Архитектурный документ. Ядро PySATL

Михаил Михайлов, Леонид Елкин

14 ноября 2025 г.

Оглавление

Раздел 1

Введение

1.1. Назначение системы

Вычислительное ядро проекта PySATL предназначено для представления и обработки вероятностных распределений в программной форме. Ядро предоставляет средства для задания распределений и их семейств, выполнения операций над ними и построения более сложных структур путём функциональных и алгебраических преобразований.

Система поддерживает задание как конкретных распределений с определёнными параметрами, так и абстрактных семейств, из которых могут быть получены конкретные экземпляры. Кроме того, предусмотрена возможность определения пользовательских распределений и преобразований, расширяющих базовые возможности.

Ядро служит универсальной основой для статистических и вероятностных вычислений в рамках проекта PySATL и может использоваться другими подсистемами при построении моделей.

1.2. Область применимости

Вычислительное ядро `core` используется во всех подсистемах проекта PySATL, где требуется работа с распределениями вероятностей. Оно предназначено как для непосредственного вычисления характеристик распределений (например, плотности, функции распределения, квантилей), так и для построения и трансформации более сложных моделей на их основе. Оно может быть использовано:

- при определении конкретных распределений, используемых в анализе данных;
- для задания пользовательских распределений, комбинации распределений и создания новых семейств;
- при трансформации распределений через функциональные отображения;
- в задачах символьной или численной обработки распределений.

Вне проекта PySATL ядро может быть применимо в любых системах, где необходима гибкая и расширяемая работа с вероятностными распределениями, особенно в контексте численных симуляций, статистического моделирования и прикладного машинного обучения.

1.3. Общие сведения о системе

Ядро представляет собой модульную систему, реализованную на языке Python, предназначенную для работы с вероятностными распределениями, их семействами и преобразованиями. Архитектура системы построена на разделении функциональности между независимыми компонентами, каждый из которых отвечает за определённый класс задач.

Компоненты взаимодействуют друг с другом через чётко определённые интерфейсы. Такая организация позволяет изолировать ответственность отдельных частей системы, обеспечивать гибкость при расширении функциональности и облегчать сопровождение кода.

Некоторые интерфейсы, предоставляемые одним модулем, используются другими модулями для построения более сложных вычислений или абстракций. Это позволяет комбинировать базовые элементы в составные структуры и формировать цепочки преобразований.

Проект спроектирован таким образом, чтобы допускать расширение без модификации существующих компонентов, в соответствии с принципами модульности и открытости/закрытости. Это обеспечивает стабильную основу для развития ядра и его интеграции с другими частями проекта PySATL.

Раздел 2

Глоссарий

Основным понятиям в статистике и стохастическом моделировании является *распределения случайной величины* или, более общо, *распределения случайного объекта* (далее, под случайной величиной понимается любой случайный объект, реализации которого не обязательно суть вещественные числа) [18]. Ниже изложены основные теоретические сведения касающиеся случайных величин, а также задач в которых они возникают, в соответствии с монографиями [37] и [39].

Случайные величины и способы их задания

Для случайной величины ξ, принимающей значения в некотором пространстве $\mathcal{X}$, её распределением называется (см. [39]) вероятностная мера $\mathbb{P}_\xi(\cdot)$ на $\mathcal{X}$, такая что $\mathbb{P}_\xi(A)$ есть вероятность того что реализация ξ попадет в множество $A \subset \mathcal{X}$[1]. Как правило (см. [37]), выделяют следующие виды случайных величин

- Дискретные случайные величины. В этом случае $\mathcal{X}$ представляет собой некоторое дискретное (конечное или счетное) множество. Например число выпадений монеты орлом при нескольких бросках (биномиальное распределение); уровень образования у случайно выбранного человека (категориальное распределение); случайная величина которая принимает одно значение (вырожденное распределение);
- Одномерные непрерывные случайные величины[2]. В этом случае $\mathcal{X} = \mathbb{R}$ или $\mathcal{X} \subset \mathbb{R}$ ненулевой меры. Такие величины используются для описания случайных времен, расстояний и т.д. Согласно [18] наиболее важными представителями являются: равномерное распределение $\mathcal{U}(a; b)$, нормальное распределение $\mathcal{N}(\mu, \sigma^2)$ и экспоненциальное распределение $\mathrm{Exp}(\lambda)$;
- Многомерные непрерывные случайные величины. В этом случае $\mathcal{X} \subseteq \mathbb{R}^d$, ненулевой меры. Во много многомерные случайные величины являются аналогами одномерных непрерывных случайных величин, однако решение стандартных задач, таких как моделирование или вычисление числовых характеристик затруднено из-за проклятия размерности [7].

Отдельное направление статистики работает с данными о направлении (англ. directional data), в связи с этим часто можно также отдельно выделить следующую категорию случайных величин.

- Случайные геометрические примитивы (англ. geometrical random primitives). Примерами таких случайных величин служат случайные углы или случайные матрицы симметрий. Согласно [29], геометрической случайной величиной называется случайная величина принимающая значения на замкнутой и ограниченной поверхности в евклидовом пространстве (более общо—компактном Римановом многообразии).

С точки зрения ПО, работа с распределением, как с вероятностной мерой, является неудобной, так как компьютер не может работать с произвольными множествами. Однако, как правило, с распределением можно связать некоторую функцию, которая полностью определяет распределение. Так, чтобы идентифицировать распределение дискретной случайной величины, достаточно знать *функцию вероятности*, определяемую равенством (pmf).

$$f_\xi(x) = \mathbb{P}_\xi(\{x\}), \text{ т.е. вероятность того что } \xi = x,\ x \in \mathcal{X} \tag{pmf}$$

Если на пространстве возможных значений $\mathcal{X}$ задана некоторая мера μ, плотностью распределения $\mathbb{P}_\xi$ относительно μ называется[3] такая функция $f_\xi(x)\colon \mathcal{X} \to \mathbb{R}$, что выполнено тождество (pdf):

$$\mathbb{P}_\xi(A) = \int_A f_\xi(x)\, d\mu \tag{pdf}$$

В случае если μ это считающая мера, плотность f_ξ определяется равенством (pmf), в случае если $\mu = m_{\mathrm{Leb}}$, говорят просто о плотности непрерывной случайной величины/случайного вектора.

[1] Строго говоря, $\mathcal{X}$ должно быть снабжено некоторой σ-алгеброй $\mathcal{F}$, и $\mathbb{P}_\xi$ должна быть определена только для $A \in \mathcal{F}$. Иначе говоря, тройка $(\mathcal{X}, \mathcal{F}, \mathbb{P}_\xi)$ должна образовывать вероятностное пространство.

[2] Здесь и далее под непрерывными случайными величинами подразумеваются абсолютно-непрерывные случайные величины, т.е. распределение которых имеет плотность относительно меры Лебега

[3] Условия существования плотности описываются теоремой Радона-Никодима, см. например [21]

Несмотря на то что плотность распределения полностью его характеризует, для того чтобы вычислять вероятности $\mathbb{P}_\xi(A)$ необходимо производить интегрирование (или суммирование), поэтому для некоторых задач представление распределения в виде плотности является неудобным. В частности, если ξ — случайная величина (дискретная или непрерывная) принимающие значения из $\mathbb{R}^d$, довольно часто приходится смотреть на вероятность попадания в некоторую ячейку $\langle \mathbf{a}; \mathbf{b} \rangle$. Под ячейкой подразумевается множество:

$$\langle \mathbf{a}; \mathbf{b} \rangle = \left\{ \begin{pmatrix} c_1 \\ \vdots \\ c_d \end{pmatrix} \in \mathbb{R}^d \,\middle|\, a_1 < c_1 \leq b_1, \cdots, a_d < c_d \leq b_d \right\}$$

Для доступа к таким вероятностям эффективнее работать с *функцией распределения случайной величины*, определяемой равенством (cdf).

$$F_\xi(\mathbf{x}) = \mathbb{P}_\xi(\langle -\infty; \mathbf{x} \rangle) \tag{cdf}$$

В этом случае $\mathbb{P}_\xi(\langle a; b \rangle)$ выражается через значения $F_\xi(\cdot)$ с помощью формулы включения-исключения [37].

С понятием функции распределения тесно связано понятие квантильной функции. Для случайной величины ξ со значениями из $\mathbb{R}$, её квантильная функция определяется равенством (ppf) (подробно о различных определениях см. в обзоре [20]).

$$\omega_\xi(p) = \inf_u \{ F_\xi(u) \geq p \} \tag{ppf}$$

Такое определение гарантирует, что случайная величина $\omega_\xi(U), U \sim \mathcal{U}[0; 1]$ имеет такое же распределение как и сама величина ξ. В случае когда $F_\xi(\cdot)$ строго возрастает на всей области определения, квантильная функция является обратной функцией $\omega_\xi(\cdot) = F_\xi^{-1}(\cdot)$. Отдельно следует отметить что существуют обобщения понятия квантильной функции на случай случайных величин со значениями из $\mathbb{R}^d$ [12], однако для их вычисления необходимо решать уравнения в частных производных [10].

