2017-05-09-ff-pdbp-paper_replication

October 22, 2017

0.1 1. Data load and cleansing

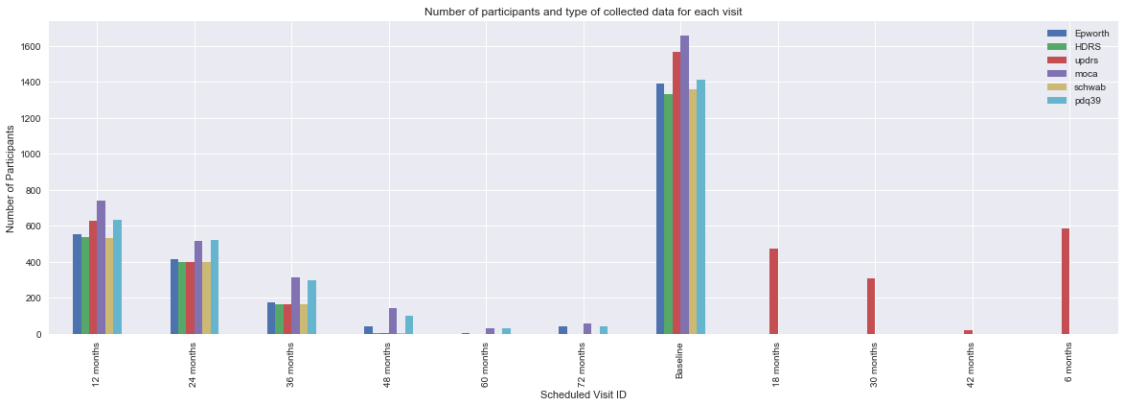
In [2]:
```
# loading libraries and settings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

## loading data and selecting the necessary columns
cols = {} # column names
cols["epworth"] = ["EpworthSleepinessScale.Required Fields.GUID", "EpworthS
cols["HDRS"] = ["HDRS.Required Fields.VisitTypPDBP", "HDRS.Required Fields.
cols["updrs"] = ["MDS_UPDRS.Required Fields.VisitTypPDBP", "MDS_UPDRS.Requi
cols["moca"] = ["MoCA.Required Fields.VisitTypPDBP", "MoCA.Required Fields.
cols["schwab"] = ["ModSchwabAndEnglandScale.Required Fields.VisitTypPDBP",
# cols["pdq39"] = ["PDQ39.Required Fields.VisitTypPDBP", "PDQ39.Required Fi
cols["pdq39"] = ["PDQ39.Required Fields.VisitTypPDBP", "PDQ39.Required Fiel
cols["upsit"] = ["UnivOfPennSmellIdenTest.Required Fields.VisitTypPDBP", "U
cols["demographics"] = ["Demographics.Required Fields.GUID", "Demographics.
cols["biosample"] = ["BiosampleCatalogV5.Subject Information.GUID", "Biosam
cols["family_history_col"] = ["FamilyHistory.Required Fields.GUID", "Family

## data load
epworth = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-4
epworth = epworth.rename(columns={col: col.split('.')[-1] for col in epwort
epworth = epworth.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'

HDRS = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47/c
HDRS = HDRS.rename(columns={col: col.split('.')[-1] for col in HDRS.columns
HDRS = HDRS.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})

updrs = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47/
updrs = updrs.rename(columns={col: col.split('.')[-1] for col in updrs.colu
updrs = updrs.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})
```

```
moca = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47/c
moca = moca.rename(columns={col: col.split('.')[-1] for col in moca.columns
moca = moca.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})

schwab = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47
schwab = schwab.rename(columns={col: col.split('.')[-1] for col in schwab.c
schwab = schwab.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})

pdq39 = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47/
pdq39 = pdq39.rename(columns={col: col.split('.')[-1] for col in pdq39.colu
pdq39 = pdq39.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})

upsit = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05-47/
upsit = upsit.rename(columns={col: col.split('.')[-1] for col in upsit.colu
upsit = upsit.rename(columns={'VisitTypPDBP':'EVENT_ID', 'GUID':'PATNO'})

# Patient info
demographics = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15
demographics = demographics.rename(columns={col: col.split('.')[-1] for col
demographics = demographics.rename(columns={'GUID':'PATNO', 'AgeYrs':'Age',
demographics = demographics.drop_duplicates(subset='PATNO', keep='last')

pdbp_info = demographics

biosample = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-26T15-05
biosample = biosample.rename(columns={col: col.split('.')[-1] for col in bi
biosample = biosample.rename(columns={'GUID':'PATNO', 'InclusnXclusnCntrlIn
biosample = biosample.drop_duplicates(subset='PATNO', keep='last')

pdbp_info = pd.concat([pdbp_info.set_index('PATNO'), biosample.set_index('P

family_history_pd = pd.read_csv("../PDBP replication/BL-102616-All_2016-10-
family_history_pd = family_history_pd.rename(columns={col: col.split('.')[-
family_history_pd = family_history_pd.rename(columns={'GUID':'PATNO'})
family_history_pd = family_history_pd.dropna(axis=0).drop_duplicates(subset
```

