nhanes_glm

February 9, 2020

Generalized Linear Models (GLM), a case study with the NHANES data

This notebook demonstrates a framework for regression analysis called "generalized linear modeling", or GLM. The focus here will be on fitting these models to data using Python statistical modeling libraries in the Jupyter notebook environment. This notebook is primarily presented as a case study using the NHANES data.

Most of the GLM 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 the software to fit GLM's. But more advanced users may want to understand how the fitting is implemented. GLM's constitute a fairly mature and well-established methodology. Therefore, the results of fitting a GLM 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.

Note that the NHANES data were collected as a designed survey, and in general should be analyzed as such. This means that survey design information such as weights, strata, and clusters should be accounted for in any analysis using NHANES. But to introduce how generalized linear models are used with independent data or with convenience samples, we will not incorporate the survey structure of the NHANES sample into the analyses conducted here.

As with our previous work, we will be using the Statsmodels library for statistical modeling, the Pandas library for data management, the Seaborn library for graphing, and the Numpy library for numerical calculations.

We begin by importing 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 will load the data. The NHANES study encompasses multiple waves of data collection. Here we will only use the 2015-2016 data. As with most data sets, there are some missing

values in the NHANES files. While many of the methods demonstrated below would handle missing values automatically (at least in a crude way), here we drop up-front all rows with missing values in any of the key variables that we will use in this notebook.

[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 = ["BPXSY1", "RIDAGEYR", "RIAGENDR", "RIDRETH1", "DMDEDUC2", "BMXBMI",␣
 ↪"SMQ020"]
da = da[vars].dropna()
#-

## Mean/variance relationships

# In linear regression, we fit models for the population mean structure
# that have the form

# $$
# E[y | x_1, \ldots, x_p] = b_0 + b_1x_1 + \cdots + b_px_p,
# $$

# where $b_0 + b_1x_1 + \cdots + x_px_p$ is called the *linear
# predictor*.

# Linear regression works best when the population variance structure is
# homoscedastic:

# $$
# {\rm Var}[y | x_1, \ldots, x_p] = \sigma^2.
# $$

# Linear regression models are most commonly fit using the fitting
# procedure called *least squares*.  When using least squares to fit a
# linear regression model, the estimates of the mean structure
# parameters $b_0, \ldots, b_p$ achieve their greatest accuracy when the
# population variance is approximately homoscedastic.  But with a loss
# of efficiency (that in many settings is not large), the mean structure
# parameters remain accurately estimated by least squares even when the
# variance structure is not homoscedastic.

# Generalized linear models (GLMs) are a collection of approaches for
# regression analysis that extend the basic linear model in two ways:

# * The mean structure can be related to the linear prediction via a
# *link function*, e.g. the $\log$ link function:
```

```
# $$
# \log E[y | x_1, \ldots, x_p] = b_0 + b_1x_1 + \cdots b_px_p.
# $$

# In general, the link function $g$ satisfies

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

# * The conditional variance can be a function of the conditional mean,
#   e.g.

# $$
# {\rm Var} [y | x_1, \ldots, x_p] = E[y | x_1, \ldots, x_p].
# $$

# In general, the conditional variance can be expressed as

# $$
# {\rm Var} [y | x_1, \ldots, x_p] = \phi f(E[y | x_1, \ldots, x_p]),
# $$

# Writing $\mu = E[y | x_1, \ldots, x_p]$, we have ${\rm Var} =
# \mu\cdot(1-\mu)$.

# Thus, GLM's explicitly incorporate heteroscedasticicity (non-constant
# variance) through this "mean/variance relationship".  The parameter
# $\phi$ is called the *scale parameter* and will be discussed further
# below.

# Note that we can use the identity link function $g(x) = x$ and the
# constant mean/variance relationship $f(x) = 1$, which gives us back
# the linear model.  Thus, the linear model is one type of GLM.

# Note that linear regression is often used after applying a
# transformation to the dependent variable.  For example, if we log
# transform the dependent variable and then fit a linear regression
# model, we are estimating the mean structure

# $$
# E[\log(y) | x_1, \ldots, x_p] = b_0 + b_1x_1 + \cdots + b_px_p.
# $$

# This is closely related to fitting a GLM with a log link function, but
# is not exactly equivalent to it, since $E[\log(y)]$ is not equal to
# $\log(E[y])$.  Either of these approaches may provide a good model,
```

```
# but the two models that are being fit are not the same.

# ## Limited dependent variables

# Another perspective on GLMs is that they are useful when the dependent
# variable is restricted to lie in a subset of values.  The most common
# such subsets are the non-negative integers, the positive real line,
# and the unit interval.  Specific GLMs that we will discuss below are
# defined in terms of probability distributions that obey these
# constraints.

# ## Logistic regression

# Logistic regression is by far the most commonly used GLM.  It is used
# when the outcome variable is *binary*, meaning that it can take on
# only two distinct values.  The behavior of a binary random value is
# entirely determined by its "success probability", where "success" is a
# generic term for one of the two possible outcomes.  For example, in a
# clinical trial, substantial improvement might be deemed a success,
# with less than substantial improvement being viewed as a failure.  The
# success probability in this setting would be the probability that a
# substantial improvement in symptoms occurs.  Sometimes this is called
# instead the "event probability".

# The distribution of a binary random variable is entirely determined by
# the success probability $p$.  In a regression analysis, we have many
# observations $y_i$ of the dependent variable, and each is accompanied
# with a covariate vector $x_i$.  This covariate vector may determine
# the success probability, so we write $p(x)$ for the success
# probability of an observation with covariate vector $x$.  Thus, for
# each $i$, $y_i$ is a binary random variable with success probability
# $p(x_i)$.

