Multilevel and marginal modeling, a case study with the NHANES data

This notebook is a counterpart to our introductory regression modeling case study for independent data using the NHANES data. Here we will build on the basic linear and logistic regression approaches discussed in that notebook. We will be exploring the use of some more advanced regression approaches that can be used for data that are statistically dependent.

We begin by importing the libraries that we will be using.

In [3]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
import numpy as np
```

Data can be *dependent*, or have a *mutilevel structure* for many reasons. In practice, most datasets exhibit some form of dependence, and it is arguably independent data, not dependent data, that should be treated as the exceptional case. Here we will reconsider the NHANES data from the perspective of dependence, focusing in particular on dependence in the data that arises due to *clustering* which we will define below.

First, we read in the data, as we have done before. For simplicity, we remove all rows of data with missing values in the variables of interest (note that there are more sophisticated approaches for handling missing data that generally will give better results, but for simplicity we do not use them here).

Note that we retain two variables here SDMVSTRA and SDMVPSU that will be used below to define the clustering structure in these data.

In [4]:
```
# Read the data file
da = pd.read_csv("nhanes_2015_2016.csv")

# Drop unused columns, drop rows with any missing values.
vars = ["BPXSY1", "RIDAGEYR", "RIAGENDR", "RIDRETH1", "DMDEDUC2", "BMXBMI",
        "SMQ020", "SDMVSTRA", "SDMVPSU"]
da = da[vars].dropna()
```

Introduction to clustered data

One common reason that data are dependent is that the data values were collected in clusters. This essentially means that the population was partitioned into groups, a limited number of these groups were somehow selected, and then a limited number of individuals were selected from each of the selected groups. In a proper survey, there is a well-planned design, in which both the groups, and the individuals within groups, are selected randomly. The goal in doing this is to maximize the chances that the sample is representative of the population of interest in all relevant ways. But many other data sets exhibit clustering structure, even when the data collection was not so carefully planned.

Regardless of how the clustering in the sample arose, it is likely to be the case that observations within a cluster are more similar to observations in different clusters. To make this concrete, note that clustering is often geographic. Data may be collected by visiting several locations, then recruiting participants within each location. People within a location may share similarities, for example in demography and lifestyle, or they may share environmental circumstances such as climate. When we have clustered data, it is usually advisable to account for this in the analysis.

Clustering structure in NHANES

The detailed process of collecting data for a study like NHANES is very complex, so we will simplify things substantially here (more details can be found here but are not needed for this course). Roughly speaking, in NHANES the data are collected by selecting a limited number of counties in the US, then selecting subregions of these counties, then selecting people within these subregions. Since counties are geographically constrained, it is expected that people within a county are more similar to each other than they are to people in other counties.

If we could obtain the county id where each NHANES participant resides, we could directly study this clustering structure. However for privacy reasons this information is not released with the data. Instead, we have access to "masked variance units" (MVUs), which are formed by combining subregions of different counties into artificial groups that are not geographically contiguous. While the MVUs are not the actual clusters of the survey, and are not truly contiguous geographic regions, they are deliberately selected to mimic these things, while minimizing the risk that a subject can be "unmasked" in the data. For the remainder of this notebook, we will treat the MVUs as clusters, and explore the extent to which they induce correlations in some of the NHANES variables that we have been studying.

The MVU identifiers can be obtained by combining the `SDMVSTRA` and `SDMVPSU` identifiers, which we do next:

In [5]:
```
da["group"] = 10*da.SDMVSTRA + da.SDMVPSU
```

Intraclass correlation

Similarity among observations within a cluster can be measured using a statistic called the *intraclass correlation*, or ICC. This is a distinct form of correlation from Pearson's correlation. The ICC takes on values from 0 to 1, with 1 corresponding to "perfect clustering" -- the values within a cluster are identical, and 0 corresponding to "perfect independence" -- the mean value within each cluster is identical across all the clusters.

We can assess ICC using two regression techniques, *marginal regression*, and *multilevel regression*. We will start by using a technique called "Generalized Estimating Equations" (GEE) to fit marginal linear models, and to estimate the ICC for the NHANES clusters.

We will first look at the ICC for systolic blood pressure:

In [6]:
```
model = sm.GEE.from_formula("BPXSY1 ~ 1", groups="group",
            cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
print(result.cov_struct.summary())
```