0.2 2. Explore data

In [4]:
```
a = epworth.reset_index().groupby('EVENT_ID').size().reset_index().rename(c
b = HDRS.reset_index().groupby('EVENT_ID').size().reset_index().rename(colu
c = updrs.reset_index().groupby('EVENT_ID').size().reset_index().rename(col
d = moca.reset_index().groupby('EVENT_ID').size().reset_index().rename(colu
e = schwab.reset_index().groupby('EVENT_ID').size().reset_index().rename(co
f = pdq39.reset_index().groupby('EVENT_ID').size().reset_index().rename(col
g = upsit.reset_index().groupby('EVENT_ID').size().reset_index().rename(col

plot = a.merge(b, on='EVENT_ID', how='outer').merge(c, on='EVENT_ID', how='
plot.set_ylabel("Number of Participants");
```

```
plot.set_xlabel("Scheduled Visit ID");
```



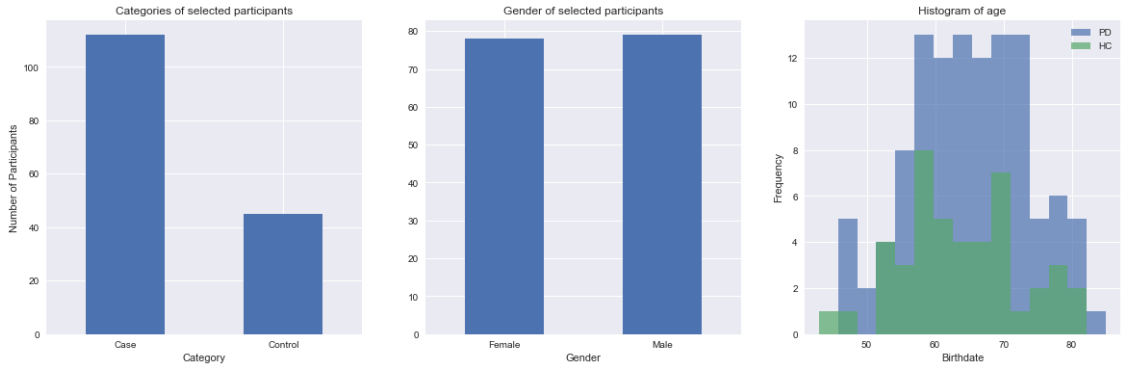
0.2.1 2.1. Selecct the ones in the study for 36 months

```
In [5]: # datasets and visits of interest, limiting the progression to 36 months (v
        datasets_of_interest = ['epworth', 'HDRS', 'updrs', 'moca', 'schwab', 'pdq3
        visits_of_interest = ['Baseline', '12 months', '24 months', '36 months']#,
        last_visit = visits_of_interest[-1]

        # selecting participants with data from BL to last_visit
        dataset_first_noindx = eval(datasets_of_interest[0]).reset_index()
        patno_filtered_visited = dataset_first_noindx[ dataset_first_noindx.EVENT_I

        for dataset in datasets_of_interest[1:]:
            dataset_noindx = eval(dataset).reset_index()
            temp_patno = dataset_noindx[ dataset_noindx.EVENT_ID == last_visit ]['E

            patno_filtered_visited = patno_filtered_visited[ patno_filtered_visited

        # constructing the data_visits dictionary with all the information
        data_visits = {}
        # status_o = status[status.index.isin(patno_filtered_visited)].ENROLL_CAT
        # screening_o = screening[screening.index.isin(patno_filtered_visited)]
        data_visits["info"] = pdbp_info[pdbp_info.index.isin(patno_filtered_visited

        for dataset in datasets_of_interest:
            dataset_noindx = eval(dataset).reset_index()
            data_visits[dataset] = dataset_noindx[ dataset_noindx['PATNO'].isin(pat
```

Visualizing selected cohort:

```
In [6]: plt.figure(1, figsize=(20, 6))
```

```
# plot the categories distribution
plt.subplot(1,3,1)
plot_1 = data_visits["info"].CASE.reset_index().groupby("CASE").size().plot
plot_1.set_ylabel("Number of Participants"); plot_1.set_xlabel("Category");

# plot the gender distribution
plt.subplot(1,3,2)
data_visits["info"].Gender.reset_index().groupby("Gender").size().plot(kind

# plot the age histogram
plt.subplot(1,3,3)
hc_birthdt = data_visits["info"].Age.reset_index()[data_visits["info"].rese
pd_birthdt = data_visits["info"].Age.reset_index()[data_visits["info"].rese
concat_birthdt = pd.concat([pd_birthdt.PD, hc_birthdt.HC], axis=1)
plot_3 = concat_birthdt.plot(kind='hist', title="Histogram of age", alpha=0
plot_3.set_xlabel("Birthdate");
```



0.2.2 2.2. Stats for Table 1 paper

```
In [7]: data_visits["info"].CASE.reset_index().groupby("CASE").size()

Out[7]: CASE
        Case       112
        Control     45
        dtype: int64

In [8]: data_visits["info"].Gender.reset_index()[data_visits["info"].reset_index()

Out[8]: Gender
        Female    25
        Male      20
        dtype: int64

In [9]: data_visits["info"].Age.reset_index()[data_visits["info"].reset_index()['CA
```

```
Out[9]:              Age
        count  45.000000
        mean   63.711111
        std     9.114480
        min    43.000000
        25%    57.000000
        50%    63.000000
        75%    69.000000
        max    82.000000
```

```
In [10]: list_case_control = data_visits["info"].reset_index().PATNO[data_visits["i

         selected_family_history_pd = family_history_pd.set_index(['PATNO']).loc[li
         selected_family_history_pd[selected_family_history_pd.FamHistMedclCondInd.
```

```
Out[10]: FamHistMedclCondInd    12
         dtype: int64
```