# As discussed above, in a GLM the link function maps the expected value
# of the outcome to the linear predictor.  In logistic regression, the
# link function is the *logit function*, or the *log odds function*
# $\log(p/(1-p)$.  Note that while $p$ is restricted to lie in $[0, 1]$
# the odds $p/(1-p)$ can be any positive real number, and the log odds
# can be any real number.

# Thus, in logistic regression, we have

# $$
# \log \frac{E[y|x_1, \ldots, x_p]}{1 - E[y|x_1, \ldots, x_p]} = b_0
```

[3]:
```
# $$
```

```
# The variance function for logistic regression is determined by the
# fact that for a binary random variable with success probability $p$,
# the variance is $p\cdot(1-p)$.  Thus, the mean variance relationship
# is given by the ${\rm Var}(\mu) = \mu(1-\mu)$.  Unlike other GLM's
# that we will see later, the variance is uniquely determined by the
# mean, so the scale parameter $\phi$ is fixed at 1.

# As an initial illustration, we will work with the NHANES variable
# [SMQ020](https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/SMQ_I.htm#SMQ020),
# which asks whether a person has smoked at least 100 cigarettes in
# their lifetime (if this is the case, we say that the person has a
# "smoking history").  Below we create a version of this variable in
# which smoking and non-smoking are coded as 1 and 0, respectively, and
# rare responses like *don't know* and *refused to answer* are coded as
# missing values.

da["smq"] = da.SMQ020.replace({2: 0, 7: np.nan, 9: np.nan})

### Odds and log odds

# Logistic regression provides a model for the *odds* of an event
# happening.  Recall that if an event has probability `p`, then the odds
# for this event is `p/(1-p)`.  The odds is a mathematical
# transformation of the probability onto a different scale.  For
# example, if the probability is 1/2, then the odds is 1.

# To begin, we look at the odds of alcohol use for women and men separately.

# Create a labeled version of the gender variable
```

[4]:
```
da["RIAGENDRx"] = da.RIAGENDR.replace({1: "Male", 2: "Female"})

c = pd.crosstab(da.RIAGENDRx, da.smq).apply(lambda x: x/x.sum(), axis=1)
c["odds"] = c.loc[:, 1] / c.loc[:, 0]
#-

# We see that the probability that a woman has ever smoked is
# substantially lower than the probability that a man has ever smoked
# (30% versus 51%).  This is reflected in the odds for a woman smoking
# being much less than 1 (around 0.47), while the odds for a man smoking
# is around 1.14.

# It is common to work with *odds ratios* when comparing two groups.
# This is simply the odds for one group divided by the odds for the
# other group.  The odds ratio for smoking, comparing males to females,
# is around 2.4.  In other words, a man has around 2.4 times greater
# odds of smoking than a woman (in the population represented by these
```

```
# data).

c.odds.Male / c.odds.Female

# It is conventional to work with odds on the logarithmic scale.  To
# understand the motivation for doing this, first note that the neutral
# point for a probability is 0.5, which is equivalent to an odds of 1
# and a log odds of 0.  Populations where men smoke more than women will
# have odds between 1 and infinity, with the exact value depending on
# the magnitude of the relationship between the male and female smoking
# rates.  Populations where women smoke more than men would have odds
# falling between 0 and 1.

# We see that the scale of the odds statistic is not symmetric.  It is
# usually arbitrary in which order we compare two groups -- we could
# compare men to women, or compare women to men.  An odds of 2 (men have
# twice the odds of smoking as women) is equivalent in strength to an
# odds of 1/2 (women have twice the odds of smoking as men).  Taking the
# log of the odds centers the scale at zero, and symmetrizes the
# interpretation of the scale.

# To interpret the log odds when comparing the odds for two groups, it
# is important to remember the following facts:

# * A probability of 1/2, an odds of 1, and a log odds of 0 are all
# equivalent.

# * A positive log odds indicates that the first group being compared
# has greater odds (and greater probability) than the second group.

# * A negative log odds indicates that the second group being compared
# has greater odds (and greater probability) than the first group.

# * The scale of the log odds statistic is symmetric in the sense that a
# log odds of, say, 2, is equivalent in strength to a log odds of -2
# (but with the groups swapped in terms of which has the greater
# probability).

# If you know that the log odds when comparing two groups is a given
# value, say 2, and you want to report the odds, you simply exponentiate
# the log odds to get the odds, e.g. `exp(2)` is around 7.4. Note
# however that you cannot recover the individual probabilities (or their
# ratio) from an odds ratio.`

# Below we show the log odds for smoking history status of females and
# males in the NHANES data.  The fact that the log odds for females is
# negative reflects that fact that substantially less than 50% of
```

```
# females have a history of smoking.  The log odds for males is closer to
# 0, consistent with around half of males having a history of smoking.

# Add a column containing the log odds
c["logodds"] = np.log(c.odds)

### A basic logistic regression model

# Now that we have a clear understanding of log odds statistics, we will
# fit a logistic regression.  The dependent variable (outcome) of this
# initial model is smoking status, and the only covariate is gender.
# Thus, we are looking at gender as a predictor of smoking status.  We
# fit the model using the `GLM` function, which fits a Generalized
# Linear Model.  Logistic regression is one type of GLM.  The `family`
# parameter specifies which particular GLM we are fitting.  Here we use
# the `Binomial` family to perform a logistic regression.

model = sm.GLM.from_formula("smq ~ RIAGENDRx", family=sm.families.Binomial(),␣
 ↪data=da)
result = model.fit()
result.summary()

# To see the connection between logistic regression and the log odds
# statistic, note that the logistic regression coefficient for male
# gender is exactly equal to the difference between the log odds
# statistics for males and females:

c.logodds.Male - c.logodds.Female

# This relationship will always hold when conducting a logistic
# regression with a single binary covariate.

# In general, a logistic regression model will have multiple covariates
# that may not be binary, but there is still an important connection
# between logistic regression and odds ratios.  In this more general
# setting, we will use a more general type of odds ratio, which we will
# explore further next.