```
The correlation between two observations in the same cluster is 0.030
```

The estimated ICC is 0.03, which is small but not negligible. Although an ICC is a type of correlation, its values are not directly comparable to Pearson correlation values. While 0.03 would generally be considered to be very small as a Pearson correlation coefficient, it is not especially small as an ICC.

To get a more systematic view of the ICC values induced by clustering in these data, we calculate the ICC for a number of different variables that appear in our analyses, either as outcomes or as predictors.

In [7]:
```
# Recode smoking to a simple binary variable
da["smq"] = da.SMQ020.replace({2: 0, 7: np.nan, 9: np.nan})

for v in ["BPXSY1", "RIDAGEYR", "BMXBMI", "smq", "SDMVSTRA"]:
    model = sm.GEE.from_formula(v + " ~ 1", groups="group",
            cov_struct=sm.cov_struct.Exchangeable(), data=da)
    result = model.fit()
    print(v, result.cov_struct.summary())
```

```
BPXSY1 The correlation between two observations in the same cluster is 0.030
RIDAGEYR The correlation between two observations in the same cluster is 0.035
BMXBMI The correlation between two observations in the same cluster is 0.039
smq The correlation between two observations in the same cluster is 0.026
SDMVSTRA The correlation between two observations in the same cluster is 0.959
```

The values are generally similar to what we saw for blood pressure, except for `SDMVSTRA`, which is one component of the cluster definition itself, and therefore has a very high ICC.

To illustrate that the ICC values shown above are not consistent with a complete absence of dependence, we simulate 10 sets of random data and calculate the ICC value for each set:

In [8]:
```
for k in range(10):
    da["noise"] = np.random.normal(size=da.shape[0])
    model = sm.GEE.from_formula("noise ~ 1", groups="group",
           cov_struct=sm.cov_struct.Exchangeable(), data=da)
    result = model.fit()
    print(v, result.cov_struct.summary())
```

```
SDMVSTRA The correlation between two observations in the same cluster is 0.003
SDMVSTRA The correlation between two observations in the same cluster is 0.002
SDMVSTRA The correlation between two observations in the same cluster is -0.000
SDMVSTRA The correlation between two observations in the same cluster is -0.002
SDMVSTRA The correlation between two observations in the same cluster is -0.002
SDMVSTRA The correlation between two observations in the same cluster is -0.001
SDMVSTRA The correlation between two observations in the same cluster is -0.000
SDMVSTRA The correlation between two observations in the same cluster is -0.001
SDMVSTRA The correlation between two observations in the same cluster is 0.004
SDMVSTRA The correlation between two observations in the same cluster is -0.003
```

We see that the estimated ICC for pure simulated noise is random but highly concentrated near zero, varying from around `-0.002` to `+0.002`.

Conditional intraclass correlation

The ICC's studied above were *marginal*, in the sense that we were looking at whether, say, the SBP values were more similar within versus between clusters. To the extent that such "cluster effects" are found, it may be largely explained by demographic differences among the clusters. For example, we know from our previous analyses with the NHANES data that older people have higher SBP than younger people. Also, some clusters may contain a slightly older or younger set of people than others. Thus, by controlling for age, we might anticipate that the ICC will become smaller. This is shown in the next analysis:

In [9]:
```
model = sm.GEE.from_formula("BPXSY1 ~ RIDAGEYR", groups="group",
           cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
print(result.cov_struct.summary())
```

```
The correlation between two observations in the same cluster is 0.019
```

The ICC for SBP drops from 0.03 to 0.02. We can now assess whether it drops even further when we add additional covariates that we know to be predictive of blood pressure.

In [10]:
```
# Create a labeled version of the gender variable
da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})

model = sm.GEE.from_formula("BPXSY1 ~ RIDAGEYR + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          groups="group",
          cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
print(result.cov_struct.summary())
```

```
The correlation between two observations in the same cluster is 0.013
```

The variable RIDRETH1 is a categorical variable containing 5 levels of race/ethnicity information. Since NHANES categorical variables are coded numerically, Statsmodels would have no way of knowing that these are codes and not quantitative data, thus we must use the `C()` syntax in the formula above to force this variable to be treated as being categorical. We see here that the ICC has further reduced, to 0.013, due to controlling for these additional factors including ethnicity.

Marginal linear models with dependent data

Above we focused on quantifying the dependence induced by clustering. By understanding the clustering structure, we have gained additional insight about the data that complements our understanding of the mean structure. Another facet of working with dependent data is that while the mean structure (i.e. the regression coefficients) can be estimated without considering the dependence structure of the data, the standard errors and other statistics relating to uncertainty will be wrong when we ignore dependence in the data.

To illustrate this, below we fit two models with the same mean structure to the NHANES data. The first is a multiple regression model fit using "ordinary least squares" (the default method for independent data). The second is fit using GEE, which allows us to account for the dependence in the data.

In [11]:
```
# Fit a linear model with OLS
model1 = sm.OLS.from_formula("BPXSY1 ~ RIDAGEYR + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          data=da)
result1 = model1.fit()