0.3 3. Vectorizing time-series data into one series

```
In [11]: t1 = data_visits["epworth"].drop(['index'],axis=1).set_index(['PATNO','EVE
         t2 = data_visits['HDRS'].drop(['index'],axis=1).set_index(['PATNO','EVENT_
         t3 = data_visits['updrs'].drop(['index', 'MDSUPDRSLstLDOPADoseTm', 'MDSUPD
         t4 = data_visits['moca'].drop(['MOCA_DelydRecalOptnlMultChoice', 'MOCA_Del
         t5 = data_visits['schwab'].drop(['index'],axis=1).set_index(['PATNO','EVEN
         t6 = data_visits['pdq39'].drop(['index'],axis=1).set_index(['PATNO','EVENT
         t7 = data_visits['upsit'].drop(['index'],axis=1).set_index(['PATNO','EVENT

         # plt.pcolor(t6.isnull())
         M = pd.concat([t1, t2, t3, t4, t5, t6, t7], axis=1).interpolate(method='li
```

0.4 4. Normalization

```
In [12]: # normalize values based on z-score
         data_visits_zs = {}
         for i in range(1, len(datasets_of_interest) + 1):
             dataset = 't' + str(i)
             dataset_columns = eval(dataset).columns.levels[0][0:-1]

             # create an empty dataframe: t16_zs = pd.DataFrame(index=t16.index, co
             data_visits_zs[dataset] = pd.DataFrame(index=eval(dataset).index, colu

             for col in dataset_columns:
                 # assign normalized: t16_zs['a_state'] = (t16['a_state'] - t16['a_
                 data_visits_zs[dataset][col] = (eval(dataset)[col] - eval(dataset)

         data_visits_zs['t7']
```

```
        # construct full M
        M_zs = pd.concat([data_visits_zs['t1'], data_visits_zs['t2'], data_visits_
                          data_visits_zs['t5'], data_visits_zs['t6'] , data_visits
```

In [13]:
```
        # normalize values based on min-max
        data_visits_minmax = {}
        minmax_min = {}
        minmax_max = {}
        for i in range(1, len(datasets_of_interest) + 1):
            dataset = 't' + str(i)
            dataset_columns = eval(dataset).columns.levels[0][0:-1]

            # create an empty dataframe: t16_zs = pd.DataFrame(index=t16.index, co
            data_visits_minmax[dataset] = pd.DataFrame(index=eval(dataset).index,
            minmax_min[dataset] = pd.DataFrame(index=[1], columns=eval(dataset).co
            minmax_max[dataset] = pd.DataFrame(index=[1], columns=eval(dataset).co

            for col in dataset_columns:
                # assign normalized: t16_zs['a_state'] = (t16['a_state'] - t16['a_
                data_visits_minmax[dataset][col] = (eval(dataset)[col] - eval(data
                minmax_min[dataset][col] = eval(dataset)[col].min().min()
                minmax_max[dataset][col] = eval(dataset)[col].max().max()

        data_visits_minmax['t7'].min()

        # construct full M
        M_minmax = pd.concat([data_visits_minmax['t1'], data_visits_minmax['t2'],
                          data_visits_minmax['t5'], data_visits_minmax['t6'] , dat

        # construct min array for export
        M_minmax_min = pd.concat([minmax_min['t1'], minmax_min['t2'], minmax_min['
                          minmax_min['t5'], minmax_min['t6'] , minmax_min['t7']],

        M_minmax_max = pd.concat([minmax_max['t1'], minmax_max['t2'], minmax_max['
                          minmax_max['t5'], minmax_max['t6'] , minmax_max['t7']],
```

0.5 5. Dimension reduction to progression space

In [73]:
```
        M_chosen = M_minmax.drop(['index'],axis=1) #M_zs #M # choosing Min-Max nor

        M_cat = pd.concat([M_chosen, data_visits["info"].CASE], axis=1) # labels c

        M_W_columns = ['PCA_1', 'PCA_2', 'PCA_3', 'ICA_1', 'ICA_2', 'NMF_2_1', 'NM
                       'NMF_3_1', 'NMF_3_2', 'NMF_3_3',
                      'NMF_4_1', 'NMF_4_2', 'NMF_4_3', 'NMF_4_4']
        M_W = pd.DataFrame(index=M_chosen.index, columns=M_W_columns)

        # PCA
```

```
        from sklearn.decomposition import PCA as sklearnPCA
        model_pca = sklearnPCA(n_components=3)
        M_W[['PCA_1', 'PCA_2', 'PCA_3']] = model_pca.fit_transform(M_chosen)

        # NMF
        from sklearn import decomposition
        model_NMF = decomposition.NMF(n_components=2, init='nndsvda', max_iter=200
        model_NMF3 = decomposition.NMF(n_components=3, init='nndsvda', max_iter=20
        model_NMF4 = decomposition.NMF(n_components=4, init='nndsvda', max_iter=20
        M_W[['NMF_2_1', 'NMF_2_2']] = model_NMF.fit_transform(M_chosen)
        M_W[['NMF_3_1', 'NMF_3_2', 'NMF_3_3']] = model_NMF3.fit_transform(M_choser
        M_W[['NMF_4_1', 'NMF_4_2', 'NMF_4_3', 'NMF_4_4']] = model_NMF4.fit_transfo

        # ICA
        model_ICA = decomposition.FastICA(n_components=2)
        M_W[['ICA_1', 'ICA_2']] = model_ICA.fit_transform(M_chosen)
```

