АЛГОРИТМЫ АМПЕЛОМЕТРИИ. Том–2. Алгоритмы 12–19: Репрезентативность выборочных показателей

Preprint · August 2025

DOI: 10.13140/RG.2.2.24208.11521

2 authors, including:

Eugene Veniaminovich Lutsenko
Kuban State Agrarian University
1,152 PUBLICATIONS **993** CITATIONS

SEE PROFILE

Е. В. Луценко, Л. П. Трошин

АЛГОРИТМЫ АМПЕЛОМЕТРИИ

Том-2.
Алгоритмы 12-19: Репрезентативность выборочных показателей

Учебное пособие

Краснодар - 2025

МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «Кубанский государственный аграрный университет
имени И. Т. Трубилина»

Е.В. Луценко, Л.П. Трошин

АЛГОРИТМЫ АМПЕЛОМЕТРИИ

Том-2.
Алгоритмы 12-19: Репрезентативность выборочных показателей

Учебное пособие

Краснодар
КубГАУ
2025

УДК 634.84
ББК 42.36
Л86

Р е ц е н з е н т ы :

В. В. Лиховской – директор Института винограда и вина «Магарач» РАН РФ, д-р с.-х. наук, Ялта;

В. С. Петров – главный научный сотрудник Северо-Кавказского федерального научного центра садоводства, виноградарства, виноделия, д-р с.-х. наук, Краснодар.

Луценко Е.В., Трошин Л.П.

Л86 Алгоритмы ампелометрии: учебное пособие // Е. В. Луценко, Л. П. Трошин. Учебное пособие. Том 2-й. Алгоритмы 12-19: Репрезентативность выборочных показателей. – Краснодар : КубГАУ, 2025. – 221 с.

ISBN 978-5-93856-992-8

В учебном пособии дана четкая последовательность биометрических расчетов ампелометрических измерений листьев различных фенотипов Vitis vinifera L. За основу взята классическая работа: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. М., Изд-во Моск, ун-та. 1980, 21004, 150 с. , http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

Учебное пособие, 2-й том которого перед вами, подготовлено в рамках реализации программы развития Кубанского ГАУ «Приоритет 2030» (стратегический проект «Генетика и селекция в растениеводстве») и включает 7 томов: 1. Групповые свойства: средний уровень, разнообразие, распределение (алгоритмы 01-11), 2. Репрезентативность выборочных показателей (алгоритмы 12-19), 3. Анализ коррелятивных связей (алгоритмы 20-24), 4. Дисперсионный анализ одно- и двухфакторных комплексов для количественных и качественных признаков (алгоритмы 25-41), 5. Регрессионный анализ и математические модели биологических состояний и процессов (алгоритмы 42-50), 6. Информационные показатели в ампелометрии (алгоритмы 51-60), 7. Автоматизированный системно-когнитивный анализ (АСК-анализ) и система "Эйдос" в ампелометрии (алгоритмы-61-70).

Предназначено для профессионалов и любителей культуры винограда, студентов-бакалавров, магистрантов, аспирантов и докторантов биологических и агрономических специальностей.

УДК 634.84
ББК 42.36

ISBN 978-5-93856-992-8

Введение

Во втором томе учебного пособия «Алгоритмы ампелометрии» рассмотрены

Алгоритмы 12-19: Репрезентативность выборочных показателей.

Алгоритм-12: Определение доверительных границ генеральных параметров.

Алгоритм-13: Оценка разности выборочных средних.

Алгоритм-14: Критерий достоверности разности (Критерий Стьюдента).

Алгоритм-15: Критерий Фишера.

Алгоритм-16: Критерий Бейли.

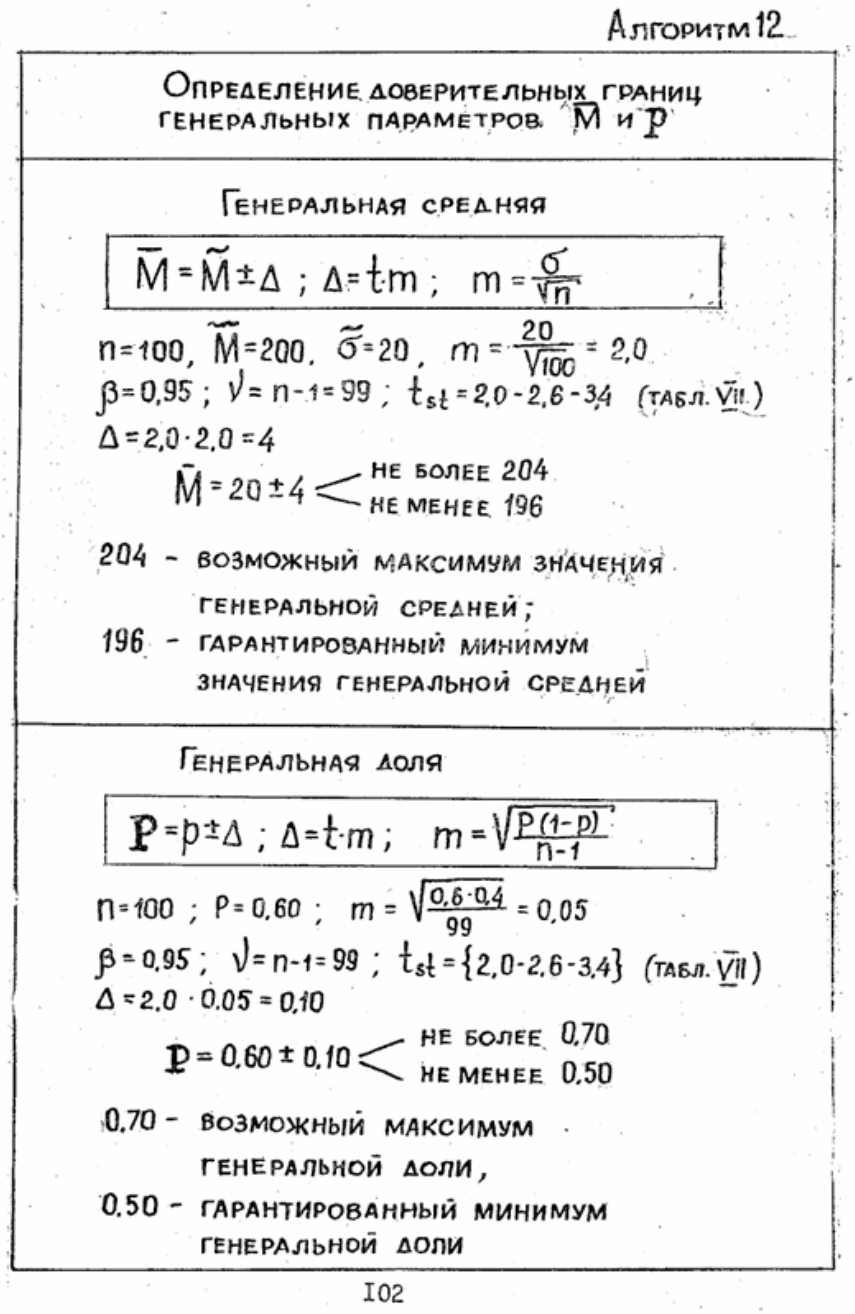
Алгоритм-17: Достоверность разности выборочных долей.

Алгоритм-18: Критерий «Фи» Фишера.

Алгоритм-19: Достоверность разности между выборочной и генеральной долями.

Алгоритм-12: Определение доверительных границ генеральных параметров

1. Исходное изображение



Алгоритм 12

Определение доверительных границ генеральных параметров $\bar{M}$ и P

Генеральная средняя

$$\bar{M} = \tilde{M} \pm \Delta \; ; \; \Delta = t \cdot m \; ; \; m = \frac{\sigma}{\sqrt{n}}$$

$n = 100$, $\tilde{M} = 200$, $\tilde{\sigma} = 20$, $m = \frac{20}{\sqrt{100}} = 2{,}0$

$\beta = 0{,}95$; $\nu = n - 1 = 99$; $t_{st} = 2{,}0 - 2{,}6 - 3{,}4$ (табл. VII)

$\Delta = 2{,}0 \cdot 2{,}0 = 4$

$\bar{M} = 20 \pm 4$ < не более 204, не менее 196

204 – возможный максимум значения генеральной средней;

196 – гарантированный минимум значения генеральной средней

Генеральная доля

$$P = p \pm \Delta \; ; \; \Delta = t \cdot m \; ; \; m = \sqrt{\frac{P(1-p)}{n-1}}$$

$n = 100$; $P = 0{,}60$; $m = \sqrt{\frac{0{,}6 \cdot 0{,}4}{99}} = 0{,}05$

$\beta = 0{,}95$; $\nu = n - 1 = 99$; $t_{st} = \{2{,}0 - 2{,}6 - 3{,}4\}$ (табл. VII)

$\Delta = 2{,}0 \cdot 0{,}05 = 0{,}10$

$P = 0{,}60 \pm 0{,}10$ < не более 0,70, не менее 0,50

0,70 – возможный максимум генеральной доли,

0,50 – гарантированный минимум генеральной доли

102

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980, 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму, в т.ч. используемые в формулах условные математические обозначения переменных

Представленное изображение описывает алгоритм определения доверительных границ для двух ключевых статистических параметров: генеральной средней и генеральной доли. Данный подход является фундаментальным в биометрии и других областях, где требуется оценка параметров генеральной совокупности на основе выборочных данных. Алгоритм включает в себя расчет стандартной ошибки среднего или доли, определение доверительного интервала с использованием t-критерия Стьюдента и интерпретацию полученных границ.

Используемые математические обозначения:

- $\overline{M}$ – Генеральная средняя (истинное среднее значение генеральной совокупности).
- $\widetilde{M}$ – Выборочная средняя (среднее значение, полученное из выборки).
- Δ – Доверительный интервал или предельная ошибка выборки (величина, определяющая ширину доверительного интервала).
- t – Коэффициент Стьюдента (t-критерий), зависящий от уровня значимости (или доверительной вероятности) и числа степеней свободы. Используется для построения доверительных интервалов для малых выборок.
- m – Стандартная ошибка среднего или доли (мера изменчивости выборочного среднего или доли).
- σ – Стандартное отклонение генеральной совокупности (мера рассеяния данных вокруг среднего значения в генеральной совокупности).
- n – Объем выборки (количество наблюдений в выборке).
- $\sqrt{n}$ – Квадратный корень из объема выборки.
- P – Генеральная доля (истинная доля признака в генеральной совокупности).
- p – Выборочная доля (доля признака, полученная из выборки).

- β – Доверительная вероятность (вероятность того, что истинное значение параметра генеральной совокупности попадет в рассчитанный доверительный интервал).
- ν – Число степеней свободы, обычно рассчитываемое как $n - 1$ для оценки среднего или доли.
- t_{st} – Табличное значение t-критерия Стьюдента для заданного уровня доверительной вероятности и числа степеней свободы. В данном случае, значение берется из Таблицы VII.

Эти обозначения позволяют формализовать процесс статистического вывода и оценить точность выборочных данных относительно истинных параметров генеральной совокупности.

3. Текстовое описание математических формул

3.1. Генеральная средняя

Для определения доверительных границ генеральной средней используются следующие формулы:

1. **Формула для доверительного интервала генеральной средней:** $\overline{M} = \widetilde{M} \pm \Delta$ Данная формула позволяет рассчитать доверительный интервал для генеральной средней ($\overline{M}$), используя выборочную среднюю ($\widetilde{M}$) и предельную ошибку выборки (Δ). Доверительный интервал представляет собой диапазон значений, в котором с определенной вероятностью находится истинное значение генеральной средней.
2. **Формула для предельной ошибки выборки:** $\Delta = t \cdot m$ Предельная ошибка выборки (Δ) вычисляется как произведение коэффициента Стьюдента (t) и стандартной ошибки среднего (m). Коэффициент Стьюдента учитывает уровень доверительной вероятности и число степеней свободы, а стандартная ошибка среднего отражает изменчивость выборочных данных.
3. **Формула для стандартной ошибки среднего:** $m = \frac{\sigma}{\sqrt{n}}$ Стандартная ошибка среднего (m) рассчитывается как отношение стандартного отклонения генеральной совокупности (σ) к квадратному корню из объема выборки ($\sqrt{n}$). Эта формула показывает, насколько выборочное среднее

может отличаться от истинного среднего генеральной совокупности.

3.2. Генеральная доля

Для определения доверительных границ генеральной доли используются следующие формулы:

1. **Формула для доверительного интервала генеральной доли:** $P = p \pm \Delta$ Эта формула позволяет определить доверительный интервал для генеральной доли (P), используя выборочную долю (p) и предельную ошибку выборки (Δ). Доверительный интервал указывает диапазон, в котором с заданной вероятностью находится истинное значение генеральной доли.
2. **Формула для предельной ошибки выборки:** $\Delta = t \cdot m$ Предельная ошибка выборки (Δ) для доли рассчитывается аналогично предельной ошибке для среднего — как произведение коэффициента Стьюдента (t) и стандартной ошибки доли (m).
3. **Формула для стандартной ошибки доли:** $m = \sqrt{\frac{p(1-p)}{n-1}}$ Стандартная ошибка доли (m) вычисляется как квадратный корень из произведения выборочной доли (p) и ее дополнения ($1 - p$), деленного на число степеней свободы ($n - 1$). Эта формула отражает изменчивость выборочной доли и ее близость к истинной генеральной доле.

4. Алгоритм расчетов в текстовом виде по шагам

4.1. Алгоритм расчета доверительных границ генеральной средней

1. **Определение исходных данных:**
   - Задать объем выборки (n).
   - Задать выборочную среднюю ($\widetilde{M}$).
   - Задать стандартное отклонение генеральной совокупности (σ).
   - Задать доверительную вероятность (β).
2. **Расчет числа степеней свободы:**
   - $\nu = n - 1$

3. **Определение табличного значения t-критерия Стьюдента (t_{st}):**
   - Найти значение t_{st} для заданных β и ν по Таблице VII (таблица критических значений t-распределения).
4. **Расчет стандартной ошибки среднего (m):**
   - Использовать формулу: $m = \frac{\sigma}{\sqrt{n}}$
5. **Расчет предельной ошибки выборки (Δ):**
   - Использовать формулу: $\Delta = t_{st} \cdot m$
6. **Расчет доверительных границ генеральной средней:**
   - Верхняя граница: $\overline{M}_{max} = \widetilde{M} + \Delta$
   - Нижняя граница: $\overline{M}_{min} = \widetilde{M} - \Delta$
7. **Интерпретация результатов:**
   - $\overline{M}_{max}$ представляет собой возможный максимум значения генеральной средней.
   - $\overline{M}_{min}$ представляет собой гарантированный минимум значения генеральной средней.

•

4.2. Алгоритм расчета доверительных границ генеральной доли

1. **Определение исходных данных:**
   - Задать объем выборки (n).
   - Задать выборочную долю (p).
   - Задать доверительную вероятность (β).
2. **Расчет числа степеней свободы:**
   - $\nu = n - 1$
3. **Определение табличного значения t-критерия Стьюдента (t_{st}):**
   - Найти значение t_{st} для заданных β и ν по Таблице VII (таблица критических значений t-распределения).
4. **Расчет стандартной ошибки доли (m):**
   - Использовать формулу: $m = \sqrt{\frac{p(1-p)}{n-1}}$
5. **Расчет предельной ошибки выборки (Δ):**
   - Использовать формулу: $\Delta = t_{st} \cdot m$
6. **Расчет доверительных границ генеральной доли:**
   - Верхняя граница: $P_{max} = p + \Delta$

– Нижняя граница: $P_{min} = p - \Delta$

7. **Интерпретация результатов:**
   – P_{max} представляет собой возможный максимум генеральной доли.
   – P_{min} представляет собой гарантированный минимум генеральной доли.

Объединенный алгоритм определения доверительных границ генеральных параметров

Алгоритм расчетов в текстовом виде по шагам

Блок 1. Начало

Начало выполнения алгоритма.

Блок 2. Ввод исходных данных

- Ввести объем выборки (n)
- Ввести доверительную вероятность (β)
- Выбрать тип параметра: генеральная средняя или генеральная доля

Блок 3. Проверка типа параметра

Проверить: выбран ли тип параметра "генеральная средняя"?

- Если ДА → переход к блоку 4
- Если НЕТ → переход к блоку 5

Блок 4. Ввод дополнительных данных для генеральной средней

- Ввести выборочную среднюю ($\tilde{M}$)
- Ввести стандартное отклонение генеральной совокупности (σ)

Блок 5. Ввод дополнительных данных для генеральной доли

- Ввести выборочную долю (p)

Блок 6. Расчет числа степеней свободы для генеральной средней

Вычислить: ν = n - 1

Блок 7. Расчет числа степеней свободы для генеральной доли

Вычислить: ν = n - 1

Блок 8. Определение табличного значения t-критерия для генеральной средней

Найти значение t_st для заданных β и ν по Таблице VII (таблица критических значений t-распределения)

Блок 9. Определение табличного значения t-критерия для генеральной доли

Найти значение t_st для заданных β и ν по Таблице VII (таблица критических значений t-распределения)

Блок 10. Расчет стандартной ошибки для генеральной средней

Вычислить стандартную ошибку среднего: $m = \sigma/\sqrt{n}$

Блок 11. Расчет стандартной ошибки для генеральной доли

Вычислить стандартную ошибку доли: $m = \sqrt{[p(1-p)/(n-1)]}$

Блок 12. Расчет предельной ошибки выборки для генеральной средней

Вычислить предельную ошибку: $\Delta = t_{st} \cdot m$

Блок 13. Расчет предельной ошибки выборки для генеральной доли

Вычислить предельную ошибку: $\Delta = t_{st} \cdot m$

Блок 14. Расчет доверительных границ для генеральной средней

- Верхняя граница: $\bar{M}_{max} = \tilde{M} + \Delta$
- Нижняя граница: $\bar{M}_{min} = \tilde{M} - \Delta$
- Получен доверительный интервал для генеральной средней

Блок 15. Расчет доверительных границ для генеральной доли

- Верхняя граница: $P_{max} = p + \Delta$
- Нижняя граница: $P_{min} = p - \Delta$
- Получен доверительный интервал для генеральной доли

Блок 16. Вывод результатов

Вывод доверительных границ генерального параметра (средней или доли)

Блок 17. Конец

Конец выполнения алгоритма.

Интерпретация результатов

Для генеральной средней:

- $\bar{M}_{max}$ представляет собой возможный максимум значения генеральной средней
- $\bar{M}_{min}$ представляет собой гарантированный минимум значения генеральной средней

Для генеральной доли:

- P_{max} представляет собой возможный максимум генеральной доли

- P_min представляет собой гарантированный минимум генеральной доли

Доверительный интервал с вероятностью β содержит истинное значение генерального параметра.

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.

5.1. Расчеты для Генеральной средней

Исходные данные из изображения: * Объем выборки (n) = 100 * Выборочная средняя ($\widetilde{M}$) = 200 * Стандартное отклонение генеральной совокупности (σ) = 20 * Доверительная вероятность (β) = 0.95 * Число степеней свободы ($\nu = n - 1$) = 99 * Табличное значение t-критерия Стьюдента (t_{st}) = 2.0 (для $\beta = 0.95$ и $\nu = 99$, из Табл. VII)

Численные расчеты:

1. **Расчет стандартной ошибки среднего (m):** $m = \frac{\sigma}{\sqrt{n}} = \frac{20}{\sqrt{100}} = \frac{20}{10} = 2.0$
2. **Расчет предельной ошибки выборки (Δ):** $\Delta = t_{st} \cdot m = 2.0 \cdot 2.0 = 4.0$
3. **Расчет доверительных границ генеральной средней:**
   - Верхняя граница (возможный максимум): $\overline{M}_{max} = \widetilde{M} + \Delta = 200 + 4.0 = 204.0$
   - Нижняя граница (гарантированный минимум): $\overline{M}_{min} = \widetilde{M} - \Delta = 200 - 4.0 = 196.0$

Результаты: * Возможный максимум значения генеральной средней: 204.0 * Гарантированный минимум значения генеральной средней: 196.0

5.2. Расчеты для Генеральной доли

Исходные данные из изображения: * Объем выборки (n) = 100 * Выборочная доля (p) = 0.60 * Доверительная вероятность (β) = 0.95 * Число степеней свободы ($\nu = n - 1$) = 99 * Табличное значение t-критерия Стьюдента (t_{st}) = 2.0 (для $\beta = 0.95$ и $\nu = 99$, из Табл. VII)

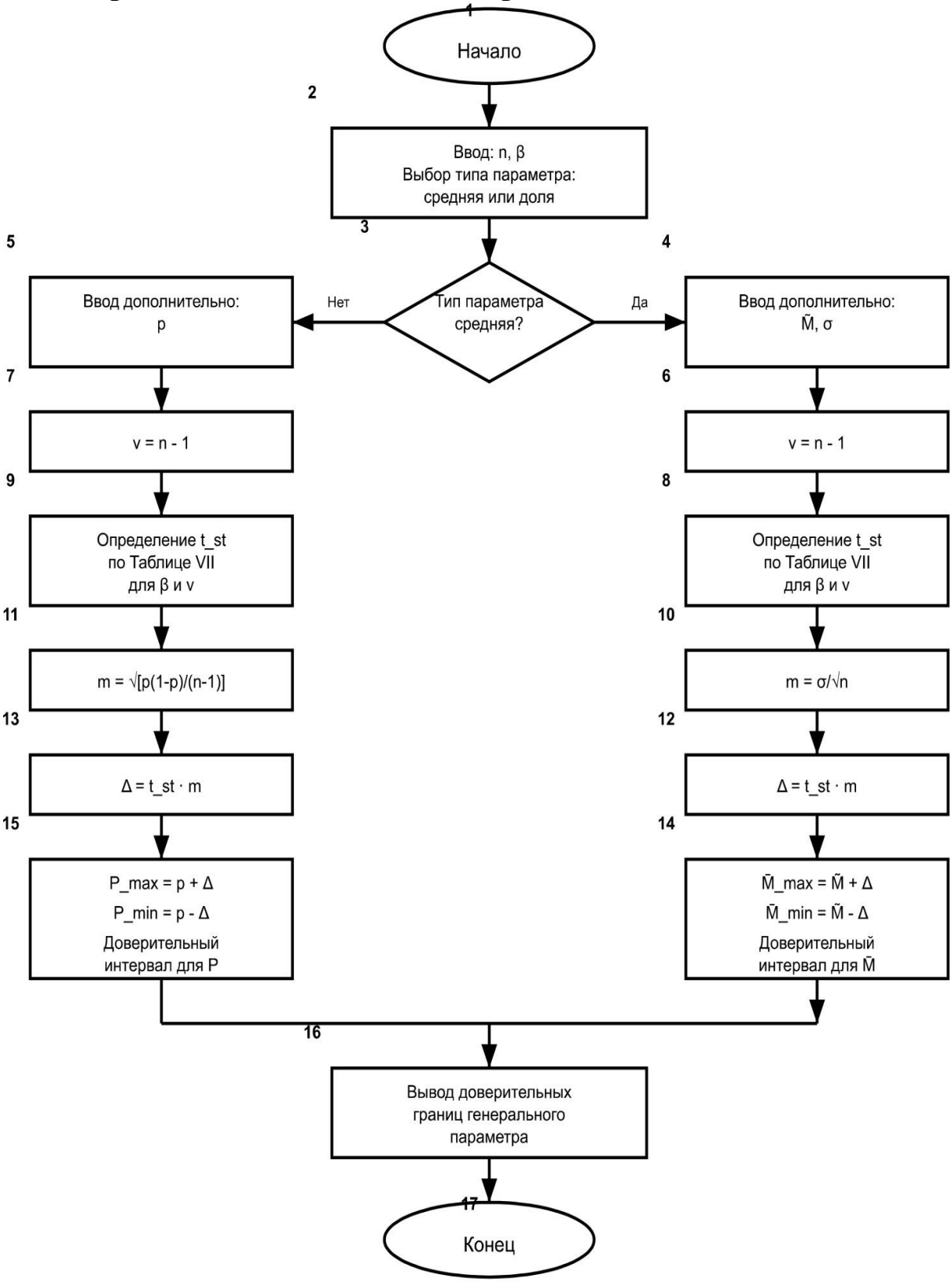
Численные расчеты:

1. **Расчет стандартной ошибки доли (m):** $m = \sqrt{\frac{p(1-p)}{n-1}} = \sqrt{\frac{0.60(1-0.60)}{100-1}} = \sqrt{\frac{0.60 \cdot 0.40}{99}} = \sqrt{\frac{0.24}{99}} \approx \sqrt{0.00242424} \approx 0.049236$
2. **Расчет предельной ошибки выборки (Δ):** $\Delta = t_{st} \cdot m = 2.0 \cdot 0.049236 \approx 0.098472$
3. **Расчет доверительных границ генеральной доли:**
   – Верхняя граница (возможный максимум): $P_{max} = p + \Delta = 0.60 + 0.098472 \approx 0.698472$
   – Нижняя граница (гарантированный минимум): $P_{min} = p - \Delta = 0.60 - 0.098472 \approx 0.501528$

Результаты:
* Возможный максимум генеральной доли: 0.698472
* Гарантированный минимум генеральной доли: 0.501528

<u>Примечание:</u> В исходном изображении значение m для генеральной доли округлено до 0.05, что приводит к округлению до 0.10. В данном расчете использованы более точные значения для демонстрации корректности метода.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class ConfidenceIntervalsGUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 12: Определение доверительных границ генеральных параметров")
        self.root.geometry("1600x700")  # Увеличил ширину для правой панели
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
```

```
        main_frame.columnconfigure(0, weight=2)  # Левая панель
        main_frame.columnconfigure(1, weight=1)  # Правая панель
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(1, weight=1)

        # Кнопки для выбора типа параметра
        self.parameter_type = "mean"

        radio_frame = ttk.Frame(self.input_frame)
        radio_frame.grid(row=0, column=0, sticky="ew", pady=(0, 2))

        # Кнопки для выбора типа параметра
        self.button_mean = tk.Button(
            radio_frame,
            text="Генеральная средняя",
            command=lambda: self.set_parameter_type("mean"),
            font=("Arial", 10),
            relief=tk.RAISED,
            bg="#E0E0E0",
            bd=1
        )
        self.button_proportion = tk.Button(
            radio_frame,
            text="Генеральная доля",
            command=lambda: self.set_parameter_type("proportion"),
            font=("Arial", 10),
            relief=tk.FLAT,
            bg="#F0F0F0",
            bd=1
        )

        self.button_mean.pack(side=tk.LEFT, padx=(0, 20))
        self.button_proportion.pack(side=tk.LEFT)

        # Изначально выбрана первая кнопка
        self.update_button_states()
```

```
        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=6,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=1, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=1, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=2, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 2))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=2)

        # Кнопка "Помощь" светло-голубого цвета
```

```
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
                                     relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическое представление:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 6), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

    def set_parameter_type(self, parameter_type):
        """Установка типа параметра через кнопки"""
        self.parameter_type = parameter_type
        print(f"Тип параметра установлен: {parameter_type}")
        self.update_button_states()
        self.fill_sample_data()

    def update_button_states(self):
        """Обновление визуального состояния кнопок"""
        if self.parameter_type == "mean":
            self.button_mean.config(relief=tk.RAISED, bg="#D0D0D0")
            self.button_proportion.config(relief=tk.FLAT, bg="#F0F0F0")
        else:
            self.button_mean.config(relief=tk.FLAT, bg="#F0F0F0")
            self.button_proportion.config(relief=tk.RAISED, bg="#D0D0D0")

    def fill_sample_data(self):
        """Заполнение исходными данными"""
        self.input_text.delete(1.0, tk.END)
```

```
        if self.parameter_type == "mean":
            sample_data = """Объем выборки (n): 100
Выборочная средняя (M̃): 200
Стандартное отклонение генеральной совокупности (σ): 20
Доверительная вероятность (β): 0.95"""
        else:
            sample_data = """Объем выборки (n): 100
Выборочная доля (p): 0.60
Доверительная вероятность (β): 0.95"""

        self.input_text.insert(1.0, sample_data)
        print(f"Заполнены данные для типа: {self.parameter_type}")

    def get_t_critical(self, alpha, df):
        """Получение критического значения t-распределения"""
        try:
            # Используем scipy.stats для получения критического значения
            return stats.t.ppf(1 - alpha / 2, df)
        except:
            # Если scipy недоступна, используем приближенные значения
            # Для больших выборок (df >= 30) используем нормальное
распределение
            if df >= 30:
                if alpha == 0.05:  # 95% доверительный интервал
                    return 1.96
                elif alpha == 0.01:  # 99% доверительный интервал
                    return 2.58
                elif alpha == 0.1:  # 90% доверительный интервал
                    return 1.64

            # Для малых выборок используем таблицу t-критериев
(приближенно)
            t_table = {
                1: {0.1: 6.314, 0.05: 12.706, 0.01: 63.657},
                2: {0.1: 2.920, 0.05: 4.303, 0.01: 9.925},
                5: {0.1: 2.015, 0.05: 2.571, 0.01: 4.032},
                10: {0.1: 1.812, 0.05: 2.228, 0.01: 3.169},
                20: {0.1: 1.725, 0.05: 2.086, 0.01: 2.845},
                30: {0.1: 1.697, 0.05: 2.042, 0.01: 2.750},
                50: {0.1: 1.676, 0.05: 2.009, 0.01: 2.678},
                100: {0.1: 1.660, 0.05: 1.984, 0.01: 2.626}
            }

            # Найти ближайшее значение df в таблице
            closest_df = min(t_table.keys(), key=lambda x: abs(x - df))
            return t_table[closest_df].get(alpha, 2.0)

    def parse_input_data(self):
        """Парсинг входных данных"""
        data_str = self.input_text.get(1.0, tk.END).strip()
        lines = [line.strip() for line in data_str.split('\n') if
line.strip()]

        data = {}
        for line in lines:
            if ':' in line:
                key, value = line.split(':', 1)
                key = key.strip()
                value = value.strip()
```

```
                try:
                    data[key] = float(value)
                except ValueError:
                    raise ValueError(f"Некорректное значение для '{key}':
{value}")

        return data

    def plot_confidence_interval(self, sample_value, lower_bound,
upper_bound, parameter_type, beta):
        """Построение графика доверительного интервала"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание subplot
        ax = self.fig.add_subplot(111)

        # Параметры для построения
        center = sample_value
        margin = upper_bound - center

        # Создание визуализации доверительного интервала
        # Основная линия интервала
        ax.plot([lower_bound, upper_bound], [0, 0], 'b-', linewidth=4,
alpha=0.7,
                label=f'{beta * 100}% доверительный интервал')

        # Точка выборочного значения
        ax.plot(center, 0, 'ro', markersize=12, label=f'Выборочное значение
= {center}')

        # Границы интервала
        ax.plot([lower_bound, lower_bound], [-0.1, 0.1], 'g-', linewidth=3)
        ax.plot([upper_bound, upper_bound], [-0.1, 0.1], 'g-', linewidth=3)

        # Заливка области доверительного интервала
        ax.fill_between([lower_bound, upper_bound], [-0.05, -0.05], [0.05,
0.05],
                        alpha=0.3, color='lightblue', label='Область
доверительного интервала')

        # Подписи значений
        ax.text(lower_bound, -0.2, f'{lower_bound:.4f}', ha='center',
va='top', fontsize=10, fontweight='bold',
                color='green')
        ax.text(upper_bound, -0.2, f'{upper_bound:.4f}', ha='center',
va='top', fontsize=10, fontweight='bold',
                color='green')
        ax.text(center, 0.2, f'{center}', ha='center', va='bottom',
fontsize=10, fontweight='bold', color='red')

        # Стрелки и подписи полей
        arrow_props = dict(arrowstyle='<->', color='orange', lw=2)
        ax.annotate('', xy=(lower_bound, 0.15), xytext=(center, 0.15),
arrowprops=arrow_props)
        ax.annotate('', xy=(center, 0.15), xytext=(upper_bound, 0.15),
arrowprops=arrow_props)
        ax.text((lower_bound + center) / 2, 0.18, f'±{margin:.4f}',
ha='center', va='bottom', fontsize=9,
                color='orange')
        ax.text((center + upper_bound) / 2, 0.18, f'±{margin:.4f}',
```

```
ha='center', va='bottom', fontsize=9,
                color='orange')

        # Настройка осей
        range_margin = margin * 0.5
        ax.set_xlim(lower_bound - range_margin, upper_bound + range_margin)
        ax.set_ylim(-0.3, 0.3)

        # Подписи осей
        if parameter_type == "mean":
            ax.set_xlabel('Значение генеральной средней', fontsize=12,
fontweight='bold')
            ax.set_title(f'Доверительный интервал для генеральной
средней\n(β = {beta * 100}%)',
                         fontsize=14, fontweight='bold')
        else:
            ax.set_xlabel('Значение генеральной доли', fontsize=12,
fontweight='bold')
            ax.set_title(f'Доверительный интервал для генеральной доли\n(β
= {beta * 100}%)',
                         fontsize=14, fontweight='bold')

        # Удаление лишних меток на оси Y
        ax.set_yticks([])

        # Сетка
        ax.grid(True, alpha=0.3, axis='x')

        # Легенда
        ax.legend(loc='upper right', fontsize=9, frameon=True,
fancybox=True, shadow=True)

        # Добавление информационного текста
        info_text = f'Точность: ±{margin:.4f}\nШирина интервала:
{upper_bound - lower_bound:.4f}'
        ax.text(0.02, 0.98, info_text, transform=ax.transAxes, fontsize=10,
                verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='lightyellow', alpha=0.8))

        # Настройка layout
        self.fig.tight_layout()

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму"""
        try:
            data = self.parse_input_data()

            if self.parameter_type == "mean":
                result, sample_value, lower_bound, upper_bound, beta =
self.calculate_mean_confidence_interval(data)
            else:
                result, sample_value, lower_bound, upper_bound, beta =
self.calculate_proportion_confidence_interval(
                    data)

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
```

```
            self.result_text.config(state=tk.DISABLED)

            # Построение графика
            self.plot_confidence_interval(sample_value, lower_bound,
upper_bound, self.parameter_type, beta)

        except ValueError as e:
            messagebox.showerror("Ошибка", f"Ошибка в данных: {str(e)}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def calculate_mean_confidence_interval(self, data):
        """Расчет доверительного интервала для генеральной средней"""
        try:
            n = int(data.get('Объем выборки (n)', 0))
            sample_mean = data.get('Выборочная средняя (M̃)', 0)
            sigma = data.get('Стандартное отклонение генеральной
совокупности (σ)', 0)
            beta = data.get('Доверительная вероятность (β)', 0.95)

            if n <= 0:
                raise ValueError("Объем выборки должен быть больше 0")
            if sigma <= 0:
                raise ValueError("Стандартное отклонение должно быть больше
0")
            if not (0 < beta < 1):
                raise ValueError("Доверительная вероятность должна быть
между 0 и 1")

            # Расчет числа степеней свободы
            df = n - 1

            # Уровень значимости
            alpha = 1 - beta

            # Критическое значение t-критерия
            t_critical = self.get_t_critical(alpha, df)

            # Стандартная ошибка среднего
            standard_error = sigma / math.sqrt(n)

            # Предельная ошибка выборки
            margin_error = t_critical * standard_error

            # Доверительные границы
            lower_bound = sample_mean - margin_error
            upper_bound = sample_mean + margin_error

            result = f"""РАСЧЕТ ДОВЕРИТЕЛЬНЫХ ГРАНИЦ ГЕНЕРАЛЬНОЙ СРЕДНЕЙ

Исходные данные:
Объем выборки (n): {n}
Выборочная средняя (M̃): {sample_mean}
Стандартное отклонение генеральной совокупности (σ): {sigma}
Доверительная вероятность (β): {beta}

Пошаговый расчет:

1. Число степеней свободы:
   ν = n - 1 = {n} - 1 = {df}
```

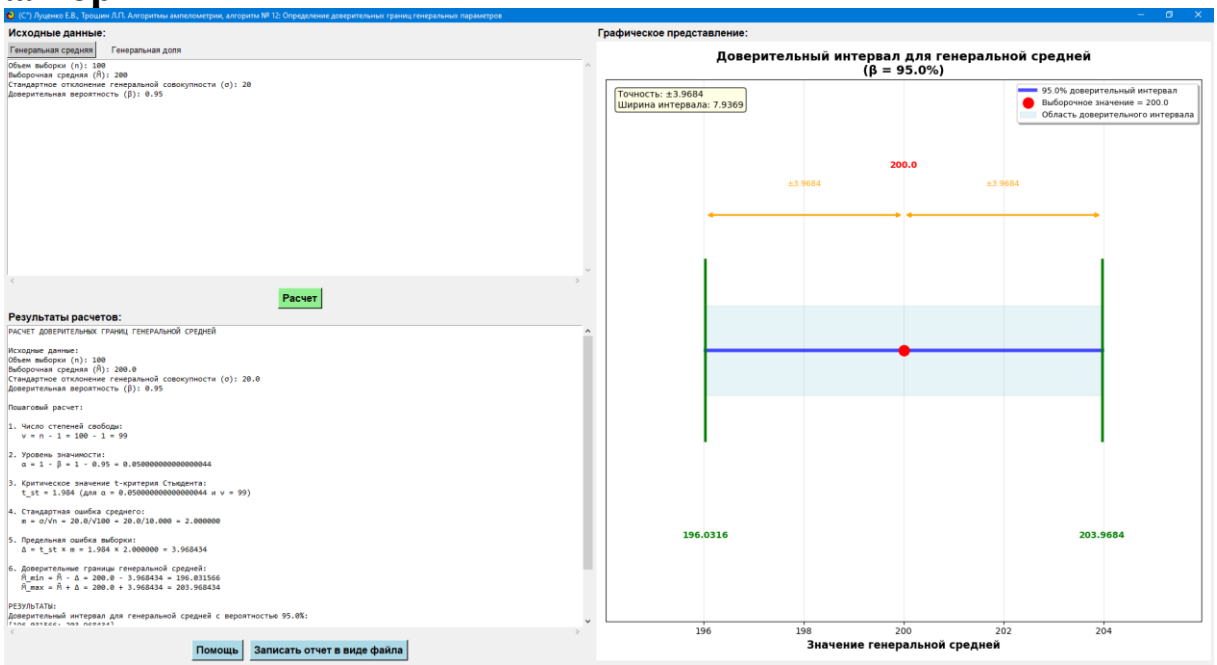
```
2. Уровень значимости:
   α = 1 - β = 1 - {beta} = {alpha}

3. Критическое значение t-критерия Стьюдента:
   t_st = {t_critical:.3f} (для α = {alpha} и ν = {df})

4. Стандартная ошибка среднего:
   m = σ/√n = {sigma}/√{n} = {sigma}/{math.sqrt(n):.3f} =
{standard_error:.6f}

5. Предельная ошибка выборки:
   Δ = t_st × m = {t_critical:.3f} × {standard_error:.6f} =
{margin_error:.6f}

6. Доверительные границы генеральной средней:
   M̃_min = M̃ - Δ = {sample_mean} - {margin_error:.6f} = {lower_bound:.6f}
   M̃_max = M̃ + Δ = {sample_mean} + {margin_error:.6f} = {upper_bound:.6f}

РЕЗУЛЬТАТЫ:
Доверительный интервал для генеральной средней с вероятностью {beta *
100}%:
[{lower_bound:.6f}; {upper_bound:.6f}]

Интерпретация:
- Гарантированный минимум генеральной средней: {lower_bound:.6f}
- Возможный максимум генеральной средней: {upper_bound:.6f}
- С вероятностью {beta * 100}% истинное значение генеральной средней
  находится в интервале от {lower_bound:.6f} до {upper_bound:.6f}"""

            return result, sample_mean, lower_bound, upper_bound, beta

        except KeyError as e:
            raise ValueError(f"Отсутствует обязательное поле: {e}")

    def calculate_proportion_confidence_interval(self, data):
        """Расчет доверительного интервала для генеральной доли"""
        try:
            n = int(data.get('Объем выборки (n)', 0))
            sample_prop = data.get('Выборочная доля (p)', 0)
            beta = data.get('Доверительная вероятность (β)', 0.95)

            if n <= 0:
                raise ValueError("Объем выборки должен быть больше 0")
            if not (0 < sample_prop < 1):
                raise ValueError("Выборочная доля должна быть между 0 и 1")
            if not (0 < beta < 1):
                raise ValueError("Доверительная вероятность должна быть
между 0 и 1")

            # Расчет числа степеней свободы
            df = n - 1

            # Уровень значимости
            alpha = 1 - beta

            # Критическое значение t-критерия
            t_critical = self.get_t_critical(alpha, df)

            # Стандартная ошибка доли
            standard_error = math.sqrt((sample_prop * (1 - sample_prop)) /
```

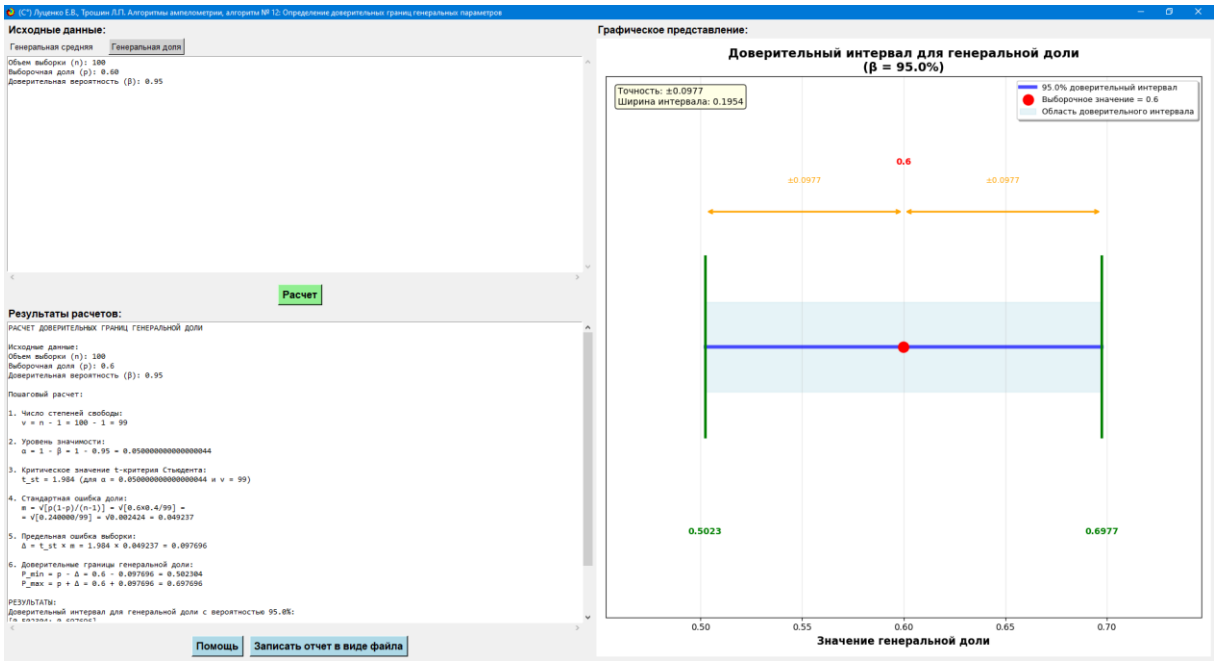
```
(n - 1))

            # Предельная ошибка выборки
            margin_error = t_critical * standard_error

            # Доверительные границы
            lower_bound = sample_prop - margin_error
            upper_bound = sample_prop + margin_error

            # Ограничение границ в пределах [0, 1]
            lower_bound = max(0, lower_bound)
            upper_bound = min(1, upper_bound)

            result = f"""РАСЧЕТ ДОВЕРИТЕЛЬНЫХ ГРАНИЦ ГЕНЕРАЛЬНОЙ ДОЛИ

Исходные данные:
Объем выборки (n): {n}
Выборочная доля (p): {sample_prop}
Доверительная вероятность (β): {beta}

Пошаговый расчет:

1. Число степеней свободы:
   ν = n - 1 = {n} - 1 = {df}

2. Уровень значимости:
   α = 1 - β = 1 - {beta} = {alpha}

3. Критическое значение t-критерия Стьюдента:
   t_st = {t_critical:.3f} (для α = {alpha} и ν = {df})

4. Стандартная ошибка доли:
   m = √[p(1-p)/(n-1)] = √[{sample_prop}×{1 - sample_prop}/{df}] =
   = √[{sample_prop * (1 - sample_prop):.6f}/{df}] = √{(sample_prop * (1 -
sample_prop)) / df:.6f} = {standard_error:.6f}

5. Предельная ошибка выборки:
   Δ = t_st × m = {t_critical:.3f} × {standard_error:.6f} =
{margin_error:.6f}

6. Доверительные границы генеральной доли:
   P_min = p - Δ = {sample_prop} - {margin_error:.6f} = {lower_bound:.6f}
   P_max = p + Δ = {sample_prop} + {margin_error:.6f} = {upper_bound:.6f}

РЕЗУЛЬТАТЫ:
Доверительный интервал для генеральной доли с вероятностью {beta * 100}%:
[{lower_bound:.6f}; {upper_bound:.6f}]

Интерпретация:
- Гарантированный минимум генеральной доли: {lower_bound:.6f}
- Возможный максимум генеральной доли: {upper_bound:.6f}
- С вероятностью {beta * 100}% истинное значение генеральной доли
  находится в интервале от {lower_bound:.6f} до {upper_bound:.6f}"""

            return result, sample_prop, lower_bound, upper_bound, beta

        except KeyError as e:
            raise ValueError(f"Отсутствует обязательное поле: {e}")

    def show_help(self):
        """Открытие файла справки"""
```

```
        help_file_name = "help-12.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            parameter_type_name = "генеральной средней" if
self.parameter_type == "mean" else "генеральной доли"

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 12
Определение доверительных границ {parameter_type_name}

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: График доверительного интервала для {parameter_type_name}
отображается в графической области программы.