# Fit a marginal linear model using GEE to handle dependent data
model2 = sm.GEE.from_formula("BPXSY1 ~ RIDAGEYR + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          groups="group",
          cov_struct=sm.cov_struct.Exchangeable(), data=da)
result2 = model2.fit()

x = pd.DataFrame({"OLS_params": result1.params, "OLS_SE": result1.bse,
                  "GEE_params": result2.params, "GEE_SE": result2.bse})
x = x[["OLS_params", "OLS_SE", "GEE_params", "GEE_SE"]]
print(x)
```

```
                    OLS_params    OLS_SE  GEE_params    GEE_SE
Intercept            91.736583  1.339378   92.168530  1.384309
RIAGENDRx[T.Male]     3.671294  0.453763    3.650245  0.454498
C(RIDRETH1)[T.2]      0.855488  0.819486    0.159296  0.767025
C(RIDRETH1)[T.3]     -1.796132  0.671954   -2.233280  0.760228
C(RIDRETH1)[T.4]      3.813314  0.732355    3.105654  0.881580
C(RIDRETH1)[T.5]     -0.455347  0.808948   -0.439831  0.813675
RIDAGEYR              0.478699  0.012901    0.474101  0.018493
BMXBMI                0.278015  0.033285    0.280205  0.038553
```

In the results above, we see that the point estimates are similar between the OLS and GEE fits of the model, but the standard errors tend to be larger in the GEE fit. For example, the standard errors for BMI and age are 20-40% larger in the GEE fit. Since we know that there is dependence in these data that is driven by clustering, the OLS approach is not theoretically justified (the OLS parameter estimates remain in meaningful, but the standard errors do not). GEE parameter estimates and standard errors are meaningful in the presence of dependence, as long as the dependence is exclusively between observations within the same cluster.

Marginal logistic regression with dependent data

Above we used GEE to fit marginal linear models in the presence of dependence. GEE can also be used to fit any GLM in the presence of dependence. We illustrate this using models relating smoking history to several demographic predictor variables. These are the same models we fit in our previous notebook using GLM, which ignores the clustering. Below we fit this same GLM again for comparison, then we fit the marginal model using GEE. One thing to emphasize

using GLM, which ignores the clustering. Below we fit this same GLM again for comparison, then we fit the marginal model using GEE. One thing to emphasize in doing this is that the GLM and GEE are both legitimate estimators of the marginal mean structure here. However GLM will not give correct standard errors, so everything derived from standard errors (e.g. confidence intervals and hypothesis tests) will not be correct.

In [12]:
```
# Relabel the levels, convert rare categories to missing.
da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "lt9", 2: "x9_11", 3: "HS", 4: "SomeCollege",
                                       5: "College", 7: np.nan, 9: np.nan})

# Fit a basic GLM
model1 = sm.GLM.from_formula("smq ~ RIDAGEYR + RIAGENDRx + C(DMDEDUC2x)",
          family=sm.families.Binomial(), data=da)
result1 = model1.fit()
result1.summary()

# Fit a marginal GLM using GEE
model2 = sm.GEE.from_formula("smq ~ RIDAGEYR + RIAGENDRx + C(DMDEDUC2x)",
           groups="group", family=sm.families.Binomial(),
           cov_struct=sm.cov_struct.Exchangeable(), data=da)
result2 = model2.fit(start_params=result1.params)

x = pd.DataFrame({"OLS_params": result1.params, "OLS_SE": result1.bse,
                  "GEE_params": result2.params, "GEE_SE": result2.bse})
x = x[["OLS_params", "OLS_SE", "GEE_params", "GEE_SE"]]
print(x)
```

