regpyhdfe

andy

Jul 13, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

This package provides a semi-convenient way of performing regression with high dimensional fixed effects in python.

To use this, Your data must be in a pandas dataframe. Please see Examples and Tutorial sections for instructions.

1.1 Limitations

1.1.1 Does not work with Weighting (yet).

In order to change address this, one would have to change the underyling model used for regressions. This is currently standard OLS from statsmodels, which does not support weighting. Statsmodels has other linear regression objects which do support weighting, but those have their own issues (Weighted Linear Regression does not support all kinds of weighting, Generalized Linear Models lacks appropriate summary statistics etc.)

1.1.2 Replication of Stata not _quite_ identical when using clustering

By not quite I mean an F-Value of 114.8 v.s. 114.58 etc.:

```
#                            OLS Regression Results
    #␣
↪==============================================================================
    # Dep. Variable:                ttl_exp   R-squared (uncentered):                  ␣
↪0.454
    # Model:                            OLS   Adj. R-squared (uncentered):           -
↪623.342
    # Method:                 Least Squares   F-statistic:                              ␣
↪114.8
    # Date:                Sun, 07 Feb 2021   Prob (F-statistic):                    4.
↪28e-08
    # Time:                        19:42:56   Log-Likelihood:                          -
↪29361.
    # No. Observations:               12568   AIC:                                   5.
↪873e+04
    # Df Residuals:                      11   BIC:                                   5.
↪874e+04
    # Df Model:                           2
    # Covariance Type:              cluster
    # ==============================================================================
    #                  coef    std err          t      P>|t|      [0.025  ␣
```

(continues on next page)

(continued from previous page)

```
↪  0.975]
   # ------------------------------------------------------------------------------
   # wks_ue         0.0307      0.016      1.975      0.074      -0.004       0.065
   # tenure         0.8514      0.066     12.831      0.000       0.705       0.997
   # ==============================================================================
   # Omnibus:                     2467.595   Durbin-Watson:                   1.819
   # Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8034.980
   # Skew:                           0.993   Prob(JB):                         0.00
   # Kurtosis:                       6.376
   # ==============================================================================
   # ==============================================================================
   #
   # V.S.
   #
   # ==============================================================================
   # ==============================================================================
   # HDFE Linear regression                            Number of obs   =     12,568
   # Absorbing 1 HDFE group                            F(   2,     11) =     114.58
   # Statistics robust to heteroskedasticity           Prob > F        =     0.0000
   #                                                   R-squared       =     0.6708
   #                                                   Adj R-squared   =     0.5628
   # Number of clusters (idcode)  =      3,102         Within R-sq.    =     0.4536
   # Number of clusters (year)    =         12         Root MSE        =     2.8836
   #
   #                           (Std. Err. adjusted for 12 clusters in idcode year)
   # ------------------------------------------------------------------------------
   #             |               Robust
   #     ttl_exp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
   # ------------+-----------------------------------------------------------------
   #      wks_ue |   .0306653   .0155436     1.97   0.074    -.0035459    .0648765
   #      tenure |   .8513953   .0663892    12.82   0.000     .7052737    .9975169
   #       _cons |   3.784107   .4974451     7.61   0.000     2.689238    4.878976
   # ------------------------------------------------------------------------------
```

A note of interest is that the fewer degrees of freedom are involved, the worse this discrepancy is, here is an example where clustering leads to just two degrees of freedom in the residuals, so basically as bad as the discrepancy can be:

```
#                            OLS Regression Results
#==============================================================================
#Dep. Variable:                ttl_exp   R-squared (uncentered):                   0.454
#Model:                            OLS   Adj. R-squared (uncentered):          -6866.766
#Method:                 Least Squares   F-statistic:                           1.354e+04
#Date:                Sun, 07 Feb 2021   Prob (F-statistic):                     0.00608
#Time:                        20:32:41   Log-Likelihood:                        -29361.
#No. Observations:               12568   AIC:                                   5.873e+04
#Df Residuals:                       1   BIC:                                   5.874e+04
#Df Model:                           2
#Covariance Type:              cluster
#==============================================================================
#                 coef    std err          t      P>|t|      [0.025      0.975]
#------------------------------------------------------------------------------
#wks_ue         0.0307      0.004      7.483      0.085      -0.021       0.083
```

(continues on next page)

