statistics_with_python_specialization

January 6, 2021

[1]:
```
from IPython.display import Image
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
%config InlineBackend.figure_format = 'retina'
```

		Difference	Order	Similar intervals	Meaningful zero point
Categorical	nominalordinal	++	-+	–	–
Quantitive	intervalratio	++	++	++	-+

Levels of Measurement

- Quantitative variables can also be distinguished in **discrete** and **continuous** variables.
- A variable is discrete if it's possible categories form a set of separate numbers.
  - For instance, the number of goals scored by a football player. A player can score, for instance, one goal or two goals, but not 1.21 goals.
- A variable is continuous if the possible values of the variable form an interval. An example is again, the height of a player.
  - Someone can be 170 centimeters, 171 centimeters tall. But also for instance, 170.2461 centimeters tall. We don't have a set of separate numbers, but an infinite region of values.

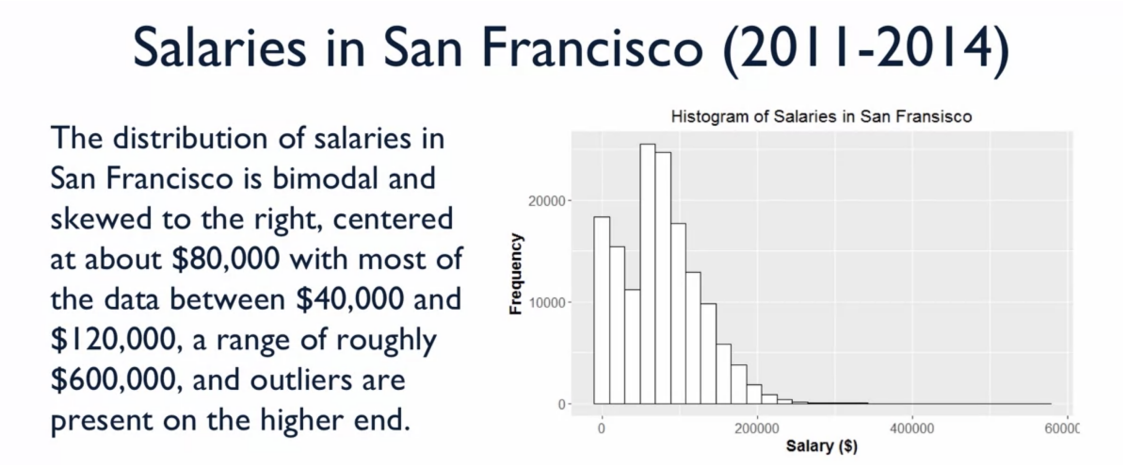
Data Types in Python Numerical or Quantitative (taking the mean makes sense) - Discrete - Integer (int) - Stored exactly - Continuous - Float (float) - Stored similarly to scientific notation. Allows for decimal places but loses precision.

Categorical or Qualitative - Nominal - Boolean (bool) - String (str) - None (NoneType) - Ordinal - Only defined by how you use the data - Often important when creating visuals - Lists can hold ordinal information because they have indices

Quantitative Data: Histograms 4 Main Aspects - **Shape:** Overall appearance of histogram. Can be symmetric, bell-shaped, left skewed, right skewed, etc. - **Center:** Mean or median. - **Spread:** How far our data spreads. Range, Interquartile Range (IQR), standard deviation, variance. - **Outliers:** Data points that fall far from the bulk of the data.

[3]:
```
Image('images/example_1.png', width=600)
```

[3]:



- unimodal: if the histogram has a peak
- bimodal: if the histogram has two peaks
- As you can see in the graph, it's right skewed distribution. So the mean value greater than the median value in this distrubution.
- We have outliers on the higher end and so that's pulling the mean towards it.
- **The median is what we call a robust estimate of the center, meaning it's not influenced by outliers.**

To summarize, generally if the distribution of data is skewed to the left, the mean is less than the median, which is often less than the mode. If the distribution of data is skewed to the right, the mode is often less than the median, which is less than the mean.

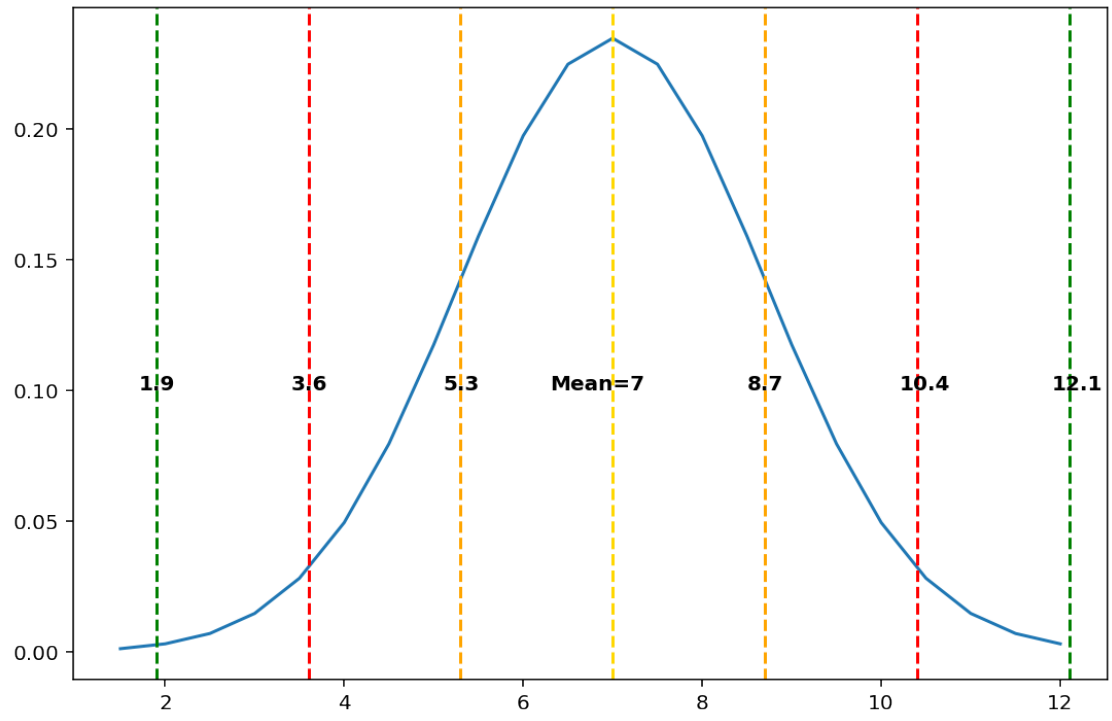
Empirical Rule (68-95-99.7 Rule) The empirical rule, also referred to as the three-sigma rule or 68-95-99.7 rule, is a statistical rule which states that for a normal distribution, almost all observed data will fall within three standard deviations (denoted by) of the mean or average (denoted by μ).

In particular, the empirical rule predicts that 68% of observations falls within the first standard deviation ($\mu \pm$), 95% within the first two standard deviations ($\mu \pm 2$), and 99.7% within the first three standard deviations ($\mu \pm 3$).

[4]:
```
mean = 7
standard_deviation = 1.7
```

[5]:
```
# normal distribution
x = np.arange(1.5, 12.5, 0.5)
y = norm.pdf(x, mean, standard_deviation)
```

[6]:
```
colors = ['orange', 'red', 'green']

plt.figure(figsize=(9, 6))

# bell curve
ax = plt.plot(x, y)

# mean line
plt.axvline(7, color='gold', linestyle='--')
plt.text(6.3, 0.1, 'Mean=7', fontweight='bold')

for i in range(1, 4):
    pos_line = round(mean + i * standard_deviation, 2)
    neg_line = round(mean - i * standard_deviation, 2)

    plt.axvline(pos_line, color=colors[i - 1], linestyle='--')
    plt.axvline(neg_line, color=colors[i - 1], linestyle='--')

    plt.text(pos_line - 0.2, 0.1, f'{pos_line}', fontweight='bold')
    plt.text(neg_line - 0.2, 0.1, f'{neg_line}', fontweight='bold')
```



In the graph above, you see a normal distribution of people's sleep times. In this data, the mean value is 7 and the standard deviation is 1.7. So, we can calculate the points for empirical rule. - For the 68%, - mean + 1 * standart deviation = 8.7 - mean - 1 * standart deviation = 5.3 - For

the 95%, - mean + 2 * standart deviation = 3.6 - mean - 2 * standart deviation = 10.4 - For the 99.7%, - mean + 3 * standart deviation = 1.9 - mean - 3 * standart deviation = 12.1

For instance, I sleep 10 hours a day. Let's calculate the standard score (z-score) for me! Here is the formula,

Z score = (Observation - Mean) / Standard Deviation

[7]:
```
z_score = (10 - mean) / standard_deviation
z_score
```

[7]:
```
1.7647058823529411
```

Hmm! I sleep a little more than average. Let's take a look at a friend who sleeps 6 hours a day!

[8]:
```
z_score = (6 - mean) / standard_deviation
z_score
```

[8]:
```
-0.5882352941176471
```

This value is much less than me! And as you can see from the minus, this is slightly below average.

Pitfall: Simpson's Paradox In this section, we're going to introduce a Pitfall within Statistics: Simpson's Paradox.

What if I told you that the rate of completing Course 2 after you've completed Course 1 of a Specialization is higher for males than for females? What would you think?

Now, what if I told you that more males are enrolled in a full Specialization, while more females are interested in taking each course as a freestanding course? Does that change how you think about the situation?

What might actually be happening is that there are different rates of Course 2 completion for those in the Specialization versus those taking freestanding courses, and that the gender ratio may also be different for those two situations.

Let's take a look at the numbers:

	Specialization	Course	Total
Men	55/70 (79%)	10/20 (50%)	65/90 (72%)
Women	25/30 (83%)	17/30 (57%)	42/60 (70%)
Total	80/100 (80%)	27/50 (54%)	

You may notice that the completion rate of Course 2 is higher for women within each of the methods of taking this course, but overall it appears that the completion rate of Course 2 is higher for men. The way of taking courses hides the true direction of the relationship between gender and completion rate of Course 2.

The method of taking the course can be considered a **confounding variable**. A confounding

variable is an outside influence that changes the relationship between the independent and the dependent variable. It oftentimes works by affecting the causal relationship between the primary independent variable and the dependent variable. This confounding variable confuses the relationship between two other variables; it may act by hiding, obscuring, or enhancing the existing relationship.

For example, suppose that you are interested in examining how activity level affects weight change. Other factors, like diet, age, and gender may also affect weight change and come into play when looking at the relationship between activity level and weight change. If one doesn't control for these factors, the relationship between activity level and weight change can be distorted.

This also serves as a reminder that **Correlation does not imply Causation**. There may be other factors involved with a causal relationship. For this reason, you shouldn't place too much stock in correlation alone.

From pandas docs: * [] column indexing * **.loc** is primarily label based, but may also be used with a boolean array.
* **.iloc** is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.

[1]:
```
from IPython.display import Image
from statsmodels.distributions.empirical_distribution import ECDF
import matplotlib.pyplot as plt
import numpy as np
import random
import seaborn as sns

sns.set()
```

[2]:
```
%config InlineBackend.figure_format = 'retina'
```

Populations vs. Samples First of all, we need to well-defined target population. - Important first step: Define target population of interest in concrete terms! - Who are we measuring? - Males? - African-American males? - Older African-American males? - What does "older" mean? Be specific! - What time frame are we insterested in? - 2018? April 2018? - The first half of April in 2018? - Where is the population located in? - The Midwest? Michigan? - Washtenaw county? - The city of Ann Arbor?

Given a target population, now what? How can we make inferential statements about it? - **Option 1:** Conduct a census! - Eaiser for smaller target populations - Incredibly expensive for larger populations - Requires a careful evaluation of, - how much it will cost to measure all population units - what administrative data sources already available - **Option 2:** Select a scientific probability sample - Construct list of all units in population = sampling frame - Determine probability of selection for every unit on list (known and non-zero) - Select units from list at random, with sampling rates for different subgroups determined by probabilities of selection - Attempt to measure randomly selected units - **Option 3:** Select a non-probability sample - Generally does not involve random selection - Probabilities of selection can't be determined for population units - Examples, - **opt-in web surveys:** In these opt-in web surveys, you're really just trying to take whoever's interested in taking that web survey. You're not selecting people at random from some well-defined

list or a sampling frame. It's whoever wants to volunteer, to participate in that particular web survey. And as a result, we can't determine those probabilities of selection. - **quota sampling:** That is, you try to recruit as many people as you can who fit certain subgroup definitions. For example, older African-American males until you hit some target. Some number of individuals that you wish to measure. And in some of those cases with quota sampling, researchers try to collect as many individuals as they can not according to any probability scheme, but just based on whoever's available. Just as long as they hit their targets or their quotas. This too is another example of non-probability sampling where again, we can't write down probabilities of being selected in that sample. - **snowball sampling:** This is where, as you can imagine a snowball, if you roll it down a hill, it keeps getting bigger and bigger and you're gathering more and more snow as the snowball rolls. In this case, you recruit somebody to participate in a study, and then they might tell a friend, and then that friend might tell a friend, and your sample ultimately gets bigger by collecting individuals from this chain that you can see up here on the screen. So in these cases, again, the friends are recruiting friends and social networks. And we don't really have control over who they recruit, or the probabilities with which they're going to recruit these other individuals. So snowball sampling, it's a convenient tool for recruiting a sample. But as researchers, we don't have control over those probabilities of selection. - **convenience sampling:** This is another way that we could collect non-probability samples. For example, you might go out on the street and just talk to people who are available to collect data and ask questions. If you're teaching in a university or business setting, you might just collect data from the individuals in your courses. Or from your coworkers or whoever's close to you. Again, no probabilities of selection involved.

Probability Sampling

Option 1: Simple Random Sampling (SRS)

- Notice that, this method is often expensive for large populations
- Start with known list of N population units, and randomly select n units from the list
- Every unit has **equal probability of selection** = n / N
- All possible samples of size n are equally likely
- Estimates of means, proportions, and totals based on SRS are **unbiased** (equal to the population values on average!)
- SRS example,
  - Customer service database: N=2500 email requests in 2018
  - Director wants to estimate: mean email response time
  - Exact calculations require manul review of each email thread
  - Ask analytics team: sample, process and analyze n=100 emails
  - **Naive Approach:** process the first 100 emails on list
    - Estimated mean could be **biased** if customer service representatives learn or get better over time at responding more quickly
    - First 100 observations may come from a small group of staff
    - Not fully representative, independent, or identically distributed!
    - No random selection according to specific probabilities!
  - **Better SRS Approach:** number emails 1 to 2500 and randomly select 100 using a random number generator
    - Every email has known probability selection = 100 / 2500
    - Produces random, representative samples of 100 emails (in theory)
    - Estimated mean response time will be an unbiased estimate of the population mean

Option 2: Complex Samples

- Population divided into different **strata**, and part of sample is allocated to each **stratum**; -> ensures sample representation from each stratum, and reduce variance of survey estimates (stratification)
- **Clusters** of population units (e.g., counties) are randomly sampled first (with known probability) within strata, to save cost of data collection (collect data from cases close to each other geographically)
- Units randomly sampled within clusters, according to some probability of selection, and measured
- A unit's probability of selection is determined by:
  - Number of clusters sampled from each stratum
  - Total number of clusters in population each stratum
  - Number of units ultimately sampled from each cluster
  - Total number of units in population in each cluster
- Why probability sampling?
  - Having known, non-zero probability of selection for each unit in population and subsequent random sampling ensures all units will have a chance of being sampled
  - Probability sampling allows us to compute unbiased estimates, and also estimate features of the sampling distribution of estimates that we would see if many of the same types of probability samples were selected
  - Most importantly, probability sampling provides a stastistical basis for making inferences about certain quantities in larger populations

Non-Probability Sampling

- Features of non-probability samples,
  - Probabilities of selection can't be determined for sample units
  - No random selection of individual units
  - Sample divided into groups (strata) or clusters, but clusters not randomly sampled in earlier stage
  - Data collection often very cheap relative to probability sampling
  - Please keep in mind, you can find the types of non-probability sampling in the beginning of the notebook.
- What is the problem?
  - Non-probability sample -> no statistical basis for making inference about larger population from which sample selected
  - Knowing probabilities of selection (in addition to population strata and randomly sampled clusters)
    - can estimate features of sampling distribution (if were to take many random samples using same design)
  - Sampled units not selected at random -> strong risk of sampling bias (e.g., people actually insterested in visiting particular web site)
  - Sampled units not generally representative of larger target population of interest
  - **Big Data** (e.g., information from millions of tweets) often from non-probability samples
- So what can we do?
  - Two possible approaches,
    - Pseudo-Randomization

    - Calibration
  - **Option 1:** Pseudo-Randomization Approach
    - Combine non-probability sample with a probability sample (that collected similar measurements)
    - Estimate probability of being included in non-probability sample as a function of auxiliary information available in both samples
    - Treat estimated probabilities of selection as "known" for non-probability sample, use probability sampling methods for analysis
  - **Option 2:** Calibration Approach
    - Compute weights for responding units in non-probability sample that allow weighted sampled to mirror a known population
    - Non-probability sample: 70% female, 30% male
    - Population: 50% female, 50% male
      - Down-weight females and up-weight males
    - Limitation: if weighting factor not related to variable(s) of interest will not reduce possible sampling bias

Twitter Example: Non-Probability Sampling - API to extract into from several hundred thousand tweets and indicator of support for President Trump computed - Probability of a tweet being selected cannot be determined - Twitter users not a random sample for larger population - Lots of data but, - high potential for sampling bias - lack of representation: may only capture people with strong opinions!

What is the sampling distribution?

- Recall: Distribution of values on a variable of interest
  - Example: Normal distribution (bell curve)
- Assume values on variable of interest would follow certain distribution if we could measure entire population
- Recall: When we select porbability samples to make inferential statements about larger populations
- Sampling distribution: distribution of survey estimates we would see if we selected many random samples using same sample design, and computed an estimate from each
- **Key properties** of sampling distribution,
  - Hypothetical! What would happen if we had luxury of drawing thousands of probability samples and measuring each of them?
  - Generally very different in appearance from distribution of values on a single variable of interest
  - With large enough probability sample size, sampling distribution of estimates will look like a normal distribution, regardless of what estimates are being computed! **Central Limit Theorem**

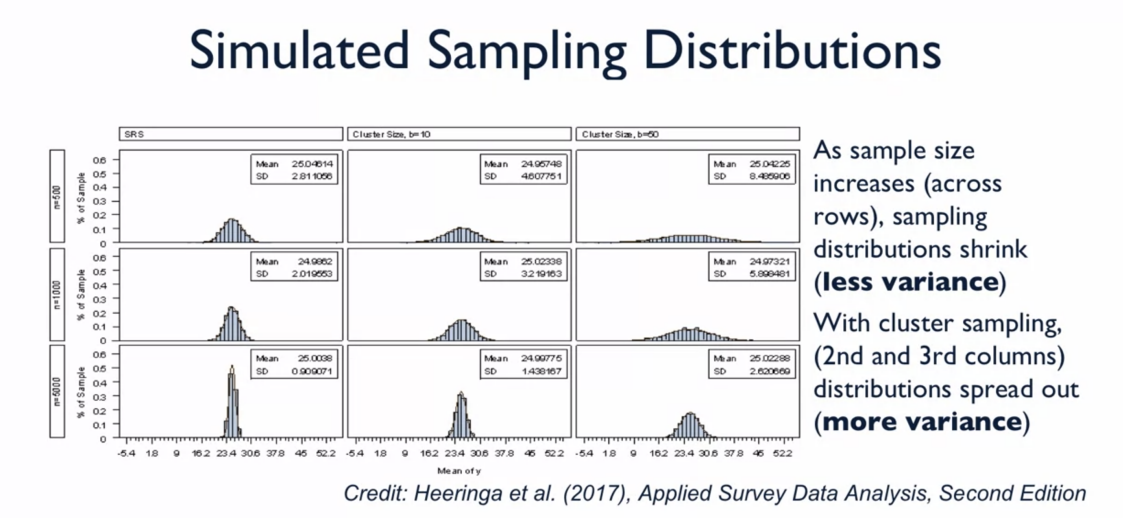
What is sampling variance?

- Variability in the estimates described by the sampling distribution
- Because we select a sample (do not measure everyone in a a population) a survey estimate based on a single sample will not be exactly equal to population quantity of interest (cases are randomly selected!)
- This called as **Sampling Error**

- Across hypothetical repeated samples, these sampling errors will randomly vary (some negative, some positive…)
- Variability of these sampling errors describes the variance of the sampling distribution
- If every sample estimate was equal to population quantity of interest (e.g., in the case of a Census), there would be no sampling error, and no sampling variance!
- With a larger probability sample size, sampling more from a given population -> in theory there will be less sampling error, and sampling errors will be less variable
- Larger samples -> less sampling variance! More precise estimates, more confidence in inferential statements (but more costly!)
- Spred of sampling distribution becomes smaller as sample size become larger

[3]:
```
Image('images/example_2.png', width=800)
```

[3]:



If you compare a simple random sample of size 500 in the upper left to a cluster sample where there's 50 units selected per 50 cluster of size 5,000 in the lower right, look at those sampling distributions, they're almost the same. So a simple random sample of size 500, the same distribution looks like a cluster sample of size 5,000, where there's 50 units selected per cluster. So, another key trade-off between the probability sampling design and how variable our estimates are.

Why is sampling variance important?

- In practice, we only have the resources to select one sample!
- Important sampling theory allows us to estimate features of sampling distribution (including variance!) based on one sample
- "Magic" of probability sampling: can select one probability sample and features of that design tell us what we need to know about the expected sampling distribution
- Because we can estimate variance of sampling distribution based only one sample, we can make inferential statements about where most estimates based on a particular sample design will fall
- Can make statements about likely values of population parameters that account for variability in sampling errors that arises from random probability sampling

Making Population Inference Based on Only One Sample

- General approaches to making population inferences based on estimated features of sampling distributions
  - Confidence interval estimate for parameters of estimate
  - Hypothesis testing about paramters of interest
  - Example of parameters of interest: a mean, a proportion, a regression coefficient, an odds ratio, and many more!
- Key Assumption: Normality
  - These approaches assume that sampling distributions for the estimate are (approximately) normal, which is often met if sample sizes are "large"
- Step 1: Compute the point estimate
  - Compute an unbiased point estimate of the parameter of interest
  - Unbiased point estimate: average of all possible values for point estimate is equal to true parameter value
  - The sampling distribution is centered at the truth!
  - Key idea: want estimate to be unbiased with respect to sample design! if cases had unequal probabilities of selection, those weights need to be used when computing the point estimate!
- Step 2: Estimate the sampling variance of the point estimate
  - Compute an unbiased estimate of the variance of the sampling distribution for the particular point estimate
  - Unbiased variance estimate: correctly describes variance of the sampling distribution under the sample design used
  - Square root of variance: standard error of the point estimate
- To form a confidence interval
  - best estimate $\pm$ margin of error
  - best estimate: unbiased point estimate
  - margin of error: "a few" estimated standard errors
  - "a few": multiplier from appropriate distribution based on desired confidence level and sample design
  - 95% confidence level = 5% significance
  - **!!! Caution !!!** important to get all 3 pieces right for correct inference!
    - if best estimate is not unbiased point estimate OR
    - if margin of error does not use correct multiplier OR
    - does not use unbiased estimate of the standard error
    - confidence interval will not have the advertised coverage!
  - Key idea,
    - interval: range of reasonable values for parameter
    - if hypothesized value for parameter lies outside confidence interval, we don't have evidence to support that value at corresponding signifiance level

Inference for Non-Probability Samples

- **Approach I: Quasi-Randomization**
  - Stack the two data sets; non-probability sample may have other response variables we are really insterested in
  - Code NSAMPLE = 1 if member of non-probability sample, NSAMPLE = 0 if member

of probability sample
  - Fit logistic regression model
    - predicting NSAMPLE with common variables weighting non-probability cases by 1 and weighting probability cases by their survey weights
  - Can predict probability of being in non-probability sample, within whatever population is represented by probability sample!
  - Invert predicted probabilities for non-probability sample, treat as survey weights in standard weighted survey analysis
    - Survey Weight = 1 / Predicted Probability
  - **Issue:** How to estimate sampling variance? Not entirely clear…
- **Approach II: Population Modeling**
  - Use predictive modeling to predict aggregate sample quantities (usually totals) on key variables of interest for population units not included in the non-probability sample
  - Compute estimates of interest using estimated totals
    - Weighted Mean = Predicted Total Estimated / Estimated Population Size
    - NOTE: Don't need probability sample with same measures
  - Need good regression models to predict key variables using other auxiliary information available at aggregate level
  - Standard errors can be based on fitted regression models, or using similar replication methods!

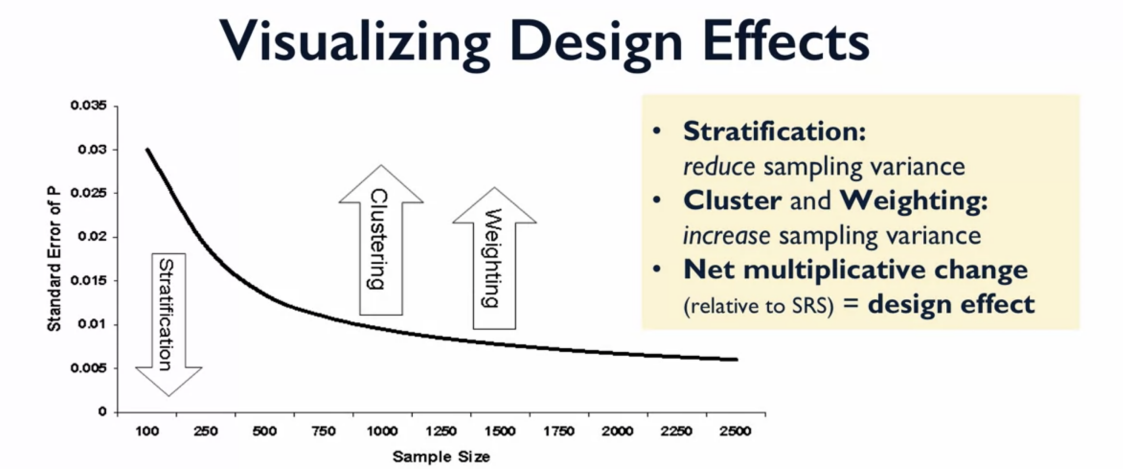
Complex Samples

- **Stratification:** Allocation of overall sample to different "strata", or mutually exclusive divisions of the population. (e.g., regions of the USA)
  - Several different allocation schemes are possible
  - Aim: minimize sampling variance for particular variables given fixed costs
  - Example: Proportionate Allocation
    - If 70% of a population appears in one stratum and 30% in the order;
    - Then 70% of the overall sample would be allocated to the first stratum, and 30% to the second
  - Stratification will eliminate between-stratum variance in means (or totals) on variable from the sampling variance!
  - Important to account for stratification in analysis; else sampling variance may be artificially large -> inferences too conservative, confidence intervals too wide!
- **Clustering**: Random sampling of larger clusters of population elements, possibly across multiple stages (e.g., counties, then segments, then households)
  - Reduces cost of data collection: expensive too visit n randomly sampled units from large and widespread population
  - Clustering reduces cost but tends to increase sampling variance of estimates
    - Why? Units within same cluster have similar (correlated)
    - Variables on variables of interest -> don't measure unique info!
  - Important to account for cluster sampling in analysis, else inferences too liberal, confidence intervals too narrow!
- **Weighting**
  - Complex samples are still probability samples, but if…
    - Multiple stages of cluster sampling within strata
    - Or certain subgroups sampled at higher rates (oversampling)

    * Unequal probabilities of selection for different units
    * Need to account for these unequal probabilites to make unbiased population inferences
  - How? Use of weights in analysis... (partly) defined by inverse of probability of selection
    * If my probability is 1/100 -> my weight is 100
    * I represent myself and 99 others in the population!
    * If my probability is 1/100 and I belong to subgroup where only 50% responded,
    * My adjust weight = (1 / 0.1) x (1 / 0.5) = 200
  - Important need to use weights so estimates are unbiased with respect to the sample design; else possible serious bias!
  - **Drawback:** like cluster sampling, highly variable adjusted survey weights tend to increase sampling variance of weighted estimates (even if they produce unbiased estimates!)

