 1 - pchisq(LM_stat, df = 1)

  # display results
  if(display){
    cat('Residual corr =', res_corr,
        '\nBreusch-Pagan Test Statistic =', LM_stat,
        '\np-value =', sprintf('%.4f', p_value), "\n")
  }

  # return
  rv <- list()
  rv$res_corr <- res_corr
  rv$stat <- LM_stat
  rv$p_value <- p_value
  return(rv)
}
```

```
/*
Example Stata script: Process monthly labor force information from CPS

Xingyun Wu

Initial: 10/20/2024
Latest: 10/20/2024
*/

* change working directory
cd "~/OneDrive - Johns Hopkins/personal/dissertation/"


* read in the data
clear
do "data/cps_14to19/cps_00008.do"


/*=====
monthly response: unit = household-person-month
=====*/

* select monthly response
tab asecflag, miss
tab asecflag, miss nol
tab month asecflag, miss
keep if asecflag != 1
tab month

** employment status
tab empstat, miss
tab empstat, miss nol
//recode empstat (0 = .)


** employed--absent last week
tab whyabsnt, miss
tab whyabsnt, miss nol
// 7 = childcare problems, 8 = other fm/ps obligation, 9 =
  maternity/paternity leave
// 5 = vacation/personal days,
// 10 = labor dispute, 11 = weather affected job
// 12 = school/training
// 6 = own illness/injury/medical problems
//recode whyabsnt (0 = .)
tab whyabsnt empstat, miss


```

```
47
48 ** part-time last week
49 tab whyptlwk, miss
50 tab whyptlwk, miss nol
51 // 121 = childcare problems, 122 = other fm/ps obligations
52 // 30 = weather affected job, 40 = labor dispute, 10 = slack work,
   business conditions
53 // 90 = holiday, 111 = vacation/personal day
54 // 123 = school/training
55 // 100 = own illness, 101 health/medical limitation
56 tab whyptlwk empstat, miss
57
58
59 ** unemployed
60 * why unemployed
61 tab whyunemp, miss
62 tab whyunemp, miss nol
63 //recode whyunemp (0 = .)
64 * why not search
65 tab wnlook, miss
66 tab wnlook, miss nol
67 // 6 = can't arrange childcare, 7 = family responsibilities
68 // 8 = in school or other training
69 // 1 = believe no work available, 2 = couldn't find work, 3 = lack
   necessary schooling/training, 4 = employer think too young/old, 5 =
   other type of discrimination, 10 = transportation problems
70 // 9 = ill-health, physical disability
71 recode wnlook (999 = .)
72 tab wnlook whyunemp, miss // the two variables are in different
   universe
73
74 ** not in labor force (NILF)
75 tab nilfact, miss
76 tab nilfact, miss nol
77 // 4 = taking care of house/family
78 // 3 = in school
79 // 1 = disabled, 2 = ill
80 //recode nilfact (99 = .)
81 tab nilfact empstat, miss
82
83
84 ** summary: whether work status infected by family reasons
85 gen fmrsn = ((whyabsnt >= 7 & whyabsnt <= 9) | (whyptlwk == 121 |
   whyptlwk == 122) | (wnlook == 6 | wnlook == 7) | (nilfact == 4))
86 recode fmrsn (0 = .) if empstat == 0
87 * whether personal, exogenous shocks to work, or school/training
   reasons
88 gen psrsn = ((whyabsnt == 5) | (whyptlwk == 90 | whyptlwk == 111))
```

```
recode psrsn (0 = .) if empstat == 0
gen wkrsn = ((whyabsnt == 10 | whyabsnt == 11) | (whyptlwk == 30 | whyptlwk == 40 | whyptlwk == 10) | ((wnlook >= 1 & wnlook <= 5) | wnlook == 10))
recode wkrsn (0 = .) if empstat == 0
gen strsn = ((whyabsnt == 12) | (whyptlwk == 123) | (wnlook == 8) | (nilfact == 3))
recode strsn (0 = .) if empstat == 0
gen hlrsn = ((whyabsnt == 6) | (whyptlwk == 100 | whyptlwk == 101) | (wnlook == 9) | (nilfact == 1 | nilfact == 2))
recode hlrsn (0 = .) if empstat == 0
* inspect
tab empstat fmrsn, miss
tab1 fmrsn psrsn wkrsn strsn
tab month fmrsn, miss row


/*=====
yearly summary: unit = household-person
=====*/

* number of obs
bysort cpsidp: gen num_month = _N
tab num_month

* sum fmrsn over months
bysort cpsidp: egen sum_fmrsn = total(fmrsn)
* sum other
bysort cpsidp: egen sum_psrsn = total(psrsn)
bysort cpsidp: egen sum_wkrsn = total(wkrsn)
bysort cpsidp: egen sum_strsn = total(strsn)
bysort cpsidp: egen sum_hlrsn = total(hlrsn)
* inspect
tab sum_fmrsn, miss

* unique record each person: keep the last
sort cpsidp year month
bysort cpsidp: keep if _n == _N

* prop of fmrsn == 1 over months
gen prop_fmrsn = sum_fmrsn / num_month

* inspect
tab sum_fmrsn fmrsn, miss
sum prop_fmrsn, detail


/*=====
```

```
133 merge & output
134 =====*/
135
136 preserve
137
138 /* partner */
139
140 merge 1:1 cpsidp using "data/atus_14to19/extracted_partner_cpsidp.dta"
    , keepusing(caseid cpsidp)
141 // 31,850 matched, 388 non-matched from using, 2,032,187 non-matched
    from master
142
143 keep if _merge == 3
144
145 bysort caseid: gen num_obs = _N
146 tab num_obs // unique obs for each caseid
147
148 drop num_obs _merge
149
150 * output
151 save "data/cps_14to19/cps_partner.dta", replace
152
153 * revert to full data
154 restore, preserve
155
156
157 /* self */
158
159 merge 1:1 cpsidp using "data/atus_14to19/extracted_self_cpsidp.dta",
    keepusing(caseid cpsidp)
160 // 62,241 matched, 2,001,796 unmatched from master
161
162 keep if _merge == 3
163
164 bysort caseid: gen num_obs = _N
165 tab num_obs // unique obs for each caseid
166
167 drop num_obs _merge
168
169 * output
170 save "data/cps_14to19/cps_self.dta", replace
171
172 * revert to full data
173 restore, preserve
174
```