(continued from previous page)

```
#tenure         0.8514      0.013     66.321      0.010       0.688       1.015
#==============================================================================
#Omnibus:                     2467.595   Durbin-Watson:                   1.819
#Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8034.980
#Skew:                           0.993   Prob(JB):                         0.00
#Kurtosis:                       6.376
#==============================================================================
#==============================================================================
#
# v.s.
#
#==============================================================================
#==============================================================================
# HDFE Linear regression                            Number of obs   =     12,568
# Absorbing 1 HDFE group                            F(   2,      1) =    7212.19
# Statistics robust to heteroskedasticity           Prob > F        =     0.0083
#                                                   R-squared       =     0.6708
#                                                   Adj R-squared   =     0.5628
# Number of clusters (union)   =          2        Within R-sq.    =     0.4536
# Number of clusters (idcode)  =      3,102        Root MSE        =     2.8836
#
#                            (Std. Err. adjusted for 2 clusters in union idcode)
# ------------------------------------------------------------------------------
#              |               Robust
#      ttl_exp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
# -------------+----------------------------------------------------------------
#       wks_ue |   .0306653   .0043113     7.11   0.089     -.024115    .0854455
#       tenure |   .8513953   .0132713    64.15   0.010     .6827674    1.020023
#        _cons |   3.784107   .0531894    71.14   0.009     3.108272    4.459942
# ------------------------------------------------------------------------------
#
# Absorbed degrees of freedom:
# -----------------------------------------------------+
#  Absorbed FE | Categories  - Redundant  = Num. Coefs |
# -------------+---------------------------------------|
#       idcode |      3102        3102           0    *|
# -----------------------------------------------------+
```

The T-values are still quite similar, but the F-statistic is completely wrong. It is my understanding that one is not really supposed to use clustering when fewer than 30 clusters are present. This yields:

```
#HDFE Linear regression                           Number of obs   =     12,568
#Absorbing 1 HDFE group                           F(   2,     29) =    5953.94
#Statistics robust to heteroskedasticity          Prob > F        =     0.0000
#                                                 R-squared       =     0.6708
#                                                 Adj R-squared   =     0.5628
#                                                 Within R-sq.    =     0.4536
#Number of clusters (delete_me) =         30      Root MSE        =     2.8836

#                           (Std. Err. adjusted for 30 clusters in delete_me)
#------------------------------------------------------------------------------
#             |               Robust
```

(continues on next page)

(continued from previous page)

```
#     ttl_exp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
#-------------+----------------------------------------------------------------
#      wks_ue |   .0306653   .0054123     5.67   0.000     .0195959    .0417346
#      tenure |   .8513953   .0078384   108.62   0.000     .8353639    .8674267
#       _cons |   3.784107   .0467015    81.03   0.000     3.688592    3.879622
#------------------------------------------------------------------------------

#Absorbed degrees of freedom:
#-----------------------------------------------------+
# Absorbed FE | Categories  - Redundant  = Num. Coefs |
#-------------+---------------------------------------|
#      idcode |      3102           0        3102     |
#-----------------------------------------------------+

#                                OLS Regression Results
#==============================================================================
#Dep. Variable:                ttl_exp   R-squared (uncentered):                   0.454
#Model:                            OLS   Adj. R-squared (uncentered):          -235.820
#Method:                 Least Squares   F-statistic:                             7905.
#Date:                Sun, 07 Feb 2021   Prob (F-statistic):                   2.03e-40
#Time:                        22:00:43   Log-Likelihood:                         -29361.
#No. Observations:               12568   AIC:                                  5.873e+04
#Df Residuals:                      29   BIC:                                  5.874e+04
#Df Model:                           2
#Covariance Type:              cluster
#==============================================================================
#                 coef    std err          t      P>|t|      [0.025      0.975]
#------------------------------------------------------------------------------
#wks_ue         0.0307      0.005      6.529      0.000       0.021       0.040
#tenure         0.8514      0.007    125.159      0.000       0.837       0.865
#==============================================================================
#Omnibus:                     2467.595   Durbin-Watson:                   1.819
#Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8034.980
#Skew:                           0.993   Prob(JB):                         0.00
#Kurtosis:                       6.376
#==============================================================================
```

Oh dear. What about 100?

The simplest fix would be to manually calculate these metrics - residuals and coefficients are all correct, which should give us all the information to calculate appropriately adjusted metrics. This would involve finding the method that Stata packages use, finding a reference for the method and implementing it as specified.