[4]:
```
Image('images/example_3.png', width=800)
```

[4]:



Sampling from a Biased Population

[5]:
```
# recreate the simulations from the video
mean_uofm = 155
sd_uofm = 5
mean_gym = 185
sd_gym = 5
gymperc = .3
total_pop_size = 40000

# create the two subgroups
uofm_students = np.random.normal(
    mean_uofm,
    sd_uofm,
    int(total_pop_size * (1 - gymperc))
)
```

```
students_at_gym = np.random.normal(
    mean_gym,
    sd_gym,
    int(total_pop_size * gymperc)
)

# create the population from the subgroups
population = np.append(uofm_students, students_at_gym)

plt.figure(figsize=(10, 12))

# plot the UofM students only
plt.subplot(3, 1, 1)
sns.histplot(uofm_students, kde=True)
plt.title('UofM Students Only')
plt.xlim([140, 200])

# plot the Gym Goers only
plt.subplot(3, 1, 2)
sns.histplot(students_at_gym, kde=True)
plt.title('Gym Goers Only')
plt.xlim([140, 200])

# plot both groups together
plt.subplot(3, 1, 3)
sns.histplot(population, kde=True)
plt.title('Full Population of UofM Students')
plt.axvline(np.mean(population))
plt.xlim([140, 200])

plt.show()
```

What happens if we sample from the entire population?

[6]:
```
# simulation parameters
number_samps = 5000
samp_size = 50

# get the sampling distribution of the mean from only the gym
mean_distribution = np.empty(number_samps)
for i in range(number_samps):
    random_students = np.random.choice(population, samp_size)
```

```
    mean_distribution[i] = np.mean(random_students)

plt.figure(figsize=(10, 8))

# plotting the population again
plt.subplot(2, 1, 1)
sns.histplot(population, kde=True)
plt.title('Full Population of UofM Students')
plt.axvline(np.mean(population))
plt.xlim([140, 200])

# plotting the sampling distribution
plt.subplot(2, 1, 2)
sns.histplot(mean_distribution, kde=True)
plt.title('Sampling Distribution of the Mean Weight of Gym Goers')
plt.axvline(np.mean(population))
plt.axvline(np.mean(mean_distribution), color='black')
plt.xlim([140, 200])

plt.show()
```

What happens if we take a non-representative sample?

[7]:
```
number_samps = 5000
samp_size = 3

# get the sampling distribution of the mean from only the gym
mean_distribution = np.empty(number_samps)
for i in range(number_samps):
    random_students = np.random.choice(students_at_gym, samp_size)
    mean_distribution[i] = np.mean(random_students)

plt.figure(figsize=(10, 8))

# plotting the population again
plt.subplot(2, 1, 1)
sns.histplot(population, kde=True)
plt.title('Full Population of UofM Students')
plt.axvline(np.mean(population))
plt.xlim([140, 200])

# plotting the sampling distribution
plt.subplot(2, 1, 2)
sns.histplot(mean_distribution, kde=True)
plt.title('Sampling Distribution of the Mean Weight of Gym Goers')
plt.axvline(np.mean(population))
plt.axvline(np.mean(students_at_gym), color='black')
plt.xlim([140, 200])

plt.show()
```

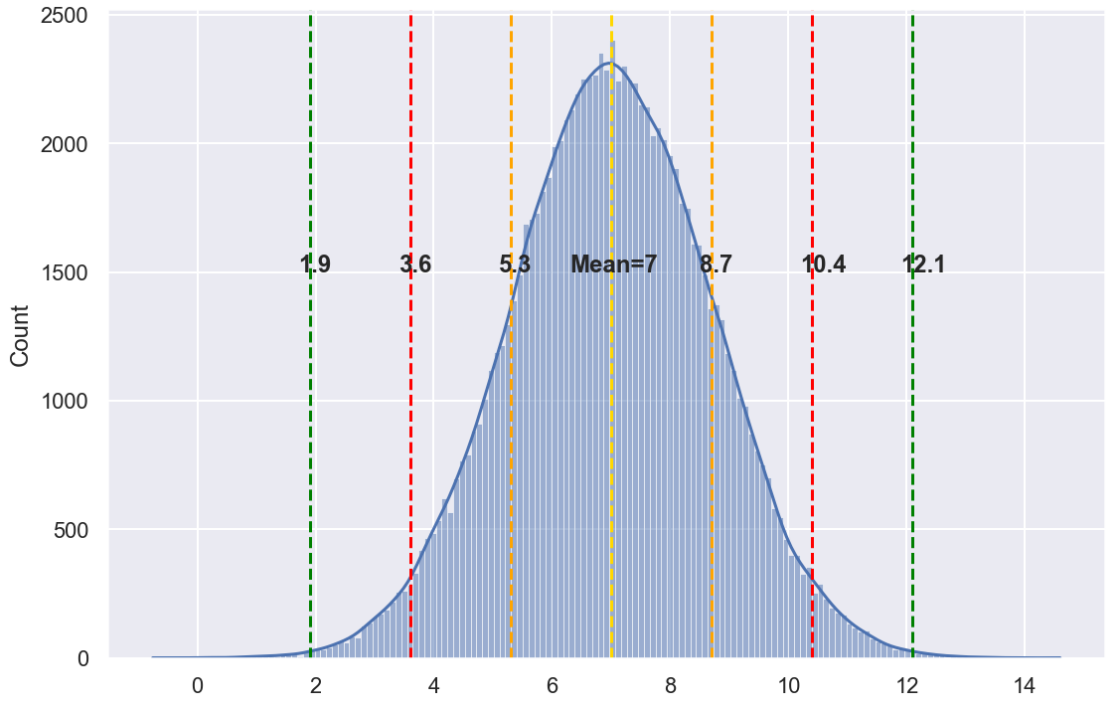
Emprical Rule

[8]:
```
random.seed(1738)
```

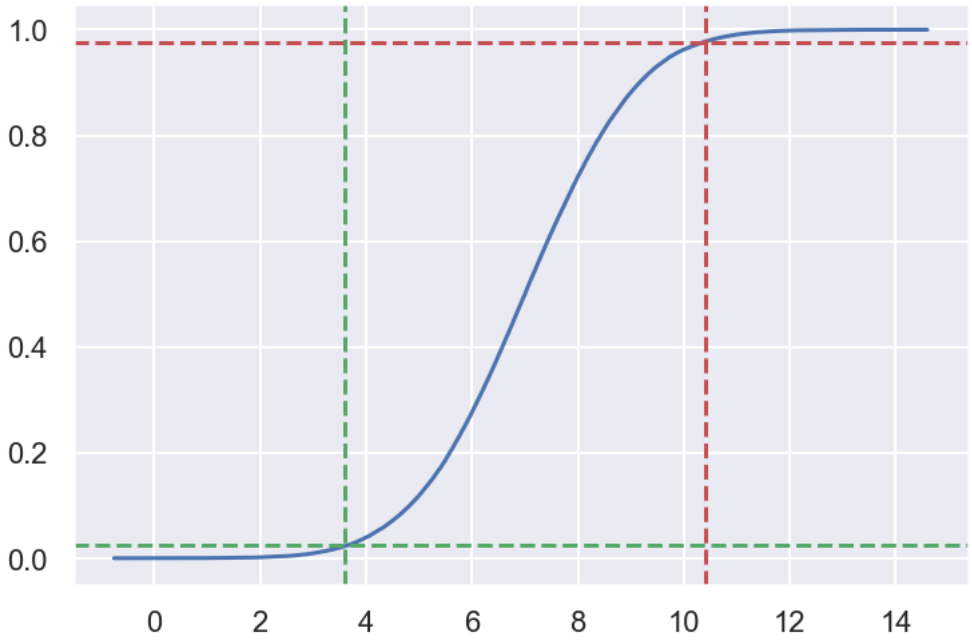
[9]:
```
# create the population like in week-2 lesson
mu = 7
sigma = 1.7

observations = [random.normalvariate(mu, sigma) for _ in range(100000)]
```

[10]:
```
colors = ['orange', 'red', 'green']

plt.figure(figsize=(9, 6))

# bell curve
ax = sns.histplot(observations, kde=True)

# mean line
plt.axvline(mu, color='gold', linestyle='--')
plt.text(6.3, 1500, 'Mean=7', fontweight='bold')
```

```
for i in range(1, 4):
    pos_line = round(mu + i * sigma, 2)
    neg_line = round(mu - i * sigma, 2)

    plt.axvline(pos_line, color=colors[i - 1], linestyle='--')
    plt.axvline(neg_line, color=colors[i - 1], linestyle='--')

    plt.text(pos_line - 0.2, 1500, f'{pos_line}', fontweight='bold')
    plt.text(neg_line - 0.2, 1500, f'{neg_line}', fontweight='bold')
```

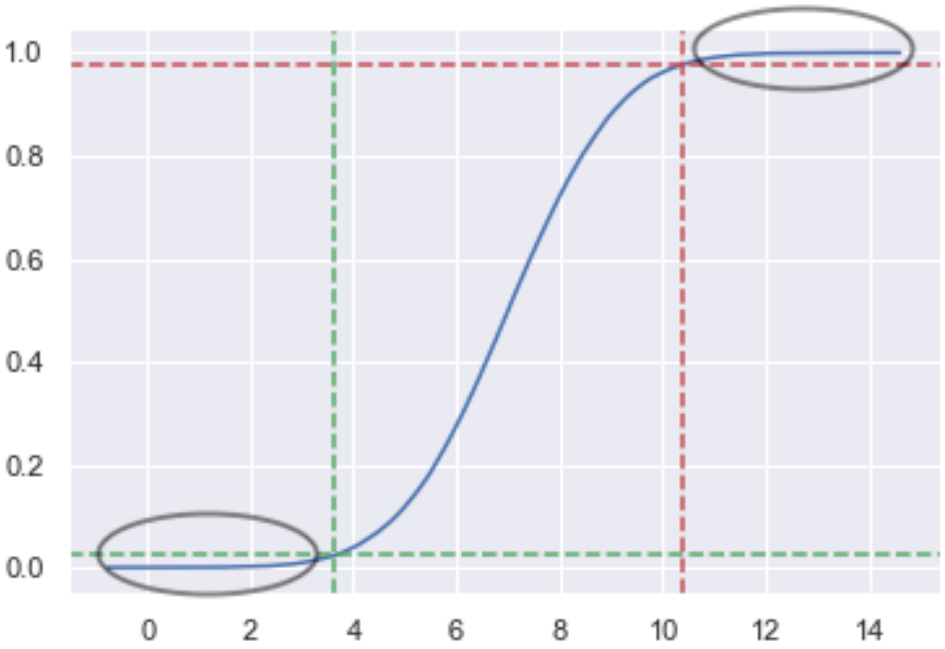


[11]:
```
# ECDF: Empirical Cumulative Density Function
ecdf = ECDF(observations)
plt.plot(ecdf.x, ecdf.y)

plt.axhline(y=0.025, color='g', linestyle='--')
plt.axvline(x=mu - 2 * sigma, color='g', linestyle='--')

plt.axhline(y=0.975, color='r', linestyle='--')
plt.axvline(x= mu + 2 * sigma, color='r', linestyle='--')

plt.savefig('/Users/ytu-egemen-zeytinci/Desktop/img.png')
plt.show()
```



[12]:
```
Image('images/example_4.png', width=400)
```

[12]:



- As you can see in the graph, the parts inside the circle are 5% of the data.
- For example, by looking at this graph, we can say that the number of students who sleep over 10 and less than 4 hours makes up 5%.

[12]:
```
from IPython.display import Image
import numpy as np
```

Introduction to Bayesian In the field of statistics, there are two primary frameworks. One is **frequentist** and the other is **Bayesian**.

- In frequentist statistics, probabilities are made of the world. So, whether something is actually correct or incorrect.
- In Bayesian statistics, probabilities are made in your mind.

For example, I have four chocolates here, three are silver and one is purple, and I'm going to place them into two different bags. So, I had one bag that has two silver chocolates and one bag that has a silver chocolate and a purple chocolate. I'll put that behind my back, and I'll end up picking one of the bags. I don't actually know which bag I picked, but I'll pick one chocolate out of it. In that case, this chocolate is silver. That tells me something about these two bags. I know that there were two ways I could have picked a silver chocolate from the silver-silver bag, but only one way that I could've picked a silver chocolate from the silver-purple bag. In Bayesian statistics, I use the updated information to update the probability that this bag is either silver-silver or silver chocolate. So, I think that there's a two-thirds chance that this bag is silver-silver, and a one-third chance that this bag is silver-purple. In the frequentist framework because I know that I have two bags, this is 50 percent likely to be either bag or equally likely.

- **What's the difference between Python lists and numpy arrays?**

So, the biggest difference is really what you need to store in them, and this is intuitive. Because if you think about lists, you can have many different data types, whereas with the numpy arrays, you can only have one data type. So when you're storing a list, you have to store the value but then also the data type for each value. Whereas in the numpy array, you store the value but then you only need to store the data type once. So, you can see that. This will save you memory. Numpy arrays are also often faster when you're using them in functions. Another way they're different is what you can do with them. For example,

```
import numpy as np

my_list = [1, 2, 3]
my_array = np.array(my_list)

# you can't do it with list because it gives an error
my_array / 3
```

Estimating a Population Proportion with Confidence A hospital based in Ann Arbor MI, C.S. Mott Children's Hospital frequently conducts national polls on children's health. We will be looking at an example of children's safety precautions that parents use when driving.

- What proportion of parents report they use a car seat for all travel with their toddler?
  - Population: Parents with a toddler

  - Parameter of interest: A proportion
  - Construct a 95% confidence interval for the population proportion of parents reporting they use a car seat for all travel with their toddler

- A sample of 659 parents with a toddler was taken and asked if they used a car seat for all travel with their toddler.
  - 540 parents responded "yes" to this question.
  - 95% confidence interval = best estimate ± margin of error
  - 95% confidence interval = $\hat{p}$ ± margin of error

$$SE = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

[2]:
```
# sample size
n = 659

# number of people saying yes
x = 540

# best estimate
p_hat = x / n

# margin of error
moe = 1.96 * np.sqrt((p_hat * (1 - p_hat)) / n)

p_hat - moe, p_hat + moe
```

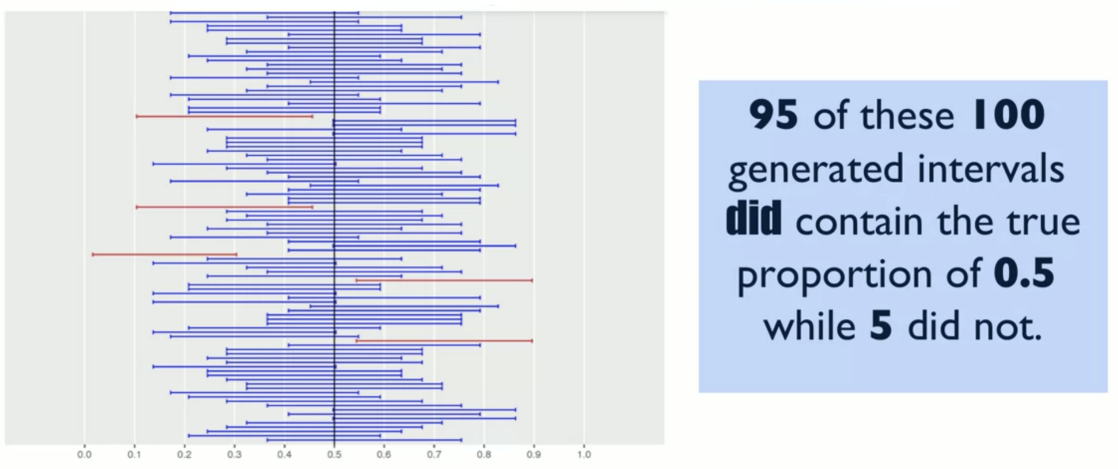
[2]:
```
(0.7900537499137914, 0.8487929875672404)
```

We estimate, with 95% confidence, the population proportion of parents with toddlers who report they use a car seat for all travel with their toddler is somewhere between 0.7900 and 0.8487. *** Based on our sample of 659 parents with toddlers, with 95% confidence, we estimate between %79.0 and %84.9 of all such parents report they use a car seat for all travel with their toddler.

Caution!!! The confidence level refers to the confidence we have in the process we use to make our interval. Not as a probability after our interval was made but how confident we are about the statistical procedure that was used.

[3]:
```
Image('images/example_5.png', width=800)
```

[3]:



As you can see from the graph, with a 95% confidence level, we would except (in the long run) about 95% of the intervals to contain the true proportion.

Assumptions for a Single Population Proportion Confidence Interval

- Best Estimate - in order to get a reliable best estimate, we need a SIMPLE RANDOM SAMPLE
  - Simple Random Sample - a representative subset of the population made up of observations (or subjects) that have an equal probability of being chosen
- Margin of error - in order to use the critical z-value in our calculations, we need a *large enough sample size*
  - But what is large enough?
  - At least 10 of each response outcome (i.e. yes/no)

Conservative Approach & Sample Size Consideration As you know, we defined SE based on the $\hat{p}$ value before. But what if $\hat{p}$ is not accurate? At this point, we want to maximize estimated standard error (SE). Which is maximized when $\hat{p}$=0.5. So, when we put 0.5 instead of $\hat{p}$ in the SE formula, it looks like as follows,

$$CSE = \frac{1}{2\sqrt{n}}$$

And we can called that as **conservative standard error**.

[4]:
```
# sample size
n = 659

# number of people saying yes
x = 540

# best estimate
p_hat = x / n
```

```
# margin of error
moe = 1.96 * (1 / (2 * np.sqrt(n)))

p_hat - moe, p_hat + moe
```

[4]: (0.7812479887710665, 0.8575987487099653)

Also, if we choose the multiplier as 2, the MoE (margin of error) formula looks like as follows,

$$MoE = \frac{1}{\sqrt{n}}$$

The 2 in the denominator and the multiplier 2 cancel each other out. As we said before, sometimes we can use the 2 instead of 1.96 for the %95 confidence.

[5]:
```
# sample size
n = 659

# number of people saying yes
x = 540

# best estimate
p_hat = x / n

# margin of error
moe = 1 / np.sqrt(n)

p_hat - moe, p_hat + moe
```

[5]: (0.7804688993839349, 0.8583778380970969)

If we choose the confidence level as 95%, MoE only depends on the sample size, but for the other confidence levels, the MoE depends on two things,

1) Our confidence level
2) Our sample size

Okey for example, What sample size would we need to have a 95% (conservative) confidence interval with a Margin of Error of only 3%?

$$MoE = \frac{1}{\sqrt{n}}$$

$$n = (\frac{1}{MoE})^2$$

[6]:
```
moe = 0.03
n = np.ceil((1 / moe) ** 2)
```

```
n
```

[6]: 1112.0

What if want to 99% confidence?

$$n = (\frac{2,576}{2.MoE})^2$$

[7]:
```
moe = 0.03
n = np.ceil((2.576 / (2 * moe)) ** 2)
n
```

[7]: 1844.0

Estimating a Difference in Population Proportions with Confidence What is the difference in population proportions of parents reporting that their children age 6-18 have had some swimming lessons between white children and black children?

- Populations - All parents of white children age 6-18 and all parents of black children age 6-18
- Parameter of interest - Difference in population proportions (p1 - p2)
- A sample of 247 parents of black children age 6-18 was taken with 91 saying that their child has had some swimming lessons.
- A sample of 988 parents of white children age 6-18 was taken with 543 saying that their child has had some swimming lessons.

$\hat{p1} - \hat{p2} \pm 1,96\sqrt{\frac{\hat{p1}(1-\hat{p1})}{n1} + \frac{\hat{p2}(1-\hat{p2})}{n2}}$

[8]:
```
n1 = 988
x1 = 543

n2 = 247
x2 = 91

p1 = round(x1 / n1, 2)
p2 = round(x2 / n2, 2)

moe = 1.96 * np.sqrt(((p1 * (1 - p1)) / n1) + ((p2 * (1 - p2)) / n2))
p1 - p2 - moe, p1 - p2 + moe
```

[8]: (0.11226701746507159, 0.2477329825349285)

With 95% confidence, the population proportion of parents with white children who have taken swimming lessons is 11.23 to 24.77% higher than the population proportion of parents with black children who have taken swimming lessons.

Assumptions for difference in population proportions with confidence

- We need to assume that we have two independent random samples.
- We also need large enough sample sizes to assume that the distibution of our estimate is normal. That is, we need $n1\hat{p1}$, $n1(1-\hat{p1})$, $n2\hat{p2}$, $n2(1-\hat{p2})$ to all be at at least 10.
  - In other words, we need at least 10 yes' and at least 10 no's for each sample.

Estimating a Population Mean with Confidence What is the average cartwheel distance (in inches) for adults? - Population - All adults - Parameter of interest - Population mean cartwheel distance μ - Construct a 95% confidence interval for the mean cartwheel distance for the population of all such adults.

Standard error of the sample mean looks like as follows,

$$SE = \frac{\sigma}{\sqrt{n}}$$

But the standard deviation in the formula is the standard deviation, that is, the true standard deviation, which is valid for all adults. But we cannot reach this value, as we do not have a chance to experiment with this for all adults. We use the sample's standard deviation in this situation so, we need to t-multiplier. Here is the confidence interval formula for the mean,

$\hat{x} \pm t(\frac{s}{\sqrt{n}})$

t multiplier comes from a t-distribution with n - 1 degrees of freedom. Here is the t multiplier for 95% confidence,

- n = 25 -> t = 2.064
- n = 1000 -> t = 1.962

!!! Notice that as the population size increases, the t-value seems to approach 1.96, which is valid for the 95% confidence level.

[9]:
```
t = 2.064
n = 25
mean = 82.48
sd = 15.06

moe = t * (sd / np.sqrt(n))
mean - moe, mean + moe
```

[9]:
```
(76.263232, 88.696768)
```

With 95% confidence, the population mean cartwheel distance for all adults is estimated to be between 76.26 inches and 88.70 inches. *** If this procedure were repeated over and over, each time producing a 95% confidence interval estimate, we would expect 95% of those resulting intervals to contain the population mean cartwheel distance.

Assumptions for population mean with confidence

- Data considered a random sample
- Population of responses is normal (else n large helps)

Estimating a Mean Difference for Paired Data

- Want to treat the two sets of values simultaneously
- Other ways paired data arise:
  - Measurements collected on the same individual
  - Measurements collected on matched individual

What is the average difference between the older twin's and younger twin's self-reported education? - Population - All identical twins - Parameter of interest - Population mean difference of self-reported education level - difference = older - younger - Construct a 95% confidence interval for the mean difference of self-reported education for a set of identical twins.

$\hat{x} \pm t(\frac{sd}{\sqrt{n}})$

This formula is very close to the previous one. Differently, we use the difference between standard deviations.

[10]:
```
t = 1.967
n = 340
mean = 0.0838
sd = 0.7627

moe = t * (sd / np.sqrt(n))
mean - moe, mean + moe
```

[10]:
```
(0.0024385560073263424, 0.16516144399267366)
```

With 95% confidence, the population mean difference of the older twin's less the younger twin's self-reported education is estimated to be between 0.0025 years and 0.1652 years.

Assumptions for mean difference for paired data

- We need to assume that we have a random sample of identical twin sets.
- Population of differences is normal (or a large enough sample size can help to bypass this assumption)

Estimating a Difference in Population Means with Confidence (for Independent Groups) I passed this lecture, because there are too many formulas. We can see the example in the coding section of this lecture.