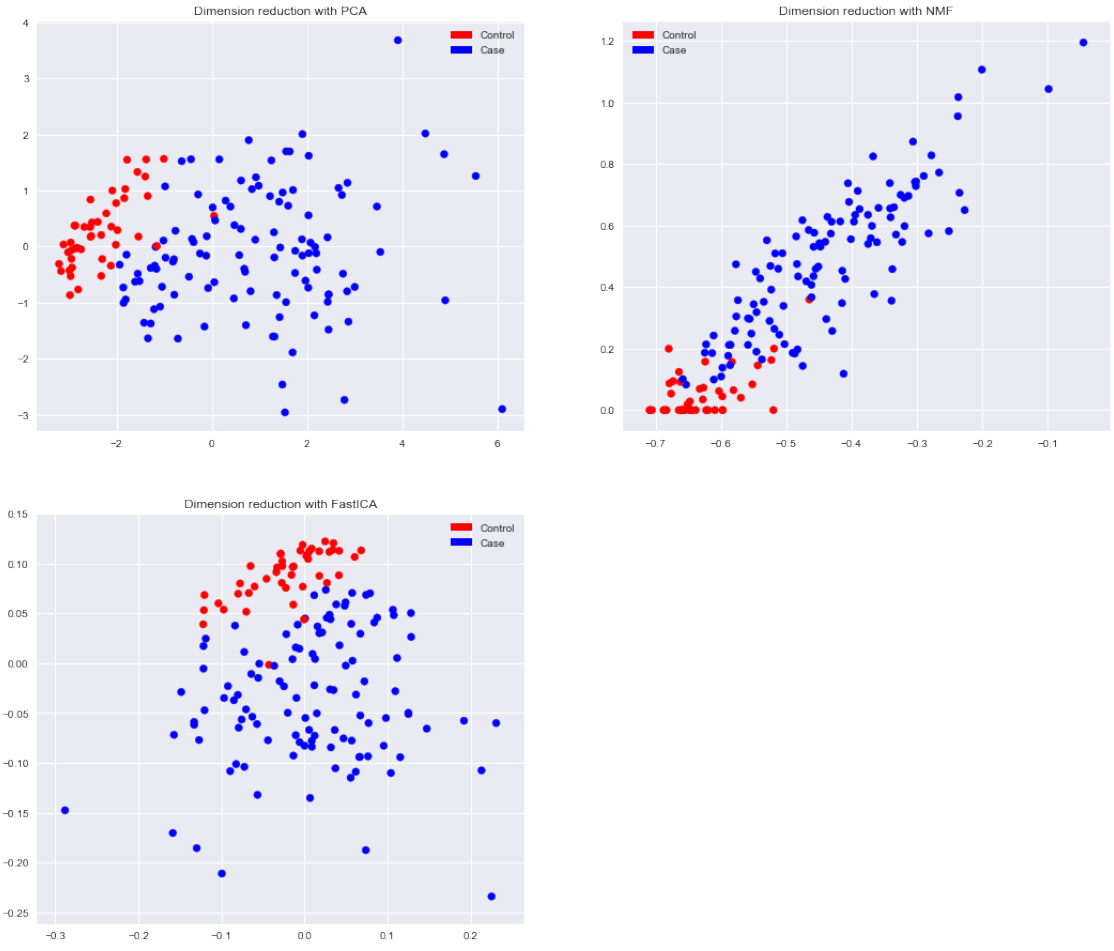
```
/Users/faraz/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:1: Performa
  if __name__ == '__main__':
```

0.6 6. Visualization of progression space in 2D

```
In [49]: # plot the dimension reduction color makrked with participants' "categorie
         %matplotlib inline
         plt.figure(1, figsize=(18, 24))

         ## PCA
         plt.subplot(3,2,1)
         colors_categories = pd.concat([M_W, data_visits["info"].CASE], axis=1).CAS

         plot_1 = plt.scatter(M_W[['PCA_1']], M_W[['PCA_2']], c = colors_categories

         p1 = plt.Rectangle((0, 0), 0.1, 0.1, fc='red')
         p2 = plt.Rectangle((0, 0), 0.1, 0.1, fc='blue')
         plt.legend((p1, p2), ('Control', 'Case'), loc='best');
         plt.title('Dimension reduction with PCA')


         ## NMF
         plt.subplot(3,2,2)

         plot_1 = plt.scatter(-M_W[['NMF_2_1']], M_W[['NMF_2_2']], c = colors_categ

         p1 = plt.Rectangle((0, 0), 0.1, 0.1, fc='red')
         p2 = plt.Rectangle((0, 0), 0.1, 0.1, fc='blue')
         plt.legend((p1, p2), ('Control', 'Case'), loc='best');
         plt.title('Dimension reduction with NMF')
```

```
        # plt.plot([0.3,0.6], [0,0.5])
        # plt.plot([0.2,0.5], [0.1,0.6])

        ## ICA
        plt.subplot(3,2,3)

        plot_1 = plt.scatter(M_W[['ICA_1']], M_W[['ICA_2']], c = colors_categories

        p1 = plt.Rectangle((0, 0), 0.1, 0.1, fc='red')
        p2 = plt.Rectangle((0, 0), 0.1, 0.1, fc='blue')
        plt.legend((p1, p2), ('Control', 'Case'), loc='best');
        plt.title('Dimension reduction with FastICA')
```