CHAPTER

TWO

REGPYHDFE PACKAGE

2.1 Submodules

2.2 regpyhdfe.regpyhdfe module

class regpyhdfe.regpyhdfe.**Regpyhdfe**(*df*, *target*, *predictors*, *absorb_ids=[]*, *cluster_ids=[]*, *drop_singletons=True*, *intercept=False*)

Bases: `object`

__init__(*df*, *target*, *predictors*, *absorb_ids=[]*, *cluster_ids=[]*, *drop_singletons=True*, *intercept=False*)

Regression wrapper for PyHDFE.

Parameters

- **df** (*pandas Dataframe*) – dataframe containing referenced data which includes target, predictors and absorb and cluster.
- **target** (*string*) – name of target variable - the y in y = X*b + e.
- **predictors** (*string or list of strings*) – names of predictors, the X in y = X*b + e.
- **absorb_ids** (*string or list of strings*) – names of variables to be absorbed for fixed effects.
- **cluster_ids** (*string or list of strings*) – names of variables to be clustered on.
- **drop_singletons** (*bool*) – indicates whether to drop singleton groups. Defaults is True, same as stata. Setting to False is equivalent to passing keepsingletons to reghdfe.

fit()

Generate linear regression coefficients for given data.

The regression will cluster on variables provided during initialization.

Returns

statsmodels.regression.linear_model.RegressionResults.

regpyhdfe.regpyhdfe.**summary**(*self*, *regpyhdfe*, *yname=None*, *xname=None*, *title=None*, *alpha=0.05*)

Summarize the Regression Results.

Parameters

- **yname** (*str, optional*) – Name of endogenous (response) variable. The Default is *y*.
- **xname** (*list[str], optional*) – Names for the exogenous variables. Default is *var_##* for ## in the number of regressors. Must match the number of parameters in the model.

- **title** (*str, optional*) – Title for the top table. If not None, then this replaces the default title.
- **alpha** (*float*) – The significance level for the confidence intervals.

Returns
Instance holding the summary tables and text, which can be printed or converted to various output formats.

Return type
Summary

See also:

statsmodels.iolib.summary.Summary
A class that holds summary results.

2.3 regpyhdfe.utils module

regpyhdfe.utils.**add_intercept**(*X*)

Prepends a column of 1s (an intercept column) to a a 2D numpy array.

Parameters
X (*numpy array*) – 2D numpy array.

Returns
X with an appended column of 1s.

regpyhdfe.utils.**get_np_columns**(*df*, *columns*, *intercept=False*)

Helper used to retreive columns as numpy array.

Parameters

- **df** (*pandas dataframe*) – dataframe containing desired columns
- **columns** (*list of strings*) – list of names of desired columns. Must be a list even if only 1 column is desired.
- **intercept** (*bool*) – set to True if You'd like resulting numpy array to have a column of 1s appended to it.

Returns
2D numpy array with columns of array consisting of feature vectors, i.e. the first column of the result is a numpy vector of the first column named in columns argument.

regpyhdfe.utils.**sklearn_to_df**(*sklearn_dataset*)

Converts (as well as it can) an sklearn dataset to a Pandas dataframe.

Parameters
sklearn_dataset (*sklearn.utils.Bunch*) – this parameter is usually the result of using sklearn to quickly get a dataset, e.g. the object resulting from calling sklearn.load_datasets.load_boston().

Returns
Pandas dataframe df where df['target'] is the target variable in the original dataset.

CHAPTER

THREE

TUTORIAL

3.1 Installation

Should work with just `pip install regpyhdfe`.

3.2 Load in data

We need a pandas dataframe. For the purposes of this example You can go to `https://github.com/lod531/regPyHDFE/blob/main/data/cleaned_nlswork.dta` and download the cleaned nlswork dataset. This dataset contains entries that can be acquired in stata by typing `use nlswork`, except rows containing NA values have already been dropped (hence cleaned_nlswork.dta, rather than nlswork.dta).

Once You have a file, importing the data is as simple as

```
import pandas as pd
# load dataframe
df = pd.read_stata('path/to/cleaned_nlswork.dta')
```

Pandas has other import functions if You have a file in a different format, e.g. `pd.read_csv`.

3.3 Regress

Target is of course the target variable.