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

warnings.filterwarnings('ignore')
```

[2]:
```
%config InlineBackend.figure_format = 'retina'
```

Estimating a Population Proportion with Confidence

[3]:
```
# sample size
n = 659

# number of people saying yes
x = 540

# best estimate
p_hat = x / n

# margin of error
moe = 1.96 * np.sqrt((p_hat * (1 - p_hat)) / n)

p_hat - moe, p_hat + moe
```

[3]:
```
(0.7900537499137914, 0.8487929875672404)
```

[4]:
```
sm.stats.proportion_confint(x, n)
```

[4]:
```
(0.7900542895874294, 0.8487924478936024)
```

Estimating a Population Mean with Confidence

[5]:
```
df = pd.read_csv('data/cartwheel_data.csv')
```

[6]:
```
t = 2.064

# necessary values
n = len(df)
mean = df['CWDistance'].mean()
sd = df['CWDistance'].std()

moe = t * (sd / np.sqrt(n))
mean - moe, mean + moe
```

[6]:
```
(76.26382957453707, 88.69617042546294)
```

[7]:
```
sm.stats.DescrStatsW(df['CWDistance']).zconfint_mean()
```

[7]:
```
(76.57715593233024, 88.38284406766977)
```

A simple random sample of 500 undergraduates at a large university self-administered a political knowledge test, where the maximum score is 100 and the minimum score is 0. The mean score was 62.5, and the standard deviation of the scores was 10. What is a 95% confidence interval for the overall undergraduate mean at the university?

!!!Caution In this case, we have a standard deviation from sampling. So, we need estimated standard deviation to calculating to confidence interval for the overall. We can calculate the

estimated standard deviation as follows,

$$ESD = \frac{SD}{\sqrt{n}}$$

- SD: Standard deviation
- ESD: Estimated standard deviation for the overall

[8]:
```
n = 500
mean = 62.5
sd = 10

# estimated standard deviation
esd = sd / np.sqrt(n)

upper = mean + 1.96 * esd
lower = mean - 1.96 * esd

lower, upper
```

[8]:
```
(61.62346135282008, 63.37653864717992)
```

Confidence intervals case study using NHANES data

[9]:
```
df = pd.read_csv('data/NHANES.csv')
```

Confidence intervals for one proportion We will calculate the proportions of smokers separately for females and for males. Initially we can compare these two proportions and their corresponding confidence intervals informally, but later we will discuss methods to compare two proportions formally using confidence intervals.

First we replace the numeric codes in the variables of interest with text labels, and set the rare answers other than "yes" and "no" to be missing (so they will automatically be omitted from all the analyses below).

[10]:
```
mapping = {
    1: 'Yes',
    2: 'No',
    7: np.nan,
    9: np.nan
}

gender_map = {
    1: 'Male',
    2: 'Female'
}

df['SMQ020x'] = df.SMQ020.replace(mapping)
df['RIAGENDRx'] = df.RIAGENDR.replace(gender_map)
```

We can now tabulate the numbers of female and male smokers and non-smokers:

[11]:
```
dx = df[['SMQ020x', 'RIAGENDRx']].dropna()
pd.crosstab(dx.SMQ020x, dx.RIAGENDRx)
```

[11]:
```
RIAGENDRx  Female  Male
SMQ020x
No           2066  1340
Yes           906  1413
```

The confidence interval (CI) is constructed using two inputs: the sample proportion of smokers, and the total sample size for smokers and non-smokers combined. We calculate these values next.

[12]:
```
dz = dx \
    .groupby(dx.RIAGENDRx) \
    .agg({'SMQ020x': [lambda x: np.mean(x == 'Yes'), np.size]})

dz.columns = ['Proportion', 'Total_n']
dz
```

[12]:
```
           Proportion  Total_n
RIAGENDRx
Female       0.304845     2972
Male         0.513258     2753
```

Confidence intervals are closely connected to standard errors. Recall that the standard error essentially tells you how far you should expect an estimate to fall from the truth. A confidence interval is an interval that under repeated sampling covers the truth a defined proportion of the time. In most settings, this "coverage probability" is set to 95%.

It turns out that in many settings, a 95% confidence interval can be constructed as the interval consisting of all points that are within two (or 1.96) standard errors of the point estimate. More concisely, the confidence interval approximately spans from e - 2 • SE to e + 2 • SE, where e is the point estimate and SE is the standard error.

Since the standard error plays such an important role here, we calculate it separately first.

[13]:
```
p = dz.Proportion.Female
n = dz.Total_n.Female

se_female = np.sqrt(p * (1 - p) / n)
print(se_female)

p = dz.Proportion.Male
n = dz.Total_n.Male

se_male = np.sqrt(p * (1 - p) / n)
print(se_male)
```

```
0.008444152146214435
0.009526078653689868
```

We can see that the standard errors for the estimated proportions of females and males who smoke are similar, and are each around 1% (since we are studying a proportion here, 0.01 corresponds to a 1 percentage point change in the smoking rate).

The standard error for a proportion is maximized when the true proportion is around 1/2, and gets smaller as the true proportion approaches either 0 or 1. The estimated male smoking proportion is closer to 1/2 than the estimated female smoking proportion, and the male sample size is smaller than the female sample size. Both of these factors lead to the male standard error being larger than the female standard error, although the difference is very small in this case.

Next we calculate the 95% confidence intervals for the proportions of female and male smokers using the formula for the one-sample confidence interval for a proportion:

[14]:
```
p = dz.Proportion.Female
n = dz.Total_n.Female

lcb = p - 1.96 * se_female
ucb = p + 1.96 * se_female

lcb, ucb
```

[14]:
```
(0.288294683866098, 0.32139576027925865)
```

[15]:
```
p = dz.Proportion.Male
n = dz.Total_n.Male

lcb = p - 1.96 * se_male
ucb = p + 1.96 * se_male

lcb, ucb
```

[15]:
```
(0.49458714955108174, 0.531929377873546)
```

These results indicate that any population proportion (for male lifetime smokers) between 0.493 and 0.531 would be compatible with the NHANES data.

In a routine data analysis, we do not need to calculate these intervals manually. We can use the Statsmodels library to calculate the CI for us in one line:

[16]:
```
sm.stats.proportion_confint(906, 906 + 2066)
```

[16]:
```
(0.2882949879861214, 0.32139545615923526)
```

[17]:
```
sm.stats.proportion_confint(1413, 1413 + 1340)
```

[17]:
```
(0.49458749263718593, 0.5319290347874418)
```

Confidence intervals comparing two independent proportions The confidence intervals for the proportions of female and male smokers shown above are quite narrow and do not overlap. This suggests that there is a substantial difference between the lifetime smoking rates for women and men. However there is no explicit information here about how different the two population proportions might be. To address this question, we can form a confidence interval for the difference between the proportion of females who smoke and the proportion of males who smoke.

The point estimate of the difference between female and male smoking rates is -0.208 (0.305 - 0.513). That is, the smoking rate is about 20 percentage points higher in men than in women. This difference of around 20 percentage points is only a point estimate of the underlying true value – it is not exactly equal to the difference between the unknown proportions of females and males who smoke in the population. A confidence interval helps us assess how far the estimated difference may be from the true difference.

As above, we start with the standard error. The difference between two sample proportions based on independent data has a standard error that reflects the combined uncertainty in the two proportions being differenced. This standard error can be calculated very easily. If SE1 and SE2 are the standard errors for two proportions, then sqrt(SE1^2 + SE2^2) is the standard error for the difference of these proportions (`sqrt` is the square root function). Note that this formula is only accurate if the two sample proportions being differenced are independent.

In the next cell we calculate the standard error for the difference between the proprotion of females who smoke and the proportion of males who smoke.

[18]:
```
se_diff = np.sqrt(se_female ** 2 + se_male ** 2)
se_diff
```

[18]:
```
0.012729881381407434
```

The standard error of around 0.013 indicates that the estimated difference statistic -0.208 is expected to fall around 0.013 units from the true value. We do not know in which direction the error lies, and we do not know that the error is exactly 0.013, only that it is around this large on average. For most purposes, a standard error of 0.013 relative to an observed difference of -0.21 would be considered very small. That is, we have a very accurate estimate of the difference between smoking rates in women and in men.

Now that we have the standard error, we can construct a 95% confidence interval for the difference in proportions by taking the estimate and subtracting and adding two (or 1.96) standard errors from it.

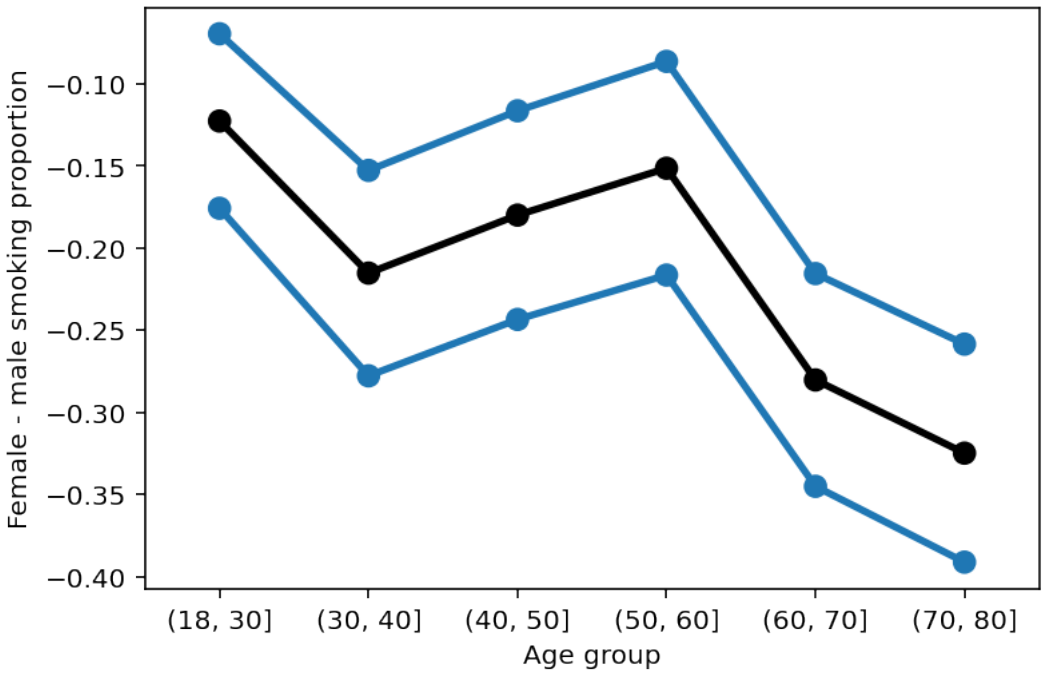
[19]:
```
d = dz.Proportion.Female - dz.Proportion.Male
lcb = d - 2 * se_diff
ucb = d + 2 * se_diff
lcb, ucb
```

[19]:
```
(-0.2338728044024504, -0.18295327887682067)
```

The 95% confidence interval above shows us that any value for the difference of population proportions (between females and males) lying between -0.233 and -0.183 is consistent with the observed data.

Confidence intervals for subpopulations Since smoking rates vary strongly with age, it might be more informative to stratify the data into homogeneous age bands and compare the proportions of female and male smokers within each age band. We can also calculate the 95% confidence interval for this difference within each age band. These data can be displayed as a plot, with the difference in proportions plotted as a curve. The confidence intervals can then be used to construct a "confidence band" around the estimates.

[20]:
```
# calculate the smoking rates within age/gender groups
df['agegrp'] = pd.cut(df.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])
pr = df \
    .groupby(['agegrp', 'RIAGENDRx']) \
    .agg({'SMQ020x': lambda x: np.mean(x == 'Yes')}) \
    .unstack()
pr.columns = ['Female', 'Male']

# the number of people for each calculated proportion
dn = df \
    .groupby(['agegrp', 'RIAGENDRx']) \
    .agg({'SMQ020x': np.size}).unstack()
dn.columns = ['Female', 'Male']

# standard errors for each proportion
se = np.sqrt(pr * (1 - pr) / dn)

# standard error for the difference in female/male smoking rates in every age␣
 ↪band
se_diff = np.sqrt(se.Female ** 2 + se.Male ** 2)

# the difference in smoking rates between genders
pq = pr.Female - pr.Male

x = np.arange(pq.size)

pp = sns.pointplot(x, pq.values, color='black')
sns.pointplot(x, pq - 2 * se_diff)
sns.pointplot(x, pq + 2 * se_diff)

pp.set_xticklabels(pq.index)
pp.set_xlabel('Age group')
pp.set_ylabel('Female - male smoking proportion');
```



The plot above shows for each age band, the point estimate of the difference in smoking rates between genders (black dot), and the lower and upper end points of the 95% confidence interval (blue points). Based on this plot, we see that in the United States, smoking is more common in men than in women, not just overall, but also in every one of the age bands. The difference is largest for older people – for people older than 60, the smoking rate for males is around 30 percentage points greater than the smoking rate for females, while for people younger than 30, the smoking rate for males is only around 15 percentage points greater than the smoking rate for females.

Also note that the 95% confidence bands shown above are much wider than the 95% confidence intervals for the data that were not stratified by age. Stratifying by age leads to smaller sample sizes, which in turn results in wider confidence intervals.

Confidence intervals for the mean In this section, we discuss how to construct confidence intervals for the mean. First note that the proportion discussed above is also a mean – for example, if the data are 0, 1, 0, then the mean is 1/3, which is also the proportion of 1's in the data. However the proportion has the special property that the variance is completely determined by the mean. That is why we constructed the standard errors for the sample proportion above using $p \bullet (1 - p)$ as the variance. In general, the variance of quantitative data will not be a function of the mean, as this is a very special property of binary data. Therefore, in general we must estimate the variance as a separate step after estimating the mean.

To illustrate the construction of confidence intervals for the population mean of a quantitative variable, we will use the body mass inde (BMI) data from NHANES. To begin, we calculate the mean BMI for all women and for all men in the NHANES sample.

[21]:
```
df.groupby('RIAGENDRx').agg({'BMXBMI': np.mean})
```

[21]:
```
               BMXBMI
RIAGENDRx
Female     29.939946
Male       28.778072
```

The numbers in the first column of the table above are estimates of the population mean BMI for all women and for all men in the United States (the population that the NHANES study represents). As with the sample proportions, these numbers are not exactly equal to the mean BMI for all women and men, they are only estimates. To establish the uncertainty for these estimates, we can use the standard errors for these two estimated means.

The standard error for the mean based on an independent and identically distributed sample is equal to the standard deviation of the variable divided by the square root of the sample size. We next calculate all the relevant values needed to compute the standard error.

[22]:
```
df.groupby('RIAGENDRx').agg({'BMXBMI': [np.mean, np.std, np.size]})
```

[22]:
```
               BMXBMI
                 mean       std    size
RIAGENDRx
Female     29.939946  7.753319  2976.0
Male       28.778072  6.252568  2759.0
```

We can now calculate the standard error of the mean BMI for women and for men:

[23]:
```
sem_female = 7.753 / np.sqrt(2976)
sem_male = 6.253 / np.sqrt(2759)
sem_female, sem_male
```

[23]:
```
(0.14211938534506902, 0.119045388988243)
```

We see that the sample mean BMI for women is expected to be off by around 0.14 relative to the population mean BMI for women, and the sample mean BMI for men is expected to be off by around 0.12 relative to the population mean BMI for men.

The standard error of the mean for women is slightly larger for women than for men. The reason for this is that even though the NHANES sample size for women is slightly larger than that for men, the data for women appears to be more spread out. The greater standard deviation for the female BMI values leads in turn to less precision when estimating the population mean BMI for females.

As was the case for proportions, the 95% confidence interval for the mean can be calculated by taking the estimate plus and minus 2 (or 1.96) times the standard error. The 95% confidence interval for female BMI is thus calculated as follows:

[24]:
```
lcb_female = 29.94 - 1.96 * 7.753 / np.sqrt(2976)
ucb_female = 29.94 + 1.96 * 7.753 / np.sqrt(2976)
lcb_female, ucb_female
```

[24]:
```
(29.661446004723665, 30.218553995276338)
```

[25]:
```
female_bmi = df.loc[df.RIAGENDRx == 'Female', 'BMXBMI'].dropna()
sm.stats.DescrStatsW(female_bmi).zconfint_mean()
```

[25]:
```
(29.659875498090155, 30.22001580625768)
```

Confidence intervals for the difference between two means Now we turn to studying the difference between two means, taking the difference between mean female and male BMI for illustration. As discussed above, the standard error for the difference of two means taken from independent samples is sqrt(SE1^2 + SE2^2), where SE1 and SE2 are the standard errors for the two means being compared. Below we see that this gives us a value around 0.19 when comparing the female BMI to the male BMI. This is substantially larger than either the SEM for estimating the female mean (0.14) or the SEM for estimating the male mean (0.12). It is expected that the standard error for the difference between two means is greater than the standard errors for estimating a single mean, since the uncertainty of both gender-specific proportions impacts the statistic.

[26]:
```
sem_diff = np.sqrt(sem_female ** 2 + sem_male ** 2)
sem_diff
```

[26]:
```
0.18539073420811059
```

[27]:
```
bmi_diff = 29.94 - 28.78
lcb = bmi_diff - 2 * sem_diff
ucb = bmi_diff + 2 * sem_diff
lcb, ucb
```

[27]:
```
(0.789218531583779, 1.5307814684162213)
```

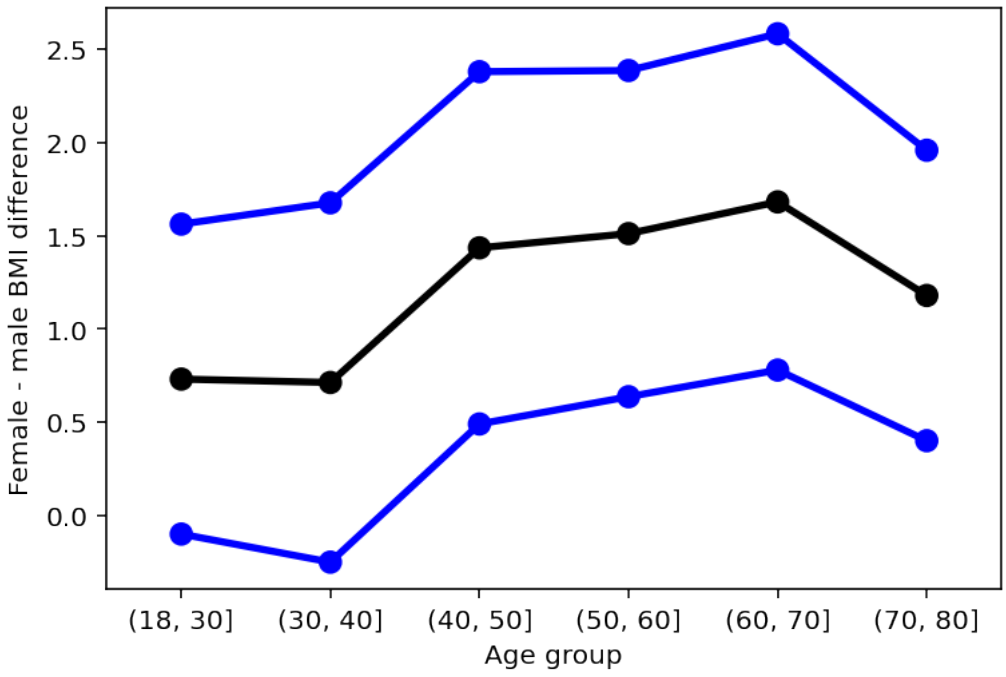
This finding indicates that while the point estimate shows that the women in our sample have around 1.1 unit greater BMI than the men in our sample, the true difference between the mean for all women in the population and for all men in the population could fall between 0.79 and 1.53, and still be consistent with the observed data.

Age-stratified confidence intervals As a final example, we refine the analysis above by considering the difference of mean BMI values between females and males within age bands. We see below that the overall average difference of 1.1 units results from differences that are very different based on age. Specifically, the difference between female and male BMI is much smaller than 1.1 for younger people, and much larger than 1.1 for older people.

Since the confidence bands for people under 40 contain 0, the data are consistent with there being no difference between female and male BMI in this age range. For people older than 40, a hypothetical zero difference between the mean BMI values for females and males is not very consistent with the data. Informally, we can say that the data strongly suggest that the female mean BMI is greater than the male mean BMI in this age band, with the difference being anywhere from 0.5 to 2 units.

[28]:
```
# calculate the mean, SD, and sample size for BMI within age/gender groups
df['agegrp'] = pd.cut(df.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])
pr = df \
```

```
    .groupby(['agegrp', 'RIAGENDRx']) \
    .agg({'BMXBMI': [np.mean, np.std, np.size]}) \
    .unstack()

# calculate the SEM for females and for males within each age band
pr['BMXBMI', 'sem', 'Female'] = pr['BMXBMI', 'std', 'Female'] / np.
 ↪sqrt(pr['BMXBMI', 'size', 'Female'])
pr['BMXBMI', 'sem', 'Male'] = pr['BMXBMI', 'std', 'Male'] / np.
 ↪sqrt(pr['BMXBMI', 'size', 'Male'])

# calculate the mean difference of BMI between females and males within each␣
 ↪age band, also  calculate
# its SE and the lower and upper limits of its 95% CI
pr['BMXBMI', 'mean_diff', ''] = pr['BMXBMI', 'mean', 'Female'] - pr['BMXBMI',␣
 ↪'mean', 'Male']
pr['BMXBMI', 'sem_diff', ''] = np.sqrt(pr['BMXBMI', 'sem', 'Female'] ** 2 +␣
 ↪pr['BMXBMI', 'sem', 'Male'] ** 2)
pr['BMXBMI', 'lcb_diff', ''] = pr['BMXBMI', 'mean_diff', ''] - 1.96 *␣
 ↪pr['BMXBMI', 'sem_diff', '']
pr['BMXBMI', 'ucb_diff', ''] = pr['BMXBMI', 'mean_diff', ''] + 1.96 *␣
 ↪pr['BMXBMI', 'sem_diff', '']

# plot the mean difference in black and the confidence limits in blue
x = np.arange(pr.shape[0])
pp = sns.pointplot(x, pr['BMXBMI', 'mean_diff', ''], color='black')
sns.pointplot(x, pr['BMXBMI', 'lcb_diff', ''], color='blue')
sns.pointplot(x, pr['BMXBMI', 'ucb_diff', ''], color='blue')

pp.set_xticklabels(pr.index)
pp.set_xlabel('Age group')
pp.set_ylabel('Female - male BMI difference');
```



Inter-group and intra-group differences: As the sample size grows, estimates become increasingly precise, but it is important to remember that a highly precise estimate for the mean does not imply that individuals within a population do not vary from each other. To put the differences shown above in context, below we show the underlying summaries on which the plot above was based. Note that the standard deviation of BMI within both females and males ranges from around 5 to around 8 depending on the age band. This means, for example, that two randomly-selected males will tend to have BMI values that differ by around 6 units. This is a far greater difference than the mean difference of up to around 1.5 BMI units between females and males. Thus, while there is a tendency for females to have slightly higher BMI than males, the heterogeneity within genders is substantially greater than the difference of means between genders.

Confidence intervals and sample size Confidence intervals reflect the precision of an estimate, which is largely driven by the amount of data used to construct the estimate. We can explore the relationship between precision and sample size by subampling data from NHANES and calculating confidence intervals for the subsamples. Below we calculate confidence intervals based on subsamples of size 100, 200, 400, and 800.

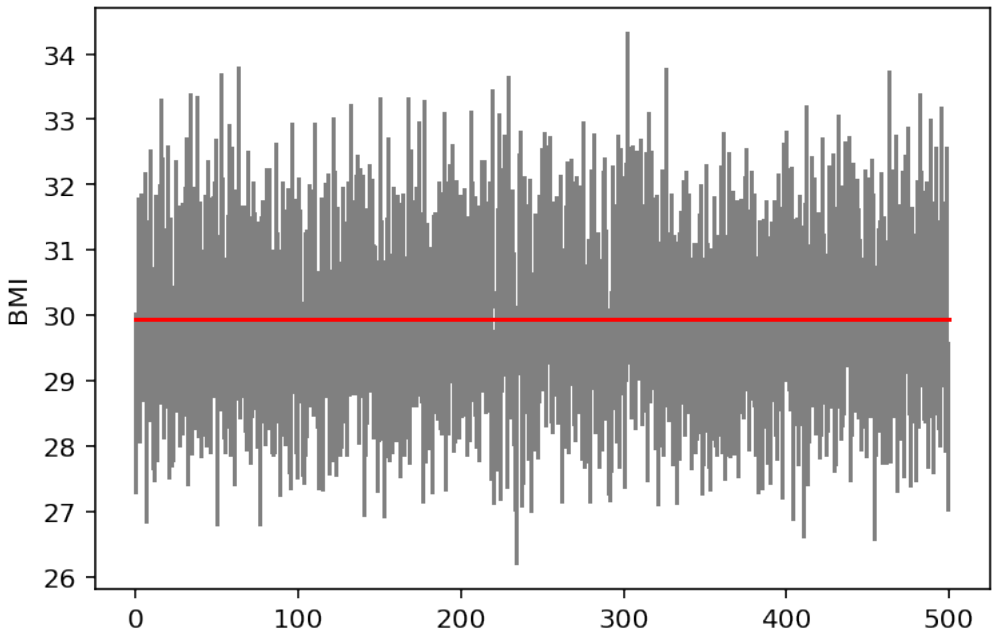
A wider confidence interval implies that we have less precision in our estimate. In the simulation below, we calculate the average width of the confidence intervals constructed for each sample size. We see that the confidence interval steadily becomes shorter as the sample size grows. For most settings, the confidence interval will become around half as wide when the sample size is increased by a factor of 4. Below we see this scaling when the sample size increases from 100 to 400, and when it increases from 200 to 800, both of which are increases by a factor of 4.

[29]:
```
dx = df.loc[df.RIAGENDRx == 'Female', ['RIAGENDRx', 'BMXBMI']].dropna()

all_cis = []
for n in 100, 200, 400, 800:
    cis = []

    for i in range(500):
        dz = dx.sample(n)
        ci = sm.stats.DescrStatsW(dz.BMXBMI).zconfint_mean()
        cis.append(ci)

    cis = np.asarray(cis)
    mean_width = cis[:, 1].mean() - cis[:, 0].mean()

    print(n, mean_width)
    all_cis.append(cis)
```

```
100 3.046386477701411
200 2.1391359377946593
400 1.5186830707329158
800 1.0751063890699015
```

It is also informative to plot the individual confidence intervals, computed for 500 subsamples of size 100, to see how they vary. The vertical grey bars below each correspond to a confidence interval. The red horizontal line is the mean BMI calculated using the entire data set, which can be taken as a proxy for othe population mean. While the individual intervals are quite different from each other, it appears that the vast majority of them cover the population value.

[30]:
```
ci = all_cis[0]

for j, x in enumerate(ci):
    plt.plot([j, j], x, color='grey')
    plt.gca().set_ylabel('BMI')

mn = dx.BMXBMI.mean()
plt.plot([0, 500], [mn, mn], color='red');
```



Hypothesis Testing From lecture, we know that hypothesis testing is a critical tool in determing what the value of a parameter could be.

We know that the basis of our testing has two attributes:

Null Hypothesis: H_0

Alternative Hypothesis: H_a

The tests we have discussed in lecture are:

- One Population Proportion
- Difference in Population Proportions
- One Population Mean
- Difference in Population Means

In this tutorial, I will introduce some functions that are extremely useful when calculating a t-statistic and p-value for a hypothesis test.

Let's quickly review the following ways to calculate a test statistic for the tests listed above.

The equation is:

$$\frac{Best\ Estimate - Hypothesized\ Estimate}{Standard\ Error\ of\ Estimate}$$

We will use the examples from our lectures and use python functions to streamline our tests.

[1]:
```
import numpy as np
import pandas as pd
import scipy.stats.distributions as dist
import statsmodels.api as sm
```

One Population Proportion Research Question:

In previous years 52% of parents believed that electronics and social media was the cause of their teenager's lack of sleep. Do more parents today believe that their teenager's lack of sleep is caused due to electronics and social media?

Population: Parents with a teenager (age 13-18)
Parameter of Interest: p
Null Hypothesis: p = 0.52
Alternative Hypthosis: p > 0.52 (note that this is a one-sided test)

1018 Parents

56% believe that their teenager's lack of sleep is caused due to electronics and social media.

[2]:
```
n = 1018
p_null = .52
p_hat = .56

sm.stats.proportions_ztest(
    p_hat * n,
    n,
    p_null,
    alternative='larger',
    prop_var=0.52
)
```

[2]:
```
(2.5545334262132955, 0.005316510991822442)
```

Difference in Population Proportions Research Question:

Is there a significant difference between the population proportions of parents of black children and parents of Hispanic children who report that their child has had some swimming lessons?

Populations: All parents of black children age 6-18 and all parents of Hispanic children age 6-18
Parameter of Interest: p1 - p2, where p1 = black and p2 = hispanic
Null Hypothesis: p1 - p2 = 0
Alternative Hypthosis: p1 - p2 ≠ 0

91 out of 247 (36.8%) sampled parents of black children report that their child has had some swimming lessons.

120 out of 308 (38.9%) sampled parents of Hispanic children report that their child has had some swimming lessons.

[5]:
```
# sample sizes
n1 = 247
n2 = 308

# number of parents reporting that their child had some swimming lessons
y1 = 91
y2 = 120

# estimates of the population proportions
p1 = round(y1 / n1, 2)
p2 = round(y2 / n2, 2)

# estimate of the combined population proportion
p_hat = (y1 + y2) / (n1 + n2)

# estimate of the variance of the combined population proportion
va = p_hat * (1 - p_hat)

# estimate of the standard error of the combined population proportion
se = np.sqrt(va * (1 / n1 + 1 / n2))

# test statistic and its p-value
test_stat = (p1 - p2) / se
p_value = 2 * dist.norm.cdf(-np.abs(test_stat))

# print the test statistic its p-value
print('Test Statistic')
print(round(test_stat, 2))

print('\nP-Value')
print(round(p_value, 2))
```

```
Test Statistic
-0.48

P-Value
0.63
```

One Population Mean Research Question:

Is the average cartwheel distance (in inches) for adults more than 80 inches?

Population: All adults
Parameter of Interest: μ, population mean cartwheel distance
Null Hypothesis: $\mu = 80$
Alternative Hypthosis: $\mu > 80$

25 Adults

$\mu = 82.46$

$\sigma = 15.06$

[6]:
```
df = pd.read_csv('data/cartwheel_data.csv')
```

[7]:
```
n = len(df)
mean = df['CWDistance'].mean()
sd = df['CWDistance'].std()
(n, mean, sd)
```

[7]:
```
(25, 82.48, 15.058552387264852)
```

[8]:
```
sm.stats.ztest(df['CWDistance'], value=80, alternative='larger')
```

[8]:
```
(0.8234523266982029, 0.20512540845395266)
```

Difference in Population Means Research Question:

Considering adults in the NHANES data, do males have a significantly higher mean Body Mass Index than females?

Population: Adults in the NHANES data.
Parameter of Interest: $\mu_1 - \mu_2$, Body Mass Index.
Null Hypothesis: $\mu_1 = \mu_2$
Alternative Hypthosis: $\mu_1 \neq \mu_2$

2976 Females $\mu_1 = 29.94$
$\sigma_1 = 7.75$

2759 Male Adults
$\mu_2 = 28.78$
$\sigma_2 = 6.25$

$\mu_1 - \mu_2 = 1.16$

[9]:
```
df = pd.read_csv('data/NHANES.csv')
```

[10]:
```
females = df[df['RIAGENDR'] == 2]
male = df[df['RIAGENDR'] == 1]
```

[11]:
```
n1 = len(females)
mu1 = females['BMXBMI'].mean()
sd1 = females['BMXBMI'].std()

(n1, mu1, sd1)
```

[11]:
```
(2976, 29.93994565217392, 7.753318809545674)
```

[12]:
```
n2 = len(male)
mu2 = male['BMXBMI'].mean()
sd2 = male['BMXBMI'].std()

(n2, mu2, sd2)
```

[12]:
```
(2759, 28.778072111846942, 6.2525676168014614)
```

[13]:
```
sm.stats.ztest(females['BMXBMI'].dropna(), male['BMXBMI'].dropna())
```

[13]:
```
(6.1755933531383205, 6.591544431126401e-10)
```

Assumptions Consistency

One Population Proportion

- Sample can be considered a simple random sample
- Large enough sample size
  - Confidence Interval: At least 10 of each outcome
  - Hypothesis Test: At least 10 of each outcome

Two Population Proportions

- Samples can be considered two simple random samples
- Samples can be considered independent of one another
- Large enough sample sizes ()
  - Confidence Interval: At least 10 of each outcome
  - Hypothesis Test: At least 10 of each outcome - Where (the common population proportion estimate)

One Population Mean

- Sample can be considered a simple random sample
- Sample comes from a normally distributed population
  - This assumption is less critical with a large enough sample size (application of the C.L.T.)

One Population Mean Difference

- Sample of differences can be considered a simple random sample
- Sample of differences comes from a normally distributed population of differences
  - This assumption is less critical with a large enough sample size (application of the C.L.T.)

Two Population Means

- Samples can be considered a simple random samples
- Samples can be considered independent of one another
- Samples each come from normally distributed populations
  - This assumption is less critical with a large enough sample size (application of the C.L.T.)
- Populations have equal variances – pooled procedure used

– If this assumption cannot be made, unpooled procedure used

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import scipy.stats.distributions as dist
import statsmodels.api as sm
```

[2]:
```
df = pd.read_csv('data/NHANES.csv')
```

[3]:
```
mapping = {
    1: 'Yes',
    2: 'No',
    7: np.nan,
    9: np.nan
}

gender_map = {
    1: 'Male',
    2: 'Female'
}

citizen_map = {
    1: 'Yes',
    2: 'No',
    7: np.nan,
    9: np.nan
}

df['SMQ020x'] = df.SMQ020.replace(mapping)
df['RIAGENDRx'] = df.RIAGENDR.replace(gender_map)
df['DMDCITZN'] = df.DMDCITZN.replace(citizen_map)
```

Hypothesis tests for one proportions The most basic hypothesis test may be the one-sample test for a proportion. This test is used if we have specified a particular value as the null value for the proportion, and we wish to assess if the data are compatible with the true parameter value being equal to this specified value. One-sample tests are not used very often in practice, because it is not very common that we have a specific fixed value to use for comparison.

