## Model Development Programming Exercise

Before beginning, be sure to create a clean new directory that will hold all the output you produce as part of this exercise (CSV files, images, etc.).

- **Task 1**: **Import data.**
  - Read in all the historical data. Historical data are included in CSV files in folders named contracts, fuelPrices, plantParameters, and powerPrices.
  - For fuel prices and power prices, combine the data from individual files into a single data frame so that you have four data frames corresponding to the four types of input data.

### Power Price Statistics

Power prices (provided in $/Megawatt-hour, or $/MWh) vary by location on the grid due to supply/transmission availability, regional demand, and many other factors.

- **Task 2**: **Calculate basic descriptive statistics**
  - Compute the mean, min, max, and standard deviations of the hourly prices for each settlement point and year-month in the historical dataset (48 year-months: January 2016 through December 2019). The resulting data frame should have 48 rows for each settlement point.
- **Task 3**: **Calculate volatility**
  - Volatility is defined as the standard deviation in the log returns of hourly prices and measures how much prices change from one hour to the next. Calculate average volatility for each year-month-settlement point combination as well.
- **Task 4**: **Write the results to file**
  - Write the computed monthly statistics to file as a CSV named **MonthlyPowerPriceStatistics.csv**
  - The CSV file you write should have columns with the following names: SettlementPoint, Year, Month, Mean, Min, Max, SD, Volatility.

### Contract Valuation

The Contracts.csv input file contains data for two basic types of financial instruments, swaps and options. Swaps simply involve exchanging a fixed strike price for a variable asset price. Options give the owner the right but not the obligation to exchange the strike price for the asset price.

- **Task 5: Expand the contracts across relevant time periods.**
  - The input data contains both daily and hourly contracts, identified by the text in the Granularity column.
  - The input data also contain StartDate and EndDate columns. These represent the time frame over which each contract is active. For example:
    - A contract with daily granularity, StartDate = 2017-01-01 and EndDate = 2017-01-31 represents 31 individual contracts, one for each day in the time frame, including both bounds.
    - A contract with hourly granularity, StartDate = 2018-07-01, and EndDate = 2018-07-07 represents 168 individual contracts, one for each hour in the 7-day time frame, including both bounds.
  - Separate the contracts into two separate data frames for hourly and daily contracts.
  - Expand each contract into multiple rows across the individual time periods (either days or hours) within the StartDate – EndDate time frame.

- Note that the Volume column in the original input data represents the contract volume for *each* of the individual periods with the time frame, so it does not need to be adjusted for the number of relevant time periods.

- **Task 6: Join relevant prices**
  - The PriceName column identifies the relevant asset price to use for the calculations.
  - Join the corresponding prices (fuel prices for daily contracts and power prices for hourly contracts) to the resulting data frames from Task 5.
- **Task 7: Calculate payoffs**
  - The formulas for payoffs are:
    - Swaps: (AssetPrice – StrikePrice) x Volume
    - Options: [MAX(AssetPrice – StrikePrice, 0) - Premium] x Volume
- **Task 8: Calculate aggregated payoffs**
  - For each contract-year-month combination, sum up the payoffs into a column named TotalPayoff
- **Task 9: Write the results to file**
  - Write the resulting data frame to a CSV named **MonthlyContractPayoffs.csv**
  - The output should have 41 total rows, and columns named ContractName, Year, Month, TotalPayoff

## Plant Dispatch Modeling

Power plants are dispatched based on market prices to maximize profit. In general, power plants generate power whenever the market power price is greater than the cost to generate power, and turn off whenever the market power price is less than the generation cost.

- **Task 10**: **Calculate the hourly running cost of each power plant included in the Plant_Parameters.csv input file.**
  - The formula for running cost (in $/MWh) is:
    - RunningCost = ((FuelPrice + FuelTransportationCost) x HeatRate) + VOM
  - Note that the FuelTransportationCost, HeatRate, and VOM parameters are provided in Plant_Parameters and are assumed to be constant for each plant-year-month. Fuel prices, however, change on a daily basis. Names of the corresponding fuel prices for each plant are provided in the FuelPriceName column, and daily prices must be joined from the data provided in the 'fuelPrices' input folder.
  - The result of this calculation should be a data frame that contains data for 2 plants x 2 years (2017 and 2018) x 365 days per year = 1460 total rows
- **Task 11: Join the hourly power prices.**
  - Names of the corresponding power prices are provided in the PowerPriceName column of the Plant_Parameters input file.
  - Note that the various plant parameters, fuel prices, and running costs calculated above should be constant for all 24 hours in each day, but the power price may vary on an hourly basis.
  - The results of this calculation should be a data frame that contains data for 2 plants x 2 years x 365 days per year x 24 hours per day = 35040 rows
- **Task 12**: **Identify hours in which the power plant should be on.**

  - Create a new column called 'Generation'. This variable should be equal to the plant's capacity whenever the power price > running cost, and 0 whenever power price < running cost.
  - Create a new column called 'RunningMargin'. The formula is:
    - RunningMargin = (PowerPrice – RunningCost) * Generation
- **Task 13: Account for start costs**
  - Power plants incur a fixed startup cost whenever they turn on. Plants must be able to generate enough margin between the hour in which the plant starts and the hour in which the plant shuts down to overcome this initial start cost.
  - For example, a plant may be 'in the money' at 6am and 7am on a given day, with a running margin of $1,000 in each hour, and out of the money in all other hours. However, if the start cost for the plant in $10,000, then the plant should not turn on for only those two hours.
  - For each continuous block of generating hours following a start, calculate the total running margin, and compare it to the plant's start cost. For continuous blocks when the plant does not make enough margin to overcome the start cost, turn the plant off by setting the plant's generation to zero. Do not change the Generation column calculated in Task 12, but instead add these updated values that account for start costs to your existing data frame in a column ambiguously named 'Generation2'.
  - (For now, ignore the potential for the plant to stay on during periods of negative running margin. Assume that the plant must shut down during all hours in which the Generation value calculated in Task 12 is 0.)
- **Task 14: Write the results to file**
  - Write the resulting data frame to a CSV named **PowerPlantDispatch.csv**

## Bonus: Open-Ended Analysis

If you have crushed all of these tasks, with what time you have left, write code to analyze the data further in some way of your choosing. That is, tell us and/or show us something interesting about the data. This is a chance for you to show off. Options include:

- Plots visualizing any of the results you generated above
- Statistical models illustrating relationships between underlying prices and asset value
- Machine learning models or statistical to try to simulate time series behavior
- Monte Carlo simulation
- Anything else you can think of

You may provide narrative around your analysis within comments in your code; we will read the comments and the associated code for your analysis upon reviewing your response. Save any relevant output to your output folder.

Again, this question is entirely open-ended; use your intellectual curiosity and analytical prowess to guide you to some interesting data-driven insights.