nhanes_bp

February 24, 2020

1 Multilevel analysis of NHANES blood pressure data

In this notebook, we fit mixed models to blood pressure data from NHANES. The study has data for two blood pressure measurement types (systolic BP and diastolic BP), with up to 4 repeated measures per subject for each type.

https://wwwn.cdc.gov/Nchs/Nhanes/2011-2012/DEMO_G.XPT https://wwwn.cdc.gov/Nchs/Nhanes/2011-2012/BPX_G.XPT https://wwwn.cdc.gov/Nchs/Nhanes/2011-2012/BMX_G.XPT

[1]:

```
import statsmodels.api as sm
import pandas as pd
import numpy as np
```

```
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/compat/pandas.py:23: FutureWarning: The Panel class is
removed from pandas. Accessing it from the top-level namespace will also be
removed in the next version
  data_klasses = (pandas.Series, pandas.DataFrame, pandas.Panel)
```

First, load and merge the data sets. These are SAS Xport format files, which can be read with Pandas.

[2]:

```
demog = pd.read_sas("../data/DEMO_G.XPT")
bpx = pd.read_sas("../data/BPX_G.XPT")
bmx = pd.read_sas("../data/BMX_G.XPT")
df = pd.merge(demog, bpx, left_on="SEQN", right_on="SEQN")
df = pd.merge(df, bmx, left_on="SEQN", right_on="SEQN")
```

Next we convert the data from wide to long, pivoting the four BP measures from columns to rows.

[3]:

```
syvars = ["BPXSY%d" % j for j in (1,2,3,4)]
divars = ["BPXDI%d" % j for j in (1,2,3,4)]
vvars = syvars + divars
idvars = ['SEQN', 'RIDAGEYR', 'RIAGENDR', 'BMXBMI']
dx = pd.melt(df, id_vars=idvars, value_vars=vvars,
             var_name='bpvar', value_name='bp')
```

We drop rows where any of the variables are missing. Multilevel modeling can accommodate missing data, but here we use a basic complete case analysis.

[4]:

```
dx = dx.sort_values(by='SEQN')
dx = dx.reset_index(drop=True)
dx['SEQN'] = dx.SEQN.astype(np.int)
dx = dx.dropna()
```

Since we have pivoted all BP measures to rows, we will need variables telling us whether we are looking at systolic (SY) or diastolic (DI) blood pressure, and we need a way to to know the order of the BP values within each person. These repeated measures are not exchangeable, since the BP readings tend to drop slightly as people relax.

[5]:

```
# Blood pressure type (systolic or diastolic)
dx["bpt"] = dx.bpvar.str[3:5]

dx["bpi"] = dx.bpvar.str[5].astype(np.int)
dx["female"] = (dx.RIAGENDR == 2).astype(np.int)

di_mean = dx.loc[dx.bpt=="DI", :].groupby("SEQN")["bp"].aggregate(np.mean)
di_mean.name = "di_mean"
dx = pd.merge(dx, di_mean, left_on="SEQN", right_index=True)

print(dx.head())
```

```
    SEQN  RIDAGEYR  RIAGENDR  BMXBMI   bpvar     bp bpt  bpi  female  \
0  62161      22.0       1.0    23.3  BPXSY1  110.0  SY    1       0
2  62161      22.0       1.0    23.3  BPXDI2   68.0  DI    2       0
3  62161      22.0       1.0    23.3  BPXSY3  118.0  SY    3       0
4  62161      22.0       1.0    23.3  BPXDI3   74.0  DI    3       0
5  62161      22.0       1.0    23.3  BPXSY2  104.0  SY    2       0

     di_mean
0  74.666667
2  74.666667
3  74.666667
4  74.666667
5  74.666667
```

Subsample to make the script run faster. Statsmodels MixedLM is unfortunatley not very fast.

[6]:

```
dx = dx.iloc[0:5000, :]
```

Fit a linear mean structure model using OLS. The variance structure of this model is misspecified. Since this is a linear model, the coefficient point estimates are still meaningful despite the variance misspecification.