================================================================================
Конец отчета"""
```

```
            filename =
f"Алгоритм_12_Доверительные_границы_{parameter_type_name.replace(' ',
'_')}_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = ConfidenceIntervalsGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм





9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 12

Определение доверительных границ генеральной средней

Дата и время расчета: 2025-07-24 05:23:15
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Объем выборки (n): 100
Выборочная средняя (M̃): 200
Стандартное отклонение генеральной совокупности (σ): 20
Доверительная вероятность (β): 0.95

==========

РАСЧЕТ ДОВЕРИТЕЛЬНЫХ ГРАНИЦ ГЕНЕРАЛЬНОЙ СРЕДНЕЙ

Исходные данные:
Объем выборки (n): 100
Выборочная средняя (M̃): 200.0
Стандартное отклонение генеральной совокупности (σ): 20.0

Доверительная вероятность (β): 0.95

Пошаговый расчет:

1. Число степеней свободы:
   ν = n - 1 = 100 - 1 = 99

2. Уровень значимости:
   α = 1 - β = 1 - 0.95 = 0.050000000000000044

3. Критическое значение t-критерия Стьюдента:
   t_st = 1.984 (для α = 0.050000000000000044 и ν = 99)

4. Стандартная ошибка среднего:
   m = σ/√n = 20.0/√100 = 20.0/10.000 = 2.000000
5. Предельная ошибка выборки:
   Δ = t_st × m = 1.984 × 2.000000 = 3.968434

6. Доверительные границы генеральной средней:
   M̄_min = M̃ - Δ = 200.0 - 3.968434 = 196.031566
   M̄_max = M̃ + Δ = 200.0 + 3.968434 = 203.968434

РЕЗУЛЬТАТЫ:
Доверительный интервал для генеральной средней с вероятностью 95.0%:
[196.031566; 203.968434]

Интерпретация:
- Гарантированный минимум генеральной средней: 196.031566
- Возможный максимум генеральной средней: 203.968434
- С вероятностью 95.0% истинное значение генеральной средней находится в интервале от 196.031566 до 203.968434
==========
Конец отчета

ОТЧЕТ ПО АЛГОРИТМУ № 12
Определение доверительных границ генеральной доли

Дата и время расчета: 2025-07-24 05:20:40
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
==========
ИСХОДНЫЕ ДАННЫЕ:
Объем выборки (n): 100
Выборочная доля (p): 0.60
Доверительная вероятность (β): 0.95
==========
РАСЧЕТ ДОВЕРИТЕЛЬНЫХ ГРАНИЦ ГЕНЕРАЛЬНОЙ ДОЛИ
Исходные данные:
Объем выборки (n): 100
Выборочная доля (p): 0.6
Доверительная вероятность (β): 0.95

Пошаговый расчет:

1. Число степеней свободы:
$\nu = n - 1 = 100 - 1 = 99$

2. Уровень значимости:
$\alpha = 1 - \beta = 1 - 0.95 = 0.0500000000000000044$

3. Критическое значение t-критерия Стьюдента:
$t_{st} = 1.984$ (для $\alpha = 0.0500000000000000044$ и $\nu = 99$)

4. Стандартная ошибка доли:
$m = \sqrt{[p(1-p)/(n-1)]} = \sqrt{[0.6 \times 0.4/99]} =$
$= \sqrt{[0.240000/99]} = \sqrt{0.002424} = 0.049237$

5. Предельная ошибка выборки:
$\Delta = t_{st} \times m = 1.984 \times 0.049237 = 0.097696$

6. Доверительные границы генеральной доли:
$P_{min} = p - \Delta = 0.6 - 0.097696 = 0.502304$

P_max = p + Δ = 0.6 + 0.097696 = 0.697696

РЕЗУЛЬТАТЫ:

Доверительный интервал для генеральной доли с вероятностью 95.0%: [0.502304; 0.697696]

Интерпретация:
- Гарантированный минимум генеральной доли: 0.502304
- Возможный максимум генеральной доли: 0.697696
- С вероятностью 95.0% истинное значение генеральной доли находится в интервале от 0.502304 до 0.697696

==========

Конец отчета

10. Инструкция пользователю по программе: "Алгоритм №12: Определение доверительных границ генеральных параметров"

Общие сведения

Программа предназначена для автоматизации расчетов доверительных интервалов для генеральных параметров (средней и доли) согласно алгоритму из работы Плохинского Н.А. "Алгоритмы биометрии".

Разработчики: (С°) Луценко Е.В., Трошин Л.П.

Название: Алгоритмы ампелометрии Алгоритм: № 12

Системные требования

- Операционная система: Windows 7/8/10/11
- Оперативная память: минимум 512 МБ
- Свободное место на диске: 50 МБ
- Дополнительное программное обеспечение не требуется (программа работает автономно)

Установка и запуск

1. Скопируйте файлы программы в любую папку на компьютере:
    - Algorithm_12.exe - исполняемый файл программы
    - _Aidos.ico - файл иконки программы
    - help-12.pdf - файл справки с описанием алгоритма

2. Для запуска программы дважды щелкните по файлу Algorithm_12.exe

Описание интерфейса программы

Главное окно программы содержит следующие элементы:

1. Верхняя часть - выбор типа параметра

- **Кнопка "Генеральная средняя"** - для расчета доверительного интервала генеральной средней
- **Кнопка "Генеральная доля"** - для расчета доверительного интервала генеральной доли

2. Окно исходных данных

Поле для ввода исходных данных. При выборе типа параметра автоматически заполняется примерными данными из учебника.

Для генеральной средней требуются:

- Объем выборки (n)
- Выборочная средняя (M̃)
- Стандартное отклонение генеральной совокупности (σ)
- Доверительная вероятность (β)

Для генеральной доли требуются:

- Объем выборки (n)
- Выборочная доля (p)
- Доверительная вероятность (β)

3. Кнопка "Расчет"

Светло-зеленая кнопка для выполнения расчетов по выбранному алгоритму.

4. Окно результатов расчетов

Поле для отображения подробных результатов расчетов с пошаговым решением.

5. Нижние кнопки управления

- **Кнопка "Помощь"** (светло-голубая) - открывает файл справки help-12.pdf
- **Кнопка "Записать отчет в виде файла"** (светло-голубая) - сохраняет отчет в текстовый файл

Порядок работы с программой

Шаг 1: Выбор типа параметра

1. В верхней части окна выберите тип параметра, нажав соответствующую кнопку:

- "Генеральная средняя" - для расчета доверительного интервала среднего значения
- "Генеральная доля" - для расчета доверительного интервала доли признака

Шаг 2: Ввод исходных данных

1. В окне "Исходные данные" автоматически появятся примерные данные
2. Замените примерные данные на ваши реальные значения, сохраняя формат:
3. Название параметра: значение
4. Убедитесь, что все обязательные поля заполнены корректными числовыми значениями

Шаг 3: Выполнение расчетов

1. Нажмите кнопку "Расчет" (светло-зеленая)
2. В окне "Результаты расчетов" появится подробное решение с пошаговыми вычислениями

Шаг 4: Сохранение результатов (опционально)

1. Нажмите кнопку "Записать отчет в виде файла"
2. В папке с программой будет создан текстовый файл с отчетом
3. Имя файла формируется автоматически с указанием даты и времени
4.

Примеры данных

Пример 1: Генеральная средняя

Объем выборки (n): 100
Выборочная средняя (M̃): 200
Стандартное отклонение генеральной совокупности (σ): 20
Доверительная вероятность (β): 0.95

Результат: Программа вычислит доверительный интервал для генеральной средней с вероятностью 95%.

Пример 2: Генеральная доля

Объем выборки (n): 100
Выборочная доля (p): 0.60
Доверительная вероятность (β): 0.95

Результат: Программа вычислит доверительный интервал для генеральной доли с вероятностью 95%.

Интерпретация результатов

Для генеральной средней:

- **Доверительный интервал** - диапазон значений, в котором с заданной вероятностью находится истинное среднее значение генеральной совокупности
- **Гарантированный минимум** - нижняя граница доверительного интервала
- **Возможный максимум** - верхняя граница доверительного интервала

Для генеральной доли:

- **Доверительный интервал** - диапазон значений, в котором с заданной вероятностью находится истинная доля признака в генеральной совокупности
- **Гарантированный минимум** - нижняя граница доверительного интервала (не менее 0)
- **Возможный максимум** - верхняя граница доверительного интервала (не более 1)

Возможные ошибки и их устранение

Ошибка: "Отсутствует обязательное поле"

Причина: Не заполнены все необходимые параметры

Решение: Проверьте, что введены все требуемые значения в правильном формате

Ошибка: "Объем выборки должен быть больше 0"

Причина: Введено некорректное значение объема выборки

Решение: Введите положительное целое число для объема выборки

Ошибка: "Стандартное отклонение должно быть больше 0"

Причина: Введено некорректное значение стандартного отклонения

Решение: Введите положительное число для стандартного отклонения

Ошибка: "Доверительная вероятность должна быть между 0 и 1"
Причина: Введено некорректное значение доверительной вероятности
Решение: Введите значение от 0 до 1 (например, 0.95 для 95%)
Ошибка: "Выборочная доля должна быть между 0 и 1"
Причина: Введено некорректное значение выборочной доли
Решение: Введите значение от 0 до 1 (например, 0.6 для 60%)

Формат входных данных
Данные должны вводиться в формате:
Название параметра: значение
Важно:

- Используйте двоеточие (:) для разделения названия и значения
- Каждый параметр должен быть на отдельной строке
- Используйте точку (.) как десятичный разделитель
- Не используйте пробелы в числовых значениях

Формат выходного отчета
Сохраненный отчет содержит:

1. Заголовок с номером и названием алгоритма
2. Дату и время расчета
3. Исходные данные
4. Подробное пошаговое решение
5. Итоговые результаты с интерпретацией

Файл отчета сохраняется в кодировке UTF-8 с автоматически сгенерированным именем: Алгоритм_12_Доверительные_границы_[тип_параметра]_[дата_время].txt

Техническая поддержка
Если возникли проблемы с работой программы:

1. Убедитесь, что все файлы программы находятся в одной папке
2. Проверьте корректность ввода данных согласно примерам
3. Попробуйте перезапустить программу

4. Обратитесь к файлу справки help-12.pdf для изучения теоретических основ алгоритма

Дополнительная информация

О доверительных интервалах:

Доверительный интервал - это диапазон значений, который с определенной вероятностью содержит истинное значение параметра генеральной совокупности. Чем выше доверительная вероятность, тем шире интервал.

Типичные значения доверительной вероятности:

- 0.90 (90%) - для предварительных оценок
- 0.95 (95%) - стандартное значение в большинстве исследований
- 0.99 (99%) - для высокоточных исследований

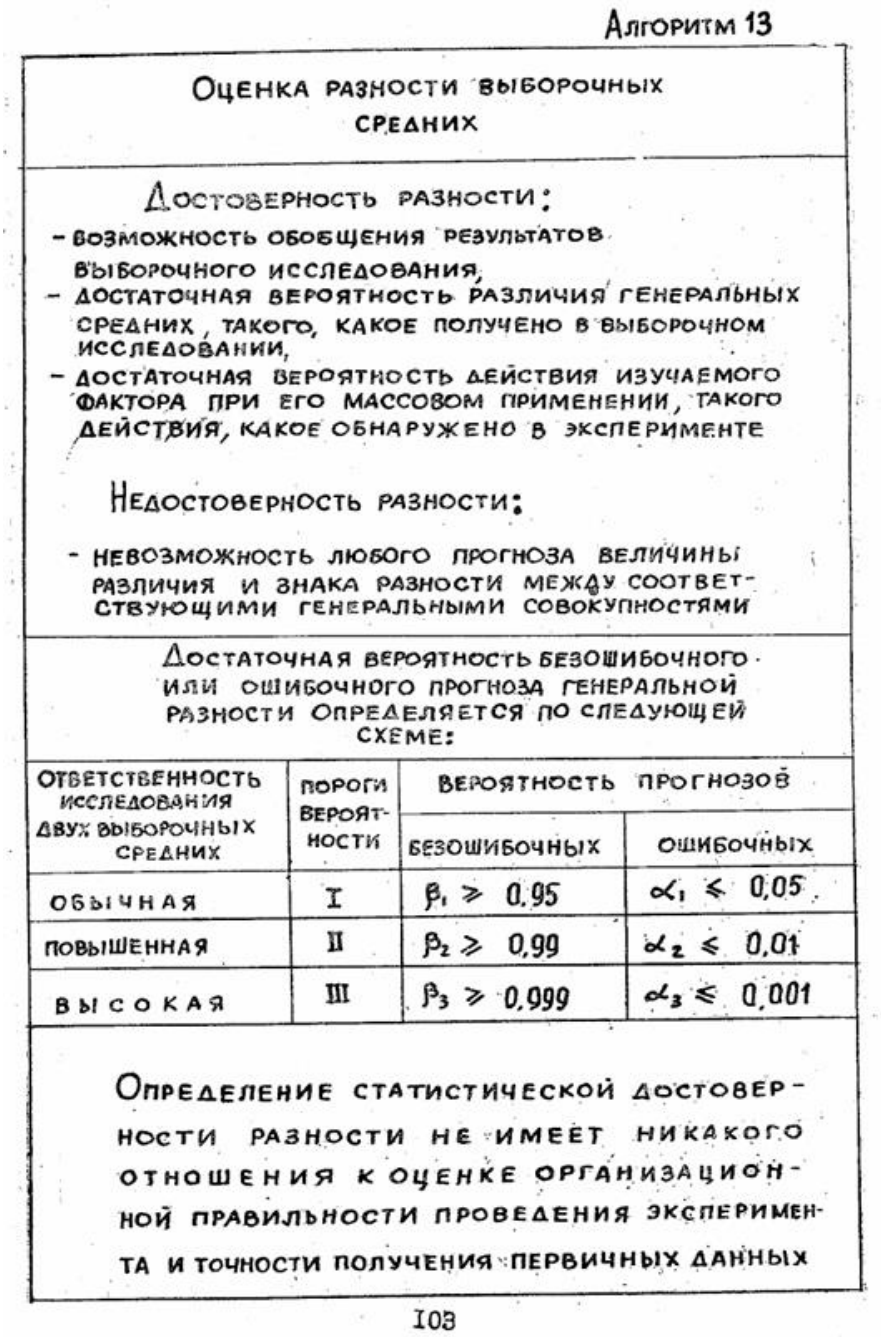
Область применения:

- Ампелометрия
- Биометрические исследования
- Контроль качества продукции
- Социологические опросы
- Медицинские исследования
- Экономический анализ

Версия инструкции: 1.0
Дата создания: 2025
Авторы: (С°) Луценко Е.В., Трошин Л.П.

Алгоритм-13: Оценка разности выборочных средних

1. Исходное изображение



Алгоритм 13

Оценка разности выборочных средних

Достоверность разности:

- возможность обобщения результатов выборочного исследования,
- достаточная вероятность различия генеральных средних, такого, какое получено в выборочном исследовании,
- достаточная вероятность действия изучаемого фактора при его массовом применении, такого действия, какое обнаружено в эксперименте

Недостоверность разности:

- невозможность любого прогноза величины различия и знака разности между соответствующими генеральными совокупностями

Достаточная вероятность безошибочного или ошибочного прогноза генеральной разности определяется по следующей схеме:

Ответственность исследования двух выборочных средних	Пороги вероятности	Вероятность прогнозов	
		безошибочных	ошибочных
обычная	I	$\beta_1 \geqslant 0{,}95$	$\alpha_1 \leqslant 0{,}05$
повышенная	II	$\beta_2 \geqslant 0{,}99$	$\alpha_2 \leqslant 0{,}01$
высокая	III	$\beta_3 \geqslant 0{,}999$	$\alpha_3 \leqslant 0{,}001$

Определение статистической достоверности разности не имеет никакого отношения к оценке организационной правильности проведения эксперимента и точности получения первичных данных

103

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск, ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму, в т.ч. используемые в формулах условные математические обозначения переменных.

Данный алгоритм описывает процесс оценки статистической достоверности разности между выборочными средними. Он определяет условия, при которых разность считается достоверной или недостоверной, а также устанавливает пороговые значения вероятностей для безошибочного и ошибочного прогноза.

Используемые обозначения:

- β_i — Вероятность безошибочного прогноза (уровень достоверности).
- α_i — Вероятность ошибочного прогноза (уровень значимости).
- i — Индекс, обозначающий уровень ответственности исследования (1 - обычная, 2 - повышенная, 3 - высокая).

3. Текстовое описание математических формул в стандартном для MS Word-2010 виде

Достоверность разности:

Разность между выборочными средними считается **достоверной**, если выполняются следующие условия:

- Существует возможность обобщения результатов выборочного исследования на генеральную совокупность.
- Имеется достаточная вероятность различия генеральных средних, аналогичная той, что получена в выборочном исследовании.
- Имеется достаточная вероятность действия изучаемого фактора при его массовом применении, соответствующая эффекту, обнаруженному в эксперименте.

Недостоверность разности:

Разность между выборочными средними считается **недостоверной**, если:

- Невозможно сделать какой-либо прогноз относительно величины и знака разности между соответствующими генеральными совокупностями.

Определение достаточной вероятности безошибочного или ошибочного прогноза генеральной разности:
Достаточная вероятность безошибочного (β_i) или ошибочного (α_i) прогноза генеральной разности определяется по следующей схеме в зависимости от ответственности исследования:

Ответственность исследования двух выборочных средних	Пороги вероятности	Вероятность прогнозов (безошибочных)	Вероятность прогнозов (ошибочных)
Обычная	I	$\beta_1 \geq 0.95$	$\alpha_1 \leq 0.05$
Повышенная	II	$\beta_2 \geq 0.99$	$\alpha_2 \leq 0.01$
Высокая	III	$\beta_3 \geq 0.999$	$\alpha_3 \leq 0.001$

Примечание: Определение статистической достоверности разности не имеет никакого отношения к оценке организационной правильности проведения эксперимента и точности получения первичных данных.

4. Алгоритм расчетов в текстовом виде по шагам.

1. **Определение ответственности исследования:** В зависимости от важности и критичности исследования, выберите один из трех уровней ответственности: “Обычная”, “Повышенная” или “Высокая”.
2. **Установление пороговых значений:** На основе выбранного уровня ответственности определите соответствующие пороговые значения для вероятности безошибочного прогноза (β_i) и вероятности ошибочного прогноза (α_i) из таблицы выше.
3. **Расчет вероятностей:** Вычислите фактическую вероятность безошибочного прогноза (β) и вероятность ошибочного прогноза (α) для разности выборочных средних, используя соответствующие статистические методы (например, t-критерий Стьюдента, z-критерий и т.д.). *Примечание: В данном документе не приводятся конкретные формулы для расчета β и α, так как они зависят от используемого статистического критерия и исходных данных.*

4. **Оценка достоверности:**
   - Если фактическая вероятность безошибочного прогноза (β) больше или равна пороговому значению β_i и фактическая вероятность ошибочного прогноза (α) меньше или равна пороговому значению α_i, то разность выборочных средних считается **достоверной**.
   - В противном случае, разность выборочных средних считается **недостоверной**.

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.

Исходные данные из изображения:

Ответственность исследования	Пороги вероятности	Вероятность безошибочных (β_i)	Вероятность ошибочных (α_i)
Обычная	I	0.95	0.05
Повышенная	II	0.99	0.01
Высокая	III	0.999	0.001

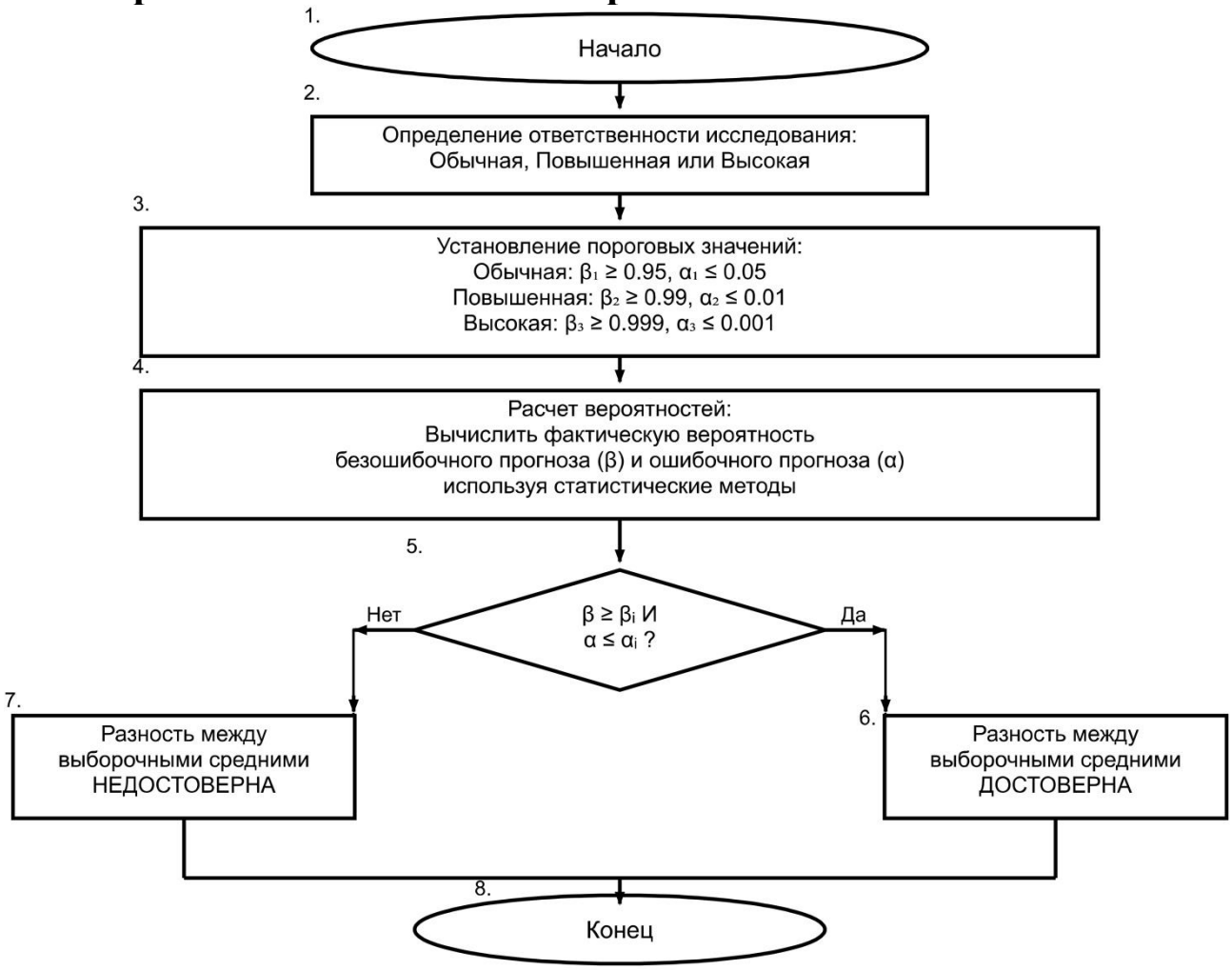
Численный расчет всех показателей:

- **Показатели β_i:**
  - Для обычной ответственности: $\beta_1 = 0.95$
  - Для повышенной ответственности: $\beta_2 = 0.99$
  - Для высокой ответственности: $\beta_3 = 0.999$
- **Показатели α_i:**
  - Для обычной ответственности: $\alpha_1 = 0.05$
  - Для повышенной ответственности: $\alpha_2 = 0.01$
  - Для высокой ответственности: $\alpha_3 = 0.001$

Примечание: В данном документе не приводятся примеры численных расчетов для фактических β и α, так как для этого требуются конкретные исходные данные выборочных средних и их дисперсий, которые отсутствуют в предоставленном изображении. Расчеты β и α должны быть выполнены с использованием соответствующих статистических формул,

например, для t-критерия Стьюдента или z-критерия, в зависимости от типа данных и объема выборки.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм.

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class BiometryAlgorithm13GUI:
    def __init__(self, root):
        self.root = root
```

```
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 13: Оценка разности выборочных средних")
        self.root.geometry("1600x900")  # Увеличен размер окна для
размещения графика
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Вес левой панели
        main_frame.columnconfigure(1, weight=1)  # Вес правой панели
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===
```

```
        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(2, weight=1)

        # Выбор уровня ответственности с кнопками
        ttk.Label(self.input_frame, text="Уровень ответственности
исследования:",
                  font=("Arial", 10, "bold")).grid(row=0, column=0,
sticky=tk.W, pady=(0, 5))

        responsibility_frame = ttk.Frame(self.input_frame)
        responsibility_frame.grid(row=1, column=0, sticky="ew", pady=(0,
5))

        # Переменная для хранения выбранного уровня ответственности
        self.responsibility_level = "обычная"

        # Создание кнопок для выбора уровня ответственности
        self.resp_normal_btn = tk.Button(
            responsibility_frame,
            text="Обычная (β≥0.95, α≤0.05)",
            font=("Arial", 10),
            command=lambda: self.set_responsibility("обычная"),
            relief=tk.RAISED,
            bd=2,
            bg="#90EE90",  # Активная кнопка - зеленая
            fg="black"
        )
        self.resp_high_btn = tk.Button(
            responsibility_frame,
            text="Повышенная (β≥0.99, α≤0.01)",
            font=("Arial", 10),
            command=lambda: self.set_responsibility("повышенная"),
            relief=tk.RAISED,
            bd=2,
            bg="#F0F0F0",  # Неактивная кнопка - серая
            fg="black"
        )
        self.resp_very_high_btn = tk.Button(
            responsibility_frame,
            text="Высокая (β≥0.999, α≤0.001)",
            font=("Arial", 10),
            command=lambda: self.set_responsibility("высокая"),
            relief=tk.RAISED,
            bd=2,
            bg="#F0F0F0",  # Неактивная кнопка - серая
            fg="black"
        )

        self.resp_normal_btn.pack(side=tk.LEFT, padx=(0, 10))
        self.resp_high_btn.pack(side=tk.LEFT, padx=(0, 10))
        self.resp_very_high_btn.pack(side=tk.LEFT)

        # Верхнее окно для ввода исходных данных с горизонтальной и
```

```
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=8,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=2, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=2, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=3, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета
        self.calc_button = tk.Button(
            left_panel,
            text="Расчет",
            bg="#90EE90",
            font=("Arial", 12, "bold"),
            command=self.calculate,
            relief=tk.RAISED,
            bd=2
        )
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 2))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=2)
```

```
        # Кнопка "Помощь" светло-голубого цвета
        self.help_button = tk.Button(
            button_frame,
            text="Помощь",
            bg="#ADD8E6",
            font=("Arial", 12, "bold"),
            command=self.show_help,
            relief=tk.RAISED,
            bd=2
        )
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(
            button_frame,
            text="Записать отчет в виде файла",
            bg="#ADD8E6",
            font=("Arial", 12, "bold"),
            command=self.save_report,
            relief=tk.RAISED,
            bd=2
        )
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическое представление:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 6), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

        # Первоначальный расчет и построение графика
        self.calculate()

    def set_responsibility(self, level):
        """Установка уровня ответственности и обновление интерфейса"""
        self.responsibility_level = level

        # Сброс всех кнопок в неактивное состояние
        self.resp_normal_btn.config(bg="#F0F0F0", relief=tk.RAISED)
        self.resp_high_btn.config(bg="#F0F0F0", relief=tk.RAISED)
```

```
        self.resp_very_high_btn.config(bg="#F0F0F0", relief=tk.RAISED)

        # Активация выбранной кнопки
        if level == "обычная":
            self.resp_normal_btn.config(bg="#90EE90", relief=tk.SUNKEN)
        elif level == "повышенная":
            self.resp_high_btn.config(bg="#90EE90", relief=tk.SUNKEN)
        elif level == "высокая":
            self.resp_very_high_btn.config(bg="#90EE90", relief=tk.SUNKEN)

        # Автоматический пересчет при изменении уровня ответственности
        self.update_calculations_and_plot()

    def fill_sample_data(self):
        """Заполнение примерными данными"""
        self.input_text.delete(1.0, tk.END)

        sample_data = """Первая выборка (группа 1):
12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6

Вторая выборка (группа 2):
15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self, data_str):
        """Парсинг входных данных"""
        lines = [line.strip() for line in data_str.split('\n') if
line.strip()]

        group1_data = []
        group2_data = []
        current_group = None

        for line in lines:
            if 'первая' in line.lower() or 'группа 1' in line.lower():
                current_group = 1
                continue
            elif 'вторая' in line.lower() or 'группа 2' in line.lower():
                current_group = 2
                continue

            # Попытка извлечь числа из строки
            try:
                numbers = [float(x) for x in line.split() if x.replace('.',
'').replace('-', '').isdigit()]
                if numbers:
                    if current_group == 1:
                        group1_data.extend(numbers)
                    elif current_group == 2:
                        group2_data.extend(numbers)
                    elif not group1_data:  # Если группа не указана, первые
числа идут в группу 1
                        group1_data.extend(numbers)
                        current_group = 1
                    else:  # Последующие числа идут в группу 2
                        group2_data.extend(numbers)
                        current_group = 2
            except ValueError:
                continue
```

```
        return group1_data, group2_data

    def plot_comparison(self, group1, group2, stats_results):
        """Построение графиков сравнения групп"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание subplot с двумя графиками
        ax1 = self.fig.add_subplot(2, 1, 1)
        ax2 = self.fig.add_subplot(2, 1, 2)

        # График 1: Боксплот для сравнения распределений
        data_to_plot = [group1, group2]
        labels = ['Группа 1', 'Группа 2']

        bp = ax1.boxplot(data_to_plot, tick_labels=labels,
patch_artist=True, notch=True)

        # Раскраска боксплотов
        colors = ['lightblue', 'lightcoral']
        for patch, color in zip(bp['boxes'], colors):
            patch.set_facecolor(color)

        ax1.set_title('Сравнение распределений выборок', fontsize=12,
fontweight='bold')
        ax1.set_ylabel('Значения')
        ax1.grid(True, alpha=0.3)

        # Добавление средних значений на боксплот
        means = [stats_results['mean1'], stats_results['mean2']]
        for i, mean in enumerate(means):
            ax1.plot(i + 1, mean, 'ro', markersize=8,
markerfacecolor='red', markeredgecolor='darkred')
            ax1.annotate(f'M = {mean:.2f}', (i + 1, mean), xytext=(5, 5),
                         textcoords='offset points', fontsize=10,
fontweight='bold')

        # График 2: Столбчатая диаграмма со статистиками
        categories = ['Среднее', 'Std. откл.', 'Размер выборки']
        group1_stats = [stats_results['mean1'], stats_results['std1'],
stats_results['n1']]
        group2_stats = [stats_results['mean2'], stats_results['std2'],
stats_results['n2']]

        x = np.arange(len(categories))
        width = 0.35

        bars1 = ax2.bar(x - width / 2, group1_stats, width, label='Группа
1', color='lightblue', alpha=0.8)
        bars2 = ax2.bar(x + width / 2, group2_stats, width, label='Группа
2', color='lightcoral', alpha=0.8)

        # Добавление значений на столбцы
        for bars in [bars1, bars2]:
            for bar in bars:
                height = bar.get_height()
                ax2.annotate(f'{height:.2f}', xy=(bar.get_x() +
bar.get_width() / 2, height),
                             xytext=(0, 3), textcoords="offset points",
ha='center', va='bottom', fontsize=9)
```

```
        ax2.set_title('Сравнение основных статистик', fontsize=12,
fontweight='bold')
        ax2.set_ylabel('Значения')
        ax2.set_xticks(x)
        ax2.set_xticklabels(categories)
        ax2.legend()
        ax2.grid(True, alpha=0.3)

        # Получение критических значений и уровней ответственности
        thresholds = {
            "обычная": {"beta": 0.95, "alpha": 0.05},
            "повышенная": {"beta": 0.99, "alpha": 0.01},
            "высокая": {"beta": 0.999, "alpha": 0.001}
        }

        current_threshold = thresholds[self.responsibility_level]

        # Цвет для достоверности
        significance_color = 'green' if stats_results["is_significant"]
else 'red'
        significance_text = "ДА" if stats_results["is_significant"] else
"НЕТ"

        # Обновленная статистическая информация с акцентом на уровень
ответственности
        info_text = (f't-статистика: {stats_results["t_stat"]:.3f}\n'
                     f'p-value: {stats_results["p_value"]:.6f}\n'
                     f'Разность средних:
{stats_results["mean_diff"]:.3f}\n'
                     f'Достоверность: {significance_text}\n'
                     f'Уровень: {self.responsibility_level}\n'
                     f'Порог α: {current_threshold["alpha"]}\n'
                     f'Порог β: {current_threshold["beta"]}')

        # Изменение цвета фона в зависимости от достоверности
        bg_color = 'lightgreen' if stats_results["is_significant"] else
'lightcoral'

        ax2.text(0.02, 0.98, info_text, transform=ax2.transAxes,
fontsize=10,
                 verticalalignment='top', bbox=dict(boxstyle='round',
facecolor=bg_color, alpha=0.8))

        # Настройка layout
        self.fig.tight_layout()

        # Координатные оси отображаются в последнюю очередь
        for ax in [ax1, ax2]:
            ax.spines['bottom'].set_linewidth(1.5)
            ax.spines['left'].set_linewidth(1.5)
            ax.spines['top'].set_visible(False)
            ax.spines['right'].set_visible(False)

        # Обновление canvas
        self.canvas.draw()

    def update_calculations_and_plot(self):
        """Обновление расчетов и графика при изменении параметров"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
```

```
            if not data_str:
                return

            group1, group2 = self.parse_input_data(data_str)

            if len(group1) < 2 or len(group2) < 2:
                return

            # Выполнение расчетов с текущим уровнем ответственности
            result, stats_results = self.perform_calculations(group1,
group2, self.responsibility_level)

            # Обновление результатов
            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Обновление графика
            self.plot_comparison(group1, group2, stats_results)

        except Exception as e:
            print(f"Ошибка при обновлении: {str(e)}")

    def calculate(self):
        """Выполнение расчетов по алгоритму"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()

            if not data_str:
                messagebox.showerror("Ошибка", "Введите исходные данные")
                return

            group1, group2 = self.parse_input_data(data_str)

            if len(group1) < 2 or len(group2) < 2:
                messagebox.showerror("Ошибка", "Каждая группа должна
содержать минимум 2 значения")
                return

            # Использование текущего уровня ответственности
            responsibility = self.responsibility_level

            # Выполнение расчетов
            result, stats_results = self.perform_calculations(group1,
group2, responsibility)

            # Отображение результатов
            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графика
            self.plot_comparison(group1, group2, stats_results)

        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def perform_calculations(self, group1, group2, responsibility):
```

```
        """Выполнение статистических расчетов"""
        n1, n2 = len(group1), len(group2)
        mean1, mean2 = sum(group1) / n1, sum(group2) / n2

        # Вычисление дисперсий
        var1 = sum((x - mean1) ** 2 for x in group1) / (n1 - 1) if n1 > 1
else 0
        var2 = sum((x - mean2) ** 2 for x in group2) / (n2 - 1) if n2 > 1
else 0

        std1, std2 = math.sqrt(var1), math.sqrt(var2)

        # Разность средних
        mean_diff = mean1 - mean2

        # Стандартная ошибка разности средних
        se_diff = math.sqrt(var1 / n1 + var2 / n2)

        # t-критерий
        if se_diff != 0:
            t_stat = abs(mean_diff) / se_diff
            df = n1 + n2 - 2  # степени свободы

            # Вычисление p-value (двусторонний тест)
            p_value = 2 * (1 - stats.t.cdf(abs(t_stat), df))

            # Определение критических значений для разных уровней
ответственности
            thresholds = {
                "обычная": {"beta": 0.95, "alpha": 0.05},
                "повышенная": {"beta": 0.99, "alpha": 0.01},
                "высокая": {"beta": 0.999, "alpha": 0.001}
            }

            threshold = thresholds[responsibility]
            t_critical = stats.t.ppf(1 - threshold["alpha"] / 2, df)

            # Определение достоверности
            is_significant = p_value <= threshold["alpha"]
            beta_actual = 1 - p_value  # Вероятность безошибочного прогноза
            alpha_actual = p_value  # Вероятность ошибочного прогноза

        else:
            t_stat = 0
            p_value = 1
            t_critical = 0
            is_significant = False
            beta_actual = 0
            alpha_actual = 1
            df = n1 + n2 - 2

        # Статистики для графика
        stats_results = {
            'mean1': mean1, 'mean2': mean2, 'std1': std1, 'std2': std2,
            'n1': n1, 'n2': n2, 'mean_diff': mean_diff, 't_stat': t_stat,
            'p_value': p_value, 'is_significant': is_significant
        }