For illustration, imagine that the rate of lifetime smoking in another country was known to be 40%, and we wished to assess whether the rate of lifetime smoking in the US were different from 40%. In the following notebook cell, we carry out the (two-sided) one-sample test that the population proportion of smokers is 0.4, and obtain a p-value of 0.43. This indicates that the NHANES data are compatible with the proportion of (ever) smokers in the US being 40%.

[4]:
```
x = df.SMQ020x.dropna() == 'Yes'

p = x.mean()
se = np.sqrt(0.4 * 0.6 / len(x))
test_stat = (p - 0.4) / se

p_value = 2 * dist.norm.cdf(-np.abs(test_stat))
test_stat, p_value
```

[4]:
```
(0.7823563854332805, 0.4340051581348052)
```

The following cell carries out the same test as performed above using the Statsmodels library. The results in the first (default) case below are slightly different from the results obtained above because Statsmodels by default uses the sample proportion instead of the null proportion when computing the standard error. This distinction is rarely consequential, but we can specify that the null proportion should be used to calculate the standard error, and the results agree exactly with what we calculated above. The first two lines below carry out tests using the normal approximation to the sampling distribution of the test statistic, and the third line below carries uses the exact binomial sampling distribution. We can see here that the p-values are nearly identical in all three cases. This is expected when the sample size is large, and the proportion is not close to either 0 or 1.

[5]:
```
print(sm.stats.proportions_ztest(x.sum(), len(x), 0.4))
print(sm.stats.proportions_ztest(x.sum(), len(x), 0.4, prop_var=0.4))
print(sm.stats.binom_test(x.sum(), len(x), 0.4))
```

```
(0.7807518954896244, 0.43494843171868214)
(0.7823563854332805, 0.4340051581348052)
0.4340360854459431
```

Hypothesis tests for two proportions Comparative tests tend to be used much more frequently than tests comparing one population to a fixed value. A two-sample test of proportions is used to assess whether the proportion of individuals with some trait differs between two subpopulations. For example, we can compare the smoking rates between females and males. Since smoking rates vary strongly with age, we do this in the subpopulation of people between 20 and 25 years of age. In the cell below, we carry out this test without using any libraries, implementing all the test procedures covered elsewhere in the course using Python code. We find that the smoking rate for men is around 10 percentage points greater than the smoking rate for females, and this difference is statistically significant (the p-value is around 0.01).

[6]:
```
dx = df[['SMQ020x', 'RIDAGEYR', 'RIAGENDRx']].dropna()
dx = dx.loc[(dx.RIDAGEYR >= 20) & (dx.RIDAGEYR <= 25), :]

p = dx \
    .groupby('RIAGENDRx')['SMQ020x'] \
    .agg([lambda z: np.mean(z == 'Yes'), 'size'])

p.columns = ['Smoke', 'N']
```

```
p_comb = (dx.SMQ020x == 'Yes').mean()
va = p_comb * (1 - p_comb)
se = np.sqrt(va * (1 / p.N.Female + 1 / p.N.Male))

test_stat = (p.Smoke.Female - p.Smoke.Male) / se
pvalue = 2 * dist.norm.cdf(-np.abs(test_stat))

test_stat, pvalue
```

[6]: (-2.5833303066279414, 0.009785159057508375)

Essentially the same test as above can be conducted by converting the "Yes"/"No" responses to numbers (Yes=1, No=0) and conducting a two-sample t-test, as below:

[7]:
```
mapping = {'Yes': 1, 'No': 0}

dx_females = dx.loc[dx.RIAGENDRx == "Female", 'SMQ020x'].replace(mapping)
dx_males = dx.loc[dx.RIAGENDRx == 'Male', 'SMQ020x'].replace(mapping)

sm.stats.ttest_ind(dx_females, dx_males)
```

[7]: (-2.5949731446269344, 0.009725902321212 54, 522.0)

Hypothesis tests comparing means Tests of means are similar in many ways to tests of proportions. Just as with proportions, for comparing means there are one and two-sample tests, z-tests and t-tests, and one-sided and two-sided tests. As with tests of proportions, one-sample tests of means are not very common, but we illustrate a one sample test in the cell below. We compare systolic blood pressure to the fixed value 120 (which is the lower threshold for "pre-hypertension"), and find that the mean is significantly different from 120 (the point estimate of the mean is 126).

[8]:
```
dx = df[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']].dropna()
dx = dx.loc[(dx.RIDAGEYR >= 40) & (dx.RIDAGEYR <= 50) & (dx.RIAGENDRx ==␣
 ↪'Male'), :]
sm.stats.ztest(dx.BPXSY1, value=120)
```

[8]: (7.469764137102597, 8.033869113167905e-14)

In the cell below, we carry out a formal test of the null hypothesis that the mean blood pressure for women between the ages of 50 and 60 is equal to the mean blood pressure of men between the ages of 50 and 60. The results indicate that while the mean systolic blood pressure for men is slightly greater than that for women (129 mm/Hg versus 128 mm/Hg), this difference is not statistically significant.

There are a number of different variants on the two-sample t-test. Two often-encountered variants are the t-test carried out using the t-distribution, and the t-test carried out using the normal approximation to the reference distribution of the test statistic, often called a z-test. Below we

display results from both these testing approaches. When the sample size is large, the difference between the t-test and z-test is very small.

[9]:
```
dx = df[['BPXSY1', 'RIDAGEYR', 'RIAGENDRx']].dropna()
dx = dx.loc[(dx.RIDAGEYR >= 50) & (dx.RIDAGEYR <= 60), :]

bpx_female = dx.loc[dx.RIAGENDRx == 'Female', 'BPXSY1']
bpx_male = dx.loc[dx.RIAGENDRx == 'Male', 'BPXSY1']

print(bpx_female.mean(), bpx_male.mean())
print(sm.stats.ztest(bpx_female, bpx_male))
print(sm.stats.ttest_ind(bpx_female, bpx_male))
```

```
127.92561983471074 129.23829787234044
(-1.105435895556249, 0.2689707570859362)
(-1.105435895556249, 0.26925004137768577, 952.0)
```

Another important aspect of two-sample mean testing is "heteroscedasticity", meaning that the variances within the two groups being compared may be different. While the goal of the test is to compare the means, the variances play an important role in calibrating the statistics (deciding how big the mean difference needs to be to be declared statistically significant). In the NHANES data, we see that there are moderate differences between the amount of variation in BMI for females and for males, looking within 10-year age bands. In every age band, females having greater variation than males.

[10]:
```
dx = df[['BMXBMI', 'RIDAGEYR', 'RIAGENDRx']].dropna()
df['agegrp'] = pd.cut(df.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])
df.groupby(['agegrp', 'RIAGENDRx'])['BMXBMI'].agg(np.std).unstack()
```

[10]:

RIAGENDRx	Female	Male
agegrp		
(18, 30]	7.745893	6.649440
(30, 40]	8.315608	6.622412
(40, 50]	8.076195	6.407076
(50, 60]	7.575848	5.914373
(60, 70]	7.604514	5.933307
(70, 80]	6.284968	4.974855

The standard error of the mean difference (e.g. mean female blood pressure minus mean male blood pressure) can be estimated in at least two different ways. In the statsmodels library, these approaches are referred to as the "pooled" and the "unequal" approach to estimating the variance. If the variances are equal (i.e. there is no heteroscedasticity), then there should be little difference between the two approaches. Even in the presence of moderate heteroscedasticity, as we have here, we can see that the results for the two methods are quite similar. Below we have a loop that considers each 10-year age band and assesses the evidence for a difference in mean BMI for women and for men. The results printed in each row of output are the test-statistic and p-value.

[11]:
```
for k, v in df.groupby('agegrp'):
    bmi_female = v.loc[v.RIAGENDRx == 'Female', 'BMXBMI'].dropna()
```

```
    bmi_female = sm.stats.DescrStatsW(bmi_female)

    bmi_male = v.loc[v.RIAGENDRx == 'Male', 'BMXBMI'].dropna()
    bmi_male = sm.stats.DescrStatsW(bmi_male)

    print(k)
    print('pooled: ', sm.stats.CompareMeans(bmi_female, bmi_male).
↪ztest_ind(usevar='pooled'))
    print('unequal: ', sm.stats.CompareMeans(bmi_female, bmi_male).
↪ztest_ind(usevar='unequal'))
    print()
```

```
(18, 30]
pooled:  (1.7026932933643388, 0.08862548061449649)
unequal:  (1.7174610823927268, 0.08589495934713022)

(30, 40]
pooled:  (1.4378280405644916, 0.1504828511464818)
unequal:  (1.4437869620833494, 0.14879891057892475)

(40, 50]
pooled:  (2.8933761158070186, 0.003811246059501354)
unequal:  (2.9678691663536725, 0.0029987194174035366)

(50, 60]
pooled:  (3.362108779981367, 0.0007734964571391746)
unequal:  (3.375494390173923, 0.0007368319423226574)

(60, 70]
pooled:  (3.6172401442432753, 0.00029776102103193 6)
unequal:  (3.62848309454456, 0.0002850914147149227)

(70, 80]
pooled:  (2.926729252512258, 0.0034254694144858636)
unequal:  (2.937779886769224, 0.003305716331519299)
```

Paired tests A common situation in applied research is to measure the same quantity multiple times on each unit of analysis. For example, in NHANES, systolic blood pressure is measured at least two times (sometimes there is a third measurement) on each subject. Although the measurements are repeated, there is no guarantee that the mean is the same each time, i.e. the mean blood pressure may be slightly lower on the second measurement compared to the first, since people are a bit more nervous the first time they are measured. A paired test is a modified form of mean test that can be used when we are comparing two repeated measurements on the same unit.

A paired t-test for means is equivalent to taking the difference between the first and second measurement, and using a one-sample test to compare the mean of these differences to zero. Below we see that in the entire NHANES sample, the first measurement of systolic blood pressure is on

average 0.67 mm/Hg greater than the second measurement. While this difference is not large, it is strongly statistically significant. That is, there is strong evidence that the mean values for the first and second blood pressure measurement differ.

[12]:
```
dx = df[['BPXSY1', 'BPXSY2']].dropna()
db = dx.BPXSY1 - dx.BPXSY2

sm.stats.ztest(db)
```

[12]:
```
(9.800634425497911, 1.1188070930963587e-22)
```

To probe this effect further, we can divide the population into 10 year wide age bands and also stratify by gender, then carry out the paired t-test within each of the resulting 12 strata. We see that the second systolic blood pressure measurement is always lower on average than the first. The difference is larger for older people and for males. The difference is statistically significant for females over 30, and for males over 60.

Conducting many hypothesis tests and "cherry picking" the interesting results is usually a bad practice. Here we are doing such "multiple testing" for illustration, and acknowledge that the strongest differences may be over-stated. Nevertheless, there is a clear and consistent trend with age – older people tend to have greater differences between their first and second blood pressure measurements than younger people. There is also a difference between the genders, with older men having a stronger difference between the first and second blood pressure measurements than older women. The gender difference for younger peple is less clear.

[13]:
```
dx = df[['RIAGENDRx', 'BPXSY1', 'BPXSY2', 'RIDAGEYR']].dropna()
dx['agegrp'] = pd.cut(dx.RIDAGEYR, [18, 30, 40, 50, 60, 70, 80])

for k, g in dx.groupby(['RIAGENDRx', 'agegrp']):
    db = g.BPXSY1 - g.BPXSY2
    print(k, db.mean(), db.size, sm.stats.ztest(db.values, value=0))
```

```
('Female', Interval(18, 30, closed='right')) 0.13708260105448156 569
(0.7612107360791227, 0.4465312067051751)
('Female', Interval(30, 40, closed='right')) 0.6713615023474179 426
(3.307398751951031, 0.0009416674523368051)
('Female', Interval(40, 50, closed='right')) 0.5970149253731343 469
(2.6040611621024654, 0.009212631487347644)
('Female', Interval(50, 60, closed='right')) 0.7685393258426966 445
(3.1023718750881724, 0.001919766301204196)
('Female', Interval(60, 70, closed='right')) 0.8787878787878788 396
(3.1024528501809625, 0.0019192411825181255)
('Female', Interval(70, 80, closed='right')) 1.4512820512820512 390
(5.141706875154317, 2.722536503552981e-07)
('Male', Interval(18, 30, closed='right')) 0.00390625 512 (0.01959622841647691,
0.9843654725443948)
('Male', Interval(30, 40, closed='right')) 0.46296296296296297 432
(1.9451535788714596, 0.05175649697939119)
('Male', Interval(40, 50, closed='right')) 0.17894736842105263 380
```

```
(0.7201800810138878, 0.47141412641258706)
('Male', Interval(50, 60, closed='right')) 0.3691588785046729 428
(1.4391115097646396, 0.1501189315054144)
('Male', Interval(60, 70, closed='right')) 1.2736077481840193 413
(4.781940964515296, 1.7361067031915549e-06)
('Male', Interval(70, 80, closed='right')) 2.031413612565445 382
(6.8013414549535005, 1.036494265013724e-11)
```

Research Question:

Considering elderly Hispanic adults (80+) living in the U.S. in 2015-2016, did the proportions of males and females who smoked vary significantly?

- Males: proportion=0.565, n=16
- Females: proportion=0.250, n=32
- Approach 1 - Chi-square Test
  - Are all expected counts for each cell of the 2x2 table under the null hypothesis greater than 5? YES!

	Smoker	Non-smoker
Female	8	24
Male	7	9

- Approach 2 - Fisher's Exact Z Test
  - Especially for the smaller ones

[14]:
```
# chi-square test

# 9 males out of 16 are smokers
# 8 females out of 32 are smokers

result = sm.stats.proportions_chisquare([9,8],[16,32])
print('X2 statistic:', result[0])
print('P-value:', result[1])
```

```
X2 statistic: 4.554079696394687
P-value: 0.032840408476191844
```

[15]:
```
# fisher's exact test

# 7 male smokers
# 9 male non-smokers
# 8 female smokers
# 24 female non-smokers

odds_ratio, p_value = stats.fisher_exact([[7, 9], [24, 8]])
```

```
print('P-value:', p_value)
```

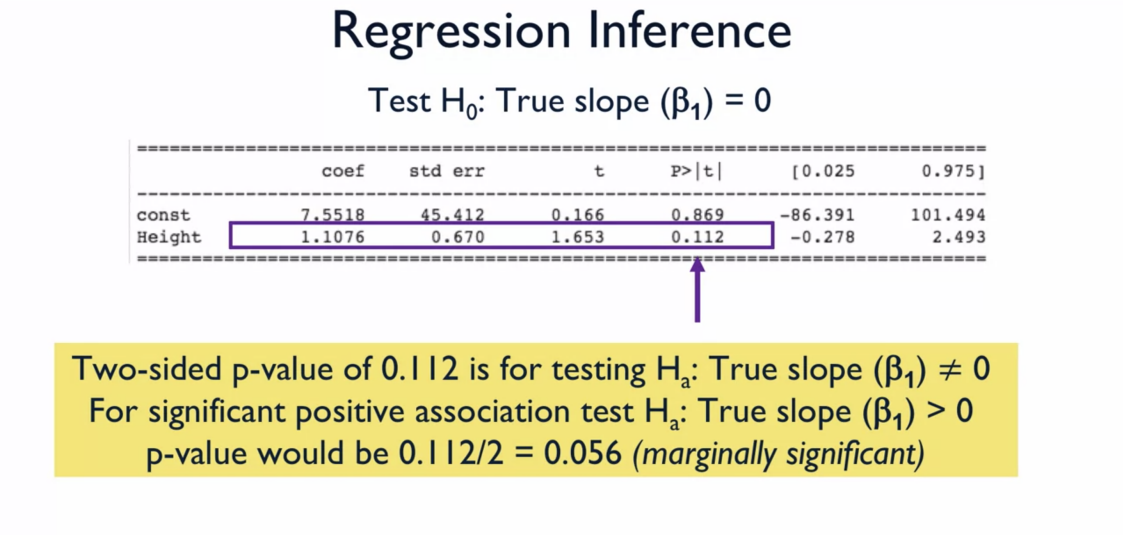
```
P-value: 0.05411770597952523
```

[1]:
```
from IPython.display import Image
```

Linear Regression

[2]:
```
Image('images/example_6.png', width=600)
```

[2]:

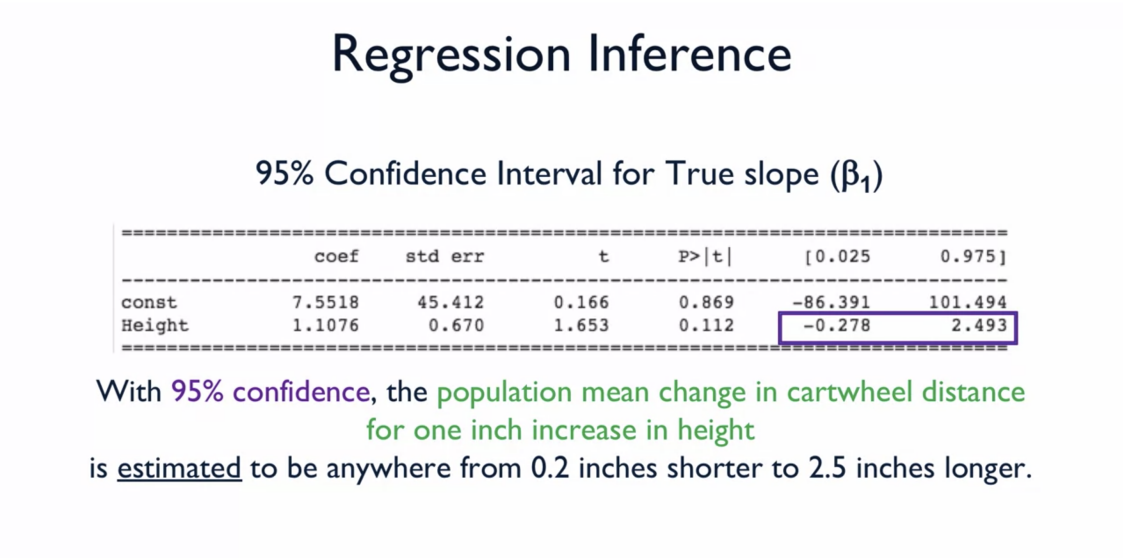


Next to our coefficients where we pulled off the values to give us our equation of our line, we've got some information that does test hypotheses about those underlying true intercept and slope. Looking at the row with our label of height, we have our coefficient, our estimated slope, our B1 of 1.1. Now, that's not equal to zero, but we're interested in seeing how far away from zero is it to show it there's evidence to conclude that the true slope might not be zero. Well, one piece of information we need is that standard error. So, here, this 0.67 is estimating for us how far away estimated slopes, like the one we got here, are going to be from the true slope on average. Taking those two pieces of information, we form our T statistic. It is measuring how close is our estimated slope, 1.1 from zero in standard error units. Our estimated slope was 1.65 standard errors above zero. So, that's starting to be a pretty good distance from zero. Converting that to a probability value, our p-value is given next, 0.112, which is not that small, certainly not significantly at even at 10 percent level, but this is the p-value for a two-sided alternative. So, if you wanted to assess whether the true slope is zero or not zero, we would report this level for our p-value. Our initial research question was a significant positive relationship between our two variables. So, our alternative theory would be looking for the true slope being greater than zero because that was the direction that made sense. So, our p-value would not be the two tails together for probabilities, but just the one tail. So, we need to take our two-sided p-value and cut that in half. Our p-value for assessing a significant positive linear relationship between cartwheel distance and height turns out

to be 0.56. Significant at a 10 percent level, but not at a five percent level, marginally significant.

[3]:
```
Image('images/example_7.png', width=600)
```

[3]:



We might also be interested in reporting a range of values for which we might say is reasonable for this true slope, a confidence interval. That is reported a little further down at 95 percent level for us. So, with 95 percent confidence, the population mean change in cartwheel distance for a one-inch increase in height would be estimated to be anywhere from 0.2 inches shorter, but up to as high as 2.5 inches longer.

[4]:
```
Image('images/example_8.png', width=600)
```

[4]:

In 1905, R.J. Gladstone conducted a study of the relationship between brain weight and size of the head. Brain weight (grams) and head size (cubic cm) measurements were performed for 237 adults. Two categorical variables for Sex (0=male, 1=female) and Age (0=young, 20-46 years old, 1=old, 46+ years old) are available. The linear regression results for regressing brain weight on the head size are summarized below.

	mean	sd	se(mean)	n
Brain	1282.873	120.34	7.82	237
Head	3633.992	365.26	23.73	237

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	325.573	47.141	6.906	4.61e-11
Head	0.26343	0.0129	20.409	< 2e-16

One subject in the study has a head size of 3500 cm3 and a brain weight of 1430.86 grams. What is the value of the observed error (residual) for this subject?

You see a quiz question above. When looking at the coefficients, the estimated coefficient of **head** feature is **0.26343** and also, estimated **intercept** value is **325.573**. So, we can calculate the estimated **brain weight** as follows,

```
brain_weight = 0.26343 * head + 325.573
```

[5]:
```
actual_weight = 1430.86
estimated_weight = 0.26343 * 3500 + 325.573

observed_error = abs(estimated_weight - actual_weight)
observed_error
```

[5]:
```
183.28199999999993
```

And the second question is here,

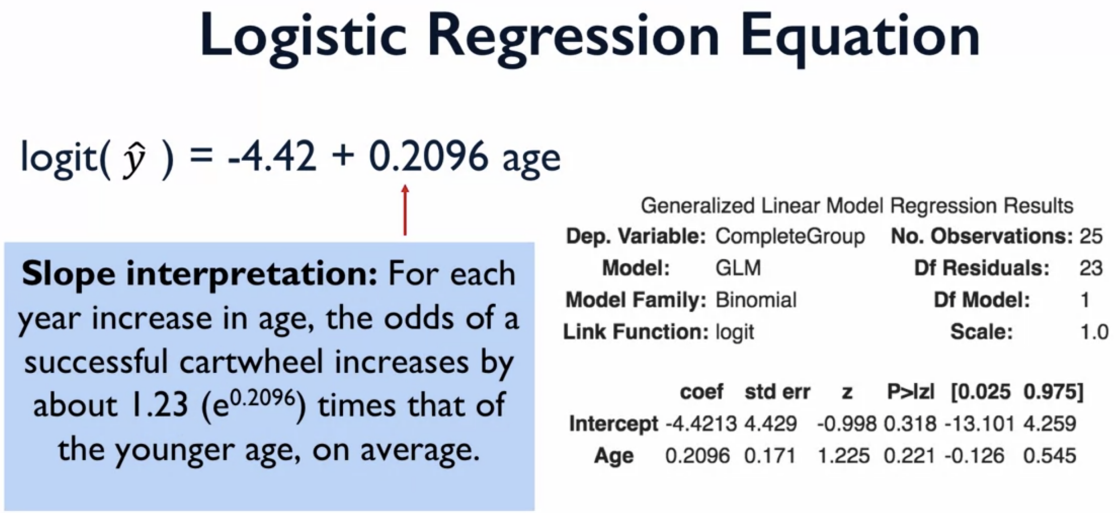
What is the appropriate p-value for testing if there is a significant positive linear relationship between brain weight and head size?

As we mentioned before, p value is calculated as two-tailed in the question above. In fact, our null hypothesis is that the **true slope = 0** and our alternative hypothesis is **slope > 0**, not **slope ≠ 0**. Because we are investigating the positive linear relationship between two variables. So, our appropriate p-value is **<1e-16**

Logistic Regression

[6]:
```
Image('images/example_9.png', width=600)
```

[6]:



Please notice that, in this example, our primary variable of interest is whether or not they completed a cartwheel. Either yes, they successfully completed a cartwheel, which is a one. Or no, they didn't successfully complete the cartwheel, was just coded as a zero. We want to know if based on age, we can predict whether cartwheel is completed or not.

For the results, we can interpret this a little bit differently. So, for each year increase in age, the odds of a successful cartwheel increases by about 1.23 times that of the younger age on average. That 1.23 is calculated by doing e to the slope to that 0.2096. What this means is that for each year increase, the odds of successfully completing a cartwheel increases by a multiplicative factor, so you're multiplying it to the previous odds. In essence what it means, when the odds are greater than one or when the odds increases greater than one is that each year you're more likely to successfully complete a cartwheel. If it were less than one, you would be less likely to successfully complete a cartwheel.

Data Modeling in Python As discussed in week one of this course, we will be investigating how to develop various statistical models around data.

Modeling plays a significant role in data analysis and builds upon fundamental concepts of statistics. By fitting models to data we are able to accomplish the following:

- **Estimate** distributional properties of variables, potentially conditional on other variables.
- Concisely **summarize relationship** between variables, and make inferential statements about those relationships.
- **Predict** values of variables of interest conditional on values of other predictor varialbes, and characterize prediction uncertainty.

With these concepts in mind, we will overview modeling structure and carrying out exploratory data analysis on a dataset that contains information about homes in Boston, MA and how we may want to approach modeling prices of homes.