Out[49]: <matplotlib.text.Text at 0x1191a0240>



0.7 7. Visualization of progression space in 3D

In [207]:
```
        # NMF 3 color makrked with participants' "categories"
        %matplotlib notebook
```

```
            from mpl_toolkits.mplot3d import Axes3D
            ax = plt.axes(projection='3d')

            # plots all
            # ax.scatter(M_W[['NMF_3_1']], M_W[['NMF_3_2']], M_W[['NMF_3_3']], c = co
            # plt.legend((p1, p2, p3, p4, p5, p6, p7), ('HC', 'SWEDD', 'PD', 'REGPD',

            # plots onlt PD and HC
            M_W_PD_HC = M_W.loc[ M_cat.CASE.isin(['Control', 'Case']) ]
            ax.scatter(M_W_PD_HC[['NMF_3_1']], M_W_PD_HC[['NMF_3_2']], M_W_PD_HC[['NM
            plt.legend((p2, p1), ('PD patient','Healthy Control' ), loc='best');

            ax.grid(True)
            ax.set_xticklabels([])
            ax.set_yticklabels([])
            ax.set_zticklabels([])
            ax.set_xlabel('Cognitive impairment')
            ax.set_ylabel('Movement disorder')
            ax.set_zlabel('Sleep disorder')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[207]: <matplotlib.text.Text at 0x1203a68d0>
```

0.8 8. Bring in PPMI GMM

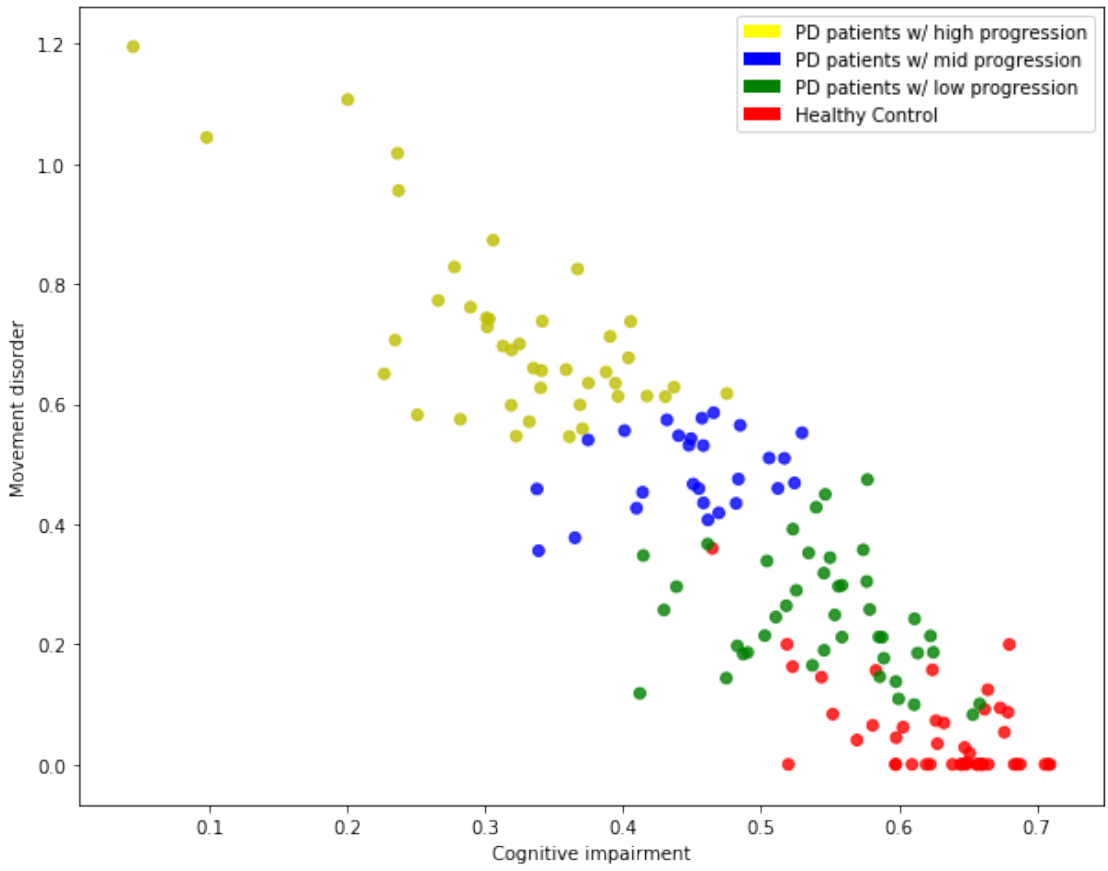
```
In [55]: # label HC as HC
         M_label_columns = ['GMM']
         M_label = pd.DataFrame(index=M_chosen.index, columns=M_label_columns)

         M_label[ M_cat.CASE == "Control" ] = 'Control'


         %store -r model_gmm
         M_W_PD_chosen = M_W_PD_HC[M_cat.CASE == "Case"]

         M_label.loc[ M_cat.CASE == "Case", 'GMM' ] = model_gmm.predict(M_W_PD_chos
         M_label.replace([1,0,2],['PD_l', 'PD_m', 'PD_h'], inplace=True)
         M_label.head()
```

```
Out[55]:                GMM
         PATNO
         PDAA503EF5     PD_h
         PDAB411CTU  Control
         PDAC066ZP4  Control
```

```
PDAG669PAY  Control
PDBB602VX6     PD_1
```

