

# nhanes\_gee

February 17, 2020

## 1 Generalized estimating equations (GEE)

Generalized estimating equations (GEE) is an approach to fitting regression models that was first formalized in the 1980's, building on and unifying earlier work. The main goal of GEE is to extend the framework of generalized linear modeling (GLM) to handle certain types of dependent data. Like GLM, GEE focuses primarily on the *mean structure*  $E[y|x_1, \dots, x_p]$ , where  $y$  here is the dependent variable of the regression, and  $x_1, \dots, x_p$  are the covariates. As in GLM, a link function  $g$  is used, so we will be relating the mean to the linear predictor through the mean structure model

$$g(E[y|x_1, \dots, x_p]) = b_0 + b_1x_1 + \dots + b_px_p.$$

A GEE regression also involves a mean/variance relationship function  $f$ , which plays the same role as in a GLM:

$$\text{Var}[y|x_1, \dots, x_p] = \phi \cdot f(E[y|x_1, \dots, x_p]).$$

Other concepts from GLM like the scale parameter ( $\phi$ ) and family (e.g. `binomial`) play similar roles in GEE as they do in GLM.

The main aspect of a GEE that is not present in a GLM is the *working dependence structure*, or *working correlation*. This specifies how observations are related to each other. In a GLM, the observations are treated as being independent of each other, while GEE can accommodate many different forms of dependence between observations.

The reason that the dependence structure in a GEE is referred to as a “working” dependence structure is that it does not have to be correct in order for the results of the regression to be valid. If the working dependence structure is correct, the GEE results will be more efficient (i.e. the estimates will be more accurate, and the standard errors will tend to be smaller). If the working dependence structure is incorrect, the parameter estimates will be less accurate, and the standard errors will be correspondingly larger. However the standard errors continue to correctly reflect the uncertainty in the parameter estimates, and with enough data, the parameter estimates will still become arbitrarily accurate.

This notebook is organized as a case study. We begin by using data from [NHANES](#), the *National Health and Nutrition Examination Survey*.

First we import the libraries that we will be using.

```
[1]: import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
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)
```

Next we read the data. For simplicity, here we will use “complete case analysis”, meaning that we drop all cases with a missing value on any variable of potential interest.

```
[2]: url = "https://raw.githubusercontent.com/kshedden/statswpy/master/NHANES/merged/  
      ↪nhanes_2015_2016.csv"  
da = pd.read_csv(url)  
  
# Drop unused columns, drop rows with any missing values.  
vars = ["SEQN", "BPXSY1", "BPXSY2", "RIDAGEYR", "RIAGENDR", "RIDRETH1",  
        "DMDEDUC2", "BMXBMI", "SMQO20", "INDFMPIR", "SDMVSTRA", "SDMVPSU",  
        "DMDMARTL"]  
da = da[vars].dropna()
```

Below we will be using several NHANES variables that are categorical. To make the results easier to interpret, we next create variables that express the groups using text labels.

```
[3]: da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})  
  
da["RIDRETH1x"] = da.RIDRETH1.replace({1: "MexicanAmerican", 2: "OtherHispanic",  
                                       3: "NonHispanicWhite", 4: "NonHispanicBlack", 5: "Other"})  
  
da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "Lt9", 2: "g9-11", 3: "HS", 4: "  
      ↪SomeCollege",  
                                       5: "College", 6: np.nan, 7: np.nan, 9: np.nan})  
  
da["DMDMARTLx"] = da.DMDMARTL.replace({1: "Married", 2: "Widowed", 3: "  
      ↪Divorced",  
                                       4: "Separated", 5: "NeverMarried", 6: "Partnered",  
                                       77: np.nan, 99: np.nan})
```

## 1.1 Cluster samples and grouped data

The overall NHANES sample for one wave is intended to represent the adult US population at a particular point in time (strictly speaking, sampling weights are required for the sample to be representative, but here we ignore the weights for simplicity). To produce one wave of data for the NHANES study, the research staff visit multiple sites in the US, and sample people at each site. Since people who live in the same geographical region tend to be more alike than people living in different regions, the cluster sampling used in NHANES induces correlations in the data. For this reason, it is not fully correct to analyze the NHANES data using a method like GLM that treats the cases as an independent sample.

Below we create a “group” variable corresponding to the distinct combinations of a masked stratum (SDMVSTRA) and a masked sampling cluster (SDMVPSU), and treat this as a single level of “grouping structure” in the data. The reference to “masking” is due to the fact that for confiden-

tiality reasons, the strata and clusters provided in the dataset are not the actual strata and clusters in the survey. See the NHANES data documentation for more details on this point.

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

# Make sure we distinguish all unique values
a = da.SDMVSTRA.unique().size * da.SDMVPSU.unique().size
b = da.group.unique().size
assert(a == b)
```

When data are collected in groups, it is usually the case that units within a group are more similar than units in different groups. Not accounting for this similarity has several consequences, most notably that the standard errors and other quantities needed for statistical inference are not correct. Below we consider several approaches for addressing this issue.

## 1.2 Fixed effects analysis

One way to account for the relationships induced by cluster sampling is to include “fixed effects” for each group, i.e. to include the group as a covariate in the model. Since group is categorical, it will automatically be converted into a series of indicator variables (omitting one group as the reference). This approach can be useful, especially if there are relatively few groups that are large. But when there are a large number of small groups, the model parameters can be highly inaccurate (technically, “inconsistent”) when using fixed effects analysis. This is called the “Neyman-Scott” problem.

Below we will focus on regression models in which the dependent variable is a measure of poverty, INDFMPIR. This is the ratio between a household’s income and the poverty threshold. Since it is a ratio, it makes sense to logarithmically transform it.

Below we fit a fixed effects model using OLS:

