

Санкт-Петербургский государственный университет

ГОЛОФАСТОВ Лев Дмитриевич

Выпускная квалификационная работа

**PySATL: Внедрение автоматического
анализа критериев в пакет проверки
гипотез согласия**

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2021 «Программная инженерия»*

Научный руководитель:
доцент кафедры системного программирования СПбГУ, к. ф.-м. н. Гориховский В. И.

Рецензент:
Инженер ключевых проектов, ООО «МПП АйТи Солюшнз», к. ф.-м. н. Пусев Р. С.

Санкт-Петербург
2025

Saint Petersburg State University

Lev Golofastov

Bachelor's Thesis

PySATL: Integration of an automatic
analysis of criteria into the goodness-of-fit
package

Education level: bachelor

Speciality *09.03.04 "Software Engineering"*

Programme *CB.5080.2021 "Software Engineering"*

Scientific supervisor:
C.Sc., docent V.I. Gorikhovskii

Reviewer:
Key projects engineer at "MPG IT Solutions LLC", C.Sc. R.S. Pusev

Saint Petersburg
2025

Оглавление

Введение	4
1. Обзор существующих решений	6
1.1. Существующие решения	6
1.2. Недостатки существующих решений	7
1.3. Как PySATL решает имеющиеся проблемы	8
2. Постановка задачи	10
3. PySATL-Criterion	11
3.1. Архитектура библиотеки	11
3.2. Пакет критериев экспоненциальности	13
4. Архитектура	16
4.1. Требования к фреймворку	16
4.2. Структура компонентов	19
5. Реализация	25
5.1. PySATL-Criterion	25
5.2. PySATL-Experiment	25
6. Аprobация	30
6.1. Аprobация на пакете экспоненциальных критериев . . .	30
6.2. Прочая аprobация	36
Заключение	37
Список литературы	38

Введение

Во многих практических задачах одним из ключевых шагов является проверка истинности некоторой гипотезы [31] [7] [25]. Одной из основных гипотез является предположение о структуре данных, то есть соотнесение этих данных с некоторым распределением. Для проверки таких гипотез существует гипотеза согласия.

Для проверки гипотез согласия используются критерии, которые представляют собой формализованные правила принятия решения об отклонении гипотезы. Эти критерии основаны на расчете статистик по выборочным данным. Например, для проверки гипотезы нормальности разработано множество критериев, более 40 из которых рассматриваются в обзоре [2].

Основной вопрос заключается в том, какой из критериев предпочтительнее использовать при решении конкретной задачи, поскольку результаты их работы зависят от параметров эксперимента, таких как уровень значимости, объем выборки, альтернативная гипотеза и её параметры. Кроме того, со временем разрабатываются новые критерии, которые подходят для специфичных задач и конкретных альтернатив, однако далеко не для всех из них имеется программная реализация.

В настоящее время существуют инструменты для проверки гипотез согласия, такие как SciPy.stats [26], Statsmodels.stats [28], пакеты R (stats, goftest) [24] [23] и другие. Не рассматривая современные алгоритмы, они предоставляют классические критерии, которые могут не подходить для решения имеющейся задачи, и не предлагают никакой аналитики для подбора критерия против правильной альтернативы.

Все это обуславливает необходимость создания state-of-art инструмента для проверки гипотез и инструмента для проведения анализа критериев для выбора лучшего или получения информации о невозможности проверки гипотезы.

Проверка гипотез согласия является одним из направлений проекта PySATL (Statistical Analysis, Testing and Learning)[20], где ключевыми проектами в данной области являются:

- PySATL-Criterion [21] — библиотека для проверки гипотез, где собираются современные критерии, имеется собранная аналитика по критериям с разными гиперпараметрами для принятия решений о выборе критерия;
- PySATL-Experiment [22] — фреймворк для бенчмаркинга статистических алгоритмов (в том числе критериев согласия) с помощью Монте–Карло моделирования.

В рамках данной работы предлагается автоматизировать процесс сравнения и бенчмаркинга критериев согласия в фреймворке PySATL-Experiment.

1. Обзор существующих решений

Существует ряд инструментов, которые предлагают функциональность для проверки гипотез согласия.

1.1. Существующие решения

1. Классические статистические пакеты

- **R (stats, goftest)**

Язык программирования R предоставляет пакеты, такие как *stats*, включающие функции для стандартных критериев согласия (например, критерий хи-квадрат Пирсона через *chisq.test*, критерий Колмогорова-Смирнова через *ks.test* и критерий Шапиро-Уилка для проверки нормальности через *shapiro.test*). Пакет *goftest* расширяет функциональность, добавляя такие критерии, как критерий Андерсона-Дарлинга, и инструменты визуализации. Однако эти средства не поддерживают встроенного анализа критериев или настройки параметров экспериментов (например, изменение объёма выборки, уровня значимости).

- **Python (SciPy, statsmodels)**

Библиотеки *scipy.stats* (критерии Колмогорова-Смирнова, хи-квадрат и другие) и *statsmodels* (критерии нормальности, такие как Лиллиефорса) позволяют проводить проверку гипотезы согласия, но фокусируются на отдельных критериях, а не на систематическом анализе критериев. Например, сравнение критериев для нормальности требует ручного сбора и анализа результатов, так как отсутствует инфраструктура для автоматического бенчмаркинга.

- **HypothesisTests.jl (Julia)**

Пакет Julia для проверки гипотез согласия с возможностью расширения. Однако экосистема Julia менее распространена,

а инструмент не поддерживает инфраструктуру для анализа критериев.

2. Специализированные платформы

- **MATLAB (Statistics and Machine Learning Toolbox)**

Содержит функции, такие как *chi2gof*, *kstest* и *lillietest*, но, как и в случае с R/Python, в нём отсутствует встроенная поддержка сравнения критериев в рамках единого фреймворка. Пользователи вынуждены вручную программировать цикличное применение критериев для их анализа.

- **SAS (PROC UNIVARIATE)**

Предоставляет критерии на нормальность (например, Шапиро-Уилка, Колмогорова-Смирнова), но ограничен проприетарным интерфейсом и слабой кастомизацией.

- **JMP (Distribution Platform)**

Коммерческое ПО с графическим интерфейсом для анализа распределений с автоматическим применением критериев. Однако анализ критериев требует ручного экспорта данных и постобработки.

- **Minitab, SPSS**

Коммерческое ПО с графическим интерфейсом, включающее базовые критерии согласия (например, хи-квадрат, Андерсона-Дарлинга). Однако в данном продукте отсутствует API для автоматического анализа критериев.

1.2. Недостатки существующих решений

1. **Отсутствие инфраструктуры для анализа критериев.** Большинство инструментов реализуют отдельные критерии, но не предоставляют инфраструктуры для систематического анализа. Пользователи вынуждены вручную настраивать эксперименты и агрегировать результаты.

2. **Ограниченная гибкость.** Даже в развитых средах, таких как R или Python, интеграция новых критериев требует разработки пользовательского кода, подключения к существующим конвейерам и валидации.
3. **Недостаточная поддержка параметров экспериментов.** Многие инструменты не позволяют легко варьировать следующие параметры:
 - Уровни значимости;
 - Объемы выборок;
 - Параметры альтернативных распределений;
 - Настройку генерации данных и визуализации.

1.3. Как PySATL решает имеющиеся проблемы

1.3.1. PySATL-Criterion

В PySATL имеется библиотека для проверки гипотез PySATL-Criterion. На момент начала работы в ней имеется лишь набор пакетов со статистиками критериев согласия для разных распределений.

Ее цель — стать одним из самых наполненных критериями инструментом для проверки гипотез согласия. Кроме того, для решения проблемы выбора правильного критерия против конкретной альтернативы библиотека должна расшириться базой знаний о критериях с разными гиперпараметрами: критическими значениями для разных размеров выборок, мощностью, чувствительностью к изменению параметров и т.д.

1.3.2. PySATL-Experiment

Также в PySATL ведутся работы по развитию фреймворка для бенчмаркинга статистических алгоритмов с помощью Монте – Карло моделирования. С помощью фреймворка планируется собирать аналитику по критериям согласия и расширять базу знаний PySATL-Criterion.

