Fundamentals

Thursday, January 14, 2016 11:20 PM

- Cross-section study: Captures data from a point in time of some group
- Longitudinal study: Captures data over a period of time
- Sample drawn from population
  - Those who participate in a survey: Respondents
  - Meant to be representative(every member of target population has an equal chance of participating
- Oversampling and undersampling: Distorting the classes from which you sample from
- Codebook: Documentation of a study
- Stata dct files are used to describe data files(txt) - LOOK MORE INTO THE CODE
- Data frame: pandas object which contains a row for each record, and a column for each variable
  - Also contains the variable names and their types
  - To access columns of some dataframe x, use x.columns
    - Returns and index of the column names
  - To call a column within a dataframe, use x.['columnname'] or use dot notation - x.columname
    - Returns a series - similar to a list plus some extra features
      - Indices with the accompanying values, its name, and type
      - Indices can be any orderable type, and the values can be any type
    - Access the row elements within the column by indexing through the column like you would a list
- In the NSFG data set:
  - `caseid` is the integer ID of the respondent.
  - `prglngth` is the integer duration of the pregnancy in weeks.
  - `outcome` is an integer code for the outcome of the pregnancy. The code 1 indicates a live birth.
  - `pregordr` is a pregnancy serial number; for example, the code for a respondent's first pregnancy is 1, for the second pregnancy is 2, and so on.
  - `birthord` is a serial number for live births; the code for a respondent's first child is 1, and so on. For outcomes other than live birth, this field is blank.
  - `birthwgt_lb` and `birthwgt_oz` contain the pounds and ounces parts of the birth weight of the baby.
  - `agepreg` is the mother's age at the end of the pregnancy.
  - `finalwgt` is the statistical weight associated with the respondent. It is a floating-point value that indicates the number of people in the U.S. population this respondent represents.
- Recodes: variables that are not part of the raw data but are calculated using the data - can be useful in instances where not all variables are present - often wise to use recodes instead of raw data
- Data cleaning: Checking the validity of the data, whether it can be processed(formats, err vals, etc.),transforming it
- Can replace values in a DataFrame or a series with x.replace(to_replace,value,inplace=True)
- To add a column, you should use dictionary syntax. x['newcol'] = vals
- Series class has method value_counts() which returns a series of values and how frequent they are(.sort_index() sorts the series by the index number instead of the amount int)

- Validation: Useful to check summary statistics of dataset to the already determined summary statistics in the codebook to be sure data cleaning went well
- Can use conditional operators to return a boolean series of the values that match the condition
    - Can then use the boolean series to index the original series(to replace values and such)
- Use defaultdict(defaultfactory) from collections module to build a dictionary that instantiates values by calling the defaultfactory function when the key isn't found
- .iteritems() creates an iterator of a dictionary that spits out tuples of the key/value pair
    - Works on series as well
- The .values attribute of a series returns the numpy array

Ch2. Distributions

Monday, January 18, 2016 10:33 PM

- Distribution: Describes how frequent a value of a variable appears
- Histogram: Visualizes the distribution
- Mapping frequencies with python:
  - Use dict and get() method or
  - use the Counter(subclass of dictionary) from collections module
- Histograms are a good way to explore variables to get a feel for the data
- Normal distribution/gaussian distribution: bell curve
  - Ends are tails
- Uniform: constant distribution
- Outlier: Extreme measurement of rare event
- Summary statistics:
  - Central tendency: do the values cluster around a particular point?
  - Modes: More than one cluster?
  - Spread: How much variability exists
  - Tails: The quickness with which probabilities drop away from the modes
  - Outliers: Extreme values far from nodes
- Variance: Describes the spread
  - $S^2 = \frac{1}{n}\sum_i (x_i - \bar{x})^2$
  - Square root of the variance is the standard deviation
- Effect size: Such as a difference between means
- Cohen's *d*: Compares the means to the variability between groups:
  - $d = \frac{\bar{x_1} - \bar{x_2}}{s}$
    - $s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$
- Clinical significance: Whether a statistical difference affects decisions

Ch3. Probability Mass Functions

Tuesday, January 26, 2016 3:18 PM

- Probability mass function: maps each value to its probability
- Probability: Frequency as fraction of its sample size
  - frequencies -> probabilities = normalization(dividing by sample size)
- Use thinkstats2.Pmf() to create a pmf
  - To find the probability of a value x, use pmf.Prob(x)
  - You may modify(increment) a pmf with pmf.Incr(x, val)
    - Or with(multiply by factor) pmf.Mult(x, val)
    - Won't be normalized until you call pmf.Normalize()
      - pmf.Total() will amount to 1.0
- PMFs useful to compare relative measures(given they are normalized)
- To compare data, it's useful to narrow in on some range of interest and then find the differences
- Sometimes a biased PMF serves as a better measure
  - Asking about class sizes:
    - From class data(unbiased)
      - To get from biased data: Divide each probability by the # of students
    - From perspective of students: Mean is higher(biased)
      - To get from unbiased data: Multiply each probability by the # of students in the class, then normalize again