### Adding additional covariates

# As with linear regression, we can include multiple covariates in a
# logistic regression.  Below we fit a logistic regression for smoking
# status using age (RIDAGEYR) and gender as covariates.

model = sm.GLM.from_formula("smq ~ RIDAGEYR + RIAGENDRx", family=sm.families.
 ↪Binomial(), data=da)
result = model.fit()
```

```
result.summary()

# Adding age to the model leads to a very small shift in the gender
# parameter (it changed from 0.885 to 0.892).  In general, regression
# coefficients can change a lot when adding or removing other variables
# from a model.  But in this case the change is quite minimal.  This
# fitted model suggests that older people are more likely to have a
# history of smoking than younger people.  The log odds for smoking
# increases by 0.017 for each year of age.  This effect is additive, so
# that comparing two people whose ages differ by 20 years, the log odds
# of the older person smoking will be around 0.34 units greater than the
# log odds for the younger person smoking, and the odds for the older
# person smoking will be around `exp(0.34) = 1.4` times greater than the
# odds for the younger person smoking.

# The greater prevalence of smoking history among older people could be
# partly due to the definition of smoking status that we are using here
# -- an older person has had more time to smoke 99 cigarettes than a
# younger person.  However most people who smoke begin when they are
# young, and the smoking rate in the US has been slowly declining for
# several decades.  Thus, it is likely that the increased smoking levels
# in older people are driven primarily by real shifts in behavior.

# As with linear regression, the roles of age and gender in the logistic
# regression model can be seen as being additive, but here the
# additivity is on the scale of log odds, not odds or probabilities.  If
# we compare a 30 year old female to a 50 year old male, the log odds
# for the male being a smoker are `0.89 + 0.34 = 1.23` units greater
# than the log odds for the female being a smoker.  The value of 0.89 in
# this expression is the change attributable to gender, and the value of
# 0.34 is the change attributable to age.  Again, we can exponentiate to
# convert these effects from the log odds scale to the odds scale.
# Since `exp(0.89 + 0.34) = exp(0.89)*exp(0.34) = 2.44*1.41` we can
# state that male gender is associated with a 2.44 fold increase in the
# odds of smoking, and 20 years of age is associated with a 2.44 fold
# increase in the odds for smoking.  These two effects are multiplied
# when discussing the odds, so a 50 year old man has `exp(1.23) = 3.42`
# fold greater odds of smoking than a 30 year old woman.

# In this logistic regression model with two covariates, the
# coefficients for age and gender both have interpretations in terms of
# *conditional log odds*.  This generalizes the interpretation of a
# logistic regression coefficient in terms of marginal log odds that we
# discussed above.  When there are two or more covariates in a logistic
# regression model, we always need to think in terms of conditional, not
# marginal log odds.
```

```
# Specifically, the coefficient of around 0.89 for male gender impacts
# the conditional log odds in the sense that when comparing a male to a
# female at a fixed age, the male will have 0.89 units greater log odds
# for smoking than the female.  This relationship holds within any age
# (i.e. it holds among all people of age 30, and among all people of age
# 70).  In this sense, it is a *conditional* coefficient because it is
# only interpretable when holding the other variables in the model
# fixed.  Similarly, the coefficient of around 0.02 for age holds within
# a gender.  Comparing two females whose ages differ by one year, the
# older female has 0.02 units greater log odds for smoking than the
# younger female.  This same contrast holds for males.

# ### A logistic regression model with three predictors

# Next we fit a logistic regression model, again for smoking, including
# educational attainment as a predictor.  The educational attainment in
# NHANES is called
# [DMDEDUC2](https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.htm#DMDEDUC2),
# and we will recode it so that the meaning of the levels becomes more
# clear.  We will call the recoded variable `DMDEDUC2x`.

da["DMDEDUC2x"] = da.DMDEDUC2.replace({1: "lt9", 2: "x9_11", 3: "HS", 4:␣
 ↪"SomeCollege",
                                       5: "College", 7: np.nan, 9: np.nan})
model = sm.GLM.from_formula("smq ~ RIDAGEYR + RIAGENDRx + DMDEDUC2x", family=sm.
 ↪families.Binomial(), data=da)
result = model.fit()
result.summary()

# We see that the "Female" level of the gender variable, and the
# "College" level of the educational attainment variable are the
# reference levels, as they are not shown in the output above.  We have
# discussed the gender and age variables above, but the educational
# attainment variable is new for us.  All non-reference coefficients for
# the educational attainment are positive, while the `College`
# coefficient, as the reference coefficient, is exactly zero.  Thus, we
# see that people with a college degree have the lowest rate of smoking,
# followed by people with less than 9 years of schooling, then (after a
# large gap) people with some college, then people with a high school
# degree (and no college), and finally (with the greatest rate of
# smoking), people with 9-11 years of schooling.  The overall story here
# is that smoking rates are much lower for people who graduated from
# college or did not start high school, presumably for very different
# reasons.  On the other hand, people with some high school, people who
# completed high school, and people who began but did not complete
# college have much higher rates of smoking.  The odds ratio between the
# former and the latter group depends on the specific subgroups being
```

```
# compared, but can be almost `3 = exp(1.09)`.