```
[5]: da = da.loc[da.INDFMPIR > 0, :]
da["logINDFMPIR"] = np.log(da.INDFMPIR)

model = sm.OLS.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx + RIDRETH1x +_
    ↪C(group)",
                             data=da)
result = model.fit()
print(result.summary())
```

OLS Regression Results			
=====			
Dep. Variable:	logINDFMPIR	R-squared:	0.105
Model:	OLS	Adj. R-squared:	0.098
Method:	Least Squares	F-statistic:	15.02
Date:	Mon, 17 Feb 2020	Prob (F-statistic):	2.08e-83
Time:	10:14:24	Log-Likelihood:	-5580.7
No. Observations:	4499	AIC:	1.123e+04
Df Residuals:	4463	BIC:	1.146e+04
Df Model:	35		
Covariance Type:	nonrobust		

[0.025      0.975]		coef	std err	t	P> t
-----					
-----					
Intercept		0.1231	0.094	1.316	0.188
-0.060	0.307				
RIAGENDRx[T.Male]		0.0538	0.025	2.138	0.033
0.004	0.103				
RIDRETH1x[T.NonHispanicBlack]		0.3213	0.050	6.471	0.000
0.224	0.419				
RIDRETH1x[T.NonHispanicWhite]		0.6236	0.044	14.022	0.000
0.536	0.711				
RIDRETH1x[T.Other]		0.4937	0.051	9.648	0.000
0.393	0.594				
RIDRETH1x[T.OtherHispanic]		0.1968	0.052	3.808	0.000
0.095	0.298				
C(group) [T.1192]		0.0436	0.116	0.378	0.706
-0.183	0.270				
C(group) [T.1201]		0.0598	0.105	0.572	0.568
-0.145	0.265				
C(group) [T.1202]		0.1178	0.115	1.028	0.304
-0.107	0.343				
C(group) [T.1211]		-0.0300	0.102	-0.294	0.769
-0.230	0.170				
C(group) [T.1212]		0.2200	0.103	2.143	0.032
0.019	0.421				
C(group) [T.1221]		0.0180	0.105	0.172	0.863
-0.188	0.224				
C(group) [T.1222]		0.1826	0.105	1.743	0.081
-0.023	0.388				
C(group) [T.1231]		0.1512	0.102	1.486	0.137
-0.048	0.351				
C(group) [T.1232]		0.1741	0.106	1.637	0.102
-0.034	0.382				
C(group) [T.1241]		0.0831	0.097	0.852	0.394
-0.108	0.274				
C(group) [T.1242]		0.2913	0.111	2.614	0.009
0.073	0.510				
C(group) [T.1251]		-0.3155	0.097	-3.238	0.001
-0.506	-0.124				
C(group) [T.1252]		-0.1569	0.099	-1.583	0.113
-0.351	0.037				
C(group) [T.1261]		-0.1236	0.105	-1.172	0.241
-0.330	0.083				
C(group) [T.1262]		-0.2402	0.102	-2.353	0.019
-0.440	-0.040				

C(group) [T.1271]	0.1410	0.103	1.372	0.170
-0.060      0.342				
C(group) [T.1272]	0.3905	0.102	3.841	0.000
0.191      0.590				
C(group) [T.1281]	-0.0072	0.102	-0.071	0.943
-0.207      0.193				
C(group) [T.1282]	0.4178	0.110	3.790	0.000
0.202      0.634				
C(group) [T.1291]	0.1888	0.105	1.801	0.072
-0.017      0.394				
C(group) [T.1292]	0.2376	0.105	2.259	0.024
0.031      0.444				
C(group) [T.1301]	0.0506	0.103	0.493	0.622
-0.151      0.252				
C(group) [T.1302]	-0.0036	0.102	-0.036	0.971
-0.203      0.196				
C(group) [T.1311]	0.1389	0.102	1.361	0.174
-0.061      0.339				
C(group) [T.1312]	-0.1156	0.100	-1.162	0.245
-0.311      0.080				
C(group) [T.1321]	0.1175	0.102	1.154	0.249
-0.082      0.317				
C(group) [T.1322]	-0.3259	0.098	-3.334	0.001
-0.518      -0.134				
C(group) [T.1331]	0.2183	0.113	1.923	0.055
-0.004      0.441				
C(group) [T.1332]	0.2364	0.103	2.295	0.022
0.034      0.438				
RIDAGEYR	0.0005	0.001	0.691	0.489
-0.001      0.002				
=====				
Omnibus:	1304.756	Durbin-Watson:	2.003	
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4742.129	
Skew:	-1.417	Prob(JB):	0.00	
Kurtosis:	7.155	Cond. No.	1.76e+03	
=====				

#### Warnings:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 1.76e+03. This might indicate that there are strong multicollinearity or other numerical problems.

There are many rows in the above output for the fixed effects parameters for groups. As with any categorical covariate, one group was dropped and used as the reference group. The parameter estimates and p-values in the table above are contrasts relative to the reference group. Since the reference group was chosen arbitrarily, it is not very meaningful to consider the p-values.

### 1.3 GEE

GEE is another method for analyzing a clustered sample. In GEE, the groups are viewed as inducing a correlation between observations in the same group. Thus, instead of modeling the impact of the groups through the mean structure, as in fixed effects analysis, the group effect is treated as a form of correlation (dependence) structure.

To begin, we fit a series of “marginal” mean structure models using OLS and GEE. The term “marginal” here refers to the fact that the parameter estimates reflect average effects over the groups, as opposed to being conditional on a group. For example, if we are interested in the difference between women and men, and we have an indicator variable female for female sex, with coefficient  $b$ , then  $b$  represents the average difference between a female and a male, who could be either in the same group, or in different groups.

The models below use age, gender, and ethnicity to predict family income. We first fit a marginal mean structure model using linear least squares (OLS), ignoring the “group” information.