To compute the mean, given a PMF:	$\bar{x} = \sum_i p_i \, x_i$
To compute variance, given the PMF:	$S^2 = \sum_i p_i \, (x_i - \bar{x})^2$

- Pandas - working with row selection
  - To create a data frame(out of an array): pandas.DataFrame(array)
    - Rows are numbered starting from 0, as are columns
    - You can provide column names
      - df.columns = ['a','b']
    - You can provide row names(set is called index, each row is called a label)
      - df.index = ['a','b']
  - Simple indexing a dataframe uses the column names to select a series(a column)
    - df['A']
  - To index by a row, use loc attribute
    - df.loc['a']
    - Also use loc for labels(returns a series)
      - df.loc['a','c']
        - Returns dataframe
    - Use slice to return a range of rows
      - df['a':'c']
      - or with int df[0:3]
  - If you know the integer position of the row, you can use:
    - df.iloc[0]
  -

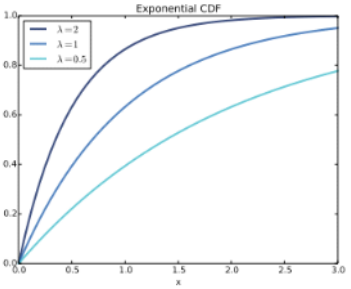
Ch4. Cumulative Distribution Functions

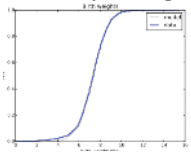
Thursday, January 28, 2016 6:43 PM

- PMFs become not as revealing when there are many values - noise may interfere - and it may be hard to see important data and patterns(like, which distribution has the higher mean)
    - Binning the data - collecting ranges of values in separate bins - is an alternative, but it can be tricky, and may obfuscate useful info
- Percentile rank: The percentage of scores a value is greater than
    - Percentile: The value the percentile rank is connected to
- Cumulative distribution function(CDF): A function that maps from a value to its percentile rank
    - CDF(x) computes the fraction of values in the distribution less than or equal to x
    - Is a step function
    - Median: 50th percentile(describes central tendency)
    - Interquartile range(IQR): Measure of the spread of the distribution
        - Difference between the 75th and 25th percentiles
    - Quantiles: statistics which represent equally spaced points

Ch5. Modeling Distributions

Friday, January 29, 2016 2:27 PM

- Empirical distributions: based on empirical observations(finite samples)
- Analytic distributions: characterized by a CDF that is a mathematical function.
  - Used to model empirical distributions - simplifications
- Exponential distributions:
  - $\mathrm{CDF}(x) = 1 - e^{-\lambda x}$
  - 

  - Used to model interarrival times(time between events)
    - If events likely to occur at any time, the distribution looks exponential
  - If you plot the complementary CDF(1-CDF(x), which tells you how often a variable is above a particular level), you should expect a straight line with slope -λ
  - Mean of an exponential distribution is 1/λ
- Gaussian/Normal distribution:
  - Standard normal distribution:
    - Characterized by μ(mean) = 0 and σ(standard deviation) = 1
    - Defined by an integral without a closed form solution
      - Is implemented within scipy.stats.norm
        - Can use scipy.stats.norm.cdf(x) to find the percentile rank of x
    - Recognized by its sigmoid shape:
      - 
- Normal probability plot:
  - Used to test whether the data from a dataset has a normal distribution
  - How to:
    1) Sort the values in the a sample.
    2) From a standard normal distribution, generate a random sample with the same size as the data sample and sort it
    3) Plot the sorted values vs the standard values
       - If the resulting graph is a straight line, then the dataset sample is normal
- Lognormal distribution:
  - A dataset is lognormal if the logarithms of the values define a normal distribution
    - $CDF_{lognormal}(x) = CDF_{normal}(\log x)$
  - To test lognormality, you can plot its cdf with the log of the values against that of normal cdf
  - To visualize lognormality easier, use the normal probability plot with the log of the data
- The Pareto distribution:
  - Originally used to describe wealth distribution, now applied to natural and social sciences alike

- $CDF(x) = 1 - \left(\frac{x}{x_m}\right)^{-\alpha}$
    - x_m as minimum value
- Generating random values
    - Take the inverse of the CDF and choose a p from a uniform distribution(0-1), then use x = ICDF(p)
- Why model?
    - Useful to compress large amounts of data and define it as a few parameters
    - If we attach data to a model, sometimes we know why the model has a particular form, and so we can use it in an explanatory light of the data