**1. Библиотека / данные**

Построение графиков, а также статистическая или интерактивная визуализация - это одни из важнейших задач анализа данных. Они могут быть частью процесса исследования, например, применяться при выявлении выбросов, определения необходимых преобразований данных или поиска идей для построения моделей. Как обычно нам понадобятся библиотеки pandas и NumPy. Дополнительно загрузим из библиотеки matplotlib модуль pyplot. Magic-команда matplotlib inline необходима для вывода графика на экран Jupiter Notebook.

Мы использовали параметр parse\_dates метода read\_csv() для корректной установки формата столбца. Затем благодаря методу set\_index столбец был назначен индексом строк объекта DataFrame.

**2. Настройка**

Сама по себе библиотека pandas не выполняет визуализацию данных. Для выполнения этой задачи библиотека pandas предлагает тесную интеграцию с другими надежными библиотеками визуализации, которые является частью экосистемы Python. Наиболее часто используемой библиотекой для визуализации является библиотека matplotlib, поэтому в примерах, приведенных в этом Notebook будет использоваться библиотека matplotlib. Однако есть и другие возможные библиотеки, с которыми вы можете поработать самостоятельно.

**2.1 пример**

Мы рассмотрим настройку параметров построения графиков на примере визуализации временных рядов из объекта DataFrame. Визуализировать временной ряд в библиотеке pandas очень просто - достаточно применить метод plot() объекта DataFrame или Series, в который записан временной ряд.

Метод plot () объектов pandas является функцией-оберткой вокруг одноимённой функции библиотеки matplotlib. Данный метод делает построение графиков в pandas очень простой процедурой, поскольку программный код, лежащий в его основе, позволяет строить самые различные типы визуализации данных.

В этом примере метод plot() определил, что объект Series проиндексирован по датам, поэтому ось X должна быть представлена в виде дат. Кроме того, в методе plot() задан цвет, который будет использоваться по умолчанию для отображения данных. Если объект DataFrame состоит из нескольких столбцов, то метод plot() добавит несколько элементов в легенду и подберёт для каждой линии свой цвет.

Из полученного графика сложно выявить какие-то зависимости, так как масштаб рассматриваемых цен акций разный. Давайте проведем нормализацию значений: вычислим среднее значение и разделим на стандартное отклонение. Из графика видно, что значения цен акций высоко скоррелированы. Также можно определить, когда были минимумы или максимумы цен на акции.

**2.2 размер**

Встроенный метод plot() предлагает множество параметров, которые вы можете использовать для изменения содержимого графика. Давайте рассмотрим некоторые часто используемые настройки графиков. Размер графика регулируется параметром figsize. Передаём через данный параметр ширину и высоту в виде «питоновского» кортежа.

**2.3 заголовок / подписи**

Заголовок графика можно задать с помощью параметра title. Метки осей задаются с помощью функции xlabel и ylabel сразу после вызова метода plot().

**2.4 легенда**

Чтобы изменить текст, который используется в легенде для идентификации каждой серии данных, сохраним в переменную ax объект, который будет возвращён методом plot(), и затем воспользуемся методом legend(). Данный объект можно использовать для изменения различных настроек графика непосредственно перед его построением.

Местоположение легенды можно задать с помощью параметра loc метода legend(). По умолчанию библиотека pandas устанавливает данный параметр, равный ‘best’, что дает инструкцию библиотеке matplotlib исследовать график и определить наилучшее расположение легенды. Однако, вы можете указать любой вариант месторасположения: можно использовать как строковое значение, так и числовой код. Вывод легенды можно отключить с помощью значения параметра legend = False.

**2.5 цвет / стиль / толщина / маркер**

Библиотека pandas автоматически задает цвета для каждой серии при построении графика. Чтобы задать свои собственные цвета, просто передайте кодовое обозначение цветов параметру style. Библиотека matplotlib предлагает ряд встроенных односимвольных кодов, задающих различные цвета линий. Кроме того, можно задать цвет с помощью шестнадцатеричного RGB-кода.