Predictors are... Predictors.

absorb_ids are names of variables to be absorbed as high dimensional fixed effects

cluster_ids are names of variables containing cluster information (i.e. if there are N clusters, then each row of a cluster variables contains one of N distinct values.)

```
target = "ln_wage"
predictors = ["hours", "tenure", "ttl_exp"]
absorb_ids = ["year", "idcode"]
cluster_ids = ["year"]

from regpyhdfe import Regpyhdfe
model = Regpyhdfe(df=df, target=target, predictors=predictors,
                    absorb_ids=absorb_ids,
```

(continues on next page)

(continued from previous page)

```
                    cluster_ids=cluster_ids)
results = model.fit()
```

3.4 Examine results

At the time of writing, the `results` object is of type `statsmodels.regression.linear_model.RegressionResults`, documentation for which can be viewed here.

The *statsmodels.regression.linear_model.RegressionResults* ` object has a variety of statistics, but chances are all You're looking is a summary, like so:

```
print(results.summary())
```

The output of that looks like

```
                            OLS Regression Results
==============================================================================================
Dep. Variable:                  ln_wage   R-squared (uncentered):                   0.059
Model:                             OLS   Adj. R-squared (uncentered):          -1313.428
Method:                  Least Squares   F-statistic:                              185.2
Date:                 Thu, 14 Jan 2021   Prob (F-statistic):                    2.09e-08
Time:                         13:21:24   Log-Likelihood:                          766.62
No. Observations:                12568   AIC:                                     -1527.
Df Residuals:                        9   BIC:                                     -1505.
Df Model:                            3
Covariance Type:              cluster
==============================================================================
                 coef    std err          z      P>|z|      [0.025      ␣
↪0.975]
------------------------------------------------------------------------------
hours         -0.0017      0.001     -3.371      0.001      -0.003      -0.001
tenure         0.0109      0.003      3.858      0.000       0.005       0.016
ttl_exp        0.0348      0.003     12.650      0.000       0.029       0.040
==============================================================================
Omnibus:                     1709.175   Durbin-Watson:                   2.171
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            21109.707
Skew:                          -0.174   Prob(JB):                         0.00
Kurtosis:                       9.340   Cond. No.                         6.87
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a␣
↪constant.
[2] Standard Errors are robust to cluster correlation (cluster)
```

And for Your convenience the whole script is

```
import pandas as pd
# load dataframe
df = pd.read_stata('/path/to/cleaned_nlswork.dta')
```

(continues on next page)

(continued from previous page)

```
target = "ln_wage"
predictors = ["hours", "tenure", "ttl_exp"]
absorb_ids = ["year", "idcode"]
cluster_ids = ["year"]

from regpyhdfe import Regpyhdfe
model = Regpyhdfe(df=df, target=target,
    predictors = predictors,
    absorb_ids=absorb_ids,
    cluster_ids=cluster_ids)
results = model.fit()
print(results.summary())
```

CHAPTER

FOUR

EXAMPLES

4.1 Installation

`pip install regpyhdfe`, simple as that.

4.2 Examples

The examples consist of two parts: the python code and the comments.

The python code(s) are minimal examples of a regression. One could simply copy/paste the code, change the dataset and the features of regression and have a working script.

The comments consists of two parts: first part is an identical regression using the reghdfe package in stata. The second part is the output of a corresponding python regression using regPyHDFE. Those comments are there for comparison purposes.

Timing information is trivial and at this time not included - both stata and python run instantly on a laptop CPU.

4.2.1 Using fixed effects only

These examples do not use clustering. As You can see, all that's really needed is a pandas dataframe. Then simply pass in the arguments in appropriate order (or simply pass named arguments. For details on parameters look at the Regpyhdfe object documentation)

```
import pandas as pd
import numpy as np
from regpyhdfe import Regpyhdfe

from sklearn.datasets import load_boston

def sklearn_to_df(sklearn_dataset):
        df = pd.DataFrame(sklearn_dataset.data, columns=sklearn_dataset.feature_names)
        df['target'] = pd.Series(sklearn_dataset.target)
        return df

df = sklearn_to_df(load_boston())

df.to_stata("boston.dta")
# . reghdfe target CRIM ZN INDUS NOX AGE, absorb(CHAS RAD)
```

(continues on next page)