```
[6]: model = sm.OLS.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx + RIDRETH1x",  
    ↪data=da)  
result = model.fit()  
print(result.summary())
```

```
OLS Regression Results  
=====
```

Dep. Variable:	logINDFMPIR	R-squared:	0.063
Model:	OLS	Adj. R-squared:	0.062
Method:	Least Squares	F-statistic:	50.50
Date:	Mon, 17 Feb 2020	Prob (F-statistic):	2.19e-60
Time:	10:14:24	Log-Likelihood:	-5684.4
No. Observations:	4499	AIC:	1.138e+04
Df Residuals:	4492	BIC:	1.143e+04
Df Model:	6		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t
[0.025	0.975]			
-----				
Intercept	0.2342	0.048	4.886	0.000
0.140	0.328			
RIAGENDRx[T.Male]	0.0503	0.026	1.966	0.049
0.000	0.101			
RIDRETH1x[T.NonHispanicBlack]	0.2447	0.042	5.859	0.000
0.163	0.327			
RIDRETH1x[T.NonHispanicWhite]	0.5781	0.038	15.298	0.000
0.504	0.652			
RIDRETH1x[T.Other]	0.5167	0.045	11.392	0.000
0.428	0.606			
RIDRETH1x[T.OtherHispanic]	0.2016	0.047	4.304	0.000

0.110	0.293				
RIDAGEYR		-5.34e-05	0.001	-0.072	0.942
-0.001	0.001				

```
=====
```

Omnibus:	1273.517	Durbin-Watson:	2.009
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4437.847
Skew:	-1.397	Prob(JB):	0.00
Kurtosis:	6.983	Cond. No.	308.

```
=====
```

Warnings:

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

Next we fit the model using GEE with an “independent” working dependence structure. Here, the working covariance structure states that observations within a cluster are independent. As noted above, the working dependence structure does not need to be correct. However to the extent that there is correlation within groups, this fit will be less efficient than one that accounts for the dependence.

```
[7]: model = sm.GEE.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx + RIDRETH1x",
    →groups="group",
    data=da)
result = model.fit()
print(result.summary())
print(result.cov_struct.summary())
```

#### GEE Regression Results

```
=====
```

```
===
```

Dep. Variable:	logINDFMPIR	No. Observations:
4499		
Model:	GEE	No. clusters:
30		
Method:	Generalized	Min. cluster size:
93		
	Estimating Equations	Max. cluster size:
210		
Family:	Gaussian	Mean cluster size:
150.0		
Dependence structure:	Independence	Num. iterations:
2		
Date:	Mon, 17 Feb 2020	Scale:
0.734		
Covariance type:	robust	Time:
10:14:24		

```
=====
```

```
=====
```

		coef	std err	z	P> z
[0.025	0.975]				
-----					
Intercept		0.2342	0.077	3.032	0.002
0.083	0.386				
RIAGENDRx[T.Male]		0.0503	0.019	2.654	0.008
0.013	0.088				
RIDRETH1x[T.NonHispanicBlack]		0.2447	0.078	3.134	0.002
0.092	0.398				
RIDRETH1x[T.NonHispanicWhite]		0.5781	0.067	8.626	0.000
0.447	0.709				
RIDRETH1x[T.Other]		0.5167	0.093	5.532	0.000
0.334	0.700				
RIDRETH1x[T.OtherHispanic]		0.2016	0.082	2.446	0.014
0.040	0.363				
RIDAGEYR		-5.34e-05	0.001	-0.049	0.961
-0.002	0.002				
=====					
Skew:		-1.3972	Kurtosis:		3.9831
Centered skew:		-1.4205	Centered kurtosis:		4.1579
=====					
Observations within a cluster are modeled as being independent.					

Note that the estimated regression parameters in the OLS and independence GEE are identical (this will always be the case), but the standard errors differ. In general, the standard errors in the OLS fit will be too small, and GEE will give larger, and more correct standard errors. Here we see that the standard errors are around twice as large in GEE compared to OLS.

```
[8]: model = sm.GEE.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx + RIDRETH1x",
    →groups="group",
                                cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
print(result.summary())
print(result.cov_struct.summary())
```

#### GEE Regression Results

```
=====
===
Dep. Variable:          logINDFMPIR   No. Observations:
4499
Model:                  GEE           No. clusters:
30
Method:                 Generalized   Min. cluster size:
93
                        Estimating Equations   Max. cluster size:
210
Family:                 Gaussian       Mean cluster size:
```



```

150.0
Dependence structure:      Exchangeable    Num. iterations:
8
Date:                      Mon, 17 Feb 2020    Scale:
0.735
Covariance type:          robust    Time:
10:14:24
=====
=====

```

	coef	std err	z	P> z
[0.025      0.975]				
-----				
Intercept	0.2009	0.087	2.301	0.021
0.030      0.372				
RIAGENDRx[T.Male]	0.0535	0.018	2.894	0.004
0.017      0.090				
RIDRETH1x[T.NonHispanicBlack]	0.3091	0.076	4.083	0.000
0.161      0.458				
RIDRETH1x[T.NonHispanicWhite]	0.6161	0.070	8.741	0.000
0.478      0.754				
RIDRETH1x[T.Other]	0.4971	0.088	5.673	0.000
0.325      0.669				
RIDRETH1x[T.OtherHispanic]	0.1961	0.088	2.220	0.026
0.023      0.369				
RIDAGEYR	0.0004	0.001	0.420	0.674
-0.002      0.002				
=====				
Skew:	-1.3887	Kurtosis:		3.9525
Centered skew:	-1.4180	Centered kurtosis:		4.1571
=====				