Стили линий можно задать с помощью кода. Их можно использовать в сочетании с кодами цветов, задав сразу после кодового обозначения цвета.

Толщину линий можно задать с помощью параметра lw. Можно настроить толщину сразу нескольких строк, передав этому параметру список значений ширины или одно значение ширины, которое будет применено ко всем линиям.

Маркеры точек можно указать с помощью специальных обозначений в программном коде.

**3. Графики**

Теперь, когда вы знаете, как настраивать параметры графиков, мы рассмотрим построение различных диаграмм, которые могут пригодиться для визуализации статистической информации.

**3.1 bar**

Чтобы построить не линейный график, а столбчатую диаграмму, достаточно передать параметр kind в метод plot().

Для построения составной столбчатой диаграммы по объекту DataFrame нужно задать параметр stacked = True. Тогда столбики, соответствующие значениям в каждой строке, будут приставлены друг к другу. Вертикальную ориентацию диаграммы, которая используется по умолчанию, можно сменить на горизонтальную с помощью значение параметра kind = ‘barh’.

**3.2 hist**

Гистограмма, с которой все мы хорошо знакомы, - это разновидность столбчатой диаграммы, показывающая дискретизированное представление частоты. Результаты измерений распределяются по дискретным интервалам равной ширины, а на гистограмме отображается количество точек в каждом интервале.

Параметр bins показывает количество интервалов, на которое делится непрерывная величина. по умолчанию данный параметр равен 10.

Есть ли в объекте DataFrame есть несколько числовых столбцов, то метод hist(), вызванный у этого объекта DataFrame автоматически сгенерирует несколько гистограмм: по одной для каждого столбца.

Чтобы наложить несколько гистограмм друг на друга в рамках одного и того же рисунка и тем самым визуализировать разницу распределений, несколько раз вызовем функцию plt.hist().

С гистограммой тесно связан график плотности, который строится на основе оценки непрерывного распределения вероятности по результатам измерений. Обычно стремятся аппроксимировать это распределение комбинацией ядер, то есть более простых распределений, например, нормального, Гаусса, поэтому графики плотности еще называют графиками ядерной оценки плотности (kernel density estimate – kde).

Функция plot с параметром kind = ‘kde’ строит график плотности, применяя стандартный метод комбинирования нормальных распределений.

Гистограмма в нормированном виде, показывающая дискретизированную плотность, и поверх неё график ядерной оценки плотности часто рисуются вместе.

**3.3 box**

Boxplot, часто называемый ящик с усами, - это график, который используется в описательной статистике, компактно изображающий одномерные статистики распределения переменный. Такой вид диаграммы в удобном формате показывает медиану, 25-процентный квантиль, 75-процентный квантиль (оба этих квантиля называют квартилями), минимальное значение и максимальное значение, а также выбросы. Расстояние между различными частями ящика позволяет определить степень разброса асимметрии данных и выявить выбросы.

**3.4 scatter**

Диаграмма рассеивания - это полезный способ исследования соотношений между двумя одномерными рядами данных. Диаграмму рассеивания можно создать на основе объекта DataFrame с помощью метода plot(), при этом указав значения параметра kind = ‘scatter’, а также в столбце x и y исходного объекта DataFrame.

В разведочном анализе данных полезно видеть все диаграммы рассеивания для группы переменных. Это называется матрица диаграмм рассеивания. Построение такого графика с нуля - довольно утомительное занятие, поэтому в бибилиотеке pandas имеется функция scatter\_matrix() для построения матрицы на основе объекта DataFrame. Она поддерживает также размещение гистограмм или графиков плотности для каждой переменной вдоль диагонали.

**3.5 heat map**

Heat map - это графическое представление данных, при котором значения внутри матрицы представлены цветами. Это эффективный инструмент, который позволяет визуализировать значения, получаемые на пересечении двух переменных.

Наиболее распространенный сценарий - это визуализация матрицы корреляций. Каждый квадрат показывает значение корреляции, выраженное цветом.