[7]:

```
model1 = sm.OLS.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI", dx)
result1 = model1.fit()
print(result1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     bp   R-squared:                       0.762
Model:                            OLS   Adj. R-squared:                  0.762
Method:                 Least Squares   F-statistic:                     4005.
Date:                Mon, 24 Feb 2020   Prob (F-statistic):               0.00
Time:                        14:31:43   Log-Likelihood:                -20686.
No. Observations:                5000   AIC:                         4.138e+04
Df Residuals:                    4995   BIC:                         4.141e+04
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       46.1460      0.863     53.463      0.000      44.454      47.838
C(bpt)[T.SY]    52.0088      0.429    121.281      0.000      51.168      52.849
RIDAGEYR         0.2911      0.010     27.819      0.000       0.271       0.312
female          -2.1229      0.434     -4.888      0.000      -2.974      -1.272
BMXBMI           0.3696      0.032     11.584      0.000       0.307       0.432
==============================================================================
Omnibus:                      424.509   Durbin-Watson:                   1.178
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2687.614
Skew:                          -0.060   Prob(JB):                         0.00
Kurtosis:                       6.590   Cond. No.                         209.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Fit a mixed model to the systolic data with a simple random intercept per subject. Then calculate the ICC.

[8]:

```
ds2 = dx.loc[dx.bpt == "SY"]
model2 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + BMXBMI",
                                 groups="SEQN", data=ds2)
result2 = model2.fit()
icc2 = result2.cov_re / (result2.cov_re + result2.scale)
print(result2.summary())
print("icc=%f\n" % icc2.values.flat[0])
```

```
         Mixed Linear Model Regression Results
========================================================
Model:              MixedLM Dependent Variable: bp
No. Observations: 2500    Method:             REML
No. Groups:       837     Scale:              16.2140
Min. group size:  1       Likelihood:         -8563.6384
Max. group size:  3       Converged:          Yes
Mean group size:  3.0
--------------------------------------------------------
             Coef.  Std.Err.   z    P>|z| [0.025  0.975]
```

```
--------------------------------------------------------
Intercept    96.233    1.960 49.100 0.000 92.391 100.074
RIDAGEYR      0.406    0.025 16.529 0.000  0.358   0.454
female       -2.765    1.018 -2.715 0.007 -4.762  -0.769
BMXBMI        0.289    0.075  3.864 0.000  0.143   0.436
SEQN Var    206.352    3.158
========================================================

icc=0.927150
```

Partial out the mean diastolic blood pressure per subject. This leads to slightly weaker random effects.

[9]:
```
model3 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + BMXBMI + di_mean",
                                 groups="SEQN", data=ds2)
result3 = model3.fit()
icc3 = result3.cov_re / (result3.cov_re + result3.scale)
print(result3.summary())
print("icc=%f\n" % icc3.values.flat[0])
```

```
         Mixed Linear Model Regression Results
========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 2500    Method:             REML
No. Groups:       837     Scale:              16.2135
Min. group size:  1       Likelihood:         -8517.7505
Max. group size:  3       Converged:          Yes
Mean group size:  3.0
--------------------------------------------------------
              Coef.  Std.Err.   z    P>|z| [0.025 0.975]
--------------------------------------------------------
Intercept     79.975    2.450 32.639 0.000 75.173 84.778
RIDAGEYR       0.345    0.024 14.443 0.000  0.299  0.392
female        -2.238    0.963 -2.324 0.020 -4.126 -0.351
BMXBMI         0.140    0.072  1.932 0.053 -0.002  0.281
di_mean        0.337    0.033 10.122 0.000  0.272  0.402
SEQN Var     183.365    2.816
========================================================

icc=0.918761
```

Fit a mixed model to the diastolic data with a random intercept per subject. Then calculate the ICC.

[10]:
```
ds3 = dx.loc[dx.bpt == "DI"]
model4 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + BMXBMI",
                                 groups="SEQN", data=ds3)
result4 = model4.fit()
```

```
icc4 = result4.cov_re / (result4.cov_re + result4.scale)
print(result4.summary())
print("icc=%f\n" % icc4.values.flat[0])
```