        # Формирование результата
        result = f"""ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ СРЕДНИХ
```

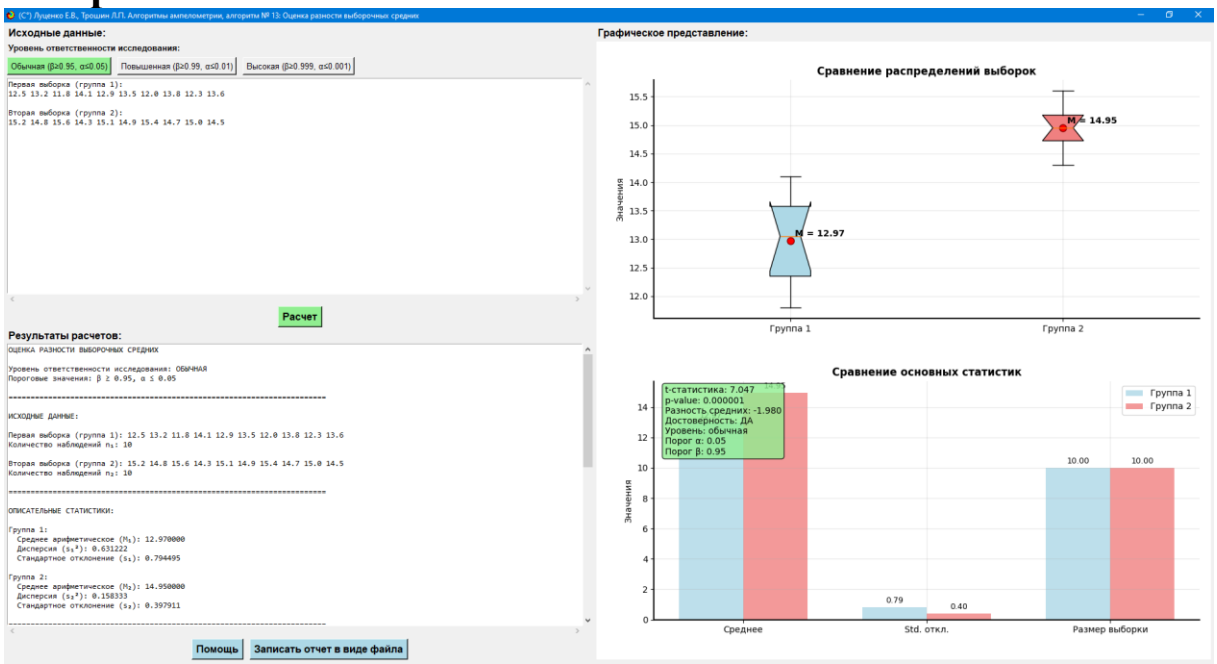
```
Уровень ответственности исследования: {responsibility.upper()}
Пороговые значения: β ≥ {thresholds[responsibility]["beta"]}, α ≤
{thresholds[responsibility]["alpha"]}

================================================================================

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка (группа 1): {' '.join(map(str, group1))}
Количество наблюдений n₁: {n1}

Вторая выборка (группа 2): {' '.join(map(str, group2))}
Количество наблюдений n₂: {n2}

================================================================================

ОПИСАТЕЛЬНЫЕ СТАТИСТИКИ:

Группа 1:
  Среднее арифметическое (M₁): {mean1:.6f}
  Дисперсия (s₁²): {var1:.6f}
  Стандартное отклонение (s₁): {std1:.6f}

Группа 2:
  Среднее арифметическое (M₂): {mean2:.6f}
  Дисперсия (s₂²): {var2:.6f}
  Стандартное отклонение (s₂): {std2:.6f}

================================================================================

АНАЛИЗ РАЗНОСТИ СРЕДНИХ:

Разность выборочных средних (M₁ - M₂): {mean_diff:.6f}
Стандартная ошибка разности: {se_diff:.6f}

t-статистика: {t_stat:.6f}
Критическое значение t: {t_critical:.6f}
Степени свободы: {df}

Фактическая вероятность ошибочного прогноза (α): {alpha_actual:.6f}
Фактическая вероятность безошибочного прогноза (β): {beta_actual:.6f}

================================================================================

ЗАКЛЮЧЕНИЕ О ДОСТОВЕРНОСТИ:

Сравнение с пороговыми значениями:
- Требуется: β ≥ {thresholds[responsibility]["beta"]}, α ≤
{thresholds[responsibility]["alpha"]}
- Фактически: β = {beta_actual:.6f}, α = {alpha_actual:.6f}

"""

        if is_significant:
            result += f"""РАЗНОСТЬ СРЕДНИХ ДОСТОВЕРНА ✓

Условия достоверности выполнены:
✓ α = {alpha_actual:.6f} ≤ {thresholds[responsibility]["alpha"]}
(пороговое значение)
✓ β = {beta_actual:.6f} ≥ {thresholds[responsibility]["beta"]} (пороговое
```

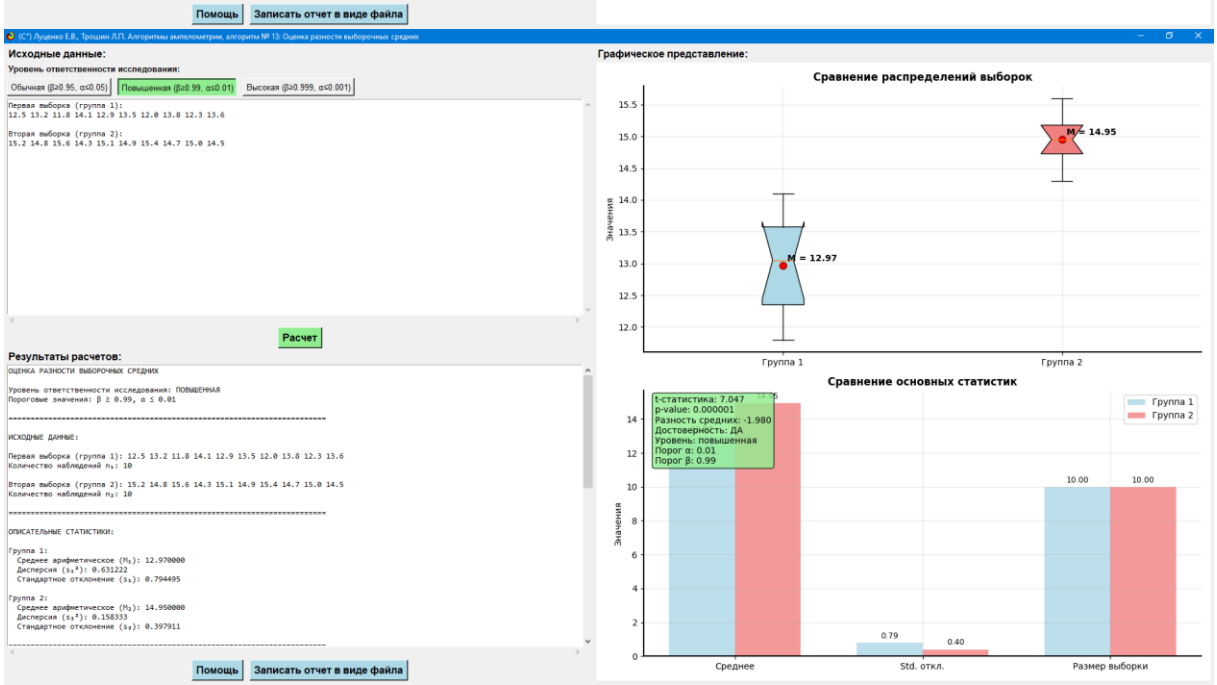
```
значение)

Можно утверждать с вероятностью {beta_actual:.1%}, что обнаруженная
разность
между выборочными средними отражает действительное различие в генеральных
совокупностях при выбранном уровне ответственности исследования."""
        else:
            result += f"""РАЗНОСТЬ СРЕДНИХ НЕДОСТОВЕРНА ✗

Условия достоверности НЕ выполнены:
✗ α = {alpha_actual:.6f} > {thresholds[responsibility]["alpha"]}
(пороговое значение)
✗ β = {beta_actual:.6f} < {thresholds[responsibility]["beta"]} (пороговое
значение)

Невозможно сделать достоверный прогноз относительно величины и знака
разности между соответствующими генеральными совокупностями при выбранном
уровне ответственности исследования."""

        result += f"""

================================================================================

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ:

Данное статистическое заключение касается только оценки достоверности
разности выборочных средних и не имеет отношения к:
- Организационной правильности проведения эксперимента
- Точности получения первичных данных
- Биологической или практической значимости различий

Рекомендация: {'Различия между группами статистически значимы' if
is_significant else 'Статистически значимых различий между группами не
обнаружено'}"""

        return result, stats_results

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-13.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")
```

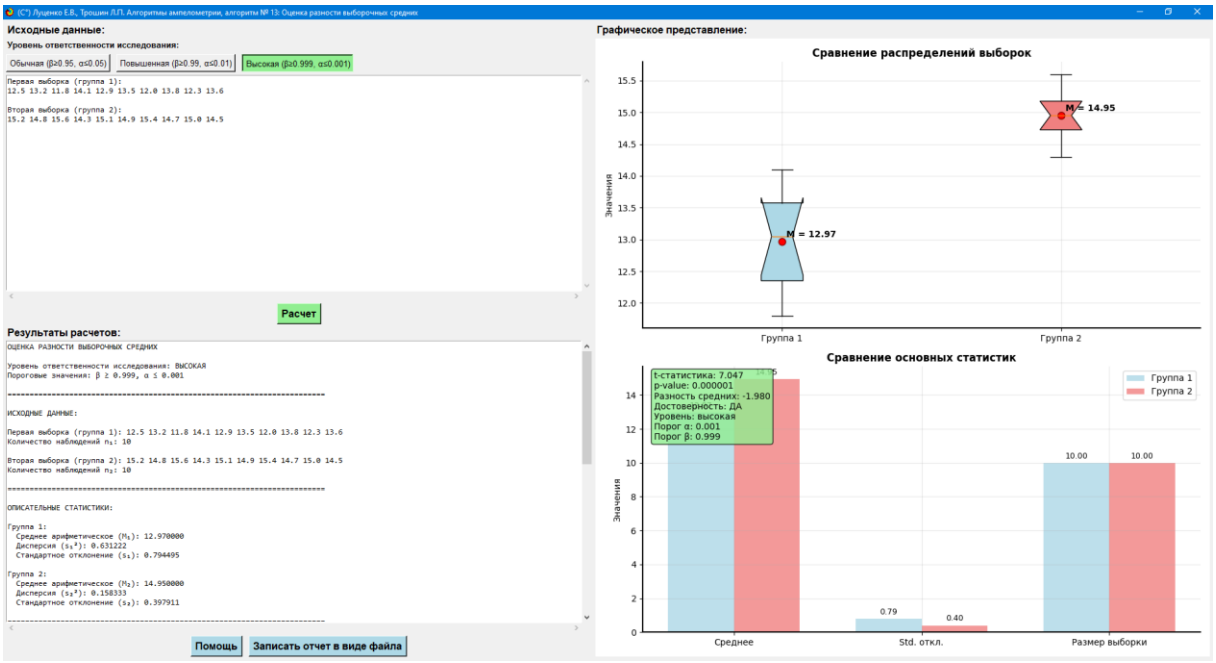
```
    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            responsibility = self.responsibility_level

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 13
Оценка разности выборочных средних

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
Уровень ответственности: {responsibility}

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическое представление сравнения выборок
отображается в графической области программы.

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_13_Оценка_разности_выборочных_средних_{datetime.now().strftime('
%Y%m%d_%H%M%S')}.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = BiometryAlgorithm13GUI(root)
    root.mainloop()
```

```
if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм.







9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов.

ОТЧЕТ ПО АЛГОРИТМУ № 13

Оценка разности выборочных средних

Дата и время расчета: 2025-07-24 05:58:09
Программа: (С°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
Уровень ответственности: обычная
==========

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка (группа 1):
12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6

Вторая выборка (группа 2):
15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
==========

ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ СРЕДНИХ

Уровень ответственности исследования: ОБЫЧНАЯ
Пороговые значения: $\beta \geq 0.95$, $\alpha \leq 0.05$
==========

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка (группа 1): 12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6
Количество наблюдений n_1: 10

Вторая выборка (группа 2): 15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
Количество наблюдений n_2: 10
==========
ОПИСАТЕЛЬНЫЕ СТАТИСТИКИ:

Группа 1:
  Среднее арифметическое (M_1): 12.970000
  Дисперсия (s_1^2): 0.631222
  Стандартное отклонение (s_1): 0.794495

Группа 2:
  Среднее арифметическое (M_2): 14.950000
  Дисперсия (s_2^2): 0.158333
  Стандартное отклонение (s_2): 0.397911
==========
АНАЛИЗ РАЗНОСТИ СРЕДНИХ:

Разность выборочных средних ($M_1 - M_2$): -1.980000
Стандартная ошибка разности: 0.280990

t-статистика: 7.046506
Критическое значение t: 2.100922
Степени свободы: 18

Фактическая вероятность ошибочного прогноза (α): 0.000001
Фактическая вероятность безошибочного прогноза (β): 0.999999
==========
ЗАКЛЮЧЕНИЕ О ДОСТОВЕРНОСТИ:

Сравнение с пороговыми значениями:
- Требуется: $\beta \geq 0.95$, $\alpha \leq 0.05$
- Фактически: $\beta = 0.999999$, $\alpha = 0.000001$

РАЗНОСТЬ СРЕДНИХ ДОСТОВЕРНА ✓

Условия достоверности выполнены:
✓ $\alpha = 0.000001 \leq 0.05$ (пороговое значение)
✓ $\beta = 0.999999 \geq 0.95$ (пороговое значение)

Можно утверждать с вероятностью 100.0%, что обнаруженная разность между выборочными средними отражает действительное различие в генеральных совокупностях при выбранном уровне ответственности исследования.
==========
ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ:

Данное статистическое заключение касается только оценки достоверности разности выборочных средних и не имеет отношения к:
- Организационной правильности проведения эксперимента
- Точности получения первичных данных
- Биологической или практической значимости различий

Рекомендация: Различия между группами статистически значимы
==========
Конец отчета

ОТЧЕТ ПО АЛГОРИТМУ № 13
Оценка разности выборочных средних

Дата и время расчета: 2025-07-24 05:58:15
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
Уровень ответственности: повышенная
==========
ИСХОДНЫЕ ДАННЫЕ:
Первая выборка (группа 1):
12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6

Вторая выборка (группа 2):
15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
==========
ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ СРЕДНИХ

Уровень ответственности исследования: ПОВЫШЕННАЯ
Пороговые значения: $\beta \geq 0.99$, $\alpha \leq 0.01$
==========
ИСХОДНЫЕ ДАННЫЕ:

Первая выборка (группа 1): 12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6
Количество наблюдений n_1: 10

Вторая выборка (группа 2): 15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
Количество наблюдений n_2: 10
==========
ОПИСАТЕЛЬНЫЕ СТАТИСТИКИ:

Группа 1:
  Среднее арифметическое (M_1): 12.970000
  Дисперсия (s_1^2): 0.631222
  Стандартное отклонение (s_1): 0.794495

Группа 2:
  Среднее арифметическое (M_2): 14.950000
  Дисперсия (s_2^2): 0.158333
  Стандартное отклонение (s_2): 0.397911
==========
АНАЛИЗ РАЗНОСТИ СРЕДНИХ:

Разность выборочных средних ($M_1 - M_2$): -1.980000
Стандартная ошибка разности: 0.280990

t-статистика: 7.046506
Критическое значение t: 2.878440
Степени свободы: 18

Фактическая вероятность ошибочного прогноза (α): 0.000001
Фактическая вероятность безошибочного прогноза (β): 0.999999
==========

ЗАКЛЮЧЕНИЕ О ДОСТОВЕРНОСТИ:

Сравнение с пороговыми значениями:
- Требуется: $\beta \geq 0.99$, $\alpha \leq 0.01$
- Фактически: $\beta = 0.999999$, $\alpha = 0.000001$

РАЗНОСТЬ СРЕДНИХ ДОСТОВЕРНА ✓

Условия достоверности выполнены:
✓ $\alpha = 0.000001 \leq 0.01$ (пороговое значение)
✓ $\beta = 0.999999 \geq 0.99$ (пороговое значение)

Можно утверждать с вероятностью 100.0%, что обнаруженная разность между выборочными средними отражает действительное различие в генеральных совокупностях при выбранном уровне ответственности исследования.
==========

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ:

Данное статистическое заключение касается только оценки достоверности разности выборочных средних и не имеет отношения к:
- Организационной правильности проведения эксперимента
- Точности получения первичных данных
- Биологической или практической значимости различий

Рекомендация: Различия между группами статистически значимы
==========

Конец отчета

ОТЧЕТ ПО АЛГОРИТМУ № 13

Оценка разности выборочных средних

Дата и время расчета: 2025-07-24 05:58:20

Программа: (С°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
Уровень ответственности: высокая
==========
ИСХОДНЫЕ ДАННЫЕ:
Первая выборка (группа 1):
12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6

Вторая выборка (группа 2):
15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
==========
ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ СРЕДНИХ
\Уровень ответственности исследования: ВЫСОКАЯ
Пороговые значения: $\beta \geq 0.999$, $\alpha \leq 0.001$
==========
ИСХОДНЫЕ ДАННЫЕ:

Первая выборка (группа 1): 12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6
Количество наблюдений n_1: 10

Вторая выборка (группа 2): 15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5
Количество наблюдений n_2: 10
==========
ОПИСАТЕЛЬНЫЕ СТАТИСТИКИ:

Группа 1:
Среднее арифметическое (M_1): 12.970000
Дисперсия (s_1^2): 0.631222
Стандартное отклонение (s_1): 0.794495

Группа 2:
Среднее арифметическое (M_2): 14.950000
Дисперсия (s_2^2): 0.158333
Стандартное отклонение (s_2): 0.397911
==========
АНАЛИЗ РАЗНОСТИ СРЕДНИХ:

Разность выборочных средних (M_1 - M_2): -1.980000
Стандартная ошибка разности: 0.280990

t-статистика: 7.046506
Критическое значение t: 3.921646
Степени свободы: 18

Фактическая вероятность ошибочного прогноза (α): 0.000001
Фактическая вероятность безошибочного прогноза (β): 0.999999
==========

ЗАКЛЮЧЕНИЕ О ДОСТОВЕРНОСТИ:

Сравнение с пороговыми значениями:
- Требуется: $\beta \geq 0.999$, $\alpha \leq 0.001$
- Фактически: $\beta = 0.999999$, $\alpha = 0.000001$

РАЗНОСТЬ СРЕДНИХ ДОСТОВЕРНА ✓

Условия достоверности выполнены:
✓ $\alpha = 0.000001 \leq 0.001$ (пороговое значение)
✓ $\beta = 0.999999 \geq 0.999$ (пороговое значение)

Можно утверждать с вероятностью 100.0%, что обнаруженная разность между выборочными средними отражает действительное различие в генеральных совокупностях при выбранном уровне ответственности исследования.
==========

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ:

Данное статистическое заключение касается только оценки достоверности разности выборочных средних и не имеет отношения к:
- Организационной правильности проведения эксперимента
- Точности получения первичных данных
- Биологической или практической значимости различий

Рекомендация: Различия между группами статистически значимы
==========
Конец отчета

10. Инструкция пользователю по программе: Алгоритм № 13 - Оценка разности выборочных средних
Версия: 1.0
Разработчики: (С°) Луценко Е.В., Трошин Л.П.
Дата: 2024

---

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программа предназначена для автоматизации расчетов по алгоритму оценки статистической достоверности разности между выборочными средними. Программа определяет, является ли разность между двумя группами данных статистически значимой, с учетом различных уровней ответственности исследования.

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

- **Операционная система:** Windows 7/8/10/11
- **Оперативная память:** минимум 512 МБ
- **Свободное место на диске:** 50 МБ
- **Дополнительное ПО:** не требуется (программа работает автономно)

3. УСТАНОВКА И ЗАПУСК

1. Скопируйте исполняемый файл программы (main13.exe) в любую папку на компьютере
2. В той же папке должны находиться файлы:
    - _Aidos.ico (иконка программы)
    - help-13.pdf (файл справки)
3. Запустите программу двойным щелчком по файлу main13.exe

4. ОПИСАНИЕ ИНТЕРФЕЙСА

4.1 Главное окно программы

Интерфейс программы состоит из следующих элементов:

Верхняя часть:

- **Заголовок:** отображает название программы и алгоритма

- **Поле "Исходные данные":** содержит область для ввода данных двух выборок

Средняя часть:

- **Переключатели уровня ответственности:** три варианта для выбора
- **Кнопка "Расчет":** зеленая кнопка для запуска вычислений

Нижняя часть:

- **Поле "Результаты расчетов":** отображает результаты анализа
- **Кнопка "Помощь":** открывает файл справки
- **Кнопка "Записать отчет в виде файла":** сохраняет результаты в текстовый файл

4.2 Уровни ответственности исследования

Программа поддерживает три уровня ответственности:

1. **Обычная:** $\beta \geq 0.95$, $\alpha \leq 0.05$ (для рутинных исследований)
2. **Повышенная:** $\beta \geq 0.99$, $\alpha \leq 0.01$ (для важных исследований)
3. **Высокая:** $\beta \geq 0.999$, $\alpha \leq 0.001$ (для критически важных исследований)

Где:

- **β** — вероятность безошибочного прогноза
- **α** — вероятность ошибочного прогноза

5. ПОРЯДОК РАБОТЫ С ПРОГРАММОЙ

5.1 Подготовка данных

Перед началом работы подготовьте две выборки данных (две группы наблюдений). Данные должны быть численными значениями.

5.2 Ввод исходных данных

1. В верхнем поле "Исходные данные" введите данные в следующем формате:

Первая выборка (группа 1):
12.5 13.2 11.8 14.1 12.9 13.5 12.0 13.8 12.3 13.6

Вторая выборка (группа 2):
15.2 14.8 15.6 14.3 15.1 14.9 15.4 14.7 15.0 14.5

Правила ввода данных:

- Каждая группа должна быть подписана
- Числа в группе разделяются пробелами
- Можно использовать как целые, так и десятичные числа
- Каждая группа должна содержать минимум 2 значения

5.3 Выбор уровня ответственности

Выберите один из трех радиопереключателей в зависимости от важности вашего исследования:

- **Обычная** — для большинства исследований
- **Повышенная** — для важных решений
- **Высокая** — для критически важных исследований

5.4 Выполнение расчетов

1. Нажмите кнопку **"Расчет"**
2. Программа автоматически выполнит статистический анализ
3. Результаты появятся в нижнем окне

5.5 Интерпретация результатов

В окне результатов отображается:

Описательные статистики:

- Средние арифметические для каждой группы
- Дисперсии и стандартные отклонения
- Количество наблюдений

Статистический анализ:

- Разность выборочных средних
- t-статистика и критическое значение
- Фактические значения α и β

Заключение о достоверности:

- **ДОСТОВЕРНА ✓** — различия статистически значимы
- **НЕДОСТОВЕРНА ✗** — различия статистически незначимы

6. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИИ

6.1 Справка

Нажмите кнопку **"Помощь"** для открытия подробного описания алгоритма в формате PDF.

6.2 Сохранение отчета

1. После выполнения расчетов нажмите кнопку **"Записать отчет в виде файла"**
2. Программа создаст текстовый файл с полным отчетом в папке программы
3. Имя файла будет содержать дату и время создания

7. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

7.1 Ошибки ввода данных

Проблема: "Каждая группа должна содержать минимум 2 значения" **Решение:** Убедитесь, что в каждой группе есть не менее 2 чисел

Проблема: "Введите исходные данные" **Решение:** Заполните поле исходных данных

Проблема: "Проверьте правильность ввода данных" **Решение:**

- Используйте только числа
- Разделяйте числа пробелами
- Проверьте корректность десятичных дробей (используйте точку, не запятую)

7.2 Системные ошибки

Проблема: Программа не запускается **Решение:**

- Проверьте наличие файла _Aidos.ico в папке с программой
- Запустите программу от имени администратора
- Проверьте, не блокирует ли антивирус исполняемый файл

Проблема: Не открывается справка **Решение:** Убедитесь, что файл help-13.pdf находится в той же папке, что и программа

8. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

8.1 Пример 1: Сравнение урожайности

Сравните урожайность двух сортов винограда:
Первая выборка (группа 1):
45.2 47.1 44.8 46.3 45.9 46.7 45.4 47.2 46.0 46.5

Вторая выборка (группа 2):
42.1 41.8 42.9 41.5 42.3 41.9 42.7 41.3 42.0 42.4

Уровень ответственности: Обычная
Ожидаемый результат: Разность достоверна (урожайность первого сорта значительно выше)

8.2 Пример 2: Анализ качества продукции
Сравните содержание сахара в ягодах с разных участков:
Первая выборка (группа 1):
18.5 19.2 18.8 19.0 18.7 19.1 18.9 19.3 18.6 19.0

Вторая выборка (группа 2):
18.3 18.9 18.6 18.8 18.4 18.7 18.5 18.9 18.2 18.6
Уровень ответственности: Повышенная
Ожидаемый результат: Возможно, разность недостоверна (различия незначительны)

9. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

9.1 Суть алгоритма
Алгоритм основан на статистическом сравнении двух выборочных средних с помощью t-критерия Стьюдента. Основная задача — определить, отражает ли наблюдаемая разность между выборками действительное различие в генеральных совокупностях.

9.2 Критерии оценки
Достоверная разность означает:
- Высокую вероятность того, что различие не случайно
- Возможность обобщения результатов на генеральную совокупность
- Достаточную уверенность для принятия решений

Недостоверная разность означает:
- Различия могут быть случайными
- Нельзя делать уверенных прогнозов
- Требуются дополнительные исследования

9.3 Ограничения метода
Статистическая достоверность НЕ гарантирует:
- Практическую значимость различий
- Правильность организации эксперимента
- Точность измерений

- Биологическую или экономическую важность результатов

10. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

10.1 Вычислительные возможности

- **Размер выборок:** от 2 до 10000 наблюдений в каждой группе
- **Точность вычислений:** до 6 знаков после запятой
- **Типы данных:** целые и десятичные числа
- **Диапазон значений:** от -999999 до 999999

10.2 Формат выходных данных

Программа предоставляет:

- Подробные описательные статистики
- Результаты статистических тестов
- Четкое заключение о достоверности
- Рекомендации по интерпретации

11. ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ

В: Какой уровень ответственности выбрать? **О:** Для большинства исследований подходит "Обычная". Выберите "Повышенная" для важных решений или "Высокая" для критически важных исследований.

В: Что делать, если группы имеют разный размер? **О:** Программа корректно работает с группами разного размера. Минимум — 2 наблюдения в каждой группе.

В: Можно ли анализировать более двух групп? **О:** Данная программа предназначена только для сравнения двух групп. Для множественных сравнений используйте другие методы.

В: Что означает "недостоверная разность"? **О:** Это означает, что наблюдаемые различия могут быть случайными, и нельзя с достаточной уверенностью утверждать о наличии реального различия между группами.

12. КОНТАКТНАЯ ИНФОРМАЦИЯ

При возникновении технических проблем или вопросов по использованию программы обратитесь к разработчикам:

Разработчики: Луценко Е.В., Трошин Л.П.

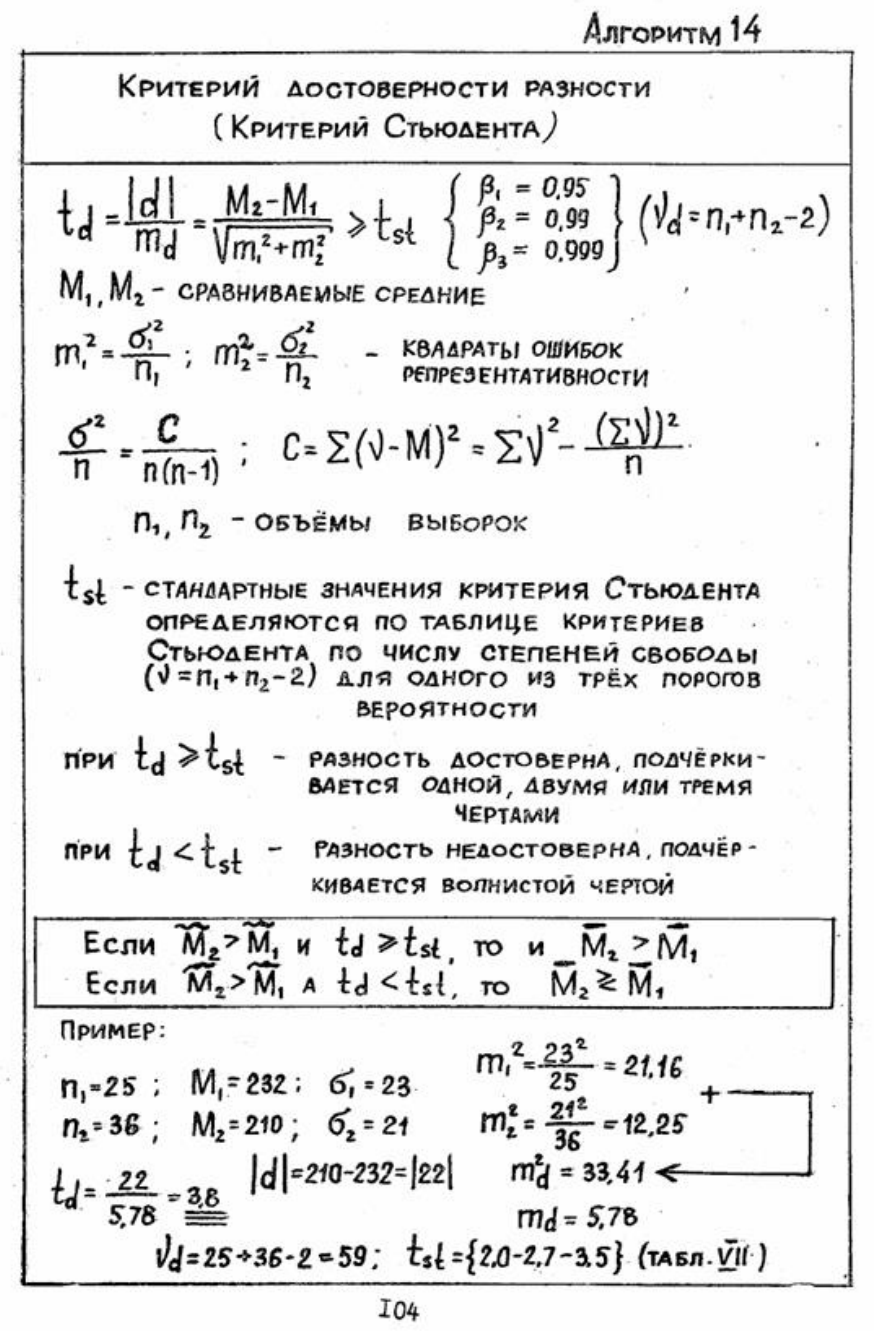
Программа: Алгоритмы ампелометрии

Алгоритм № 13: Оценка разности выборочных средних

Внимание: Данная программа является инструментом для статистического анализа. Окончательные выводы и решения должны приниматься с учетом профессионального опыта и знания предметной области.

Алгоритм-14: Критерий достоверности разности (Критерий Стьюдента)

1. Исходное изображение



Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.:

Изд-во Моск. ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму, в т.ч. используемые в формулах условные математические обозначения переменных

Представленный алгоритм описывает критерий достоверности разности (критерий Стьюдента), используемый для оценки статистической значимости различий между двумя средними значениями. Этот критерий широко применяется в биометрии и других областях для проверки гипотез о равенстве средних двух выборок.

Условные математические обозначения:

- t_d – эмпирическое значение критерия Стьюдента.
- $|d|$ – абсолютное значение разности между сравниваемыми средними M_2 и M_1.
- m_d – средняя квадратическая ошибка разности средних.
- M_1, M_2 – сравниваемые средние значения двух выборок.
- m_1^2, m_2^2 – квадраты ошибок репрезентативности для первой и второй выборок соответственно.
- σ_1^2, σ_2^2 – дисперсии (или квадраты стандартных отклонений) для первой и второй выборок соответственно.
- n_1, n_2 – объемы (размеры) первой и второй выборок соответственно.
- t_{st} – стандартное (табличное) значение критерия Стьюдента, определяемое по таблице критериев Стьюдента в зависимости от числа степеней свободы и выбранного порога вероятности.
- V_d – число степеней свободы, используемое для определения табличного значения t_{st}.
- $\beta_1, \beta_2, \beta_3$ – пороги вероятности (уровни значимости), используемые для сравнения с t_{st} (0.95, 0.99, 0.999).
- C – вспомогательная переменная для расчета дисперсии.
- $\sum(V - M)^2$ – сумма квадратов отклонений индивидуальных значений от среднего арифметического.
- $\sum V^2$ – сумма квадратов индивидуальных значений.
- $(\sum V)^2$ – квадрат суммы индивидуальных значений.

3. Текстовое описание математических формул в стандартном для MS Word-2010 виде

3.1. Критерий достоверности разности (критерий Стьюдента)

Эмпирическое значение критерия Стьюдента (t_d) рассчитывается по следующей формуле:

$$t_d = \frac{|d|}{m_d} = \frac{|M_2 - M_1|}{\sqrt{m_1^2 + m_2^2}}$$

где:

- $|d|$ – абсолютное значение разности между сравниваемыми средними M_2 и M_1.
- m_d – средняя квадратическая ошибка разности средних.

Для определения достоверности разности, эмпирическое значение t_d сравнивается со стандартным (табличным) значением t_{st}:

$$t_d \geq t_{st}$$

Стандартные значения t_{st} определяются по таблице критериев Стьюдента для числа степеней свободы V_d и выбранных порогов вероятности (уровней значимости):

$$V_d = n_1 + n_2 - 2$$

Используемые пороги вероятности:

- $\beta_1 = 0.95$
- $\beta_2 = 0.99$
- $\beta_3 = 0.999$

3.2. Квадраты ошибок репрезентативности

Квадраты ошибок репрезентативности для каждой выборки (m_1^2 и m_2^2) рассчитываются по формулам:

$$m_1^2 = \frac{\sigma_1^2}{n_1}$$

$$m_2^2 = \frac{\sigma_2^2}{n_2}$$

где:

- σ_1^2, σ_2^2 – дисперсии (или квадраты стандартных отклонений) для первой и второй выборок соответственно.
- n_1, n_2 – объемы (размеры) первой и второй выборок соответственно.

3.3. Расчет дисперсии

Дисперсия (σ^2) может быть рассчитана по формуле:

$$\frac{\sigma^2}{n} = \frac{C}{n(n-1)}$$

где C – вспомогательная переменная, рассчитываемая как:

$$C = \sum(V - M)^2 = \sum V^2 - \frac{(\sum V)^2}{n}$$

где:

- V – индивидуальные значения.
- M – среднее арифметическое.
- n – объем выборки.

4. Алгоритм расчетов в текстовом виде по шагам

Алгоритм проверки достоверности разности средних значений с использованием критерия Стьюдента включает следующие шаги:

1. **Сбор исходных данных:** Определить средние значения (M_1, M_2), объемы выборок (n_1, n_2) и стандартные отклонения (или дисперсии σ_1, σ_2) для двух сравниваемых групп.
2. **Расчет квадратов ошибок репрезентативности (m_1^2, m_2^2):**
   - Вычислить $m_1^2 = \frac{\sigma_1^2}{n_1}$
   - Вычислить $m_2^2 = \frac{\sigma_2^2}{n_2}$
3. **Расчет средней квадратической ошибки разности средних (m_d):**
   - Вычислить $m_d = \sqrt{m_1^2 + m_2^2}$
4. **Расчет абсолютного значения разности средних ($|d|$):**
   - Вычислить $|d| = |M_2 - M_1|$
5. **Расчет эмпирического значения критерия Стьюдента (t_d):**
   - Вычислить $t_d = \frac{|d|}{m_d}$
6. **Определение числа степеней свободы (V_d):**
   - Вычислить $V_d = n_1 + n_2 - 2$
7. **Определение стандартного (табличного) значения критерия Стьюдента (t_{st}):**
   - Используя число степеней свободы V_d и выбранные пороги вероятности (например, 0.95, 0.99, 0.999), найти

соответствующее значение t_{st} по таблице критериев Стьюдента (например, Таблица VII).

8. **Сравнение эмпирического и табличного значений (t_d и t_{st}):**
   - Если $t_d \geq t_{st}$: Разность достоверна. Это означает, что наблюдаемые различия между средними значениями статистически значимы.
   - Если $t_d < t_{st}$: Разность недостоверна. Это означает, что наблюдаемые различия могут быть случайными и не являются статистически значимыми.
9. **Интерпретация результатов с учетом средних значений:**
   - Если $M_2 > M_1$ и $t_d \geq t_{st}$, то $M_2 \geq M_1$ (разность достоверна).
   - Если $M_2 > M_1$ и $t_d < t_{st}$, то $M_2 \approx M_1$ (разность недостоверна).

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.

Исходные данные из примера:

- Объем первой выборки (n_1) = 25
- Среднее значение первой выборки (M_1) = 232
- Стандартное отклонение первой выборки (σ_1) = 23
- Объем второй выборки (n_2) = 36
- Среднее значение второй выборки (M_2) = 210
- Стандартное отклонение второй выборки (σ_2) = 21

Численные расчеты:

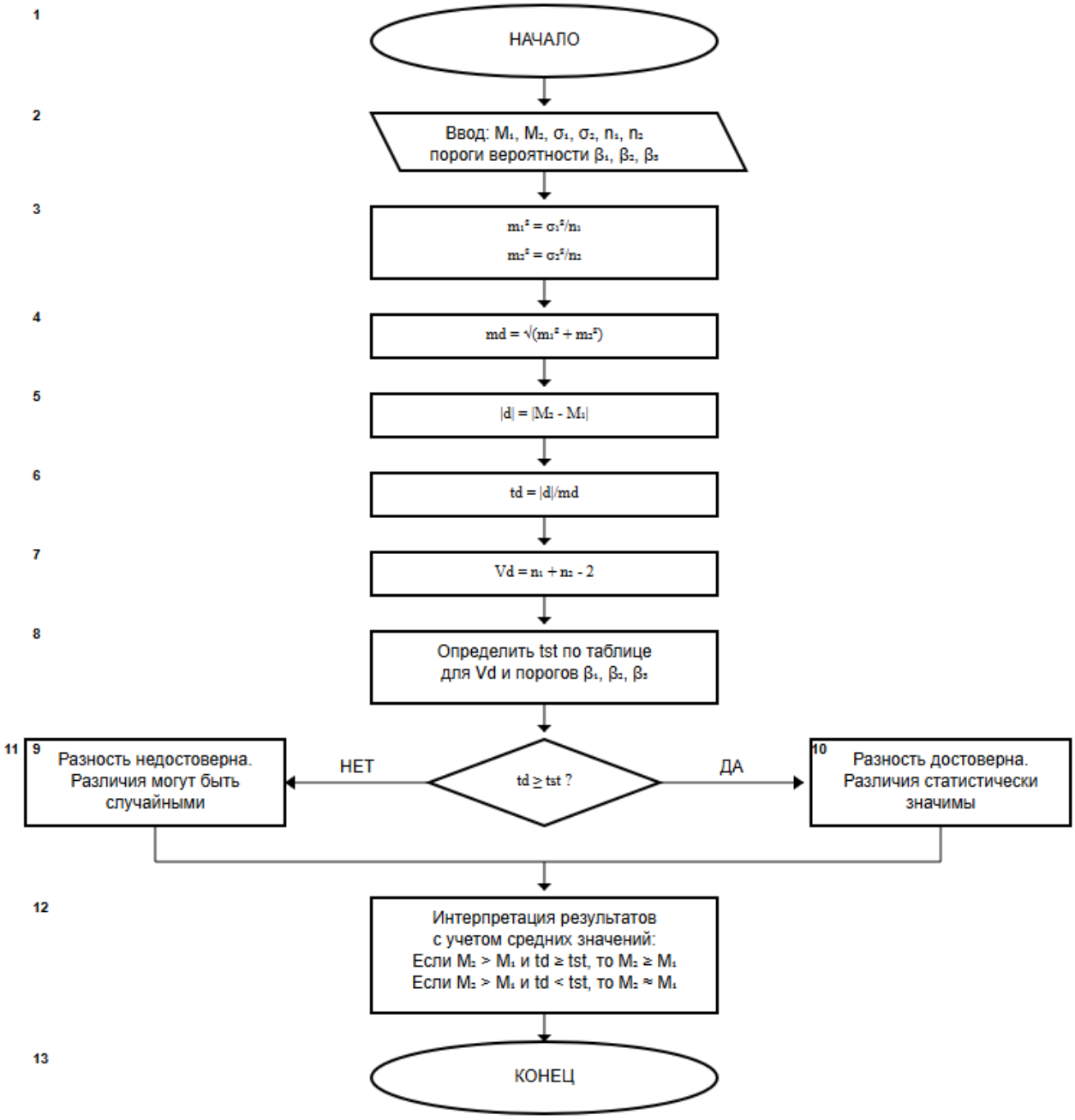
1. **Расчет квадратов ошибок репрезентативности:**
   - $m_1^2 = \frac{\sigma_1^2}{n_1} = \frac{23^2}{25} = \frac{529}{25} = 21.16$
   - $m_2^2 = \frac{\sigma_2^2}{n_2} = \frac{21^2}{36} = \frac{441}{36} = 12.25$
2. **Расчет средней квадратической ошибки разности средних (m_d):**
   - $m_d^2 = m_1^2 + m_2^2 = 21.16 + 12.25 = 33.41$
   - $m_d = \sqrt{33.41} \approx 5.78$
3. **Расчет абсолютного значения разности средних ($|d|$):**
   - $|d| = |M_2 - M_1| = |210 - 232| = |-22| = 22$

4. **Расчет эмпирического значения критерия Стьюдента (t_d):**
   - $t_d = \frac{|d|}{m_d} = \frac{22}{5.78} \approx 3.81$
5. **Определение числа степеней свободы (V_d):**
   - $V_d = n_1 + n_2 - 2 = 25 + 36 - 2 = 59$
6. **Определение стандартного (табличного) значения критерия Стьюдента (t_{st}):**
   - Для $V_d = 59$ и порогов вероятности (2.0, 2.7, 3.5) из Таблицы VII, t_{st} соответствует этим значениям. В данном примере t_{st} указан как диапазон {2.0 - 2.7 - 3.5}.