Существуют и другие функциональные характеристики распределения, многие из которых приходят из анализа выживаемости [17]. *Функцией выживаемости* называется функция определяемая равенством (sdf).

$$S_\xi(\mathbf{x}) = 1 - F_\xi(\mathbf{x}) \tag{sdf}$$

Для случайных величин со значениями из $\mathbb{R}$, функцией интенсивности отказов и кумулятивной функцией интенсивности отказов называются функции определяемые равенствами (hrdf) и (chdf) соответственно.

$$h_\xi(x) = -\frac{S'_\xi(x)}{S_\xi(x)}; \tag{hrdf}$$

$$H_\xi(x) = -\ln(S_\xi(x)); \tag{chdf}$$

Однако, некоторые распределения, например α-устойчивые распределения [37], не допускают явного задания с помощью плотности или функции распределения, однако допускают задания с помощью так называемых *интегральных преобразований*. Такие распределения все чаще возникают в современных моделях стохастического анализа, (см. например [4]). В случае случайной величины ξ со значениями из $\mathbb{R}$, её

- Характеристической функцией ξ называется преобразование Фурье, определяемое равенством (cf).

$$\varphi_\xi(u) = \int_{\mathbb{R}} \exp(itu) \mathbb{P}_\xi(dt), \quad u \in \mathbb{R} \tag{cf}$$

- Момент-производящей функцией называется преобразование определяемое равенством (mgf).

$$M_\xi(u) = \int_{\mathbb{R}} \exp(tu) \mathbb{P}_\xi(dt), \quad u \in \mathbb{R} \tag{mgf}$$

Характеристическая функция всегда существует и полностью определяет распределение случайной величины [39]. В свою очередь, момент-производящая функция существует не всегда, но в тех случаях когда существует, также однозначно характеризует распределение. В случае когда ξ принимает только неотрицательные значения, определены также преобразование Лапласа и преобразование Меллина, задаваемые равенствами (lt) и (mt) соответственно.

$$\mathcal{L}_\xi(u) = \int_{\mathbb{R}_+} \exp(-tu) \mathbb{P}_\xi(dt), \quad u \in \mathbb{R}_+ \tag{lt}$$

$$\mathcal{M}_\xi(u) = \int_{\mathbb{R}_+} t^u \mathbb{P}_\xi(dt), \quad u \in \mathbb{R}_+ \tag{mt}$$

Эти преобразования также однозначно характеризуют распределение ξ [37], [11]. В случае если ξ многомерная случайная величина, также определяется характеристическая функция (см. [39]), преобразования (mgf), (lt), (mt) в некоторых ситуациях допускают обобщение на многомерный случай, см. например [2].

На рис. 2.1 схематично изображены основные способы задания непрерывных вероятностных распределений, и связь между ними. В связи с тем что в теории многомерных квантилей нет результатов напрямую выражающих квантильные функции через плотности или интегральные преобразования, переходы которые имеют место быть только в одномерном случае, изображены пунктирными стрелками.

Рис. 2.1: Способы задания непрерывных распределений

Замечение. Плотность и функция распределения непрерывной случайной величины со значениями в $\mathbb{R}^d$ связаны соотношениями

$$f_\xi(\mathbf{x}) = \frac{\partial F_\xi}{\partial x_1 \cdots \partial x_d}(\mathbf{x}) \quad F_\xi(\mathbf{x}) = \int\limits_{\langle -\infty;\mathbf{x}\rangle} f_\xi(\mathbf{t})\, d\mathbf{t} \tag{2.1}$$

Формулы обращения для интегральных преобразований представлены в [39], [11]. Отдельно стоит отметить, что в работе [34] показано как можно вычислять квантильную функцию по плотности распределения и наоборот, не прибегая к вычислению функции распределения. Этот подход может оказаться полезным при работе с достаточно сложными плотностями.

Семейства вероятностных распределений

В задачах статистики, как правило, оперируют не с одним каким-то конкретным распределением, а с набором распределений, из которого надо выбрать наиболее подходящее, или проверить какую-то гипотезу. Более строго, *параметрическим семейством распределений* называется некоторое множество $\{\mathbb{P}_\theta\}_{\theta\in\Theta}$ распределений, зависящих от скалярного или векторного параметра θ, Θ — множество возможных значений параметра [38].

Для любого распределения $\mathbb{P}_\xi$ случайной величины ξ определено семейство локации и масштаба, т.е. семейство распределений всех аффинных преобразований величины ξ:

$$\mathrm{loc} + \mathrm{scale}\cdot\xi \sim \mathbb{P}^{\xi}_{(\mathrm{loc},\mathrm{scale});} \tag{loc-scale-family}$$

где параметры $\mathrm{loc}, \mathrm{scale} \in \mathbb{R}$ для вещественнозначных случайных величин, и $\mathrm{loc} \in \mathbb{R}^d$, $\mathrm{scale} \in \mathbb{R}^{d\times d}$ для векторозначных случайных величин. Примером такого семейства является семейство нормальных распределений $\mathcal{N}(\mu,\sigma)$, определяемых равенством (normal-family).

$$\mu + \sigma\cdot\xi, \quad \xi\sim\mathcal{N}(0,1), \text{ т.е. } f_\xi(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right) \tag{normal-family}$$

Более общим понятием является понятие семейства замкнутого относительно действия группы, см. [23] и [28].

Другим, в некотором смысле ортогональным, понятием является понятие экспоненциального семейства распределений [3]. Параметрическое семейство распределений $\{\mathbb{P}_\theta\}_{\theta\in\Theta}$ относится к экспоненциальному типу, если плотности (или функции вероятностей) которых можно записать в виде (exp-family) [4].

$$f(\mathbf{x}|\theta) = \exp(\langle \mathbf{T}(\mathbf{x}), \vec{\eta}(\theta)\rangle + A(\mathbf{x}) + D(\theta)) \tag{exp-family}$$

Многие распространенные семейства распределений являются экспоненциальными, см. [27] . Для моделей относящихся к экспоненциальным семействам существует богатая теория оценивания параметров [23]. Большой список параметрических семейств и связывающие их соотношения представлены в [22].

Приведенные выше семейства интересны с точки зрения теоретической статистики. С точки зрения прикладной статистики, интерес представляют распределения, которые допускают гибкость в плане оценивания параметров: так, для нормального распределения два параметра не только локацию и масштаб, но и всю форму распределения, причем такое поведение присуще не только нормальному распределению. Для того чтобы решить эту проблему, было предложено несколько гибких семейств распределений, среди которых широко распространены семейство распределений Пирсона [8] и металогическое семейство [16].

[4] При этом требуется чтобы множество точек $\mathbf{x}$, в которых плотность отлична от 0, не зависело от параметра θ

Отдельно стоит отметить, что многие параметрические семейства зачастую имеют несколько параметризаций, каждая из которых может быть удобна в том или ином контексте, например в работе [30] приведены четыре параметризации для обобщенного гиперболического распределения. Множество других различных параметризаций для одних и тех же семейств собраны в базе проекта ProbOnto [35].

Преобразования случайных величин

Во многих моделях распределения могут быть составлены из более простых распределений с помощью различных методов. В [3] отмечается что, в контексте статистического вывода, любая модель для данных может рассматриваться как вероятностное распределение. Существует множество комбинировать и преобразовывать вероятностные распределения, интересная практическая реализация этого взгляда доступна в библиотеке Pomegranate, [32]. Ниже рассмотрены основные способы для непрерывных вещественнозначных случайных величин, большинство которых относят к теории алгебры случайных величин, см. [33].

- *Аффинное преобразование.* Если случайная величина ξ имеет распределение с функцией плотности $f_\xi(x)$, то плотность её линейного преобразования $a\xi + b$ описывается равенством (aff-tr);

$$f_{a+b\xi}(y) = \frac{1}{|a|} f_\xi\left(\frac{y-b}{a}\right), \quad a \neq 0. \tag{aff-tr}$$

- *Биективное преобразование.* Плотность распределения случайной величины $g(\xi)$, где g — строго монотонная функция,определяется через обратную функцию $g^{-1}(y)$ с помощью равенства (bij-tr).

$$f_{g(\xi)}(y) = f_\xi(g^{-1}(y)) \cdot \left|\frac{d}{dy} g^{-1}(y)\right|. \tag{bij-tr}$$

  Уравнение (aff-tr) является частным случаем (bij-tr). Для немонотонных функций распределение вычисляется с использованием разбиения на участки монотонности;

- *Распределение суммы независимых случайных величин* Если случайные величины ξ_1 и ξ_2 независимы[5], то плотность суммы $\xi_1 + \xi_2$ вычисляется с помощью свёртки (sum-rv).

$$(f_{\xi_1} \oplus f_{\xi_2})(z) = \int_{\mathbb{R}} f_{\xi_1}(x) f_{\xi_2}(z-x)\, dx. \tag{sum-rv}$$

- *Распределение произведения независимых случайных величин.* Для независимых случайных величин ξ_1 и ξ_2, плотность их произведения $\xi_1 \cdot \xi_2$ задается с помощью мультипликативной свертки (prod-rv).

$$(f_{\xi_1} \odot f_{\xi_2})(z) = \int_{\mathbb{R}} \frac{1}{|x|} f_{\xi_1}(x) f_{\xi_2}\left(\frac{z}{x}\right) dx. \tag{prod-rv}$$

Другие две важные операции возникают при работе со случайными векторами — маргинализация (взятие проекции) и вычисление порядковых статистик. Умение вычислять такие преобразования позволяет получать точные оценки качества в некоторых моделях статистического вывода.

- *Проекции.* Для случайного вектора $\xi = (\xi_1, \xi_2, \ldots, \xi_d)$ (с возможно зависимыми компонентами) с функцией распределения $F_\xi(\mathbf{x})$, функция распределения случайного подвектора $(\xi_{i_1}, \ldots, \xi_{i_k})$, описывается пределом (proj-tr)

$$F_{\Pr(\xi; i_1, \ldots, i_k)}(x_{i_1}, \ldots, x_{i_k}) = \lim_{\substack{x_i \to +\infty \\ i \neq i_1, \ldots, i_k}} F_\xi(x_1, \ldots, x_d) \tag{proj-tr}$$

- *Длины и углы.* Как правило, под случайными вектором понимается случайный элемент $\mathbb{R}^d$ представимый своими координатами. Однако, в некоторых ситуациях (см. например [13]) куда удобнее оперировать со сферическими или другими координатами. В этом случае работает многомерный аналог формулы (bij-tr).