# As noted above when we were discussing linear regression, it is
# important to remember that a coefficient in a logistic regression are
# "conditional" on the other variables being held fixed.  For example,
# the log odds ratio of 1.09 between people with 9-11 years of schooling
# and people who completed college applies only when comparing people
# with the same age and gender.

# ### Visualization of the fitted models

# Visualization of fitted logistic regression models is more challenging
# than visualization of fitted linear models, but is still worth
# pursuing.  We can begin by plotting the fitted proportion of the
# population that smokes, for various subpopulations defined by the
# regression model.  We will focus here on how the smoking rate varies
# with age, so we restrict the population to female college graduates.

# The following plot shows the fitted log odds (or logit) probability
# for the smoking outcome as a function of age.  The grey band is a
# simultaneous 95% simultaneous confidence band, as discussed above in
# the case of a linear model.

from statsmodels.sandbox.predict_functional import predict_functional

values = {"RIAGENDRx": "Female", "RIAGENDR": 1, "BMXBMI": 25,
          "DMDEDUC2": 1, "RIDRETH1": 1, "SMQ020": 1,
          "DMDEDUC2x": "College", "BPXSY1": 120}

pr, cb, fv = predict_functional(result, "RIDAGEYR",
                values=values, ci_method="simultaneous")

ax = sns.lineplot(fv, pr, lw=4)
ax.fill_between(fv, cb[:, 0], cb[:, 1], color='grey', alpha=0.4)
ax.set_xlabel("Age")
ax.set_ylabel("Smoking")

# We can display the same plot in terms of probabilities instead of in
# terms of log odds.  The probability can be obtained from the log odds
# using the relationship `p = 1 / (1 + exp(-o))` where `o` is the log
# odds.  Note that while the age and log odds are linearly related, age
# has a curved relationship with probability.  This is necessary since
# probabilities must remain between 0 and 1, a linear relationship would
# eventually exit the domain.

pr1 = 1 / (1 + np.exp(-pr))
cb1 = 1 / (1 + np.exp(-cb))
```

```
ax = sns.lineplot(fv, pr1, lw=4)
ax.fill_between(fv, cb1[:, 0], cb1[:, 1], color='grey', alpha=0.4)
ax.set_xlabel("Age", size=15)
ax.set_ylabel("Smoking", size=15)

# Next we turn to diagnostic plots that are intended to reveal certain
# aspects of the data that may not be correctly captured by the model.
# The three plots below are intended to reveal any curvature in the mean
# relationship between the outcome and one of the covariates. We used
# the partial regression plotting technique above for this same purpose
# when working with linear models.

# In the case of logistic regression, the three techniques demonstrated
# below can identify major discrepancies between the fitted model and
# the population, but evidence for small discrepancies is not reliable
# unless the sample size is very large.  The CERES technique has the
# strongest theoretical support.  The red curve in the plots below shows
# an estimate of the true role `RIDAGEYR` in the model, not assuming it
# to be linear (but leaving everything else linear).  Thus, we are
# working with the linear predictor

# $$
# \beta_0 + \beta_1{\rm Male} + \cdots + \beta_5{\rm Educ9_11} + h({\rm␣
 ↪RIDAGEYR}),
# $$

# and the CERES analysis estimates the shape of the function $h$.

# Taken at face value, the plots below suggest that smoking rates may
# rise slightly faster for people between the ages of 20 and 35, and
# again for people between the ages of 50 and 60, with a period of
# minimal increase between these age intervals.  This would contradict
# the perfectly linear model for age (on the log odds scale) that we
# have specified in our model.  These plotting techniques can be useful
# at identifying possible opportunities for future analysis with
# additional data, but do not identify features that can be claimed with
# high confidence using the present data.


fig = result.plot_partial_residuals("RIDAGEYR")
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)

from statsmodels.graphics.regressionplots import add_lowess
_ = add_lowess(ax)

fig = result.plot_added_variable("RIDAGEYR")
```

```
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)
_ = add_lowess(ax)

fig = result.plot_ceres_residuals("RIDAGEYR")
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)
_ = add_lowess(ax)

## Poisson regression

# Poisson regression is a type of GLM based on the Poisson distribution.
# A Poisson random variable is supported on the non-negative integers,
# and its distribution is fully determined by its mean $\mu$.  The
# Poisson distribution can be used to describe many phenomena, but is
# particularly useful when we are observing the number of times that
# some event happened, when there were a large number of opportunities
# for the event to happen, each with a small probability.  For example,
# suppose we have a large sample of drivers and divide them into
# disjoint groups of 10,000 drivers at random.  If we were then to count
# the total number of traffic accidents per group, these counts may
# follow a Poisson distribution.

# In a regression setting, we want the distribution of the dependent
# variable $y$ to be related to the covariate vector $x$.  Thus, for
# Poisson regression we want the mean of $y$ to be expressed as
# $\mu(x)$.  In Poisson regression, this is usually accomplished using
# the log link function, so that

# $$
# \log(\mu) = \log E[y|x_1, \ldots, x_p] = b_0 + b_1x_1 + \cdots + b_px_p.
# $$

# The Poisson distribution has the property that the variance is equal
# to the mean.  Thus the mean/variance relationship in a Poisson
# regression is described by ${\rm Var}(\mu) = \mu$.  In practice, data
# that seem to be good candidates for modeling with a Poisson
# distribution often turn out to not follow a Poisson distribution very
# well because the mean/variance relationship in the data is very
# different from ${\rm Var}(\mu) = \mu$.  We will see this below and
# discuss some additional GLM approaches that are closely related to the
# Poisson GLM, but that specify the mean/variance relationship in a
# different way.