```

The correlation between two observations in the same cluster is 0.043

```

Next we use another type of working correlation structure called “exchangeable” that is suitable for use with grouped data. The exchangeable correlation structure stipulates that any two observations in the same group have the same level of correlation between them. This common correlation parameter can be estimated from the data. Any two observations in different groups are modeled as being independent. The correlation between two observations in the same group is called the “intraclass correlation coefficient”, or “ICC”. We are able to see here that the estimated ICC is around 0.04. The ICC ranges from 0 to 1, with an ICC of zero meaning that the groups are irrelevant to the outcome, and an ICC of one meaning that the outcome is totally determined by the groups. While an ICC of 0.04 seems small, for reasons discussed below it is large enough to have a noticeable impact, especially when the groups are large, as is the case here.

The presence of positive correlations within groups reduces the amount of information in the data for some purposes, and enhances the information for other purposes. The impact of within-cluster dependence can be better understood by considering two quantities called the “design effect” and the “effective sample size”. These values quantify the loss of information when the goal is to estimate a mean. When the ICC is positive, the effective sample size represents a hypo-

thetical sample size (smaller than the nominal sample size) such that the precision for estimating a mean using an independent sample of size equal to the effective sample size is the same as would be obtained using the actual (dependent) sample.

For example, if the ICC is zero, then the effective sample size is the same as the nominal sample size (the number of observed data values). If the ICC is close to 1, then the effective sample size will become close to the number of groups (irrespective of how many observations are made within a group). The design effect and effective sample size are calculated in the cell below.

```
[9]: icc = result.cov_struct.dep_params
print("ICC = %f\n" % icc)
n = da.groupby("group").size().mean()
print("Average cluster size = %f\n" % n)
de = 1 + (n - 1) * icc
print("Design effect = %f\n" % de)
ess = model.nobs / de
print("Number of observations = %f\n" % model.nobs)
print("Effective sample size = %f\n" % ess)
```

ICC = 0.042893

Average cluster size = 149.966667

Design effect = 7.389598

Number of observations = 4499.000000

Effective sample size = 608.828778

In this example, the design effect is around 7, meaning that the ICC of around 0.04 reduces the information in our sample by around a factor of 7. Since standard errors scale with the square root of the sample size, this means that the standard errors of the parameters may increase by a factor of 2-3 when properly accounting for the dependence structure. This is roughly consistent with the results shown above, comparing the OLS standard errors to either of the GEE standard errors. Note however that regression parameters do not behave exactly like means, so the design effect is only a rough indication of how the standard errors for regression parameters are impacted by dependence due to the clusters.

One way to think about the intraclass correlation is to imagine that it is driven by unobserved cluster-level covariates. That is, there are likely to be characteristics shared by all people in the same cluster that we do not observe. Our failure to account for these covariates in the mean structure model induces intra-cluster dependence. In fact, we do have access to many additional covariates in NHANES that could be included in the regressions, and we see below that as we include fewer covariates the ICC tends to increase, and as we include more covariates, the ICC tends to decrease:

```
[10]: model = sm.GEE.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx",
    →groups="group",
                                cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
```

```
print(result.summary())
print(result.cov_struct.summary())
```

```

                                GEE Regression Results
=====
===
Dep. Variable:                logINDFMPIR    No. Observations:
4499
Model:                        GEE           No. clusters:
30
Method:                      Generalized    Min. cluster size:
93
                                Estimating Equations    Max. cluster size:
210
Family:                      Gaussian       Mean cluster size:
150.0
Dependence structure:        Exchangeable    Num. iterations:
7
Date:                        Mon, 17 Feb 2020    Scale:
0.782
Covariance type:            robust           Time:
10:14:24
=====
=====
                                coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
Intercept                   0.5241      0.073      7.218      0.000      0.382
0.666
RIAGENDRx[T.Male]          0.0662      0.019      3.521      0.000      0.029
0.103
RIDAGEYR                   0.0015      0.001      1.291      0.197     -0.001
0.004
=====
Skew:                      -1.3793    Kurtosis:                      3.7642
Centered skew:             -1.4172    Centered kurtosis:             4.0956
=====
The correlation between two observations in the same cluster is 0.055

```

```
[11]: model = sm.GEE.from_formula("logINDFMPIR ~ RIDAGEYR + RIAGENDRx + RIDRETH1x +
    ↳DMDMARTLx + DMDEDUC2x", groups="group",
                                cov_struct=sm.cov_struct.Exchangeable(), data=da)
result = model.fit()
print(result.summary())
print(result.cov_struct.summary())
```

# GEE Regression Results

```

=====
===
Dep. Variable:          logINDFMPIR   No. Observations:
4497
Model:                  GEE           No. clusters:
30
Method:                 Generalized    Min. cluster size:
92
                        Estimating Equations   Max. cluster size:
210
Family:                 Gaussian       Mean cluster size:
149.9
Dependence structure:   Exchangeable   Num. iterations:
9
Date:                   Mon, 17 Feb 2020   Scale:
0.600
Covariance type:        robust           Time:
10:14:25
=====
=====

```

		coef	std err	z	P> z
[0.025	0.975]				
-----					
-----					
Intercept		0.5822	0.087	6.714	0.000
0.412	0.752				
RIAGENDRx[T.Male]		0.0497	0.019	2.669	0.008
0.013	0.086				
RIDRETH1x[T.NonHispanicBlack]		0.1300	0.059	2.192	0.028
0.014	0.246				
RIDRETH1x[T.NonHispanicWhite]		0.3042	0.057	5.374	0.000
0.193	0.415				
RIDRETH1x[T.Other]		0.1769	0.073	2.431	0.015
0.034	0.320				
RIDRETH1x[T.OtherHispanic]		0.1079	0.067	1.622	0.105
-0.022	0.238				
DMDMARTLx[T.Married]		0.3286	0.038	8.584	0.000
0.254	0.404				
DMDMARTLx[T.NeverMarried]		0.0467	0.057	0.826	0.409
-0.064	0.158				
DMDMARTLx[T.Partnered]		0.0315	0.069	0.455	0.649
-0.104	0.167				
DMDMARTLx[T.Separated]		-0.0986	0.073	-1.356	0.175
-0.241	0.044				
DMDMARTLx[T.Widowed]		0.0872	0.051	1.709	0.087
-0.013	0.187				
DMDEDUC2x[T.HS]		-0.5730	0.035	-16.458	0.000

-0.641	-0.505				
DMDEDUC2x[T.Lt9]		-1.0331	0.049	-20.946	0.000
-1.130	-0.936				
DMDEDUC2x[T.SomeCollege]		-0.3640	0.036	-10.144	0.000
-0.434	-0.294				
DMDEDUC2x[T.g9-11]		-0.8189	0.042	-19.430	0.000
-0.902	-0.736				
RIDAGEYR		0.0021	0.001	2.076	0.038
0.000	0.004				