Отдельно стоит отметить что важную роль играют порядковые статистики (см. [24]). Если компоненты случайного вектора $\xi = (\xi_1, \xi_2, \ldots, \xi_d)$ независимы и одинаково распределены с плотностью распределения $f(x)$ и функцией распределения $F(x)$, то их порядковые статистики $\xi_{(1:d)} \leq \xi_{(2:d)} \leq \ldots \leq \xi_{(n)}$ имеют плотности, описываемые равенством (ord-stat).

$$f_{\xi_{(k)}}(x) = \frac{n!}{(k-1)!(n-k)!} [F(x)]^{k-1} [1-F(x)]^{n-k} f(x). \tag{ord-stat}$$

[5] Т.е. $\mathbb{P}(\xi_1 \in B_1 \text{ и } \xi_2 \in B_2) = \mathbb{P}(\xi_1 \in B_1) \cdot \mathbb{P}(\xi_2 \in B_2)$ верно для всех $B_1, B_2 \subset \mathbb{R}$, являющихся борелевскими

Помимо трансформации одного распределения в другое и алгебраических операций над распределениями, еще одним способом образования сложного распределения из нескольких простых является образование *смесей*. Смеси используется для построения сложных вероятностных моделей, в которых присутствуют скрытые параметры и широко применяются в кластерном анализе и для изучения ядерных оценок плотности. Можно выделить два класса смесей; дискретные и непрерывные.

- *Дискретная смесь*. Под дискретной смесью подразумевают комбинацию конечного числа распределений, каждое из которых взвешено определённым коэффициентом. Функция распределения дискретной смеси случайных величин $\xi_1, \ldots, \xi_n$ с весами $\vec{w} = (w_1, \ldots, w_n)$ задаётся равенством (dmix);

$$F_{\mathrm{mix}(\vec{w};\xi)}(x) = \sum_{i=1}^{n} w_i F_{\xi_i}(x), \quad \sum_{i=1}^{n} w_i = 1 \tag{dmix}$$

- *Непрерывная смесь* является непрерывным аналогом дискретной смеси. Пусть $F(x \mid \theta)$ семейство плотностей, зависящее от параметра $\theta \in \Theta \subset \mathbb{R}^n$. Если на Θ задано некоторое распределение параметров с плотностью $\omega(\theta)$, непрерывная смесь плотностей $F(x \mid \theta)$ определяется равенством (cmix)

$$F_{\mathrm{mix}(\omega;F_\xi(\cdot|\theta))}(x) = \int_\Theta F_\xi(x \mid \theta) \cdot w(\theta)\, d\theta, \tag{cmix}$$

  В данном случае параметр θ рассматривается как случайная величина с заданным распределением.

Общая теория смесей в абстрактном случае и конкретные примеры приведены в работе [6].

В приложениях часто возникают понятия цензурированных и урезанных распределений [9]. Для распределения $\mathbb{P}_\xi$ случайной величины ξ принимающей вещественные значения, урезанным называется условное распределение, определяемое равенством (truncated-dist).

$$\mathbb{P}_{\mathrm{Truncated}(\xi,L,R)}(B) = \frac{\mathbb{P}_\xi([L;R] \cap B)}{\mathbb{P}_\xi([L;R])} \tag{truncated-dist}$$

Распределение (truncated-dist) это условное распределение ξ если априори известно, что значение ξ лежит в отрезке $[L;R]$. В свою очередь, цензурированным на отрезке $[L;R]$ распределением называется распределение случайной величины, определяемой равенством (censored-dist).

$$.\mathrm{Censored}(\xi, L, R) = \begin{cases} L & \xi < L \\ \xi & L \le \xi \le R \\ R & R < \xi \end{cases} \tag{censored-dist}$$

Такие распределения часто возникают в задачах регрессионного анализа, см. [5].

Числовые характеристики вероятностных распределений

Для анализа моделей важную роль играют не только функциональные, но и числовые характеристики распределений. Согласно [9], [36], для случайной величины ξ со значениями из $\mathbb{R}$ можно выделить следующие характеристики.

- Меры центральной тенденции, описывающие, вокруг какого значения сконцентрированы реализации случайной величины.
  - *Математическое ожидание*. Для случайной величины ξ, её математическое ожидание определяется равенством (mean);

$$\mathbb{E}[\xi] = \int_{\mathbb{R}} x \mathbb{P}_\xi(dx) \tag{mean}$$

  - *Медиана*. Для случайной величины ξ, её медиана определяется как множество всех значений m, таких что $F_\xi(m) = \frac{1}{2}$, иначе говоря, медиана определяется равенством (med).

$$\mathrm{Med}[\xi] = F_\xi^{-1}\left(\tfrac{1}{2}\right) \tag{med}$$

    В некоторых случаях, в качестве медианы берут какое-то конкретное значение из множества $F_\xi^{-1}(\frac{1}{2})$, такое значение называется *точной медианой* [36];
  - *Мода*. Для случайной величины ξ её мода определяется равенством (mode)

$$\mathrm{Mode}[\xi] = \mathrm{argmax}_{\mathbb{R}}\, f_\xi(x) \tag{mode}$$

    где $f_\xi(x)$ это функция вероятности, если ξ — дискретная случайная величина, и плотность, если ξ непрерывная случайная величина.

- Меры разброса (иногда меры рассеивания) указывают на склонность величины отклоняться от своего центрального значения.
  - *Дисперсия* и *среднеквадратичное отклонение* определяются равенствами (std) и (std) соответственно;

$$\mathbb{D}[\xi] = \mathbb{E}\big[(\xi - \mathbb{E}[\xi])^2\big]; \tag{var}$$

$$\operatorname{std}[\xi] = \sqrt{\mathbb{D}[\xi]} \tag{std}$$

  - *Среднее абсолютное отклонение* определяются равенством (mad).

$$\operatorname{mad}[\xi] = \mathbb{E}[|\xi - \mathbb{E}[\xi]|] \tag{mad}$$

  - *Межквартильный размах* определяется равенством (iqr).

$$\operatorname{IQR}[\xi] = \omega_\xi(0.75) - \omega_\xi(0.25) \tag{iqr}$$

- Меры скоса—мера асимметрии распределения относительно среднего значения.
  - Коэффициент скоса определяется равенством (skew);

$$\operatorname{skew}[\xi] = \frac{\mathbb{E}\big[(\xi - \mathbb{E}[\xi])^3\big]}{\operatorname{std}^3[\xi]} \tag{skew}$$

  - Коэффициент скоса Пирсона определяется равенством (pskew);

$$\operatorname{Skew}^{\mathrm{P}}[\xi] = \frac{\mathbb{E}[\xi] - \operatorname{Med}[\xi]}{\operatorname{mad}[\xi]} \tag{pskew}$$

  - Обобщенный коэффициент скоса Грюневельда определяется равенством (pskew).

$$\gamma(u) = \frac{\omega_\xi(1-u) + \omega_\xi(u) - 2\omega_\xi\left(\frac{1}{2}\right)}{\omega_\xi(u) - \omega_\xi(1-u)} \quad \tfrac{1}{2} < u < 1 \tag{qskew}$$

- Меры эксцесса и тяжести хвостов
  - *Коэффициент эксцесса* измеряет степень остроты вершины распределения и определяется равенством (kurt).

$$\operatorname{kurt}[\xi] = \frac{\mathbb{E}\big[(\xi - \mathbb{E}[\xi])^4\big]}{\operatorname{std}^4[\xi]} - 3 \tag{kurt}$$

  - *Квантильный коэффициент эксцесса* является квантильным аналогом стандартного коэффициента эксцесса [31] и определяется равенством (qkurt);

$$\kappa(u, v) = \frac{\omega_\xi(1-u) - \omega_\xi(u)}{\omega_\xi(v) - \omega_\xi(v)}, \quad 0 < u < v < \frac{1}{2} \tag{qkurt}$$

  - *Экспонента хвоста* определяется для распределений, функция выживания которых убывает согласно степенному закону, как число α при котором верна асимптотическая эквивалентность (tail-idx).

$$S_\xi(x) \sim x^{-\alpha}, \quad x \to \infty. \tag{tail-idx}$$

    В общей ситуации отдельно определяется индекс для левого хвоста и для правого хвоста.

Также, для случайной величины ξ и натурального числа $n \in \mathbb{N}$ определены моменты, центральные моменты (m_n и μ_n соответственно в равенстве (moment)), абсолютные моменты, абсолютные центральные моменты (v_n и ν_n в равенстве (abs-moment)) и факториальные моменты (κ_n в равенстве (fact-moment)) порядка n. На основе этих характеристик можно производить оценку параметров распределения.

$$m_n = \mathbb{E}[\xi^n] \qquad \mu_n = \mathbb{E}[(\xi - \mathbb{E}[\xi])^n] \tag{moment}$$

$$v_n = \mathbb{E}\Big[|\xi|^k\Big], \qquad \nu_n = \mathbb{E}[|\xi - \mathbb{E}[\xi]|^n] \tag{abs-moment}$$

$$\kappa_n = \mathbb{E}[\xi(\xi-1)(\xi-2)\dots(\xi-n+1)] \tag{fact-moment}$$

Обобщением моментов являются так называемые L-моменты [15] и их квантильные аналоги LQ-моменты [26].

Другое семейство числовых характеристик приходит из области теории информации, см. например монографию [19]. Далее, подразумевается что ξ необязательно вещественнозначная случайная величина, со значениями из некоторого пространства $\mathcal{X}$ и под плотностью подразумевается плотность в смысле (pdf).