# ### Poisson models for cluster totals

# As an initial illustration, we will construct a summarized version of
```

```
# the NHANES data that is a good candidate for Poisson regression.
# NHANES aims to provide a sample of subjects that is representative of
# the US adult population.  It does this using a technique called
# *stratified cluster sampling*, which is widely utilized in surveys.
# For our purposes, we can think of the strata and clusters as being
# distinct geographical locations where NHANES recruited subjects (the
# reality is more complex than this, as the actual location information
# cannot be disclosed for confidentiality reasons).  For illustration,
# we will treat each combination of the stratum variable `SDMVSTRA` and
# the cluster variable `SDMVPSU` as defining a block of subjects.
# Within each block, we will calculate summary statistics and analyze
# them using Poisson regression.

url = "https://raw.githubusercontent.com/kshedden/statswpy/master/NHANES/merged/
 ↪nhanes_2015_2016.csv"
da = pd.read_csv(url)

dx = da.groupby(["SDMVSTRA", "SDMVPSU"]).agg({"BPXSY1": [np.size, lambda x: np.
 ↪sum(x > 140)],
                                              "RIDAGEYR": np.mean, "INDFMPIR":␣
 ↪np.mean,
                                              "RIAGENDR": lambda x: np.mean(x␣
 ↪== 1),
                                              "DMDMARTL": lambda x: np.mean(x␣
 ↪== 1)})

# Give the variables better names
dx.columns = [':'.join(c).strip() for c in dx.columns.values]
dx = dx.rename(columns={"RIDAGEYR:mean": "Age", "BPXSY1:size": "N",
                        "BPXSY1:<lambda_0>": "HighBP", "INDFMPIR:mean":␣
 ↪"INDFMPIR",
                        "RIAGENDR:<lambda>": "Male", "DMDMARTL:<lambda>":␣
 ↪"Married"})
print(dx.head())

# As a specific example, below we take the number of people in each
# block who have high blood pressure (systolic blood pressure greater
# than 140) as the dependent variable, and use block-level mean age,
# poverty level (mean INDFMPIR), gender (proportion of males), and
# marriage rates as predictors.

# Note that in practice, we would not normally take an individual-level
# dataset like NHANES and summarize it like this before doing regression
# analysis.  Instead, we would generally prefer to analyze the
# individual-level data directly.  However there are many important data
# sets that provide only block-level summaries, so this is an
```

```
# illustration of a technique that is often useful in practice.

# As discussed above, Poisson regression is especially useful for an
# outcome variable that is a count.  Here, all the people in one block
# are "at risk" for having high blood pressure, and only a subset of
# them actually have high blood pressure.  Some blocks have more people
# than others, and all else being equal, those blocks are expected to
# have more people with high blood pressure.  We account for this by
# including the log of the sample size as a covariate (we will explain
# why this variable is log transformed below).  In addition, some blocks
# will have older people (on average), or will have greater prevalence
# of other risk factors for high blood pressure.  We can use Poisson
# regression to assess which block-level characteristics are associated
# with higher or lower prevalence of high blood pressure.

dx["logN"] = np.log(dx.N)

model = sm.GLM.from_formula("HighBP ~ logN + Age + INDFMPIR + Male + Married",␣
 ↪family=sm.families.Poisson(), data=dx)
result = model.fit(scale="X2")
print(result.summary())
print(result.scale)

# The results above indicate that larger blocks have more people with
# high blood pressure.  Also, blocks with an older population or with
# lower marriage rates also have higher blood pressure.  There is no
# evidence here that wealth (INDFMPIR) or gender (Male) relate to
# blood pressure.  The lack of a role for wealth potentially relates
# to its correlation with marriage rate (take marriage rate out of the
# model and the wealth effect becomes statistically significant).  It
# is well-established that men have higher blood pressure than women
# on average (controlling for age).  The lack of an apparent role for
# gender here may be because the blocks we are using have little
# variation in gender.

# Regarding the log-transformation of the block size value, recall
# that the mean for the Poisson model can be written

# $$
# \mu_i = \exp(\beta_0 + \beta_1\log(N_i) + \cdots) =
# \exp(\beta_0)\cdot N_i^{\beta_1} \cdots
# $$

# Thus we see that when $\beta_1 \approx 1$, the number of people with
# high blood pressure scales directly with the block size, for fixed
# values of the other covariates in the model.  Here however, the
# coefficient for $\log(N_i)$ is somewhat bigger than $1$, suggesting
```

```
# that the larger blocks have even more cases of high blood pressure
# than expected under 1:1 scaling.

# The mean/variance relationship for a Poisson model is ${\rm
# Var}(\mu) = \mu$.  There is a larger class of GLM's called
# *quasi-Poisson GLMs* which have the mean/variance relationship ${\rm
# Var}(\mu) = \phi\mu$, for an unknown scale parameter $\phi$ that can
# be estimated from the data.  It turns out that the parameter
# estimates of the $\beta_j$ are the same under a quasi-Poisson and a
# Poisson model.  However the standard errors and other quantities
# differ.  By using `scale='X2'` in the `fit` method call, we are
# stipulating that a quasi-Poisson model be fit.  We can then inspect
# the estimated value of the scale parameter to see that it is very
# close to $1$.  Thus, in this case, the Poisson model seems
# well-justified.