In [210]:
```
# plot only predicted PD
%matplotlib inline

plt.figure(1, figsize=(10, 8))
colors = ['r' if i=='Control' else 'g' if i=='PD_1' else 'b' if i=='PD_m'
ax = plt.scatter(M_W_PD_HC[['NMF_2_1']], M_W_PD_HC[['NMF_2_2']], c=colors
# ax.scatter(M_PD_HC_gmm_chosen[[0]], M_PD_HC_gmm_chosen[[1]], c=colors,
p1 = plt.Rectangle((0, 0), 0.1, 0.1, fc='yellow')
p2 = plt.Rectangle((0, 0), 0.1, 0.1, fc='blue')
p3 = plt.Rectangle((0, 0), 0.1, 0.1, fc='green')
p4 = plt.Rectangle((0, 0), 0.1, 0.1, fc='red')
plt.legend((p1, p2, p3, p4), ('PD patients w/ high progression', 'PD pati
# plt.title('Adding HC to the labeld PD')
# plt.axes().set_xticklabels([])
# plt.axes().set_yticklabels([])
plt.axes().set_xlabel('Cognitive impairment')
plt.axes().set_ylabel('Movement disorder')
# ax.set_zlabel('Sleep disorder')
```

Out[210]: <matplotlib.text.Text at 0x11f9da860>



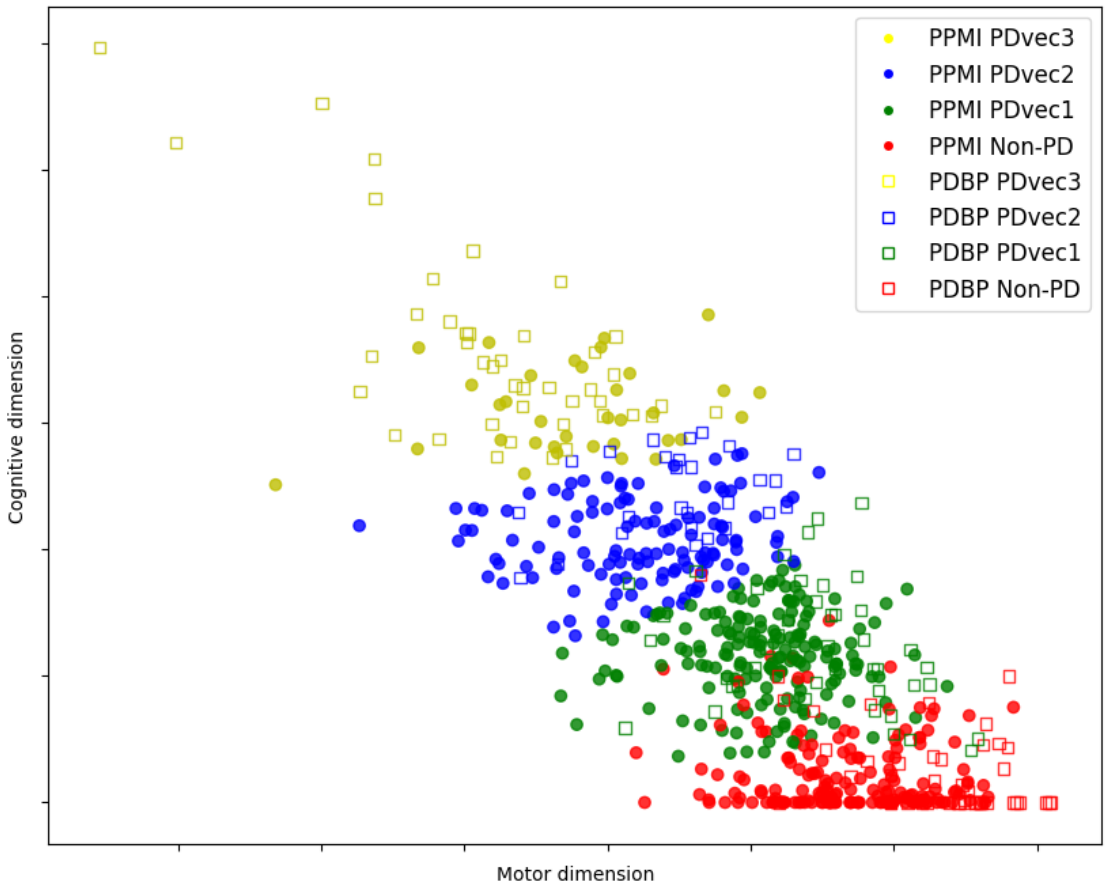
0.9 8. Bring in PPMI dataset

In [208]:
```
%store -r M_PD_HC_gmm_chosen
M_label_PD_HC = pd.read_csv("../M_label_PD_HC.csv", index_col='PATNO')
M_nmf_PD_HC = pd.read_csv("../M_nmf_PD_HC.csv", index_col='PATNO')
M_nmf_PD_HC.columns = ['NMF_3_1', 'NMF_3_2', 'NMF_3_3']

ppmi_pdbp_nmf3d = pd.concat([M_nmf_PD_HC, M_W_PD_HC[['NMF_3_1', 'NMF_3_2'
ppmi_pdbp_nmf2d = pd.concat([M_PD_HC_gmm_chosen, M_W_PD_HC[['NMF_2_1', 'N

ppmi_pdbp_label = pd.concat([M_label_PD_HC, M_cat.CASE.to_frame("GMM")])
ppmi_pdbp_label_gmm = pd.concat([M_label_PD_HC, M_label.replace(['PD_l',
```

In [209]:
```
# plot only predicted PD
%matplotlib inline
import matplotlib as mpl
mpl.style.use('default')

plt.figure(1, figsize=(10, 8))