```
         Mixed Linear Model Regression Results
========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 2500    Method:             REML
No. Groups:       837     Scale:              36.7592
Min. group size:  1       Likelihood:         -9229.4228
Max. group size:  3       Converged:          Yes
Mean group size:  3.0
--------------------------------------------------------
              Coef.  Std.Err.   z    P>|z| [0.025 0.975]
--------------------------------------------------------
Intercept     48.224    1.926 25.045 0.000 44.450 51.998
RIDAGEYR       0.178    0.024  7.401 0.000  0.131  0.226
female        -1.562    1.001 -1.561 0.119 -3.523  0.399
BMXBMI         0.445    0.074  6.044 0.000  0.300  0.589
SEQN Var     192.039    2.024
========================================================

icc=0.839337
```

Fit a mixed model to the diastolic data with a random intercept per subject.

[11]:

```
ds3 = dx.loc[dx.bpt == "DI"]
model5 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + BMXBMI + bpi",
                                 groups="SEQN", re_formula="1+bpi",
                                 data=ds3)
result5 = model5.fit()
print(result5.summary())
```

```
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2059: ConvergenceWarning:
Retrying MixedLM optimization with lbfgs
  ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2059: ConvergenceWarning:
```

```
Retrying MixedLM optimization with cg
  ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2063: ConvergenceWarning:
MixedLM optimization failed, trying a different optimizer may help.
  warnings.warn(msg, ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2075: ConvergenceWarning:
Gradient optimization failed, |grad| = 11.616035
  warnings.warn(msg, ConvergenceWarning)

          Mixed Linear Model Regression Results
==========================================================
Model:              MixedLM Dependent Variable: bp
No. Observations:   2500    Method:             REML
No. Groups:         837     Scale:              35.6655
Min. group size:    1       Likelihood:         -9204.0121
Max. group size:    3       Converged:          No
Mean group size:    3.0
----------------------------------------------------------
                Coef.  Std.Err.   z    P>|z| [0.025 0.975]
----------------------------------------------------------
Intercept       49.150    1.865 26.359 0.000 45.496 52.805
RIDAGEYR         0.193    0.023  8.253 0.000  0.147  0.239
female          -1.512    0.965 -1.566 0.117 -3.404  0.380
BMXBMI           0.422    0.071  5.942 0.000  0.283  0.561
bpi             -0.448    0.149 -3.010 0.003 -0.739 -0.156
SEQN Var       136.382    2.444
SEQN x bpi Cov  11.527    0.406
bpi Var          1.244    0.199
==========================================================
```

Fit the same model as above, now centering the bpi (index) variable.

[12]:
```
ds3.loc[:, "bpi_cen"] = ds3.loc[:, "bpi"] - 1
model6 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + BMXBMI + bpi",
                                 groups="SEQN", re_formula="1+bpi_cen",
                                 data=ds3)
result6 = model6.fit()
print(result6.summary())
```

```
/nfs/kshedden/python3/lib/python3.7/site-packages/pandas/core/indexing.py:376:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[key] = _infer_fill_value(value)
/nfs/kshedden/python3/lib/python3.7/site-packages/pandas/core/indexing.py:494:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[item] = s

            Mixed Linear Model Regression Results
==============================================================
Model:              MixedLM   Dependent Variable:   bp
No. Observations:   2500      Method:               REML
No. Groups:         837       Scale:                31.6095
Min. group size:    1         Likelihood:           -9199.2717
Max. group size:    3         Converged:            Yes
Mean group size:    3.0
--------------------------------------------------------------
                    Coef.  Std.Err.   z    P>|z| [0.025 0.975]
--------------------------------------------------------------
Intercept           49.156    1.891 25.997 0.000 45.450 52.861
RIDAGEYR             0.191    0.024  8.077 0.000  0.145  0.237
female              -1.524    0.977 -1.559 0.119 -3.439  0.392
BMXBMI               0.425    0.072  5.909 0.000  0.284  0.566
bpi                 -0.458    0.156 -2.946 0.003 -0.763 -0.153
SEQN Var           170.158    2.448
SEQN x bpi_cen Cov   8.576    0.399
bpi_cen Var          4.835    0.253
==============================================================
```

Fit a mixed model to both types of BP jointly with a random intercept per subject. Note that the random intercept is share between the two types of blood pressure (systolic and diastolic).

[13]:
```
model7 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI",
                                 groups="SEQN", data=dx)
result7 = model7.fit()
print(result7.summary())
```