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
from sklearn.datasets import load_boston
from statsmodels.graphics.regressionplots import add_lowess, plot_ccpr
from statsmodels.sandbox.predict_functional import predict_functional
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
```

[3]:
```
%config InlineBackend.figure_format = 'retina'
```

[4]:
```
boston_dataset = load_boston()
boston = pd.DataFrame(
    boston_dataset.data,
    columns=boston_dataset.feature_names
)

boston['MEDV'] = boston_dataset.target
```

Now that we've seen our various columns, lets take a look at what each column represents:

- **CRIM:** Per capita crime rate by town
- :**ZN::** Proportion of residential land zoned for lots over 25,000 sq. ft
- **INDUS:** Proportion of non-retail business acres per town
- **CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **NOX:** Nitric oxide concentration (parts per 10 million)
- **RM:** Average number of rooms per dwelling
- **AGE:** Proportion of owner-occupied units built prior to 1940
- **DIS:** Weighted distances to five Boston employment centers
- **RAD:** Index of accessibility to radial highways
- **TAX:** Full-value property tax rate per $\$10,000$
- **PTRATIO:** Pupil-teacher ratio by town
- **B** $1000(Bk—0.63)^2$, where Bk is the proportion of [people of African American descent] by town
- **LSTAT:** Percentage of lower status of the population
- **MEDV:** Median value of owner-occupied homes in $1000s

[5]:
```
boston.head()
```

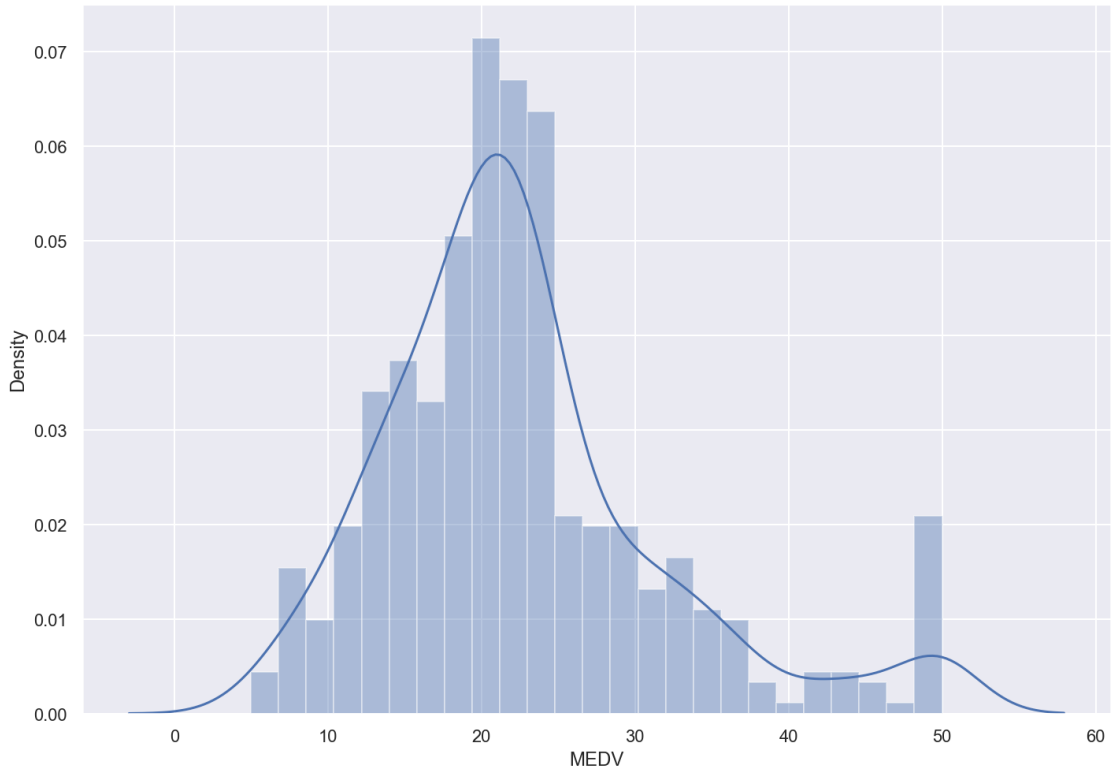
[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

```
   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

EDA

[6]:
```
sns.set(rc={'figure.figsize': (11.7, 8.27)})
sns.distplot(boston['MEDV'], bins=25)
plt.show()
```



As you can see, our target variable **MEDV** follows a normal distribution. However, there are outliers near the upper quantiles.

One of the most important steps in modeling is understanding the relationship our variables have with each other. A relationship between two variables is also denoted as correlation is statistics. A figure known as a correlation matrix, can be used to measure the linear relationships between variables.

[7]:
```
correlation_matrix = boston.corr().round(2)

sns.set(rc={'figure.figsize':(11.7, 8.27)})
sns.heatmap(data=correlation_matrix, annot=True)
plt.show()
```

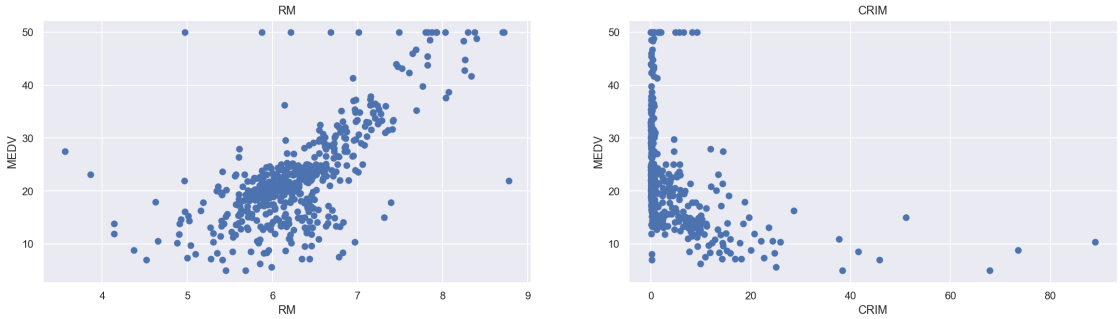
Correlation coefficients range from -1 to 1, which determines the strength of correlation. Values close to 1 signify a strong positive relationship, whereas values close to -1 indicate a strong negative relationship.

With this heatmap, we can see corellation coefficients for each of our potential predictors values and **MEDV**, our target values. Interestingly, our initial gut instincts of **RM** and **CRIM**. **RM's** coefficient is 0.7, and **CRIM's** is -0.39, signifying a postive and negative relationship as suggested.

To further investigate individual individual predictors, we can plot their values against the value of **MEDV**. This with allow us to infer whether or not the relationship is linear, or if further transformations are required.

[8]:
```
plt.figure(figsize=(20, 5))

features = ['RM', 'CRIM']
target = boston['MEDV']
```

```
for i, col in enumerate(features):
    plt.subplot(1, len(features) , i + 1)

    x = boston[col]
    y = target

    plt.scatter(x, y, marker='o')

    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



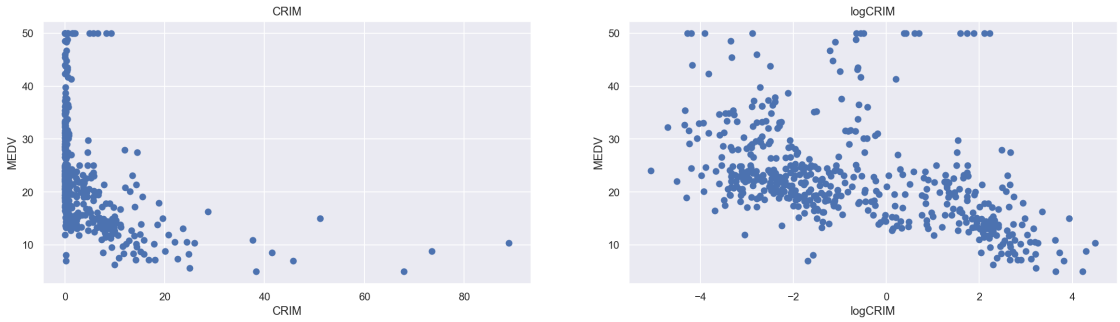
Here we can see that our relationship between **RM** and **MEDV** is very linear which suggests we do not need to do any further manipulation of our data. However, our plot of **CRIM** and **MEDV** is extremely skewed. Notice that there little separation between our **CRIM** values. There is a cluster of values between 0 and 20.

When thinking about linear models, we can do transformations that manipulate our data so that our variables have stronger linear relationships.

One common transformation is the **log transformatio**, which *stretches* our values out. This can be shown below:

[9]:
```
plt.figure(figsize=(20, 5))

boston['logCRIM'] = np.log(boston['CRIM'])

features = ['CRIM', 'logCRIM']
target = boston['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i + 1)

    x = boston[col]
    y = target
```

```
    plt.scatter(x, y, marker='o')

    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



The plots above show a before and after where the log transformation is applied to **CRIM**. On the righthand side, we can see that there is a drastic difference in the separation of our values and the negative linear correlation between **MEDV** and **logCRIM** is more apparent than between **MEDV** and **CRIM** (without the log transformation).

Now that we have identified some potential predictors for our model and have throughly investigated their relationship with our target variable, we can begin constructing a model, however thats going to be all for this notebook. These topics will be discussed in more detail in the coming weeks and you will have the opportunity to read case studies and practice developing models on your own in python!

If you have some extra time this week and would like to practice, feel free to use this notebook as a template for carrying out your own exploratory analysis on a dataset of your choice.

Getting in the mindset of exploring data and visualizing relationships between variables will pay dividends for the rest of the course as we delve further into fitting statistical models to data with python!

Linear and logistic regression modeling, a case study with the NHANES data

[10]:
```
da = pd.read_csv('data/NHANES.csv')

# drop unused columns
# drop rows with any missing values
columns = [
    'BPXSY1',
    'RIDAGEYR',
    'RIAGENDR',
    'RIDRETH1',
    'DMDEDUC2',
    'BMXBMI',
```

```
    'SMQ020'
]
da = da[columns].dropna()
```

Linear regression We will focus initially on regression models in which systolic blood pressure (SBP) is the outcome (dependent) variable. That is, we will predict SBP from other variables. SBP is an important indicator of cardiovascular health. It tends to increase with age, is greater for overweight people (i.e. people with greater body mass index or BMI), and also differs among demographic groups, for example among gender and ethnic groups.

Since SBP is a quantitative variable, we will model it using linear regression. Linear regression is the most widely-utilized form of statistical regression. While linear regression is commonly used with quantitative outcome variables, it is not the only type of regression model that can be used with quantitative outcomes, nor is it the case that linear regression can only be used with quantitative outcomes. However, linear regression is a good default starting point for any regression analysis using a quantitative outcome variable.

Interpreting regression parameters in a basic model We start with a simple linear regression model with only one covariate, age, predicting SBP. In the NHANES data, the variable BPXSY1 contains the first recorded measurement of SBP for a subject, and RIDAGEYR is the subject's age in years. The model that is fit in the next cell expresses the expected SBP as a linear function of age:

[11]:
```
model = sm.OLS.from_formula("BPXSY1 ~ RIDAGEYR", data=da)
result = model.fit()
result.summary()
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  BPXSY1   R-squared:                       0.207
Model:                             OLS   Adj. R-squared:                  0.207
Method:                  Least Squares   F-statistic:                     1333.
Date:                 Fri, 01 Jan 2021   Prob (F-statistic):          2.09e-259
Time:                         18:25:38   Log-Likelihood:                -21530.
No. Observations:                5102   AIC:                          4.306e+04
Df Residuals:                    5100   BIC:                          4.308e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    102.0935      0.685    149.120      0.000     100.751     103.436
RIDAGEYR       0.4759      0.013     36.504      0.000       0.450       0.501
==============================================================================
Omnibus:                      690.261   Durbin-Watson:                   2.039
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1505.999
Skew:                           0.810   Prob(JB):                         0.00
Kurtosis:                       5.112   Cond. No.                          156.
==============================================================================

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

Much of the output above is not relevant for us, so focus on the center section of the output where the header begins with **coef**. This section contains the estimated values of the parameters of the regression model, their standard errors, and other values that are used to quantify the uncertainty in the regression parameter estimates. Note that the parameters of a regression model, which appear in the column labeled **coef** in the table above, may also be referred to as *slopes* or *effects*.

This fitted model implies that when comparing two people whose ages differ by one year, the older person will on average have 0.48 units higher SBP than the younger person. This difference is statistically significant, based on the p-value shown under the column labeled `P>|t|`. This means that there is strong evidence that there is a real association between between systolic blood pressure and age in this population. SBP is measured in units of *millimeters of mercury*, expressed *mm/Hg*. In order to better understand the meaning of the estimated regression parameter 0.48, we can look at the standard deviation of SBP:

[12]:
```
da.BPXSY1.std()
```

[12]:
```
18.486559500782416
```

The standard deviation of around 18.5 is large compared to the regression slope of 0.48. However the regression slope corresponds to the average change in SBP for a single year of age, and this effect accumulates with age. Comparing a 40 year-old person to a 60 year-old person, there is a 20 year difference in age, which translates into a `20 * 0.48 = 9.6` unit difference in average SBP between these two people. This difference is around half of one standard deviation, and would generally be considered to be an important and meaningful shift.

R-squared and correlation In the case of regression with a single independent variable, as we have here, there is a very close correspondence between the regression analysis and a Pearson correlation analysis, which we have discussed earlier in course 2. The primary summary statistic for assessing the strength of a predictive relationship in a regression model is the *R-squared*, which is shown to be 0.207 in the regression output above. This means that 21% of the variation in SBP is explained by age. Note that this value is exactly the same as the squared Pearson correlation coefficient between SBP and age, as shown below.

[13]:
```
cc = da[['BPXSY1', 'RIDAGEYR']].corr()
cc.BPXSY1.RIDAGEYR ** 2
```

[13]:
```
0.20715459625188243
```

There is a second way to interpret the R-squared, which makes use of the *fitted values* of the

regression. The fitted values are predictions of the blood pressure for each person in the data set, based on their covariate values. In this case, the only covariate is age, so we are predicting each NHANES subject's blood pressure as a function of their age. If we calculate the Pearson correlation coefficient between the fitted values from the regression, and the actual SBP values, and then square this correlation coefficient, we see that we also get the R-squared from the regression:

[14]:
```
cc = np.corrcoef(da.BPXSY1, result.fittedvalues)
cc[0, 1] ** 2
```

[14]:
```
0.20715459625186952
```

Thus, we see that in a linear model fit with only one covariate, the regression R-squared is equal to the squared Pearson correlation between the covariate and the outcome, and is also equal to the squared Pearson correlation between the fitted values and the outcome.

Adding a second variable Above we considered a simple linear regression analysis with only one covariate (age) predicting systolic blood pressure (SBP). The real power of regression analysis arises when we have more than one covariate predicting an outcome. As noted above, SBP is expected to be related to gender as well as to age, so we next add gender to the model. The NHANES variable for gender is named RIAGENDR

We begin by creating a relabeled version of the gender variable:

[15]:
```
# create a labeled version of the gender variable
da['RIAGENDRx'] = da.RIAGENDR.replace({1: 'Male', 2: 'Female'})
```

[16]:
```
model = sm.OLS.from_formula('BPXSY1 ~ RIDAGEYR + RIAGENDRx', data=da)
result = model.fit()
result.summary()
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 BPXSY1   R-squared:                       0.215
Model:                            OLS   Adj. R-squared:                  0.214
Method:                 Least Squares   F-statistic:                     697.4
Date:                Fri, 01 Jan 2021   Prob (F-statistic):          1.87e-268
Time:                        18:25:38   Log-Likelihood:                -21505.
No. Observations:                5102   AIC:                         4.302e+04
Df Residuals:                    5099   BIC:                         4.304e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
=====
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
```

```
-----
Intercept           100.6305      0.712    141.257      0.000      99.234
102.027
RIAGENDRx[T.Male]     3.2322      0.459      7.040      0.000       2.332
4.132
RIDAGEYR              0.4739      0.013     36.518      0.000       0.448
0.499
==============================================================================
Omnibus:                      706.732   Durbin-Watson:                   2.036
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1582.730
Skew:                           0.818   Prob(JB):                         0.00
Kurtosis:                       5.184   Cond. No.                         168.
==============================================================================

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

The syntax `RIDAGEYR + RIAGENDRx` in the cell above does not mean that these two variables are literally added together. Instead, it means that these variables are both included in the model as predictors of blood pressure (`BPXSY1`).

The model that was fit above uses both age and gender to explain the variation in SBP. It finds that two people with the same gender whose ages differ by one year tend to have blood pressure values differing by 0.47 units, which is essentially the same age parameter that we found above in the model based on age alone. This model also shows us that comparing a man and a woman of the same age, the man will on average have 3.23 units greater SBP.

It is very important to emphasize that the age coefficient of 0.47 is only meaningful when comparing two people of the same gender, and the gender coefficient of 3.23 is only meaningful when comparing two people of the same age. Moreover, these effects are additive, meaning that if we compare, say, a 50 year old man to a 40 year old woman, the man's blood pressure will on average be around 3.23 + 10*0.47 = 7.93 units higher, with the first term in this sum being attributable to gender, and the second term being attributable to age.

We noted above that the regression coefficient for age did not change by much when we added gender to the model. It is important to note however that in general, the estimated coefficient of a variable in a regression model will change when other variables are added or removed. The only circumstance in which a regresion parameters is unchanged when other variables are added or removed from the model is when those variables are uncorrelated with the variables that remain in the model.

Below we confirm that gender and age are nearly uncorrelated in this data set (the correlation of around -0.02 is negligible). Thus, it is expected that when we add gender to the model, the age coefficient is unaffected.

[17]:
```
# we need to use the original, numerical version of the gender
# variable to calculate the correlation coefficient
```

```
da[['RIDAGEYR', 'RIAGENDR']].corr()
```

[17]:
```
           RIDAGEYR  RIAGENDR
RIDAGEYR   1.000000 -0.021398
RIAGENDR -0.021398  1.000000
```

Observe that in the regression output shown above, an R-squared value of 0.215 is listed. Earlier we saw that for a model with only one covariate, the R-squared from the regression could be defined in two different ways, either as the squared correlation coefficient between the covariate and the outcome, or as the squared correlation coefficient between the fitted values and the outcome. When more than one covariate is in the model, only the second of these two definitions continues to hold:

[18]:
```
cc = np.corrcoef(da.BPXSY1, result.fittedvalues)
cc[0, 1] ** 2
```

[18]:
```
0.21478581086243784
```

Categorical variables and reference levels In the model fit above, gender is a categorical variable, and only a coefficient for males is included in the regression output (i.e. there is no coefficient for females in the tables above). Whenever a categorical variable is used as a covariate in a regression model, one level of the variable is omitted and is automatically given a coefficient of zero. This level is called the *reference level* of the covariate. Here, the female level of the gender variable is the reference level. This does not mean that being a woman has no impact on blood pressure. It simply means that we have written the model so that female blood pressure is the default, and the coefficient for males (3.23) shifts the blood pressure by that amount for males only.

We could alternatively have set 'male' to be the reference level, in which case males would be the default, and the female coefficient would have been around -3.23 (meaning that female blood pressure is 3.23 units lower than the male blood pressure).

When using a categorical variable as a predictor in a regression model, it is recoded into "dummy variables" (also known as "indicator variables"). A dummy variable for a single level, say `a`, of a variable `x`, is a variable that is equal to `1` when `x=a` and is equal to `0` when `x` is not equal to `a`. These dummy variables are all included in the regression model, to represent the variable that they are derived from.

Statsmodels, like most software, will automatically recode a categorical variable into dummy variables, and will select a reference level (it is possible to override this choice, but we do not cover that here). When interpreting the regression output, the level that is omitted should be seen as having a coefficient of 0, with a standard error of 0. It is important to note that the selection of a reference level is arbitrary and does not imply an assumption or constraint about the model, or about the population that it is intended to capture.

A model with three variables Next we add a third variable, body mass index (BMI), to the model predicting SBP. BMI is a measure that is used to assess if a person has healthy weight given their height. BMXBMI is the NHANES variable containing the BMI value for each subject.

[19]:
```
model = sm.OLS.from_formula('BPXSY1 ~ RIDAGEYR + BMXBMI + RIAGENDRx', data=da)
result = model.fit()
result.summary()
```

[19]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  BPXSY1   R-squared:                       0.228
Model:                             OLS   Adj. R-squared:                  0.228
Method:                  Least Squares   F-statistic:                     502.0
Date:                 Fri, 01 Jan 2021   Prob (F-statistic):          8.54e-286
Time:                         18:25:38   Log-Likelihood:                -21461.
No. Observations:                5102   AIC:                          4.293e+04
Df Residuals:                    5098   BIC:                          4.296e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     91.5840      1.198     76.456      0.000      89.236      93.932
RIAGENDRx[T.Male]     3.5783      0.457      7.833      0.000       2.683       4.474
RIDAGEYR       0.4709      0.013     36.582      0.000       0.446       0.496
BMXBMI         0.3060      0.033      9.351      0.000       0.242       0.370
==============================================================================
Omnibus:                      752.325   Durbin-Watson:                   2.040
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1776.087
Skew:                           0.847   Prob(JB):                         0.00
Kurtosis:                       5.343   Cond. No.                         316.
==============================================================================

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

Not surprisingly, BMI is positively associated with SBP. Given two subjects with the same gender and age, and whose BMI differs by 1 unit, the person with greater BMI will have, on average, 0.31 units greater systolic blood pressure (SBP). Also note that after adding BMI to the model, the coefficient for gender became somewhat greater. This is due to the fact that the three covariates

in the model, age, gender, and BMI, are mutually correlated, as shown next:

[20]:
```
da[['RIDAGEYR', 'RIAGENDR', 'BMXBMI']].corr()
```

[20]:

	RIDAGEYR	RIAGENDR	BMXBMI
RIDAGEYR	1.000000	-0.021398	0.023089
RIAGENDR	-0.021398	1.000000	0.080463
BMXBMI	0.023089	0.080463	1.000000

Although the correlations among these three variables are not strong, they are sufficient to induce fairly substantial differences in the regression coefficients (e.g. the gender coefficient changes from 3.23 to 3.58). In this example, the gender effect becomes larger after we control for BMI - we can take this to mean that BMI was masking part of the association between gender and blood pressure. In other settings, including additional covariates can reduce the association between a covariate and an outcome.

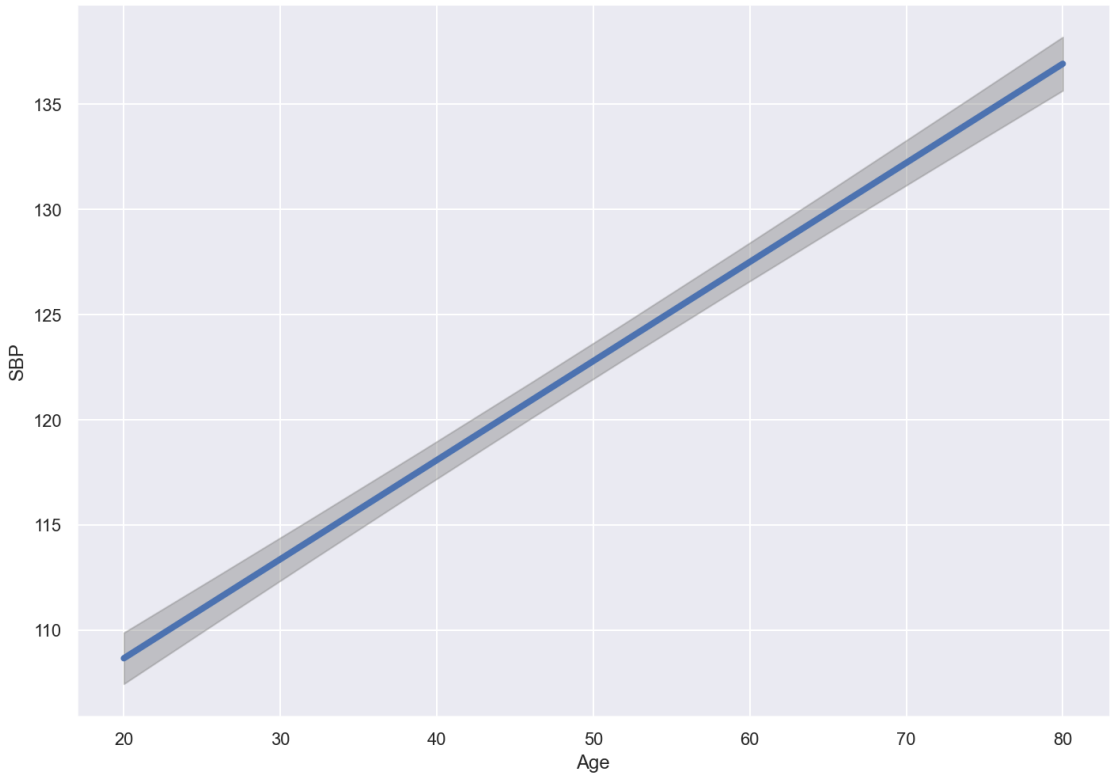
Visualization of the fitted models In this section we demonstrate some graphing techniques that can be used to gain a better understanding of a regression model that has been fit to data.

We start with plots that allow us to visualize the fitted regression function, that is, the mean systolic blood pressure expressed as a function of the covariates. These plots help to show the estimated role of one variable when the other variables are held fixed. We will also plot 95% *simultaneous confidence bands* around these fitted lines. Although the estimated mean curve is never exact based on a finite sample of data, we can be 95% confident that the true mean curve falls somewhere within the shaded regions of the plots below.

This type of plot requires us to fix the values of all variables other than the independent variable (SBP here), and one independent variable that we call the *focus variable* (which is age here). Below we fix the gender as "female" and the BMI as 25. Thus, the graphs below show the relationship between expected SBP and age for women with BMI equal to 25.

[21]:
```
# fix certain variables at reference values. Not all of these
# variables are used here, but we provide them with a value anyway
# to prevent a warning message from appearing
values = {
    'RIAGENDRx': 'Female',
    'RIAGENDR': 1,
    'BMXBMI': 25,
    'DMDEDUC2': 1,
    'RIDRETH1': 1,
    'SMQ020': 1
}

# the returned values are the predicted values (pr), the confidence bands (cb),
# and the function values (fv)
pr, cb, fv = predict_functional(
    result,
    'RIDAGEYR',
    values=values,
```

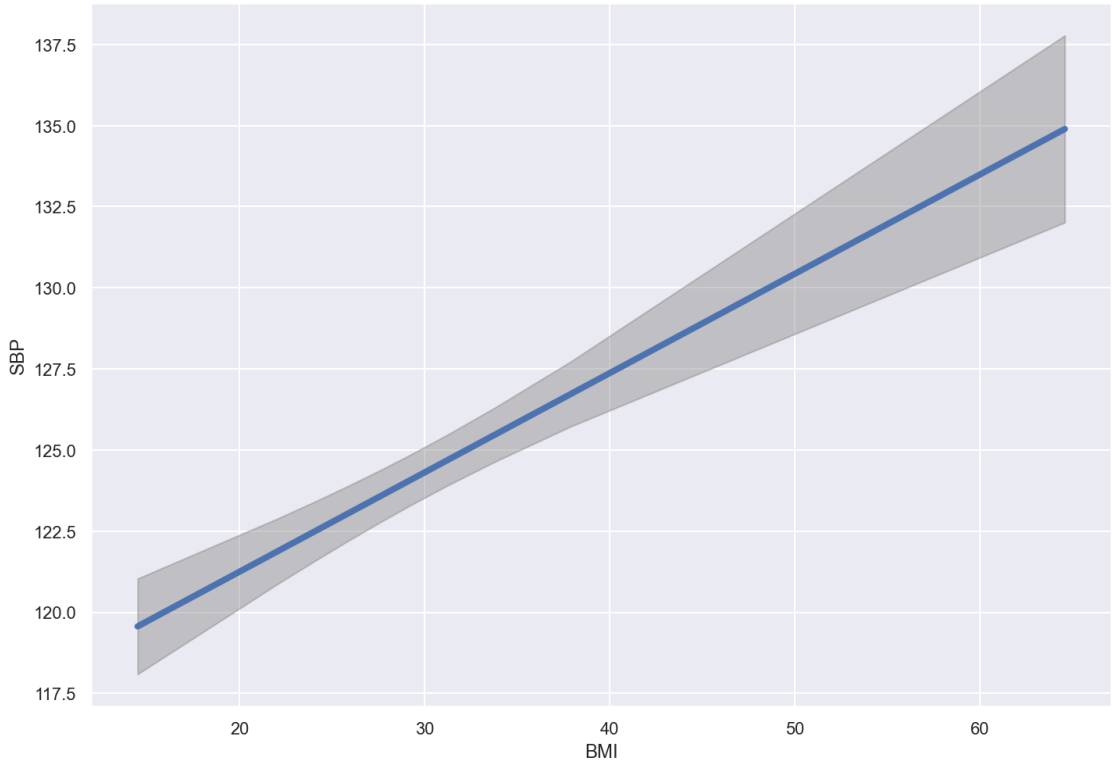
```
    ci_method='simultaneous'
)

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('SBP')
```



The analogous plot for BMI is shown next. Here we fix the gender as "female" and the age at 50, so we are looking at the relationship between expected SBP and age for women of age 50.