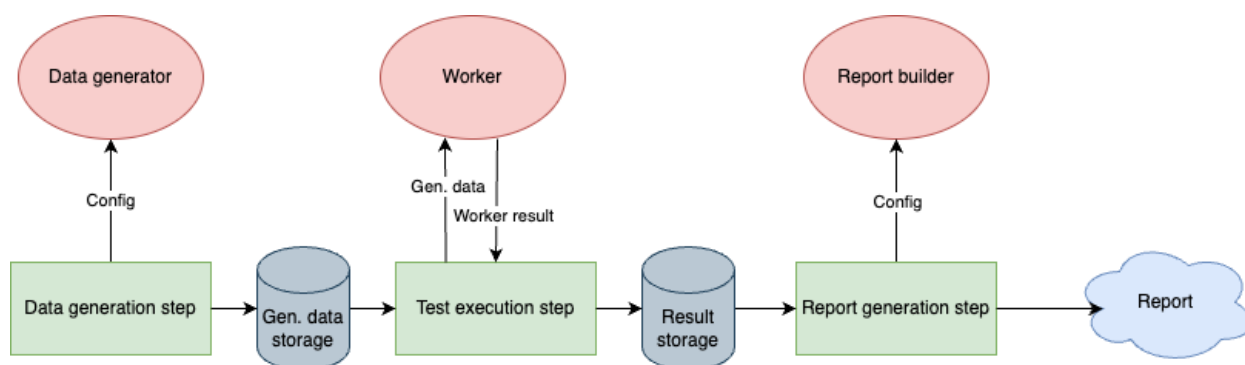


Рис. 1: Пайплайн эксперимента в PySATL-Experiment

На момент начала работы фреймворк предлагал лишь функциональность для проведения экспериментов по расчету мощностей критериев согласия против разных альтернатив.

Эксперимент устроен в формате некоторого пайплайна¹ (рис. 1) и состоит из трех ключевых этапов:

1. **Генерация случайных данных.** По заданной пользователем конфигурации генерируются данные. Формат данных зависит от реализации генератора. Например, в модуле проверки гипотез генерируются данные из конкретных распределений, в то время как в модуле для поиска разладки генерируются данные с разладкой и без. После генерации случайные выборки сохраняются в хранилище.
2. **Проведение полезной работы.** На этапе проведения полезной работы из хранилища случайных значений извлекаются выборки и подаются на вход worker'у расчета мощности. По окончании работы worker'а результаты сохраняются в хранилище.
3. **Создание отчета по результатам эксперимента.** После работы worker'а генерируется отчет в формате таблиц с мощностями.

¹https://github.com/PySATL/pysatl-experiment/blob/main/pysatl_flow.png

2. Постановка задачи

Целью работы является автоматизация сравнения и бенчмаркинга критериев в PySATL-Experiment. Для её выполнения были поставлены следующие задачи:

1. Разработка пакета критериев экспоненциальности для сбора требований к PySATL-Experiment.
2. Разработка новой архитектуры PySATL-Experiment.
3. Реализация оценки мета-параметров критериев: критические значения.
4. Реализация оценки качества критериев: мощность.
5. Реализация алгоритмического бенчмаркинга критериев: время работы.
6. Автоматизация анализа критериев, добавляемых в PySATL-Criterion.
7. Апробация на критериях экспоненциальности, нормальности, вейбульности.

3. PySATL-Criterion

Основная задача, которую решает библиотека PySATL-Criterion, — проверка гипотез согласия. Для этого в ней предоставлены различные критерии согласия для определенных распределений (нормальное, экспоненциальное, Вейбула и т.д.).

3.1. Архитектура библиотеки

При разработке архитектуры библиотеки учитывался контекст предметной области:

- Для проверки гипотезы необходимо иметь анализируемую выборку, критерий и уровень значимости (вероятность отклонить верную нулевую гипотезу).
- В общем случае критерий согласия представляет из себя пару «статистика-критическое значение». Проверить гипотезу можно через сравнение значения статистики и критического значения.
- Критическое значение — это квантиль порядка $(1 - \alpha)$ из предельного распределения статистики, где α — уровень значимости.
- Предельное распределение критерия может быть неизвестно. Его можно рассчитать, используя метод Монте – Карло.
- При расчете предельного распределения методом Монте – Карло критическое значение зависит от размера выборки, для которого проводились вычисления.

Также были рассмотрены случаи использования библиотеки (рис. 2). Основной пользователь — человек, целью которого является проверить гипотезу согласия. Кроме стандартных уровней значимости, пользователю может понадобиться специфичное значение.

Получившаяся архитектура библиотеки представлена на рисунке 3. Обсудим ее подробнее.