```
          Mixed Linear Model Regression Results
=========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 5000    Method:             REML
No. Groups:       837     Scale:              110.9918
```

```
Min. group size:  2       Likelihood:        -19709.8882
Max. group size:  6       Converged:         Yes
Mean group size:  6.0
----------------------------------------------------------
              Coef.  Std.Err.    z    P>|z| [0.025 0.975]
----------------------------------------------------------
Intercept     46.215    1.592  29.038 0.000 43.096 49.335
C(bpt)[T.SY]  52.009    0.298 174.537 0.000 51.425 52.593
RIDAGEYR       0.292    0.020  14.715 0.000  0.253  0.331
female        -2.157    0.823  -2.620 0.009 -3.771 -0.544
BMXBMI         0.367    0.061   6.068 0.000  0.249  0.486
SEQN Var     119.727    0.705
==========================================================
```

Fit a mixed model to both types of BP with subject random intercept and unique random effect per BP type with common variance.

[14]:

```python
model8 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI",
                                 groups="SEQN", re_formula="1",
                                 vc_formula={"bpt": "0+C(bpt)"},
                                 data=dx)
result8 = model8.fit()
print(result8.summary())
```

```
          Mixed Linear Model Regression Results
==========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 5000    Method:             REML
No. Groups:       837     Scale:              26.4863
Min. group size:  2       Likelihood:         -17909.6150
Max. group size:  6       Converged:          Yes
Mean group size:  6.0
----------------------------------------------------------
               Coef.  Std.Err.   z    P>|z| [0.025 0.975]
----------------------------------------------------------
Intercept      46.221    1.613 28.656 0.000 43.060 49.383
C(bpt)[T.SY]   52.018    0.598 86.919 0.000 50.845 53.191
RIDAGEYR        0.292    0.020 14.716 0.000  0.253  0.331
female         -2.164    0.824 -2.628 0.009 -3.779 -0.550
BMXBMI          0.367    0.061  6.059 0.000  0.248  0.486
SEQN Var       63.569    1.530
bpt Var       140.980    1.593
==========================================================
```

Fit a mixed model to both types of BP with subject random intercept and unique random effect per BP type with unique variance.

[15]:

```
dx["sy"] = (dx.bpt == "SY").astype(np.int)
dx["di"] = (dx.bpt == "DI").astype(np.int)
model9 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI",
                                 groups="SEQN", re_formula="1",
                                 vc_formula={"sy": "0+sy", "di": "0+di"},
                                 data=dx)
result9 = model9.fit()
print(result9.summary())
```

```
          Mixed Linear Model Regression Results
=========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 5000    Method:             REML
No. Groups:       837     Scale:              26.4867
Min. group size:  2       Likelihood:         -17909.4687
Max. group size:  6       Converged:          Yes
Mean group size:  6.0
---------------------------------------------------------
               Coef.  Std.Err.   z    P>|z| [0.025 0.975]
---------------------------------------------------------
Intercept      46.274    1.614 28.669 0.000 43.111 49.438
C(bpt)[T.SY]   52.018    0.598 86.914 0.000 50.845 53.191
RIDAGEYR        0.289    0.021 14.030 0.000  0.249  0.329
female         -2.148    0.824 -2.608 0.009 -3.763 -0.534
BMXBMI          0.369    0.061  6.083 0.000  0.250  0.488
SEQN Var       63.540    1.529
di Var        137.038    2.105
sy Var        144.954    2.168
=========================================================
```

Below we consider the possibility that theremay be heteroscedasticity between the two blood pressure types. That is, systolic blood pressure measurements may be more variable than diastolic measurements, or vice versa. This analysis is a bit awkward to conduct. Below we fit two models, one in which diastolic measurements are allowed to be more variable than systolic measurements, and one in which systolic measurements are allowed to be more variable than diastolic measurements. In theory, a variance parameters should be equal to zero in one of these models, revealing which type of blood pressure has more variability.

[16]:

```
dx["sy1"] = (dx.bpvar == "BPXSY1").astype(np.int)
dx["sy2"] = (dx.bpvar == "BPXSY2").astype(np.int)
dx["sy3"] = (dx.bpvar == "BPXSY3").astype(np.int)
dx["di1"] = (dx.bpvar == "BPXDI1").astype(np.int)
dx["di2"] = (dx.bpvar == "BPXDI2").astype(np.int)
dx["di3"] = (dx.bpvar == "BPXDI3").astype(np.int)
model10 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI",
                                 groups="SEQN", re_formula="1",
                                 vc_formula={"sy": "0+sy", "di": "0+di",
                                              "dye": "0+di1+di2+di3"},
                                  data=dx)
result10 = model10.fit()
print(result10.summary())

model11 = sm.MixedLM.from_formula("bp ~ RIDAGEYR + female + C(bpt) + BMXBMI",
                                  groups="SEQN", re_formula="1",
                                  vc_formula={"sy": "0+sy", "di": "0+di",
                                              "sye": "0+sy1+sy2+sy3"},
                                  data=dx)
result11 = model11.fit()
print(result11.summary())
```

```
          Mixed Linear Model Regression Results
=========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 5000    Method:             REML
No. Groups:       837     Scale:              16.1732
Min. group size:  2       Likelihood:         -17769.2792
Max. group size:  6       Converged:          Yes
Mean group size:  6.0
---------------------------------------------------------
               Coef.  Std.Err.   z    P>|z| [0.025 0.975]
---------------------------------------------------------
Intercept      46.301    1.616 28.655 0.000 43.134 49.468
C(bpt)[T.SY]   52.003    0.599 86.752 0.000 50.828 53.178
RIDAGEYR        0.289    0.021 13.991 0.000  0.248  0.329
female         -2.153    0.825 -2.611 0.009 -3.769 -0.537
BMXBMI          0.369    0.061  6.082 0.000  0.250  0.488
SEQN Var       63.534    1.996
di Var        134.571    2.807
dye Var        21.006    0.454
sy Var        148.524    2.930
=========================================================
```

```
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2059: ConvergenceWarning:
Retrying MixedLM optimization with lbfgs
  ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

```
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2059: ConvergenceWarning:
Retrying MixedLM optimization with cg
  ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2063: ConvergenceWarning:
MixedLM optimization failed, trying a different optimizer may help.
  warnings.warn(msg, ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2075: ConvergenceWarning:
Gradient optimization failed, |grad| = 145.966482
  warnings.warn(msg, ConvergenceWarning)
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/regression/mixed_linear_model.py:2115: ConvergenceWarning:
The Hessian matrix at the estimated parameter values is not positive definite.
  warnings.warn(msg, ConvergenceWarning)

          Mixed Linear Model Regression Results
=========================================================
Model:            MixedLM Dependent Variable: bp
No. Observations: 5000    Method:             REML
No. Groups:       837     Scale:              28.7455
Min. group size:  2       Likelihood:         -17944.9876
Max. group size:  6       Converged:          No
Mean group size:  6.0
---------------------------------------------------------
              Coef.  Std.Err.    z    P>|z| [0.025 0.975]
---------------------------------------------------------
Intercept     46.045    1.617  28.471 0.000 42.876 49.215
C(bpt)[T.SY]  52.017    0.518 100.461 0.000 51.003 53.032
RIDAGEYR       0.302    0.021  14.605 0.000  0.261  0.342
female        -2.216    0.827  -2.681 0.007 -3.836 -0.596
BMXBMI         0.360    0.061   5.918 0.000  0.241  0.479
SEQN Var      83.637    1.215
di Var       112.382    1.240
sy Var        92.482    1.220
sye Var        0.641
=========================================================


/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/base/model.py:1286: RuntimeWarning: invalid value
encountered in sqrt
  bse_ = np.sqrt(np.diag(self.cov_params()))
```