[22]:
```
# delete this as it is now the focus variable
values.pop('BMXBMI')
values['RIDAGEYR'] = 50
pr, cb, fv = predict_functional(
    result,
    'BMXBMI',
    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('BMI')
_ = ax.set_ylabel('SBP')
```



The error band for BMI is notably wider than the error band for age, indicating that there is less certainty about the relationship between BMI and SBP compared to the relationship between age and SBP.

The discussion so far has primarily focused on the mean structure of the population, that is, the model for the average SBP of a person with a given age, gender, and BMI. A regression model can also be used to assess the *variance structure* of the population, that is, how much and in what manner the observations deviate from their mean. We will focus on informal, graphical methods for assessing this.

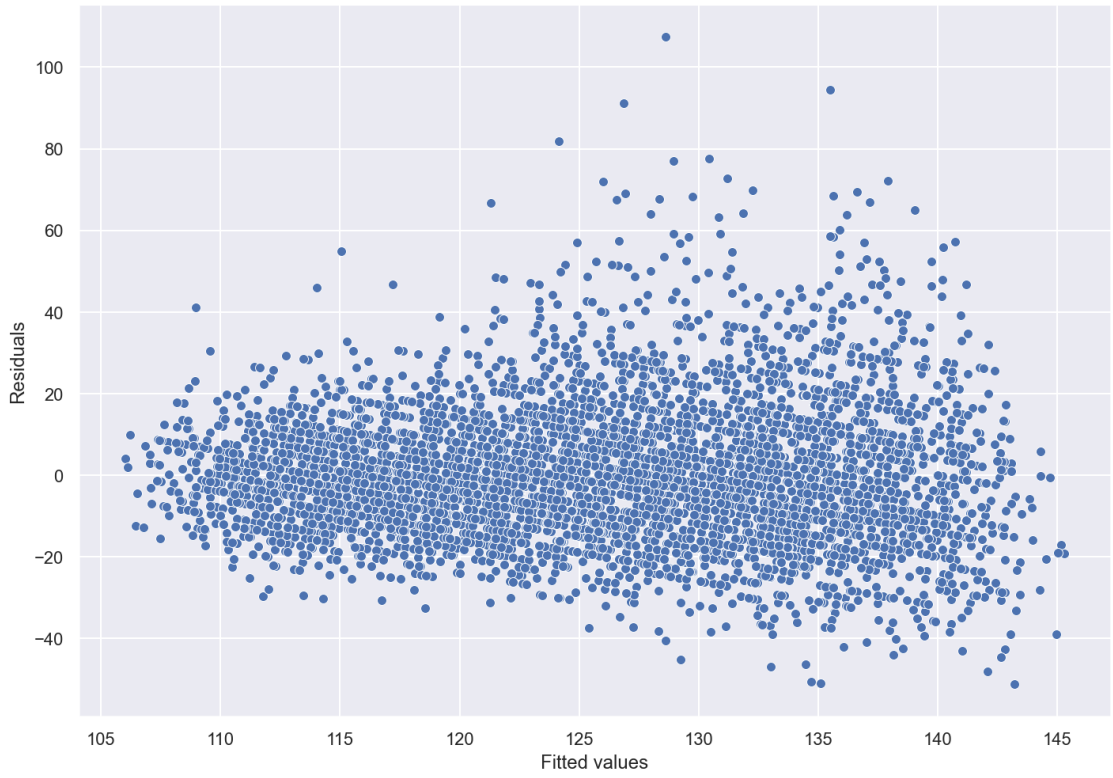
To begin with, we plot the residuals against the fitted values. Recall that the fitted values are the estimated means for each observation, and the residuals are the difference between an observation and its fitted mean. For example, the model may estimate that a 50 year old female will have on average an SBP of 125. But a specific 50 year old female may have a blood pressure of 110 or 150, for example. The fitted values for both of these women are 125, and their residuals are -15, and 25, respectively.

The simplest variance pattern that we can see in a linear regression occurs when the points are scattered around the mean, with the same degree of scatter throughout the range of the covariates. When there are multiple covariates, it is hard to assess whether the variance is uniform throughout this range, but we can easily check for a "mean/variance relationship", in which there is a systematic

relationship between the variance and the mean, i.e. the variance either increases or decreases systematically with the mean. The plot of residuals on fitted values is used to assess whether such a mean/variance relationship is present.

Below we show the plot of residuals on fitted values for the NHANES data. It appears that we have a modestly increasing mean/variance relationship. That is, the scatter around the mean blood pressure is greater when the mean blood pressure itself is greater.

[23]:
```
pp = sns.scatterplot(result.fittedvalues, result.resid)
pp.set_xlabel('Fitted values')
_ = pp.set_ylabel('Residuals')
```

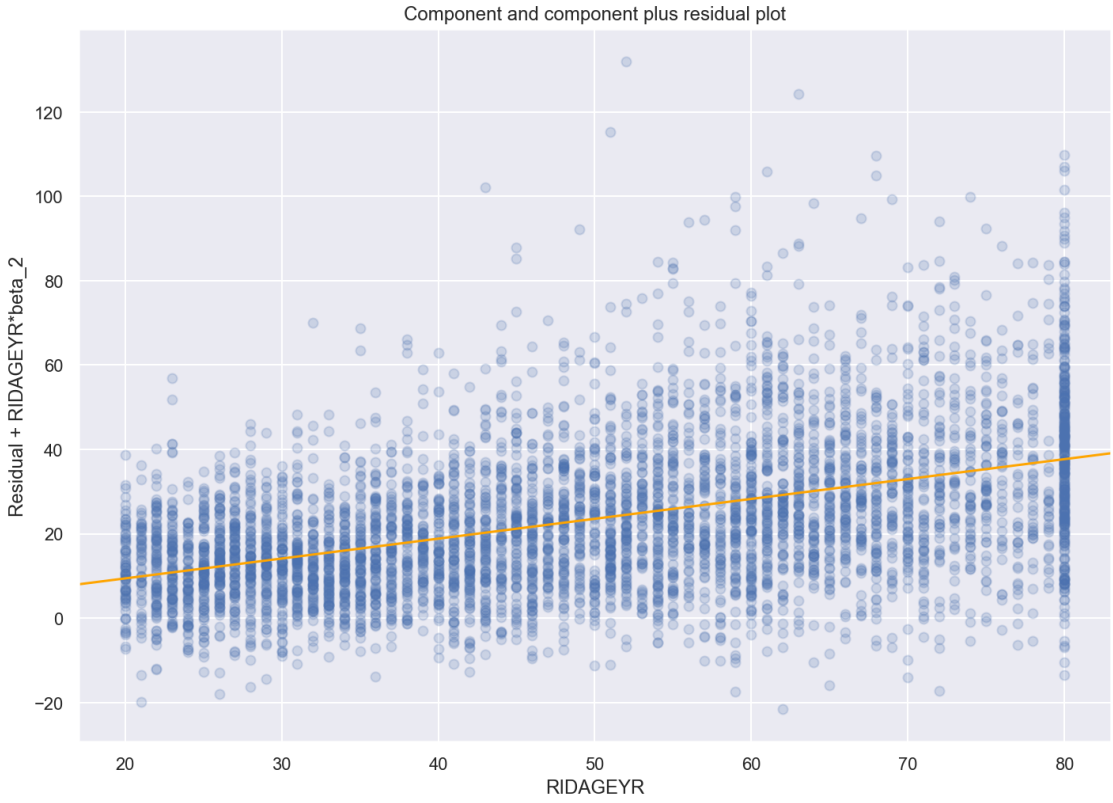


A "component plus residual plot" or "partial residual plot" is intended to show how the data would look if all but one covariate could be fixed at reference values. By controlling the values of these covariates, all remaining variation is due either to the "focus variable" (the one variable that is left unfixed, and is plotted on the horizontal axis), or to sources of variation that are unexplained by any of the covariates.

For example, the partial residual plot below shows how age (horizontal axis) and SBP (vertical axis) would be related if gender and BMI were fixed. Note that the origin of the vertical axis in these plots is not meaningful (we are not implying that anyone's blood pressure would be negative), but the differences along the vertical axis are meaningful. This plot implies that when BMI and gender are held fixed, the average blood pressures of an 80 and 18 year old differ by around 30 mm/Hg. This plot also shows, as discussed above, that the deviations from the mean are somewhat smaller

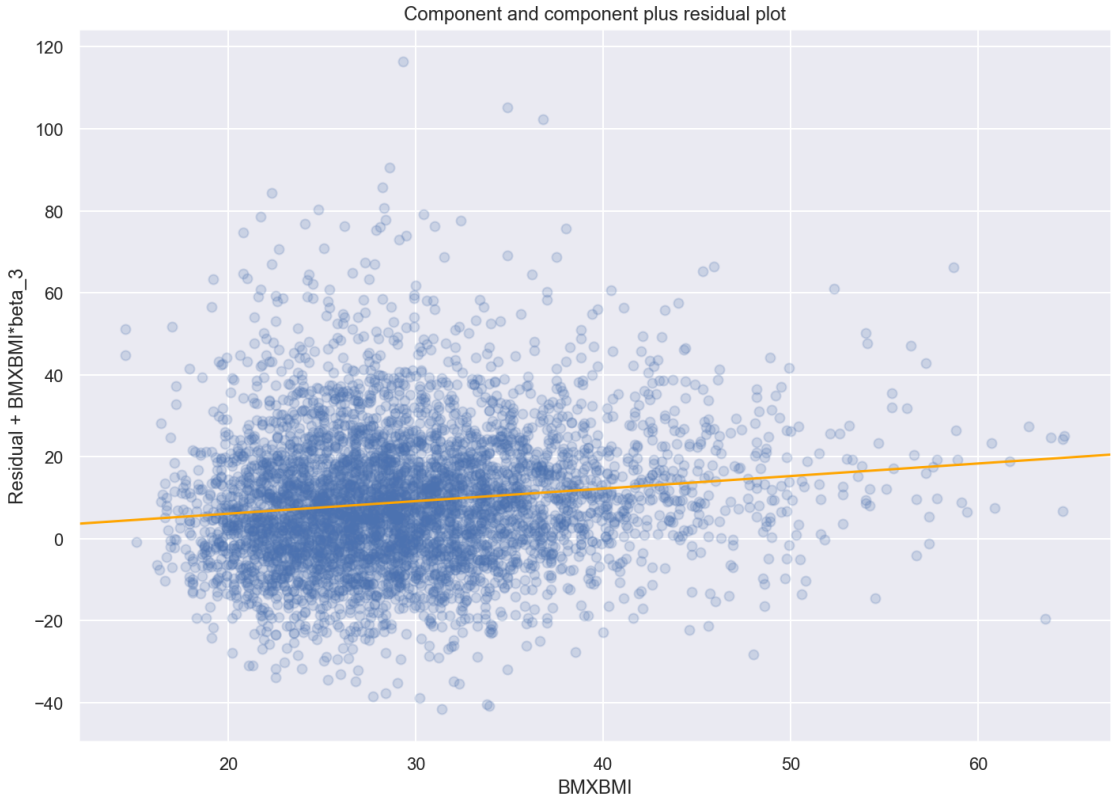
at the low end of the range compared to the high end of the range. We also see that at the high end of the range, the deviations from the mean are somewhat right-skewed, with exceptionally high SBP values being more common than exceptionally low SBP values.

[24]:
```
ax = plt.axes()
plot_ccpr(result, 'RIDAGEYR', ax)

# reduce overplotting with transparency
ax.lines[0].set_alpha(0.2)
_ = ax.lines[1].set_color('orange')
```



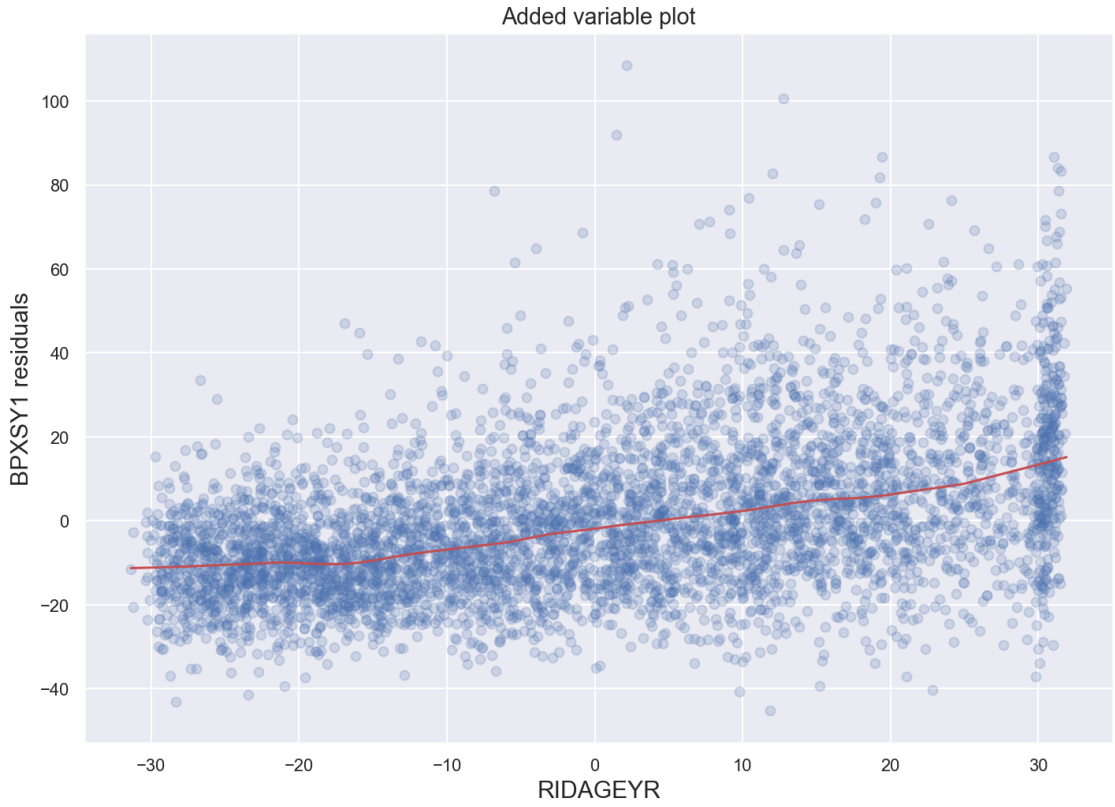
Next we have a partial residual plot that shows how BMI (horizontal axis) and SBP (vertical axis) would be related if gender and age were fixed. Compared to the plot above, we see here that age is more uniformly distributed than BMI. Also, it appears that there is more scatter in the partial residuals for BMI compared to what we saw above for age. Thus there seems to be less information about SBP in BMI, although a trend certainly exists.

[25]:
```
ax = plt.axes()
plot_ccpr(result, 'BMXBMI', ax)
ax.lines[0].set_alpha(0.2)
ax.lines[1].set_color('orange')
```



Another important plot used for understanding a regression model is an "added variable plot". This is a plot that may reveal nonlinearity in the relationship between one covariate and the outcome. Below, we create an added variable plot for age as a predictor of SBP. Note that the two variables being plotted (age and blood pressure) have been centered. The scale of the variables is unchanged, but the origin has been translated to zero. The red line is an estimte of the relationship between age and blood pressure. Unlike the relationship in the model, it is not forced to be linear, and there is in fact a hint that the shape is slightly flatter for the first 15 years or so of age. This would imply that blood pressure increases slightly more slowly for people in theie 20s and early 30s, then begins increasing faster after that point.

[26]:
```
# this is an equivalent way to fit a linear regression model, it needs to be
# done this way to be able to make the added variable plot
model = sm.GLM.from_formula('BPXSY1 ~ RIDAGEYR + BMXBMI + RIAGENDRx', data=da)
result = model.fit()
result.summary()

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



Logistic regression We now turn to regression models for *binary* outcome variables, meaning an outcome that can take on only two distinct values. For illustration, we will work with the NHANES variable 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.

[27]:
```
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.

[28]:
```
c = pd.crosstab(da.RIAGENDRx, da.smq).apply(lambda x: x / x.sum(), axis=1)
c['odds'] = c.loc[:, 1] / c.loc[:, 0]
c
```

[28]:
```
smq             0.0       1.0      odds
RIAGENDRx
```

```
Female     0.680197  0.319803  0.470162
Male       0.467453  0.532547  1.139252
```

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).

[29]:
```
c.odds.Male / c.odds.Female
```

[29]:
```
2.423105552613186
```

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 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.

[30]:
```
c['logodds'] = np.log(c.odds)
c
```

[30]:
```
smq             0.0       1.0      odds   logodds
RIAGENDRx
Female     0.680197  0.319803  0.470162 -0.754679
Male       0.467453  0.532547  1.139252  0.130371
```

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, where `GLM` stands for *Generalized Linear Model*. Logistic regression is one type of GLM, a class which also includes many other regression methods such as Poisson regression that we do not discuss further here. As with linear regression, logistic models also include an intercept parameter, but we are not focusing on that parameter now.

[31]:
```
model = sm.GLM.from_formula('smq ~ RIAGENDRx', family=sm.families.Binomial(),␣
 ↪data=da)
result = model.fit()
result.summary()
```

[31]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    smq   No. Observations:                 5094
Model:                            GLM   Df Residuals:                     5092
Model Family:                Binomial   Df Model:                           1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -3350.6
Date:                Fri, 01 Jan 2021   Deviance:                       6701.2
Time:                        18:25:42   Pearson chi2:                 5.09e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
=====
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----
Intercept           -0.7547      0.042    -18.071      0.000      -0.837
-0.673
RIAGENDRx[T.Male]    0.8851      0.058     15.227      0.000       0.771
0.999
```

```
==============================================================================
=====
"""
```

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:

[32]:
```
c.logodds.Male - c.logodds.Female
```

[32]:
```
0.8850500036644218
```

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.

[33]:
```
model = sm.GLM.from_formula('smq ~ RIDAGEYR + RIAGENDRx', family=sm.families.
↪Binomial(), data=da)
result = model.fit()
result.summary()
```

[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    smq   No. Observations:                 5094
Model:                            GLM   Df Residuals:                     5091
Model Family:                Binomial   Df Model:                           2
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -3296.6
Date:                Fri, 01 Jan 2021   Deviance:                       6593.2
Time:                        18:25:42   Pearson chi2:                 5.10e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
=====
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-----
Intercept           -1.6166      0.095    -16.985      0.000      -1.803
-1.430
RIAGENDRx[T.Male]    0.8920      0.059     15.170      0.000       0.777
```

```
1.007
RIDAGEYR               0.0172       0.002      10.289       0.000        0.014
0.021
==============================================================================
=====
"""
```

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 1.41 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, and we will recode it so that the meaning of the levels becomes more clear. We will call the recoded variable `DMDEDUC2x`.

[34]:
```
# create a labeled version of the educational attainment variable
values = {
    1: 'lt9',
    2: 'x9_11',
    3: 'HS',
    4: 'SomeCollege',
    5: 'College',
    7: np.nan,
    9: np.nan
}
da['DMDEDUC2x'] = da.DMDEDUC2.replace(values)

model = sm.GLM.from_formula(
    'smq ~ RIDAGEYR + RIAGENDRx + DMDEDUC2x',
    family=sm.families.Binomial(),
    data=da
)
result = model.fit()
result.summary()
```

[34]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    smq   No. Observations:                 5093
Model:                            GLM   Df Residuals:                     5086
Model Family:                Binomial   Df Model:                            6
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -3201.2
Date:                Fri, 01 Jan 2021   Deviance:                       6402.4
Time:                        18:25:42   Pearson chi2:                 5.10e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
============
                                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
------------
Intercept                     -2.3060      0.114    -20.174      0.000      -2.530
-2.082
RIAGENDRx[T.Male]              0.9096      0.060     15.118      0.000       0.792
1.028
```