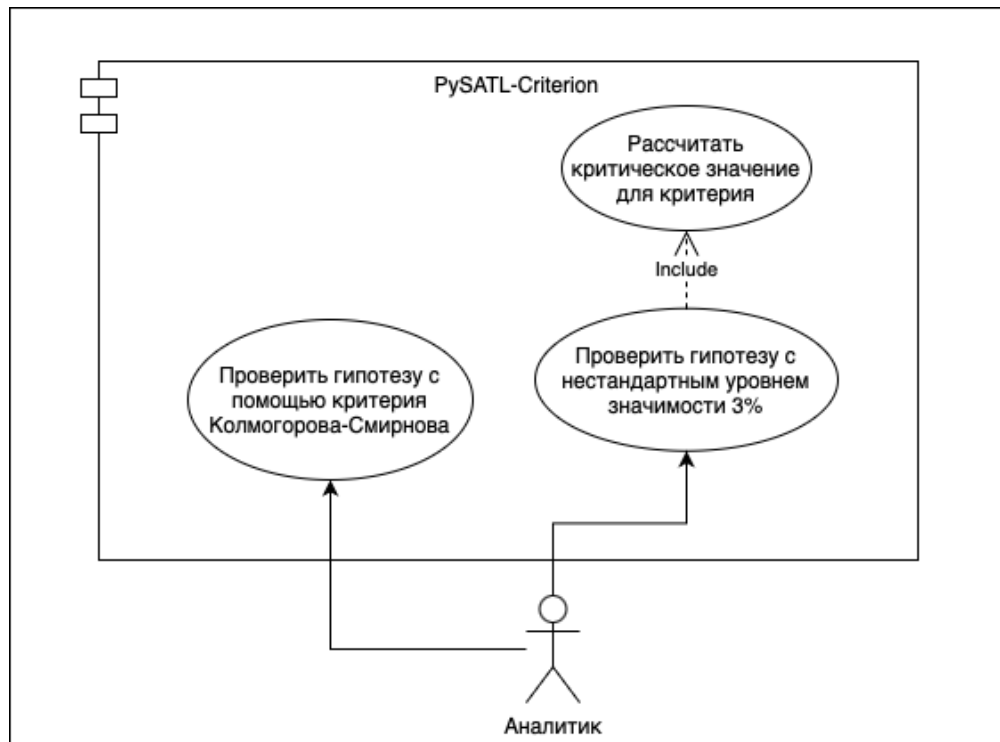


Рис. 2: Случаи использования PySATL-Criterion

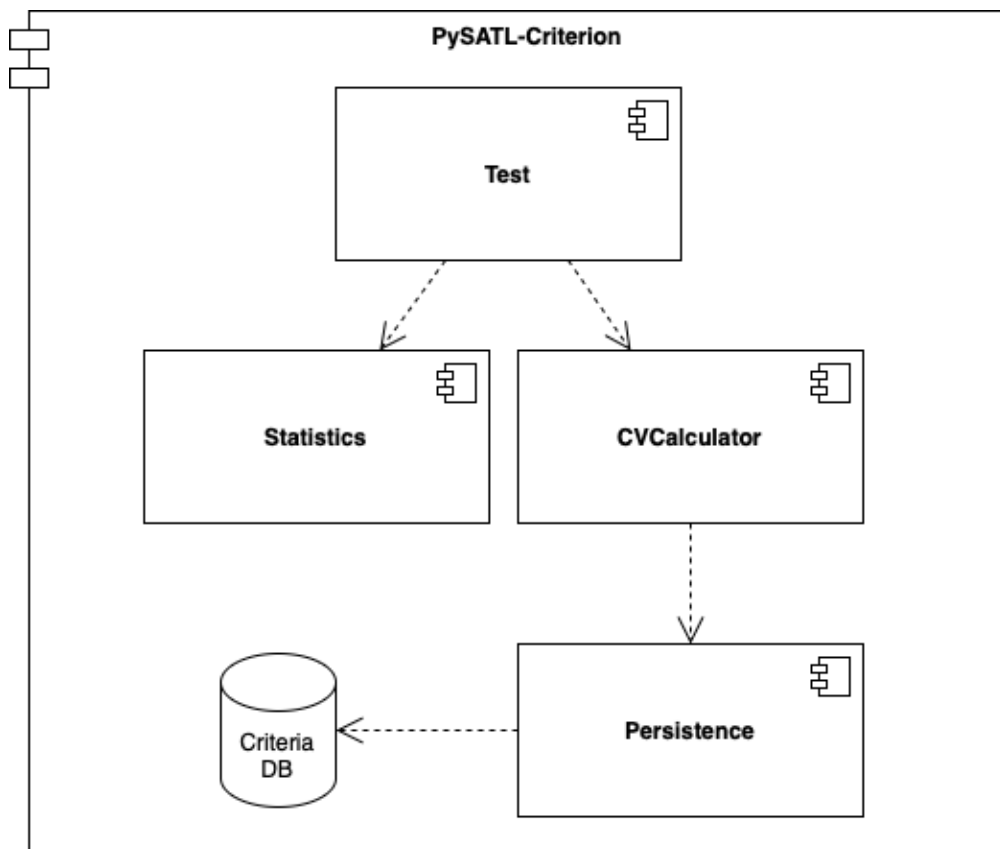


Рис. 3: Диаграмма компонентов PySATL-Criterion

3.1.1. База знаний о критериях

Имеется база данных с предельными распределениями статистик и некоторыми знаниями о критериях: мощности, время расчета статистик.

3.1.2. Persistence

Компонент Persistence предоставляет интерфейс и реализацию сущности базы данных с предельными распределениями.

3.1.3. CVCalculator

В компоненте CVCalculator содержится логика расчета критического значения по заданному уровню значимости и размеру выборки. Данный компонент использует предоставленный Persistence интерфейс базы для получения предельного распределения статистики.

3.1.4. Test

Компонент Test предоставляет сущность для проверки гипотезы с помощью заданных статистики из Statistics и уровня значимости. Также используется CVCalculator для расчета критического значения, которое зависит от размера выборки, предоставленной пользователем.

3.2. Пакет критериев экспоненциальности

Для сбора требований к PySATL-Experiment был разработан пакет с критериями экспоненциальности:

1. Epps – Pulley (EP) [5];
2. Kolmogorov – Smirnov (KS) [1];
3. Atkinson (ATK) [17];
4. Cox – Oakes (CO) [3];

5. Cramer-von Mises (CVM) [19];
6. Deshpande (DSP) [4];
7. Epstein (EPS) [6];
8. Frozini (FZ) [8];
9. Gini (Gini) [9];
10. Pietra (PT) [9];
11. Gnedenko (GD) [11];
12. Harris (HM) [16];
13. Hegazy – Green 1 (HG1) [13];
14. Hegazy – Green 2 (HG2) [13];
15. Kimber – Michael (KM) [14];
16. Kochar (KC) [15];
17. Lorenz (LZ) [10];
18. Moran (MN) [29];
19. Shapiro – Wilk (SW) [27];
20. WE test (WE) [12];
21. Wong – Wong (WW) [30].

Приведем некоторую классификацию критериев.

Решение об отвержении гипотезы принимается на основе некоторой информации по выборке. Источники информации бывают разные:

- **Критерии на основе эмпирической функции распределения (ЭФР):** Колмогорова–Смирнова (KS), Крамера–фон Мизеса (CVM), Фрозини (FZ), Хегази–Грина 1 (HG1), Хегази–Грина 2 (HG2), Кимбера–Майкла (KM).

- **Критерии на основе моментов распределения:** Аткинсона (АТК), Пьетры (РТ), WE критерий.
- **Критерии на основе порядковых статистик:** Эпштейна (EPS), Джини (Gini), Гнеденко (GD), Харриса (НМ), Вонга–Вонга (WW).
- **Критерии на основе специальных статистик:** Эппса–Палли (ЕР), Кокса–Оукса (СО), Дешпанде (DSP), Лоренца (LZ), Шапиро–Вилка (SW).
- **Критерии на основе энтропийных характеристик:** Морана (MN).
- **Критерии, использующие смешанные подходы:** Кочара (КС).

Статистики критериев разнообразны, и их вычислительные сложности тоже (таблица 1):

Сложность	Критерии
$O(n)$	АТК, РТ, WW, WE
$O(n \log n)$	KS, CVM, GD, HG1, HG2, KM LZ, SW, MN, KC, CO, FZ
$O(n^2)$	ЕР, EPS, Gini, DSP, НМ

Таблица 1: Классификация критериев по сложности вычисления статистик

4. Архитектура

4.1. Требования к фреймворку

Ранее был проведен ручной эксперимент по анализу критериев экспоненциальности на основе разработанного пакета, по результатам которого стало ясно, что критерии можно сравнивать по разным характеристикам, например:

- Время работы, в частности, время расчета статистики;
- Мощность;
- Количество используемой памяти;
- Чувствительность к размеру выборки;
- Чувствительность к изменению параметров;

Для проведения эксперимента по сравнению критериев по мощности было необходимо получить следующие данные:

- Сгенерированные выборки из целевого распределения разного размера, которые использовались для расчета значений статистик для поиска предельных распределений;
- Полученные методом Монте–Карло предельные распределения статистик критериев для расчета критических значений;
- Критические значения для возможности воспользоваться критерием;
- Сгенерированные выборки разного размера из альтернативных распределений для расчета мощностей критериев.

Эти данные необходимо куда-то сохранять и оттуда же получать, чтобы при запуске нового эксперимента не работать с имеющимися