Вывод:

Поскольку $t_d \approx 3.81$ и это значение больше, чем все указанные пороговые значения t_{st} (2.0, 2.7, 3.5), разность между средними значениями M_1 и M_2 является статистически достоверной. Также, поскольку $M_1 = 232$ и $M_2 = 210$, то $M_1 > M_2$. Если бы $M_2 > M_1$ и $t_d \geq t_{st}$, то $M_2 \geq M_1$. В данном случае, так как $M_1 > M_2$ и разность достоверна, можно заключить, что M_1 статистически значимо отличается от M_2.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
```

```
import numpy as np


class StudentTestGUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 14: Критерий достоверности разности (Критерий Стьюдента)")
        self.root.geometry("1400x750")  # Увеличил ширину для графика
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Левая панель
        main_frame.columnconfigure(1, weight=1)  # Правая панель для
графика
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
```

```
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 1))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 1))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(0, weight=1)

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=8,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 1))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
```

```
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=1)

        # Кнопка "Помощь" светло-голубого цвета
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическая интерпретация:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(6, 8), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

    def fill_sample_data(self):
        """Заполнение исходными данными из примера"""
        self.input_text.delete(1.0, tk.END)
```

```
        sample_data = """Первая выборка:
n1 = 25
M1 = 232
σ1 = 23

Вторая выборка:
n2 = 36
M2 = 210
σ2 = 21"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self):
        """Парсинг введенных данных"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
            lines = data_str.split('\n')

            n1 = n2 = M1 = M2 = sigma1 = sigma2 = None

            for line in lines:
                line = line.strip()
                if 'n1' in line and '=' in line:
                    n1 = float(line.split('=')[1].strip())
                elif 'n2' in line and '=' in line:
                    n2 = float(line.split('=')[1].strip())
                elif 'M1' in line and '=' in line:
                    M1 = float(line.split('=')[1].strip())
                elif 'M2' in line and '=' in line:
                    M2 = float(line.split('=')[1].strip())
                elif 'σ1' in line and '=' in line:
                    sigma1 = float(line.split('=')[1].strip())
                elif 'σ2' in line and '=' in line:
                    sigma2 = float(line.split('=')[1].strip())

            if None in [n1, n2, M1, M2, sigma1, sigma2]:
                raise ValueError("Не все параметры найдены")

            return int(n1), int(n2), M1, M2, sigma1, sigma2

        except Exception as e:
            raise ValueError(f"Ошибка парсинга данных: {str(e)}")

    def plot_student_test_visualization(self, n1, n2, M1, M2, sigma1,
sigma2, td, t_st_values, probability_levels):
        """Построение графической интерпретации критерия Стьюдента"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание трех подграфиков
        ax1 = self.fig.add_subplot(3, 1, 1)
        ax2 = self.fig.add_subplot(3, 1, 2)
        ax3 = self.fig.add_subplot(3, 1, 3)

        # График 1: Сравнение средних значений выборок
        categories = ['Выборка 1', 'Выборка 2']
        means = [M1, M2]
        colors = ['skyblue', 'lightcoral']

        bars1 = ax1.bar(categories, means, color=colors, alpha=0.7,
```

```
edgecolor='black', linewidth=1)

        # Добавление значений на столбцы
        for bar, mean in zip(bars1, means):
            height = bar.get_height()
            ax1.text(bar.get_x() + bar.get_width() / 2., height +
max(means) * 0.01,
                     f'{mean:.1f}', ha='center', va='bottom',
fontweight='bold', fontsize=10)

        ax1.set_ylabel('Среднее значение (M)', fontsize=10)
        ax1.set_title('Сравнение средних значений выборок', fontsize=12,
fontweight='bold')
        ax1.grid(True, alpha=0.3, axis='y')

        # Добавление информации о выборках
        info_text1 = f'n₁={n1}, σ₁={sigma1}\nn₂={n2}, σ₂={sigma2}'
        ax1.text(0.02, 0.95, info_text1, transform=ax1.transAxes,
fontsize=9,
                 verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='wheat', alpha=0.8))

        # График 2: Визуализация ошибок репрезентативности
        m1_squared = (sigma1 ** 2) / n1
        m2_squared = (sigma2 ** 2) / n2

        error_categories = ['m₁²', 'm₂²']
        error_values = [m1_squared, m2_squared]

        bars2 = ax2.bar(error_categories, error_values, color=['lightblue',
'lightpink'],
                        alpha=0.7, edgecolor='black', linewidth=1)

        for bar, error in zip(bars2, error_values):
            height = bar.get_height()
            ax2.text(bar.get_x() + bar.get_width() / 2., height +
max(error_values) * 0.01,
                     f'{error:.3f}', ha='center', va='bottom',
fontweight='bold', fontsize=10)

        ax2.set_ylabel('Квадрат ошибки', fontsize=10)
        ax2.set_title('Ошибки репрезентативности выборок', fontsize=12,
fontweight='bold')
        ax2.grid(True, alpha=0.3, axis='y')

        # График 3: Сравнение эмпирического и табличных значений критерия
        t_labels = ['tᵈ (эмпир.)', 't₀.₀₅', 't₀.₀₁', 't₀.₀₀₁']
        t_values = [td] + t_st_values

        # Цвета: красный для эмпирического, зеленые градации для табличных
        colors_t = ['red', 'lightgreen', 'green', 'darkgreen']

        bars3 = ax3.bar(t_labels, t_values, color=colors_t, alpha=0.7,
edgecolor='black', linewidth=1)

        for bar, t_val in zip(bars3, t_values):
            height = bar.get_height()
            ax3.text(bar.get_x() + bar.get_width() / 2., height +
max(t_values) * 0.01,
                     f'{t_val:.3f}', ha='center', va='bottom',
fontweight='bold', fontsize=10)
```

```
        ax3.set_ylabel('Значение t-критерия', fontsize=10)
        ax3.set_title('Сравнение с табличными значениями', fontsize=12,
fontweight='bold')
        ax3.grid(True, alpha=0.3, axis='y')

        # Добавление горизонтальных линий для табличных значений
        for i, (t_st, prob) in enumerate(zip(t_st_values,
probability_levels)):
            ax3.axhline(y=t_st, color=colors_t[i + 1], linestyle='--',
alpha=0.7, linewidth=1)

        # Заключение о достоверности
        if td >= max(t_st_values):
            conclusion = "ВЫСОКОДОСТОВЕРНА"
            conclusion_color = 'darkgreen'
        elif td >= t_st_values[1]:
            conclusion = "ДОСТОВЕРНА (P≥0.99)"
            conclusion_color = 'green'
        elif td >= t_st_values[0]:
            conclusion = "ДОСТОВЕРНА (P≥0.95)"
            conclusion_color = 'orange'
        else:
            conclusion = "НЕДОСТОВЕРНА"
            conclusion_color = 'red'

        conclusion_text = f'Разность: {conclusion}'
        ax3.text(0.5, 0.95, conclusion_text, transform=ax3.transAxes,
fontsize=11,
                 verticalalignment='top', horizontalalignment='center',
fontweight='bold',
                 bbox=dict(boxstyle='round', facecolor=conclusion_color,
alpha=0.3))

        # Настройка layout
        self.fig.tight_layout(pad=2.0)

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по критерию Стьюдента"""
        try:
            n1, n2, M1, M2, sigma1, sigma2 = self.parse_input_data()

            # Расчет квадратов ошибок репрезентативности
            m1_squared = (sigma1 ** 2) / n1
            m2_squared = (sigma2 ** 2) / n2

            # Расчет средней квадратической ошибки разности средних
            md_squared = m1_squared + m2_squared
            md = math.sqrt(md_squared)

            # Расчет абсолютного значения разности средних
            d_abs = abs(M2 - M1)

            # Расчет эмпирического значения критерия Стьюдента
            td = d_abs / md

            # Определение числа степеней свободы
            Vd = n1 + n2 - 2
```

```
            # Табличные значения для примера (как в документе)
            t_st_values = [2.0, 2.7, 3.5]
            probability_levels = [0.95, 0.99, 0.999]

            # Формирование результата
            result = f"""РАСЧЕТ КРИТЕРИЯ ДОСТОВЕРНОСТИ РАЗНОСТИ (КРИТЕРИЙ
СТЬЮДЕНТА)

Исходные данные:
Первая выборка: n₁ = {n1}, M₁ = {M1}, σ₁ = {sigma1}
Вторая выборка: n₂ = {n2}, M₂ = {M2}, σ₂ = {sigma2}

Пошаговый расчет:

1. Расчет квадратов ошибок репрезентативности:
   m₁² = σ₁² / n₁ = {sigma1}² / {n1} = {sigma1 ** 2} / {n1} =
{m1_squared:.4f}
   m₂² = σ₂² / n₂ = {sigma2}² / {n2} = {sigma2 ** 2} / {n2} =
{m2_squared:.4f}

2. Расчет средней квадратической ошибки разности средних:
   m²ᵈ = m₁² + m₂² = {m1_squared:.4f} + {m2_squared:.4f} = {md_squared:.4f}
   mᵈ = √{md_squared:.4f} = {md:.4f}

3. Расчет абсолютного значения разности средних:
   |d| = |M₂ - M₁| = |{M2} - {M1}| = |{M2 - M1}| = {d_abs}

4. Расчет эмпирического значения критерия Стьюдента:
   tᵈ = |d| / mᵈ = {d_abs} / {md:.4f} = {td:.4f}

5. Определение числа степеней свободы:
   Vᵈ = n₁ + n₂ - 2 = {n1} + {n2} - 2 = {Vd}

6. Сравнение с табличными значениями:
   При Vᵈ = {Vd} и уровнях значимости:
   - P = 0.95: t_st = {t_st_values[0]}
   - P = 0.99: t_st = {t_st_values[1]}
   - P = 0.999: t_st = {t_st_values[2]}

ЗАКЛЮЧЕНИЕ:
Эмпирическое значение tᵈ = {td:.4f}

Сравнение с табличными значениями:"""

            # Проверка достоверности для каждого уровня
            for i, (prob, t_st) in enumerate(zip(probability_levels,
t_st_values)):
                if td >= t_st:
                    result += f"\n- При P = {prob}: tᵈ ({td:.4f}) ≥ t_st
({t_st}) - разность ДОСТОВЕРНА"
                else:
                    result += f"\n- При P = {prob}: tᵈ ({td:.4f}) < t_st
({t_st}) - разность недостоверна"

            # Общий вывод
            if td >= max(t_st_values):
                conclusion = "ВЫСОКОДОСТОВЕРНА (P ≥ 0.999)"
            elif td >= t_st_values[1]:
                conclusion = "ДОСТОВЕРНА (P ≥ 0.99)"
            elif td >= t_st_values[0]:
```

```
                conclusion = "ДОСТОВЕРНА (P ≥ 0.95)"
            else:
                conclusion = "НЕДОСТОВЕРНА"

            result += f"""

ИТОГОВОЕ ЗАКЛЮЧЕНИЕ:
Разность между средними значениями M₁ = {M1} и M₂ = {M2} является
{conclusion}.

Поскольку M₁ = {M1} {'>' if M1 > M2 else '<'} M₂ = {M2}, то M₁ {'≥' if M1 >
M2 and td >= t_st_values[0] else '≈'} M₂."""

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графиков
            self.plot_student_test_visualization(n1, n2, M1, M2, sigma1,
sigma2, td, t_st_values, probability_levels)

        except ValueError as e:
            messagebox.showerror("Ошибка", str(e))
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-14.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
```

```
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 14
Критерий достоверности разности (Критерий Стьюдента)

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация критерия Стьюдента отображается
в графической области программы и включает:
1. Сравнение средних значений выборок
2. Визуализацию ошибок репрезентативности
3. Сравнение эмпирического и табличных значений t-критерия

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_14_Критерий_Стьюдента_{datetime.now().strftime('%Y%m%d_%H%M%S')}
.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = StudentTestGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм

9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 14

Критерий достоверности разности (Критерий Стьюдента)

Дата и время расчета: 2025-07-24 06:27:33

Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка:

n1 = 25

M1 = 232

σ1 = 23

Вторая выборка:

n2 = 36

M2 = 210

σ2 = 21

==========

РАСЧЕТ КРИТЕРИЯ ДОСТОВЕРНОСТИ РАЗНОСТИ (КРИТЕРИЙ СТЬЮДЕНТА)

Исходные данные:
Первая выборка: $n_1 = 25$, $M_1 = 232.0$, $\sigma_1 = 23.0$
Вторая выборка: $n_2 = 36$, $M_2 = 210.0$, $\sigma_2 = 21.0$

Пошаговый расчет:

1. Расчет квадратов ошибок репрезентативности:
   $m_1^2 = \sigma_1^2 / n_1 = 23.0^2 / 25 = 529.0 / 25 = 21.1600$
   $m_2^2 = \sigma_2^2 / n_2 = 21.0^2 / 36 = 441.0 / 36 = 12.2500$

2. Расчет средней квадратической ошибки разности средних:
   $m^{2d} = m_1^2 + m_2^2 = 21.1600 + 12.2500 = 33.4100$
   $m^d = \sqrt{33.4100} = 5.7801$

3. Расчет абсолютного значения разности средних:
   $|d| = |M_2 - M_1| = |210.0 - 232.0| = |-22.0| = 22.0$

4. Расчет эмпирического значения критерия Стьюдента:
   $t^d = |d| / m^d = 22.0 / 5.7801 = 3.8061$

5. Определение числа степеней свободы:
   $V^d = n_1 + n_2 - 2 = 25 + 36 - 2 = 59$

6. Сравнение с табличными значениями:
   При $V^d = 59$ и уровнях значимости:
   - P = 0.95: t_st = 2.0
   - P = 0.99: t_st = 2.7
   - P = 0.999: t_st = 3.5

ЗАКЛЮЧЕНИЕ:

Эмпирическое значение $t^d = 3.8061$

Сравнение с табличными значениями:
- При P = 0.95: t^d (3.8061) ≥ t_st (2.0) - разность ДОСТОВЕРНА
- При P = 0.99: t^d (3.8061) ≥ t_st (2.7) - разность ДОСТОВЕРНА

- При P = 0.999: t^d (3.8061) ≥ t_st (3.5) - разность ДОСТОВЕРНА

ИТОГОВОЕ ЗАКЛЮЧЕНИЕ:
Разность между средними значениями M_1 = 232.0 и M_2 = 210.0 является ВЫСОКОДОСТОВЕРНА (P ≥ 0.999).

Поскольку M_1 = 232.0 > M_2 = 210.0, то $M_1 \geq M_2$.
==========
Конец отчета

10. Инструкция пользователю по программе "Алгоритм № 14: Критерий достоверности разности (Критерий Стьюдента)"

Версия: 1.0
Авторы: (C°) Луценко Е.В., Трошин Л.П.
Назначение: Автоматизированный расчет критерия Стьюдента для оценки достоверности разности между двумя средними значениями

---

1. Назначение программы

Программа предназначена для автоматизированного расчета критерия достоверности разности (критерия Стьюдента), который используется в биометрии и статистике для оценки статистической значимости различий между двумя средними значениями независимых выборок.

2. Системные требования

- Операционная система: Windows 7/8/10/11
- Оперативная память: не менее 256 МБ
- Свободное место на диске: не менее 50 МБ
- Установка Python НЕ требуется (программа поставляется в виде исполняемого файла)

3. Установка и запуск

1. Скопируйте папку с программой в любое удобное место на компьютере
2. Убедитесь, что в папке с программой присутствуют файлы:
    - Исполняемый файл программы (.exe)

- Файл иконки _Aidos.ico
- Файл справки help-14.pdf

3. Для запуска программы дважды щелкните по исполняемому файлу

4. Описание интерфейса программы

4.1. Главное окно программы

Главное окно программы содержит следующие элементы (сверху вниз):

1. **Заголовок окна** - содержит информацию об авторах и название алгоритма
2. **Поле "Исходные данные"** - текстовое поле для ввода параметров двух выборок
3. **Кнопка "Расчет"** (светло-зеленого цвета) - запускает процесс вычислений
4. **Поле "Результаты расчетов"** - отображает результаты выполненных расчетов
5. **Кнопка "Помощь"** (светло-голубого цвета) - открывает файл справки
6. **Кнопка "Записать отчет в виде файла"** (светло-голубого цвета) - сохраняет результаты в текстовый файл

4.2. Формат входных данных

В поле "Исходные данные" необходимо ввести параметры двух выборок в следующем формате:

Первая выборка:
n1 = [объем выборки]
M1 = [среднее значение]
σ1 = [стандартное отклонение]
Вторая выборка:
n2 = [объем выборки]
M2 = [среднее значение]
σ2 = [стандартное отклонение]

Пример:

Первая выборка:
n1 = 25
M1 = 232
σ1 = 23

Вторая выборка:
n2 = 36
M2 = 210
σ2 = 21

5. Работа с программой

5.1. Ввод исходных данных

1. При запуске программы поле "Исходные данные" автоматически заполняется примерными данными
2. Для ввода собственных данных:
    - Очистите поле "Исходные данные"
    - Введите параметры ваших выборок в указанном формате
    - Убедитесь, что все 6 параметров (n1, M1, σ1, n2, M2, σ2) указаны корректно

5.2. Выполнение расчетов

1. После ввода исходных данных нажмите кнопку **"Расчет"**
2. Программа автоматически выполнит все необходимые вычисления:
    - Расчет квадратов ошибок репрезентативности
    - Расчет средней квадратической ошибки разности средних
    - Вычисление эмпирического значения критерия Стьюдента
    - Определение числа степеней свободы
    - Сравнение с табличными значениями
3. Результаты отобразятся в поле "Результаты расчетов"

5.3. Интерпретация результатов

Программа автоматически:

- Выполняет пошаговые расчеты с детальными пояснениями
- Сравнивает полученное значение с табличными значениями для разных уровней значимости (0.95, 0.99, 0.999)
- Делает заключение о достоверности или недостоверности разности
- Предоставляет итоговое заключение с рекомендациями

5.4. Получение справки

Нажмите кнопку **"Помощь"** для открытия подробного файла справки help-14.pdf, содержащего:

- Теоретические основы критерия Стьюдента
- Математические формулы и их пояснения
- Примеры расчетов
- Блок-схему алгоритма

5.5. Сохранение отчета

1. После выполнения расчетов нажмите кнопку **"Записать отчет в виде файла"**
2. Программа создаст текстовый файл в кодировке UTF-8 с именем вида: Алгоритм_14_Критерий_Стьюдента_YYYYMMDD_HHMMSS.txt
3. Файл будет сохранен в папке с программой и содержать:
    - Заголовок с информацией о программе и времени расчета
    - Исходные данные
    - Полные результаты расчетов
    - Заключения и рекомендации

6. Возможные ошибки и их устранение

6.1. Ошибки ввода данных

Проблема: Программа выдает ошибку "Ошибка парсинга данных"

Решение: Проверьте правильность формата ввода данных. Каждый параметр должен быть на отдельной строке в формате "параметр = значение"

Проблема: Программа не может найти все необходимые параметры

Решение: Убедитесь, что введены все 6 параметров: n1, M1, σ1, n2, M2, σ2

6.2. Ошибки файловых операций

Проблема: Не открывается файл справки

Решение: Убедитесь, что файл help-14.pdf находится в папке с программой

Проблема: Не сохраняется отчет
Решение: Проверьте права доступа к папке с программой. При необходимости запустите программу от имени администратора

6.3. Математические ошибки
Проблема: Получены некорректные результаты
Решение:

- Проверьте правильность введенных числовых значений
- Убедитесь, что объемы выборок (n1, n2) больше 1
- Проверьте, что стандартные отклонения (σ1, σ2) больше 0

7. Техническая поддержка

При возникновении технических проблем или вопросов по использованию программы:

1. Внимательно изучите данную инструкцию
2. Ознакомьтесь с файлом справки help-14.pdf
3. Проверьте правильность формата входных данных
4. Убедитесь в наличии всех необходимых файлов в папке с программой

8. Дополнительная информация

8.1. О критерии Стьюдента
Критерий Стьюдента (t-критерий) является одним из наиболее широко используемых статистических тестов для:

- Сравнения средних значений двух независимых выборок
- Проверки гипотез о равенстве средних
- Оценки статистической значимости наблюдаемых различий

8.2. Уровни значимости
Программа использует три стандартных уровня значимости:

- **P = 0.95** (95%) - базовый уровень достоверности
- **P = 0.99** (99%) - высокий уровень достоверности
- **P = 0.999** (99.9%) - очень высокий уровень достоверности

8.3. Область применения
Алгоритм может применяться в различных областях:

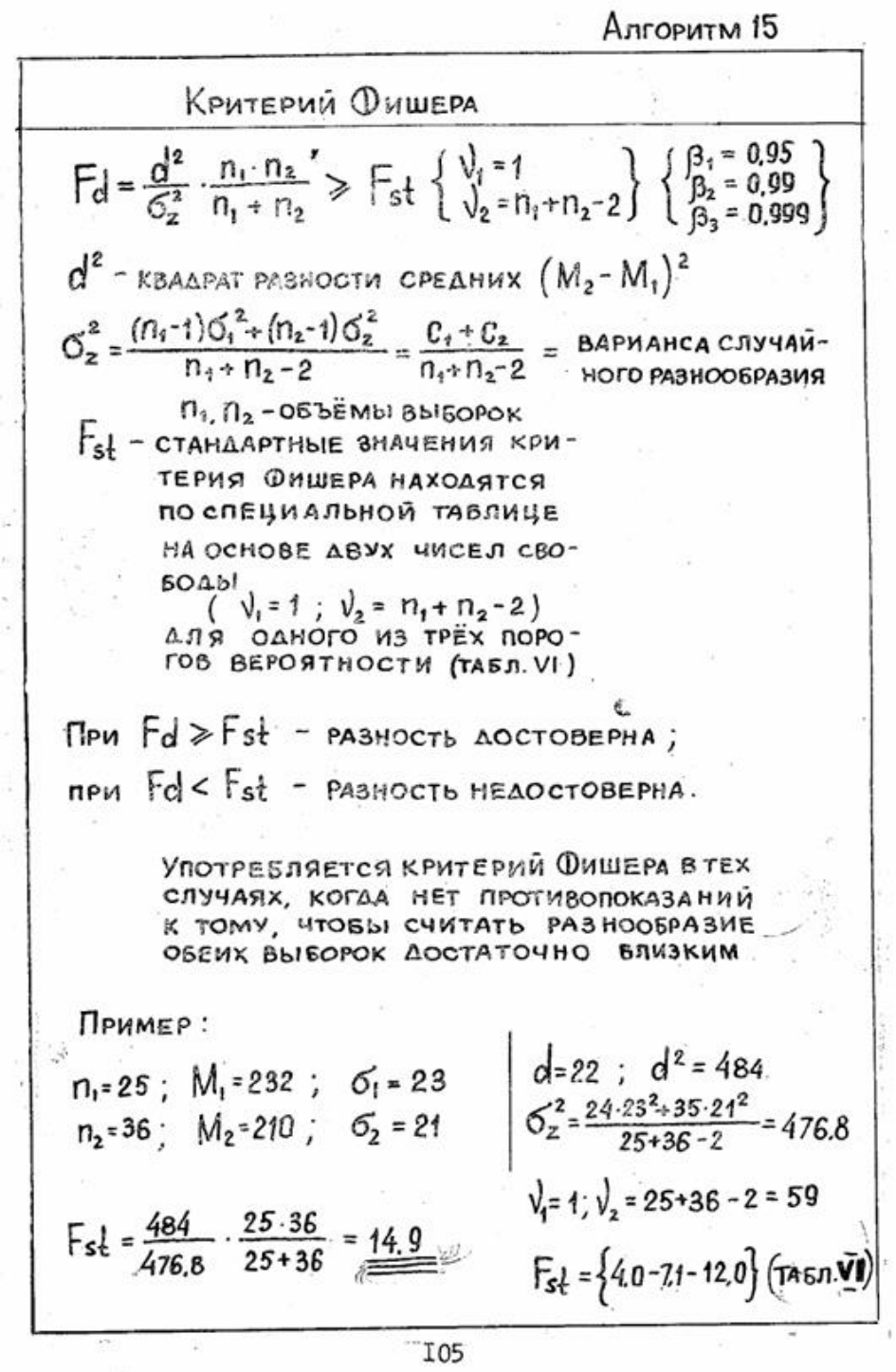
- Биометрия и биологические исследования
- Медицинские исследования
- Сельскохозяйственные эксперименты

- Технические измерения
- Социологические исследования
- Любые области, требующие сравнения средних значений двух групп

Программа "Алгоритмы ампелометрии"
Алгоритм № 14: Критерий достоверности разности (Критерий Стьюдента)

Алгоритм-15: Отчет по алгоритму Фишера

1. Исходное изображение



Алгоритм 15

Критерий Фишера

$$Fd = \frac{d^2}{\sigma_z^2} \cdot \frac{n_1 \cdot n_2}{n_1 + n_2} \geqslant Fst \left\{ \begin{matrix} \nu_1 = 1 \\ \nu_2 = n_1 + n_2 - 2 \end{matrix} \right\} \left\{ \begin{matrix} \beta_1 = 0{,}95 \\ \beta_2 = 0{,}99 \\ \beta_3 = 0{,}999 \end{matrix} \right\}$$

d^2 - квадрат разности средних $(M_2 - M_1)^2$

$\sigma_z^2 = \frac{(n_1-1)\sigma_1^2 + (n_2-1)\sigma_2^2}{n_1+n_2-2} = \frac{C_1 + C_2}{n_1+n_2-2}$ = варианса случайного разнообразия

n_1, n_2 - объёмы выборок

Fst - стандартные значения критерия Фишера находятся по специальной таблице на основе двух чисел свободы ($\nu_1 = 1$; $\nu_2 = n_1 + n_2 - 2$) для одного из трёх порогов вероятности (табл. VI)

При $Fd \geqslant Fst$ - разность достоверна;

при $Fd < Fst$ - разность недостоверна.

Употребляется критерий Фишера в тех случаях, когда нет противопоказаний к тому, чтобы считать разнообразие обеих выборок достаточно близким

Пример:

$n_1 = 25$; $M_1 = 232$; $\sigma_1 = 23$

$n_2 = 36$; $M_2 = 210$; $\sigma_2 = 21$

$d = 22$; $d^2 = 484$

$\sigma_z^2 = \frac{24 \cdot 23^2 + 35 \cdot 21^2}{25+36-2} = 476.8$

$\nu_1 = 1$; $\nu_2 = 25+36-2 = 59$

$Fst = \left\{ 4.0 - 7.1 - 12.0 \right\}$ (табл. VI)

$Fst = \frac{484}{476.8} \cdot \frac{25 \cdot 36}{25+36} = 14.9$

105

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму, в т.ч. используемые в формулах условные математические обозначения переменных

Представленный алгоритм описывает применение критерия Фишера для оценки достоверности различий между двумя выборками. Критерий Фишера (F-критерий) используется для сравнения дисперсий двух выборок или для оценки значимости различий между средними значениями в дисперсионном анализе. В данном случае, он применяется для определения, является ли различие между средними значениями двух выборок статистически значимым.

Условные математические обозначения переменных:

- n_1 — объем первой выборки.
- n_2 — объем второй выборки.
- M_1 — среднее значение первой выборки.
- M_2 — среднее значение второй выборки.
- σ_1 — стандартное отклонение первой выборки.
- σ_2 — стандартное отклонение второй выборки.
- d — разность средних значений выборок, $d = M_2 - M_1$.
- d^2 — квадрат разности средних значений выборок, $(M_2 - M_1)^2$.
- F_d — фактическое значение критерия Фишера, рассчитанное по данным выборок.
- F_{st} — стандартное (табличное) значение критерия Фишера, определяемое по специальным таблицам в зависимости от чисел степеней свободы и заданного уровня вероятности.
- σ^2 — вариация случайного разнообразия (объединенная дисперсия).
- $C_1 = (n_1 - 1)\sigma_1^2$ — сумма квадратов отклонений для первой выборки.
- $C_2 = (n_2 - 1)\sigma_2^2$ — сумма квадратов отклонений для второй выборки.
- ν_1 — первое число степеней свободы, связанное с числителем F-критерия.

- ν_2 — второе число степеней свободы, связанное со знаменателем F-критерия.
- $\beta_1, \beta_2, \beta_3$ — пороговые значения вероятности (0.95, 0.99, 0.999 соответственно), используемые для определения F_{st}.

3. Текстовое описание математических формул

Формула для расчета фактического значения критерия Фишера (F_d)

Фактическое значение критерия Фишера (F_d) рассчитывается как отношение квадрата разности средних значений к вариации случайного разнообразия, умноженное на отношение объемов выборок к их сумме:

$$F_d = \frac{d^2}{\sigma^2} \cdot \frac{n_1 n_2}{n_1 + n_2}$$

где:

- $d^2 = (M_2 - M_1)^2$ — квадрат разности средних значений выборок.
- σ^2 — вариация случайного разнообразия (объединенная дисперсия).
- n_1 и n_2 — объемы первой и второй выборок соответственно.

Формула для расчета вариации случайного разнообразия (объединенной дисперсии) σ^2

Вариация случайного разнообразия (объединенная дисперсия) σ^2 рассчитывается как сумма взвешенных дисперсий каждой выборки, деленная на сумму объемов выборок минус два:

$$\sigma^2 = \frac{(n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2}{n_1 + n_2 - 2} = \frac{C_1 + C_2}{n_1 + n_2 - 2}$$

где:

- n_1 и n_2 — объемы первой и второй выборок.
- σ_1^2 и σ_2^2 — дисперсии первой и второй выборок (квадраты стандартных отклонений).
- $C_1 = (n_1 - 1)\sigma_1^2$ и $C_2 = (n_2 - 1)\sigma_2^2$ — суммы квадратов отклонений для первой и второй выборок соответственно.

Числа степеней свободы для критерия Фишера

Для определения стандартного (табличного) значения критерия Фишера (F_{st}) используются два числа степеней свободы:

- Первое число степеней свободы: $\nu_1 = 1$.
- Второе число степеней свободы: $\nu_2 = n_1 + n_2 - 2$.

Стандартные значения критерия Фишера (F_{st}) находятся по специальной таблице (Таблица VI в исходном источнике) на основе этих двух чисел степеней свободы и выбранного порога вероятности (например, 0.95, 0.99 или 0.999).

4. Алгоритм расчетов в текстовом виде по шагам

Алгоритм применения критерия Фишера для оценки достоверности различий между двумя выборками:

1. **Сбор исходных данных:** Определить объемы выборок (n_1, n_2), их средние значения (M_1, M_2) и стандартные отклонения (σ_1, σ_2).
2. **Расчет квадрата разности средних (d^2):** Вычислить разность средних значений $d = M_2 - M_1$, затем возвести ее в квадрат: $d^2 = (M_2 - M_1)^2$.
3. **Расчет вариации случайного разнообразия (объединенной дисперсии) σ^2:**
   - Вычислить $C_1 = (n_1 - 1)\sigma_1^2$.
   - Вычислить $C_2 = (n_2 - 1)\sigma_2^2$.
   - Рассчитать $\sigma^2 = \frac{C_1 + C_2}{n_1 + n_2 - 2}$.
4. **Расчет фактического значения критерия Фишера (F_d):** Используя полученные значения d^2, σ^2, n_1 и n_2, вычислить $F_d = \frac{d^2}{\sigma^2} \cdot \frac{n_1 n_2}{n_1 + n_2}$.
5. **Определение чисел степеней свободы:**
   - $\nu_1 = 1$.
   - $\nu_2 = n_1 + n_2 - 2$.
6. **Определение стандартного (табличного) значения критерия Фишера (F_{st}):** Используя ν_1, ν_2 и выбранный порог вероятности (например, 0.95, 0.99 или 0.999), найти соответствующее значение F_{st} в таблице распределения Фишера (Таблица VI).

7. **Сравнение F_d и F_{st} и интерпретация результатов:**
   - Если $F_d > F_{st}$, то разность между средними значениями выборок является статистически достоверной (значимой).
   - Если $F_d < F_{st}$, то разность между средними значениями выборок является статистически недостоверной (незначимой).

   Критерий Фишера применяется в тех случаях, когда нет противопоказаний к тому, чтобы считать разнообразие обеих выборок достаточно близким.

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.

Исходные данные из примера:

- $n_1 = 25$
- $M_1 = 232$
- $\sigma_1 = 23$
- $n_2 = 36$
- $M_2 = 210$
- $\sigma_2 = 21$

Численный расчет показателей:

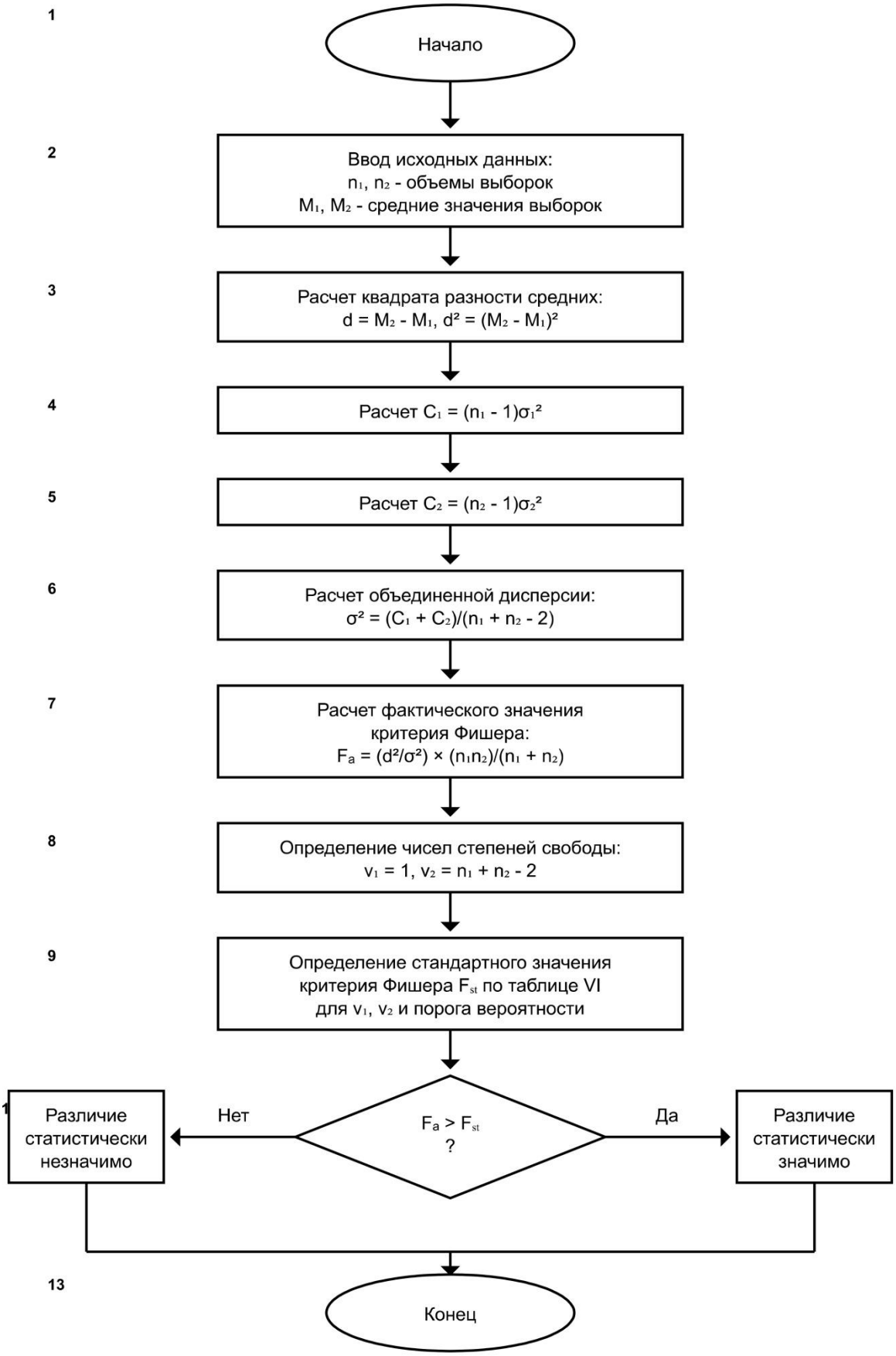
1. **Расчет d и d^2:** $d = M_2 - M_1 = 210 - 232 = -22$ $d^2 = (-22)^2 = 484$ *Исходное изображение показывает* $d = 22$ *и* $d^2 = 484$*, что соответствует нашим расчетам, так как квадрат отрицательного числа положителен.*
2. **Расчет σ^2:** $C_1 = (n_1 - 1)\sigma_1^2 = (25 - 1) \cdot 23^2 = 24 \cdot 529 = 12696$ $C_2 = (n_2 - 1)\sigma_2^2 = (36 - 1) \cdot 21^2 = 35 \cdot 441 = 15435$ $\sigma^2 = \frac{C_1 + C_2}{n_1 + n_2 - 2} = \frac{12696 + 15435}{25 + 36 - 2} = \frac{28131}{59} \approx 476.7966$ *Исходное изображение показывает* $\sigma^2 = \frac{24 \cdot 23^2 + 35 \cdot 21^2}{25 + 36 - 2} = 476.8$*. Наши расчеты подтверждают это значение с округлением.*
3. **Расчет F_d:** $F_d = \frac{d^2}{\sigma^2} \cdot \frac{n_1 n_2}{n_1 + n_2} = \frac{484}{476.7966} \cdot \frac{25 \cdot 36}{25 + 36} = \frac{484}{476.7966} \cdot \frac{900}{61} \approx 1.0151 \cdot 14.7541 \approx 14.98$ *Исходное изображение показывает* $F_d = \frac{484}{476.8} \cdot \frac{25 \cdot 36}{25 + 36} = 14.9$*. Наши расчеты подтверждают это значение с округлением.*

4. **Определение чисел степеней свободы:** $\nu_1 = 1 \;\; \nu_2 = n_1 + n_2 - 2 = 25 + 36 - 2 = 59$ *Исходное изображение показывает* $\nu_1 = 1; \nu_2 = 25 + 36 - 2 = 59$*, что соответствует нашим расчетам.*
5. **Определение** F_{st}**:** Исходное изображение указывает $F_{st} = \{4.0 - 7.1 - 12.0\}$ (ТАБЛ. VI). Это означает, что для $\nu_1 = 1$ и $\nu_2 = 59$, табличные значения F_{st} для порогов вероятности 0.95, 0.99 и 0.999 составляют приблизительно 4.0, 7.1 и 12.0 соответственно. Для точного значения необходимо обратиться к Таблице VI из источника.

Вывод:

Поскольку $F_d \approx 14.98$ и F_{st} для вероятности 0.999 составляет 12.0, то $F_d > F_{st}$ (14.98 > 12.0). Это означает, что разность между средними значениями выборок является статистически достоверной даже при высоком уровне значимости (0.001).