```

=====
Skew:                -1.5643    Kurtosis:                5.3198
Centered skew:       -1.5705    Centered kurtosis:      5.5112
=====
The correlation between two observations in the same cluster is 0.020

```

## 1.4 Grouped data with nonlinear mean structures

The GEE analysis above uses a linear mean structure. In this setting, GEE is equivalent to an older procedure called “Generalized Least Squares” (GLS). While GEE and GLS estimate the regression parameters in the same way, GEE carries out inference (standard error calculations) using a robust approach that gives correct results even if the working dependence is wrong. This robust inference also has a history predating GEE, and is alternatively known as “Huber-White” or “sandwich” covariance estimation.

GEE can be used with any GLM link function and family, giving rise to nonlinear mean relationships and non-constant mean/variance relationships. For example, there are GEE analogues of logistic and Poisson regression. In the case where the link function is not linear, the fitting algorithm is not least squares, but it can be viewed as the limit of a sequence of least squares fits, and thus may be referred to as “iteratively reweighted least squares”.

To demonstrate GEE with a nonlinear mean structure, we use the smoking status variable in NHANES, which is binary. First, for reference, we fit a binomial GLM (standard logistic regression). This analysis does not account for the cluster (grouping) structure in the data, so has the potential to give misleading results.

```
[12]: da["SMQ020x"] = da.SMQ020.replace({1: "yes", 2: "no", 7: np.nan, 9: np.nan})
dx = da[["SMQ020x", "RIDAGEYR", "RIAGENDRx", "RIDRETH1x", "group"]].dropna()
dx["Smoke"] = (da.SMQ020x == "yes").astype(np.int)

model = sm.GLM.from_formula("Smoke ~ RIDAGEYR + RIAGENDRx",
                           family=sm.families.Binomial(), data=dx)
result = model.fit()
print(result.summary())
```

```

Generalized Linear Model Regression Results
=====
Dep. Variable:          Smoke    No. Observations:          4496
Model:                  GLM      Df Residuals:              4493
Model Family:           Binomial  Df Model:                  2
Link Function:          logit     Scale:                    1.0000

```

```

Method:                IRLS    Log-Likelihood:          -2916.5
Date:                  Mon, 17 Feb 2020    Deviance:              5833.1
Time:                  10:14:25    Pearson chi2:          4.50e+03
No. Iterations:        4
Covariance Type:      nonrobust

```

```

=====
=====
              coef      std err          z      P>|z|      [0.025
-----
-----
Intercept      -1.6100      0.101     -15.944      0.000     -1.808
-1.412
RIAGENDRx[T.Male]  0.8582      0.062      13.739      0.000      0.736
0.981
RIDAGEYR        0.0177      0.002       9.861      0.000      0.014
0.021
=====
=====

```

Next we use GEE to fit the logistic regression while accounting for the groups. We use here the “independence” working covariance, which models the data as being independent, but continues to give meaningful results even if the observations are not independent. Note that the parameter estimates are the same as obtained above using GLM, but the standard errors are slightly different. In this case, the difference in standard errors is very minor, but in other cases the differences can be large.

GEE with the independent working covariance will always give the same parameter estimates as GLM. This tells us that GLM can safely be used with dependent data, as long as only the point estimates are considered. If (as is usually the case), uncertainty is to be assessed, then GEE (or another appropriate approach such as mixed modeling) should be used.

```

[13]: model = sm.GEE.from_formula("Smoke ~ RIDAGEYR + RIAGENDRx",
                                groups="group",
                                family=sm.families.Binomial(), data=dx)

result = model.fit()
print(result.summary())
print(result.cov_struct.summary())

```

#### GEE Regression Results

```

=====
=====
Dep. Variable:          Smoke    No. Observations:
4496
Model:                  GEE      No. clusters:
30
Method:                 Generalized    Min. cluster size:
93
                        Estimating Equations    Max. cluster size:

```

```

210
Family:                      Binomial    Mean cluster size:
149.9
Dependence structure:      Independence  Num. iterations:
2
Date:                      Mon, 17 Feb 2020  Scale:
1.000
Covariance type:          robust    Time:
10:14:25
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
Intercept          -1.6100      0.122     -13.147      0.000     -1.850
-1.370
RIAGENDRx[T.Male]    0.8582      0.072     11.986      0.000      0.718
0.998
RIDAGEYR            0.0177      0.002     10.048      0.000      0.014
0.021
=====
Skew:                0.2882    Kurtosis:                -1.6464
Centered skew:        0.2654    Centered kurtosis:        -1.5526
=====
Observations within a cluster are modeled as being independent.

```

As with the linear GEE, it is often desirable to introduce a non-independent covariance structure into the model. Doing so provides us with some insight into the form of the dependence structure, and also has the potential to give more accurate point estimates and tighter uncertainty bounds.

The estimates and standard errors for the fit using the exchangeable covariance structure are very similar to what was obtained above using the independence working dependence structure. The estimated ICC is small, but as discussed above, ICC's around 0.02 can have sizeable impact on the parameter estimates and standard errors if the clusters are large, as is the case here.