- *Энтропия* распределения с плотностью p относительно меры μ определяется равенством (entr)

$$H_r(p) = -\int_{\mathcal{X}} p(x) \log_r p(x)\, d\mu \tag{entr}$$

- *Кросс-энтропия* из распределения с плотностью p в распределение с плотностью q определяется равенством (entr)

$$CE_r(p\|q) = -\int_{\mathcal{X}} q(x) \log_r p(x)\, d\mu \tag{cross-enrt}$$

- *KL-дивергенция* является мерой расхождения между двумя распределениями с плотностями p и q и определятся равенством (kl-div)

$$\mathcal{D}(p\|q) = -\int_{\mathcal{X}} q(x) \log_r \frac{p(x)}{q(x)}\, d\mu \tag{kl-div}$$

Методы из теории информации активно применяются в статистике, см. например [1]. В частности, зачастую рассматривают обобщенный вариант KL-дивергенции — *f-дивергенцию*, которая определяется для любой выпуклой функции $f\colon \mathbb{R}_+ \to \mathbb{R}$ равенством (f-div).

$$\mathcal{D}_f(p\|q) = \int_{\mathbb{R}} q(x) f\left(\frac{p(x)}{q(x)}\right) dx \tag{f-div}$$

С информационными характеристиками тесно связана информация Фишера. Для параметрического семейства плотностей $f(x;\theta), \theta \in \Theta \subseteq \mathbb{R}$, информация Фишера это функция от параметра, задаваемая равенством (FI).

$$\mathcal{I}(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta} \log f_\xi(x;\theta)\right)^2\right] \tag{FI}$$

Способы моделирования вероятностных распределений

Для вычисления метрик качества и моделирования поведения вероятностных систем необходимо уметь производить стохастическое моделирование случайных величин. Согласно [14], для генерации выборок можно выделить три основных метода:

- *Метод обратного преобразования*, который базируется на том факте, что для случайной величины ξ с квантильной функцией $\omega_\xi(p)$, распределение случайной величины $\omega_\xi(U), U \sim \mathcal{U}(0;1)$ будет совпадать с распределением ξ.
- *Метод декомпозиции*, который используется для генерации выборок из смешанных распределений. Общая идея заключается в том, что сначала генерируется значение параметра (например номер кластера), а затем уже случайная величина при условии зафиксированного значения параметра.
- *Метод отбора (rejection sampling)*, который используют когда предыдущие два метода не могут быть использованы. Этот метод значительно медленнее предыдущих и при его использовании возникает много нюансов, но для его использования необходим доступ только к плотности распределения.

При этом дискретные распределения требуют отдельного рассмотрения. Существуют также методы основанные на методе отбора, для генерации выборок из распределений заданных с помощью интегральных преобразований. При этом, нередки ситуации, плотность распределения $f_\xi(x)$ известна только с точностью до нормализующей константы или если требуется производить генерацию в сложных или многомерных пространствах. Для таких случаев разработаны методы на основе марковских цепей Монте-Карло (MCMC). Суть метода заключается в том, что на пространстве реализаций надо завести некоторое случайное блуждание, которое в пределе будет давать желанное распределение, детали см. например в книге [25].

Примеры вероятностных моделей в PySATL

В заключение этого раздела отметим, что поддержка работы с представленными ранее объектами является необходимой для PySATL в рамках существующих и будущих проектов. Пакет `MPEst`[6] использует различные числовые характеристики, такие как L-моменты для оценок параметров в моделях смеси.

Пакет `NMVMEstimation`[7] занимается специальными видами непрерывных смесей и использует различные интегральные преобразования. Библиотека `Experiment`[8] использует базовые характеристики распределений для оценок мощностей статистических тестов методом Монте-Карло. С помощью арифметики распределений можно будет получать точные распределения статистик используемых при проверке гипотез В ближайшем будущем планируется начать разработку библиотек для регрессионного анализа и оценки параметров, где также широко потребуется использование различных свойств и характеристик распределений.

[6] https://github.com/PySATL/MPEst
[7] https://github.com/PySATL/PySATL_NMVM_Module
[8] https://github.com/PySATL/pysatl-experiment

Раздел 3

Заинтересованные лица

Настоящий раздел описывает различные типы заинтересованных сторон, которым будет интересен данный документ, а также их потенциальные опасения/запросы связанные с дизайном ядра.

Мы отметим, что представленные здесь стороны/лица необязательно относятся к пользователям, которые взаимодействуют с системой.

3.1. Разработчики ядра PySATL

Это лица непосредственно принимающие участие в написании кода для ядра. Для них приоритетным является ряд возможностей:

- Добавление новой функциональности или оптимизация уже существующей;
- Покрытие кодовой базы тестами; в частности регрессионными тестами;
- Сопровождение существующей кодовой базы;
- Расширение коллектива разработчиков;

1. Добавление новой функциональности Разработчики ядра расширяют и модифицируют библиотеку в контексте следующих задач:

- Добавление новых параметрических семейств/новых операций над распределениями;
- Добавление новых числовых и/или функциональных характеристик распределений;
- Добавление новых численных методов для вычисления характеристик распределений;
- Добавление новых численных методов для вычисления операций над распределениями;

2. Тестирование Помимо расширения функциональности, разработчики ядра покрывают кодовую базу тестами; Им необходимо иметь ряд механизмов для:

- автоматического тестирования новой функциональности, касающейся свойств добавляемых распределений/семейств/операций
- создания регрессионных тестов для функциональности, оперирующей с псевдо-случайным данными

Пример. При создании нового семейств распределений разработчик указывает что все его представители имеют носитель на отрезке $[0; 1]$. Это свойство распределения которое можно проверить например сгенерировав выборку из данного распределения и проверив что все элементы выборки лежат в этом отрезке.

Пример. Валидационные примеры использования статистических процедур часто служат основной для регрессионых тестов этих самых процедур. Библиотека должна обеспечивать воспроизводимость генерации выборок.

3. Сопровождение кодовой базы и расширения коллектива разработчиков

Сопровждение кодовой базы и расширенние коллектива разработчиков осуществляется засчет нескольких инструментов

- Архитектурная документация; настоящий документ является её частью. В частности, так как ядро планируется активно использовать в других библиотеках, необходимо явно отразить какие части системы являются публично доступными
- Техническая и пользовательская документация; наличие качественной документации, содержащей примеры использования и описывающей роли компонент сисемы

3.2. Разработчики других библиотек в PySATL

К этой категории относятся все разработчики PySATL, которые либо не принимают непосредственного участия в разработке ядра, либо делают это эпизодически, добавляя функциональность под конкретные нужды проектов (в последнем случае они относятся к первой категории заинтерсованных лиц). Их основные интересы по отношению к ядру состоят в следующем.

1 Простота интеграции ядра в другие библиотеки PySATL;
2 Конфигурируемость ядра для типичных сценариев использования;

1. Простота интеграции ядра в другие библиотеки PySATL

Сейчас PySATL активно использует связку Numpy/SciPy для большинства математических вычислений; Это означает что при переходе на ядро, во всех местах использующих эту связку, замена не должна вызвать осложнений. В частности ядро

- Не должно обязывать разработчика других библиотек к конфигурации численных методов, использующихся в ядре;
- Должно иметь всю ту же функциональность, что и модуль `statistics` библиотеки SciPy;
- Должно предоставлять унифицированный интерфейс для прочих математических функций;

Здесь надо пояснить что такое прочие математические функции. При реализации различных оценок, иногда вознкиают довольно нетривиальные объекты (так pysatl-nmvm использует многочлены Белла для вычислений, а mpest - различные методы оптимизации). Так как существует множество математических библиотек, которые могут предоставлять данные возможности, то:

1 Возможно, конечный пользователь библиотеки (не обязательно ядра) захочет чтобы использовались какие-то конкретные математические пакеты (что очень может быть в случае интегрирования/оптимизации)
2 Наличие единного интерфейса для математических утилит не приведет к тому что внутри разных библиотек (или даже одной библиотеки) используются разные математические пакеты

К тому же, многие из математических пакетов, предоставляющих редкие, но нужные функции, являются либо проприетарными, либо с вирусными лицензиями. Использование и тех, и других, не подходит под лицензионные ограничения; создание единого интерфейса для математических утилит позволяет распостранять PySATL под MIT лицензией, используя вирусную лицензию только для реализации этих интерфейсов. Это значительно позволит сократить ресурсы на прототипирование разлчиных библиотек на базе ядра.

2. Конфигурируемость ядра для типичных сценариев использования

Различные библиотеки проекта PySATL предъявляют различные, зачастую противоречивые, требования к вычислительным характеристикам ядра. Библиотека должна предоставлять механизмы тонкой настройки без необходимости модификации её исходного кода. Это включает в себя:

- **Выбор численных методов:** Возможность выбора конкретной реализации алгоритма (например, для вычисления интеграла или оптимизации) в зависимости от требований к точности, скорости или устойчивости. Библиотека `experiment` может требовать максимальной скорости для генерации больших выборок, в то время как `mpest` — максимальной надёжности и точности для сходимости алгоритмов оценки параметров.
- **Управление вычислительными ресурсами:** Настройка параметров, влияющих на производительность и использование памяти. Например:
  - Установка допусков (tolerances) для итеративных алгоритмов;
  - Задание лимитов на количество итераций;
  - Управление размером кэшей для повторных вычислений.
- **Стратегии семплирования:** Конфигурация генераторов псевдослучайных чисел (ГПСЧ), включая:
  - Выбор конкретного алгоритма ГПСЧ (например, Mersenne Twister, PCG64);
  - Установка начального значения (seed) для обеспечения воспроизводимости результатов;
  - Управление параллельными потоками генерации для эффективного использования многоядерных систем.
- **Политики обработки ошибок:** Возможность настройки реакции на исключительные ситуации — от строгого прерывания вычисления с выводом подробной ошибки до возврата специальных значений (NaN, inf) или использования fallback-алгоритмов для обеспечения отказоустойчивости.
- **Абстракция математического бэкенда:** Единый интерфейс для низкоуровневых математических операций (линейная алгебра, специальные функции, оптимизация), позволяющий подменять реализации без изменения кода, зависящего от ядра. Это гарантирует, что библиотеки `nmvm` и `mpest` будут использовать согласованный computational stack, конфигурируемый централизованно.