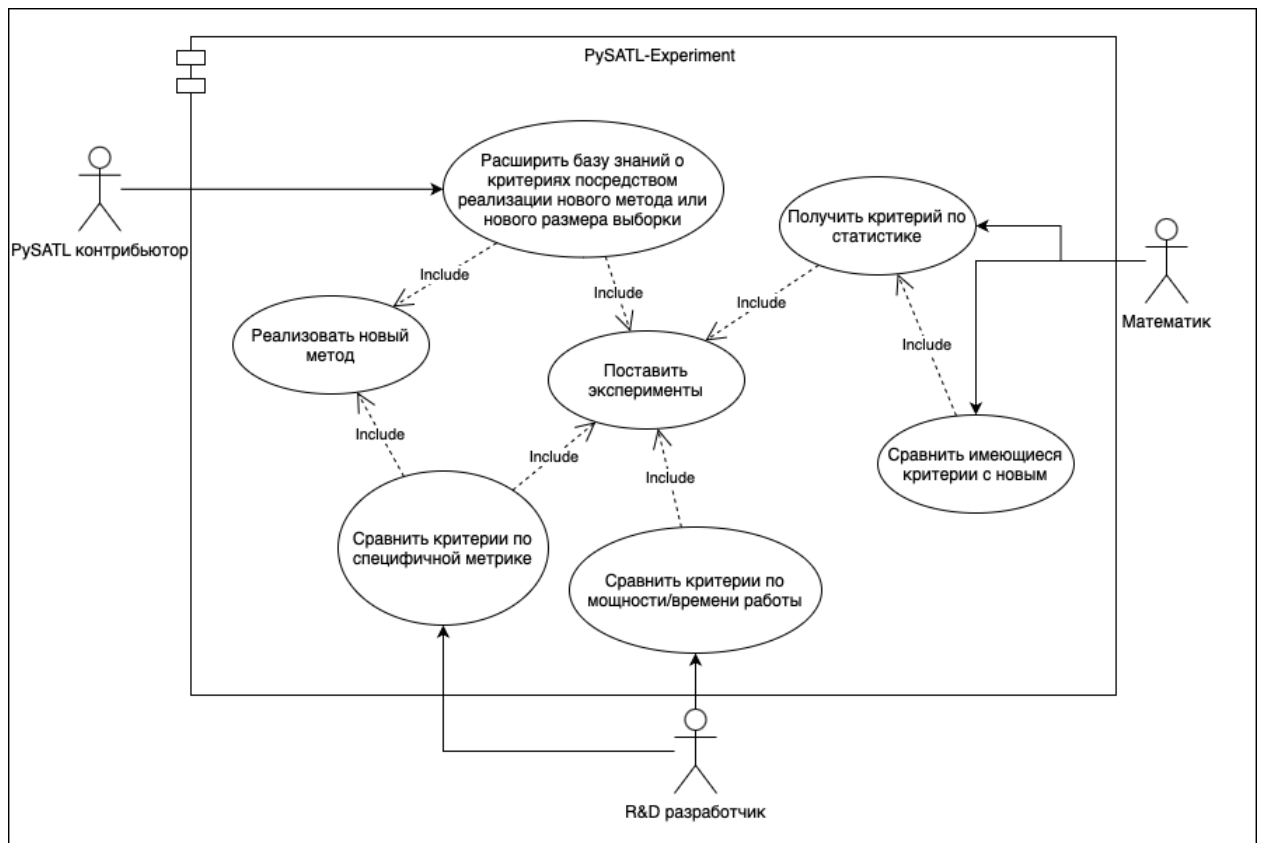


Рис. 4: Случаи использования PySATL-Experiment

данными заново, а также для того чтобы сравнение критериев было справедливым с точки зрения входных данных.

Сам эксперимент предполагает задание некоторых параметров, таких как набор критериев, уровни значимости, размеры выборок, количество Монте–Карло итераций и многие другие.

Кроме того, были рассмотрены несколько групп пользователей, которые потенциально могут обслуживаться PySATL-Experiment (рис. 4). Все они имеют свои цели:

1. Математики — после разработки нового критерия может возникнуть необходимость получить некоторую мета-информацию о нем. Например, критические значения для разных уровней значимости, мощности против различных альтернатив, время расчета статистики на разных размерах выборок.
2. R&D разработчики — во время решения какой-то исследовательской задачи может возникнуть необходимость проверки гипотезы

согласия. Для выбора наиболее подходящего критерия необходимо сравнить несколько из них по необходимой метрике на каких-то данных.

3. Пользователь, развивающий проект — человек, стремящийся расширить имеющуюся базу знаний о критериях в PySATL-Criterion, может предложить, например, новый способ анализа критериев или рассмотреть новую альтернативу и узнать, насколько точно критерии справляются с ее распознаванием.

Все эти рассуждения позволяют сформулировать требования к PySATL-Experiment:

1. Фреймворк должен поддерживать имеющийся пайплайн эксперимента.
2. Эксперимент должен быть хорошо конфигурируемым, обеспечивая возможность проверки разных гипотез. Данное требование подразумевает под собой, с одной стороны, гибкость в конфигурации отдельных этапов эксперимента (генерация случайных данных, полезная работа, создание отчета), а с другой — обработку взаимной зависимости данных этапов. В качестве примера рассмотрим два эксперимента: расчет критических значений для критерия согласия и сравнение критериев по мощности между собой. В первом случае требуется реализовать генератор случайных выборок из целевого распределения, соответствие которому проверяет критерий, во втором — генераторы выборок из многих альтернативных распределений для подсчета мощностей критериев. Использование первого генератора в эксперименте по подсчету мощностей критериев будет неосмысленным. То же касается сущностей по проведению полезной работы, генератора отчета и хранилищ. В этом смысле отдельные этапы эксперимента зависимы между собой. Нужно дать пользователю возможность ошибаться меньше и предоставить сущность для конфигурирования осмысленного эксперимента.

3. Хотя требование 1 подразумевает взаимную зависимость генератора случайных данных, сущности, выполняющей полезную работу и генератора отчета, должна быть обеспечена возможность их независимой разработки.
4. Должны быть предоставлены интерфейсы сущности для проведения полезной работы (далее — worker), генератора случайных данных, генератора отчета (далее — report builder), а также интерфейсы для реализации хранилищ случайных выборок и хранилищ результатов.
5. Компоненты системы должны быть расширяемыми.
6. Полученная информация должна быть сохранена в базу данных.
7. При наличии необходимой в рамках эксперимента информации в базе данных нужно пользоваться ей, если в конфигурации эксперимента не указано обратное.

4.2. Структура компонентов

В ходе работы была создана архитектура разрабатываемого фреймворка, удовлетворяющая требованиям к нему. PySATL-Criterion является подмодулем PySATL-Experiment, предоставляя имеющиеся функциональность и накопленную базу знаний о критериях.

В процессе проектирования была разработана диаграмма классов фреймворка для проработки деталей архитектуры. Посмотреть ее можно по ссылке².

На рисунке 5 изображена диаграмма компонентов проекта. Белым цветом помечены существовавшие компоненты, желтым — существовавшие, но переработанные, зеленым — новые.

В целом работа фреймворка и запуск эксперимента устроены следующим образом:

²https://drive.google.com/drive/folders/1vwo03SRZ-_lQwz6XmstV_Bx1q9YjQ7Ph?usp=sharing

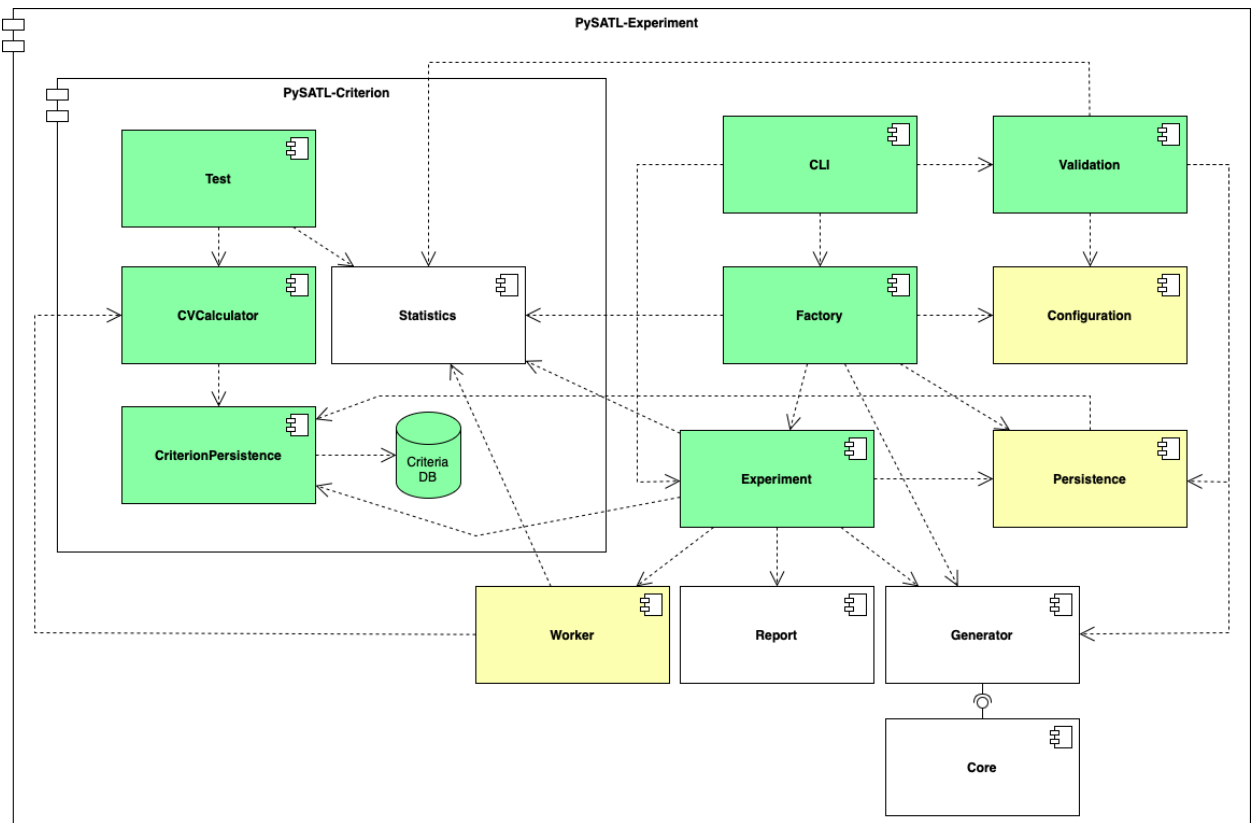


Рис. 5: Диаграмма компонентов PySATL-Experiment

1. Точка входа — CLI. Для начала необходимо сконфигурировать эксперимент, задав его параметры. Для этого имеются соответствующие команды. Заданные параметры валидируются с помощью функциональности из Validation, после чего формируется сущность из Configuration, представляющая из себя конфигурацию эксперимента.
2. Сборка эксперимента. Для сборки эксперимента используется абстрактная фабрика из Factory. Имея конфигурацию эксперимента после соответствующего процесса, фабрика собирает каждый из этапов пайплайна: в зависимости от наличия данных в базе, доступ к которой предоставляет Persistence, фабрика порождает сущности шагов пайплайна из Experiment, передавая в них необходимые параметры в специальном формате, и собирает их в единую сущность с шагами эксперимента. Для запуска этого процесса имеется соответствующая команда в CLI.