```

[14]: model = sm.GEE.from_formula("Smoke ~ RIDAGEYR + RIAGENDRx",
                                groups="group",
                                family=sm.families.Binomial(),
                                cov_struct=sm.cov_struct.Exchangeable(), data=dx)

result = model.fit()
print(result.summary())
print(result.cov_struct.summary())

```

#### GEE Regression Results

```

=====
===
Dep. Variable:          Smoke    No. Observations:

```

```

4496
Model:                      GEE    No. clusters:
30
Method:                     Generalized  Min. cluster size:
93
                        Estimating Equations  Max. cluster size:
210
Family:                     Binomial    Mean cluster size:
149.9
Dependence structure:      Exchangeable  Num. iterations:
7
Date:                      Mon, 17 Feb 2020  Scale:
1.000
Covariance type:           robust    Time:
10:14:25
=====
=====
              coef      std err          z      P>|z|      [0.025
0.975]
-----
-----
Intercept          -1.6044      0.121     -13.237      0.000     -1.842
-1.367
RIAGENDRx[T.Male]    0.8620      0.070     12.259      0.000      0.724
1.000
RIDAGEYR            0.0170      0.002     10.132      0.000      0.014
0.020
=====
Skew:              0.2888    Kurtosis:              -1.6528
Centered skew:     0.2659    Centered kurtosis:    -1.5585
=====
The correlation between two observations in the same cluster is 0.023

```

Note that while these models are estimated using the GEE framework, the interpretation of the effects is the same as in a GLM. In this case, we are working with a logistic model, thus the parameter estimates shown above, say for male gender, reflect the amount by which the log odds for smoking is greater when comparing a male to a female of the same age. More specifically, in the context of dependent, or grouped data, as we have here, this parameter should be interpreted as reflecting the average difference of log odds for smoking when comparing a male to a female of the same age, irrespective of whether they are in the same or in different groups. This follows from the fact that the mean structure estimated by a GEE is “marginal”. A related but different regression technique is “multilevel modeling”, in which case this parameter should be interpreted as reflecting the contrast between a male and a female of the same age, who also fall into the same group.

As discussed above in the linear mean structure example, including more covariates will generally reduce the ICC, but usually not to the point where it is negligible.



```
[15]: model = sm.GEE.from_formula("Smoke ~ RIDAGEYR + RIAGENDRx + RIDRETH1x",
                                groups="group",
                                family=sm.families.Binomial(),
                                cov_struct=sm.cov_struct.Exchangeable(), data=dx)

result = model.fit()
print(result.summary())
print(result.cov_struct.summary())
```

#### GEE Regression Results

```
=====
===
Dep. Variable:          Smoke    No. Observations:
4496
Model:                GEE    No. clusters:
30
Method:              Generalized    Min. cluster size:
93                                Estimating Equations    Max. cluster size:
210
Family:              Binomial    Mean cluster size:
149.9
Dependence structure: Exchangeable    Num. iterations:
9
Date:                Mon, 17 Feb 2020    Scale:
1.000
Covariance type:      robust    Time:
10:14:25
=====
=====
```

	coef	std err	z	P> z
[0.025    0.975]				
-----				
Intercept	-1.6907	0.124	-13.625	0.000
-1.934    -1.448				
RIAGENDRx[T.Male]	0.8729	0.071	12.348	0.000
0.734    1.011				
RIDRETH1x[T.NonHispanicBlack]	0.1704	0.128	1.327	0.185
-0.081    0.422				
RIDRETH1x[T.NonHispanicWhite]	0.4128	0.111	3.718	0.000
0.195    0.630				
RIDRETH1x[T.Other]	-0.2861	0.175	-1.637	0.102
-0.629    0.056				
RIDRETH1x[T.OtherHispanic]	0.1439	0.136	1.057	0.290
-0.123    0.411				
RIDAGEYR	0.0158	0.002	9.506	0.000
0.013    0.019				

```
=====
Skew:                                0.2741    Kurtosis:                                -1.6094
Centered skew:                        0.2543    Centered kurtosis:                        -1.5233
=====
```

The correlation between two observations in the same cluster is 0.014

In order to show that the ICC's we obtaining here are not artifacts, below we generate a completely random grouping variable with the same number of levels as the actual grouping variable, and show the estimated ICC values that are obtained.

```
[16]: for k in range(10):
        dx["group2"] = np.random.randint(0, 30, dx.shape[0])
        model = sm.GEE.from_formula("Smoke ~ RIDAGEYR + RIAGENDRx",
                                    groups="group2",
                                    family=sm.families.Binomial(),
                                    cov_struct=sm.cov_struct.Exchangeable(),
        →data=dx)
        result = model.fit()
        print(result.cov_struct.summary())
```

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

## 1.5 Subject-level repeated measures

Above we considered dependence that resulted from cluster sampling. Another common reason that dependence occurs is having repeated measures on individuals. This happens, for example, in a longitudinal study, where variables of interest are measured on individuals repeatedly over time. Since NHANES is a cross-sectional study, there are no repeated longitudinal measures. However, there are repeated measures of the blood pressure variables (systolic and diastolic blood pressure). Each of these variables is measured on every subject 2-4 times in short succession. The reason that this is done is that blood pressure can be quite variable, and in particular can be high on an initial measurement due to subjects being anxious. Thus the second and subsequent blood pressure measurements tend to be slightly lower than the first.

First we need to do some data management. The NHANES file that we are working with is in "wide form" (one row per subject), but for this analysis we need one row per blood pressure observation. The Pandas melt method with some subsequent cleanup accomplishes this data restructuring.