3.3. Руководители проекта PySATL

Данная категория включает лиц, ответственных за планирование развития проекта в целом, распределение ресурсов и координацию между рабочими группами. Их интересы сфокусированы на стратегических аспектах разработки и эксплуатации ядра, а также на его интеграции в общую экосистему PySATL.

- Представление о планах разработки ядра и его функциональности на каждом этапе;
- Гарантии, предоставляемые ядром;
- Эффективное использование ресурсов и соблюдение сроков;
- Минимизация рисков, связанных с техническим долгом и зависимостями.

1. Планы разработки и функциональность

Руководителям проекта необходимо иметь чёткое и актуальное представление о состоянии и roadmap ядра для принятия обоснованных решений. Это включает:

- Соответствие фактической функциональности ядра заявленным планам и требованиям других подсистем PySATL;
- Наличие документации, достаточной для оценки готовности компонент к интеграции;
- Понятный и измеримый прогресс разработки, позволяющий оценивать выполнение этапов.

2. Гарантии, предоставляемые ядром

Поскольку ядро является фундаментальным компонентом всего проекта, от его качества и стабильности зависит работа всех надстроенных над ним библиотек. Руководители заинтересованы в наличии гарантий, обеспечиваемых за счёт:

- Всестороннего тестирования, включая модульные, интеграционные и регрессионные тесты;
- Чёткого определения и поддержания публичных интерфейсов (API), что обеспечивает стабильность для потребителей ядра;
- Предсказуемой политики управления версиями, минимизирующей обратно несовместимые изменения;
- Исчерпывающей документации, снижающей риски misinterpretation API и его неправильного использования.

3. Эффективность использования ресурсов и сроки

Процесс разработки ядра должен быть организован таким образом, чтобы:

- Архитектурные решения способствовали повторному использованию кода и сокращению избыточности;
- Существовали механизмы для быстрого прототипирования и валидации новых идей без угрозы нарушения стабильности основной codebase;
- Соблюдались согласованные сроки поставки критически важной для других подсистем функциональности.

4. Управление рисками

Ключевым аспектом является проактивное выявление и снижение потенциальных рисков, в частности:

- Контроль над внешними зависимостями, их лицензионной чистотой и совместимостью;
- Минимизация накопления технического долга за счёт регулярного рефакторинга и применения современных практик разработки;
- Наличие плана по обеспечению долгосрочной поддерживаемости и расширяемости кодовой базы.

3.4. Инженеры-исследователи

К данной категории относятся специалисты, применяющие ядро PySATL для решения прикладных задач: программисты, математики, статистики и исследователи в области численных методов. Они используют систему как инструмент для реализации и проверки научных гипотез, прототипирования алгоритмов и проведения вычислительных экспериментов.

- Поддержка сложных операций над распределениями, смесями и иерархическими моделями;
- Бенчмаркинг новых алгоритмов и сравнение с существующими аналогами;
- Гибкость и выразительность для описания нестандартных вероятностных моделей;
- Воспроизводимость и контролируемость вычислительных экспериментов.

1. Поддержка сложных операций и моделей

Инженеры-исследователи работают с комплексными вероятностными структурами, выходящими за рамки стандартных распределений. Их интерес заключается в возможности:

- Конструировать составные распределения (смеси, свёртки, произведения);
- Определять иерархические (многоуровневые) вероятностные модели;
- Применять нелинейные функциональные преобразования к распределениям;
- Работать с усечёнными, цензурированными или модифицированными распределениями.

2. Бенчмаркинг и валидация алгоритмов

Для данной аудитории критически важна возможность оценивать эффективность и корректность реализуемых методов:

- Наличие встроенных средств для замеров производительности (профилирования) отдельных операций;
- Возможность сравнивать численные результаты, полученные с помощью ядра, с эталонными реализациями или аналитическими выкладками;
- Доступ к низкоуровневым параметрам численных методов (таким как точность, допуски, критерии остановки) для тонкой настройки и исследования сходимости алгоритмов.

Пример. Исследователь, разрабатывающий новый метод оценки параметров, должен иметь возможность сгенерировать набор синтетических данных из известного распределения, применить свой метод, сравнить полученные оценки с истинными значениями параметров и измерить время выполнения алгоритма, обеспечивая при этом полную воспроизводимость эксперимента.

3. Гибкость и выразительность

Система должна предоставлять достаточный уровень абстракции для лаконичного и интуитивно понятного выражения математических концепций:

- Декларативный способ задания моделей, приближенный к математической нотации;
- Возможность легко комбинировать примитивы ядра для создания специализированных конструкций без необходимости написания большого объема boilerplate-кода;
- Доступ к символьным представлениям преобразований для анализа и оптимизации выражений до их численного вычисления.

4. Воспроизводимость и контролируемость

Так как работа инженера-исследователя носит экспериментальный характер, ядро должно предоставлять инструменты для обеспечения строгости научного процесса:

- Детерминированная работа генератора псевдослучайных чисел с возможностью сохранения и восстановления состояния (seed);
- Логирование ключевых этапов вычислений и принятых решений (например, в алгоритмах оптимизации);
- Чёткое разделение между абстрактной спецификацией модели и конкретным численным методом её вычисления.

Раздел 4

Ключевые требования, определяющие архитектуру

4.1. Сценарии использования системы

todo; see ISSUE

4.2. Технические ограничения и качественные требования

4.2.1. Технические ограничения

В данном разделе перечислены внешние технические ограничения, влияющие на архитектуру и реализацию системы. Эти ограничения задаются проектом PySATL или внешними условиями и не подлежат изменению в рамках проектирования ядра.

- *Язык реализации*: система должна предоставлять основной пользовательский интерфейс на языке Python версии 3.x. Допускается использование других языков программирования (например, C/C++) во внутренней реализации с целью оптимизации вычислений, при условии полной прозрачности интерфейса для Python-пользователя.
- *Лицензирование*: система должна распространяться под MIT-лицензией. Использование копилефт-лицензий (например, GPL) запрещено.
- *Кроссплатформенность*: система должна работать на основных операционных системах (Linux, macOS, Windows) без необходимости установки платформозависимых компонентов или привязки к конкретной среде исполнения.
- *Вычислительные ограничения*: система не должна требовать наличия GPU или специализированного оборудования. Все вычисления должны корректно выполняться на CPU без дополнительных зависимостей.

4.2.2. Качественные характеристики системы

Данный раздел описывает нефункциональные требования, оказывающие влияние на архитектуру, проектные решения и организацию компонентов системы.

4.2.2.1. Расширяемость

Система должна обеспечивать возможность добавления новых распределений, семейств и преобразований без необходимости изменения существующего кода. Расширение должно происходить через зарегистрированные интерфейсы или стандартные механизмы расширения Python.

4.2.2.2. Модульность и интерфейсы:

Система должна быть реализована в виде набора Python-модулей с чётко определёнными публичными интерфейсами. Модули должны быть слабо связаны друг с другом, а взаимодействие между ними должно происходить исключительно через открытые контракты. Пакет в целом должен также иметь публичный интерфейс для взаимодействия с пользователем и другими подсистемами PySATL. Внутренние и внешние интерфейсы могут быть различны.

4.2.2.3. Совместимость

Система должна быть совместима с остальными компонентами PySATL. Переход на новое ядро должен быть максимально прозрачным для пользователя и не требовать значительных изменений в существующем коде или логике пакетов, использующих ядро.

4.2.2.4. Тестируемость

Архитектура должна способствовать покрытию тестами ключевой функциональности системы. Для стадии прототипа допускается ограниченное покрытие, однако оно должно быть достаточным для выявления ошибок при изменении поведения или интерфейсов.

4.2.2.5. Устойчивость к ошибкам

Система должна быть устойчива к некорректным пользовательским данным и иметь механизм информирования об ошибках. Поведение при ошибках должно быть контролируемым и предсказуемым, включая диагностику нарушений интерфейсов, типов и других контрактов.

4.2.2.6. Документируемость и визуализация:

Архитектура модулей, их интерфейсы и взаимосвязи должны быть представлены в виде UML-диаграмм и сопровождаться соответствующей технической документацией. Диаграммы должны отражать как общую структуру системы, так и реализацию ключевых компонентов и связей.

4.3. Ключевые функциональные требования

В данном разделе изложены функциональные требования к системе, определяющие её основную вычислительную модель, интерфейсы и возможности расширения. Эти требования отражают ожидаемое поведение ядра и набор поддерживаемых операций. Можно выделить следующие группы требований

(ГФТ1) *Определение распределения*: каждое распределение должно быть определено посредством как минимум одной функциональной характеристики (например, функция плотности или функция распределения), поддерживать генерацию выборки и вычисление логарифма правдоподобия. Распределение должно быть связано с типом случайной величины (дискретной или непрерывной) и типом значений выборки(многомерной или одномерной).

(ГФТ2) *Функциональные и числовые характеристики*: система должна поддерживать вычисление основных и дополнительных характеристик распределений, включая возможность их добавления пользователем, а также предоставлять гарантии корректности вычислений.