3. Запуск эксперимента. Имея сущность с шагами эксперимента после работы фабрики, он запускается, используя для генерации данных, полезной работы и построения отчета функциональность из `Generator`, `Worker` и `Report` соответственно. Во время выполнения каждого из шагов результаты сохраняются в базу данных, используя `Persistence`. Для запуска эксперимента имеется соответствующая команда в CLI.

Рассмотрим подробно каждый компонент.

4.2.1. Core

Ядро — это компонент, который будет предоставлять общую функциональность для остальных компонентов, используя `SciPy` и не только: генерацию выборок разных размеров и типов, расчет функции распределения, плотности распределения и так далее.

4.2.2. Persistence

Данный компонент предоставляет интерфейсы для различных видов хранилищ: конфигураций экспериментов, случайных выборок, мощностей, времени расчета статистик критериев. Все интерфейсы будут предоставлять необходимые методы для манипуляции с данными в необходимом формате.

4.2.3. Generator

Данный компонент предоставляет интерфейс абстрактного генератора и конкретные реализации этого интерфейса. В общем случае непонятно, какого формата будут генерируемые данные. Это определяет главный метод генерации случайных значений с заданным размером выборки.

В рамках задачи проверки гипотез согласия достаточно будет иметь генераторы выборок из разных типов распределений, которые будут пользоваться функциями генерации данных из `Core`.

4.2.4. Worker

Данный компонент предоставляет интерфейс абстрактного worker'а. Это сущность, которая будет выполнять полезную работу во время эксперимента. Полезная работа может быть абсолютно разной: расчет критических значений, мощности, времени работы статистик критериев. На самом деле, worker позволяет проводить бенчмаркинг любых статистических алгоритмов — нужно лишь реализовать главный метод.

Также Worker может пользоваться функциональностью расчета критических значений и расчета самой статистики критерия, предоставляемой компонентами CVCalculator и Statistics библиотеки PySATL-Criterion.

После выполнения работы worker будет создавать объект data-class'а нужного типа с полученными данными и возвращать его в качестве результата.

4.2.5. Report

Предоставлять интерфейс абстрактного report builder'а будет данный компонент. Конкретно реализованный report builder будет получать в качестве аргументов результаты работы worker'а и на их основе строить отчет.

4.2.6. Experiment

Данный компонент предоставляет инфраструктуру пайплайна эксперимента, в частности, функциональность его шагов и запуск.

Каждый из шагов имеет свою зону ответственности:

- **Генерация данных.** Получив в качестве аргументов генераторы выборок, их размеры и необходимое их количество, данный шаг проводит семплирование с помощью Generator, и сохраняет результаты в базу данных, пользуясь функциональностью из Persistence.

- **Выполнение полезной работы.** Получив в качестве аргументов `worker`, критерии и дополнительные необходимые параметры, данный шаг берет необходимые выборки из базы данных, запускает `worker` и сохраняет результаты в базу данных, пользуясь функциональностью из `Persistence`.
- **Создание отчета.** Получив в качестве аргументов `report builder` и дополнительные необходимые параметры, данный шаг берет результат работы `worker`'а из базы данных, генерирует отчет в необходимом формате с помощью `report builder`'а и сохраняет файл с ним.

4.2.7. Configuration

Данный компонент предоставляет `data-class`'ы для конфигурации эксперимента. Каждый тип эксперимента (расчет критических значений/мощности/времени) имеет разный набор параметров. Компонент позволяет сконфигурировать эксперимент каждого из этих типов и позволяет создавать новые.

4.2.8. Factory

Данный компонент предоставляет интерфейс абстрактной фабрики, ответственность которой заключается в сборке эксперимента по переданной ей конфигурации эксперимента в формате, описанном в `Configuration`.

Сборка эксперимента предполагает, в зависимости от его типа, проверку наличия данных в базе, чтобы не работать с данными заново, если в конфигурации не задано обратное, создание объектов статистик из `Statistics` и генераторов данных из `Generator`, в зависимости от конфигурации, и создание объектов шагов эксперимента из `Experiment`.

4.2.9. Validation

Данный компонент предоставляет функциональность для валидации каждого параметра задаваемой конфигурации эксперимента.

В частности, проверяет наличие заданных статистик в `Statistics`, а также корректность их выбора, в зависимости от проверяемой гипотезы, и наличие указанных генераторов альтернатив из `Generator`, если это необходимо для конкретного типа эксперимента.

В общем и целом, валидация подразумевает проверку совместимости параметров задаваемой конфигурации.

После проведения валидации конфигурация эксперимента сохраняется в базу данных.

4.2.10. CLI

Данный компонент предоставляет интерфейс командной строки для конфигурирования эксперимента с заданием каждого необходимого параметра с последующей валидацией с помощью функциональности из `Validation`, сборки и запуска. Сборка эксперимента предполагает создание фабрики из `Factory` и вызов ее метода сборки, а запуск — создание экземпляра эксперимента из `Experiment` и вызов его метода запуска.

5. Реализация

Реализация предполагала доработку PySATL-Criterion и полную переработку имеющейся кодовой базы PySATL-Experiment. Поговорим о каждом проекте отдельно.

5.1. PySATL-Criterion

На момент начала работы в библиотеке был только набор пакетов с реализованными статистиками критериев.

В ходе работы была создана инфраструктура для проверки гипотез:

1. Реализован компонент Persistence. В нем был добавлен интерфейс базы данных с предельными распределениями статистик для последующего расчета критических значений, а также проработаны модель данных в базе и модель формата запроса данных.
2. Реализован компонент CVCalculator и реализована функциональность расчета критических значений по предельному распределению, полученному из базы данных.
3. Реализован компонент Test и функциональность проверки гипотезы согласия с помощью критерия.

5.2. PySATL-Experiment

В фреймворке реализована автоматизация анализа критериев согласия посредством имплементации необходимой инфраструктуры, интерфейса командной строки для задания параметров эксперимента, его сборки и запуска, а также валидатора конфигурации эксперимента.

Поговорим о реализации каждого компонента отдельно.

5.2.1. Persistence

В рамках данного компонента добавлены интерфейсы баз данных с конфигурациями экспериментов, мощностями и временем расчета ста-

статистик критериев, а также проработаны соответствующие модели данных в базах и модели формата запроса данных.

5.2.2. Worker

Были реализованы worker'ы расчета критических значений, мощностей и времени расчета статистик с возвращением результата работы в соответствующем контейнере.

5.2.3. Experiment

Были реализованы шаги эксперимента:

1. Генератор данных, который запускает семплирование данных с помощью реализаций генераторов случайных выборок из Generator и сохраняет выборки в базу данных.
2. Executor, который берет выборки из базы случайных значений, запускает worker с необходимой конфигурацией и сохраняет результаты работы в базу данных. Поскольку эксперименты разных типов имеют разный набор параметров, были реализованы executor'ы для экспериментов по расчету критических значений, мощностей, времени работы статистик.
3. Генератор отчета, который на основе результатов работы worker'а из базы данных с помощью report builder'а строит отчет по эксперименту. Подобно executor'у, были реализованы генераторы отчетов для экспериментов по расчету критических значений, мощностей, времени работы статистик.