```
[17]: idv = ("SEQN", "BMXBMI", "SDMVSTRA", "SDMVPSU", "RIAGENDR", "RIDAGEYR",
        →"RIDRETH1")
        dx = pd.melt(da, id_vars=idv, value_vars=("BPXSY1", "BPXSY2"),
```

```

        var_name="Time", value_name="SBP")
dx["Time"] = dx.Time.apply(lambda x : x.replace("BPXSY", ""))
dx = dx.sort_values(by="SEQN")

dx["RIAGENDRx"] = dx.RIAGENDRx.replace({1: "Male", 2: "Female"})

dx["RIDRETH1x"] = dx.RIDRETH1.replace({1: "MexicanAmerican", 2: "OtherHispanic",
3: "NonHispanicWhite", 4: "NonHispanicBlack", 5: "Other"})

```

Next we fit a GEE with linear mean structure to the systolic blood pressure values (two observations per subject). The exchangeable correlation structure allows us to estimate the ICC (i.e. how correlated are the two repeated observations within a subject). Note that we are now ignoring the correlations induced by the survey structure.

```

[18]: dx["RIDAGEYRm18"] = dx.RIDAGEYR - 18

model = sm.GEE.from_formula("SBP ~ RIDAGEYRm18 * RIAGENDRx + RIDRETH1x + Time",
                           cov_struct=sm.cov_struct.Exchangeable(),
                           groups="SEQN", data=dx)

result = model.fit()
print(result.summary())
print(result.cov_struct.summary())

```

#### GEE Regression Results

```

=====
===
Dep. Variable:          SBP    No. Observations:
8998
Model:                GEE    No. clusters:
4499
Method:              Generalized    Min. cluster size:
2                                Estimating Equations    Max. cluster size:
2                                Gaussian    Mean cluster size:
Family:              2.0
Dependence structure: Exchangeable    Num. iterations:
2
Date:                Mon, 17 Feb 2020    Scale:
252.205
Covariance type:      robust    Time:
10:14:29
=====
=====

```

	coef	std err	z	P> z
[0.025      0.975]				
-----				
-----				
Intercept	106.4096	0.722	147.362	0.000

```

104.994      107.825
RIAGENDRx[T.Male]          10.4795      0.798      13.140      0.000
8.916        12.043
RIDRETH1x[T.NonHispanicBlack]      3.7896      0.766      4.945      0.000
2.288        5.292
RIDRETH1x[T.NonHispanicWhite]     -2.4811      0.649     -3.824      0.000
-3.753       -1.209
RIDRETH1x[T.Other]          -1.7459      0.790     -2.209      0.027
-3.295       -0.197
RIDRETH1x[T.OtherHispanic]     -0.2426      0.810     -0.299      0.765
-1.831        1.346
Time[T.2]                -0.7086      0.076     -9.370      0.000
-0.857       -0.560
RIDAGEYRm18              0.5658      0.019     29.338      0.000
0.528        0.604
RIDAGEYRm18:RIAGENDRx[T.Male]    -0.2210      0.027     -8.038      0.000
-0.275       -0.167
=====
Skew:                    0.8057      Kurtosis:                    2.2353
Centered skew:          -0.0000      Centered kurtosis:          1.8996
=====
The correlation between two observations in the same cluster is 0.950

```

It is also possible to have multiple levels of grouping structure in a GEE. Here, we have two repeated measures clustered within each individual, and the individuals are in turn clustered within the survey groups. We can use a Nested dependence structure to capture both levels of correlation.

```
[19]: dx["group"] = 10 * dx.SDMVSTRA + dx.SDMVPSU
```

```

model = sm.GEE.from_formula("SBP ~ RIDAGEYRm18 * RIAGENDRx + RIDRETH1x",
                             dep_data="0 + SEQN",
                             cov_struct=sm.cov_struct.Nested(),
                             groups="group", data=dx)

result = model.fit()
print(result.summary())
print(result.cov_struct.summary())

```

#### GEE Regression Results

```

=====
===
Dep. Variable:          SBP      No. Observations:
8998
Model:                  GEE      No. clusters:
30
Method:                 Generalized      Min. cluster size:
186
                        Estimating Equations      Max. cluster size:
420

```

```

Family:                                Gaussian    Mean cluster size:
299.9
Dependence structure:                  Nested    Num. iterations:
10
Date:                                Mon, 17 Feb 2020    Scale:
252.417
Covariance type:                      robust    Time:
10:14:30
=====
=====

```

	coef	std err	z	P> z
[0.025      0.975]				
Intercept	106.2401	0.912	116.433	0.000
104.452      108.028				
RIAGENDRx[T.Male]	10.4493	0.622	16.809	0.000
9.231      11.668				
RIDRETH1x[T.NonHispanicBlack]	3.4551	0.965	3.579	0.000
1.563      5.347				
RIDRETH1x[T.NonHispanicWhite]	-2.6727	0.836	-3.197	0.001
-4.311      -1.034				
RIDRETH1x[T.Other]	-1.4358	0.862	-1.665	0.096
-3.126      0.254				
RIDRETH1x[T.OtherHispanic]	-0.9758	0.864	-1.129	0.259
-2.669      0.718				
RIDAGEYRm18	0.5614	0.023	24.220	0.000
0.516      0.607				
RIDAGEYRm18:RIAGENDRx[T.Male]	-0.2202	0.020	-10.759	0.000
-0.260      -0.180				