colors_ppmi = ['r' if i in ['HC','Control'] else 'g' if i in ['PD_l', 'PD
          else 'b' if i in ['PD_m', 'PD_m_pdbp'] else
          'y' if i in ['PD_h', 'PD_h_pdbp'] else 'black' for i in M_label

colors_pdbp = ['r' if i in ['HC','Control'] else 'g' if i in ['PD_l', 'PD
          else 'b' if i in ['PD_m', 'PD_m_pdbp'] else
          'y' if i in ['PD_h', 'PD_h_pdbp'] else 'black' for i in M_label

# ax = plt.scatter(-ppmi_pdbp_nmf2d[['NMF_2_1']], ppmi_pdbp_nmf2d[['NMF_2
ax = plt.scatter(M_PD_HC_gmm_chosen[['NMF_2_1']], M_PD_HC_gmm_chosen[['NM
ax = plt.scatter(-0.1+M_W_PD_HC[['NMF_2_1']], M_W_PD_HC[['NMF_2_2']], fac

# ax.scatter(M_PD_HC_gmm_chosen[[0]], M_PD_HC_gmm_chosen[[1]], c=colors,
p1 = plt.Line2D(range(2), range(2), color="white", marker='o', markerface
p2 = plt.Line2D(range(2), range(2), color="white", marker='o', markerface
p3 = plt.Line2D(range(2), range(2), color="white", marker='o', markerface
p4 = plt.Line2D(range(2), range(2), color="white", marker='o', markerface
p5 = plt.Line2D(range(2), range(2), color="white", marker='s', markerface
p6 = plt.Line2D(range(2), range(2), color="white", marker='s', markerface
p7 = plt.Line2D(range(2), range(2), color="white", marker='s', markerface
p8 = plt.Line2D(range(2), range(2), color="white", marker='s', markerface
plt.legend((p1, p2, p3, p4, p5, p6, p7, p8),
           ('PPMI PDvec3', 'PPMI PDvec2', 'PPMI PDvec1', 'PPMI Non-PD', '
# plt.title('Adding HC to the labeld PD')
plt.axes().set_xticklabels([])
plt.axes().set_yticklabels([])
```

```
plt.axes().set_xlabel('Motor dimension')
plt.axes().set_ylabel('Cognitive dimension')
# ax.set_zlabel('Sleep disorder')
```