(continued from previous page)

```
# (MWFE estimator converged in 3 iterations)
#
# HDFE Linear regression                            Number of obs   =        506
# Absorbing 2 HDFE groups                           F(   5,    491) =      21.93
#                                                   Prob > F        =     0.0000
#                                                   R-squared       =     0.3887
#                                                   Adj R-squared   =     0.3712
#                                                   Within R-sq.    =     0.1825
#                                                   Root MSE        =     7.2929
#
# ------------------------------------------------------------------------------
#       target |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
# -------------+----------------------------------------------------------------
#         CRIM |  -.2089114   .0491012    -4.25   0.000    -.3053857   -.1124371
#           ZN |   .0679261   .0183051     3.71   0.000       .03196    .1038922
#        INDUS |  -.2279553   .0860909    -2.65   0.008    -.3971074   -.0588033
#          NOX |  -9.424849   5.556005    -1.70   0.090    -20.34133     1.49163
#          AGE |  -.0140739   .0183467    -0.77   0.443    -.0501215    .0219738
#        _cons |   31.24755    2.53596    12.32   0.000     26.26487    36.23022
# ------------------------------------------------------------------------------
#
# Absorbed degrees of freedom:
# -----------------------------------------------------+
#  Absorbed FE | Categories  - Redundant  = Num. Coefs |
# -------------+---------------------------------------|
#         CHAS |         2           0           2     |
#          RAD |         9           1           8     |
# -----------------------------------------------------+

# target~CRIM + ZN + INDUS + NOX + AGE, absorb(CHAS, RAD)
#                                 OLS Regression Results
# ==============================================================================
# Dep. Variable:                 target   R-squared (uncentered):                   0.183
# Model:                            OLS   Adj. R-squared (uncentered):              0.158
# Method:                 Least Squares   F-statistic:                             21.93
# Date:                Mon, 11 Jan 2021   Prob (F-statistic):                   7.57e-20
# Time:                        20:30:53   Log-Likelihood:                         -1715.7
# No. Observations:                 506   AIC:                                      3441.
# Df Residuals:                     491   BIC:                                      3463.
# Df Model:                           5
# Covariance Type:            nonrobust
# ==============================================================================
#                  coef    std err          t      P>|t|      [0.025      0.975]
# ------------------------------------------------------------------------------
# CRIM          -0.2089      0.049     -4.255      0.000      -0.305      -0.112
# ZN             0.0679      0.018      3.711      0.000       0.032       0.104
# INDUS         -0.2280      0.086     -2.648      0.008      -0.397      -0.059
# NOX           -9.4248      5.556     -1.696      0.090     -20.341       1.492
# AGE           -0.0141      0.018     -0.767      0.443      -0.050       0.022
# ==============================================================================
# Omnibus:                      172.457   Durbin-Watson:                   0.904
# Prob(Omnibus):                  0.000   Jarque-Bera (JB):              532.297
```

(continues on next page)

(continued from previous page)

```
# Skew:                          1.621   Prob(JB):                     2.59e-116
# Kurtosis:                      6.839   Cond. No.                         480.
# ==============================================================================
#
# Notes:
# [1] R² is computed without centering (uncentered) since the model does not contain a␣
↪constant.
# [2] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
model = Regpyhdfe(df, 'target', ['CRIM', 'ZN', 'INDUS', 'NOX', 'AGE'], ['CHAS', 'RAD'])
results = model.fit()
print("target~CRIM + ZN + INDUS + NOX + AGE, absorb(CHAS, RAD)")
print(results.summary())
```

```
import pandas as pd
import numpy as np
from regpyhdfe import Regpyhdfe

# details about dataset can be found at https://www.kaggle.com/crawford/80-cereals
df = pd.read_stata('/home/abom/Desktop/regPyHDFE/data/cereal.dta')

# . reghdfe rating fat protein carbo sugars, absorb(shelf)
# (MWFE estimator converged in 1 iterations)
#
# HDFE Linear regression                            Number of obs   =         77
# Absorbing 1 HDFE group                            F(   4,     70) =      54.98
#                                                   Prob > F        =     0.0000
#                                                   R-squared       =     0.7862
#                                                   Adj R-squared   =     0.7679
#                                                   Within R-sq.    =     0.7586
#                                                   Root MSE        =     6.7671
#
# ------------------------------------------------------------------------------
#       rating |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
# -------------+----------------------------------------------------------------
#          fat |  -5.684196   .8801468    -6.46   0.000    -7.439594   -3.928799
#      protein |   3.740386   .8430319     4.44   0.000     2.059012    5.42176
#        carbo |  -.7892276   .2041684    -3.87   0.000    -1.196429   -.3820266
#       sugars |   -2.03286   .2179704    -9.33   0.000    -2.467588   -1.598132
#        _cons |   64.49503    4.92674    13.09   0.000     54.66896    74.3211
# ------------------------------------------------------------------------------
#
# Absorbed degrees of freedom:
# -----------------------------------------------------+
#  Absorbed FE | Categories  - Redundant  = Num. Coefs |
# -------------+---------------------------------------|
#        shelf |         3           0           3     |
# -----------------------------------------------------+

# rating ~ fat + protein + carbo + sugars, absorb(shelf)
#                                OLS Regression Results
# ==============================================================================
```