5.2.4. Configuration

Были проработаны и добавлены модели данных конфигурации эксперимента для каждого типа (расчет критических значений, мощностей, времени работы статистик).

5.2.5. Factory

Была добавлена абстрактная фабрика и конкретные ее реализации для каждого типа эксперимента (расчет критических значений, мощностей, времени работы статистик).

Принцип работы фабрики а основе передаваемой ей конфигурации эксперимента таков:

1. Проверяется завершенность каждого этапа ранее запускавшегося эксперимента, если он был прерван.
2. Проверяется наличие заданных выборок в хранилище случайных значений. При отсутствии необходимых данных конфигурируется генератор данных как шаг пайплайна. Конфигурация включает в себя создание необходимых классов генераторов случайных выборок из `Generator` и передачу необходимых параметров.
3. Проверяется наличие результатов от предыдущих экспериментов в соответствующих хранилищах. При отсутствии необходимых данных конфигурируется `executor` как шаг пайплайна. Конфигурация включает в себя создание необходимых объектов классов статистик из `Statistics`, объекта `worker`'а и передачу необходимых параметров.
4. Конфигурируется шаг генератора отчета.
5. Результатом работы фабрики является сконфигурированный объект `data-class`'а, собирающий в себе все шаги эксперимента.

5.2.6. Validation

Был реализован валидатор вводимых пользователем команд с информативными сообщениями об ошибках.

Валидатор также проверяет наличие реализаций статистик критериев и генераторов случайных выборок.

По завершении всей необходимой валидации собирается объект `data-class`'а конфигурации эксперимента. При наличии эксперимента в базе

конфигураций, делается проверка на завершенность каждого этапа. Если все этапы уже пройдены, эксперимент не запускается вовсе. Если заданной конфигурации нет, она сохраняется в базу данных под новым идентификатором.

5.2.7. CLI

Был реализован интерфейс командной строки для конфигурирования эксперимента. При успешной валидации позволяет собрать эксперимент с помощью фабрики и запустить его.

Команды CLI:

- set-experiment-type
- set-storage-connection
- set-run-mode
- set-hypothesis
- set-generator-type
- set-executor-type
- set-report-builder-type
- set-sample-sizes
- set-monte-carlo-count
- set-criteria
- set-significance-levels
- set-alternatives
- set-result-path
- configure

- build
- run

6. Апробация

6.1. Апробация на пакете экспоненциальных критериев

В качестве апробации реализации PySATL-Experiment был проведен эксперимент по расчету мощностей критериев экспоненциальности.

6.1.1. Конфигурация эксперимента по расчету мощности

Эксперимент состоит из следующих этапов:

1. *Выбор альтернативных распределений.* Рассматриваются две группы распределений: симметричные и асимметричные.
2. *Генерация выборок из альтернативных распределений.* Генерируются по 1000 выборок из альтернативных распределений размерами 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000.
3. *Расчет предельных распределений статистик критериев.* Для статистики каждого критерия было рассчитано ее предельное распределение методом Монте-Карло. Для нулевой гипотезы (экспоненциального распределения с параметром $\lambda = 1$) было сгенерировано 100000 выборок упомянутых выше размеров и рассчитана статистика на каждой из них.
4. *Расчет критических значений.* Были получены критические значения из предельных распределений статистик для уровня значимости 0,05.
5. *Расчет мощности.* Расчет мощности производился для каждой альтернативы.

Симметричные альтернативные распределения, рассмотренные в рамках данного эксперимента:

- три случая распределения Бета $Beta(a; b)$: $Beta(0,5; 0,5)$, $Beta(1; 1)$ и $Beta(2; 2)$, где a и b – параметры формы;
- три случая распределения Коши $Cauchy(t; s)$: $Cauchy(0; 0,5)$, $Cauchy(0; 1)$ и $Cauchy(0; 2)$, где t и s – параметры положения и масштаба;
- один случай распределения Лапласа $Laplace(t; s)$: $Laplace(0; 1)$, где t и s – параметры положения и масштаба;
- один случай логистического распределения $Logistic(t; s)$: $Logistic(2; 2)$, где t и s – параметры положения и масштаба;
- четыре случая распределения Стьюдента $Student(v)$: $Student(1)$, $Student(2)$, $Student(4)$ и $Student(10)$, где v – число степеней свободы;
- пять случаев распределения Тьюки $Tukey(A)$: $Tukey(0.14)$, $Tukey(0,5)$, $Tukey(2)$, $Tukey(5)$ и $Tukey(10)$, где A – параметр формы;
- пять случаев нормального распределения $N(m; v)$: $N(0; 1)$, $N(0; 4)$, $N(4; 1)$, $N(-4; 1)$, $N(0; 0,5)$, где m и v – параметры положения и масштаба.

Асимметричные альтернативные распределения, рассмотренные в рамках данного эксперимента:

- четыре случая распределения Бета $Beta(a; b)$: $Beta(2; 1)$, $Beta(2; 5)$, $Beta(4; 0,5)$ и $Beta(5; 1)$;
- три случая распределения Хи-квадрат $\chi^2(v)$: $\chi^2(1)$, $\chi^2(4)$ и $\chi^2(10)$, где v – число степеней свободы;
- шесть случаев распределения Гамма $Gamma(a; b)$: $Gamma(2; 2)$, $Gamma(3; 2)$, $Gamma(5; 1)$, $Gamma(9; 1)$, $Gamma(15; 1)$ и $Gamma(100; 1)$, где a и b – параметры формы и масштаба;
- один случай распределения Гумбеля $Gumbel(t; s)$: $Gumbel(1; 2)$, где t и s – параметры положения и масштаба;

- один случай логнормального распределения $Lognormal(t; s): LN(0; 1)$, где t и s – параметры положения и масштаба;
- четыре случая распределения Вейбулла $Weibull(a; b): Weibull(0,5; 1)$, $Weibull(1; 2)$, $Weibull(2; 3,4)$ и $Weibull(3; 4)$, где a и b – параметры формы и масштаба;
- три случая экспоненциального распределения $Exp(l): Exp(1)$, $Exp(4)$, и $Exp(10)$, где l – это обратный коэффициент масштаба;
- пять случаев распределения Гомпертца $Gompertz(e; b): Gompertz(0,1; 1)$, $Gompertz(2; 1)$, $Gompertz(3; 1)$, $Gompertz(1; 2)$, $Gompertz(1; 3)$, где e и b – параметры формы и масштаба.

6.1.2. Результаты

На основе результатов эксперимента были посчитаны средние мощности критериев на объемах выборки 10, 100, 1000, отсортированы в порядке невозрастания и выделены 10 лучших критериев. Наиболее точные критерии против симметричных и асимметричных альтернативных распределений представлены в таблицах 2 и 3 соответственно. Также в таблице 4 представлена информация о наиболее точных критериях против всех альтернатив.

№	10	100	1000
1	FZ (0.921)	CVM (1.000)	CVM (1.000)
2	CVM (0.916)	HG1 (1.000)	HG1 (1.000)
3	KS (0.905)	HG2 (1.000)	HG2 (1.000)
4	HG1 (0.766)	FZ (1.000)	FZ (1.000)
5	EP (0.757)	KS (1.000)	KS (1.000)
6	WE (0.731)	GD (0.992)	GD (1.000)
7	SW (0.599)	SW (0.982)	SW (0.994)
8	GD (0.582)	HM (0.853)	HM (0.871)
9	KC (0.544)	EP (0.773)	EP (0.773)
10	LZ (0.529)	WE (0.773)	WE (0.773)

Таблица 2: Топ критериев против симметричных альтернатив

№	10	100	1000
1	FZ (0.543)	HG1 (0.926)	HG1 (0.964)
2	CVM (0.540)	HG2 (0.862)	HG2 (0.963)
3	CO (0.531)	FZ (0.834)	CVM (0.890)
4	MN (0.525)	CVM (0.830)	KS (0.890)
5	KC (0.520)	KS (0.811)	FZ (0.890)
6	LZ (0.519)	KM (0.792)	KM (0.851)
7	KS (0.512)	ATK (0.789)	ATK (0.847)
8	ATK (0.507)	SW (0.753)	EPS (0.787)
9	KM (0.495)	GD (0.737)	SW (0.774)
10	GD (0.474)	KC (0.719)	GD (0.774)

Таблица 3: Топ критериев против асимметричных альтернатив

№	10	100	1000
1	FZ (0.716)	HG1 (0.960)	HG1 (0.980)
2	CVM (0.712)	HG2 (0.925)	HG2 (0.980)
3	KS (0.692)	FZ (0.910)	CVM (0.941)
4	HG1 (0.577)	CVM (0.908)	KS (0.941)
5	KC (0.531)	KS (0.897)	FZ (0.940)
6	GD (0.524)	SW (0.858)	GD (0.878)
7	LZ (0.523)	GD (0.854)	SW (0.875)
8	KM (0.502)	HM (0.722)	HM (0.761)
9	SW (0.454)	KM (0.699)	KM (0.729)
10	HG2 (0.418)	KC (0.660)	LZ (0.672)

Таблица 4: Топ критериев против всех альтернатив

Также были построены тепловые карты мощностей критериев против рассмотренных альтернатив для размеров выборок 10, 100, 1000 на рисунках 6, 7 и 8 соответственно.