```
DMDEDUC2x[T.HS]              0.9434      0.090     10.521      0.000       0.768
1.119
DMDEDUC2x[T.SomeCollege]     0.8322      0.084      9.865      0.000       0.667
0.998
DMDEDUC2x[T.lt9]             0.2662      0.109      2.438      0.015       0.052
0.480
DMDEDUC2x[T.x9_11]           1.0986      0.107     10.296      0.000       0.889
1.308
RIDAGEYR                     0.0183      0.002     10.582      0.000       0.015
0.022
==============================================================================
============
"""
```

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)`.

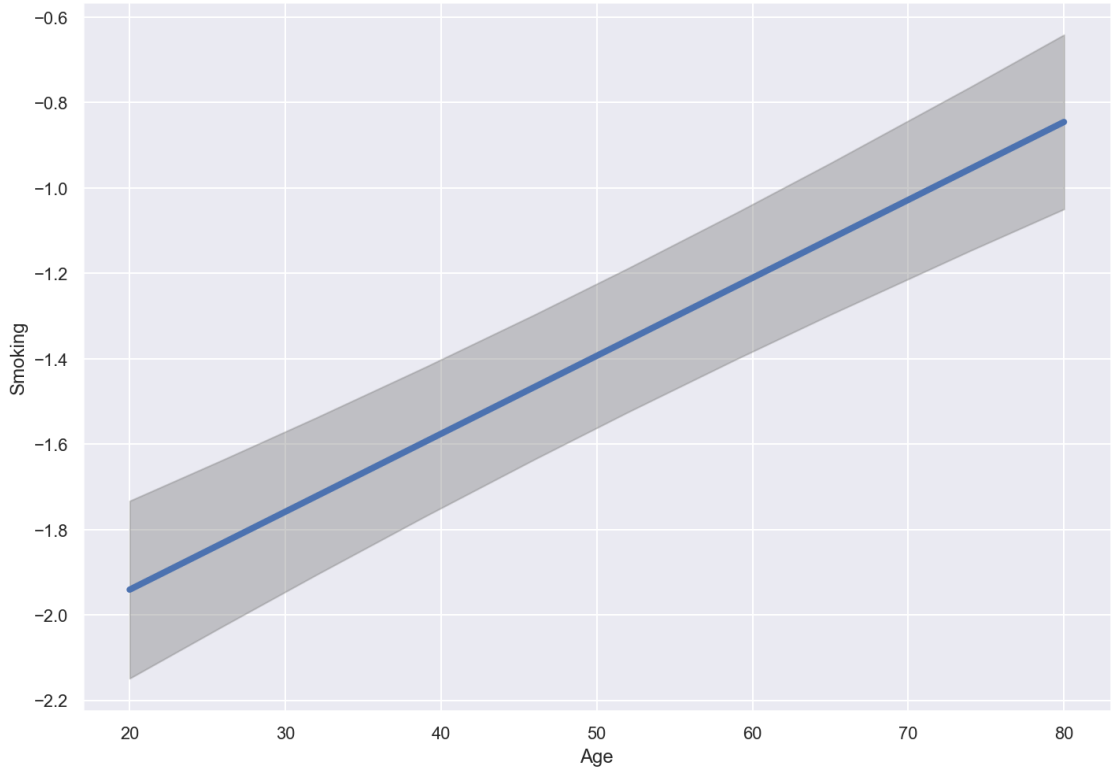
As noted above when we were discussing linear regression, it is important to remember that a coefficient in a logistic regression are "conditional" on the other variables being held fixed. For example, the log odds ratio of 1.09 between people with 9-11 years of schooling and people who completed college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

[35]:
```
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')
```

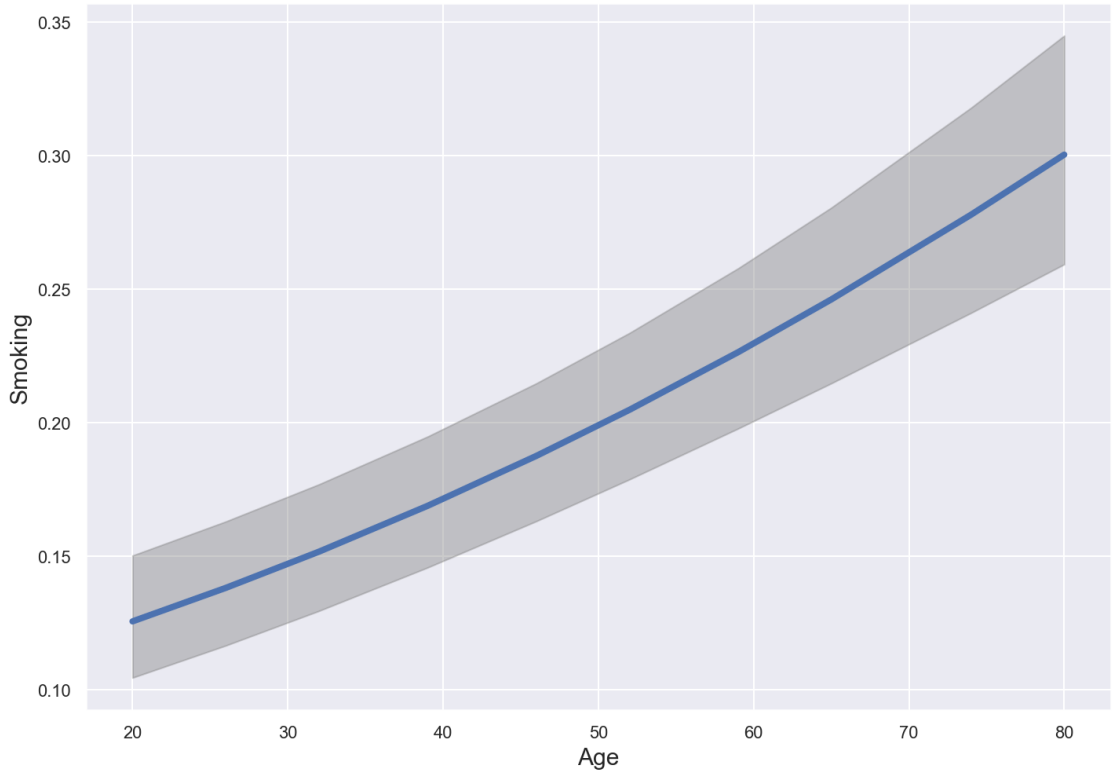
[35]: `Text(0, 0.5, 'Smoking')`



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 this interval.

[36]:
```
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)
```

[36]:
```
Text(0, 0.5, 'Smoking')
```

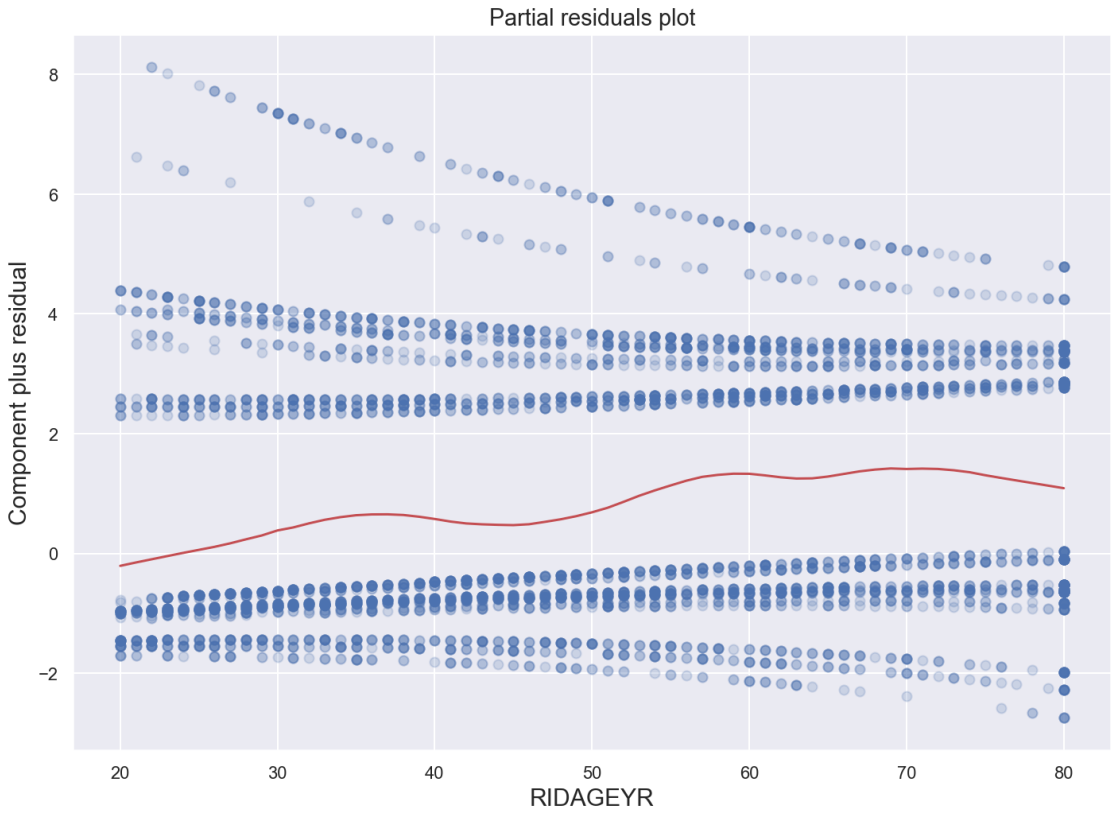


Next we turn to diagnostic plots that are intended to reveal certain aspects of the data that may not be correctly captured by the model. The three plots below are intended to reveal any curvature in the mean relationship between the outcome and one of the covariates. We used the partial regression plotting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

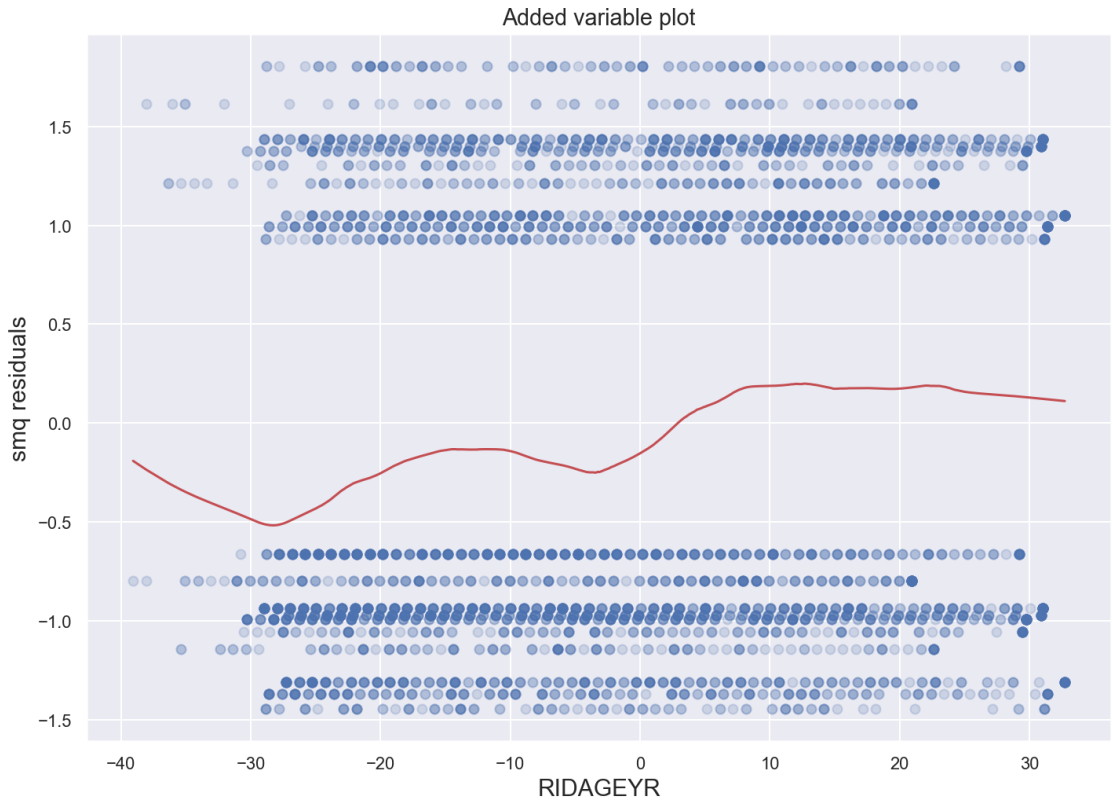
In the case of logistic regression, the three techniques demonstrated below can identify major discrepancies between the fitted model and the population, but evidence for small discrepancies is not reliable unless the sample size is very large. The CERES technique has the strongest theoretical support. 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.

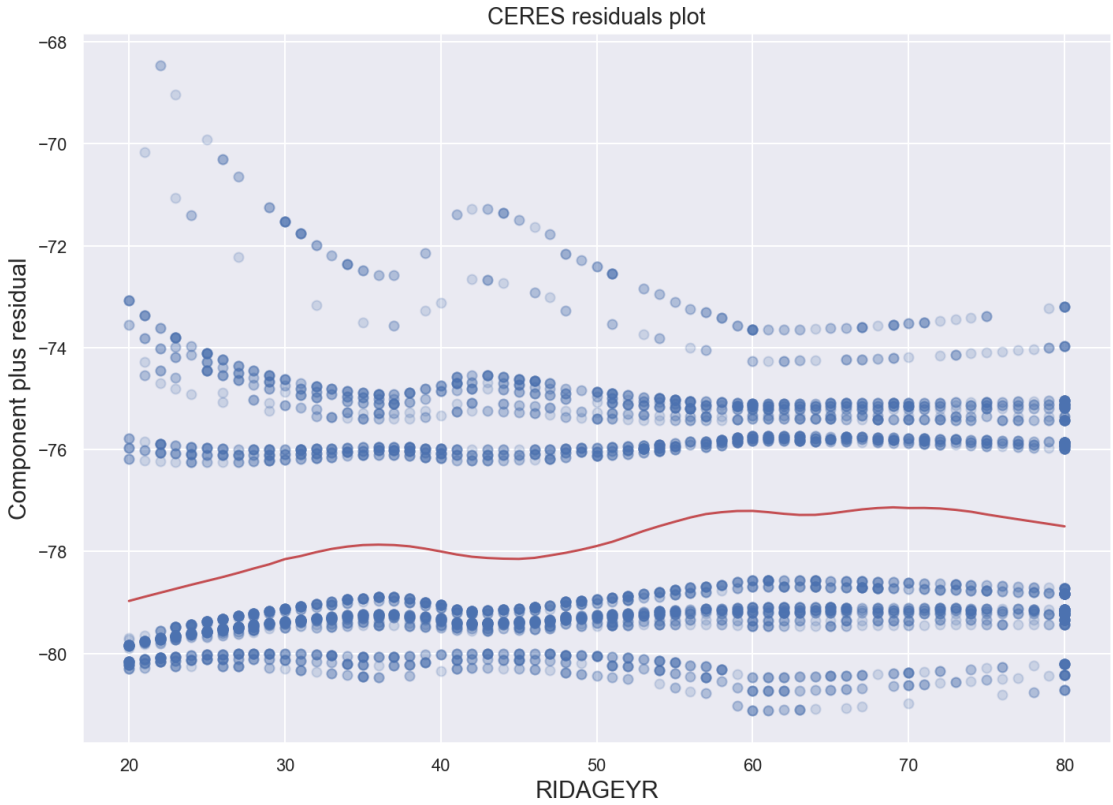
[37]:
```
fig = result.plot_partial_residuals('RIDAGEYR')
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)

_ = add_lowess(ax)
```



[38]:
```
fig = result.plot_added_variable('RIDAGEYR')
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)
_ = add_lowess(ax)
```



[39]:
```
fig = result.plot_ceres_residuals('RIDAGEYR')
ax = fig.get_axes()[0]
ax.lines[0].set_alpha(0.2)
_ = add_lowess(ax)
```



Fitting a Multilevel Model This analysis will be focusing on a longitudinal study that was conducted on children with autism1. We will be looking at several variables and exploring how different factors interact with the socialization of a child with autism as they progress throughout the beginning stages of their life.

The variables we have from the study are:

- AGE is the age of a child which, for this dataset, is between two and thirteen years
- VSAE measures a child's socialization
- SICDEGP is the expressive language group at age two and can take on values ranging from one to three. Higher values indicate more expressive language.
- CHILDID is the unique ID that is given to each child and acts as their identifier within the dataset

We will first be fitting a multilevel model with explicit random effects of the children to account for the fact that we have repeated measurements on each child, which introduces correlation in our observations.

1 Anderson, D., Oti, R., Lord, C., and Welch, K. (2009). Patterns of growth in adaptive social abilities among children with autism spectrum disorders. Journal of Abnormal Child Psychology, 37(7), 1019-1034.

[1]:
```
from IPython.display import display, HTML
from scipy.stats import chi2
from sklearn import linear_model
import csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import patsy
import statsmodels.api as sm
import warnings

warnings.filterwarnings('ignore')
```

[2]:
```
dat = pd.read_csv('data/autism.csv').dropna()
original = dat.copy()
```

[3]:
```
dat.head()
```

[3]:

	age	vsae	sicdegp	childid
0	2	6.0	3	1
1	3	7.0	3	1
2	5	18.0	3	1
3	9	25.0	3	1
4	13	27.0	3	1

We will first begin by fitting the model without centering the age component first. This model has both random intercepts and random slopes on age.

[4]:
```
mlm_mod = sm.MixedLM.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    groups='childid',
    re_formula='1 + age',
    data=dat
)

mlm_result = mlm_mod.fit()
mlm_result.summary()
```

[4]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
             Mixed Linear Model Regression Results
============================================================
Model:               MixedLM   Dependent Variable:   vsae
No. Observations:    610       Method:               REML
No. Groups:          158       Scale:                62.2592
Min. group size:     1         Log-Likelihood:       -2348.7987
Max. group size:     5         Converged:            No
```

```
Mean group size:     3.9
------------------------------------------------------------
                     Coef.  Std.Err.   z    P>|z| [0.025 0.975]
------------------------------------------------------------
Intercept             1.901    1.600  1.188 0.235 -1.235  5.038
C(sicdegp)[T.2]      -0.415    2.109 -0.197 0.844 -4.549  3.718
C(sicdegp)[T.3]      -3.917    2.345 -1.670 0.095 -8.514  0.680
age                   2.957    0.593  4.986 0.000  1.794  4.119
age:C(sicdegp)[T.2]   0.741    0.784  0.945 0.344 -0.795  2.277
age:C(sicdegp)[T.3]   4.356    0.869  5.014 0.000  2.653  6.058
childid Var          58.265    2.990
childid x age Cov   -28.736    0.697
age Var              14.204    0.283
============================================================

"""
```

We can see that the model fails to converge. Taking a step back, and thinking about the data, how should we expect children's socialization to vary at age zero? Would we expect the children to exhibit different socialization when they are first born? Or is the difference in socialization something that we would expect to manifest over time?

We would expect the socialization differences should be negligible at age zero or, at the very least, difficult to discern. This homogeneity of newborns implies the variance of the random intercept would be close to zero, and, as a result, the model is having difficulty estimating the variance parameter of the random intercept. It may not make sense to include a random intercept in this model. We will drop the random intercept and attempt to refit the model to see if the convergence warnings still manifest themselves in the fit.

[5]:
```
# build the model - note the re_formula definition now
# has a 0 instead of a 1. This removes the intercept from
# the model
mlm_mod = sm.MixedLM.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    groups='childid',
    re_formula='0 + age',
    data=dat
)

mlm_result = mlm_mod.fit()
mlm_result.summary()
```

[5]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
            Mixed Linear Model Regression Results
============================================================
Model:              MixedLM   Dependent Variable:   vsae
No. Observations:   610       Method:               REML
```

```
No. Groups:          158       Scale:                 84.5319
Min. group size:     1         Log-Likelihood:        -2427.0905
Max. group size:     5         Converged:             Yes
Mean group size:     3.9
-------------------------------------------------------------
                    Coef.  Std.Err.   z    P>|z| [0.025 0.975]
-------------------------------------------------------------
Intercept            2.482    1.271  1.952 0.051 -0.010  4.973
C(sicdegp)[T.2]     -1.293    1.674 -0.773 0.440 -4.574  1.987
C(sicdegp)[T.3]     -4.230    1.862 -2.272 0.023 -7.880 -0.580
age                  2.822    0.470  6.006 0.000  1.901  3.743
age:C(sicdegp)[T.2]  0.985    0.620  1.589 0.112 -0.230  2.199
age:C(sicdegp)[T.3]  4.463    0.688  6.482 0.000  3.113  5.812
age Var              8.198    0.124
=============================================================

"""
```

The model now converges, which is an indication that removing the random intercepts from the model was beneficial computationally.

First, we notice that the interaction term between the expressive language group and the age of children is positive and significant for the third expressive language group. This is an indication that the increase in socialization as a function of age for this group is significantly larger relative to the first expressive language group (i.e., the age slope is significantly larger for this group relative to the first expressive language group).

When we think about the interpretation of the parameters, however, we need to be cautious. The intercept can be interpreted as the mean socialization when a child in the first expressive language group is zero years old. This may not be sensible to estimate. To improve this interpretation, we should center the age variable and, again, fit the model.

[6]:
```
# center the age variable
dat['age'] = dat.groupby('childid')['age'].transform(lambda x: x - x.mean())
dat.head()
```

[6]:

	age	vsae	sicdegp	childid
0	-4.4	6.0	3	1
1	-3.4	7.0	3	1
2	-1.4	18.0	3	1
3	2.6	25.0	3	1
4	6.6	27.0	3	1

[7]:
```
# refit the model, again, without the random intercepts
mlm_mod = sm.MixedLM.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    groups='childid',
    re_formula='0 + age',
```

```
    data=dat
)

mlm_result = mlm_mod.fit()
mlm_result.summary()
```

[7]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
            Mixed Linear Model Regression Results
=============================================================
Model:              MixedLM   Dependent Variable:   vsae
No. Observations:   610       Method:               REML
No. Groups:         158       Scale:                410.7496
Min. group size:    1         Log-Likelihood:       -2752.2106
Max. group size:    5         Converged:            Yes
Mean group size:    3.9
-------------------------------------------------------------
                    Coef.  Std.Err.   z    P>|z| [0.025 0.975]
-------------------------------------------------------------
Intercept           17.421    1.470 11.848 0.000 14.539 20.303
C(sicdegp)[T.2]      6.359    1.942  3.274 0.001  2.552 10.166
C(sicdegp)[T.3]     23.403    2.157 10.852 0.000 19.176 27.630
age                  2.731    0.641  4.257 0.000  1.474  3.988
age:C(sicdegp)[T.2]  1.188    0.843  1.409 0.159 -0.465  2.840
age:C(sicdegp)[T.3]  4.555    0.931  4.891 0.000  2.730  6.381
age Var              9.609    0.112
=============================================================

"""
```

Significance Testing The next question that we need to ask is if the addition of the random age effects is actually significant; should we retain these random effects in the model? First, we will fit the multilevel model including centered age again. This time, however, we will compare it to the model that does not have random effects:

[8]:
```
# random effects mixed model
mlm_mod = sm.MixedLM.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    groups='childid',
    re_formula='0 + age',
    data=original
)

# ols model - no mixed effects
ols_mod = sm.OLS.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    data=original
```

```
)

mlm_result = mlm_mod.fit()
ols_result = ols_mod.fit()

print(mlm_result.summary())
print(ols_result.summary())
```

```
            Mixed Linear Model Regression Results
============================================================
Model:              MixedLM   Dependent Variable:   vsae
No. Observations:   610       Method:               REML
No. Groups:         158       Scale:                84.5319
Min. group size:    1         Log-Likelihood:       -2427.0905
Max. group size:    5         Converged:            Yes
Mean group size:    3.9
------------------------------------------------------------
                    Coef.  Std.Err.   z    P>|z| [0.025 0.975]
------------------------------------------------------------
Intercept            2.482    1.271  1.952 0.051 -0.010  4.973
C(sicdegp)[T.2]     -1.293    1.674 -0.773 0.440 -4.574  1.987
C(sicdegp)[T.3]     -4.230    1.862 -2.272 0.023 -7.880 -0.580
age                  2.822    0.470  6.006 0.000  1.901  3.743
age:C(sicdegp)[T.2]  0.985    0.620  1.589 0.112 -0.230  2.199
age:C(sicdegp)[T.3]  4.463    0.688  6.482 0.000  3.113  5.812
age Var              8.198    0.124
============================================================

                            OLS Regression Results
==============================================================================
Dep. Variable:                   vsae   R-squared:                       0.470
Model:                            OLS   Adj. R-squared:                  0.465
Method:                 Least Squares   F-statistic:                     107.1
Date:                Wed, 06 Jan 2021   Prob (F-statistic):           7.46e-81
Time:                        10:39:15   Log-Likelihood:                -2762.1
No. Observations:                 610   AIC:                             5536.
Df Residuals:                     604   BIC:                             5563.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
=======
                           coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept               2.6342      2.864      0.920      0.358      -2.990
8.259
```

```
C(sicdegp)[T.2]        -3.2405      3.781     -0.857      0.392     -10.666
4.185
C(sicdegp)[T.3]        -3.1418      4.256     -0.738      0.461     -11.501
5.217
age                     2.6159      0.416      6.285      0.000       1.799
3.433
age:C(sicdegp)[T.2]     1.6173      0.545      2.968      0.003       0.547
2.688
age:C(sicdegp)[T.3]     4.3644      0.607      7.190      0.000       3.172
5.556
==============================================================================
Omnibus:                      305.670   Durbin-Watson:                   1.288
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2480.083
Skew:                           2.069   Prob(JB):                         0.00
Kurtosis:                      11.969   Cond. No.                         48.8
==============================================================================

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

Now, we perform the significance test with a mixture of chi-squared distributions. We repeat the information from the Likelihood Ratio Tests writeup for this week here:

- Null hypothesis: The variance of the random child effects on the slope of interest is zero (in other words, these random effects on the slope are not needed in the model)
- Alternative hypothesis: The variance of the random child effects on the slope of interest is greater than zero
- First, fit the model WITH random child effects on the slope of interest, using restricted maximum likelihood estimation
  - -2 REML log-likelihood = 4854.18
- Next, fit the nested model WITHOUT the random child effects on the slope:
  - -2 REML log-likelihood = 5524.20 (higher value = worse fit!)
- Compute the positive difference in the -2 REML log-likelihood values ("REML criterion") for the models:
  - Test Statistic (TS) = 5524.20 – 4854.18 = 670.02
- Refer the TS to a mixture of chi-square distributions with 1 and 2 DF, and equal weight 0.5:

[9]:
```
# compute the p-value using a mixture of chi-squared distributions
# because the chi-squared distribution with zero degrees of freedom has no
# mass, we multiply the chi-squared distribution with one degree of freedom by
# 0.5
p_val = 0.5 * (1 - chi2.cdf(670.02, 1))
print(f'The p-value of our significance test is: {p_val}')
```

```
The p-value of our significance test is: 0.0
```

The p-value is so small that we cannot distiguish it from zero. With a p-value this small, we can safely reject the null hypothesis. We have sufficient evidence to conclude that the variance of the random effects on the slope of interest is greater than zero.

Marginal Models While we have accounted for correlation among observations from the same children using random age effects in the multilevel model, marginal models attempt to manage the correlation in a slightly different manner. This process of fitting a marginal model, utilizing a method known as Generalized Estimating Equations (GEEs), aims to explicitly model the within-child correlations of the observations.

We will specify two types of covariance structures for this analysis. The first will be an exchangeable model. In the exchangeable model, the observations within a child have a constant correlation, and constant variance.

The other covariance structure that we will assume is independence. An independent covariance matrix implies that observations within the same child have zero correlation.

We will see how each of these covariance structures affect the fit of the model.

[10]:
```
# fit the exchangable covariance GEE
model_exch = sm.GEE.from_formula(
    formula='vsae ~ age * C(sicdegp)',
    groups='childid',
    cov_struct=sm.cov_struct.Exchangeable(),
    data=dat
).fit()

# fit the independent covariance GEE
model_indep = sm.GEE.from_formula(
    'vsae ~ age * C(sicdegp)',
    groups='childid',
    cov_struct=sm.cov_struct.Independence(),
    data=dat
).fit()

# we cannot fit an autoregressive model, but this is how
# we would fit it if we had equally spaced ages
# model_indep = sm.GEE.from_formula(
#     'vsae ~ age * C(sicdegp)',
#     groups='age',
#     cov_struct=sm.cov_struct.Autoregressive(),
#     data=dat
# ).fit()
```

The autoregressive model cannot be fit because the age variable is not spaced uniformly for each child's measurements (every year or every two years for each measurement). If it was, we can fit it with the commented code above. We will now see how each of the model fits compare to one another:

[11]:
```
# construct a datafame of the parameter estimates and their standard errors
data = {
    'OLS_Params': ols_result.params,
    'OLS_SE': ols_result.bse,
    'MLM_Params': mlm_result.params,
    'MLM_SE': mlm_result.bse,
    'GEE_Exch_Params': model_exch.params,
    'GEE_Exch_SE': model_exch.bse,
    'GEE_Indep_Params': model_indep.params,
    'GEE_Indep_SE': model_indep.bse
}
x = pd.DataFrame(data)

# ensure the ordering is logical
x = x[list(data.keys())]
x = np.round(x, 2)

display(HTML(x.to_html()))
```

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

We can see that the estimates for the parameters are relatively consistent among each of the modeling methodologies, but the standard errors differ from model to model. Overall, the two GEE models are mostly similar and both exhibit standard errors for parameters that are slightly larger than each of their corresponding values in the OLS model. The multilevel model has the largest standard error for the age coefficient, but the smallest standard error for the intercept. Overall, we see that we would make similar inferences regarding the importance of these fixed effects, but remember that we need to interpret the multilevel models estimates conditioning on a given child. For example, considering the age coefficient in the multilevel model, we would say that as age increases by one year *for a given child* in the first expressive language group, VSAE is expected to increase by 2.73. In the GEE and OLS models, we would say that as age increases by one year *in general*, the average VSAE is expected to increase by 2.60.

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

[12]:
```
da = pd.read_csv('data/NHANES.csv')

# drop unused columns
# drop rows with any missing values
columns = [
    'BPXSY1',
    'RIDAGEYR',
    'RIAGENDR',
    'RIDRETH1',
    'DMDEDUC2',
    'BMXBMI',
    'SMQ020',
    'SDMVSTRA',
    'SDMVPSU'
]
da = da[columns].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:

[13]:
```
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:

[14]:
```
model = sm.GEE.from_formula(
    'BPXSY1 ~ 1',
    groups='group',
    cov_struct=sm.cov_struct.Exchangeable(),
    data=da
)

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

[14]:
```
'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.

[15]:
```
# 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:

[16]:
```
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 -0.001
SDMVSTRA The correlation between two observations in the same cluster is 0.001
SDMVSTRA The correlation between two observations in the same cluster is -0.001
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 -0.001
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 0.001
SDMVSTRA The correlation between two observations in the same cluster is -0.000
```

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:

[17]:
```
model = sm.GEE.from_formula(
    'BPXSY1 ~ RIDAGEYR',
    groups='group',
    cov_struct=sm.cov_struct.Exchangeable(),
    data=da
)
result = model.fit()
result.cov_struct.summary()
```

[17]:
```
'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.

[18]:
```
# 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()
result.cov_struct.summary()
```

[18]:
```
'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.

[19]:
```
# 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()

data = {
    'OLS_params': result1.params,
    'OLS_SE': result1.bse,
    'GEE_params': result2.params,
    'GEE_SE': result2.bse
}

x = pd.DataFrame(data)
x = x[list(data.keys())]
x
```

[19]:

	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 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.

[20]:
```
# relabel the levels, convert rare categories to missing
mapping = {
    1: 'lt9',
    2: 'x9_11',
    3: 'HS',
    4: 'SomeCollege',
    5: 'College',
    7: np.nan,
    9: np.nan
}
da['DMDEDUC2x'] = da.DMDEDUC2.replace(mapping)

# 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)

data = {
    'OLS_params': result1.params,
    'OLS_SE': result1.bse,
    'GEE_params': result2.params,
    'GEE_SE': result2.bse
}

x = pd.DataFrame(data)
```

```
x = x[list(data.keys())]
x
```

[20]:

	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.