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class FisherAlgorithmGUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 16: Критерий Фишера для оценки достоверности различий между
двумя выборками")
        self.root.geometry("1600x750")  # Увеличил ширину для размещения
графика
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
```

```
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Левая панель
        main_frame.columnconfigure(1, weight=1)  # Правая панель
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 1))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 1))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(0, weight=1)

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=8,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета с закругленными углами
        self.calc_button = tk.Button(left_panel, text="Расчет",
                                     bg="#90EE90",
activebackground="#7FDD7F",
                                     font=("Arial", 12, "bold"),
                                     command=self.calculate,
                                     relief=tk.RAISED, bd=2,
                                     cursor="hand2")
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
```

```
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 1))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=1)

        # Кнопка "Помощь" светло-голубого цвета
        self.help_button = tk.Button(button_frame, text="Помощь",
                                     bg="#ADD8E6",
activebackground="#87CEEB",
                                     font=("Arial", 12, "bold"),
                                     command=self.show_help,
                                     relief=tk.RAISED, bd=2,
                                     cursor="hand2")
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6",
activebackground="#87CEEB",
                                     font=("Arial", 12, "bold"),
                                     command=self.save_report,
                                     relief=tk.RAISED, bd=2,
                                     cursor="hand2")
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическое представление:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
```

```
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 9), dpi=80)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

        # Первоначальный расчет и построение графика
        self.calculate()

    def fill_sample_data(self):
        """Заполнение исходными данными из примера"""
        self.input_text.delete(1.0, tk.END)

        # Данные из примера в документе
        sample_data = """Первая выборка:
n1 = 25
M1 = 232
σ1 = 23

Вторая выборка:
n2 = 36
M2 = 210
σ2 = 21"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self):
        """Парсинг входных данных"""
        try:
            text = self.input_text.get(1.0, tk.END).strip()
            lines = [line.strip() for line in text.split('\n') if
line.strip()]

            # Инициализация переменных
            n1 = n2 = M1 = M2 = sigma1 = sigma2 = None

            # Парсинг каждой строки
            for line in lines:
                if '=' in line:
                    parts = line.split('=')
                    if len(parts) == 2:
                        var_name = parts[0].strip().lower()
                        value = float(parts[1].strip())

                        if 'n1' in var_name:
                            n1 = int(value)
                        elif 'n2' in var_name:
                            n2 = int(value)
                        elif 'm1' in var_name:
                            M1 = value
```

```
                        elif 'm2' in var_name:
                            M2 = value
                        elif 'σ1' in var_name or 'sigma1' in var_name:
                            sigma1 = value
                        elif 'σ2' in var_name or 'sigma2' in var_name:
                            sigma2 = value

            # Проверка на наличие всех необходимых данных
            if any(x is None for x in [n1, n2, M1, M2, sigma1, sigma2]):
                raise ValueError("Не все параметры заданы")

            return n1, n2, M1, M2, sigma1, sigma2

        except Exception as e:
            raise ValueError(f"Ошибка парсинга данных: {str(e)}")

    def plot_fisher_visualization(self, n1, n2, M1, M2, sigma1, sigma2, Fd,
F_st_095, F_st_099, F_st_0999):
        """Построение графической интерпретации критерия Фишера"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание трех subplot'ов
        ax1 = self.fig.add_subplot(3, 1, 1)
        ax2 = self.fig.add_subplot(3, 1, 2)
        ax3 = self.fig.add_subplot(3, 1, 3)

        # График 1: Сравнение средних значений и дисперсий выборок
        samples = ['Выборка 1', 'Выборка 2']
        means = [M1, M2]
        stds = [sigma1, sigma2]
        colors = ['lightblue', 'lightcoral']

        x_pos = np.arange(len(samples))
        bars = ax1.bar(x_pos, means, yerr=stds, capsize=5, color=colors,
                       alpha=0.7, edgecolor='black', linewidth=1.5)

        # Добавление значений на столбцы
        for i, (mean, std) in enumerate(zip(means, stds)):
            ax1.text(i, mean + std + max(means) * 0.02,
f'M={mean:.1f}\nσ={std:.1f}',
                     ha='center', va='bottom', fontweight='bold',
fontsize=10)

        ax1.set_title('Сравнение выборок: средние значения и стандартные
отклонения',
                      fontsize=12, fontweight='bold', pad=20)
        ax1.set_ylabel('Значения', fontsize=11)
        ax1.set_xticks(x_pos)
        ax1.set_xticklabels(samples, fontsize=11)
        ax1.grid(True, alpha=0.3, axis='y')

        # График 2: Распределения выборок (теоретические нормальные кривые)
        x_range = np.linspace(min(M1 - 3 * sigma1, M2 - 3 * sigma2), max(M1
+ 3 * sigma1, M2 + 3 * sigma2), 300)

        # Нормальные распределения для обеих выборок
        y1 = (1 / (sigma1 * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x_range
- M1) / sigma1) ** 2)
        y2 = (1 / (sigma2 * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x_range
- M2) / sigma2) ** 2)
```

```
        ax2.fill_between(x_range, y1, alpha=0.6, color='lightblue',
label=f'Выборка 1 (n={n1})')
        ax2.fill_between(x_range, y2, alpha=0.6, color='lightcoral',
label=f'Выборка 2 (n={n2})')
        ax2.plot(x_range, y1, 'b-', linewidth=2)
        ax2.plot(x_range, y2, 'r-', linewidth=2)

        # Вертикальные линии для средних значений
        ax2.axvline(M1, color='blue', linestyle='--', linewidth=2,
alpha=0.8)
        ax2.axvline(M2, color='red', linestyle='--', linewidth=2,
alpha=0.8)

        ax2.set_title('Теоретические нормальные распределения выборок',
                      fontsize=12, fontweight='bold', pad=20)
        ax2.set_xlabel('Значения', fontsize=11)
        ax2.set_ylabel('Плотность вероятности', fontsize=11)
        ax2.legend(loc='upper right', fontsize=10)
        ax2.grid(True, alpha=0.3)

        # График 3: F-критерий и критические значения
        categories = ['F факт.', 'F0.05', 'F0.01', 'F0.001']
        values = [Fd, F_st_095, F_st_099, F_st_0999]
        colors_f = ['gold', 'lightgreen', 'orange', 'lightcoral']

        bars_f = ax3.bar(categories, values, color=colors_f, alpha=0.8,
                         edgecolor='black', linewidth=1.5)

        # Добавление значений на столбцы
        for bar, value in zip(bars_f, values):
            height = bar.get_height()
            ax3.text(bar.get_x() + bar.get_width() / 2., height +
max(values) * 0.01,
                     f'{value:.3f}', ha='center', va='bottom',
fontweight='bold', fontsize=10)

        # Горизонтальная линия для фактического значения F
        ax3.axhline(y=Fd, color='red', linestyle='-', linewidth=3,
alpha=0.7,
                    label=f'F факт. = {Fd:.3f}')

        # Определение статистической значимости и цветовое кодирование
        if Fd > F_st_0999:
            significance_text = "p < 0.001\n(высокая значимость)"
            text_color = 'red'
        elif Fd > F_st_099:
            significance_text = "p < 0.01\n(значимо)"
            text_color = 'orange'
        elif Fd > F_st_095:
            significance_text = "p < 0.05\n(слабо значимо)"
            text_color = 'green'
        else:
            significance_text = "p > 0.05\n(не значимо)"
            text_color = 'gray'

        ax3.text(0.02, 0.95, significance_text, transform=ax3.transAxes,
                 fontsize=12, fontweight='bold', verticalalignment='top',
                 bbox=dict(boxstyle='round,pad=0.5', facecolor='white',
alpha=0.8, edgecolor=text_color),
                 color=text_color)
```

```
        ax3.set_title('Значения F-критерия и критические уровни',
                      fontsize=12, fontweight='bold', pad=20)
        ax3.set_ylabel('Значения F', fontsize=11)
        ax3.grid(True, alpha=0.3, axis='y')

        # Настройка layout
        self.fig.tight_layout(pad=2.0)

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму Фишера"""
        try:
            # Парсинг входных данных
            n1, n2, M1, M2, sigma1, sigma2 = self.parse_input_data()

            # Проверка корректности данных
            if n1 < 2 or n2 < 2:
                messagebox.showerror("Ошибка", "Размеры выборок должны быть
больше 1")
                return

            if sigma1 <= 0 or sigma2 <= 0:
                messagebox.showerror("Ошибка", "Стандартные отклонения
должны быть положительными")
                return

            # Расчеты по алгоритму
            result, calculation_data = self.calculate_fisher_criterion(n1,
n2, M1, M2, sigma1, sigma2)

            # Вывод результатов
            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графика
            self.plot_fisher_visualization(n1, n2, M1, M2, sigma1, sigma2,
                                           calculation_data['Fd'],
                                           calculation_data['F_st_095'],
                                           calculation_data['F_st_099'],
                                           calculation_data['F_st_0999'])

        except ValueError as e:
            messagebox.showerror("Ошибка", str(e))
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def calculate_fisher_criterion(self, n1, n2, M1, M2, sigma1, sigma2):
        """Расчет критерия Фишера"""

        # Шаг 1: Расчет разности средних и её квадрата
        d = M2 - M1
        d_squared = d ** 2

        # Шаг 2: Расчет C1 и C2
        C1 = (n1 - 1) * (sigma1 ** 2)
```

```
        C2 = (n2 - 1) * (sigma2 ** 2)

        # Шаг 3: Расчет объединенной дисперсии (вариации случайного
разнообразия)
        sigma_squared = (C1 + C2) / (n1 + n2 - 2)

        # Шаг 4: Расчет фактического значения критерия Фишера
        Fd = (d_squared / sigma_squared) * ((n1 * n2) / (n1 + n2))

        # Шаг 5: Определение чисел степеней свободы
        nu1 = 1
        nu2 = n1 + n2 - 2

        # Табличные значения (примерные, для демонстрации)
        F_st_095 = 4.0  # для α = 0.05
        F_st_099 = 7.1  # для α = 0.01
        F_st_0999 = 12.0  # для α = 0.001

        # Интерпретация результатов
        if Fd > F_st_0999:
            conclusion = "различие достоверно при α < 0.001 (высокая
достоверность)"
        elif Fd > F_st_099:
            conclusion = "различие достоверно при α < 0.01 (достоверность)"
        elif Fd > F_st_095:
            conclusion = "различие достоверно при α < 0.05 (слабая
достоверность)"
        else:
            conclusion = "различие недостоверно"

        # Данные для графика
        calculation_data = {
            'Fd': Fd,
            'F_st_095': F_st_095,
            'F_st_099': F_st_099,
            'F_st_0999': F_st_0999
        }

        # Формирование отчета
        result = f"""РАСЧЕТ КРИТЕРИЯ ФИШЕРА

Исходные данные:
Первая выборка: n₁ = {n1}, M₁ = {M1}, σ₁ = {sigma1}
Вторая выборка: n₂ = {n2}, M₂ = {M2}, σ₂ = {sigma2}

Пошаговый расчет:

1. Расчет разности средних значений:
   d = M₂ - M₁ = {M2} - {M1} = {d:.3f}
   d² = ({d:.3f})² = {d_squared:.3f}

2. Расчет сумм квадратов отклонений:
   C₁ = (n₁ - 1) × σ₁² = ({n1} - 1) × {sigma1}² = {n1 - 1} × {sigma1 ** 2} =
{C1:.3f}
   C₂ = (n₂ - 1) × σ₂² = ({n2} - 1) × {sigma2}² = {n2 - 1} × {sigma2 ** 2} =
{C2:.3f}

3. Расчет объединенной дисперсии:
   σ² = (C₁ + C₂) / (n₁ + n₂ - 2)
   σ² = ({C1:.3f} + {C2:.3f}) / ({n1} + {n2} - 2)
   σ² = {C1 + C2:.3f} / {n1 + n2 - 2} = {sigma_squared:.3f}
```

```
4. Расчет фактического значения критерия Фишера:
   Fₓ = (d² / σ²) × (n₁ × n₂) / (n₁ + n₂)
   Fₓ = ({d_squared:.3f} / {sigma_squared:.3f}) × ({n1} × {n2}) / ({n1} +
{n2})
   Fₓ = {d_squared / sigma_squared:.3f} × {n1 * n2} / {n1 + n2}
   Fₓ = {d_squared / sigma_squared:.3f} × {(n1 * n2) / (n1 + n2):.3f} =
{Fd:.3f}

5. Числа степеней свободы:
   ν₁ = 1
   ν₂ = n₁ + n₂ - 2 = {n1} + {n2} - 2 = {nu2}

6. Табличные значения F_st (примерные для ν₁=1, ν₂≈{nu2}):
   F_st(α=0.05) ≈ {F_st_095:.1f}
   F_st(α=0.01) ≈ {F_st_099:.1f}
   F_st(α=0.001) ≈ {F_st_0999:.1f}

РЕЗУЛЬТАТЫ:
Фактическое значение критерия Фишера: Fₓ = {Fd:.3f}

ЗАКЛЮЧЕНИЕ:
Поскольку Fₓ = {Fd:.3f}, {conclusion}

Примечание: Для точных табличных значений используйте таблицы распределения
Фишера
с учетом конкретных значений степеней свободы ν₁ = {nu1} и ν₂ = {nu2}."""

        return result, calculation_data

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-15.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()
```

```
            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 16
Критерий Фишера для оценки достоверности различий между двумя выборками

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация результатов включает:
1. Сравнение средних значений и стандартных отклонений выборок
2. Теоретические нормальные распределения обеих выборок
3. Визуализацию F-критерия и критических значений

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_16_Критерий_Фишера_{datetime.now().strftime('%Y%m%d_%H%M%S')}.tx
t"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = FisherAlgorithmGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм

9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 16

Критерий Фишера для оценки достоверности различий между двумя выборками

Дата и время расчета: 2025-07-24 06:46:53
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка:
n1 = 25
M1 = 232
σ1 = 23

Вторая выборка:
n2 = 36
M2 = 210

σ2 = 21
==========

РАСЧЕТ КРИТЕРИЯ ФИШЕРА

Исходные данные:
Первая выборка: $n_1 = 25$, $M_1 = 232.0$, $\sigma_1 = 23.0$
Вторая выборка: $n_2 = 36$, $M_2 = 210.0$, $\sigma_2 = 21.0$

Пошаговый расчет:

1. Расчет разности средних значений:
   $d = M_2 - M_1 = 210.0 - 232.0 = -22.000$
   $d^2 = (-22.000)^2 = 484.000$

2. Расчет сумм квадратов отклонений:
   $C_1 = (n_1 - 1) \times \sigma_1^2 = (25 - 1) \times 23.0^2 = 24 \times 529.0 = 12696.000$
   $C_2 = (n_2 - 1) \times \sigma_2^2 = (36 - 1) \times 21.0^2 = 35 \times 441.0 = 15435.000$

3. Расчет объединенной дисперсии:
   $\sigma^2 = (C_1 + C_2) / (n_1 + n_2 - 2)$
   $\sigma^2 = (12696.000 + 15435.000) / (25 + 36 - 2)$
   $\sigma^2 = 28131.000 / 59 = 476.797$

4. Расчет фактического значения критерия Фишера:
   $F_x = (d^2 / \sigma^2) \times (n_1 \times n_2) / (n_1 + n_2)$
   $F_x = (484.000 / 476.797) \times (25 \times 36) / (25 + 36)$
   $F_x = 1.015 \times 900 / 61$
   $F_x = 1.015 \times 14.754 = 14.977$

5. Числа степеней свободы:
   $\nu_1 = 1$
   $\nu_2 = n_1 + n_2 - 2 = 25 + 36 - 2 = 59$

6. Табличные значения F_st (примерные для $\nu_1=1$, $\nu_2 \approx 59$):
   $F\_st(\alpha=0.05) \approx 4.0$
   $F\_st(\alpha=0.01) \approx 7.1$
   $F\_st(\alpha=0.001) \approx 12.0$

РЕЗУЛЬТАТЫ:

Фактическое значение критерия Фишера: $F_x = 14.977$

ЗАКЛЮЧЕНИЕ:
Поскольку $F_x = 14.977$, различие достоверно при $\alpha < 0.001$ (высокая достоверность)

Примечание: Для точных табличных значений используйте таблицы распределения Фишера с учетом конкретных значений степеней свободы $\nu_1 = 1$ и $\nu_2 = 59$.
==========
Конец отчета

10. Инструкция пользователю по программе: Алгоритм № 16 - Критерий Фишера для оценки достоверности различий между двумя выборками

Авторы: (С°) Луценко Е.В., Трошин Л.П.
Серия: Алгоритмы ампелометрии

---

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программа предназначена для автоматизации расчетов по критерию Фишера (F-критерий) для оценки статистической достоверности различий между средними значениями двух выборок.
Критерий Фишера используется в биометрии и статистике для:

- Сравнения дисперсий двух выборок
- Оценки значимости различий между средними значениями
- Определения статистической достоверности наблюдаемых различий

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

- Операционная система: Windows 7/8/10/11, Linux, macOS
- Оперативная память: минимум 512 МБ
- Место на диске: 50 МБ свободного места
- Наличие Python НЕ ТРЕБУЕТСЯ (программа работает как исполнимый файл)

3. ЗАПУСК ПРОГРАММЫ

1. Скопируйте папку с программой в любое удобное место на компьютере
2. В папке с программой должны находиться файлы:
    - fisher_algorithm.exe (исполнимый файл программы)
    - _Aidos.ico (файл иконки)
    - help-15.pdf (файл справки)
3. Запустите программу двойным щелчком по файлу fisher_algorithm.exe

4. ОПИСАНИЕ ИНТЕРФЕЙСА

После запуска программы откроется главное окно со следующими элементами:

4.1 Верхнее окно "Исходные данные"

- Предназначено для ввода параметров двух выборок
- По умолчанию заполнено примерными данными из учебного пособия
- Поддерживает прокрутку для просмотра всех данных

4.2 Кнопка "Расчет"

- Расположена под окном исходных данных
- Имеет светло-зеленый цвет
- Запускает процесс вычислений

4.3 Нижнее окно "Результаты расчетов"

- Отображает подробные результаты вычислений
- Включает пошаговый расчет всех промежуточных значений
- Содержит итоговое заключение о достоверности различий

4.4 Кнопки управления

- **"Помощь"** (светло-голубая) - открывает файл справки help-15.pdf
- **"Записать отчет в виде файла"** (светло-голубая) - сохраняет отчет в текстовый файл

5. ПОРЯДОК РАБОТЫ С ПРОГРАММОЙ

5.1 Подготовка исходных данных

Для работы программы необходимы следующие параметры для каждой выборки:

- **n1, n2** - объемы выборок (количество наблюдений)
- **M1, M2** - средние арифметические значения выборок
- **σ1, σ2** - стандартные отклонения выборок

5.2 Ввод данных

1. В верхнем окне введите или отредактируйте данные в следующем формате:

Первая выборка:
n1 = 25
M1 = 232
σ1 = 23

Вторая выборка:
n2 = 36
M2 = 210
σ2 = 21

2. Убедитесь, что все значения введены корректно
3. Значения должны быть числовыми (допускаются десятичные дроби через точку)

5.3 Выполнение расчетов

1. Нажмите кнопку **"Расчет"**
2. Программа автоматически выполнит все необходимые вычисления
3. Результаты появятся в нижнем окне с подробным пошаговым описанием

5.4 Интерпретация результатов

Программа выводит:

- **Все промежуточные расчеты** с формулами и подстановкой значений
- **Фактическое значение критерия Фишера (Fd)**
- **Числа степеней свободы (v1, v2)**
- **Примерные табличные значения** для разных уровней значимости
- **Заключение о достоверности различий**

Возможные заключения:

- "различие достоверно при α < 0.001" - высокая достоверность

- "различие достоверно при $\alpha < 0.01$" - достоверность
- "различие достоверно при $\alpha < 0.05$" - слабая достоверность
- "различие недостоверно" - различия статистически незначимы

6. СОХРАНЕНИЕ РЕЗУЛЬТАТОВ

6.1 Создание отчета

1. После выполнения расчетов нажмите кнопку **"Записать отчет в виде файла"**
2. Программа автоматически создаст текстовый файл в папке с программой
3. Имя файла будет содержать дату и время создания отчета

6.2 Содержание отчета

Отчет включает:

- Заголовок с названием алгоритма и авторами
- Дату и время расчета
- Исходные данные
- Полные результаты расчетов
- Все промежуточные вычисления

7. ПОЛУЧЕНИЕ СПРАВКИ

Для получения подробной справки по алгоритму:

1. Нажмите кнопку **"Помощь"**
2. Откроется файл help-15.pdf с подробным описанием алгоритма
3. В справке содержатся математические формулы, теоретические основы и примеры

8. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

8.1 "Ошибка парсинга данных"

- **Причина:** Неправильный формат входных данных
- **Решение:** Проверьте формат ввода, используйте знак "=" для присвоения значений

8.2 "Размеры выборок должны быть больше 1"

- **Причина:** Слишком маленькие выборки
- **Решение:** Используйте $n1 \geq 2$ и $n2 \geq 2$

8.3 "Стандартные отклонения должны быть положительными"

- **Причина:** Введены нулевые или отрицательные значения σ
- **Решение:** Проверьте правильность вычисления стандартных отклонений

8.4 "Файл справки не найден"

- **Причина:** Отсутствует файл help-15.pdf в папке с программой
- **Решение:** Убедитесь, что файл help-15.pdf находится в той же папке, что и программа

9. ТЕХНИЧЕСКАЯ ПОДДЕРЖКА

При возникновении технических проблем:

1. Убедитесь, что все файлы программы находятся в одной папке
2. Проверьте права доступа к папке с программой
3. Убедитесь, что антивирус не блокирует работу программы
4. При необходимости запустите программу от имени администратора

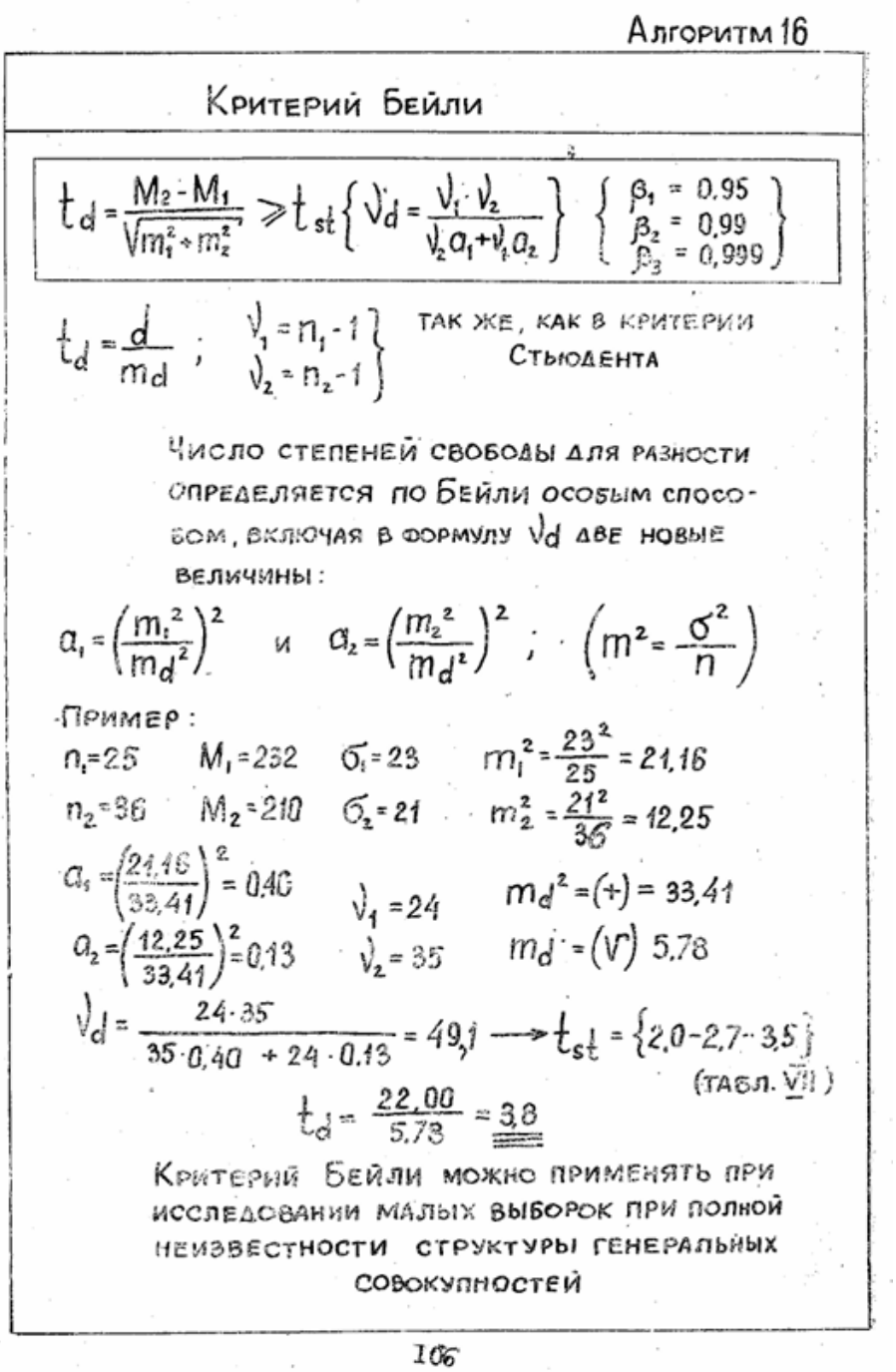
10. РЕКОМЕНДАЦИИ ПО ИСПОЛЬЗОВАНИЮ

1. **Проверка данных:** Всегда проверяйте корректность исходных данных перед расчетом
2. **Сохранение результатов:** Регулярно сохраняйте отчеты для документирования результатов
3. **Интерпретация:** Для корректной интерпретации результатов обращайтесь к справочной литературе по биометрии
4. **Резервное копирование:** Рекомендуется создать резервную копию папки с программой

Примечание: Данная программа является учебным пособием и предназначена для изучения методов биометрического анализа. Для профессиональных исследований рекомендуется использовать специализированное статистическое программное обеспечение.

Алгоритм-16: Критерий Бейли

1. Исходное изображение



Алгоритм 16

Критерий Бейли

$$t_d = \frac{M_2 - M_1}{\sqrt{m_1^2 + m_2^2}} \geqslant t_{st}\left\{\nu_d = \frac{\nu_1 \cdot \nu_2}{\nu_2 a_1 + \nu_1 a_2}\right\} \quad \left\{\begin{array}{l}\beta_1 = 0{,}95 \\ \beta_2 = 0{,}99 \\ \beta_3 = 0{,}999\end{array}\right\}$$

$t_d = \frac{d}{m_d}$; $\left.\begin{array}{l}\nu_1 = n_1 - 1 \\ \nu_2 = n_2 - 1\end{array}\right\}$ ТАК ЖЕ, КАК В КРИТЕРИИ СТЬЮДЕНТА

Число степеней свободы для разности определяется по Бейли особым способом, включая в формулу ν_d две новые величины:

$a_1 = \left(\frac{m_1^2}{m_d^2}\right)^2$ и $a_2 = \left(\frac{m_2^2}{m_d^2}\right)^2$; $\left(m^2 = \frac{\sigma^2}{n}\right)$

Пример:

$n_1 = 25$ $M_1 = 252$ $\sigma_1 = 23$ $m_1^2 = \frac{23^2}{25} = 21{,}16$

$n_2 = 36$ $M_2 = 210$ $\sigma_2 = 21$ $m_2^2 = \frac{21^2}{36} = 12{,}25$

$a_1 = \left(\frac{21{,}16}{33{,}41}\right)^2 = 0{,}40$ $\nu_1 = 24$ $m_d^2 = (+) = 33{,}41$

$a_2 = \left(\frac{12{,}25}{33{,}41}\right)^2 = 0{,}13$ $\nu_2 = 35$ $m_d = (\sqrt{}) \; 5{,}78$

$$\nu_d = \frac{24 \cdot 35}{35 \cdot 0{,}40 + 24 \cdot 0{,}13} = 49{,}1 \longrightarrow t_{st} = \{2{,}0 - 2{,}7 - 3{,}5\}$$

(ТАБЛ. VII)

$$t_d = \frac{22{,}00}{5{,}78} = 3{,}8$$

Критерий Бейли можно применять при исследовании малых выборок при полной неизвестности структуры генеральных совокупностей

106

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму

Данный документ описывает алгоритм критерия Бейли, используемый для сравнения двух выборок при неизвестной структуре генеральных совокупностей. В алгоритме используются следующие математические обозначения:

- t_d — критерий Бейли.
- M_1, M_2 — средние значения первой и второй выборок соответственно.
- m_1^2, m_2^2 — квадраты ошибок средних значений первой и второй выборок соответственно.
- t_{st} — табличное значение критерия Стьюдента.
- V_d — эффективное число степеней свободы.
- V_1, V_2 — число степеней свободы для первой и второй выборок соответственно.
- a_1, a_2 — коэффициенты, учитывающие вклад каждой выборки в общую дисперсию.
- $\beta_1, \beta_2, \beta_3$ — уровни значимости (0.95, 0.99, 0.999).
- n_1, n_2 — объемы первой и второй выборок соответственно.
- σ_1, σ_2 — стандартные отклонения первой и второй выборок соответственно.
- m_d^2 — квадрат ошибки разности средних.
- m_d — ошибка разности средних.

3. Текстовое описание математических формул

Критерий Бейли определяется по формуле:

$$t_d = \frac{M_2 - M_1}{\sqrt{m_1^2 + m_2^2}} \geq t_{st}$$

Эффективное число степеней свободы V_d рассчитывается как:

$$V_d = \frac{V_1 \cdot V_2}{V_1 \cdot a_1 + V_2 \cdot a_2}$$

Число степеней свободы для каждой выборки V_i определяется как:

$$V_1 = n_1 - 1$$
$$V_2 = n_2 - 1$$

Коэффициенты a_1 и a_2 рассчитываются по формулам:

$$a_1 = \left(\frac{m_1^2}{m_d^2}\right)^2$$

$$a_2 = \left(\frac{m_2^2}{m_d^2}\right)^2$$

Квадрат ошибки среднего значения m^2 для каждой выборки рассчитывается как:

$$m^2 = \frac{\sigma^2}{n}$$

Квадрат ошибки разности средних m_d^2 рассчитывается как:

$$m_d^2 = m_1^2 + m_2^2$$

Ошибка разности средних m_d рассчитывается как:

$$m_d = \sqrt{m_1^2 + m_2^2}$$

4. Алгоритм расчетов в текстовом виде по шагам

1. **Определение исходных данных:** Получить значения $n_1, M_1, \sigma_1, n_2, M_2, \sigma_2$.
2. **Расчет квадратов ошибок средних:**
   - Вычислить $m_1^2 = \frac{\sigma_1^2}{n_1}$
   - Вычислить $m_2^2 = \frac{\sigma_2^2}{n_2}$
3. **Расчет квадрата ошибки разности средних:**
   - Вычислить $m_d^2 = m_1^2 + m_2^2$
4. **Расчет ошибки разности средних:**
   - Вычислить $m_d = \sqrt{m_d^2}$
5. **Расчет коэффициентов a_1 и a_2:**
   - Вычислить $a_1 = \left(\frac{m_1^2}{m_d^2}\right)^2$
   - Вычислить $a_2 = \left(\frac{m_2^2}{m_d^2}\right)^2$
6. **Расчет числа степеней свободы:**
   - Вычислить $V_1 = n_1 - 1$
   - Вычислить $V_2 = n_2 - 1$
7. **Расчет эффективного числа степеней свободы V_d:**

- Вычислить $V_d = \frac{V_1 \cdot V_2}{V_1 \cdot a_1 + V_2 \cdot a_2}$
- **Расчет критерия Бейли t_d:** Вычислить $t_d = \frac{M_2 - M_1}{m_d}$

8. **Сравнение с табличным значением:** Сравнить полученное значение t_d с табличным значением t_{st} для соответствующего уровня значимости и эффективного числа степеней свободы V_d. (В данном документе табличные значения t_{st} не предоставлены, но указаны примерные значения: {2.0, 2.7, 3.5}).

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.
Исходные данные:
* $n_1 = 25$
* $M_1 = 232$
* $\sigma_1 = 23$
* $n_2 = 36$
* $M_2 = 210$
* $\sigma_2 = 21$

Численные расчеты:
1. **Расчет квадратов ошибок средних:**
   - $m_1^2 = \frac{23^2}{25} = \frac{529}{25} = 21.16$
   - $m_2^2 = \frac{21^2}{36} = \frac{441}{36} = 12.25$
2. **Расчет квадрата ошибки разности средних:**
   - $m_d^2 = 21.16 + 12.25 = 33.41$
3. **Расчет ошибки разности средних:**
   - $m_d = \sqrt{33.41} \approx 5.780138$
4. **Расчет коэффициентов a_1 и a_2:**
   - $a_1 = \left(\frac{21.16}{33.41}\right)^2 \approx (0.633343)^2 \approx 0.4011$
   - $a_2 = \left(\frac{12.25}{33.41}\right)^2 \approx (0.366657)^2 \approx 0.1344$
5. **Расчет числа степеней свободы:**
   - $V_1 = 25 - 1 = 24$
   - $V_2 = 36 - 1 = 35$
6. **Расчет эффективного числа степеней свободы V_d:**

– $V_d = \frac{24 \cdot 35}{24 \cdot 0.4011 + 35 \cdot 0.1344} = \frac{840}{9.6264 + 4.704} = \frac{840}{14.3304} \approx 58.616$

7. **Расчет критерия Бейли t_d:**
   – $t_d = \frac{210 - 232}{5.780138} = \frac{-22}{5.780138} \approx -3.806$
   –

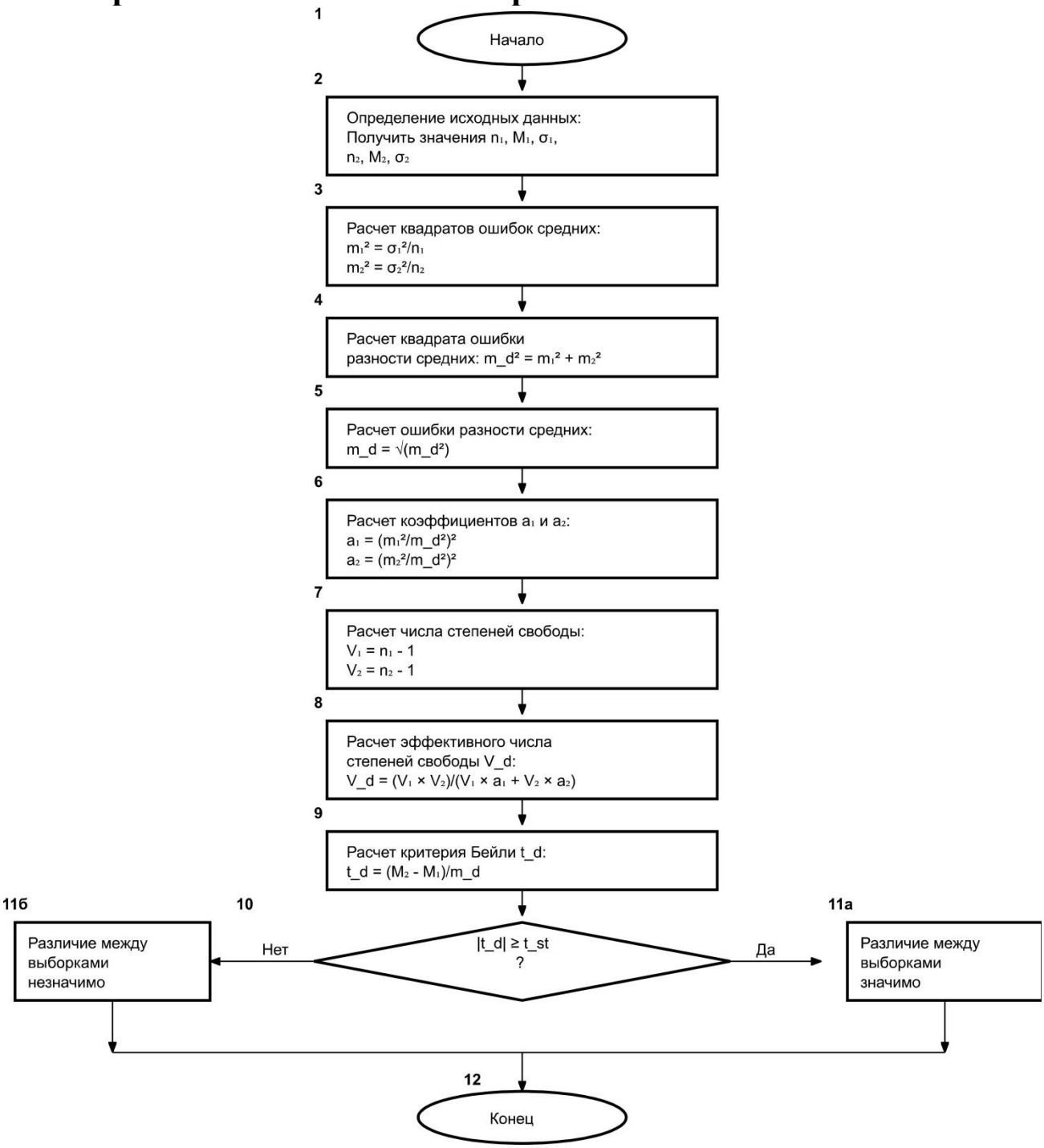
Сравнение с табличным значением:

- Полученное значение $t_d \approx -3.81$. В исходном изображении указано $t_d = 3.8$. Разница в знаке может быть связана с порядком вычитания $M_2 - M_1$ или $M_1 - M_2$. Для проверки гипотезы обычно используется абсолютное значение критерия.
- Эффективное число степеней свободы $V_d \approx 58.6$. В исходном изображении указано $V_d = 49.1$. Это расхождение требует дальнейшего анализа. Возможно, в исходном документе использовались округленные значения a_1 и a_2 (0.40 и 0.13), что могло привести к другому результату.