(ГФТ3) *Семплирование распределений*. Для каждого распределения должна быть доступна генерация выборки. При этом, пользователь должен иметь возможность добавлять свои способы семплирования и настраивать используемый способ семплирования. Система также должна предоставлять механизмы проверки качества семплирования.

(ГФТ4) *Поддержка основных распределений*: система должна предоставлять встроенную поддержку наиболее распространённых распределений (нормальное, экспоненциальное, биномиальное и др.), а также предусматривать механизм регистрации пользовательских распределений.

(ГФТ5) *Работа с семействами распределений*: система должна поддерживать задание семейств распределений, определяемых через параметрические пространства. Необходимо обеспечить возможность описания связей между семействами, включая асимптотику и граничные переходы (например, предельные случаи распределений при изменении параметров). Поддержка пользовательских семейств и добавления новых связей обязательна.

(ГФТ6) *Преобразования распределений*: система должна поддерживать стандартные операции преобразования распределений, включая арифметические операции, функциональные отображения, свёртки, смеси и т.п. Должна быть предусмотрена возможность компоновки и каскадирования таких преобразований.

4.3.1. Требования к числовым и функциональным характеристикам

Под корректным вычислением числовой или функциональной характеристики заданного распределения подразумевается следующее поведение системы:

1 В случае если значение характеристики определено и *всегда* является числом, оно должно вычисляться с предопределенной гарантией точности (например, в виде верхней оценки на относительную погрешность);

2 В случае если значение характеристики *равно* $\pm\infty$, должно выдаваться значение специального типа, представляющего собой абстракцию бесконечности;

3 В случае если значение характеристики *неопределенно*, система должна выдать объект типа NaN. Опционально должно выдаваться предупреждение о том что соответствующее значение не является числом или $\pm\infty$.

4 В случае если значение характеристики *может быть равно* $\pm\infty$ *или может быть неопределенно*, и заранее это невозможно проверить, необходимо предоставлять набор условий, при выполнений которых, с близкой к 1 вероятностью, значение можно считать вычисленным корректно, в смысле п.1;

Полнота:

- Для любого распределения должно быть доступно вычисление всех характеристик моментного/квантильного/информационного типа приведенных в глоссарии;

- Для любого распределения должно быть доступно вычисление всех функциональных характеристик (если они существуют для данного распределения) приведенных в глоссарии.

Гибкость и конфигурируемость:

- При наличии нескольких алгоритмов для расчета числовой/функциональной характеристики данного распределения, должен быть определен алгоритм, который используется по умолчанию;
- При наличии нескольких алгоритмов для расчета числовой/функциональной характеристики пользователь должен иметь возможность указать, какой алгоритм использовать.

Расширяемость:

- Пользователь должен иметь возможность добавлять свои собственные алгоритмы вычисления для уже существующих функциональных/числовых характеристик;
- Пользователь должен иметь возможность добавлять собственные числовые характеристики. В случае если числовая характеристика определена не для всех распределений, пользователь должен иметь возможность явно указать для какого класса распределений определена характеристика;
- Пользователь должен иметь возможность добавлять собственные функциональные характеристики. В случае если функциональная характеристика определена не для всех распределений, пользователь должен иметь возможность явно указать для какого класса распределений определена характеристика.

4.3.2. Требования к семействам распределений

- Для любого семейства распределений должна быть возможность создать новое семейство распределений, получаемое из исходного частичной подстановкой значений параметров;
- Для любого семейства распределений должна быть определена каноническая параметризация;
- Для любого семейства распределений должна быть возможность добавления новой параметризации. При добавлении новой параметризации, пользователь должен указать как параметры добавляемой параметризации выражаются через каноническую параметризацию и наоборот;
- Над семействами необходимо уметь производить операции образования смесей: непрерывных и дискретных;
- Должна быть возможность указать как представители семейства взаимодействуют с представителями других семейств при выполнении операций.

4.3.3. Требования к операциям над распределениями

- Для одномерных непрерывных распределений должны быть доступны операции описанные в параграфе «Преобразования распределений» в глоссарии;
- При наличии нескольких способов вычисления операции, должен быть определён способ вычисления по умолчанию.

4.3.4. Требования к генерации выборок

- Для любого распределения, за исключением относящихся к распределениям геометрических примитивов, должна быть возможность генерации выборок;
- При создании пользовательского распределения, система должна предоставлять методы для генерации выборок без необходимости реализовывать алгоритмы генерации;
- При наличии нескольких алгоритмов генерации выборок, пользователь должен иметь возможность выбирать алгоритм;
- Если алгоритм генерации выборок имеет параметры, то пользователь должен уметь настраивать эти параметры. К числу параметров также относится используемый источник случайных или псевдослучайных чисел.

Раздел 5

Избранные архитектурные точки зрения

5.1. Контекст

todo; see ISSUE

5.2. Композиция

Ядро строится вокруг трёх компонент: *Distributions*, *Families* и *Transformations*. Они объединены общей идеей: все вычисления характеристик случайных величин происходят по единым контрактам, а сами компоненты остаются слабо связаны и расширяемы.

5.2.1. Роли компонент

Families. Families отвечает за параметризацию и рождение конкретных распределений. Под параметризацией понимается множество имён параметров, а под параметрами — отображение этих имён в значения. Поддерживается несколько параметризаций для одной и той же семьи, однако все вычисления выполняются через базовую параметризацию. Перед любыми вычислениями параметры приводятся к базовой форме, чтобы остальные компоненты работали с единым представлением. Families сохраняет смысловую целостность параметров, их допустимость и область определения будущего распределения.

Distributions. Distributions — единая поверхность доступа к характеристикам распределений и точка их композиции. Компонента материализует распределение, зная его семью и параметры, и предоставляет согласованный интерфейс к характеристикам: функциям распределения, плотности, квантилям, моментам и т. д. Distributions не решает, *как* именно получена та или иная характеристика; она лишь координирует вычисление по зависимостям и контрактам, в том числе для составных объектов, которые появились в результате преобразований.

Transformations. Transformations фиксирует операции над распределениями и правила пересчёта характеристик после этих операций. Сюда попадают преобразования вида «аффинное», «сдвиг и масштаб», «сумма/микс», «свёртка», «обрезка (truncation)», «монотонные отображения» и другие композиции. Компонента описывает, *какие* характеристики нового распределения можно выразить через характеристики исходных и в каком порядке они должны вычисляться. На момент написания модуль проектно важен, но ещё не реализован; он задаёт общий язык описания преобразований и обеспечивает единообразную логику пересчёта поддержек, моментов, квантилей и прочих характеристик при переходе к результату преобразования.

5.2.2. Поток данных и точки соприкосновения

Рабочий цикл выглядит просто. Пользователь выбирает семью и задаёт параметры в удобной для него параметризации. Families приводит параметры к базовой форме и передаёт контекст в Distributions. Когда требуется характеристика, Distributions строит локальный граф зависимостей для этого запроса. Если характеристика получена применением преобразования, узлы этого графа добавляются модулем Transformations, а рёбра указывают, какие характеристики исходных распределений нужны для пересчёта. Таким образом, вычисления идут от стандартизованных параметров через согласованные правила преобразований к запрошенной характеристике результата.

5.2.3. Границы и контракты

Межкомпонентное взаимодействие опирается на несколько устойчивых сущностей: базовая параметризация, область значения случайной величины (support), именованный реестр характеристик и спецификация преобразований.

Компоненты не разделяют внутреннее состояние и обмениваются только тем, что необходимо для вычисления. Это позволяет добавлять новые семьи и новые преобразования без изменения существующих модулей, сохраняя обратную совместимость.

5.2.4. Композиционные свойства

Композиция прозрачна: результат преобразования ведёт себя как обычное распределение, подчиняясь тем же контрактам Distributions. Порядок вычисления характеристик незначим для внешнего пользователя, так как зависимости и правила вывода зашиты в Transformations. Единая базовая параметризация в Families гарантирует сопоставимость результатов при смешении разных источников и представлений параметров.

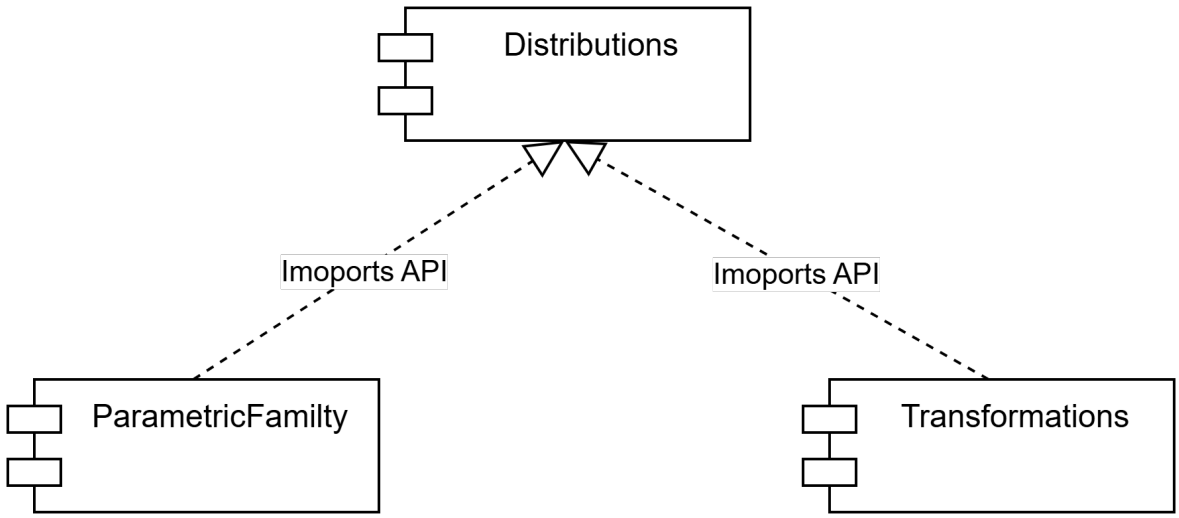

Рис. 5.1: Компоненты ядра и их роли в композиции

5.3. Логическая структура

5.3.1. Общий обзор

5.3.2. Модуль Distributions

В основе лежит класс `Distribution`, представленный в виде протокола Python. Это означает, что конкретные классы могут *реализовать* интерфейс `Distribution`, не наследуясь от общего базового класса. По этой причине почти все поля `Distribution` экспонируются как свойства (`@property`); обзор ключевых элементов модуля показан на рис. 5.2.