(continues on next page)

(continued from previous page)

```
# Dep. Variable:                  rating   R-squared (uncentered):                   0.759
# Model:                             OLS   Adj. R-squared (uncentered):              0.734
# Method:                  Least Squares   F-statistic:                              54.98
# Date:                 Mon, 11 Jan 2021   Prob (F-statistic):                    6.89e-21
# Time:                         20:45:37   Log-Likelihood:                         -252.82
# No. Observations:                   77   AIC:                                      513.6
# Df Residuals:                       70   BIC:                                      523.0
# Df Model:                            4
# Covariance Type:             nonrobust
#                  coef    std err          t      P>|t|      [0.025      0.975]
# ------------------------------------------------------------------------------
# fat           -5.6842      0.880     -6.458      0.000      -7.440      -3.929
# protein        3.7404      0.843      4.437      0.000       2.059       5.422
# carbo         -0.7892      0.204     -3.866      0.000      -1.196      -0.382
# sugars        -2.0329      0.218     -9.326      0.000      -2.468      -1.598
# ==============================================================================
# Omnibus:                        5.613   Durbin-Watson:                   1.801
# Prob(Omnibus):                  0.060   Jarque-Bera (JB):                7.673
# Skew:                           0.179   Prob(JB):                       0.0216
# Kurtosis:                       4.504   Cond. No.                         5.84
# ==============================================================================
```

```
residualized = Regpyhdfe(df, 'rating', ['fat', 'protein', 'carbo', 'sugars'], ['shelf'])
results = residualized.fit()
print("rating ~ fat + protein + carbo + sugars, absorb(shelf)")
print(results.summary())


import pandas as pd
import numpy as np
from regpyhdfe import Regpyhdfe
# show variable labels
#pd.read_stata('/home/abom/Desktop/regPyHDFE/nlswork.dta', iterator=True).variable_
↪labels()

# Load data
df = pd.read_stata('/home/abom/Desktop/regPyHDFE/data/cleaned_nlswork.dta')

df['hours_log'] = np.log(df['hours'])

# . reghdfe ln_wage hours_log, absorb(idcode year)
# (dropped 884 singleton observations)
# (MWFE estimator converged in 8 iterations)
#
# HDFE Linear regression                            Number of obs   =     12,568
# Absorbing 2 HDFE groups                           F(   1,   9454) =       0.50
#                                                   Prob > F        =     0.4792
#                                                   R-squared       =     0.7314
#                                                   Adj R-squared   =     0.6430
#                                                   Within R-sq.    =     0.0001
#                                                   Root MSE        =     0.2705
#
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------------------------------------
#      ln_wage |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
# -------------+----------------------------------------------------------------
#    hours_log |  -.0058555   .0082759    -0.71   0.479     -.022078     .010367
#        _cons |   1.736618   .0292873    59.30   0.000     1.679208    1.794027
# ------------------------------------------------------------------------------
#
# Absorbed degrees of freedom:
# -----------------------------------------------------+
#  Absorbed FE | Categories  - Redundant  = Num. Coefs |
# -------------+---------------------------------------|
#       idcode |      3102           0        3102     |
#         year |        12           1          11     |
# -----------------------------------------------------+

# ln_wage ~ hours_log, absorb(idcode, year)
#                                OLS Regression Results
# ==============================================================================
# Dep. Variable:                ln_wage   R-squared (uncentered):                   0.000
# Model:                            OLS   Adj. R-squared (uncentered):            -0.329
# Method:                 Least Squares   F-statistic:                            0.5006
# Date:                Mon, 11 Jan 2021   Prob (F-statistic):                     0.479
# Time:                        21:07:22   Log-Likelihood:                         386.59
# No. Observations:               12568   AIC:                                    -771.2
# Df Residuals:                    9454   BIC:                                    -763.7
# Df Model:                           1
# Covariance Type:            nonrobust
# ==============================================================================
#                  coef    std err          t      P>|t|      [0.025      0.975]
# ------------------------------------------------------------------------------
# hours_log     -0.0059      0.008     -0.708      0.479      -0.022       0.010
# ==============================================================================
# Omnibus:                     1617.122   Durbin-Watson:                   2.143
# Prob(Omnibus):                  0.000   Jarque-Bera (JB):            16984.817
# Skew:                          -0.215   Prob(JB):                         0.00
# Kurtosis:                       8.679   Cond. No.                         1.00
# ==============================================================================


model = Regpyhdfe(df, "ln_wage", "hours_log", ["idcode", "year"])
results = model.fit()
print("ln_wage ~ hours_log, absorb(idcode, year)")
print(results.summary())
```