```
                            OLS_params    OLS_SE  GEE_params    GEE_SE
  Intercept                 -2.305999  0.114308   -2.249820  0.140567
  RIAGENDRx[T.Male]          0.909597  0.060167    0.908682  0.062342
  C(DMDEDUC2x)[T.HS]         0.943364  0.089663    0.887965  0.095397
  C(DMDEDUC2x)[T.SomeCollege]  0.832227  0.084361    0.771636  0.104449
  C(DMDEDUC2x)[T.lt9]        0.266228  0.109183    0.321784  0.141327
  C(DMDEDUC2x)[T.x9_11]      1.098561  0.106697    1.062149  0.138401
  RIDAGEYR                   0.018257  0.001725    0.017416  0.001803
```

As expected, the results show that the GLM and the GEE give very similar estimates for the regression parameters. However the standard errors obtained using GEE are somewhat larger than those obtained using GLM. This indicates that GLM understates the uncertainty in the estimated mean structure, which is a direct consequence of it ignoring the dependence structure. The GEE results do not suffer from this weakness.

To the extent that the GLM and GEE parameter estimates differ, this is due to GEE attempting to exploit the dependence structure to obtain more efficient (i.e. more accurate) estimates of the model parameters. Thus, in summary, we can view GEE as trying to accomplish three things above and beyond what we obtain from GLM:

- GEE gives us insight into the dependence structure of the data
- GEE uses the dependence structure to obtain meaningful standard errors of the estimated model parameters.
- GEE uses the dependence structure to estimate the model parameters more accurately

In contrast, GLM does not achieve the first point at all, and in terms of the second point, the GLM standard errors can be far too optimistic (i.e. too small) -- note that in the analysis we are pursuing here, even weak clustering (ICC around 0.02-0.04) modifies some of the standard errors by 10-40%. Finally, with regard to the third point, GEE should in general have an efficiency advantage over GLM, but GLM estimates remain "valid" and cannot be completely dismissed solely on this basis.

Multilevel models

Multilevel modeling is a large topic, and some aspects of it are quite advanced. Here, we will explore one facet of multilevel modeling -- using it as an alternative way to accommodate dependence in clustered data. In this sense, multilevel modeling is an alternative to the marginal regression analysis demonstrated above.

In the setting of linear regression, mutilevel models and marginal models are similar in most ways (note that more substantial differences between marginal and multilevel models emerge in the case of logistic regression, and in other generalized linear or nonlinear models). Multilevel models and marginal models estimate the same population target, but represent this target in different ways, and utilize different estimation procedures.

A multilevel model is usually expressed in terms of *random effects*. These are variables that we do not observe, but that we can nevertheless incorporate into a statistical model. We cannot get into all of the technical details here, but it is important to understand that while these random effects are not observed, their presence can be inferred through the data, as long as each random effect is modeled as influencing at least two observations.

In the present setting, we are focusing only on dependence that arises through a single level of clustering. To put this in the context of multilevel modeling, we can imagine that each cluster has a random effect that is shared by all observations in that cluster. For example, if SBP tends to be around 0.5 units higher in one cluster, then the random effect for that cluster would be 0.5, and it would add to the predicted SBP for every observation in the cluster.

In [13]:
```
# Fit a multilevel (mixed effects) model to handle dependent data
model = sm.MixedLM.from_formula("BPXSY1 ~ RIDAGEYR + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          groups="group", data=da)
result = model.fit()
result.summary()
```

Out[13]:

Model:	MixedLM	Dependent Variable:	BPXSY1
No. Observations:	5102	Method:	REML
No. Groups:	30	Scale:	256.6952
Min. group size:	106	Likelihood:	-21409.8702
Max. group size:	226	Converged:	Yes
Mean group size:	170.1		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	92.173	1.402	65.752	0.000	89.426	94.921
RIAGENDRx[T.Male]	3.650	0.452	8.084	0.000	2.765	4.535
C(RIDRETH1)[T.2]	0.153	0.887	0.172	0.863	-1.586	1.891
C(RIDRETH1)[T.3]	-2.238	0.758	-2.954	0.003	-3.723	-0.753
C(RIDRETH1)[T.4]	3.098	0.836	3.707	0.000	1.460	4.737
C(RIDRETH1)[T.5]	-0.439	0.878	-0.500	0.617	-2.161	1.282
RIDAGEYR	0.474	0.013	36.482	0.000	0.449	0.500
BMXBMI	0.280	0.033	8.404	0.000	0.215	0.346
group Var	3.615	0.085				

The "variance structure parameters" are what distinguish a mixed model from a marginal model. Here we only have one such parameter, which is the variance for groups, estimated to be 3.615. This means that if we were to choose two groups at random, their random effects would differ on average by around 2.69 (this is calculated as the square root of `2*3.615`). This is a sizable shift, comparable to the difference between females and males, or to around 6 years of aging.