```

=====
Skew:                                0.8124    Kurtosis:                                2.2435
Centered skew:                      0.8114    Centered kurtosis:                        2.2105
=====

```

	Variance
group	3.094392
SEQN	235.982898
Residual	13.339335

The fitted nested covariance structure is expressed in terms of the variance contributed by each level of nesting. This corresponds to a “variance components model” of the form

$$y_{ijkl} = a_i + b_{ij} + c_{ijk} + \epsilon_{ijkl}$$

Above,  $a_i$ ,  $b_{ij}$ ,  $c_{ijk}$ , and  $\epsilon_{ijkl}$  are all random with mean zero, and with variances  $\tau_a^2$ ,  $\tau_b^2$ ,  $\tau_c^2$ , and  $\sigma^2$  respectively.

## 1.6 Simulation of data for GEE analysis

Sometimes it is useful to apply a regression approach to simulated data. This can be applied in power and sample size assessment, and in evaluating the performance of a statistical methodology. Simulation of data for GEE regression presents a unique challenge because GEE is not based on a complete model for the data. In a GEE, the mean structure is fully specified, and in most cases the marginal distribution of each observation is fully specified (however this is not true when using the quasi-likelihood families). The dependence structure is only specified through a working structure that does not need to be correct, and even if it is correct, it does not define the joint distribution of the data (except in very special cases like the Gaussian model).

There are various ways to resolve this challenge. Below we will demonstrate how a copula approach can be used to simulate data for a GEE analysis. The basic idea is that if  $u_1, u_2$  are two random variables that may be dependent, such that  $u_1$  and  $u_2$  are both marginally uniformly distributed, then for any cumulative distribution functions  $F_1, F_2$ , the random variables  $y_1 = F_1^{-1}(u_1)$  and  $y_2 = F_2^{-1}(u_2)$  are distributed according to  $F_1$  and  $F_2$ . In a GEE, we know the marginal distributions (except in the quasi-likelihood case that we will not cover here), so  $F_1$ , etc. are always known. Thus, we can induce dependence between  $y_1$  and  $y_2$  by simulating  $u_1$  and  $u_2$  to be dependent and (marginally) uniform. Note that this approach works for any dimension, we are only using two components here for illustration.

One way to simulate dependent  $u_j$  that are marginally uniform is to simulate dependent  $z_j$  that are marginally standard Gaussian, then define  $u_j = \Phi^{-1}(z_j)$ , where  $\Phi$  is the standard normal CDF. This approach is flexible because it is easy to generate Gaussian random vectors with a variety of different covariance structures, such that the components are marginally standard normal. This is called the “Gaussian copula” and is used in the illustration below.

```
[20]: # Number of groups
n = 1000

# Number of observations per group
q = 10

# The autocorrelation in the Gaussian copula
a = 0.7

from scipy.stats.distributions import poisson, norm

# The latent variables in the Gaussian copula
z = np.random.normal(size=(n, q))
for j in range(1, q):
    z[:, j] = a*z[:, j-1] + np.sqrt(1 - a**2)*z[:, j]
u = norm.cdf(z)

# The covariates
x1 = np.random.normal(size=(n, q))
x2 = np.random.normal(size=(n, q))

# The mean parameters for the marginal distributions
lpr = x1 - 0.5*x2
```

```

expval = np.exp(lpr)

# The response values. These are marginally Poisson with the specified means.
y = np.zeros((n, q))
for i in range(n):
    for j in range(q):
        y[i, j] = poisson.ppf(u[i, j], expval[i, j])

idv = np.outer(np.arange(n), np.ones(q))
time = np.outer(np.ones(n), np.arange(q))

df = pd.DataFrame({"y": y.flat, "x1": x1.flat, "x2": x2.flat,
                  "grp": idv.flat, "time": time.flat})

model = sm.GEE.from_formula("y ~ x1 + x2", groups="grp",
                            family=sm.families.Poisson(),
                            time=df.time,
                            cov_struct=sm.cov_struct.Stationary(max_lag=5),
                            data=df)
result = model.fit()

print(result.summary())
print(result.cov_struct.summary())

```

#### GEE Regression Results

```

=====
===
Dep. Variable:                y      No. Observations:
10000
Model:                        GEE      No. clusters:
1000
Method:                        Generalized  Min. cluster size:
10                                     Estimating Equations  Max. cluster size:
10
Family:                        Poisson    Mean cluster size:
10.0
Dependence structure:          Stationary  Num. iterations:
6
Date:                          Mon, 17 Feb 2020  Scale:
1.000
Covariance type:                robust    Time:
10:14:35
=====

```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	0.0175	0.018	0.947	0.344	-0.019	0.054

x1	0.9966	0.007	136.743	0.000	0.982	1.011
x2	-0.4967	0.006	-89.795	0.000	-0.508	-0.486

```
=====
```

Skew:	0.6503	Kurtosis:	6.9745
Centered skew:	0.3485	Centered kurtosis:	6.4925

```
=====
```

	Lag	Cov
0	0	0.000000
1	1	0.582277
2	2	0.391955
3	3	0.252930
4	4	0.180244
5	5	0.114783

## 1.7 Other applications of GEE

Here we have shown how GEE can be used with grouped data, including multilevel (nested) grouped data. We have also seen an application involving data that were simulated as a time series. GEE is widely used for grouped data, longitudinal data, time series, and spatial data. In principle, covariance structures can be defined for most applications involving dependent data. However in practice most people use the familiar dependence structures illustrated here.

Finally, note that most of the GEE support within Python Statsmodels is implemented in [this source file](#). You do not need to read or understand this source code to be able to use GEE regression in Statsmodels. But more advanced users may want to understand how the fitting is implemented. GEE constitutes a fairly mature and well-established methodology. Therefore, the results of fitting a GEE to a data set should be essentially identical in all packages (Python Statsmodels, R, Stata, SAS, etc.). Statsmodels uses a large number of unit tests (e.g. [here](#)) to ensure that the results are correct and consistent with other packages.