4.2.2 Clustering:

Very similar to standard regression, simply add a clustering_ids parameter to the parameter list passed to Regpyhdfe.

```
from regpyhdfe import Regpyhdfe
import pandas as pd
import numpy as np

df = pd.read_stata('data/cleaned_nlswork.dta')
df['hours_log'] = np.log(df['hours'])
regpyhdfe = Regpyhdfe(df=df,
    target='ttl_exp',
    predictors=['wks_ue', 'tenure'],
    ids=['idcode'],
    cluster_ids=['year', 'idcode'])

# (dropped 884 singleton observations)
# (MWFE estimator converged in 1 iterations)
#
# HDFE Linear regression                            Number of obs   =     12,568
# Absorbing 1 HDFE group                            F(   2,     11) =     114.58
# Statistics robust to heteroskedasticity           Prob > F        =     0.0000
#                                                   R-squared       =     0.6708
#                                                   Adj R-squared   =     0.5628
# Number of clusters (year)    =         12         Within R-sq.    =     0.4536
# Number of clusters (idcode)  =      3,102         Root MSE        =     2.8836
#
#                            (Std. Err. adjusted for 12 clusters in year idcode)
# ------------------------------------------------------------------------------
#              |               Robust
#      ttl_exp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
# -------------+----------------------------------------------------------------
#       wks_ue |   .0306653   .0155436     1.97   0.074    -.0035459    .0648765
#       tenure |   .8513953   .0663892    12.82   0.000     .7052737    .9975169
#        _cons |   3.784107   .4974451     7.61   0.000     2.689238    4.878976
# ------------------------------------------------------------------------------
#
# Absorbed degrees of freedom:
# -----------------------------------------------------+
#  Absorbed FE | Categories  - Redundant  = Num. Coefs |
# -------------+---------------------------------------|
#       idcode |      3102        3102           0    *|
# -----------------------------------------------------+
# * = FE nested within cluster; treated as redundant for DoF computation

#                                 OLS Regression Results
# ==============================================================================
# Dep. Variable:                ttl_exp   R-squared (uncentered):                   0.454
# Model:                            OLS   Adj. R-squared (uncentered):          -623.342
# Method:                 Least Squares   F-statistic:                             114.8
# Date:                Mon, 11 Jan 2021   Prob (F-statistic):                   4.28e-08
# Time:                        21:35:07   Log-Likelihood:                         -29361.
# No. Observations:               12568   AIC:                                  5.873e+04
```

(continues on next page)

(continued from previous page)

```
# Df Residuals:                     11   BIC:                                5.874e+04
# Df Model:                          2
# Covariance Type:              cluster
# ==============================================================================
#                  coef    std err          z      P>|z|      [0.025      0.975]
# ------------------------------------------------------------------------------
# wks_ue         0.0307      0.016      1.975      0.048       0.000       0.061
# tenure         0.8514      0.066     12.831      0.000       0.721       0.981
# ==============================================================================
# Omnibus:                     2467.595   Durbin-Watson:                   1.819
# Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8034.980
# Skew:                           0.993   Prob(JB):                         0.00
# Kurtosis:                       6.376   Cond. No.                         2.06
# ==============================================================================

results = regpyhdfe.fit()
print(results.summary())
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

Symbols

A

F

G

M

R

S