        - JointPlot
            - Does a histogram for each variable and scatterplot for both
            - sns.jointplot(data=df, x=col1, y=col2)
        - Showing the graph
            - use matplotlib show()
        - Formatting and Looks
            - color=
                - line color
            - linewidth=
                - Change line width
- Tidy dataset
    - Every column is a variable and every row is an observation

Regression

- Two types
    - Univariate
    - Multivariate
- Definition
    - Predicting a continuous variable
    - Modeling the patterns in the data to try to predict that variable
    - A form of supervised machine learning
    - The independent variables are called the dimensions
        - univariate has two dimensions (x and y)
        - multivariate will have more than two
- Goal
    - A **function** that mimics/models a relationship between one or more independent/feature variables
        - i.e., the best choice of values for the parameters
    - E.g., $y = c_0 + c_1x_1 + c_2x_2 + \ldots + c_nx_n$
- Assumptions
    - The dependent variable is **continuous** and a linear function to some approximation of independent variables
- Parameters
    - the intercept and coefficients
- ## Univariate
    - Function
        - $y = mx + b$

- b is y-intercept
- m is slope (rise / run)

- Also as, $y_i = \alpha x_i + \beta + \varepsilon_i$
  - This equation will get us back all our data points. There is an ε for each data point. It is the error.

- Goal
  - Minimize error between actual values and estimated values
- Univariate is the easiest way to demonstrate what regression does but it is rarely used in the field

- Multivariate
  - Multiple independent input/feature variables
  - Produces one dependent variable
  - Visualization
    - In two dimensions (x and y), it's a line
    - In three dimensions (x1, x2, and y), it's a plane
  - $y = c_0 + c_1x_1 + \ldots + c_nx_n$
    - x are each variables
    - c are the constants that the regression algorithm finds to minimize error
      - they can be thought of as the weights for each variable
- Polynomial Regression
  - Definition
    - models non-linear relationship of the independent and dependent variables
    - considered a linear model
      - the x^2 is considered a feature
- Evaluation of regression models
  - Residual
    - This is the error. The distance between the actual/observed value and the predicted value
  - Sum of squared errors (SSE)
    - a.k.a. residual sum of squares, sum of squared residuals
    - a small SSE indicates a tight fit/good model
    - Squared error is $(actual\ value\ -\ predicted\ value)^2$
      - Square it to account for positives and negatives
    - In Excel, we are going to look at the Total Sum of Squares.

        - If we start changing variables or the y-intercept, we can compare the sum of squares of different regressions to see which is better
        - Explained (Regression Sum of Squares)
            - How much of the variation in the dependent variable the model explains
            - sum of (predicted value - y-mean) squared
        - Residual Sum of Squares
            - How much of the dependent variable's variation the model did not explain
            - sum of (actual value - predicted value) squared
            - smaller mean better fitting model
            - larger means poorer fitting model
            - 0 is a perfect fit
        - Total Sum of Squares
            - Explained + Residual Sum of Squares
    - Mean Squared Error (MSE)
        - SSE / n
        - SSE will have a much larger number if we have big outliers
        - can be compared among different models to see which is better
    - R-squared
        - fraction of total variation in the dependent variable the model captures
        - Coefficient of determination
        - Does your error mean something. If it's high, we are capturing good information. If it's low, we need better information
        - can be compared among different models to see which is better
        - R2 x 100 = amount of variation in y explained by x
    - F-statistic
        - Less than 0.05 is good
        - Greater than .05 means we need to use other independent variables
        - can be compared among different models to see which is better
    - P-values
        - All p-values should be below 0.05.
        - Remove a p-value if it's greater than 0.05
        - can be compared among different models to see which is better
    - Confidence interval
        - the percentage of time the indicated parameter will fall within the range

        - narrow means more confidence; wide means less confidence
        - This is the shaded portion in Seaborn
        - It's like the funnel in a hurricane prediction map
    - Residual plot
        - Will tell us how good our model is
        - We want to see randomness. If we don't, we need to try something else (e.g., polynomial regression, new y-intercept, or adding a new variable)
        - can be compared among different models to see which is better
- ## Regression algorithms
    - Ordinary Least Squares
        - Minimizes sum of squared errors
        - Pros
            - fast
            - not complex
            - useful when there's not a lot of data
            - simple to explain to stakeholders
        - Cons
            - Very sensitive to outliers. Might want to remove them when modeling the data or replace them with something else
        - Steps
            - 1. Compute parameters
            - 2. Predict $\hat{y}$ for each observation
            - 3. Measure difference $\hat{y} - y$ (residual)
            - 4. Sum of squared difference -> error
    - Stepwise Regression
        - Will do a regression for each variable and rank them in terms of the amount of information we get from them
            - x1 might get us .5 of information
            - x2 might get us up to .6
            - x3 might get us up to .7
            - and we might stop here if this is good enough if x4, x5, etc don't get us much farther
        - Some criteria are laid down at the start for use in ranking variables
        - Options
            - forward selection
            - backward elimination
        - Pros