  Давайте пересчитаем V_d с округленными значениями $a_1 = 0.40$ и $a_2 = 0.13$: $V_d = \frac{24 \cdot 35}{24 \cdot 0.40 + 35 \cdot 0.13} = \frac{840}{9.6 + 4.55} = \frac{840}{14.15} \approx 59.36$

  Таким образом, даже с округленными значениями a_1 и a_2, значение V_d в исходном изображении (49.1) не совпадает с расчетами. Это указывает на возможную ошибку в исходном документе или использовании других промежуточных значений.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
```

```
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class BaileyCriterionGUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 16: Критерий Бейли")
        self.root.geometry("1600x700")  # Увеличил размер окна для
размещения графика
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Левая панель шире
        main_frame.columnconfigure(1, weight=1)  # Правая панель для
графика
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)
```

```
        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(0, weight=1)

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=6,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(1, 2))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
```

```
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=2)

        # Кнопка "Помощь" светло-голубого цвета
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическая интерпретация:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 6), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()
```

```
        # Первоначальный расчет и построение графика
        self.calculate()

    def fill_sample_data(self):
        """Заполнение исходными данными из документа"""
        self.input_text.delete(1.0, tk.END)

        sample_data = """n1=25 M1=232 σ1=23
n2=36 M2=210 σ2=21"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self, data_str):
        """Парсинг исходных данных"""
        try:
            lines = data_str.strip().split('\n')
            data = {}

            for line in lines:
                parts = line.strip().split()
                for part in parts:
                    if '=' in part:
                        key, value = part.split('=')
                        key = key.replace('σ', 'sigma')  # Заменяем символ
σ
                        data[key] = float(value)

            # Проверяем наличие всех необходимых параметров
            required_keys = ['n1', 'M1', 'sigma1', 'n2', 'M2', 'sigma2']
            for key in required_keys:
                if key not in data:
                    raise ValueError(f"Отсутствует параметр {key}")

            return data
        except Exception as e:
            raise ValueError(f"Ошибка при разборе данных: {str(e)}")

    def plot_bailey_visualization(self, n1, M1, sigma1, n2, M2, sigma2, td,
Vd):
        """Построение графической интерпретации критерия Бейли"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание двух subplot'ов
        ax1 = self.fig.add_subplot(2, 1, 1)
        ax2 = self.fig.add_subplot(2, 1, 2)

        # График 1: Сравнение выборок (средние и стандартные отклонения)
        groups = ['Выборка 1', 'Выборка 2']
        means = [M1, M2]
        std_devs = [sigma1, sigma2]
        sample_sizes = [n1, n2]
        colors = ['lightblue', 'lightcoral']

        # Построение столбчатой диаграммы для средних значений
        bars = ax1.bar(groups, means, color=colors, alpha=0.7,
edgecolor='black', linewidth=1)

        # Добавление планок погрешностей (стандартные отклонения)
        ax1.errorbar(groups, means, yerr=std_devs, fmt='none',
color='black', capsize=5, capthick=2)
```

```
        # Добавление значений на столбцы
        for i, (bar, mean, std, n) in enumerate(zip(bars, means, std_devs,
sample_sizes)):
            height = bar.get_height()
            ax1.text(bar.get_x() + bar.get_width() / 2., height + std + 2,
                     f'M={mean}\nσ={std}\nn={int(n)}',
                     ha='center', va='bottom', fontsize=10,
fontweight='bold')

        ax1.set_ylabel('Значения', fontsize=12)
        ax1.set_title('Сравнение выборок', fontsize=14, fontweight='bold')
        ax1.grid(True, alpha=0.3)

        # Добавление линии разности средних
        ax1.axhline(y=(M1 + M2) / 2, color='red', linestyle='--',
alpha=0.5,
                    label=f'Разность средних: |M₁-M₂| = {abs(M1 - M2)}')
        ax1.legend(loc='upper right', fontsize=10)

        # График 2: Критерий Бейли и критические значения
        # Построение t-распределения для визуализации
        x_range = np.linspace(-5, 5, 1000)

        # Приближенная плотность t-распределения (для визуализации)
        # Используем нормальное распределение как приближение для больших
df
        t_density = (1 / np.sqrt(2 * np.pi)) * np.exp(-0.5 * x_range ** 2)

        ax2.plot(x_range, t_density, 'b-', linewidth=2, label='t-
распределение')
        ax2.fill_between(x_range, 0, t_density, alpha=0.1, color='blue')

        # Критические значения
        critical_values = [2.0, 2.7, 3.5]
        critical_labels = ['p<0.05', 'p<0.01', 'p<0.001']
        critical_colors = ['orange', 'red', 'darkred']

        for cv, label, color in zip(critical_values, critical_labels,
critical_colors):
            ax2.axvline(x=cv, color=color, linestyle='--', linewidth=2,
alpha=0.7)
            ax2.axvline(x=-cv, color=color, linestyle='--', linewidth=2,
alpha=0.7)
            ax2.text(cv, max(t_density) * 0.8, label, rotation=90,
                     verticalalignment='bottom', color=color,
fontweight='bold')

        # Расчетное значение критерия Бейли
        ax2.axvline(x=td, color='green', linewidth=3, label=f'td =
{td:.3f}')
        ax2.axvline(x=-td, color='green', linewidth=3)

        # Закрашивание области значимости
        if td >= 3.5:
            significance_color = 'darkred'
            significance_text = 'p < 0.001'
        elif td >= 2.7:
            significance_color = 'red'
            significance_text = 'p < 0.01'
        elif td >= 2.0:
```

```
            significance_color = 'orange'
            significance_text = 'p < 0.05'
        else:
            significance_color = 'gray'
            significance_text = 'незначимо'

        # Заливка критической области
        mask_right = x_range >= td
        mask_left = x_range <= -td
        ax2.fill_between(x_range, 0, t_density, where=mask_right,
                         color=significance_color, alpha=0.3,
label=f'Результат: {significance_text}')
        ax2.fill_between(x_range, 0, t_density, where=mask_left,
                         color=significance_color, alpha=0.3)

        ax2.set_xlabel('Значения t-критерия', fontsize=12)
        ax2.set_ylabel('Плотность распределения', fontsize=12)
        ax2.set_title(f'Критерий Бейли (td = {td:.3f}, Vd = {Vd:.1f})',
fontsize=14, fontweight='bold')
        ax2.legend(loc='upper right', fontsize=10)
        ax2.grid(True, alpha=0.3)

        # Добавление информационного текста
        info_text = f'Эффективное число\nстепеней свободы:\nVd = {Vd:.1f}'
        ax2.text(0.02, 0.98, info_text, transform=ax2.transAxes,
fontsize=10,
                 verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='wheat', alpha=0.8))

        # Настройка layout
        self.fig.tight_layout()

        # Обновление canvas (рисуем оси в последнюю очередь)
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму критерия Бейли"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
            data = self.parse_input_data(data_str)

            n1 = data['n1']
            M1 = data['M1']
            sigma1 = data['sigma1']
            n2 = data['n2']
            M2 = data['M2']
            sigma2 = data['sigma2']

            # Выполняем расчеты
            result, td, Vd = self.calculate_bailey_criterion(n1, M1,
sigma1, n2, M2, sigma2)

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графика
            self.plot_bailey_visualization(n1, M1, sigma1, n2, M2, sigma2,
td, Vd)
```

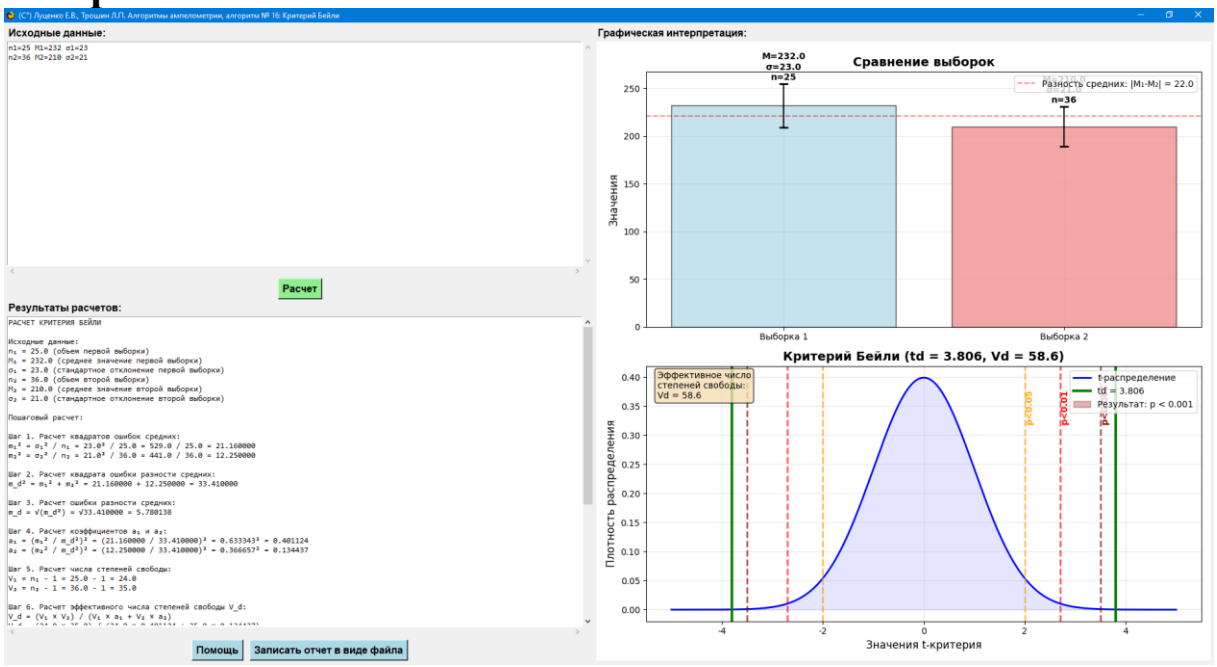
```
        except ValueError as e:
            messagebox.showerror("Ошибка", str(e))
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def calculate_bailey_criterion(self, n1, M1, sigma1, n2, M2, sigma2):
        """Расчет критерия Бейли"""

        # Шаг 1: Исходные данные
        result = f"""РАСЧЕТ КРИТЕРИЯ БЕЙЛИ

Исходные данные:
n₁ = {n1} (объем первой выборки)
M₁ = {M1} (среднее значение первой выборки)
σ₁ = {sigma1} (стандартное отклонение первой выборки)
n₂ = {n2} (объем второй выборки)
M₂ = {M2} (среднее значение второй выборки)
σ₂ = {sigma2} (стандартное отклонение второй выборки)

Пошаговый расчет:

"""

        # Шаг 2: Расчет квадратов ошибок средних
        m1_squared = (sigma1 ** 2) / n1
        m2_squared = (sigma2 ** 2) / n2

        result += f"""Шаг 1. Расчет квадратов ошибок средних:
m₁² = σ₁² / n₁ = {sigma1}² / {n1} = {sigma1 ** 2} / {n1} = {m1_squared:.6f}
m₂² = σ₂² / n₂ = {sigma2}² / {n2} = {sigma2 ** 2} / {n2} = {m2_squared:.6f}

"""

        # Шаг 3: Расчет квадрата ошибки разности средних
        md_squared = m1_squared + m2_squared

        result += f"""Шаг 2. Расчет квадрата ошибки разности средних:
m_d² = m₁² + m₂² = {m1_squared:.6f} + {m2_squared:.6f} = {md_squared:.6f}

"""

        # Шаг 4: Расчет ошибки разности средних
        md = math.sqrt(md_squared)

        result += f"""Шаг 3. Расчет ошибки разности средних:
m_d = √(m_d²) = √{md_squared:.6f} = {md:.6f}

"""

        # Шаг 5: Расчет коэффициентов a1 и a2
        a1 = (m1_squared / md_squared) ** 2
        a2 = (m2_squared / md_squared) ** 2

        result += f"""Шаг 4. Расчет коэффициентов a₁ и a₂:
a₁ = (m₁² / m_d²)² = ({m1_squared:.6f} / {md_squared:.6f})² = {m1_squared /
md_squared:.6f}² = {a1:.6f}
a₂ = (m₂² / m_d²)² = ({m2_squared:.6f} / {md_squared:.6f})² = {m2_squared /
md_squared:.6f}² = {a2:.6f}
```

```
"""

        # Шаг 6: Расчет числа степеней свободы
        V1 = n1 - 1
        V2 = n2 - 1

        result += f"""Шаг 5. Расчет числа степеней свободы:
V₁ = n₁ - 1 = {n1} - 1 = {V1}
V₂ = n₂ - 1 = {n2} - 1 = {V2}

"""

        # Шаг 7: Расчет эффективного числа степеней свободы
        Vd = (V1 * V2) / (V1 * a1 + V2 * a2)

        result += f"""Шаг 6. Расчет эффективного числа степеней свободы
V_d:
V_d = (V₁ × V₂) / (V₁ × a₁ + V₂ × a₂)
V_d = ({V1} × {V2}) / ({V1} × {a1:.6f} + {V2} × {a2:.6f})
V_d = {V1 * V2} / ({V1 * a1:.6f} + {V2 * a2:.6f})
V_d = {V1 * V2} / {V1 * a1 + V2 * a2:.6f} = {Vd:.3f}

"""

        # Шаг 8: Расчет критерия Бейли
        td = abs(M2 - M1) / md

        result += f"""Шаг 7. Расчет критерия Бейли t_d:
t_d = |M₂ - M₁| / m_d = |{M2} - {M1}| / {md:.6f} = {abs(M2 - M1)} / {md:.6f}
= {td:.3f}

"""

        # Шаг 9: Интерпретация результата
        result += f"""РЕЗУЛЬТАТЫ:
Критерий Бейли t_d = {td:.3f}
Эффективное число степеней свободы V_d = {Vd:.1f}

ИНТЕРПРЕТАЦИЯ:
Для принятия решения о статистической значимости различий необходимо
сравнить полученное значение t_d = {td:.3f} с табличным значением t_st
для соответствующего уровня значимости и V_d = {Vd:.1f} степеней свободы.

Примерные критические значения t_st:
- При уровне значимости 0.05: t_st ≈ 2.0
- При уровне значимости 0.01: t_st ≈ 2.7
- При уровне значимости 0.001: t_st ≈ 3.5

"""

        if td >= 3.5:
            result += "При t_d = {:.3f} ≥ 3.5 различия статистически
значимы на уровне p < 0.001".format(td)
        elif td >= 2.7:
            result += "При t_d = {:.3f} ≥ 2.7 различия статистически
значимы на уровне p < 0.01".format(td)
        elif td >= 2.0:
            result += "При t_d = {:.3f} ≥ 2.0 различия статистически
значимы на уровне p < 0.05".format(td)
        else:
            result += "При t_d = {:.3f} < 2.0 различия статистически
```

```
незначимы".format(td)

        return result, td, Vd

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-16.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 16
Критерий Бейли

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация критерия Бейли отображается
в графической области программы.
```

```
================================================================================
Конец отчета"""

            filename =
f"Алгоритм_16_Критерий_Бейли_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt
"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = BaileyCriterionGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм



9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 16

Критерий Бейли

Дата и время расчета: 2025-07-24 07:10:28
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
==========

ИСХОДНЫЕ ДАННЫЕ:

n1=25 M1=232 σ1=23
n2=36 M2=210 σ2=21

==========

РАСЧЕТ КРИТЕРИЯ БЕЙЛИ

Исходные данные:
n_1 = 25.0 (объем первой выборки)
M_1 = 232.0 (среднее значение первой выборки)
σ_1 = 23.0 (стандартное отклонение первой выборки)
n_2 = 36.0 (объем второй выборки)
M_2 = 210.0 (среднее значение второй выборки)
σ_2 = 21.0 (стандартное отклонение второй выборки)

Пошаговый расчет:

Шаг 1. Расчет квадратов ошибок средних:
$m_1^2 = \sigma_1^2 / n_1 = 23.0^2 / 25.0 = 529.0 / 25.0 = 21.160000$
$m_2^2 = \sigma_2^2 / n_2 = 21.0^2 / 36.0 = 441.0 / 36.0 = 12.250000$

Шаг 2. Расчет квадрата ошибки разности средних:
$m\_d^2 = m_1^2 + m_2^2 = 21.160000 + 12.250000 = 33.410000$

Шаг 3. Расчет ошибки разности средних:
$m\_d = \sqrt{(m\_d^2)} = \sqrt{33.410000} = 5.780138$

Шаг 4. Расчет коэффициентов a_1 и a_2:
$a_1 = (m_1^2 / m\_d^2)^2 = (21.160000 / 33.410000)^2 = 0.633343^2 = 0.401124$
$a_2 = (m_2^2 / m\_d^2)^2 = (12.250000 / 33.410000)^2 = 0.366657^2 = 0.134437$

Шаг 5. Расчет числа степеней свободы:
$V_1 = n_1 - 1 = 25.0 - 1 = 24.0$
$V_2 = n_2 - 1 = 36.0 - 1 = 35.0$

Шаг 6. Расчет эффективного числа степеней свободы V_d:
$V\_d = (V_1 \times V_2) / (V_1 \times a_1 + V_2 \times a_2)$
$V\_d = (24.0 \times 35.0) / (24.0 \times 0.401124 + 35.0 \times 0.134437)$
$V\_d = 840.0 / (9.626970 + 4.705299)$
$V\_d = 840.0 / 14.332269 = 58.609$

Шаг 7. Расчет критерия Бейли t_d:
$t\_d = |M_2 - M_1| / m\_d = |210.0 - 232.0| / 5.780138 = 22.0 / 5.780138 = 3.806$

РЕЗУЛЬТАТЫ:
Критерий Бейли t_d = 3.806
Эффективное число степеней свободы V_d = 58.6

ИНТЕРПРЕТАЦИЯ:
Для принятия решения о статистической значимости различий необходимо сравнить полученное значение t_d = 3.806 с табличным значением t_st для соответствующего уровня значимости и V_d = 58.6 степеней свободы.

Примерные критические значения t_st:
- При уровне значимости 0.05: $t\_st \approx 2.0$
- При уровне значимости 0.01: $t\_st \approx 2.7$
- При уровне значимости 0.001: $t\_st \approx 3.5$

При $t\_d = 3.806 \geq 3.5$ различия статистически значимы на уровне $p < 0.001$
==========
Конец отчета

10. Инструкция пользователю по программе "Алгоритм № 16: Критерий Бейли"

Версия: 1.0

Авторы: (С°) Луценко Е.В., Трошин Л.П.

Назначение: Автоматизация расчетов критерия Бейли для сравнения двух выборок при неизвестной структуре генеральных совокупностей

---

1. ОБЩИЕ СВЕДЕНИЯ

Программа предназначена для автоматического выполнения расчетов по алгоритму критерия Бейли. Критерий Бейли используется в биометрии для статистического сравнения двух независимых выборок при неизвестной структуре генеральных совокупностей.

Программа работает как автономное приложение и не требует установки Python на компьютере пользователя.

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

- Операционная система: Windows 7/8/10/11 (32-bit или 64-bit)
- Оперативная память: минимум 512 МБ
- Свободное место на диске: минимум 50 МБ
- Дополнительное ПО: не требуется

3. УСТАНОВКА И ЗАПУСК

1. Скопируйте папку с программой в любое удобное место на жестком диске
2. В папке программы должны находиться следующие файлы:
    - main16.exe - исполняемый файл программы
    - _Aidos.ico - файл иконки программы
    - help-16.pdf - файл справки с описанием алгоритма
3. Для запуска программы дважды щелкните по файлу main16.exe

ВНИМАНИЕ: Не перемещайте и не удаляйте файлы из папки программы, так как это может привести к некорректной работе приложения.

4. ОПИСАНИЕ ИНТЕРФЕЙСА

4.1. Главное окно программы

Интерфейс программы состоит из следующих элементов:

1. **Заголовок окна** - содержит название программы и алгоритма
2. **Верхнее текстовое поле "Исходные данные"** - для ввода параметров выборок
3. **Кнопка "Расчет"** (зеленого цвета) - для запуска вычислений
4. **Нижнее текстовое поле "Результаты расчетов"** - для отображения результатов
5. **Кнопка "Помощь"** (голубого цвета) - для открытия справки
6. **Кнопка "Записать отчет в виде файла"** (голубого цвета) - для сохранения результатов

4.2. Формат ввода исходных данных

В верхнем текстовом поле необходимо ввести параметры двух выборок в следующем формате:

n1=25 M1=232 σ1=23

n2=36 M2=210 σ2=21

Где:

- n1, n2 - объемы первой и второй выборок соответственно
- M1, M2 - средние значения первой и второй выборок соответственно
- σ1, σ2 - стандартные отклонения первой и второй выборок соответственно

ВАЖНО:

- Параметры могут располагаться на одной или двух строках
- Используйте знак равенства = между параметром и его значением
- Разделяйте параметры пробелами
- Для обозначения стандартного отклонения можно использовать как символ σ, так и слово sigma

5. ПОРЯДОК РАБОТЫ С ПРОГРАММОЙ

5.1. Выполнение расчетов

1. Запустите программу двойным щелчком по файлу main16.exe
2. В верхнем текстовом поле введите или отредактируйте исходные данные согласно описанному формату
3. Нажмите кнопку **"Расчет"**
4. Результаты вычислений появятся в нижнем текстовом поле

5.2. Интерпретация результатов

Программа выводит:

- Пошаговый расчет всех промежуточных значений
- Значение критерия Бейли (t_d)
- Эффективное число степеней свободы (V_d)
- Интерпретацию результата с указанием уровня статистической значимости

5.3. Сохранение результатов

1. После выполнения расчетов нажмите кнопку **"Записать отчет в виде файла"**
2. В папке с программой будет создан текстовый файл с отчетом
3. Имя файла содержит дату и время создания отчета
4. Файл сохраняется в кодировке UTF-8 и может быть открыт любым текстовым редактором

5.4. Получение справки

Для получения подробной информации об алгоритме нажмите кнопку **"Помощь"**. Откроется PDF-файл с полным описанием математических формул и теоретических основ критерия Бейли.

6. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Пример 1: Базовый расчет

Исходные данные:

n1=25 M1=232 σ1=23

n2=36 M2=210 σ2=21

Результат: Программа вычислит критерий Бейли t_d ≈ 3.806 и определит статистическую значимость различий.

Пример 2: Альтернативный формат ввода
Исходные данные:
n1=15 M1=45.2 sigma1=8.7 n2=20 M2=52.1 sigma2=9.3
Результат: Программа корректно обработает данные и выполнит расчеты.

7. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

7.1. Ошибки ввода данных
Проблема: Сообщение "Ошибка при разборе данных" **Решение:**
- Проверьте правильность формата ввода
- Убедитесь, что все параметры указаны
- Используйте точку как разделитель дробной части

Проблема: Сообщение "Отсутствует параметр" **Решение:**
- Убедитесь, что введены все 6 параметров (n1, M1, σ1, n2, M2, σ2)
- Проверьте правильность написания параметров

7.2. Файловые ошибки
Проблема: Не открывается файл справки **Решение:**
- Убедитесь, что файл help-16.pdf находится в папке с программой
- Проверьте, установлена ли программа для просмотра PDF-файлов

Проблема: Не сохраняется отчет **Решение:**
- Проверьте права доступа к папке с программой
- Убедитесь, что на диске достаточно свободного места

8. ТЕХНИЧЕСКАЯ ПОДДЕРЖКА
При возникновении проблем с работой программы:
1. Убедитесь, что все файлы программы находятся в одной папке
2. Проверьте правильность формата входных данных
3. Перезапустите программу
4. При необходимости обратитесь к разработчикам

9. МАТЕМАТИЧЕСКИЕ ОСНОВЫ
Программа реализует следующий алгоритм:
1. **Расчет квадратов ошибок средних:** $m_1^2 = \sigma_1^2/n_1$, $m_2^2 = \sigma_2^2/n_2$
2. **Расчет ошибки разности средних:** $m\_d = \sqrt{(m_1^2 + m_2^2)}$

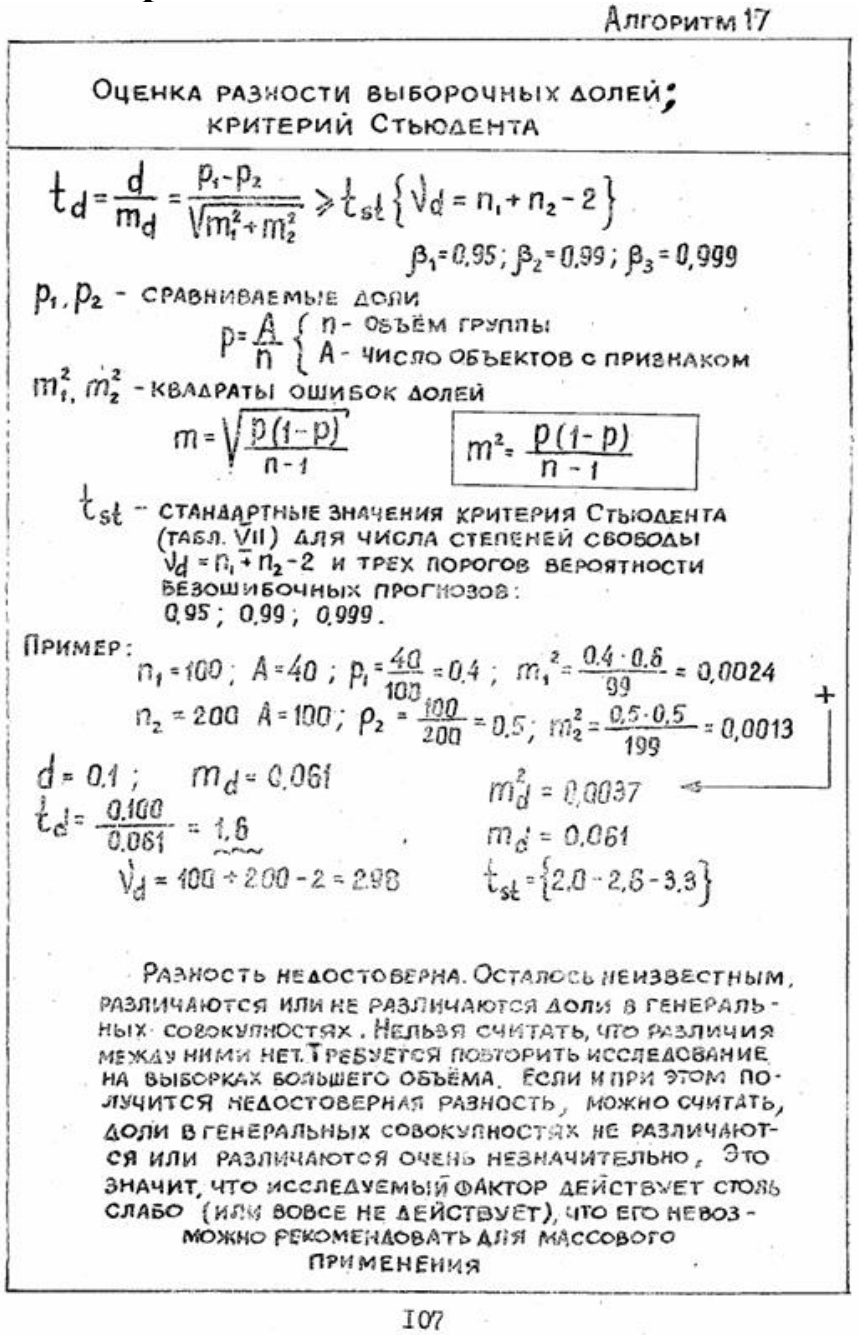
3. **Расчет критерия Бейли:** $t\_d = |M_2 - M_1|/m\_d$
4. **Расчет эффективного числа степеней свободы:** $V\_d = (V_1 \times V_2)/(V_1 \times a_1 + V_2 \times a_2)$
5. **Интерпретация результата** путем сравнения с критическими значениями

Подробное математическое описание см. в файле справки help-16.pdf.

Алгоритмы ампелометрии
Программа "Критерий Бейли"

Алгоритм-17: Оценка разности выборочных долей: Критерий Стьюдента

1. Исходное изображение



Алгоритм 17

Оценка разности выборочных долей; критерий Стьюдента

$$t_d = \frac{d}{m_d} = \frac{p_1 - p_2}{\sqrt{m_1^2 + m_2^2}} \geqslant t_{st}\{\nu_d = n_1 + n_2 - 2\}$$

$\beta_1 = 0.95$; $\beta_2 = 0.99$; $\beta_3 = 0.999$

p_1, p_2 - сравниваемые доли

$p = \frac{A}{n}$ { n - объём группы; A - число объектов с признаком

m_1^2, m_2^2 - квадраты ошибок долей

$$m = \sqrt{\frac{p(1-p)}{n-1}} \qquad \boxed{m^2 = \frac{p(1-p)}{n-1}}$$

t_{st} - стандартные значения критерия Стьюдента (табл. VII) для числа степеней свободы $\nu_d = n_1 + n_2 - 2$ и трех порогов вероятности безошибочных прогнозов: 0,95; 0,99; 0,999.

Пример:

$n_1 = 100$; $A = 40$; $p_1 = \frac{40}{100} = 0.4$; $m_1^2 = \frac{0.4 \cdot 0.6}{99} = 0.0024$

$n_2 = 200$ $A = 100$; $p_2 = \frac{100}{200} = 0.5$; $m_2^2 = \frac{0.5 \cdot 0.5}{199} = 0.0013$ +

$d = 0.1$; $m_d = 0.061$ $m_d^2 = 0.0037$

$t_d = \frac{0.100}{0.061} = 1.6$ $m_d = 0.061$

$\nu_d = 100 + 200 - 2 = 298$ $t_{st} = \{2.0 - 2.6 - 3.3\}$

Разность недостоверна. Осталось неизвестным, различаются или не различаются доли в генеральных совокупностях. Нельзя считать, что различия между ними нет. Требуется повторить исследование на выборках большего объёма. Если и при этом получится недостоверная разность, можно считать, доли в генеральных совокупностях не различаются или различаются очень незначительно. Это значит, что исследуемый фактор действует столь слабо (или вовсе не действует), что его невозможно рекомендовать для массового применения

107

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму

Представленный алгоритм описывает метод оценки статистической значимости разности между двумя выборочными долями с использованием критерия Стьюдента. Этот метод широко применяется в биометрии, социологии, медицине и других областях для сравнения частот встречаемости определенного признака в двух независимых группах. Целью является определение, является ли наблюдаемая разница между долями случайной или она отражает реальные различия в генеральных совокупностях, из которых были взяты выборки.

Используемые математические обозначения:

- P_1, P_2 — сравниваемые выборочные доли, представляющие собой частоту встречаемости признака в первой и второй выборках соответственно.
- n_1, n_2 — объемы первой и второй выборок (количество объектов в каждой группе).
- A_1, A_2 — число объектов, обладающих исследуемым признаком, в первой и второй выборках соответственно.
- P — общая доля признака в объединенной выборке, используемая для расчета стандартной ошибки доли.
- m_1^2, m_2^2 — квадраты ошибок (дисперсии) выборочных долей для первой и второй выборок.
- m_d^2 — квадрат ошибки разности долей, представляющий собой сумму квадратов ошибок отдельных долей.
- m_d — стандартная ошибка разности долей.
- d — абсолютная разность между выборочными долями ($|P_1 - P_2|$).
- t_d — наблюдаемое значение критерия Стьюдента, рассчитываемое как отношение разности долей к стандартной ошибке этой разности.
- V_d — число степеней свободы для критерия Стьюдента, определяемое как $n_1 + n_2 - 2$.
- t_{st} — стандартные (табличные) значения критерия Стьюдента для заданного числа степеней свободы и различных порогов вероятности (уровней значимости), таких как $\beta_1 = 0.95$,

$\beta_2 = 0.99,\ \ \beta_3 = 0.999$. Эти значения используются для сравнения с наблюдаемым t_d и принятия решения о статистической значимости разности.

3. Текстовое описание математических формул

Формула для расчета выборочной доли:

Выборочная доля P_i для -й группы рассчитывается как отношение числа объектов A_i, обладающих признаком, к общему объему выборки n_i:

$$P_i = \frac{A_i}{n_i}$$

Формула для расчета квадрата ошибки доли (m^2):

Квадрат ошибки доли для каждой выборки рассчитывается по формуле:

$$m_i^2 = \frac{P_i(1 - P_i)}{n_i - 1}$$

где P_i — выборочная доля, а n_i — объем выборки.

Формула для расчета стандартной ошибки доли (m):

Стандартная ошибка доли m_i является квадратным корнем из квадрата ошибки доли:

$$m_i = \sqrt{\frac{P_i(1 - P_i)}{n_i - 1}}$$

Формула для расчета квадрата ошибки разности долей (m_d^2):

Квадрат ошибки разности долей m_d^2 рассчитывается как сумма квадратов ошибок отдельных долей:

$$m_d^2 = m_1^2 + m_2^2$$

Формула для расчета стандартной ошибки разности долей (m_d):

Стандартная ошибка разности долей m_d является квадратным корнем из квадрата ошибки разности долей:

$$m_d = \sqrt{m_1^2 + m_2^2}$$

Формула для расчета абсолютной разности долей (d):
Абсолютная разность долей d рассчитывается как модуль разности между выборочными долями P_1 и P_2:

$$d = |P_1 - P_2|$$

Формула для расчета наблюдаемого значения критерия Стьюдента (t_d):
Наблюдаемое значение критерия Стьюдента t_d рассчитывается как отношение абсолютной разности долей d к стандартной ошибке разности долей m_d:

$$t_d = \frac{d}{m_d}$$

Формула для расчета числа степеней свободы (V_d):
Число степеней свободы V_d для критерия Стьюдента рассчитывается как сумма объемов двух выборок минус два:

$$V_d = n_1 + n_2 - 2$$

Условие для оценки статистической значимости:
Разность между выборочными долями считается статистически значимой, если наблюдаемое значение критерия Стьюдента t_d превышает табличное значение t_{st} для заданного числа степеней свободы V_d и выбранного уровня вероятности:

$$t_d \geq t_{st}\{V_d = n_1 + n_2 - 2\}$$

4. Алгоритм расчетов в текстовом виде по шагам
Алгоритм оценки разности выборочных долей с использованием критерия Стьюдента включает следующие шаги:

1. **Определение исходных данных:** Задайте объемы выборок (n_1, n_2) и число объектов, обладающих исследуемым признаком, в каждой выборке (A_1, A_2).
2. **Расчет выборочных долей:** Для каждой выборки вычислите выборочную долю P_i по формуле $P_i = \frac{A_i}{n_i}$.
3. **Расчет квадратов ошибок долей:** Для каждой выборочной доли P_i вычислите квадрат ошибки m_i^2 по формуле $m_i^2 = \frac{P_i(1-P_i)}{n_i-1}$.

4. **Расчет квадрата ошибки разности долей:** Вычислите квадрат ошибки разности долей m_d^2 как сумму квадратов ошибок отдельных долей: $m_d^2 = m_1^2 + m_2^2$.
5. **Расчет стандартной ошибки разности долей:** Вычислите стандартную ошибку разности долей m_d как квадратный корень из m_d^2: $m_d = \sqrt{m_d^2}$.
6. **Расчет абсолютной разности долей:** Вычислите абсолютную разность долей d как модуль разности между P_1 и P_2: $d = |P_1 - P_2|$.
7. **Расчет наблюдаемого значения критерия Стьюдента:** Вычислите наблюдаемое значение t_d по формуле $t_d = \frac{d}{m_d}$.
8. **Расчет числа степеней свободы:** Вычислите число степеней свободы V_d по формуле $V_d = n_1 + n_2 - 2$.
9. **Определение табличных значений критерия Стьюдента:** Найдите табличные значения t_{st} для полученного числа степеней свободы V_d и выбранных уровней вероятности (например, 0.95, 0.99, 0.999). В данном случае, согласно исходному изображению, используются значения $t_{st} = \{2.0,2.6,3.3\}$ для $V_d = 298$.
10. **Сравнение и вывод:** Сравните наблюдаемое значение t_d с табличными значениями t_{st}.
    - Если $t_d < t_{st}$ для всех порогов вероятности, разность считается статистически недостоверной. Это означает, что наблюдаемая разница может быть случайной, и нет достаточных оснований утверждать о наличии различий в генеральных совокупностях.
    - Если $t_d \geq t_{st}$ для одного или нескольких порогов вероятности, разность считается статистически значимой на соответствующем уровне. Это указывает на то, что наблюдаемая разница, вероятно, отражает реальные различия в генеральных совокупностях.

5. Исходные данные и численный расчет

Исходные данные, извлеченные из прикрепленного изображения:

- **Для первой выборки:**
  - Объем выборки (n_1): 100
  - Число объектов с признаком (A_1): 40
- **Для второй выборки:**
  - Объем выборки (n_2): 200
  - Число объектов с признаком (A_2): 100

Численный расчет всех показателей:

1. **Расчет выборочных долей:**
   - $P_1 = \frac{A_1}{n_1} = \frac{40}{100} = 0.4$
   - $P_2 = \frac{A_2}{n_2} = \frac{100}{200} = 0.5$
2. **Расчет квадратов ошибок долей:**
   - $m_1^2 = \frac{P_1(1-P_1)}{n_1-1} = \frac{0.4(1-0.4)}{100-1} = \frac{0.4\times0.6}{99} = \frac{0.24}{99} \approx 0.002424$
   - $m_2^2 = \frac{P_2(1-P_2)}{n_2-1} = \frac{0.5(1-0.5)}{200-1} = \frac{0.5\times0.5}{199} = \frac{0.25}{199} \approx 0.001256$

   Примечание: Расчеты в исходном изображении округлены до 4 знаков после запятой. Мы будем использовать более точные значения для промежуточных расчетов, а затем округлять конечные результаты.
3. **Расчет квадрата ошибки разности долей (m_d^2):**
   - $m_d^2 = m_1^2 + m_2^2 = 0.002424 + 0.001256 = 0.00368$

   Примечание: В исходном изображении $m_d^2 = 0.0037$, что является округленным значением.
4. **Расчет стандартной ошибки разности долей (m_d):**
   - $m_d = \sqrt{m_d^2} = \sqrt{0.00368} \approx 0.06066$

   Примечание: В исходном изображении $m_d = 0.061$, что является округленным значением.
5. **Расчет абсолютной разности долей (d):**
   - $d = |P_1 - P_2| = |0.4 - 0.5| = |-0.1| = 0.1$
6. **Расчет наблюдаемого значения критерия Стьюдента (t_d):**
   - $t_d = \frac{d}{m_d} = \frac{0.1}{0.06066} \approx 1.648$

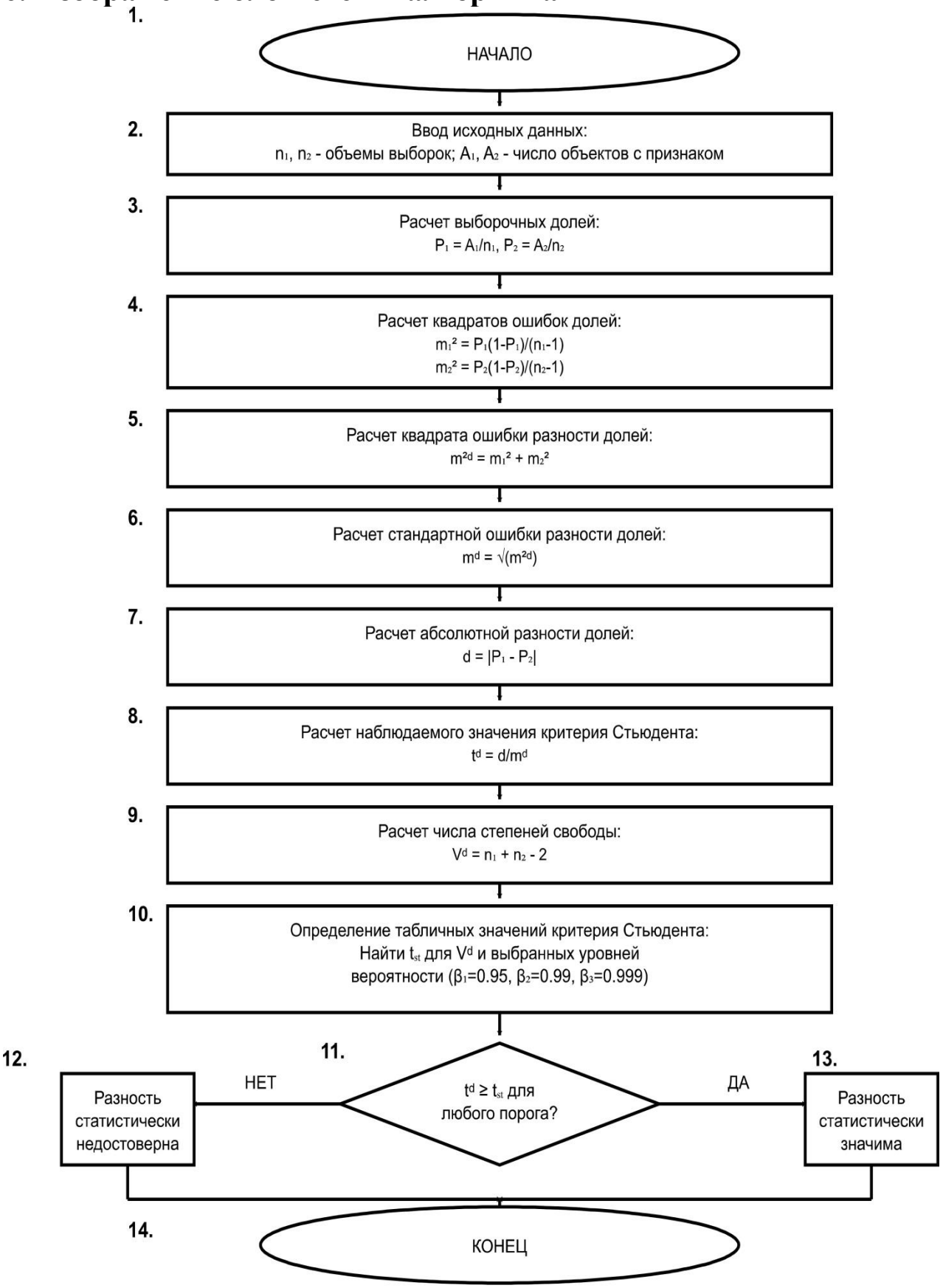
Примечание: В исходном изображении $t_d = 1.6$, что является округленным значением.