We have seen here that it least in this setting, the mixed modeling procedure accommodates dependence in the data, provides rigorous estimates of the strength of this dependence, and accounts for the dependence in both estimation and inference for the regression parameters. In this sense, the multilevel model has the same desirable properties as GEE (at least in this setting). In fact, each of these two methods is very useful and widely utilized. There are some settings where either GEE or multilevel modeling can be argued to have an advantage, but neither uniformly dominates the other.

Multilevel models can also be used to estimate ICC values. In the case of a model with one level, which is what we have here, the ICC is the variance of the grouping variable (3.615) divided by the sum of the variance of the grouping variable and the unexplained variance (256.7). Note that the unexplained variance is in upper part of the output, labeled *scale*. This ratio is around 0.014, which is very similar to the estimated ICC obtained using GEE.

Predicted random effects

While the actual random effects in a multilevel model are never observable, we can predict them from the data. This is sometimes useful, although the emphasis in multilevel regression usually lies with the structural parameters that underlie the random effects, and not the random effects themselves. In the NHANES analysis, for example, we could use the predicted random effects to quantify the uniqueness of each county relative to the mean.

The predicted random effects for the 30 groups in this analysis are shown below:

In [14]:
```
result.random_effects
```

Out[14]:
```
{1191: group   -1.630976
 dtype: float64, 1192: group   -0.086162
 dtype: float64, 1201: group   -2.042661
 dtype: float64, 1202: group   -0.147472
 dtype: float64, 1211: group    0.280623
 dtype: float64, 1212: group    1.580732
 dtype: float64, 1221: group    0.283347
 dtype: float64, 1222: group    0.131512
 dtype: float64, 1231: group   -2.038171
 dtype: float64, 1232: group    0.617651
 dtype: float64, 1241: group    2.878488
 dtype: float64, 1242: group   -0.519364
 dtype: float64, 1251: group    2.064967
 dtype: float64, 1252: group    1.521281
 dtype: float64, 1261: group   -1.261975
 dtype: float64, 1262: group    0.980846
 dtype: float64, 1271: group    0.118031
 dtype: float64, 1272: group   -0.128397
 dtype: float64, 1281: group   -0.384862
 dtype: float64, 1282: group   -3.582111
 dtype: float64, 1291: group   -3.271017
 dtype: float64, 1292: group   -0.829538
 dtype: float64, 1301: group   -0.884171
 dtype: float64, 1302: group    2.790657
 dtype: float64, 1311: group   -0.585201
 dtype: float64, 1312: group    1.198291
 dtype: float64, 1321: group   -0.195692
 dtype: float64, 1322: group    1.955515
 dtype: float64, 1331: group   -0.305559
 dtype: float64, 1332: group    1.491389
 dtype: float64}
```

Based on these predicted random effects (also known as *BLUP*s), we see, for example, that cluster 1241 has unusually high SBP, and cluster 1282 has unusually low SBP. These deviations from what is expected are observed after adjusting for the covariates in the model, and hence are presumably driven by other characteristics of these clusters that are not reflected in the covariates.

Random slopes

Multilevel modeling is a broad framework that allows many different types of models to be specified and fit. Here we are primarily focusing on the "random intercept models", that allow the data in each cluster to be shifted by a common amount, in order to account for stable confounders within clusters. Above we found some evidence that such clustering is present in the data. Next we consider a more subtle form of cluster effect, in which the slope for a specific covariate varies from cluster to cluster. This is called a *random slopes model*. To demonstrate, below we fit a model in which SBP has cluster-specific intercepts, and cluster-specific slopes for the age covariate. That is, we ask whether the rate at which blood pressure increases with age might differ from one cluster to the next.

We fit two variations on this model. In the first model, the cluster-specific intercepts and slopes are independent random variables. That is, a cluster with unusually high SBP is no more or less likely to have unusually rapid increase of SBP with age. Note that when working with random slopes, it is usually advisable to center any covariate which has a random slope. This does not change the fundamental interpretation of the model, but it does often result in models that converge faster and more robustly. Here we center within each cluster, although it is also common to center in the entire dataset, rather than centering separately by cluster.

In [15]:
```
da["age_cen"] = da.groupby("group").RIDAGEYR.transform(lambda x: x - x.mean())