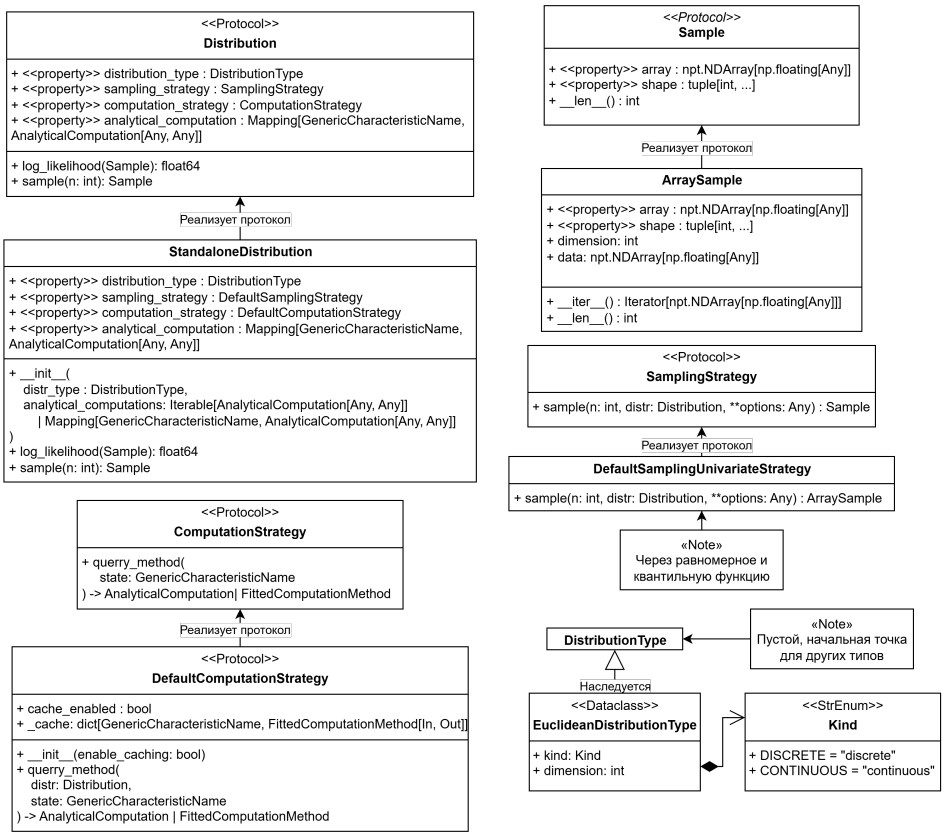

Рис. 5.2: UML-обзор модуля `Distributions`: протокол `Distribution`, типы `DistributionType`, стратегии `SamplingStrategy`/`ComputationStrategy`, выборки и базовые реализации по умолчанию.

5.3.2.1. Тип распределения (`DistributionType`)

`DistributionType` — маркерный тип, задающий родовую принадлежность распределения и цепочку наследований признаков, которые принципиально различают семейства распределений. Пример: `EuclidianDistributionType` со свойствами `kind` (`Discrete`/`Continuous`) и размерностью (евклидово пространство фиксированной размерности). Этот тип не содержит логики вычислений, но определяет область применимости правил преобразований характеристик.

5.3.2.2. Стратегии: выборки и вычислений

Стратегия сэмплирования (`SamplingStrategy`). Свойство `sampling_strategy` определяет алгоритм генерации выборки из распределения. Стратегия по умолчанию использует квантили (`ppf`) в сочетании с равномерным распределением для генерации входных вероятностей.

Стратегия вычислений (`ComputationStrategy`). Свойство `computation_strategy` задаёт политику получения функциональных характеристик. Стратегия принимает решения: выдавать ли характеристику из аналитически заданных, читать её из кэша, либо строить через цепочку преобразований. В реализации по умолчанию применяется следующий порядок:

1. проверка наличия аналитической реализации целевой характеристики;
2. проверка кэша результатов;
3. поиск пути на графе реестра преобразований (см. §5.3.2.7) и пошаговое вычисление.

Базовая стратегия поддерживает кэширование; кэш прозрачен для вызывающей стороны.

5.3.2.3. Аналитические характеристики (`AnalyticalComputation`)

`AnalyticalComputation` — это набор функциональных характеристик, заданных в явном (аналитическом) виде и не требующих вывода через другие характеристики. Для каждого конкретного распределения должен существовать хотя бы один такой «якорь», обеспечивающий возможность построения остальных характеристик через реестр преобразований. Здесь *не* подразумевается внешняя «база данных»; речь идёт о *базисе* характеристик самого распределения.

5.3.2.4. Выборки и правдоподобие

`Sample` и `ArraySample` — лёгкие обёртки над массивами `NumPy`, хранящие выборку и метаданные. Для любой реализации `Distribution` возможно:

- сгенерировать выборку согласно `sampling_strategy`;
- вычислить (лог-) функцию правдоподобия для этой выборки.

5.3.2.5. Протокол вычисления характеристики (`Computation`)

`Computation` — абстракция вычислительной процедуры для конкретной характеристики (структура на рис. 5.3):

- `target` — идентификатор целевой характеристики;
- `__call__` — унифицированный вызов вычисления.

`AnalyticalComputation` реализует `Computation`. Расширение протокола — `FittedComputationMethod`: оно вводит понятие *источников* (*sources*) — набора характеристик, из которых получается новая. Конкретная реализация `FittedComputationMethod` заполняет все поля и предоставляет функцию `func`, задающую фактическое преобразование.

Отдельно выделяется `ComputationMethod` — абстракция для сложных вычислений, которые не укладываются в сигнатуру вызова `FittedComputationMethod`. В частности, `ComputationMethod` намеренно *не* определяет `fit` как `__call__`, поскольку его сигнатура отличается. Такое разделение упрощает кэширование и повторное использование результатов.

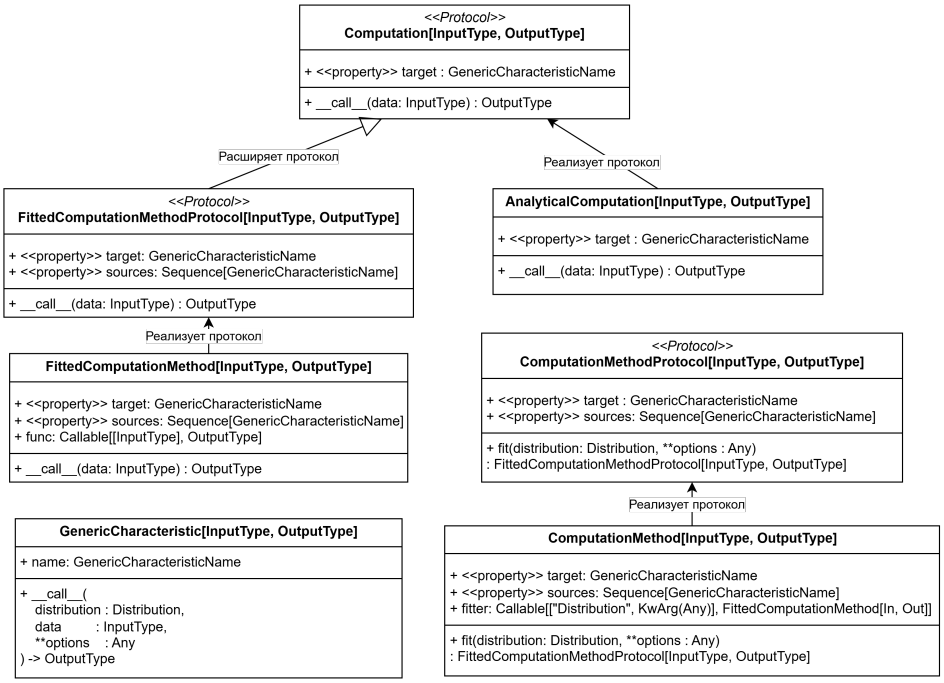

Рис. 5.3: Вычислительная модель: `Computation`, `AnalyticalComputation`, `FittedComputationMethod` и `ComputationMethod`; целевая характеристика (`target`), источники (`sources`) и единый механизм вызова.

5.3.2.6. Имена характеристик и единый механизм вызова

`GenericCharacteristicName` — типизированное имя характеристики (не протокол). Вызов для любой характеристики унифицирован:

1. у стратегии, закреплённой за распределением, запрашивается подходящий метод для `target`;
2. полученный метод вызывается над переданными данными.

Опции вычислений (`**options`). И `Characteristic`, и `ComputationMethod` поддерживают дополнительные параметры, влияющие на ход вычислений. Пример: при восстановлении `ppf` по `cdf` можно задать стратегию выбора ветви `most_left`/`most_right`. Конкретная стратегия *может* проигнорировать опцию, однако пользователь волен подключить собственную стратегию, которая эти опции учитывает.