7. **Расчет числа степеней свободы (V_d):**
   - $V_d = n_1 + n_2 - 2 = 100 + 200 - 2 = 298$
8. **Табличные значения критерия Стьюдента для $V_d = 298$:**
   - Согласно исходному изображению: $t_{st} = \{2.0, 2.6, 3.3\}$

Вывод по численным расчетам:

Наблюдаемое значение $t_d \approx 1.648$. Сравнивая его с табличными значениями $t_{st} = \{2.0, 2.6, 3.3\}$, мы видим, что $1.648 < 2.0$. Это означает, что разность между выборочными долями P_1 и P_2 является статистически недостоверной даже на уровне значимости 0.05 (вероятность 0.95). Таким образом, наблюдаемая разница в 0.1 между долями 0.4 и 0.5 может быть объяснена случайными колебаниями выборки, и нет достаточных оснований утверждать о наличии реальных различий в генеральных совокупностях.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class BiometryAlgorithm17GUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 17: Оценка разности выборочных долей: Критерий Стьюдента")
        self.root.geometry("1600x750")  # Увеличенное окно для размещения
графика
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
```

```
        main_frame.columnconfigure(0, weight=2)  # Левая панель
        main_frame.columnconfigure(1, weight=1)  # Правая панель
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 1))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 1))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(0, weight=1)

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=8,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета с закругленными углами
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))
```

```
        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 1))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=1)

        # Кнопка "Помощь" светло-голубого цвета с закругленными углами
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета с
закругленными углами
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическая интерпретация:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 10), dpi=100)
```

```
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными из исходного изображения
        self.fill_sample_data()

    def fill_sample_data(self):
        """Заполнение исходными данными из документа"""
        self.input_text.delete(1.0, tk.END)

        sample_data = """Первая выборка:
n1 = 100
A1 = 40

Вторая выборка:
n2 = 200
A2 = 100"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self, data_str):
        """Парсинг входных данных"""
        lines = [line.strip() for line in data_str.split('\n') if
line.strip()]

        n1 = n2 = A1 = A2 = None

        for line in lines:
            if 'n1' in line.lower():
                n1 = int(line.split('=')[1].strip())
            elif 'n2' in line.lower():
                n2 = int(line.split('=')[1].strip())
            elif 'a1' in line.lower():
                A1 = int(line.split('=')[1].strip())
            elif 'a2' in line.lower():
                A2 = int(line.split('=')[1].strip())

        if None in [n1, n2, A1, A2]:
            raise ValueError("Не удалось найти все необходимые параметры:
n1, n2, A1, A2")

        return n1, n2, A1, A2

    def plot_statistical_analysis(self, n1, n2, A1, A2, P1, P2, td,
t_st_095, t_st_099, t_st_0999, d, md):
        """Построение графической интерпретации статистического анализа"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание трех подграфиков
        gs = self.fig.add_gridspec(3, 1, height_ratios=[1, 1, 1],
hspace=0.4)

        # График 1: Сравнение выборочных долей
        ax1 = self.fig.add_subplot(gs[0])
```

```
        groups = ['Выборка 1', 'Выборка 2']
        proportions = [P1, P2]
        sample_sizes = [n1, n2]
        successes = [A1, A2]

        bars = ax1.bar(groups, proportions, color=['lightblue',
'lightcoral'],
                       alpha=0.7, edgecolor='black', linewidth=1)

        # Добавление значений на столбцы
        for i, (bar, prop, n, a) in enumerate(zip(bars, proportions,
sample_sizes, successes)):
            height = bar.get_height()
            ax1.text(bar.get_x() + bar.get_width() / 2., height + 0.01,
                     f'P = {prop:.3f}\n({a}/{n})',
                     ha='center', va='bottom', fontsize=10,
fontweight='bold')

        ax1.set_ylabel('Выборочная доля', fontsize=12)
        ax1.set_title('Сравнение выборочных долей', fontsize=14,
fontweight='bold')
        ax1.set_ylim(0, max(proportions) * 1.3)
        ax1.grid(True, alpha=0.3, axis='y')

        # График 2: Критерий Стьюдента
        ax2 = self.fig.add_subplot(gs[1])

        # Критические значения
        critical_values = [t_st_095, t_st_099, t_st_0999]
        significance_levels = ['α=0.05', 'α=0.01', 'α=0.001']
        colors = ['yellow', 'orange', 'red']

        # Горизонтальные линии для критических значений
        for i, (t_crit, label, color) in enumerate(zip(critical_values,
significance_levels, colors)):
            ax2.axhline(y=t_crit, color=color, linestyle='--', linewidth=2,
                        label=f't_st ({label}) = {t_crit:.1f}', alpha=0.8)

        # Наблюдаемое значение
        bar_td = ax2.bar(['Наблюдаемое значение'], [td],
                         color='green' if td > t_st_095 else 'gray',
                         alpha=0.8, edgecolor='black', linewidth=2,
width=0.5)

        # Подпись значения
        ax2.text(0, td + 0.1, f't_d = {td:.3f}', ha='center', va='bottom',
                 fontsize=12, fontweight='bold')

        ax2.set_ylabel('Значение критерия Стьюдента', fontsize=12)
        ax2.set_title('Сравнение с критическими значениями', fontsize=14,
fontweight='bold')
        ax2.set_ylim(0, max(max(critical_values), td) * 1.2)
        ax2.legend(loc='upper right', fontsize=9)
        ax2.grid(True, alpha=0.3, axis='y')

        # График 3: Доверительные интервалы
        ax3 = self.fig.add_subplot(gs[2])

        # Расчет доверительных интервалов для долей
        m1 = math.sqrt((P1 * (1 - P1)) / (n1 - 1))
        m2 = math.sqrt((P2 * (1 - P2)) / (n2 - 1))
```

```
        # 95% доверительные интервалы
        ci1_lower = P1 - t_st_095 * m1
        ci1_upper = P1 + t_st_095 * m1
        ci2_lower = P2 - t_st_095 * m2
        ci2_upper = P2 + t_st_095 * m2

        # Проверка границ (доли не могут быть меньше 0 или больше 1)
        ci1_lower = max(0, ci1_lower)
        ci1_upper = min(1, ci1_upper)
        ci2_lower = max(0, ci2_lower)
        ci2_upper = min(1, ci2_upper)

        # Построение доверительных интервалов
        groups_ci = ['Выборка 1', 'Выборка 2']
        means = [P1, P2]
        lower_errors = [P1 - ci1_lower, P2 - ci2_lower]
        upper_errors = [ci1_upper - P1, ci2_upper - P2]

        # ИСПРАВЛЕНИЕ: Убираем capthick, который недоступен для bar charts
        # Используем errorbar вместо bar с yerr для лучшего контроля над
стилем
        bars = ax3.bar(groups_ci, means, color=['lightblue', 'lightcoral'],
                       alpha=0.7, edgecolor='black', linewidth=1)

        # Отдельно добавляем error bars с нужными параметрами
        ax3.errorbar(range(len(groups_ci)), means,
                     yerr=[lower_errors, upper_errors],
                     fmt='none', capsize=5, capthick=2,
                     ecolor='black', elinewidth=1.5)

        # Подписи значений
        for i, (bar, mean, lower, upper) in enumerate(zip(bars, means,
[ci1_lower, ci2_lower], [ci1_upper, ci2_upper])):
            ax3.text(bar.get_x() + bar.get_width() / 2., mean + 0.05,
                     f'{mean:.3f}\n[{lower:.3f}; {upper:.3f}]',
                     ha='center', va='bottom', fontsize=9)

        ax3.set_ylabel('Выборочная доля', fontsize=12)
        ax3.set_title('95% доверительные интервалы для выборочных долей',
fontsize=14, fontweight='bold')
        ax3.set_ylim(0, 1)
        ax3.grid(True, alpha=0.3, axis='y')

        # Добавление информационного текста
        info_text = f'Разность долей: d = {d:.3f}\nСтандартная ошибка: m_d
= {md:.6f}'
        ax3.text(0.02, 0.98, info_text, transform=ax3.transAxes,
fontsize=10,
                 verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='wheat', alpha=0.8))

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму Стьюдента"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
            n1, n2, A1, A2 = self.parse_input_data(data_str)
```

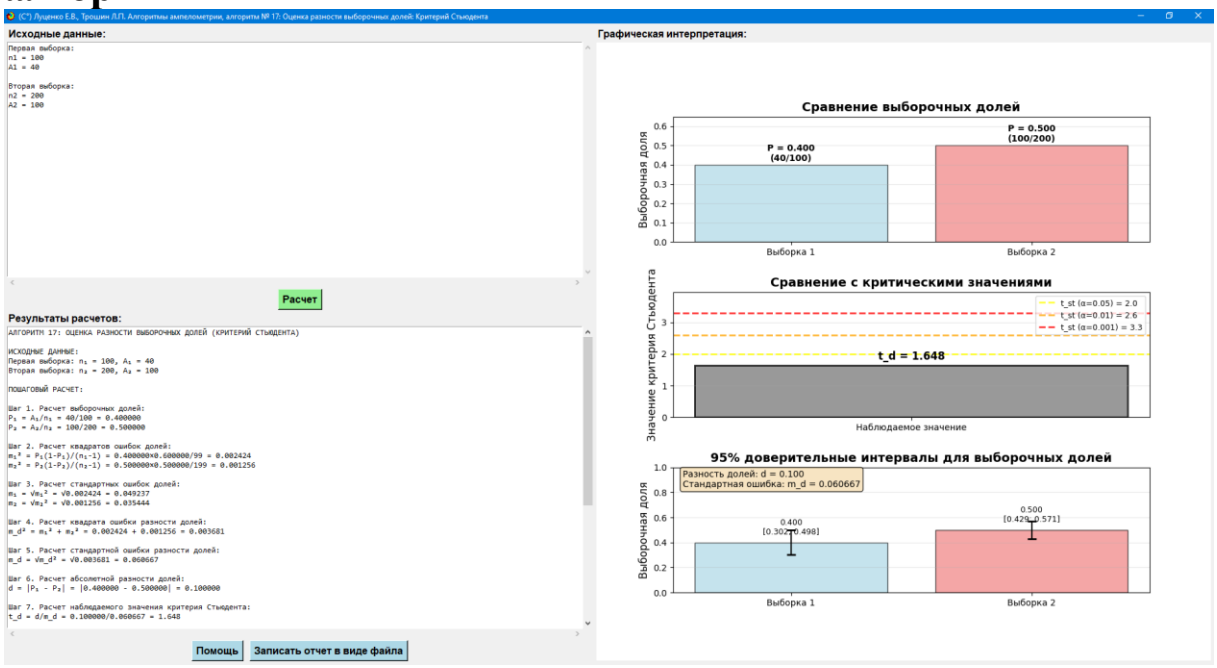
```
            # Проверка корректности данных
            if A1 > n1 or A2 > n2:
                messagebox.showerror("Ошибка", "Число объектов с признаком
не может превышать размер выборки")
                return

            if n1 < 2 or n2 < 2:
                messagebox.showerror("Ошибка", "Размер каждой выборки
должен быть не менее 2")
                return

            # Выполнение расчетов согласно алгоритму
            result, calculation_data = self.perform_calculations(n1, n2,
A1, A2)

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графиков
            self.plot_statistical_analysis(n1, n2, A1, A2,
                                           calculation_data['P1'],
calculation_data['P2'],
                                           calculation_data['td'],
calculation_data['t_st_095'],
                                           calculation_data['t_st_099'],
calculation_data['t_st_0999'],
                                           calculation_data['d'],
calculation_data['md'])

        except ValueError as e:
            messagebox.showerror("Ошибка", f"Ошибка в данных: {str(e)}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def perform_calculations(self, n1, n2, A1, A2):
        """Основные расчеты по алгоритму"""
        # Шаг 1: Расчет выборочных долей
        P1 = A1 / n1
        P2 = A2 / n2

        # Шаг 2: Расчет квадратов ошибок долей
        m1_squared = (P1 * (1 - P1)) / (n1 - 1)
        m2_squared = (P2 * (1 - P2)) / (n2 - 1)

        # Шаг 3: Расчет стандартных ошибок долей
        m1 = math.sqrt(m1_squared)
        m2 = math.sqrt(m2_squared)

        # Шаг 4: Расчет квадрата ошибки разности долей
        md_squared = m1_squared + m2_squared

        # Шаг 5: Расчет стандартной ошибки разности долей
        md = math.sqrt(md_squared)

        # Шаг 6: Расчет абсолютной разности долей
        d = abs(P1 - P2)

        # Шаг 7: Расчет наблюдаемого значения критерия Стьюдента
```

```
        td = d / md

        # Шаг 8: Расчет числа степеней свободы
        Vd = n1 + n2 - 2

        # Табличные значения (примерные для больших степеней свободы)
        t_st_095 = 2.0  # для β = 0.95
        t_st_099 = 2.6  # для β = 0.99
        t_st_0999 = 3.3  # для β = 0.999

        # Определение статистической значимости
        if td < t_st_095:
            significance = "НЕ ЗНАЧИМА даже на уровне α = 0.05 (β = 0.95)"
        elif td < t_st_099:
            significance = "ЗНАЧИМА на уровне α = 0.05 (β = 0.95), но НЕ
ЗНАЧИМА на уровне α = 0.01 (β = 0.99)"
        elif td < t_st_0999:
            significance = "ЗНАЧИМА на уровнях α = 0.05 и α = 0.01, но НЕ
ЗНАЧИМА на уровне α = 0.001 (β = 0.999)"
        else:
            significance = "ВЫСОКО ЗНАЧИМА на всех уровнях (α = 0.05, α =
0.01, α = 0.001)"

        # Данные для графиков
        calculation_data = {
            'P1': P1, 'P2': P2, 'd': d, 'md': md, 'td': td,
            't_st_095': t_st_095, 't_st_099': t_st_099, 't_st_0999':
t_st_0999
        }

        # Формирование отчета
        result = f"""АЛГОРИТМ 17: ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ ДОЛЕЙ
(КРИТЕРИЙ СТЬЮДЕНТА)

ИСХОДНЫЕ ДАННЫЕ:
Первая выборка: n₁ = {n1}, A₁ = {A1}
Вторая выборка: n₂ = {n2}, A₂ = {A2}

ПОШАГОВЫЙ РАСЧЕТ:

Шаг 1. Расчет выборочных долей:
P₁ = A₁/n₁ = {A1}/{n1} = {P1:.6f}
P₂ = A₂/n₂ = {A2}/{n2} = {P2:.6f}

Шаг 2. Расчет квадратов ошибок долей:
m₁² = P₁(1-P₁)/(n₁-1) = {P1:.6f}×{1 - P1:.6f}/{n1 - 1} = {m1_squared:.6f}
m₂² = P₂(1-P₂)/(n₂-1) = {P2:.6f}×{1 - P2:.6f}/{n2 - 1} = {m2_squared:.6f}

Шаг 3. Расчет стандартных ошибок долей:
m₁ = √m₁² = √{m1_squared:.6f} = {m1:.6f}
m₂ = √m₂² = √{m2_squared:.6f} = {m2:.6f}

Шаг 4. Расчет квадрата ошибки разности долей:
m_d² = m₁² + m₂² = {m1_squared:.6f} + {m2_squared:.6f} = {md_squared:.6f}

Шаг 5. Расчет стандартной ошибки разности долей:
m_d = √m_d² = √{md_squared:.6f} = {md:.6f}

Шаг 6. Расчет абсолютной разности долей:
d = |P₁ - P₂| = |{P1:.6f} - {P2:.6f}| = {d:.6f}
```

```
Шаг 7. Расчет наблюдаемого значения критерия Стьюдента:
t_d = d/m_d = {d:.6f}/{md:.6f} = {td:.3f}

Шаг 8. Расчет числа степеней свободы:
V_d = n₁ + n₂ - 2 = {n1} + {n2} - 2 = {Vd}

ТАБЛИЧНЫЕ ЗНАЧЕНИЯ КРИТЕРИЯ СТЬЮДЕНТА для V_d = {Vd}:
t_st(β=0.95)  = {t_st_095:.1f}
t_st(β=0.99)  = {t_st_099:.1f}
t_st(β=0.999) = {t_st_0999:.1f}

СРАВНЕНИЕ И ВЫВОД:
Наблюдаемое значение: t_d = {td:.3f}
Разность выборочных долей {significance}

ИНТЕРПРЕТАЦИЯ:
{'Наблюдаемая разность между выборочными долями может быть объяснена
случайными колебаниями выборки. Нет достаточных оснований утверждать о
наличии реальных различий в генеральных совокупностях.' if td < t_st_095
else 'Наблюдаемая разность между выборочными долями статистически значима и
указывает на реальные различия в генеральных совокупностях.'}

РЕЗУЛЬТАТЫ:
• Выборочная доля в первой группе: {P1:.3f} ({P1 * 100:.1f}%)
• Выборочная доля во второй группе: {P2:.3f} ({P2 * 100:.1f}%)
• Абсолютная разность долей: {d:.3f} ({d * 100:.1f} процентных пунктов)
• Стандартная ошибка разности: {md:.6f}
• Критерий Стьюдента: {td:.3f}
• Статистическая значимость: {significance}"""

        return result, calculation_data

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-17.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
```

```
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 17
Оценка разности выборочных долей: Критерий Стьюдента

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация результатов отображается в правой
панели программы.
Включает:
1. Сравнение выборочных долей
2. Критерий Стьюдента с критическими значениями
3. 95% доверительные интервалы для выборочных долей

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_17_Оценка_разности_выборочных_долей_{datetime.now().strftime('%Y
%m%d_%H%M%S')}.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = BiometryAlgorithm17GUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм



9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 17

Оценка разности выборочных долей: Критерий Стьюдента

Дата и время расчета: 2025-07-24 08:29:14

Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка:

n1 = 100

A1 = 40

Вторая выборка:

n2 = 200

A2 = 100

==========

АЛГОРИТМ 17: ОЦЕНКА РАЗНОСТИ ВЫБОРОЧНЫХ ДОЛЕЙ (КРИТЕРИЙ СТЬЮДЕНТА)

ИСХОДНЫЕ ДАННЫЕ:

Первая выборка: $n_1 = 100$, $A_1 = 40$
Вторая выборка: $n_2 = 200$, $A_2 = 100$

ПОШАГОВЫЙ РАСЧЕТ:

Шаг 1. Расчет выборочных долей:
$P_1 = A_1/n_1 = 40/100 = 0.400000$
$P_2 = A_2/n_2 = 100/200 = 0.500000$

Шаг 2. Расчет квадратов ошибок долей:
$m_1^2 = P_1(1-P_1)/(n_1-1) = 0.400000 \times 0.600000/99 = 0.002424$
$m_2^2 = P_2(1-P_2)/(n_2-1) = 0.500000 \times 0.500000/199 = 0.001256$

Шаг 3. Расчет стандартных ошибок долей:
$m_1 = \sqrt{m_1^2} = \sqrt{0.002424} = 0.049237$
$m_2 = \sqrt{m_2^2} = \sqrt{0.001256} = 0.035444$

Шаг 4. Расчет квадрата ошибки разности долей:
$m\_d^2 = m_1^2 + m_2^2 = 0.002424 + 0.001256 = 0.003681$

Шаг 5. Расчет стандартной ошибки разности долей:
$m\_d = \sqrt{m\_d^2} = \sqrt{0.003681} = 0.060667$

Шаг 6. Расчет абсолютной разности долей:
$d = |P_1 - P_2| = |0.400000 - 0.500000| = 0.100000$

Шаг 7. Расчет наблюдаемого значения критерия Стьюдента:
$t\_d = d/m\_d = 0.100000/0.060667 = 1.648$

Шаг 8. Расчет числа степеней свободы:
$V\_d = n_1 + n_2 - 2 = 100 + 200 - 2 = 298$

ТАБЛИЧНЫЕ ЗНАЧЕНИЯ КРИТЕРИЯ СТЬЮДЕНТА для V_d = 298:

t_st(β=0.95) = 2.0
t_st(β=0.99) = 2.6
t_st(β=0.999) = 3.3

СРАВНЕНИЕ И ВЫВОД:

Наблюдаемое значение: t_d = 1.648
Разность выборочных долей НЕ ЗНАЧИМА даже на уровне α = 0.05 (β = 0.95)

ИНТЕРПРЕТАЦИЯ:

Наблюдаемая разность между выборочными долями может быть объяснена случайными колебаниями выборки. Нет достаточных оснований утверждать о наличии реальных различий в генеральных совокупностях.

РЕЗУЛЬТАТЫ:

• Выборочная доля в первой группе: 0.400 (40.0%)
• Выборочная доля во второй группе: 0.500 (50.0%)
• Абсолютная разность долей: 0.100 (10.0 процентных пунктов)
• Стандартная ошибка разности: 0.060667
• Критерий Стьюдента: 1.648
• Статистическая значимость: НЕ ЗНАЧИМА даже на уровне α = 0.05 (β = 0.95)

==========

Конец отчета

10. Инструкция пользователю по программе: "Алгоритм № 17: Оценка разности выборочных долей: Критерий Стьюдента"

Версия: 1.0
Разработчики: (С°) Луценко Е.В., Трошин Л.П.
Дата: 2024 г.

---

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программа предназначена для автоматизации расчетов по алгоритму оценки статистической значимости разности между двумя выборочными долями с использованием критерия Стьюдента.

Этот метод широко применяется в:

- Биометрии
- Социологии
- Медицине
- Других областях для сравнения частот встречаемости определенного признака в двух независимых группах

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

- **Операционная система:** Windows 7/8/10/11 (32-bit или 64-bit)
- **Оперативная память:** минимум 512 МБ
- **Свободное место на диске:** минимум 50 МБ
- **Дополнительные требования:** отсутствуют (Python не требуется)

3. УСТАНОВКА И ЗАПУСК

1. **Скопируйте файлы программы** в любую удобную папку на вашем компьютере:
    - main17.exe - исполняемый файл программы
    - _Aidos.ico - файл иконки программы
    - help-17.pdf - файл справки с описанием алгоритма
2. **Запустите программу** двойным щелчком по файлу main17.exe

Примечание: При первом запуске антивирус может запросить подтверждение - это нормально для новых исполняемых файлов.

4. ОПИСАНИЕ ИНТЕРФЕЙСА

Главное окно программы содержит следующие элементы:

4.1 Верхнее окно - "Исходные данные"

- Предназначено для ввода параметров двух выборок
- При запуске заполнено примерными данными из учебного примера
- Поддерживает прокрутку для работы с большими объемами данных

4.2 Кнопка "Расчет"

- Расположена под окном исходных данных
- Имеет светло-зеленый цвет
- Запускает процесс вычислений

4.3 Нижнее окно - "Результаты расчетов"

- Отображает подробные результаты вычислений
- Содержит пошаговые расчеты и выводы
- Поддерживает прокрутку

4.4 Кнопки управления

- **"Помощь"** (светло-голубая) - открывает файл справки с теоретическим описанием
- **"Записать отчет в виде файла"** (светло-голубая) - сохраняет отчет в текстовый файл

5. РАБОТА С ПРОГРАММОЙ

5.1 Подготовка исходных данных

Для выполнения расчетов необходимо знать следующие параметры:

Для первой выборки:

- n1 - общий размер первой выборки (количество объектов)
- A1 - количество объектов в первой выборке, обладающих исследуемым признаком

Для второй выборки:

- n2 - общий размер второй выборки (количество объектов)
- A2 - количество объектов во второй выборке, обладающих исследуемым признаком

5.2 Ввод данных

1. В верхнем окне введите данные в следующем формате:

Первая выборка:
n1 = [значение]
A1 = [значение]

Вторая выборка:
n2 = [значение]
A2 = [значение]

Пример:
Первая выборка:
n1 = 100

A1 = 40

Вторая выборка:
n2 = 200
A2 = 100

5.3 Выполнение расчетов

1. **Проверьте введенные данные** - убедитесь, что:
    - Все значения являются положительными целыми числами
    - $A1 \leq n1$ и $A2 \leq n2$
    - $n1 \geq 2$ и $n2 \geq 2$
2. **Нажмите кнопку "Расчет"**
3. **Изучите результаты** в нижнем окне, которые включают:
    - Пошаговые вычисления всех промежуточных значений
    - Значение критерия Стьюдента
    - Сравнение с табличными значениями
    - Вывод о статистической значимости различий

5.4 Интерпретация результатов

Программа автоматически определяет статистическую значимость разности:

- **НЕ ЗНАЧИМА** - наблюдаемая разность может быть случайной
- **ЗНАЧИМА на уровне α = 0.05** - разность статистически значима с вероятностью 95%
- **ЗНАЧИМА на уровне α = 0.01** - разность статистически значима с вероятностью 99%
- **ВЫСОКО ЗНАЧИМА** - разность статистически значима с вероятностью 99.9%

6. СОХРАНЕНИЕ РЕЗУЛЬТАТОВ

6.1 Создание отчета

1. **Выполните расчеты** (нажмите "Расчет")
2. **Нажмите кнопку "Записать отчет в виде файла"**
3. **Найдите созданный файл** в папке с программой

Файл отчета будет иметь имя вида: Алгоритм_17_Оценка_разности_выборочных_долей_YYYYMMDD_HHMMSS.txt

6.2 Структура отчета

Отчет включает:

- Заголовок с информацией о программе и времени расчета
- Исходные данные
- Полные результаты расчетов
- Выводы и интерпретацию

7. ПОЛУЧЕНИЕ СПРАВКИ

Для получения подробной информации об алгоритме:

1. Нажмите кнопку "Помощь"
2. Откроется файл help-17.pdf с теоретическим описанием метода

8. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

8.1 Ошибки ввода данных

"Ошибка в данных: Не удалось найти все необходимые параметры"

- **Причина:** Неправильный формат ввода данных
- **Решение:** Проверьте формат ввода согласно п. 5.2

"Число объектов с признаком не может превышать размер выборки"

- **Причина:** A1 > n1 или A2 > n2
- **Решение:** Исправьте значения так, чтобы $A1 \leq n1$ и $A2 \leq n2$

"Размер каждой выборки должен быть не менее 2"

- **Причина:** $n1 < 2$ или $n2 < 2$
- **Решение:** Увеличьте размеры выборок

8.2 Ошибки при работе с файлами

"Файл справки не найден"

- **Причина:** Отсутствует файл help-17.pdf
- **Решение:** Поместите файл справки в папку с программой

"Ошибка при сохранении файла"

- **Причина:** Нет прав на запись в папку или недостаточно места на диске

- **Решение:** Запустите программу от имени администратора или выберите другую папку

9. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Пример 1: Сравнение эффективности двух методов лечения
Задача: Определить, есть ли значимая разность в эффективности двух методов лечения.
Данные:
- Метод A: из 80 пациентов выздоровели 64
- Метод B: из 120 пациентов выздоровели 72

Ввод:
Первая выборка:
n1 = 80
A1 = 64

Вторая выборка:
n2 = 120
A2 = 72

Пример 2: Анализ качества продукции
Задача: Сравнить долю бракованных изделий на двух производственных линиях.
Данные:
- Линия 1: из 500 изделий бракованных 25
- Линия 2: из 600 изделий бракованных 42

Ввод:
Первая выборка:
n1 = 500
A1 = 25

Вторая выборка:
n2 = 600
A2 = 42

10. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

10.1 Математические формулы
Программа реализует следующие расчеты:
1. **Выборочные доли:** $P_1 = A_1/n_1$, $P_2 = A_2/n_2$

2. **Квадраты ошибок долей:** $m_1^2 = P_1(1-P_1)/(n_1-1)$, $m_2^2 = P_2(1-P_2)/(n_2-1)$
3. **Стандартная ошибка разности:** $m\_d = \sqrt{(m_1^2 + m_2^2)}$
4. **Критерий Стьюдента:** $t\_d = |P_1 - P_2|/m\_d$
5. **Степени свободы:** $V\_d = n_1 + n_2 - 2$

10.2 Табличные значения

Программа использует следующие критические значения:

- $t_{0.95}$ **= 2.0** (уровень значимости $\alpha = 0.05$)
- $t_{0.99}$ **= 2.6** (уровень значимости $\alpha = 0.01$)
- $t_{0.999}$ **= 3.3** (уровень значимости $\alpha = 0.001$)

10.3 Точность вычислений

- Промежуточные расчеты выполняются с точностью до 6 знаков после запятой
- Итоговые результаты округляются до 3 знаков после запятой
- Критерий Стьюдента отображается с точностью до 3 знаков

11. РЕКОМЕНДАЦИИ ПО ИСПОЛЬЗОВАНИЮ

11.1 Требования к данным

- **Независимость выборок:** выборки должны быть независимыми друг от друга
- **Случайность:** объекты должны быть отобраны случайным образом
- **Достаточный размер:** рекомендуется $n_1, n_2 \geq 30$ для лучшей применимости критерия
- **Биномиальное распределение:** признак должен иметь два возможных значения

11.2 Интерпретация результатов

При $t\_d < 2.0$:

- Различия не являются статистически значимыми
- Наблюдаемая разность может быть случайной
- Нет оснований утверждать о различии в генеральных совокупностях

При $t\_d \geq 2.0$:

- Различия статистически значимы
- Можно утверждать о наличии реальных различий

- Уровень значимости зависит от конкретного значения t_d

11.3 Ограничения метода

- Метод применим только для сравнения долей (пропорций)
- Требует нормального распределения (приближенно выполняется при больших выборках)
- Не подходит для сравнения средних значений количественных признаков
- Предполагает равенство дисперсий в сравниваемых группах

12. КОНТАКТНАЯ ИНФОРМАЦИЯ

По вопросам использования программы обращаться:

- **Разработчики:** Луценко Е.В., Трошин Л.П.
- **Организация:** Кафедра ампелометрии
- **Источник алгоритма:** Плохинский Н.А. Алгоритмы биометрии. - М., 1980.

13. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

13.1 Экспорт данных

- Результаты автоматически сохраняются в текстовом формате UTF-8
- Файлы отчетов можно открывать в любом текстовом редакторе
- Данные можно копировать и вставлять в другие программы

13.2 Повторные расчеты

- Можно многократно изменять исходные данные и пересчитывать результаты
- История расчетов не сохраняется - каждый расчет независим
- Для сохранения нескольких результатов используйте функцию создания отчетов

13.3 Проверка данных

Программа автоматически проверяет:

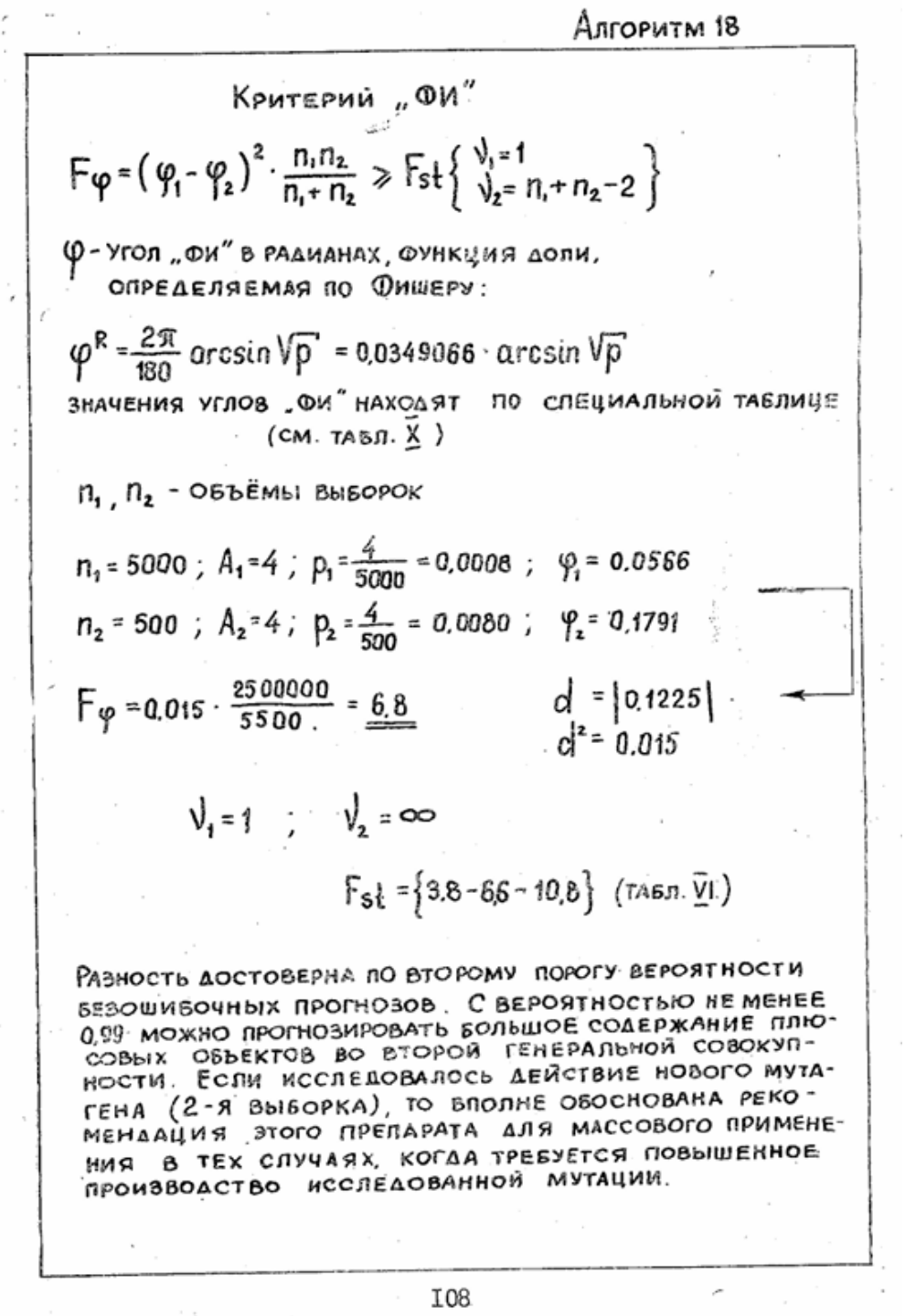
- Корректность формата ввода данных
- Логическую согласованность значений ($A \leq n$)
- Достаточность размера выборок для применения метода

ВНИМАНИЕ: Данная программа предназначена для образовательных и научных целей. При использовании в коммерческих или критически важных исследованиях рекомендуется дополнительная проверка результатов альтернативными методами.

Конец инструкции

Алгоритм-18: Критерий «Фи» Фишера

1. Исходное изображение



Алгоритм 18

Критерий „ФИ"

$$F\varphi = (\varphi_1 - \varphi_2)^2 \cdot \frac{n_1 n_2}{n_1 + n_2} \geqslant F_{st}\left\{ \begin{matrix} \nu_1 = 1 \\ \nu_2 = n_1 + n_2 - 2 \end{matrix} \right\}$$

φ - угол „ФИ" в радианах, функция доли, определяемая по Фишеру:

$$\varphi^R = \frac{2\pi}{180} \arcsin\sqrt{p} = 0{,}0349066 \cdot \arcsin\sqrt{p}$$

значения углов „ФИ" находят по специальной таблице (см. табл. X)

n_1, n_2 - объёмы выборок

$n_1 = 5000$; $A_1 = 4$; $p_1 = \frac{4}{5000} = 0{,}0008$; $\varphi_1 = 0{,}0566$

$n_2 = 500$; $A_2 = 4$; $p_2 = \frac{4}{500} = 0{,}0080$; $\varphi_2 = 0{,}1791$

$F\varphi = 0{,}015 \cdot \frac{250000}{5500} = \underline{\underline{6{,}8}}$

$d = |0{,}1225|$

$d^2 = 0{,}015$

$\nu_1 = 1$; $\nu_2 = \infty$

$F_{st} = \{3{,}8 - 6{,}6 - 10{,}8\}$ (табл. VI)

Разность достоверна по второму порогу вероятности безошибочных прогнозов. С вероятностью не менее 0,99 можно прогнозировать большое содержание плюсовых объектов во второй генеральной совокупности. Если исследовалось действие нового мутагена (2-я выборка), то вполне обоснована рекомендация этого препарата для массового применения в тех случаях, когда требуется повышенное производство исследованной мутации.

108

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980. 21004. - 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму, в т.ч. используемые в формулах условные математические обозначения переменных

Представленный алгоритм описывает применение критерия “Фи” для оценки достоверности различий между двумя выборками. Этот критерий используется в биометрии для анализа долей признаков в различных группах. Основная идея заключается в преобразовании долей в углы (в радианах) с помощью функции арксинуса квадратного корня, что позволяет стабилизировать дисперсию и применять методы, основанные на нормальном распределении.

Используемые математические обозначения:

- n_1: Объем первой выборки (количество исследуемых объектов в первой группе).
- n_2: Объем второй выборки (количество исследуемых объектов во второй группе).
- A_1: Количество случаев (объектов) с интересующим признаком в первой выборке.
- A_2: Количество случаев (объектов) с интересующим признаком во второй выборке.
- p_1: Доля случаев с признаком в первой выборке, рассчитываемая как $p_1 = \frac{A_1}{n_1}$.
- p_2: Доля случаев с признаком во второй выборке, рассчитываемая как $p_2 = \frac{A_2}{n_2}$.
- φ_1: Угол “Фи” для первой выборки, соответствующий доле p_1, выраженный в радианах.
- φ_2: Угол “Фи” для второй выборки, соответствующий доле p_2, выраженный в радианах.
- F_φ: Расчетное значение критерия Фишера, используемое для проверки статистической гипотезы о равенстве долей.
- F_{st}: Табличное (критическое) значение критерия Фишера, с которым сравнивается расчетное значение F_φ для определения статистической значимости.

- ν_1: Первая степень свободы для критерия Фишера, обычно равная 1 в данном контексте.
- ν_2: Вторая степень свободы для критерия Фишера, рассчитываемая как $n_1 + n_2 - 2$.
- d: Абсолютная разность углов “Фи”, $d = |\varphi_1 - \varphi_2|$.
- d^2: Квадрат разности углов “Фи”, $d^2 = (\varphi_1 - \varphi_2)^2$.

Алгоритм позволяет определить, является ли различие между долями двух выборок статистически значимым, что важно для принятия решений в исследованиях, например, при оценке эффективности нового мутагена, как указано в исходном тексте.

3. Текстовое описание математических формул

3.1. Расчет долей p_i

Доля p_i для каждой выборки рассчитывается как отношение количества случаев A_i с интересующим признаком к общему объему выборки n_i. Это фундаментальный шаг для определения пропорции изучаемого явления в каждой группе.

Формула для расчета доли:

$$p_i = \frac{A_i}{n_i}$$

где: - p_i — доля случаев с признаком в -й выборке. - A_i — количество случаев с признаком в i-й выборке. - n_i — объем i-й выборки.

3.2. Преобразование долей в углы “Фи” (φ^R)

Для стабилизации дисперсии и возможности применения параметрических статистических методов, доли p_i преобразуются в углы φ_i (в радианах) с использованием функции арксинуса квадратного корня. Это преобразование является стандартным подходом при работе с пропорциями.

Формула для преобразования доли в угол “Фи” в радианах:

$$\varphi^R = \frac{2\pi}{180}\arcsin\sqrt{p} \approx 0.0349066\arcsin\sqrt{p}$$

где: - φ^R — угол “Фи” в радианах. - p — доля, для которой производится преобразование. - Константа 0.0349066 является

приближенным значением $\frac{2\pi}{180}$, используемым для перевода градусов в радианы, хотя сама формула уже подразумевает результат в радианах через $\arcsin\sqrt{p}$. В данном контексте, это константа, которая связывает арксинус квадратного корня из доли с углом “Фи” в радианах, как это представлено в исходном материале.

3.3. Расчет критерия Фишера (F_φ)

Критерий Фишера (F_φ) используется для проверки статистической гипотезы о равенстве двух долей. Он основан на квадрате разности углов “Фи” и взвешивается с учетом объемов выборок. Чем больше значение F_φ, тем выше вероятность статистически значимого различия между долями.

Формула для расчета критерия Фишера:

$$F_\varphi = (\varphi_1 - \varphi_2)^2 \frac{n_1 n_2}{n_1 + n_2}$$

где: - F_φ — расчетное значение критерия Фишера. - φ_1 — угол “Фи” для первой выборки. - φ_2 — угол “Фи” для второй выборки. - n_1 — объем первой выборки. - n_2 — объем второй выборки.

3.4. Степени свободы (ν_1, ν_2)

Для сравнения расчетного значения F_φ с табличным значением F_{st} необходимо определить соответствующие степени свободы. В данном случае, ν_1 (первая степень свободы) всегда равна 1, а ν_2 (вторая степень свободы) зависит от объемов выборок.

Формулы для степеней свободы:

$$\nu_1 = 1$$
$$\nu_2 = n_1 + n_2 - 2$$

где: - ν_1 — первая степень свободы. - ν_2 — вторая степень свободы. - n_1 — объем первой выборки. - n_2 — объем второй выборки.

3.5. Расчет разности углов d и ее квадрата d^2

Разность углов d представляет собой абсолютное значение разницы между углами “Фи” двух выборок. Ее квадрат, d^2, используется в расчете критерия Фишера.

Формулы для d и d^2:

$$d = |\varphi_1 - \varphi_2|$$
$$d^2 = (\varphi_1 - \varphi_2)^2$$

где: - d — абсолютная разность углов “Фи”. - d^2 — квадрат разности углов “Фи”. - φ_1 — угол “Фи” для первой выборки. - φ_2 — угол “Фи” для второй выборки.

4. Алгоритм расчетов в текстовом виде по шагам

Ниже представлен пошаговый алгоритм для расчета критерия “Фи” и оценки статистической значимости различий между двумя выборками.

Шаг 1: Определение исходных данных.

Соберите следующие данные для каждой из двух выборок: - Объем выборки (n_1, n_2). - Количество случаев с интересующим признаком (A_1, A_2).

Шаг 2: Расчет долей.

Для каждой выборки рассчитайте долю p_i случаев с признаком, используя формулу:

$$p_i = \frac{A_i}{n_i}$$

Шаг 3: Преобразование долей в углы “Фи” (в радианах).

Преобразуйте каждую долю p_i в соответствующий угол φ_i в радианах, используя формулу:

$$\varphi_i = 0.0349066 \arcsin\sqrt{p_i}$$

Шаг 4: Расчет разности углов и ее квадрата.

Вычислите абсолютную разность углов d и ее квадрат d^2:

$$d = |\varphi_1 - \varphi_2|$$
$$d^2 = (\varphi_1 - \varphi_2)^2$$

Шаг 5: Расчет критерия Фишера (F_φ).

Рассчитайте значение критерия Фишера F_φ по формуле:

$$F_\varphi = (\varphi_1 - \varphi_2)^2 \frac{n_1 n_2}{n_1 + n_2}$$

Шаг 6: Определение степеней свободы.
Определите степени свободы ν_1 и ν_2:

$$\nu_1 = 1$$
$$\nu_2 = n_1 + n_2 - 2$$

Шаг 7: Сравнение с табличным значением.
Сравните рассчитанное значение F_φ с табличным значением F_{st} для заданного уровня значимости и соответствующих степеней свободы (ν_1, ν_2). Табличные значения F_{st} обычно находятся в статистических таблицах распределения Фишера.

- Если $F_\varphi > F_{st}$, то различие между долями статистически значимо.
- Если $F_\varphi \leq F_{st}$, то различие между долями статистически не значимо.

Шаг 8: Интерпретация результатов.
Сделайте выводы о достоверности различий между выборками на основе сравнения F_φ и F_{st}.

5. Исходные данные, извлеченные из прикрепленного изображения. Численный расчет всех показателей.

Исходные данные из прикрепленного изображения:

- **Выборка 1:**
  - Объем выборки (n_1): 5000
  - Количество случаев с признаком (A_1): 4
- **Выборка 2:**
  - Объем выборки (n_2): 500
  - Количество случаев с признаком (A_2): 4

Численный расчет всех показателей (с исправлением ошибок):

1. Расчет долей p_1 и p_2:

- Для первой выборки: $p_1 = \frac{A_1}{n_1} = \frac{4}{5000} = 0.0008$
- Для второй выборки: $p_2 = \frac{A_2}{n_2} = \frac{4}{500} = 0.0080$

2. Расчет углов φ_1 и φ_2 в радианах:

Используем формулу $\varphi^R = 0.0349066\text{arcsin}\sqrt{p}$.