# Another family of GLMs that can be considered here is the *negative
# binomial* family.  These GLMs typically also use the log as the link
# function, but have a more general mean/variance relationship of the
# form ${\rm Var}(\mu) = \mu + \alpha\mu^2$.  The additional parameter
# $\alpha$ here is a shape parameter that controls how the mean and
# variance are related.  If $\alpha = 0$, we have a Poisson-like
# mean/variance relationship.  Larger values of $\alpha$ capture one
# form of overdispersion.

# The shape parameter in a negative binomial GLM can be estimated
# using maximum likelihood.  Here we consider a sequence of possible
# shape parameter values and determine their log-likelihood values.
# Since the highest likelihood is achieved for the smallest value of
# the shape parameter, this is further evidence that the data we are
# working with here is well-described by the Poisson GLM.

# Fit some negative binomial models
print("Negative binomial log-likelihoods:")
for shape in 0.1, 0.5, 1, 2:
    model = sm.GLM.from_formula("HighBP ~ logN + Age + INDFMPIR + Male +␣
 ↪Married",
                  family=sm.families.NegativeBinomial(alpha=shape), data=dx)
    result = model.fit()
    print(shape, result.llf)

### Poisson regression for household size

# As a second illustration of Poisson GLM, we will use the household
# size variable `DMDHHSIZ` from NHANES.  The household size values
# range from 1 to 7 in NHANES (they are truncated at 7).  It is an
# interesting and important topic in demographics to understand what
```

```
# characteristics are associated with either lower or higher household
# size.

# First we create a dataset that includes the household size variable
# (`DMDHHSIZ`), along with some other variables that may be related to
# it.

url = "https://raw.githubusercontent.com/kshedden/statswpy/master/NHANES/merged/
 ↪nhanes_2015_2016.csv"
da = pd.read_csv(url)

da = da[["DMDEDUC2", "DMDHHSIZ", "DMDCITZN", "RIDRETH1", "RIAGENDR",␣
 ↪"INDFMPIR", "DMDMARTL", "RIDAGEYR"]]

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

# Check to see what we lose by dropping rows with missing values
print(pd.isnull(da).sum(0))
print(da.shape)
print(da.dropna().shape)
da = da.dropna()

# Next we fit a model predicting the size of a person's household from
# the person's age and gender, including an interaction between age
# and gender, and using splines to allow for curvature in the role of
# age.

model = sm.GLM.from_formula("DMDHHSIZ ~ bs(RIDAGEYR, 3)*RIAGENDRx", family=sm.
 ↪families.Poisson(), data=da)
result = model.fit()
result.summary()

# There is a strong association between age and household size, but
# there isn't much evidence that gender plays a role here.
# Nevertheless we will make a plot of the fitted means by gender just
# to see how they look:

values = {"DMDHHSIZ": 0, "DMDCITZN": 0, "RIAGENDR": 0, "DMDEDUC2": 0,␣
 ↪"INDFMPIR": 0,
          "RIDRETH1": 0, "DMDMARTL": 0}

for gender in "Female", "Male":
    values["RIAGENDRx"] = gender
    pr, cb, fv = predict_functional(result, "RIDAGEYR",
                    values=values, ci_method="simultaneous")
    ax = sns.lineplot(fv, pr, lw=4, label=gender)
```

```
ax.set_xlabel("Age")
ax.set_ylabel("Log expected household size")

# Marital status likely also plays a role, and it turns out that a role
# for gender also emerges now that we consider marital status:

# 1 if the person is married, 0 otherwise
da["Married"] = (da.DMDMARTL == 1).astype(np.int)

model = sm.GLM.from_formula("DMDHHSIZ ~ bs(RIDAGEYR, 3)*Married + bs(RIDAGEYR,␣
 ↪3)*RIAGENDRx + Married*RIAGENDRx", family=sm.families.Poisson(), data=da)
result = model.fit(scale="X2")
result.summary()

# The model fit above includes all two-way interactions among age,
# gender, and marital status.  The model with three-way interactions
# did not converge.  There are ways to work around such convergence
# issues, but here we will focus on the simpler model without the
# three-way interaction.

# To understand this model, we can plot the mean curves for married
# and unmarried women and for married and unmarried men:

values["Married"] = 0

for married in 0, 1:
    for gender in "Female", "Male":
        values["Married"] = married
        values["RIAGENDRx"] = gender
        pr, cb, fv = predict_functional(result, "RIDAGEYR",
                        values=values, ci_method="simultaneous")
        ax = sns.lineplot(fv, pr, lw=4, label=gender + str(married))

ax.set_xlabel("Age")
ax.set_ylabel("Log expected household size")

# There is a lot going on here, but here are some possible
# interpretations:

# * For married people, household size increases up to around age
# around age 40.  Presumably this is due to children living at home
# and subsequently leaving home.

# * Household size for married women and married men of a fixed age
# are similar, but unmarried women live in larger households than
# unmarried men, likely because they are living with children or
# parents.
```

```
# Just because a variable has integer values does not mean that it
# follows a Poisson distribution.  As noted above, one key property of
# the Poisson distribution is that its conditional variance and
# conditional mean are equal.  With enough data, we can assess this
# graphically as shown below.  We can also incorporate a scale
# parameter which allows the variance to be proportional to, but not
# necessarily equal to the mean.

da["fit"] = result.fittedvalues
da["fitd"] = pd.qcut(da.fit, 10)
dx = da.groupby("fitd").agg({"fit": np.mean, "DMDHHSIZ": np.var})

sns.scatterplot(x="fit", y="DMDHHSIZ", data=dx)

print(result.scale)

# The plot above shows little evidence of an increasing trend between
# the mean and the variance.  This leads us to wonder whether the
# Poisson model was a good choice in the first place.  We can compare
# the AIC (lower is better) for a linear model, the Poisson model, and
# a series of negative binomial models to assess this.