Основные наблюдения:

1. Мощность критериев с ростом объема выборки в среднем возрастает.
2. Уже на объеме выборки 100 критерии хорошо распознают альтернативу.
3. Во всех сводных таблицах первые пять лучших критериев повторяются. Более того, у каждого из них источником информации

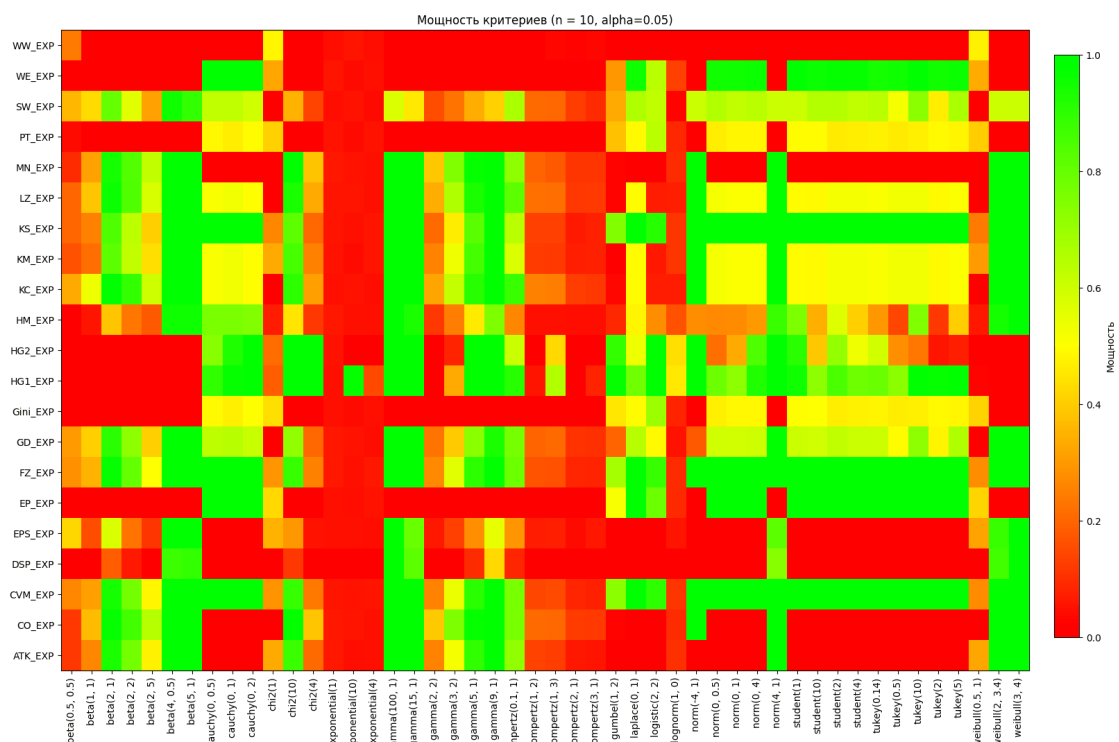


Рис. 6: Мощности критериев на объеме выборки 10

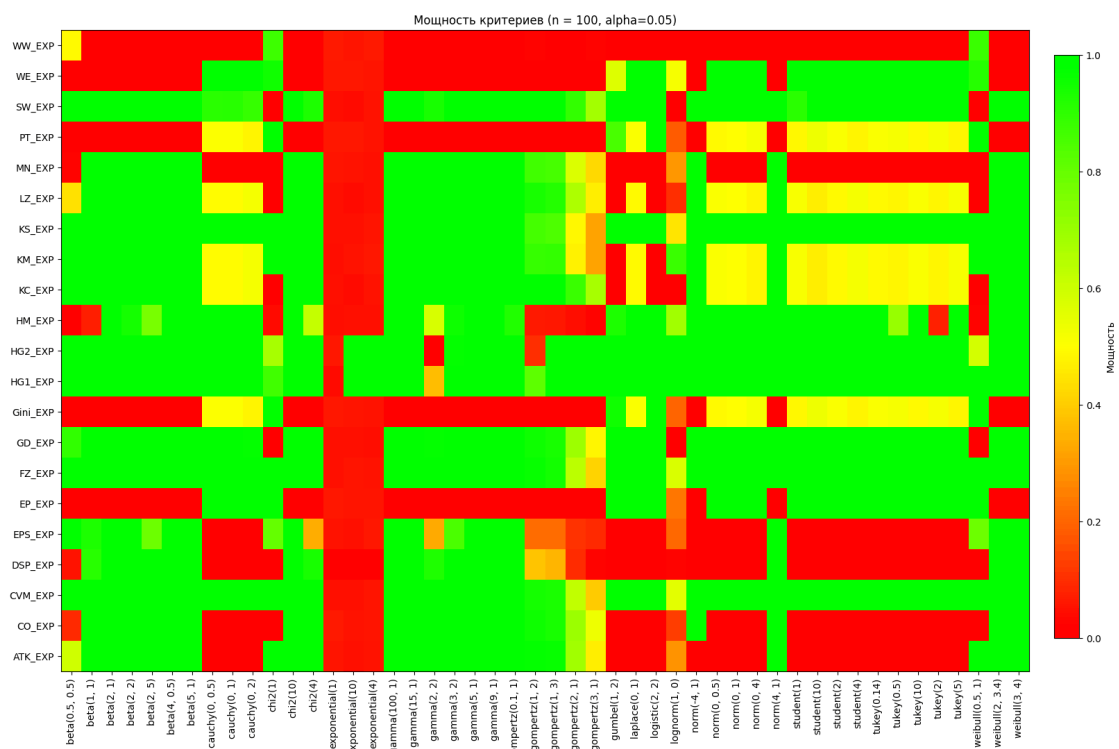


Рис. 7: Мощности критериев на объеме выборки 100



Рис. 8: Мощности критериев на объеме выборки 1000

для принятия решения об отклонении нулевой гипотезы является эмпирическая функция распределения, в то время как остальные критерии имеют другие источники информации.

4. Против асимметричных распределений ни один критерий не смог в 100% случаев отличить альтернативу. Это связано с тем, что в эксперимент было включено экспоненциальное распределение. Однако критерии HG1 и HG2 способны с какого-то момента отвергать экспоненциальное распределение, если параметр сильно отличается от нулевой гипотезы.
5. Несмотря на то, что в список лучших критериев попали критерии с линейной ($O(n)$) вычислительной сложностью расчета статистики, лучшими остаются критерии со сложностью вычисления статистики $O(n \log n)$.
6. В целом критерий HG1 показал лучшие результаты.

Полученные результаты экспериментального сравнения представлены на конференции CPS.³

6.2. Прочая апробация

6.2.1. Апробация на критериях нормальности

Совместно с Мироновым Алексеем по его пакету критериев нормальности⁴ был проведен эксперимент по их анализу. Описание PySATL-Experiment и результаты экспериментального сравнения были представлены на конференции FRUCT⁵ и опубликованы [18].