model = sm.MixedLM.from_formula("BPXSY1 ~ age_cen + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          groups="group", vc_formula={"age_cen": "0+age_cen"}, data=da)
result = model.fit()
result.summary()
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/regression/mixed_linear_model.py:2045: ConvergenceWarning: The MLE may be o
n the boundary of the parameter space.
  warnings.warn(msg, ConvergenceWarning)
```

Out[15]:

Model:	MixedLM	Dependent Variable:	BPXSY1
No. Observations:	5102	Method:	REML
No. Groups:	30	Scale:	263.7323
Min. group size:	106	Likelihood:	-21469.9240
Max. group size:	226	Converged:	Yes
Mean group size:	170.1		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	115.207	1.209	95.265	0.000	112.836	117.577
RIAGENDRx[T.Male]	3.643	0.457	7.962	0.000	2.746	4.539
C(RIDRETH1)[T.2]	1.167	0.827	1.412	0.158	-0.453	2.787
C(RIDRETH1)[T.3]	-1.659	0.679	-2.444	0.015	-2.989	-0.328
C(RIDRETH1)[T.4]	3.610	0.739	4.884	0.000	2.161	5.058
C(RIDRETH1)[T.5]	-1.208	0.816	-1.480	0.139	-2.807	0.392
age_cen	0.467	0.018	26.235	0.000	0.432	0.502
BMXBMI	0.288	0.034	8.574	0.000	0.222	0.353
age_cen Var	0.004	0.000				

We see that the estimated variance for random age slopes is 0.004, which translates to a standard deviation of 0.06. That is, from one cluster to another, the age slopes fluctuate by $\pm 0.06 - 0.12$ (1-2 standard deviations). These cluster-specific fluctuations are added/subtracted from the fixed effect for age, which is 0.467. Thus, in some clusters SBP may increase by around `0.467 + 0.06 = 0.527` mm/Hg per year, while in other clusters SBP may increase by only around `0.467 - 0.06 = 0.407` mm/Hg per year. Note also that the fitting algorithm produces a warning that the estimated variance parameter is close to the boundary. In this case, however, the algorithm seems to have converged to a point just short of the boundary.

Next, we fit a model in which the cluster-specific intercepts and slopes are allowed to be correlated.

In [16]:
```
model = sm.MixedLM.from_formula("BPXSY1 ~ age_cen + RIAGENDRx + BMXBMI + C(RIDRETH1)",
          groups="group", re_formula="1+age_cen", data=da)
result = model.fit()
result.summary()
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/regression/mixed_linear_model.py:2045: ConvergenceWarning: The MLE may be o
n the boundary of the parameter space.
  warnings.warn(msg, ConvergenceWarning)
```

Out[16]:

Model:	MixedLM	Dependent Variable:	BPXSY1
No. Observations:	5102	Method:	REML

No. Observations:	5102	Method:	REML
No. Groups:	30	Scale:	255.4451
Min. group size:	106	Likelihood:	-21413.6193
Max. group size:	226	Converged:	Yes
Mean group size:	170.1		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	115.467	1.340	86.173	0.000	112.840	118.093
RIAGENDRx[T.Male]	3.662	0.451	8.121	0.000	2.778	4.546
C(RIDRETH1)[T.2]	0.023	0.898	0.025	0.980	-1.738	1.783
C(RIDRETH1)[T.3]	-2.251	0.778	-2.893	0.004	-3.775	-0.726
C(RIDRETH1)[T.4]	3.011	0.854	3.524	0.000	1.336	4.686
C(RIDRETH1)[T.5]	-0.585	0.893	-0.655	0.512	-2.336	1.165
age_cen	0.466	0.018	26.286	0.000	0.431	0.501
BMXBMI	0.283	0.033	8.497	0.000	0.218	0.349
group Var	8.655	0.169				
group x age_cen Cov	0.119	0.004				
age_cen Var	0.004	0.000				

Again, we get a warning that the algorithm is close to the boundary. The estimated correlation coefficient between random slopes and random intercepts is estimated to be `0.119/sqrt(8.655*0.004)`, which is around `0.64`. This indicates that the clusters with unusually high average SBP also tend to have SBP increasing faster with age. Note however that these structural parameters are only estimates, and due to the variance parameter falling close to the boundary, the estimates may not be particularly precise.

Multilevel logistic regression

Logistic regression models with random effects can be fit using a technique that is somewhat analogous to the way that we fit mixed linear models. The computational algorithms used to fit these models are quite advanced. Versions of these algorithms have been implemented in the development version of Statsmodels, but are not yet released. At this point, it may not be practical to fit multilevel logistic models in Python, but this should become a possibility in the near future.