# A linear model fit with least squares
model0 = sm.OLS.from_formula("DMDHHSIZ ~ bs(RIDAGEYR, 3)*Married + bs(RIDAGEYR,␣
 ↪3)*RIAGENDRx + Married*RIAGENDRx", data=da)
result0 = model0.fit()
print("OLS AIC:", result0.aic)

# Our Poisson model
print("Poisson AIC:", result.aic)

# Fit some negative binomial models
print("Negative binomial AICs:")
for shape in 0.1, 0.5, 1, 2:
    model1 = sm.GLM.from_formula("DMDHHSIZ ~ bs(RIDAGEYR, 3)*Married +␣
 ↪bs(RIDAGEYR, 3)*RIAGENDRx + Married*RIAGENDRx", family=sm.families.
 ↪NegativeBinomial(alpha=shape), data=da)
    result1 = model1.fit()
    print(result1.aic)

# Based on the AIC's shown above, the linear model appears to fit the
# data best among these alternatives.
```

```
                      N  HighBP        Age  INDFMPIR      Male   Married
SDMVSTRA SDMVPSU
119      1        170.0    18.0  47.747059  2.324718  0.423529  0.388235
```

```
         2        127.0    23.0  53.464567  2.446147  0.346457  0.346457
120      1        199.0    22.0  43.381910  1.952376  0.502513  0.608040
         2        145.0    10.0  44.482759  2.958462  0.503448  0.572414
121      1        202.0    31.0  45.608911  1.996270  0.514851  0.405941
```

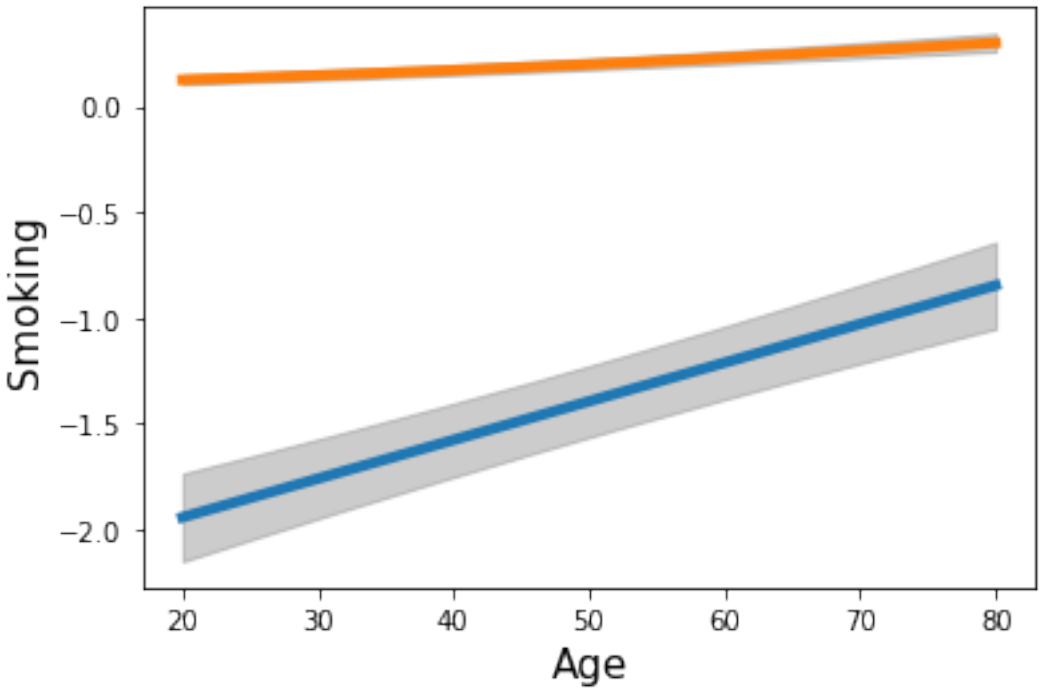
Generalized Linear Model Regression Results

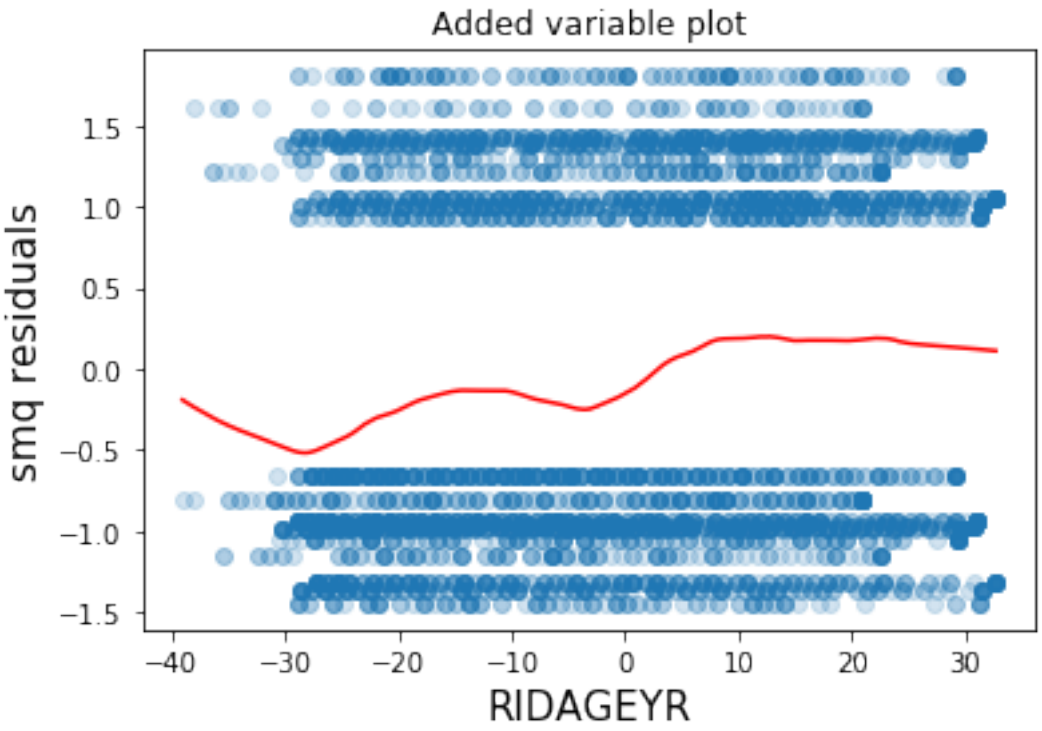
Dep. Variable:	HighBP	No. Observations:	30