        - Faster
        - Less prone to overfitting
        - Transparent process that gives insights
      - Cons
        - May include only one of two highly correlated independent variables
        - Risk of overfitting
        - Can't account for specialized knowledge of DS
        - "It is important to note that charting the individual predictors against the response is often misleading because these do not account for other predictors in the model."
- Training and test
  - if we have a small amount of data, we may want a 80-20. if we have a lot of data, we could do 60-40 or 70-30
- ## Regression in Python
  - Example
    - 1. Plan
      - Goals
        - predict final grade
        - stakeholders are students and administration
      - Deliverables
        - Presentation
      - Data
        - Data needed
          - grades (current and previous)
          - attendance
          - participation
        - Data I have: exam 1 and 2
        - target variable: final grade
  - Steps
    - 1. Create a linear object model
    - 2. Fit that object to training data -> function with coefficients
    - 3. Transform/predict on training data -> predictions
    - 4. Evaluate results -> SSE
    - 5. Transform/predict on test data -> predictions
    - 6. Evaluate results
  - Steps in more detail
    - 1. Prepare the Environment

        - imports
            - pandas
            - scipy.stats
            - matplotlib.pyplot
            - seaborn
            - statsmodels.api
            - sklearn.model_selection
            - sklearn.linear_model
            - sklearn.metrics
    - 2. Pandas: read a local csv
        - use pd.read_csv()
    - 3. Pandas: sample and summarize
        - df.shape
        - df.head()
        - df.info()
        - df.describe()
        - stats.iqr()
        - range
    - 4. Pandas: ensure no null values
        - df.isnull().sum()
        - df.columns[df.isnull().any()]
    - 5. Distribution, Skewness, Normalization, Standardization
        - histogram of independent variables
            - melt dataframe and use seaborn's FacetGrid
                - OR use matplotlib and subplots
        - boxplot of independent variables
            - sns.boxplot()
    - 6. SKlearn: split into test/train
        - 1. drop all columns (including target variable) except those holding independent variables
        - 2. extract dataframe of just target variable
        - 3. split the data using model_selection.train_test_split
            - set a random state other than 0 to ensure reproducibility; 0 means random everytime
        - 4. ensure split was done correctly
            - # of rows in indep var and dep var in training and test data sets are equal
            - # of cols in training and test data sets are the same
        - 5. concatenate into two dataframes: train and test

- 7. Matplotlib & Seaborn: Explore
    - box plts, heatmaps, histograms, density plots, feature or correlation plots
    - Use
        - sns.jointplot()
            - annotate with Pearson's r
        - sns.PairGrid()
        - sns.heatmap()
            - compare independent variables with each other and with the dependent variable to find correlations
- 8. Scipy: Pearson's correlation
    - create a dictionary of independent var: pearson's r
        - use stats.pearsonr()
- 9. Statsmodels: Feature Selection
    - Ordinary Least Squares
        - use sm.OLS()
- 10. Scikit-Learn: Fit Linear Regression Models, In-Sample Predictions
    - 1. Create linear regression object
        - LinearRegression()
    - 2. Fit/Train the model
        - model parameters are learned from the training data
        - lm.fit() on the independent variables
        - write the regression function
    - 3. In-sample prediction
        - lm.predict() on the independent variable(s) in the training data
- 11. Scikit-Learn: In-Sample Evaluations
    - 1. Train/in-sample evaluations
        - mean squared error
            - mean_squared_error()
        - r-squared
            - r2_score
    - 2. Compare models
        - melt the dataframe and do an sns.relplot()
        - draw a dotted line representing the perfect prediction
            - not sure how this is possible

        - 12. Scikit-Learn: Make any changes needed & repeat 9-11 as needed
        - 13. Scikit-Learn: Out-of-sample predictions using best model
            - 1. Predict dependent variable
                - lm.predict() on the test data independent variables
            - 2. Evaluate performance
                - mean_squared_error
                - r2_score
            - 3. Plot Residuals
                - plt. scatter()
                - plt.hlines() to draw a horizontal line along x-axis
    - Reproducible research
        - set the random seed to any number we want; just be consistent so the results are reproducible
    - Meshing training and test data
        - Maggie does this and sees it in practice, instead of keeping four separate variables (X_train, y_train, X_test, y_test)
        - pd.concat([X_train, y_train], axis=1)
    - Verifying data preparation
        - the number of rows in both the x and y training data set are equal
        - the number of rows in both the x and y testing data set are equal
        - the number of columns in the training and test data sets are the same
        - the training data set is 80% of the original data, the test set is 20%
    - SciPy
        - Pearson-R
            - pearsonr(series1, series2)
    - Scikit-Learn Regression
        - Create linear regression object
        - Fit/train the model
        - Look at intercept and coefficients
        - Use model to make predictions (lm.predict)
        - Evaluate performance of model
            - calculate mean squared error and R2
                - compare MSE and R2 to previous models

Modeling

- Overfitting