[21]:
```
# 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()
```

[21]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
           Mixed Linear Model Regression Results
============================================================
Model:             MixedLM  Dependent Variable:  BPXSY1
No. Observations:  5102     Method:              REML
No. Groups:        30       Scale:               256.6952
Min. group size:   106      Log-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:

[22]:
```
result.random_effects
```

[22]:
```
{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 *BLUPs*), 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.

[23]:
```
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()
```

[23]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
             Mixed Linear Model Regression Results
================================================================
Model:              MixedLM   Dependent Variable:   BPXSY1
No. Observations:   5102      Method:               REML
No. Groups:         30        Scale:                263.7323
Min. group size:    106       Log-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.

[24]:
```
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()
```

[24]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
              Mixed Linear Model Regression Results
================================================================
Model:                MixedLM   Dependent Variable:   BPXSY1
No. Observations:     5102      Method:               REML
No. Groups:           30        Scale:                255.4451
Min. group size:      106       Log-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.

[1]:
```
from IPython.display import Image
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as st
import seaborn as sns
```

[2]:
```
%config InlineBackend.figure_format = 'retina'
```

Bayesian Approaches to Statistics and Modeling

- Image we are trying to determine average IQ of students at the University of Michigan.
- Someone asks me, *Hey, what do you think the average IQ could be?*
- I look online, and see that the IQ test scores, for the U.S. population, are, by design, normally distributed with μ=100, σ=10. I'll start out with that as an "educated guess", knowing that it may not be best belief for the mean but will be good place to start.

[3]:
```
Image('images/bayesian_1.png')
```

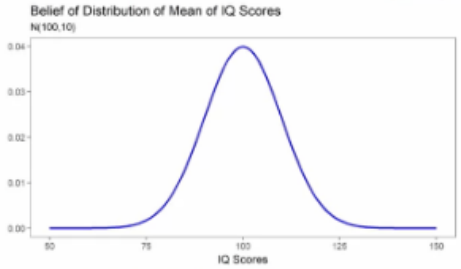
[3]:

Average IQ

Belief
Michigan IQ scores
$\sim Norm(100, 10)$

Observations
125

- Now, I go and test someone's IQ on campus. They have an IQ of 125. How does this change my belief?



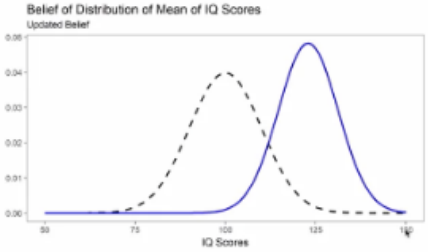
- Does my belief shift left, right, or stay the same?

Average IQ

Belief
Michigan IQ scores
$\sim Norm(100, 10)$

Observations
125

- It should shift right. We had a belief that the mean was 100…



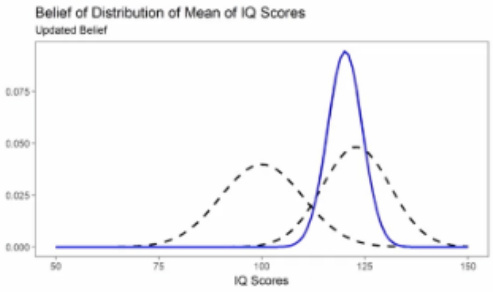
- Seeing a value of 125 indicates the mean might be higher

Average IQ

Belief
Michigan IQ scores
$\sim Norm(100, 10)$

Observations
125, 115

- Now we observe a student with a 115 IQ. How does this change our belief?

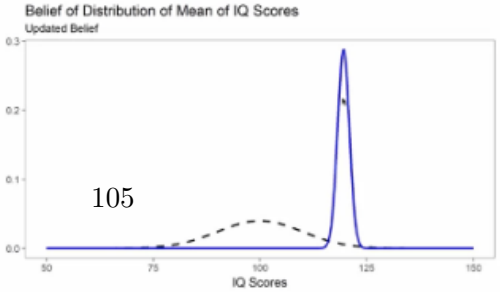


Average IQ

Belief
Michigan IQ scores
$\sim Norm(100, 10)$

Observations
125, 115, 115, 120, 125, 117

- What happens if we observe more and more data? I observe IQ scores of 115, 120, 125, and 117

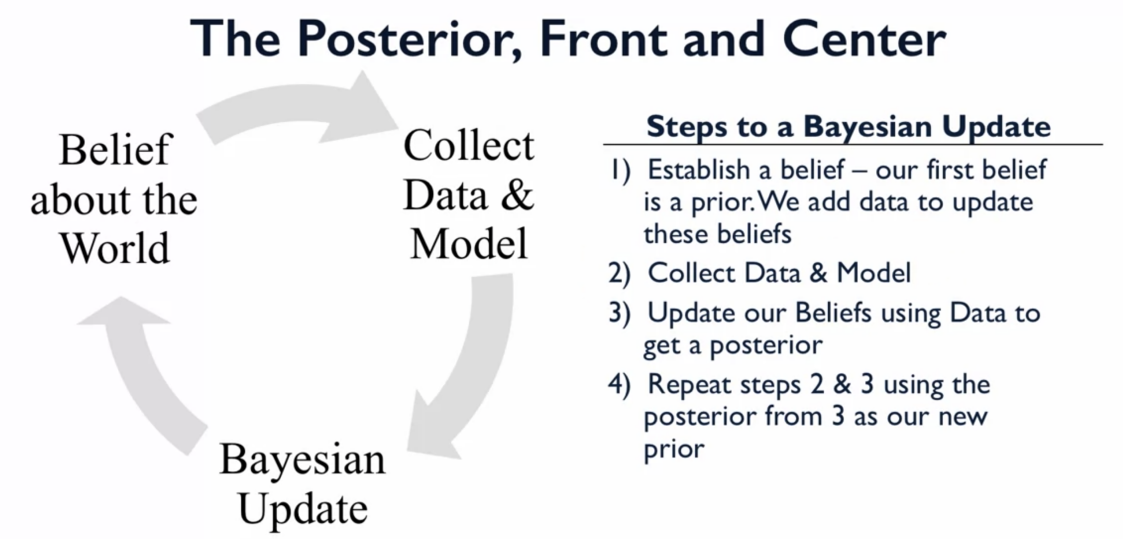


- As you can see in the images above, the mean value changes with the observations added after selecting good place to start.
- Okey, so what happens if we observe more and more data?
- We can see that more and more data allows us to better focus in our belief about the mean of IQ scores at U of M.
- The process that we went through is called **Bayesian Updating**. This provides a distribution on the quantity of interest, called the **posterior**.
- The posterior allows us to update our beliefs and answer questions about the quantity of interest.

!!!Caution Imagine this is my data in a Bayesian setting. I don't believe that there's a single mu, I believe that there's an entire distribution of them, and we're going to try to infer what this distribution is using the data. In a frequentist analysis, this is a constant. We don't know, we try to estimate. In a Bayesian analysis, this is a belief that we keep updating with new data, and this is a very different way of thinking about the world.

[4]:
```
Image('images/bayesian_2.png', width=600)
```

[4]:



Bayesian Approaches Case Study

- Question: Does a child's IQ have a relationship with the IQ of their mother?
- Data: National Longitudinal Survey of Youth. Source: Gelman and Hill (2007)
- 434 observations

	kid_score	mom_hs	mom_iq	mom_age
1	65	1	121	27
2	98	1	89	25

	kid_score	mom_hs	mom_iq	mom_age
3	85	1	115	27
4	83	1	99	25
5	115	1	93	

** mom_hs: went to high school or not?

The Model

- We'll use a linear model
- We'll start out with the most basic of linear regression models
- Regression form:

$$ChildIQ = \beta_0 + \beta_1(momIQ) + \beta_2(momAge)$$

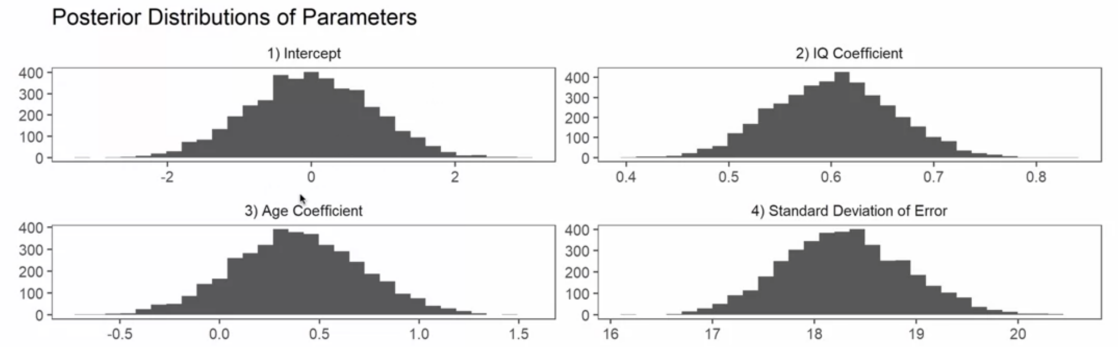
- Up until now, we haven't done anything different than what we have done in past.
- The Bayesian framework, however, says that we have to specify prior distributions for our beliefs as well as likelihoods.
- **Key Point**: Every parameter must begin with a distribution that captures our beliefs. We call this **priors**.

- In Bayesian setting, we need priors on each of our parameters.
- $\beta_0 \sim N(0, 20)$ I put a relatively weak prior on the intercept term to allow it to vary a lot if it needs to. The data will be centered and so this isn't a large concern
- $\beta_1 \sim N(1, 5)$ I put a prior centered at one with a wide variance to account for the fact that I expect that a child's IQ is able to be predicted very well by a mother's IQ but I'm not certain.
- $\beta_2 \sim N(0, 5)$ I put a prior centered at zero because I have no idea how age will affect IQ. The large variance accounts for my ignorance in what this value may be.
- Therefore, my first belief is that if the mother's IQ is 110, the child's IQ also is 110. Because I start with β_0 as 0, β_1 as 1 and β_2 as 0 in my first belief.
- Okay, my first belief is true? Probably not. So, I need to **general error**(σ_e).

$$ChildIQ_i \sim N(\beta_0 + \beta_1(momIQ_i) + \beta_2(momAge_i), \sigma_e)$$

[5]:
```
Image('images/bayesian_3.png', width=600)
```

[5]:

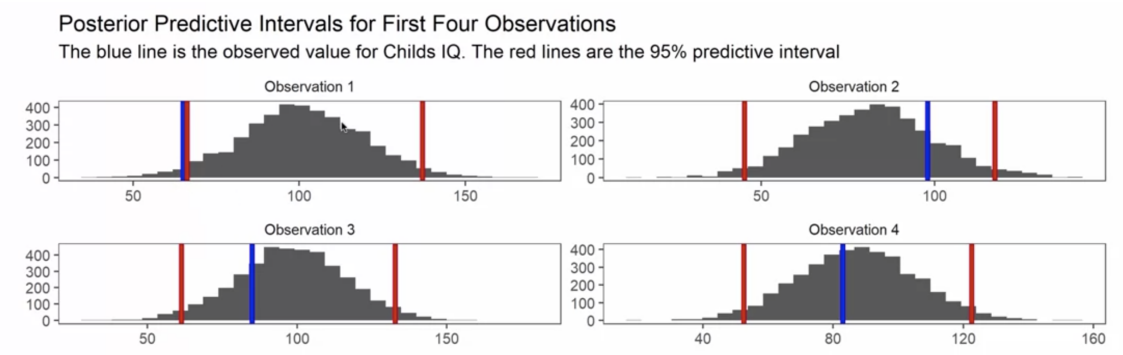


You can see the coefficient distributions above. Unlike the frequentist approach, we can examine the coefficients of the parameters, not only distribution of values in the Bayesian approach.

At first glance, we can say that the mean of the coefficient for the IQ parameter is around 0.6 and also, zero is not available value for the coefficient for this parameter. But, when we looking at the distribution of the Age coefficient, we can say that zero is available value for this coefficient. The mean of this coefficient looks around 0.45 but we shouldn't forget the zero value is available.

[6]:
```
Image('images/bayesian_4.png', width=600)
```

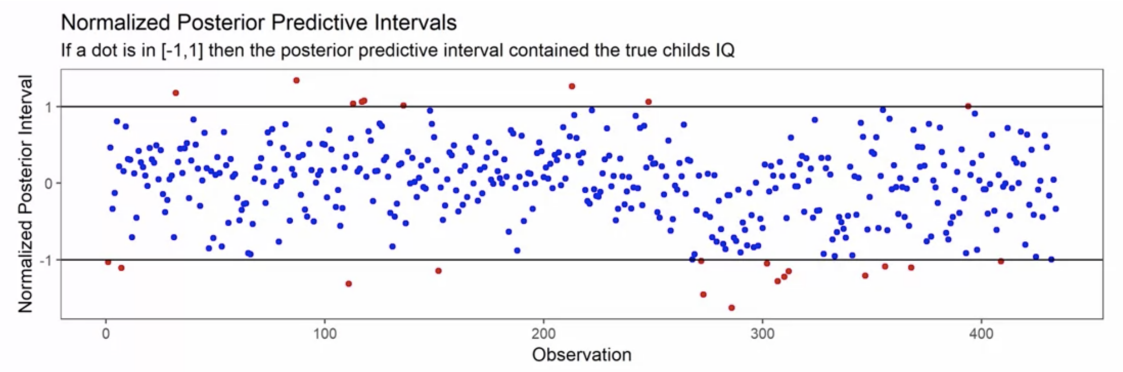
[6]:



Then, we can took the first observation and run the model. Whoops, as you can see, the blue line in the first graph is out of the confidence interval. Okay so, we can put it through the model again. And we do this over and over again.

[7]:
```
Image('images/bayesian_5.png', width=600)
```

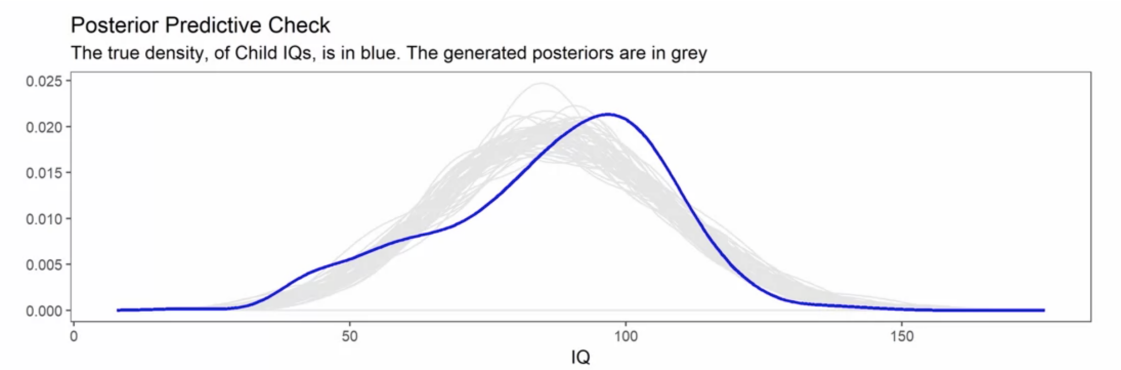
[7]:



When we repeat this for each set and represent these values between -1 and 1, we can see the graph above. The posterior interval, the 95% posterior interval, would have covered them. It would have made the correct prediction. All the points that are outside, all of these red points, these are all the points that the posterior interval would have missed.

[8]:
```
Image('images/bayesian_6.png', width=600)
```

[8]:



So this is the distribution of the actual values in the dataset. Then for each of our predictive intervals, let's say that we made predictions on every single data point, and we plotted the density of that. So this could've been one of our sets of predictions, this could've been another set. And because we're using a Bayesian framework, these sets have variability. What this is telling us is this plot can tell us if we have systemic bias in either direction. And it can also tell us if the variance of our estimates are approximately correct. So we see that the width of this interval is roughly equal to the width of this interval.

The mean of all of our predictions looks to be perhaps a little bit off center from what the mean of this blue distribution is. Furthermore, we have this tail right here. And what this is saying is, this is saying we predict that very few people will have IQs. Maybe less than 50 or from this 50 to 70 range. But we predict a lot more will have maybe from 70 to 85 range. So we're overpredicting, sorry, we're underpredicting the amount here, but overpredicting the amount here. And because of this, this is telling me that my estimates may be biased. And furthermore, in general, our estimates may be pulled to the left, trying to account for this odd fat tail right here. Overall, all of our estimates are heading that way, and so this is something that we can pick up on. And just so we can get a better intuition of these plots, imagine that our predictions were perfect. Imagine that this was the plot of the data. If our predictions were perfect, we should see a bunch of lines that very closely follow the actual data, or close to perfect. So we should expect to see this, but instead we're seeing this.

The next step goes like this,

- We are going to allow all three parameters to vary according to whether a mother attended HS and their IQ group
  - For example, we will have a different β_0, β_1, and β_2 for mothers who are high IQ and attended HS, for those mothers that did not attend HS but also have a high IQ, etc.
- This model now has six sets of $\beta_0 s$, $\beta_1 s$, and $\beta_2 s$
- We are also going to say that each of the $\beta_0 s$ come from a common distribution. We'll also do the same with the $\beta_1 s$ and the $\beta_2 s$.

CAUTION: The lecture of this model is in the training videos, but I did not add it to the notes because it is very detailed.

Bayesian in Python In this tutorial, we are going to go over basic bayesian analysis in python.

Review **Prior p(H):** Our prior reflects what we know about the value of some parameter before seeing data. This could refer to previous trials and distributions.

Likelihood p(D|H): what is the plausibility that our data is observed, given our prior?

Posterior p(H|D): This is result of the Bayesian analysis and reflects all that we know about a problem (given our data and model).

Evidence p(D): Evidence is the probability of observing the data averaged over all the possible values the parameters can take. Also knowns as the noramlziing factor. The normalising constant makes sure that the resulting posterior distribution is a true probability distribution by ensuring that the sum of the distribution is equal to 1.

Because p(D) is considered a normalizing constant we can say: $p(H|D) \propto p(D|H) * p(H)$

Coin - Flipping Problem Let's think of these terms in the context of a coin-flipping experiment.

On a standard coin, we have two sides, heads or tails. Both of which are equally likely to show after a coin flip, or a 50% probability.

In the case of a coin-flipping trials, we may want to consider this probability our prior.

Let's go ahead and create our prior distribution:

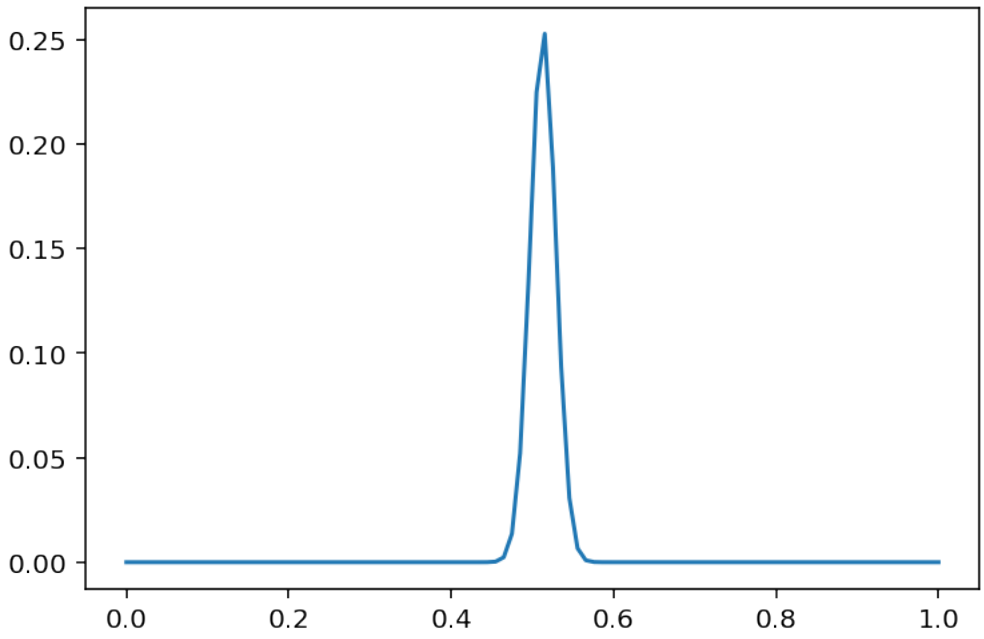
[9]:
```
coin_flips_prior = np.random.binomial(n=1, p=0.5, size=1000)
coin_flips_prior[:5]
```

[9]:
```
array([0, 1, 0, 1, 0])
```

[10]:
```
params = np.linspace(0, 1, 100)
```

[11]:
```
p_prior = np.array(
    [
        np.product(st.bernoulli.pmf(coin_flips_prior, p))
        for p in params
    ]
)
```

[12]:
```
p_prior = p_prior / np.sum(p_prior)
plt.plot(params, p_prior);
```

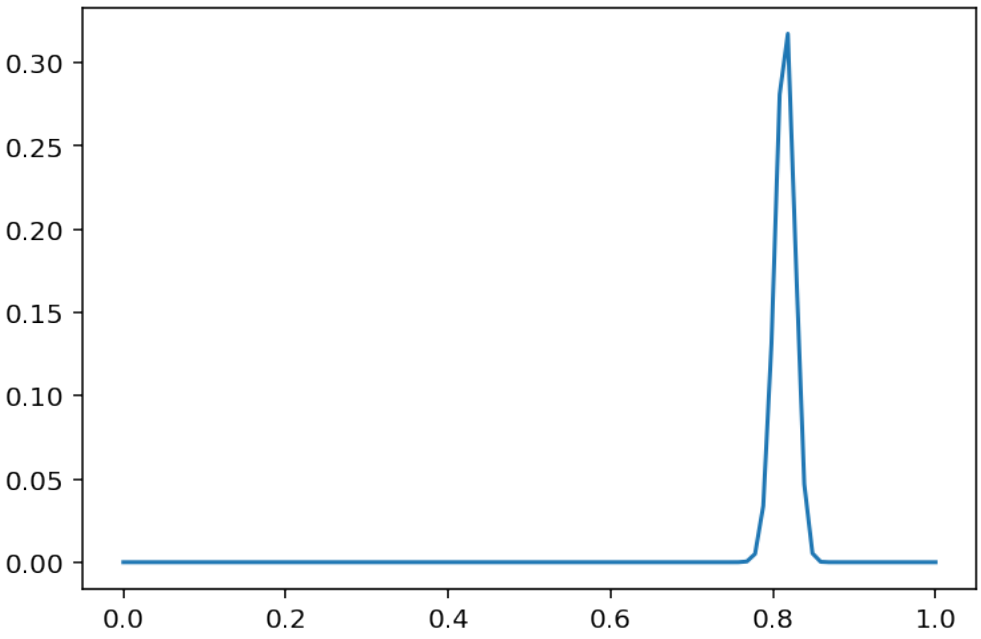


As you can see, our prior distribution peaks at 0.5 which is what our probability for our fair coin is.

Now, let's introduce some observations from trials with an unfair coin. Let's say the probability is now weight 80-20, where the probability a head is shown is 0.8.

Let's create this sampling distribution:

[13]:
```
coin_flips_observed = np.random.binomial(n=1, p=0.8, size=1000)
p_observed = np.array(
    [
        np.product(st.bernoulli.pmf(coin_flips_observed, p))
        for p in params
    ]
)

p_observed = p_observed / np.sum(p_observed)
plt.plot(params, p_observed);
```

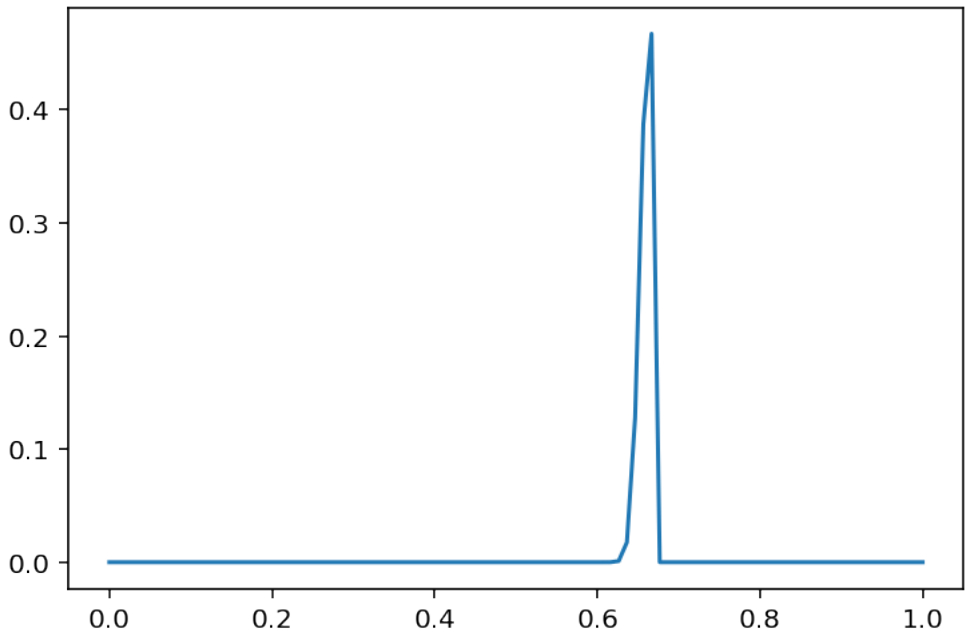


The peak for our sampling distribution is around 0.8.

While our observations from our sampling distribution indicate a probability around 0.8, because our prior is 0.5, we have to assess the likelihood that these values could be observed and find our posterior distribution.

Remember, $p(H|D) \propto p(D|H) * p(H)$ OR $Posterior \ \propto Likelihood \ * Prior$

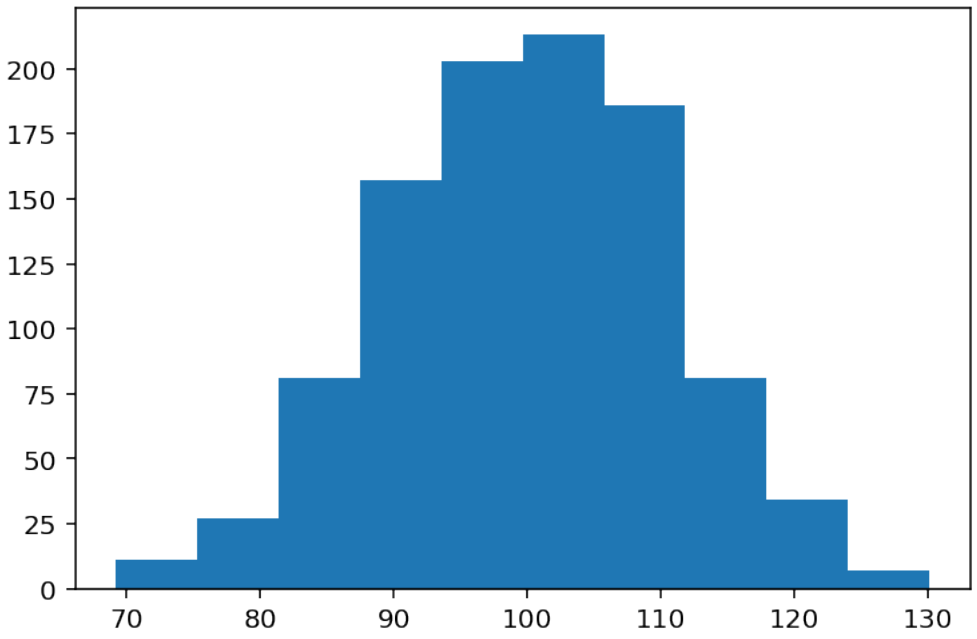
[14]:
```
p_posterior = [p_prior[i] * p_observed[i] for i in range(len(p_prior))]
p_posterior = p_posterior / np.sum(p_posterior)
plt.plot(params, p_posterior);
```



University of Michigan Student IQs We'll do another example where we have some prior belief about the IQ of University of Michigan students.

For our prior distribution, we'll have a normal distribution with a mean IQ of 100 and a standard deviation of 10.

[15]:
```
prior_distribution = np.random.normal(100, 10, 1000)
plt.hist(prior_distribution);
```



Now, let's say we are collecting some observations of student IQs which takes the shape of a normal distribution with mean 115 and standard deviation of 7.5 and want to construct our posterior distribution.

In order to do this, we update our prior by calculating the mean and variance after each observation.

The equations for our updated prior mean and variance are:

$$Updated\ Prior\ Mean = \frac{\sigma^2_{observed}\mu + \sigma^2_{prior}x}{\sigma^2_{observed} + \sigma^2_{prior}}$$

$$Updated\ Prior\ Variance = \frac{\sigma^2_{observed}\sigma^2_{prior}}{\sigma^2_{observed} + \sigma^2_{prior}}$$

[16]:
```
np.random.seed(5)
observed_distribution = np.random.normal(115, 10, 1000)
mu = [100] * 1000
sigma = [10] * 1000

# prior mean formula
mu[0] = (10 ** 2 * observed_distribution[0] + (10 ** 2) * 100) / (10 ** 2 + 10␣
 ↪** 2)

# prior variance formula
sigma[0] = (10 ** 2 * 10 ** 2) / (10 ** 2 + 10 ** 2)
```

```
for i in range(1000):
    if i == 999:
        break

    # prior mean formula
    mu[i + 1] = (sigma[i] * observed_distribution[i + 1] + (10 ** 2) * mu[i]) /␣
 ↪(sigma[i] + 10 ** 2)

    # prior variance formula
    sigma[i + 1] = (sigma[i] * 10 ** 2) / (sigma[i] + 10 ** 2)


posterior_distributions = [[]] * 20

for i in range(20):
    posterior_distributions[i] = np.random.normal(mu[i], sigma[i], 1000)

plt.hist(prior_distribution)
plt.hist(observed_distribution, alpha=0.75)
plt.hist(posterior_distributions[5], alpha=0.5);
```