Model:	GLM	Df Residuals:	24
Model Family:	Poisson	Df Model:	5
Link Function:	log	Scale:	1.0181
Method:	IRLS	Log-Likelihood:	-88.381
Date:	Sun, 09 Feb 2020	Deviance:	24.979
Time:	20:42:09	Pearson chi2:	24.4
No. Iterations:	6		
Covariance Type:	nonrobust		

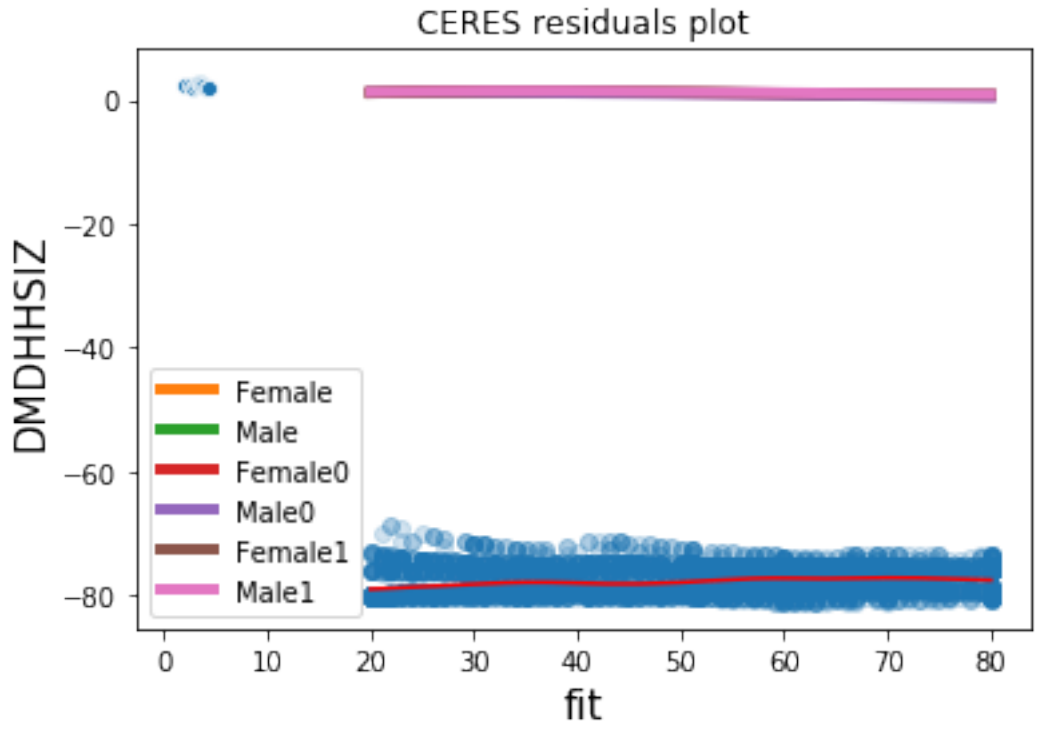
	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-6.6637	1.724	-3.866	0.000	-10.042	-3.286
logN	1.7306	0.324	5.338	0.000	1.095	2.366
Age	0.0432	0.011	3.786	0.000	0.021	0.066
INDFMPIR	-0.1311	0.118	-1.108	0.268	-0.363	0.101
Male	-0.7886	1.054	-0.748	0.455	-2.855	1.278
Married	-0.9025	0.437	-2.067	0.039	-1.758	-0.047

```
1.018132583701166
Negative binomial log-likelihoods:
0.1 -101.66238958222229
0.5 -120.24376082407467
1 -130.78059783037492
2 -142.79485300446368
DMDEDUC2     261
DMDHHSIZ       0
DMDCITZN       1
RIDRETH1       0
RIAGENDR       0
INDFMPIR     601
DMDMARTL     261
RIDAGEYR       0
RIAGENDRx      0
dtype: int64
(5735, 9)
(4908, 9)
0.7635339248997871
OLS AIC: 18101.122841508586
Poisson AIC: 23420.13043657308
Negative binomial AICs:
18406.539197253944
```

20504.083703267865
22557.149821402698
25517.52317278125




Added variable plot
smq residuals
RIDAGEYR
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−40
−30
−20
−10
0
10
20
30


CERES residuals plot
DMDHHSIZ
fit
Female
Male
Female0
Male0
Female1
Male1
0
−20
−40
−60
−80
0
10
20
30
40
50
60
70
80