6.2.2. Апробация на критериях вейбульности

Миронов Алексеем по его пакету критериев вейбульности⁶ провел эксперимент по их анализу. Результаты экспериментального сравнения были представлены на конференции CPS.

6.2.3. Апробация на графовых критериях

Похабов Иван провел эксперимент по анализу графовых критериев. Результаты экспериментального сравнения были представлены на конференции CPS.

³<http://cpsconf.ru/news/>

⁴https://github.com/PySATL/pysatl-criterion/blob/main/pysatl_criterion/normal.py

⁵<https://fruct.org/publications/volume-37/fruct37/>

⁶https://github.com/PySATL/pysatl-criterion/blob/main/pysatl_criterion/weibull.py

Заключение

В рамках работы были получены следующие результаты:

1. Реализованы 21 критерий экспоненциальности для сбора требований и последующей апробации фреймворка в виде эксперимента по расчету мощности.
2. Разработана архитектура фреймворка PySATL-Experiment с учетом требований и специфики предметной области.
3. Реализована функциональность оценки мета-параметров критериев, в частности, критических значений.
4. Реализована функциональность оценки качества критериев, в частности, мощности.
5. Реализована функциональность алгоритмического бенчмаркинга критериев, в частности, по времени работы.
6. Автоматизирован анализ критериев, добавляемых в библиотеку проверки гипотез PySATL-Criterion, посредством реализации удобной конфигурации эксперимента, имеющего конкретную цель, результат работы которого представляется в удобной форме.
7. Проведена апробация реализации на критериях нормальности, экспоненциальности, вейбульности и графовых критериях посредством проведения экспериментов по расчету мощности. Результаты экспериментов и описание фреймворка PySATL-Experiment были приняты к публикации на конференциях CPS и FRUCT.

Код реализации в PySATL-Criterion⁷ и PySATL-Experiment⁸ доступен в репозиториях на GitHub.

⁷<https://github.com/PySATL/pysatl-criterion>

⁸<https://github.com/PySATL/pysatl-experiment>

Список литературы

- [1] An Kolmogorov. Sulla determinazione empirica di una legge di distribuzione // Giorn Dell'inst Ital Degli Att. — 1933. — Vol. 4. — P. 89–91.
- [2] Arnastauskaitė Jurgita, Ruzgas Tomas, Bražėnas Mindaugas. An exhaustive power comparison of normality tests // Mathematics. — 2021. — Vol. 9, no. 7. — P. 788.
- [3] Cox D.R., Oakes D. Analysis of Survival Data. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. — Taylor & Francis, 1984. — ISBN: 9780412244902. — URL: <https://books.google.ru/books?id=Y4pdM2soP4IC>.
- [4] DESHPANDE JAYANT V. A class of tests for exponentiality against increasing failure rate average alternatives // Biometrika. — 1983. — Vol. 70, no. 2. — P. 514–518.
- [5] Epps Thomas W, Pulley Lawrence B. A test for normality based on the empirical characteristic function // Biometrika. — 1983. — Vol. 70, no. 3. — P. 723–726.
- [6] Epstein Benjamin. Tests for the validity of the assumption that the underlying distribution of life is exponential part I // Technometrics. — 1960. — Vol. 2, no. 1. — P. 83–101.
- [7] Evaluating the statistical power of goodness-of-fit tests for health and medicine survey data / Michael Steele, Neil A. Smart, Cameron P. Hurst, Janet Chaseling. — 2009.
- [8] Frozini BV. On the distribution and power of goodness-of-fit statistic with parametric applications, “Goodness-of-fit”/Ed. by Revesz P // Sarkadi K, Sen PK—Amsterdam-Oxford-New York: North Holland Publ. Comp. — 1987. — P. 123–154.

- [9] Gail Mitchell H, Gastwirth Joseph L. A scale-free goodness-of-fit test for the exponential distribution based on the Gini statistic // Journal of the Royal Statistical Society Series B: Statistical Methodology. — 1978. — Vol. 40, no. 3. — P. 350–357.
- [10] Gail Mitchell H, Gastwirth Joseph L. A scale-free goodness-of-fit test for the exponential distribution based on the Lorenz curve // Journal of the American Statistical Association. — 1978. — Vol. 73, no. 364. — P. 787–793.
- [11] Gnedenko B.V., Beliaev I.U.K., Solov'ev A.D. Mathematical Methods of Reliability Theory. Probability and mathematical statistics. — Academic Press, 1969. — ISBN: 9780122872501. — URL: <https://books.google.ru/books?id=a7c8AAAAIAAJ>.
- [12] Hahn G.J., Shapiro S.S. Statistical Models in Engineering. Systems Engineering Series. — Wiley, 1967. — ISBN: 9780471339151. — URL: <https://books.google.ru/books?id=4WzbAAAAMAAJ>.
- [13] Hegazy YAS, Green JR. Some new goodness-of-fit tests using order statistics // Journal of the Royal Statistical Society Series C: Applied Statistics. — 1975. — Vol. 24, no. 3. — P. 299–308.
- [14] Kimber AC. Tests for the exponential, Weibull and Gumbel distributions based on the stabilized probability plot // Biometrika. — 1985. — Vol. 72, no. 3. — P. 661–663.
- [15] Kochar Subhash C. Testing exponenttality against monotone failure rate average // Communications in Statistics-Theory and methods. — 1985. — Vol. 14, no. 2. — P. 381–392.
- [16] Lin Chine-Chuong, Mudholkar Govind S. A test of exponentality based on the bivariate F distribution // Technometrics. — 1980. — Vol. 22, no. 1. — P. 79–82.

- [17] Mimoto Nao, Zitikis Ričardas. The Atkinson index, the Moran statistic, and testing exponentiality // Journal of the Japan statistical society. — 2008. — Vol. 38, no. 2. — P. 187–205.
- [18] Mironov Alexey, Golofastov Lev, Gorikhovskii Viacheslav. [The PYSATL Experiment Framework](#) // 2025 37th Conference of Open Innovations Association (FRUCT). — 2025. — P. 195–202.
- [19] Modified Cramér-von Mises and Anderson-Darling tests for weibull distributions with unknown location and scale parameters / John G Bush, Brian W Woodruff, Albert H Moore, Edward J Dunne // Communications in Statistics-Theory and Methods. — 1983. — Vol. 12, no. 21. — P. 2465–2476.
- [20] PySATL. — URL: <https://github.com/PySATL> (дата обращения: 28.05.25).
- [21] PySATL-Criterion. — URL: <https://github.com/PySATL/pysatl-criterion> (дата обращения: 28.05.25).
- [22] PySATL-Experiment. — URL: <https://github.com/PySATL/pysatl-experiment> (дата обращения: 28.05.25).
- [23] R package goftest. — URL: <https://github.com/baddstats/goftest> (дата обращения: 28.05.25).
- [24] R package stats. — URL: <https://www.rdocumentation.org/packages/stats> (дата обращения: 28.05.25).
- [25] Sawyer Alan G., Page Thomas J. The use of incremental goodness of fit indices in structural equation models in marketing research // [Journal of Business Research](#). — 1984. — Vol. 12, no. 3. — P. 297–308.
- [26] SciPy.stats. — URL: <https://docs.scipy.org/doc/scipy/reference/stats.html#> (дата обращения: 28.05.25).

- [27] Shapiro Samuel S, Wilk MB. An analysis of variance test for the exponential distribution (complete samples) // Technometrics. — 1972. — Vol. 14, no. 2. — P. 355–370.
- [28] Statsmodels.stats. — URL: <https://www.statsmodels.org/dev/stats.html> (дата обращения: 28.05.25).
- [29] Tchirina AV. Bahadur efficiency and local optimality of a test for exponentiality based on the Moran statistics // Journal of Mathematical Sciences. — 2005. — Vol. 127, no. 1. — P. 1812–1819.
- [30] Wong PG, Wong SP. An extremal quotient test for exponential distributions // Metrika. — 1979. — Vol. 26. — P. 1–4.
- [31] The role of the Weibull distribution in modelling traffic in Internet access and backbone core networks / Asad Arfeen, Krzysztof Pawlikowski, Don McNickle, Andreas Willig // Journal of network and computer applications. — 2019. — Vol. 141. — P. 1–22.