Out[209]: <matplotlib.text.Text at 0x110741e80>



In [195]:
```
# NMF 3 color makrked with participants' "categories"
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
ax = plt.axes(projection='3d')

# plots all
# ax.scatter(M_W[['NMF_3_1']], M_W[['NMF_3_2']], M_W[['NMF_3_3']], c = co
# plt.legend((p1, p2, p3, p4, p5, p6, p7), ('HC', 'SWEDD', 'PD', 'REGPD',

# plots onlt PD and HC
colors_categories = ppmi_pdbp_label.GMM.replace(['HC', 'PD_l', 'PD_m', 'P
ax.scatter(ppmi_pdbp_nmf[['NMF_3_1']], ppmi_pdbp_nmf[['NMF_3_2']], ppmi_p
```

```
            # p1 = plt.Rectangle((0, 0), 0.1, 0.1, fc='yellow')
            # p2 = plt.Rectangle((0, 0), 0.1, 0.1, fc='blue')
            # p3 = plt.Rectangle((0, 0), 0.1, 0.1, fc='green')
            # p4 = plt.Rectangle((0, 0), 0.1, 0.1, fc='red')
            # plt.legend((p1, p2, p3, p4), ('PD patients w/ high progression', 'PD pa

            ax.grid(True)
            ax.set_xticklabels([])
            ax.set_yticklabels([])
            ax.set_zticklabels([])
            ax.set_xlabel('Cognitive impairment')
            ax.set_ylabel('Movement disorder')
            ax.set_zlabel('Sleep disorder')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

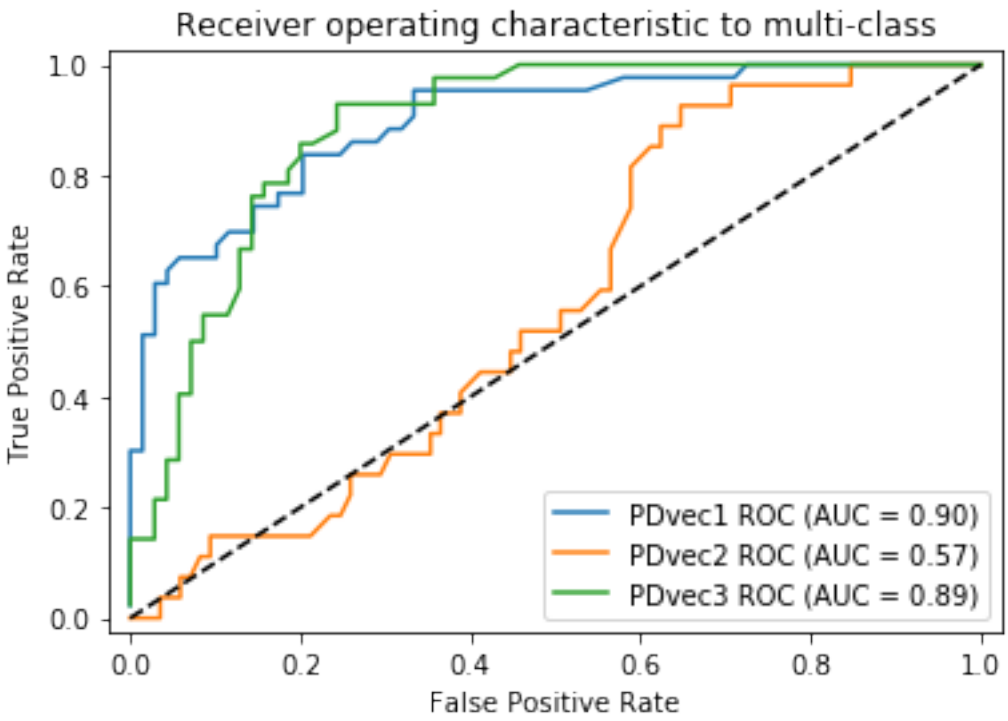
```
Out[195]: <matplotlib.text.Text at 0x1221b59b0>
```

0.10 9. Bringing in PPMI prediction model on NMF3D

```
In [211]: from sklearn.metrics import accuracy_score
          from sklearn import metrics

          %store -r clf_replication

          M_W_baseline = pd.DataFrame(index=M_chosen.index, columns=M_W_columns)

          M_W_baseline[['NMF_2_1', 'NMF_2_2']] = model_NMF.fit_transform(M_chosen.i
          M_W_baseline[['NMF_3_1', 'NMF_3_2', 'NMF_3_3']] = model_NMF3.fit_transfor
          M_W_PD_HC_baseline = M_W_baseline.loc[ M_cat.CASE.isin(['Control', 'Case'
          M_W_PD_HC_baseline[['NMF_3_1']] = -0.0+M_W_PD_HC_baseline[['NMF_3_1']] #r


          X_test = M_W_PD_HC_baseline.loc[M_cat.CASE == "Case", ['NMF_3_1', 'NMF_3_
          Y_test = M_label[ M_cat.CASE == "Case" ].replace(['HC', 'PD_l', 'PD_m', '


          # print(accuracy_score(Y_test, clf_replication.predict(X_test)))
```

```
In [180]: # plot ROC
          # ref: http://scikit-learn.org/stable/auto_examples/model_selection/plot_
          %matplotlib inline
```

```
from sklearn import metrics
n_classes = len(Y_test['GMM'].unique())
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    preds = clf_replication.predict_proba(X_test)[:,i]
    label = [1 if int(j) == (i+2) else 0 for j in Y_test['GMM']] #2,3,4
    fpr[i], tpr[i], _ = metrics.roc_curve(label, preds)
    roc_auc[i] = metrics.auc(fpr[i], tpr[i])
    class_name = 'PDvec1' if i+2==2 else 'PDvec2' if i+2==3 else 'PDvec3'
    plt.plot(fpr[i], tpr[i], label='{0} ROC (AUC = {1:0.2f})'
                                   ''.format(class_name, roc_auc[i]))

# Compute micro-average ROC curve and ROC area
# y_score = lr.predict(X_test)
# fpr["micro"], tpr["micro"], _ = metrics.roc_curve(Y_test.ravel(), y_sco
# roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

plt.figure(1, figsize=(9, 5))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.025, 1.025])
plt.ylim([-0.025, 1.025])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```


Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
PDvec1 ROC (AUC = 0.90)
PDvec2 ROC (AUC = 0.57)
PDvec3 ROC (AUC = 0.89)