5.3.2.7. Реестр преобразований характеристик

Сердцем архитектуры является `DistributionTypeRegister` — синглтон, содержащий для каждого `DistributionType` ориентированный граф переходов между характеристиками (см. рис. 5.4).

Структура графа.

- Узлы — имена характеристик.
- Рёбра — помечены парами {имя, `ComputationMethod`}. Имя по умолчанию зарезервировано как

  $$\texttt{DEFAULT\_COMPUTATION\_KEY} \equiv \texttt{"PySATL\_default\_computation"}.$$

  Зарезервированное имя используется для поставляемых реализаций по умолчанию и не рекомендуется для пользовательских методов, чтобы избежать непреднамеренного переопределения.

Классификация узлов. Все характеристики делятся на:

- *definitive* — однозначно задающие распределение (например, `pdf`, `cdf`, `ppf`);
- *indefinitive* — вычислимые по известным характеристикам, но не задающие распределение однозначно (например, математическое ожидание, дисперсия, медиана).

Инварианты.

1. Подграф *definitive*-узлов всегда сильно связен: из любой definitive-характеристики существует путь в любую другую.
2. *Indefinitive*-узлы являются стоками: обратных рёбер из них в множество definitive нет.

Операции реестра. Предоставляются следующие методы управления графом:

- `register_definitive(name)` — регистрация *единственного* начального definitive-узла в пустом графе. Если попытаться добавить второй исходный узел, сработает проверка инвариантов и будет брошено исключение.
- `add_bidirectional_conversion(a, b, method_ab, method_ba)` — либо добавляет пару взаимосвязанных узлов в пустой граф, либо присоединяет новый узел к уже существующему связному компоненту (рёбра в обе стороны обязательны, иначе нарушится связность definitive-подграфа).
- `add_conversion(a, b, method, name?)` — добавляет альтернативный маршрут между уже соединёнными узлами; тем самым поддерживаются множественные реализации преобразования.
- Служебные запросы: `is_definitive(name)`, `definitive_nodes()`, `indefinitive_nodes()`, `all_nodes()`.
- Поиск маршрута: `find_path(source, target)` — стандартный `BFS` по графу, используемый стратегией по умолчанию и доступный пользователю при реализации собственных стратегий.

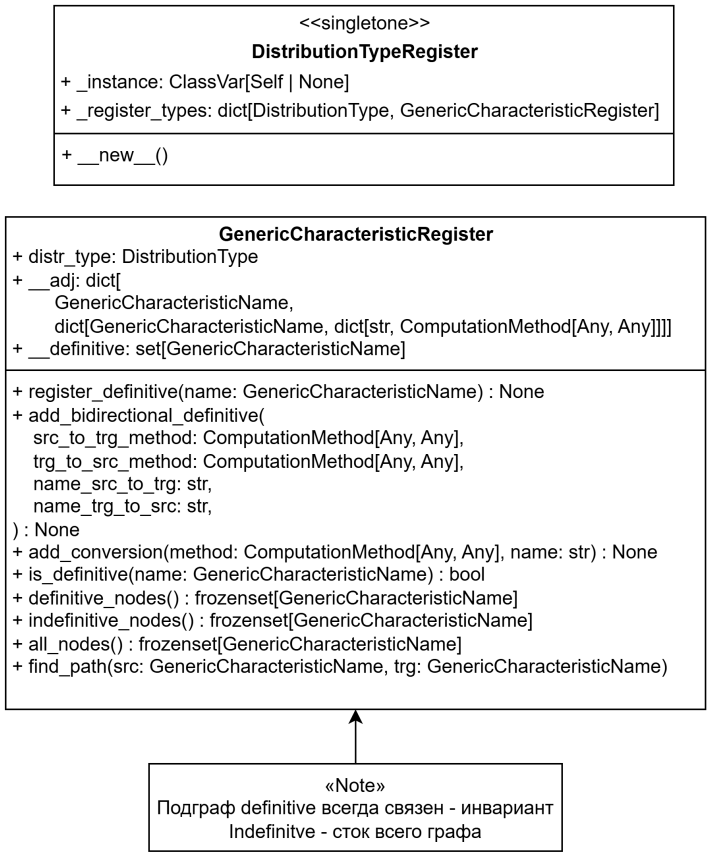

Рис. 5.4: Реестр преобразований для типа распределения: `DistributionTypeRegister` и `GenericCharacteristicRegister` с инвариантами (связность definitive-подграфа, отсутствие обратных рёбер из indefinitive).

5.3.3. Модуль `families`

Модуль `families` предоставляет фреймворк для работы с параметрическими семействами распределений в Python. Система позволяет определять семейства распределений с множественными параметризациями, ограничениями на параметры и методами вычисления характеристик распределений.

5.3.3.1. Общий обзор

На рис. 5.5 представлена UML-диаграмма классов модуля `families`

Реестр семейств, используется, чтобы в будущем можно было реализовывать переходы между распределениями из семейств, при каких-то операциях

Каждое семейство является объектом класса `ParametricFamily`. Это позволяет использовать систему с реестром, а также:

- Добавлять пользовательские параметризации для распределений;
- Добавлять новые `AnalyticalComputation` для распределений,

не модифицируя исходный код семейства.

На данный момент за работу с несколькими параметризациями отвечает класс `ParametrizationSpec`. Его объекты описывают какие параметризации есть у семейства, и кто из них является канонической. Сейчас используется следующий механизм: все аналитические вычисления задаются в базовой параметризации, далее каждая параметризация указывает, как из нее получить базовую параметризацию.

5.3.3.2. Основные компоненты системы

`ParametricFamiliesRegister` (Реестр параметрических семейств)

- **Назначение**: Синглтон-класс для глобальной регистрации и доступа ко всем параметрическим семействам.
- **Поля**:
  - `_instance` - единственный экземпляр реестра
  - `_registered_families` - словарь зарегистрированных семейств

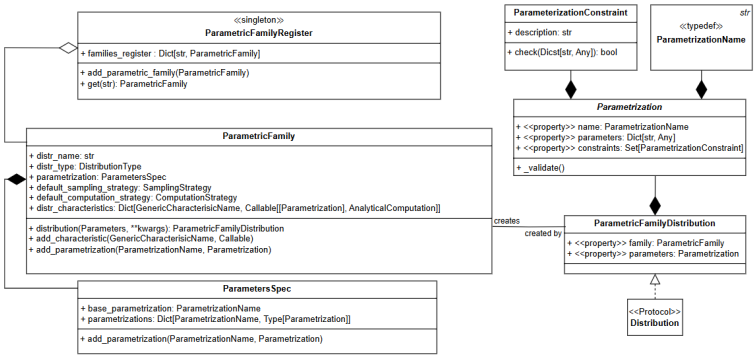

Рис. 5.5: Диаграмма классов модуля families

- **Методы**:
  - `get(name)` - получение семейства по имени
  - `register(family)` - регистрация нового семейства

`ParametrizationConstraint` (Ограничение параметризации)

- **Назначение**: Контейнер для ограничений на значения параметров распределения.
- **Поля**:
  - `description` - текстовое описание ограничения
  - `check` - функция проверки ограничения

`Parametrization` (Абстрактный класс параметризации)

- **Назначение**: Абстрактный базовый класс для всех параметризаций распределений.
- **Поля**:
  - `constraints` - список ограничений параметров
- **Абстрактные свойства**:
  - `name` - имя параметризации
  - `parameters` - словарь параметров
- **Методы**:
  - `validate()` - проверка всех ограничений
  - `transform_to_base_parametrization()` - преобразование к базовой параметризации.

`ParametrizationSpec` (Спецификация параметризаций)

- **Назначение**: Контейнер для управления множественными параметризациями семейства.
- **Поля**:
  - `parametrizations` - словарь всех параметризаций
  - `base_parametrization_name` - имя базовой параметризации
- **Методы**:
  - `base` - получение класса базовой параметризации
  - `add_parametrization()` - добавление новой параметризации
  - `get_base_parameters()` - преобразование параметров к базовой форме

`ParametricFamily` (Параметрическое семейство)

- **Назначение**: Представление семейства распределений с различными параметризациями.
- **Поля**:
  - `name` - имя семейства
  - `distr_type` - тип распределения
  - `parametrizations` - спецификация параметризаций
  - `distr_characteristics` - характеристики распределения
  - `sampling_strategy` - стратегия семплирования
  - `computation_strategy` - стратегия вычислений
- **Методы**:
  - `__call__()` - создание экземпляра распределения

`ParametricFamilyDistribution` (Конкретное распределение)

- **Назначение**: Конкретный экземпляр распределения с определенными параметрами. Реализует протокол `Distribution`
- **Поля**:
  - `distr_name` - имя семейства

    - `distribution_type` - тип распределения
    - `parameters` - параметры распределения
- **Свойства**:
    - `family` - ссылка на родительское семейство

5.3.3.3. Типичные сценарии использования

Определение нового семейства распределений Пользователь создает новое параметрическое семейство, определяя:
- Различные параметризации (каноническая, mean-var, etc.)
- Ограничения на параметры
- Функции для вычисления характеристик (PDF, CDF, etc.), доступные аналитически
- Стратегии семплирования и вычислений

Создание экземпляра распределения Пользователь создает конкретное распределение, указывая:
- Имя семейства
- Параметризацию (как минимум одну)
- Значения параметров

Вычисление характеристик распределения Система автоматически:
- Проверяет корректность параметров
- При необходимости преобразует к базовой параметризации
- Вычисляет запрошенные характеристики

Генерация выборок Используется зарегистрированная стратегия семплирования для генерации данных из распределения.