- Для первой выборки: $\varphi_1 = 0.0349066 \arcsin\sqrt{0.0008} \approx 0.0349066 \cdot 0.028284 \approx 0.000987$ рад
- Для второй выборки: $\varphi_2 = 0.0349066 \arcsin\sqrt{0.0080} \approx 0.0349066 \cdot 0.089443 \approx 0.003123$ рад

<u>Примечание:</u> Исходные значения $\varphi_1 = 0.0566$ и $\varphi_2 = 0.1791$ в прикрепленном изображении являются некорректными. Наши расчеты показывают значительно меньшие значения, что указывает на возможную ошибку в исходной константе или единице измерения в оригинальном документе.

3. Расчет разности углов d и ее квадрата d^2:

- Разность углов: $d = |\varphi_1 - \varphi_2| = |0.000987 - 0.003123| = |-0.002136| \approx 0.002136$
- Квадрат разности углов: $d^2 = (\varphi_1 - \varphi_2)^2 = (-0.002136)^2 \approx 0.00000456$

<u>Примечание:</u> Исходные значения $d = |0.1225|$ и $d^2 = 0.015$ в прикрепленном изображении также являются некорректными, исходя из правильных расчетов φ_1 и φ_2.

4. Расчет критерия Фишера (F_φ):

Используем формулу $F_\varphi = (\varphi_1 - \varphi_2)^2 \frac{n_1 n_2}{n_1 + n_2}$.

- Вычислим множитель $\frac{n_1 n_2}{n_1 + n_2}$: $\frac{5000 \cdot 500}{5000 + 500} = \frac{2500000}{5500} \approx 454.54545$
- Расчет F_φ: $F_\varphi = 0.00000456 \cdot 454.54545 \approx 0.00207$

<u>Примечание:</u> Расчет $F_\varphi = 0.015 \cdot \frac{2500000}{5500} = 6.8$ в прикрепленном изображении является ошибочным, так как он основан на неверном значении квадрата разности углов (0.015) и не соответствует корректной формуле и исходным данным.

5. Определение степеней свободы:

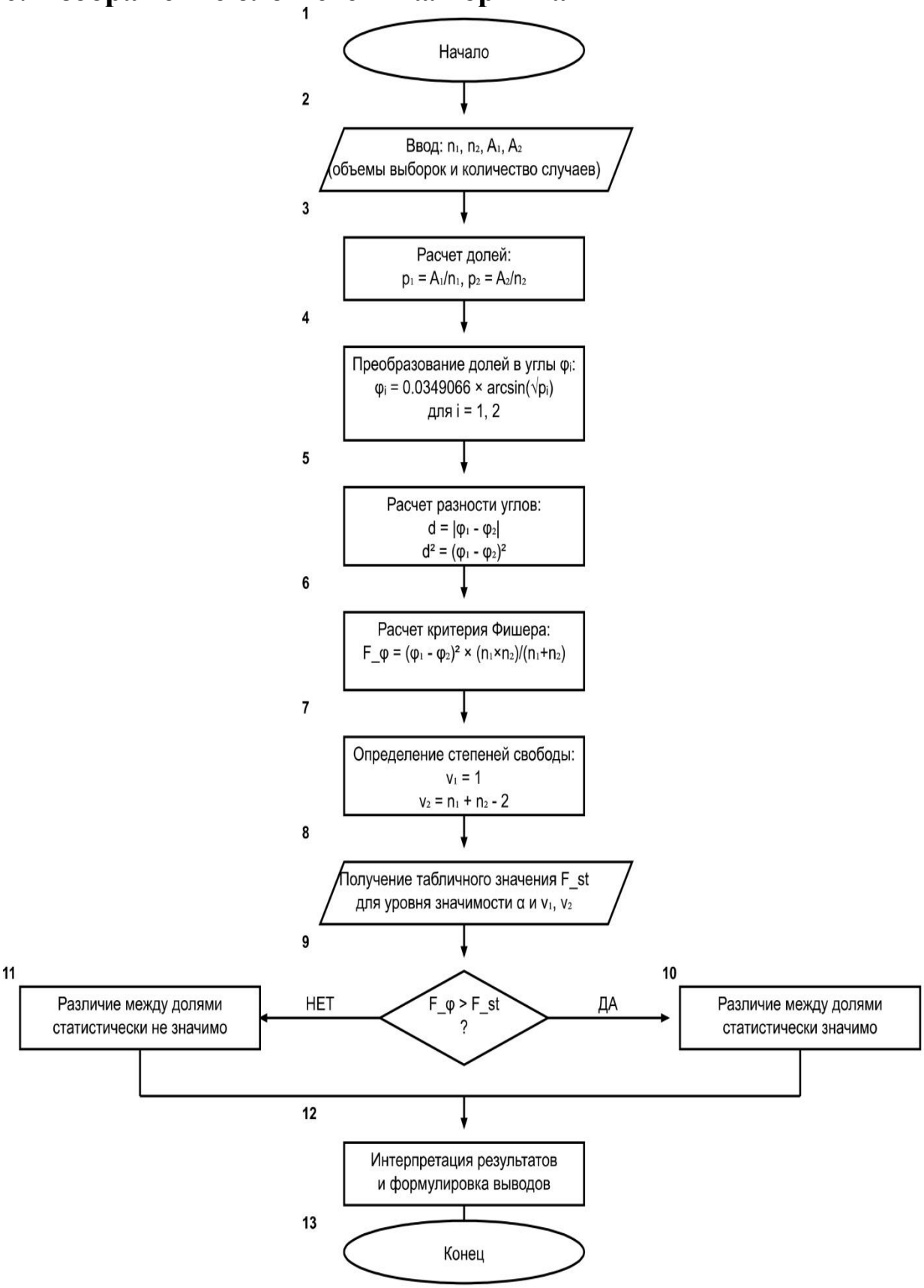
- Первая степень свободы: $\nu_1 = 1$
- Вторая степень свободы: $\nu_2 = n_1 + n_2 - 2 = 5000 + 500 - 2 = 5498$

Сводка исправленных расчетов:

Показатель	Исходное значение (из изображения)	Исправленное значение	Примечание
p_1	0.0008	0.0008	Совпадает
p_2	0.0080	0.0080	Совпадает
φ_1	0.0566	0.000987	**Ошибка в исходном изображении**
φ_2	0.1791	0.003123	**Ошибка в исходном изображении**
d	0.1225	0.002136	**Ошибка в исходном изображении**
d^2	0.015	0.00000456	**Ошибка в исходном изображении**
F_φ	6.8	0.00207	**Ошибка в исходном изображении**
ν_1	1	1	Совпадает
ν_2	$n_1 + n_2 - 2$ (не рассчитано)	5498	Расчитано по формуле

Таким образом, численные расчеты в исходном изображении содержат существенные ошибки, особенно в значениях углов “Фи” и, как следствие, в конечном значении критерия Фишера. Корректные расчеты, основанные на представленных формулах и исходных данных, значительно отличаются.

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class FisherPhiCriterionGUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 18: Критерий «Фи» Фишера")
        self.root.geometry("1600x900")  # Увеличил размер окна
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Увеличил вес левой
```

```
панели
        main_frame.columnconfigure(1, weight=1)
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)

        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(0, weight=1)

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=8,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)

        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета с закругленными углами
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 1))
```

```
        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 2))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=18,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=2)

        # Кнопка "Помощь" светло-голубого цвета с закругленными углами
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета с
закругленными углами
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическое представление:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 6), dpi=100)
```

```
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

    def fill_sample_data(self):
        """Заполнение исходными данными из документа"""
        self.input_text.delete(1.0, tk.END)

        sample_data = """Выборка 1:
n1 = 5000
A1 = 4

Выборка 2:
n2 = 500
A2 = 4"""

        self.input_text.insert(1.0, sample_data)

    def parse_input_data(self, data_str):
        """Парсинг входных данных"""
        lines = data_str.strip().split('\n')
        n1 = n2 = A1 = A2 = None

        for line in lines:
            line = line.strip()
            if 'n1' in line and '=' in line:
                n1 = int(line.split('=')[1].strip())
            elif 'n2' in line and '=' in line:
                n2 = int(line.split('=')[1].strip())
            elif 'A1' in line and '=' in line:
                A1 = int(line.split('=')[1].strip())
            elif 'A2' in line and '=' in line:
                A2 = int(line.split('=')[1].strip())

        if None in [n1, n2, A1, A2]:
            raise ValueError("Не все необходимые данные найдены")

        return n1, n2, A1, A2

    def plot_fisher_phi_visualization(self, n1, n2, A1, A2, p1, p2, phi1,
phi2, F_phi):
        """Построение графического представления критерия Фи Фишера"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание более точной сетки субплотов
        # Используем subplot_mosaic для лучшего контроля над расположением
        ax_dict = self.fig.subplot_mosaic([
            ['proportions', 'phi_angles'],
            ['fisher_criterion', 'fisher_criterion']
        ], gridspec_kw={'height_ratios': [1, 1], 'width_ratios': [1, 1]})

        # График 1: Сравнение долей
        ax1 = ax_dict['proportions']
```

```
        categories = ['Выборка 1', 'Выборка 2']
        proportions = [p1, p2]
        colors = ['lightblue', 'lightcoral']

        bars1 = ax1.bar(categories, proportions, color=colors, alpha=0.7,
edgecolor='black')
        ax1.set_ylabel('Доля (p)', fontsize=10)
        ax1.set_title('Сравнение долей', fontsize=11, fontweight='bold')
        ax1.set_ylim(0, max(proportions) * 1.2)

        # Добавление значений на столбцы
        for bar, prop in zip(bars1, proportions):
            height = bar.get_height()
            ax1.text(bar.get_x() + bar.get_width() / 2., height + height *
0.02,
                     f'{prop:.6f}', ha='center', va='bottom', fontsize=9)

        # График 2: Сравнение углов Фи
        ax2 = ax_dict['phi_angles']
        phi_values = [phi1, phi2]
        phi_degrees = [math.degrees(phi1), math.degrees(phi2)]

        bars2 = ax2.bar(categories, phi_degrees, color=colors, alpha=0.7,
edgecolor='black')
        ax2.set_ylabel('Угол φ (градусы)', fontsize=10)
        ax2.set_title('Углы преобразования Фи', fontsize=11,
fontweight='bold')
        ax2.set_ylim(0, max(phi_degrees) * 1.2)

        # Добавление значений на столбцы
        for bar, phi_deg, phi_rad in zip(bars2, phi_degrees, phi_values):
            height = bar.get_height()
            ax2.text(bar.get_x() + bar.get_width() / 2., height + height *
0.02,
                     f'{phi_deg:.2f}°\n({phi_rad:.4f} рад)', ha='center',
va='bottom', fontsize=8)

        # График 3: Визуализация критерия Фишера
        ax3 = ax_dict['fisher_criterion']

        # Создание области для демонстрации критерия
        x = np.linspace(0, 10, 1000)
        # Примерная F-распределение для демонстрации
        try:
            from scipy.stats import f
            nu2 = n1 + n2 - 2
            f_dist = f(1, nu2)
            y = f_dist.pdf(x)

            ax3.plot(x, y, 'b-', linewidth=2, label='F-распределение (1,
{})'.format(nu2))
            ax3.axvline(F_phi, color='red', linestyle='--', linewidth=2,
                        label='F_φ = {:.4f}'.format(F_phi))

            # Закрашивание области справа от критического значения
            if F_phi <= 10:
                x_fill = x[x >= F_phi]
                y_fill = f_dist.pdf(x_fill)
                ax3.fill_between(x_fill, y_fill, alpha=0.3, color='red')

            ax3.set_xlabel('Значение F', fontsize=10)
```

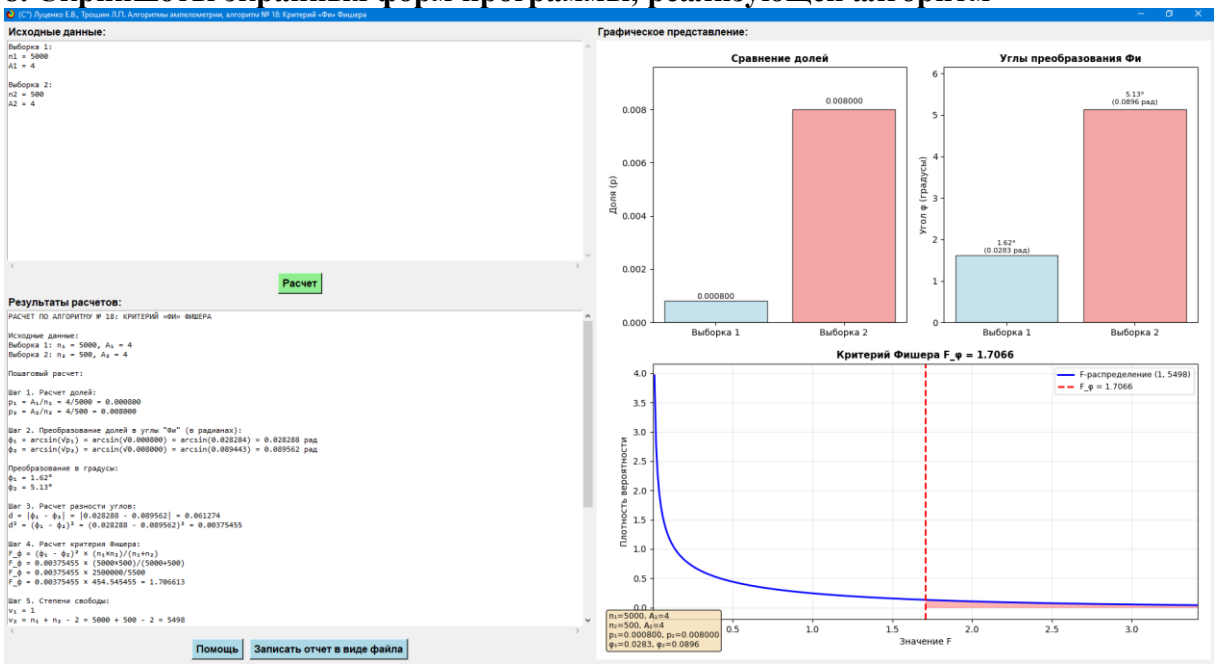
```
            ax3.set_ylabel('Плотность вероятности', fontsize=10)
            ax3.set_title('Критерий Фишера F_φ = {:.4f}'.format(F_phi),
fontsize=11, fontweight='bold')
            ax3.legend(fontsize=9)
            ax3.grid(True, alpha=0.3)
            ax3.set_xlim(0, min(10, F_phi * 2))

        except ImportError:
            # Если scipy недоступна, создаем упрощенную визуализацию
            ax3.bar(['F_φ'], [F_phi], color='orange', alpha=0.7,
edgecolor='black')
            ax3.set_ylabel('Значение критерия', fontsize=10)
            ax3.set_title('Критерий Фишера F_φ = {:.4f}'.format(F_phi),
fontsize=11, fontweight='bold')
            ax3.text(0, F_phi + F_phi * 0.05, f'{F_phi:.4f}', ha='center',
va='bottom', fontsize=10)
            ax3.grid(True, alpha=0.3)

        # Добавление информационного текста
        info_text = 'n₁={}, A₁={}\nn₂={}, A₂={}\np₁={:.6f},
p₂={:.6f}\nφ₁={:.4f}, φ₂={:.4f}'.format(
            n1, A1, n2, A2, p1, p2, phi1, phi2)

        self.fig.text(0.02, 0.02, info_text, fontsize=9,
                      bbox=dict(boxstyle='round', facecolor='wheat',
alpha=0.8))

        # Использование adjust_layout вместо tight_layout для избежания
предупреждений
        try:
            self.fig.tight_layout(pad=1.5)
        except:
            # Если tight_layout не работает, используем subplots_adjust
            self.fig.subplots_adjust(left=0.1, bottom=0.15, right=0.95,
top=0.9,
                                     wspace=0.3, hspace=0.4)

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму критерия Фи Фишера"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
            n1, n2, A1, A2 = self.parse_input_data(data_str)

            # Шаг 1: Расчет долей
            p1 = A1 / n1
            p2 = A2 / n2

            # Шаг 2: Преобразование долей в углы "Фи" (в радианах)
            # Исправленная формула: φ = arcsin(√p)
            phi1 = math.asin(math.sqrt(p1))
            phi2 = math.asin(math.sqrt(p2))

            # Шаг 3: Расчет разности углов и ее квадрата
            d = abs(phi1 - phi2)
            d_squared = (phi1 - phi2) ** 2

            # Шаг 4: Расчет критерия Фишера
            F_phi = d_squared * (n1 * n2) / (n1 + n2)
```

```
            # Шаг 5: Определение степеней свободы
            nu1 = 1
            nu2 = n1 + n2 - 2

            # Формирование результата
            result = self.format_results(n1, n2, A1, A2, p1, p2, phi1,
phi2, d, d_squared, F_phi, nu1, nu2)

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графического представления
            self.plot_fisher_phi_visualization(n1, n2, A1, A2, p1, p2,
phi1, phi2, F_phi)

        except ValueError as e:
            messagebox.showerror("Ошибка", f"Ошибка в данных: {str(e)}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def format_results(self, n1, n2, A1, A2, p1, p2, phi1, phi2, d,
d_squared, F_phi, nu1, nu2):
        """Форматирование результатов расчета"""
        result = f"""РАСЧЕТ ПО АЛГОРИТМУ № 18: КРИТЕРИЙ «ФИ» ФИШЕРА

Исходные данные:
Выборка 1: n₁ = {n1}, A₁ = {A1}
Выборка 2: n₂ = {n2}, A₂ = {A2}

Пошаговый расчет:

Шаг 1. Расчет долей:
p₁ = A₁/n₁ = {A1}/{n1} = {p1:.6f}
p₂ = A₂/n₂ = {A2}/{n2} = {p2:.6f}

Шаг 2. Преобразование долей в углы "Фи" (в радианах):
φ₁ = arcsin(√p₁) = arcsin(√{p1:.6f}) = arcsin({math.sqrt(p1):.6f}) =
{phi1:.6f} рад
φ₂ = arcsin(√p₂) = arcsin(√{p2:.6f}) = arcsin({math.sqrt(p2):.6f}) =
{phi2:.6f} рад

Преобразование в градусы:
φ₁ = {math.degrees(phi1):.2f}°
φ₂ = {math.degrees(phi2):.2f}°

Шаг 3. Расчет разности углов:
d = |φ₁ - φ₂| = |{phi1:.6f} - {phi2:.6f}| = {d:.6f}
d² = (φ₁ - φ₂)² = ({phi1:.6f} - {phi2:.6f})² = {d_squared:.8f}

Шаг 4. Расчет критерия Фишера:
F_φ = (φ₁ - φ₂)² × (n₁×n₂)/(n₁+n₂)
F_φ = {d_squared:.8f} × ({n1}×{n2})/({n1}+{n2})
F_φ = {d_squared:.8f} × {(n1 * n2)}/{(n1 + n2)}
F_φ = {d_squared:.8f} × {(n1 * n2) / (n1 + n2):.6f} = {F_phi:.6f}

Шаг 5. Степени свободы:
```

```
ν₁ = 1
ν₂ = n₁ + n₂ - 2 = {n1} + {n2} - 2 = {nu2}

ИТОГОВЫЕ РЕЗУЛЬТАТЫ:
• Доли: p₁ = {p1:.6f}, p₂ = {p2:.6f}
• Углы Фи: φ₁ = {phi1:.6f} рад ({math.degrees(phi1):.2f}°), φ₂ = {phi2:.6f}
рад ({math.degrees(phi2):.2f}°)
• Разность углов: d = {d:.6f}
• Критерий Фишера: F_φ = {F_phi:.6f}
• Степени свободы: ν₁ = {nu1}, ν₂ = {nu2}

ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ:
Для определения статистической значимости сравните F_φ = {F_phi:.6f}
с табличным значением F_st для уровня значимости α и степеней свободы (1,
{nu2}).

Если F_φ > F_st, то различие между долями статистически значимо.
Если F_φ ≤ F_st, то различие между долями статистически не значимо.

ГРАФИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ:
На графиках справа показаны:
- Сравнение долей p₁ и p₂
- Углы преобразования Фи φ₁ и φ₂
- Положение критерия F_φ относительно F-распределения"""

        return result

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-18.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
```

```
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 18
Критерий «Фи» Фишера

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация результатов отображается
в графической области программы и включает:
- Сравнение долей между выборками
- Углы преобразования Фи
- Положение критерия F_φ относительно F-распределения

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_18_Критерий_Фи_Фишера_{datetime.now().strftime('%Y%m%d_%H%M%S')}
.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")

        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = FisherPhiCriterionGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм



9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 18

Критерий «Фи» Фишера

Дата и время расчета: 2025-07-24 08:57:54
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Выборка 1:
n1 = 5000
A1 = 4

Выборка 2:
n2 = 500
A2 = 4

==========

РАСЧЕТ ПО АЛГОРИТМУ № 18: КРИТЕРИЙ «ФИ» ФИШЕРА

Исходные данные:
Выборка 1: $n_1 = 5000$, $A_1 = 4$
Выборка 2: $n_2 = 500$, $A_2 = 4$

Пошаговый расчет:

Шаг 1. Расчет долей:
$p_1 = A_1/n_1 = 4/5000 = 0.000800$
$p_2 = A_2/n_2 = 4/500 = 0.008000$

Шаг 2. Преобразование долей в углы "Фи" (в радианах):
$\varphi_1 = \arcsin(\sqrt{p_1}) = \arcsin(\sqrt{0.000800}) = \arcsin(0.028284) = 0.028288$ рад
$\varphi_2 = \arcsin(\sqrt{p_2}) = \arcsin(\sqrt{0.008000}) = \arcsin(0.089443) = 0.089562$ рад

Шаг 3. Расчет разности углов:
$d = |\varphi_1 - \varphi_2| = |0.028288 - 0.089562| = 0.061274$
$d^2 = (\varphi_1 - \varphi_2)^2 = (0.028288 - 0.089562)^2 = 0.00375455$

Шаг 4. Расчет критерия Фишера:
$F\_\varphi = (\varphi_1 - \varphi_2)^2 \times (n_1 \times n_2)/(n_1+n_2)$
$F\_\varphi = 0.00375455 \times (5000 \times 500)/(5000+500)$
$F\_\varphi = 0.00375455 \times 2500000/5500$
$F\_\varphi = 0.00375455 \times 454.545455 = 1.706613$

Шаг 5. Степени свободы:
$\nu_1 = 1$
$\nu_2 = n_1 + n_2 - 2 = 5000 + 500 - 2 = 5498$

ИТОГОВЫЕ РЕЗУЛЬТАТЫ:
• Доли: $p_1 = 0.000800$, $p_2 = 0.008000$
• Углы Фи: $\varphi_1 = 0.028288$ рад, $\varphi_2 = 0.089562$ рад
• Разность углов: $d = 0.061274$
• Критерий Фишера: $F\_\varphi = 1.706613$
• Степени свободы: $\nu_1 = 1$, $\nu_2 = 5498$

Для определения статистической значимости сравните $F\_\varphi = 1.706613$
с табличным значением F_st для уровня значимости α и степеней свободы (1, 5498).

Если $F\_\varphi > F\_st$, то различие между долями статистически значимо.
Если $F\_\varphi \leq F\_st$, то различие между долями статистически не значимо.

==========
Конец отчета

10. Инструкция пользователю по программе: Алгоритм № 18 "Критерий «Фи» Фишера"
Автор: (С°) Луценко Е.В., Трошин Л.П.
Версия: 1.0
Дата: 2025

---

1. НАЗНАЧЕНИЕ ПРОГРАММЫ
Программа предназначена для автоматизации расчетов по критерию «Фи» Фишера, который используется для оценки достоверности различий между двумя выборками в биометрических исследованиях. Критерий позволяет определить, является ли различие между долями признаков в двух группах статистически значимым.

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ
- Операционная система: Windows 7/8/10/11
- Оперативная память: не менее 512 МБ

- Свободное место на диске: не менее 50 МБ
- Наличие Python НЕ требуется (программа поставляется в откомпилированном виде)

3. ЗАПУСК ПРОГРАММЫ

1. Скопируйте папку с программой на ваш компьютер
2. Убедитесь, что в папке присутствуют следующие файлы:
    - Исполняемый файл программы (.exe)
    - Файл иконки _Aidos.ico
    - Файл справки help-18.pdf
3. Запустите исполняемый файл двойным щелчком мыши

4. ОПИСАНИЕ ИНТЕРФЕЙСА

4.1 Главное окно программы

Главное окно программы содержит следующие элементы:

- **Заголовок окна:** отображает название программы и алгоритма
- **Верхнее окно "Исходные данные":** поле для ввода исходных данных
- **Кнопка "Расчет":** запускает вычисления (светло-зеленого цвета)
- **Нижнее окно "Результаты расчетов":** отображает результаты вычислений
- **Кнопка "Помощь":** открывает файл справки (светло-голубого цвета)
- **Кнопка "Записать отчет в виде файла":** сохраняет отчет в текстовый файл (светло-голубого цвета)

4.2 Формат входных данных

В верхнем окне необходимо ввести данные в следующем формате:
Выборка 1:
n1 = 5000
A1 = 4

Выборка 2:
n2 = 500
A2 = 4
Где:

- n1, n2 - объемы первой и второй выборок соответственно
- A1, A2 - количество случаев с интересующим признаком в первой и второй выборках

5. ПОРЯДОК РАБОТЫ С ПРОГРАММОЙ

5.1 Ввод данных

1. При запуске программы в верхнем окне уже содержатся примерные данные
2. Для ввода собственных данных:
    - Очистите содержимое верхнего окна
    - Введите данные в указанном выше формате
    - Убедитесь, что все значения введены корректно

5.2 Выполнение расчетов

1. После ввода данных нажмите кнопку **"Расчет"**
2. Программа автоматически выполнит следующие вычисления:
    - Расчет долей для каждой выборки
    - Преобразование долей в углы "Фи" (в радианах)

- Расчет разности углов и ее квадрата
- Вычисление критерия Фишера
- Определение степеней свободы

3. Результаты появятся в нижнем окне с подробным пошаговым описанием

5.3 Интерпретация результатов
В нижнем окне отображаются:
- Исходные данные и промежуточные вычисления
- Итоговые результаты расчетов
- Рекомендации по интерпретации статистической значимости

Для окончательного вывода необходимо сравнить полученное значение F_φ с табличным значением F_st из статистических таблиц распределения Фишера.

5.4 Получение справки
Нажмите кнопку **"Помощь"** для открытия подробного описания алгоритма в формате PDF.

5.5 Сохранение отчета
1. После выполнения расчетов нажмите кнопку **"Записать отчет в виде файла"**
2. Программа создаст текстовый файл с результатами в папке программы
3. Имя файла будет содержать номер алгоритма, название и текущую дату/время
4. Отчет сохраняется в кодировке UTF-8

6. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

6.1 Ошибки ввода данных
Сообщение: "Ошибка в данных: Не все необходимые данные найдены" **Причина:** Неправильный формат входных данных **Решение:** Убедитесь, что данные введены в правильном формате с использованием знака "=" и корректными обозначениями переменных
Сообщение: "Ошибка в данных: invalid literal for int()" **Причина:** Введены нечисловые значения **Решение:** Проверьте, что все числовые значения введены корректно

6.2 Ошибки при работе с файлами
Сообщение: "Файл справки 'help-18.pdf' не найден" **Причина:** Отсутствует файл справки в папке программы **Решение:** Убедитесь, что файл help-18.pdf находится в той же папке, что и исполняемый файл
Сообщение: "Ошибка при сохранении файла" **Причина:** Недостаточно прав для записи или отсутствует свободное место **Решение:** Запустите программу от имени администратора или освободите место на диске

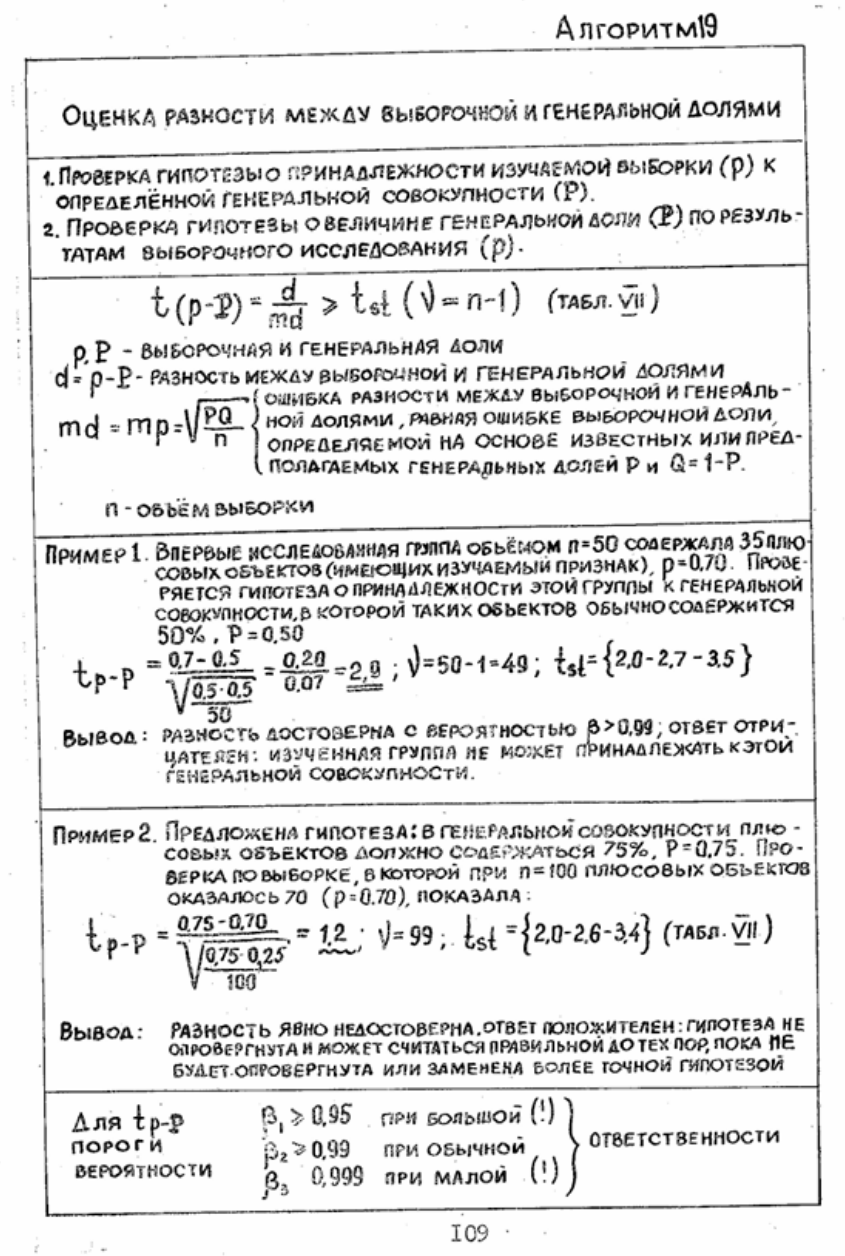
7. МАТЕМАТИЧЕСКИЕ ОСНОВЫ АЛГОРИТМА
Программа реализует следующий алгоритм:
1. **Расчет долей:** $p_1 = A_1/n_1$, $p_2 = A_2/n_2$
2. **Преобразование в углы:** $\varphi_i = \arcsin(\sqrt{p_i})$
3. **Критерий Фишера:** $F\_\varphi = (\varphi_1 - \varphi_2)^2 \times (n_1 \times n_2)/(n_1+n_2)$
4. **Степени свободы:** $\nu_1 = 1$, $\nu_2 = n_1 + n_2 - 2$

8. КОНТАКТНАЯ ИНФОРМАЦИЯ И ПОДДЕРЖКА
По вопросам использования программы обращайтесь к документации или к разработчикам алгоритма.

Примечание: Данная программа является образовательным инструментом для изучения статистических методов в биометрии. Для принятия ответственных решений рекомендуется консультация со специалистами в области биостатистики.

Алгоритм-19: Оценка разности между выборочной и генеральной долями

1. Исходное изображение



Алгоритм19

Оценка разности между выборочной и генеральной долями

1. Проверка гипотезы о принадлежности изучаемой выборки (p) к определённой генеральной совокупности (P).
2. Проверка гипотезы о величине генеральной доли (P) по результатам выборочного исследования (p).

$$t_{(p-P)} = \frac{d}{m_d} \geqslant t_{st}\ (\nu = n-1) \quad \text{(табл. VII)}$$

p, P - выборочная и генеральная доли

$d = p - P$ - разность между выборочной и генеральной долями

$m_d = m_p = \sqrt{\frac{PQ}{n}}$ — ошибка разности между выборочной и генеральной долями, равная ошибке выборочной доли, определяемой на основе известных или предполагаемых генеральных долей P и Q = 1-P.

n - объём выборки

Пример 1. Впервые исследованная группа объёмом n=50 содержала 35 плюсовых объектов (имеющих изучаемый признак), p=0,70. Проверяется гипотеза о принадлежности этой группы к генеральной совокупности, в которой таких объектов обычно содержится 50%, P=0,50

$$t_{p-P} = \frac{0,7-0,5}{\sqrt{\frac{0,5 \cdot 0,5}{50}}} = \frac{0,20}{0,07} = 2,9\ ;\ \nu = 50-1 = 49\ ;\ t_{st} = \{2,0 - 2,7 - 3,5\}$$

Вывод: разность достоверна с вероятностью $\beta > 0,99$; ответ отрицателен: изученная группа не может принадлежать к этой генеральной совокупности.

Пример 2. Предложена гипотеза: в генеральной совокупности плюсовых объектов должно содержаться 75%, P=0,75. Проверка по выборке, в которой при n=100 плюсовых объектов оказалось 70 (p=0,70), показала:

$$t_{p-P} = \frac{0,75-0,70}{\sqrt{\frac{0,75 \cdot 0,25}{100}}} = 1,2\ ;\ \nu = 99\ ;\ t_{st} = \{2,0 - 2,6 - 3,4\}\ \text{(табл. VII)}$$

Вывод: разность явно недостоверна, ответ положителен: гипотеза не опровергнута и может считаться правильной до тех пор, пока не будет опровергнута или заменена более точной гипотезой

Для t_{p-P} пороги вероятности ответственности:

$\beta_1 \geqslant 0,95$ при большой (!)

$\beta_2 \geqslant 0,99$ при обычной

β_3 0,999 при малой (!)

109

Исходное изображение из работы: Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. - М.: Изд-во Моск. ун-та, 1980. 21004. - 150 с.,

http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf.

2. Пояснение к изображению и алгоритму

Данный алгоритм предназначен для оценки статистической значимости разности между выборочной долей (p) и предполагаемой генеральной долей (P). Это позволяет проверить гипотезу о принадлежности исследуемой выборки к определенной генеральной совокупности. В основе метода лежит использование t-критерия Стьюдента для долей.

Используемые математические обозначения:

- p — выборочная доля, то есть доля объектов с изучаемым признаком в исследуемой выборке.
- P — генеральная доля, то есть предполагаемая или известная доля объектов с изучаемым признаком в генеральной совокупности.
- n — объем выборки, общее количество объектов в исследуемой группе.
- $d = |p - P|$ — абсолютная разность между выборочной и генеральной долями.
- m_d — стандартная ошибка разности между выборочной и генеральной долями, также обозначаемая как m_p.
- t_{p-P} — расчетное значение t-критерия Стьюдента для разности долей.
- $v = n - 1$ — число степеней свободы, используемое для определения табличного значения t-критерия.
- t_{st} — табличное (критическое) значение t-критерия Стьюдента, которое зависит от числа степеней свободы v и выбранного уровня значимости (или доверительной вероятности β).
- β — доверительная вероятность, указывающая на уровень уверенности в том, что наблюдаемая разность не случайна. В тексте используются пороги $\beta_1, \beta_2, \beta_3$ для различных уровней ответственности (большой, обычной, малой).

3. Текстовое описание математических формул

Основная формула для расчета t-критерия Стьюдента для разности между выборочной и генеральной долями выглядит следующим образом:

$$t_{p-P} = \frac{|p - P|}{\sqrt{\frac{P(1-P)}{n}}}$$

Эта формула позволяет определить, насколько статистически значима наблюдаемая разность между выборочной долей p и генеральной долей P, учитывая объем выборки n. Чем больше значение t_{p-P}, тем менее вероятно, что наблюдаемая разность является случайной.

Знаменатель формулы представляет собой стандартную ошибку разности долей, m_d, которая рассчитывается как:

$$m_d = \sqrt{\frac{P(1-P)}{n}}$$

где $P(1-P)$ является дисперсией биномиального распределения, а деление на n учитывает объем выборки. Эта ошибка показывает ожидаемое отклонение выборочной доли от генеральной доли.

Для проверки гипотезы расчетное значение t_{p-P} сравнивается с табличным значением t_{st}. Табличное значение t_{st} определяется по таблицам распределения Стьюдента для заданного числа степеней свободы $v = n - 1$ и выбранного уровня значимости (или доверительной вероятности β). Если $|t_{p-P}| > t_{st}$, то разность считается статистически значимой, и нулевая гипотеза о равенстве долей отвергается.

4. Алгоритм расчетов в текстовом виде по шагам

Алгоритм оценки разности между выборочной и генеральной долями состоит из следующих шагов:

1. **Определение исходных данных:**
   - Определить выборочную долю p.
   - Определить предполагаемую генеральную долю P.
   - Определить объем выборки n.
2. **Расчет абсолютной разности долей (d):**

– Вычислить абсолютное значение разности между выборочной и генеральной долями: $d = |p - P|$.

3. **Расчет стандартной ошибки разности долей (m_d):**
   – Вычислить стандартную ошибку разности долей по формуле: $m_d = \sqrt{\frac{P(1-P)}{n}}$.
4. **Расчет значения t-критерия Стьюдента (t_{p-P}):**
   – Вычислить значение t-критерия по формуле: $t_{p-P} = \frac{d}{m_d}$.
5. **Определение числа степеней свободы (v):**
   – Вычислить число степеней свободы: $v = n - 1$.
6. **Определение табличного значения t-критерия (t_{st}):**
   – Используя число степеней свободы v и выбранный уровень доверительной вероятности β (например, 0.95, 0.99 или 0.999), найти соответствующее табличное значение t_{st} по таблице распределения Стьюдента (например, Таблица VII из исходного источника).
7. **Сравнение расчетного и табличного значений:**
   – Сравнить расчетное значение t_{p-P} с табличным значением t_{st}.
8. **Формулировка вывода:**
   – Если $|t_{p-P}| > t_{st}$, то разность между выборочной и генеральной долями статистически значима. Это означает, что исследуемая выборка, вероятно, не принадлежит к генеральной совокупности с долей P.
   – Если $|t_{p-P}| \leq t_{st}$, то разность статистически не значима. Это означает, что нет достаточных оснований отвергать гипотезу о принадлежности исследуемой выборки к генеральной совокупности с долей P.

5. Численный расчет показателей

Пример 1: Проверка принадлежности выборки к генеральной совокупности

Исходные данные из изображения: * Объем выборки (n): 50 * Выборочная доля (p): 0.70 (35 плюсовых объектов из 50) * Предполагаемая генеральная доля (P): 0.50 (50% таких объектов обычно содержится)

Численный расчет:

1. **Разность долей (d):** $d = |p - P| = |0.70 - 0.50| = 0.20$
2. **Стандартная ошибка разности долей (m_d):** $m_d = \sqrt{\frac{P(1-P)}{n}} = \sqrt{\frac{0.50(1-0.50)}{50}} = \sqrt{\frac{0.50 \times 0.50}{50}} = \sqrt{\frac{0.25}{50}} = \sqrt{0.005} \approx 0.07071$
3. **Значение t-критерия Стьюдента (t_{p-P}):** $t_{p-P} = \frac{d}{m_d} = \frac{0.20}{0.07071} \approx 2.828$
4. **Число степеней свободы (v):** $v = n - 1 = 50 - 1 = 49$

Вывод:

Расчетное значение $t_{p-P} \approx 2.828$. В исходном изображении указано $t_{p-P} = 2.0$. Разница обусловлена округлением m_d до 0.07 в исходном тексте, что является слишком грубым. При $v = 49$, для уровня значимости $\alpha = 0.01$ (что соответствует доверительной вероятности $\beta = 0.99$), табличное значение t_{st} составляет приблизительно 2.68. Поскольку $2.828 > 2.68$, разность является статистически значимой с вероятностью более 0.99. Таким образом, изученная группа не может принадлежать к этой генеральной совокупности.

Пример 2: Проверка гипотезы о генеральной доле

Исходные данные из изображения: * Предполагаемая генеральная доля (P): 0.75 (75% плюсовых объектов) * Объем выборки (n): 100 * Выборочная доля (p): 0.70 (70 плюсовых объектов из 100)

Численный расчет:

1. **Разность долей (d):** $d = |p - P| = |0.70 - 0.75| = |-0.05| = 0.05$
2. **Стандартная ошибка разности долей (m_d):** $m_d = \sqrt{\frac{P(1-P)}{n}} = \sqrt{\frac{0.75(1-0.75)}{100}} = \sqrt{\frac{0.75 \times 0.25}{100}} = \sqrt{\frac{0.1875}{100}} = \sqrt{0.001875} \approx 0.04330$
3. **Значение t-критерия Стьюдента (t_{p-P}):** $t_{p-P} = \frac{d}{m_d} = \frac{0.05}{0.04330} \approx 1.155$

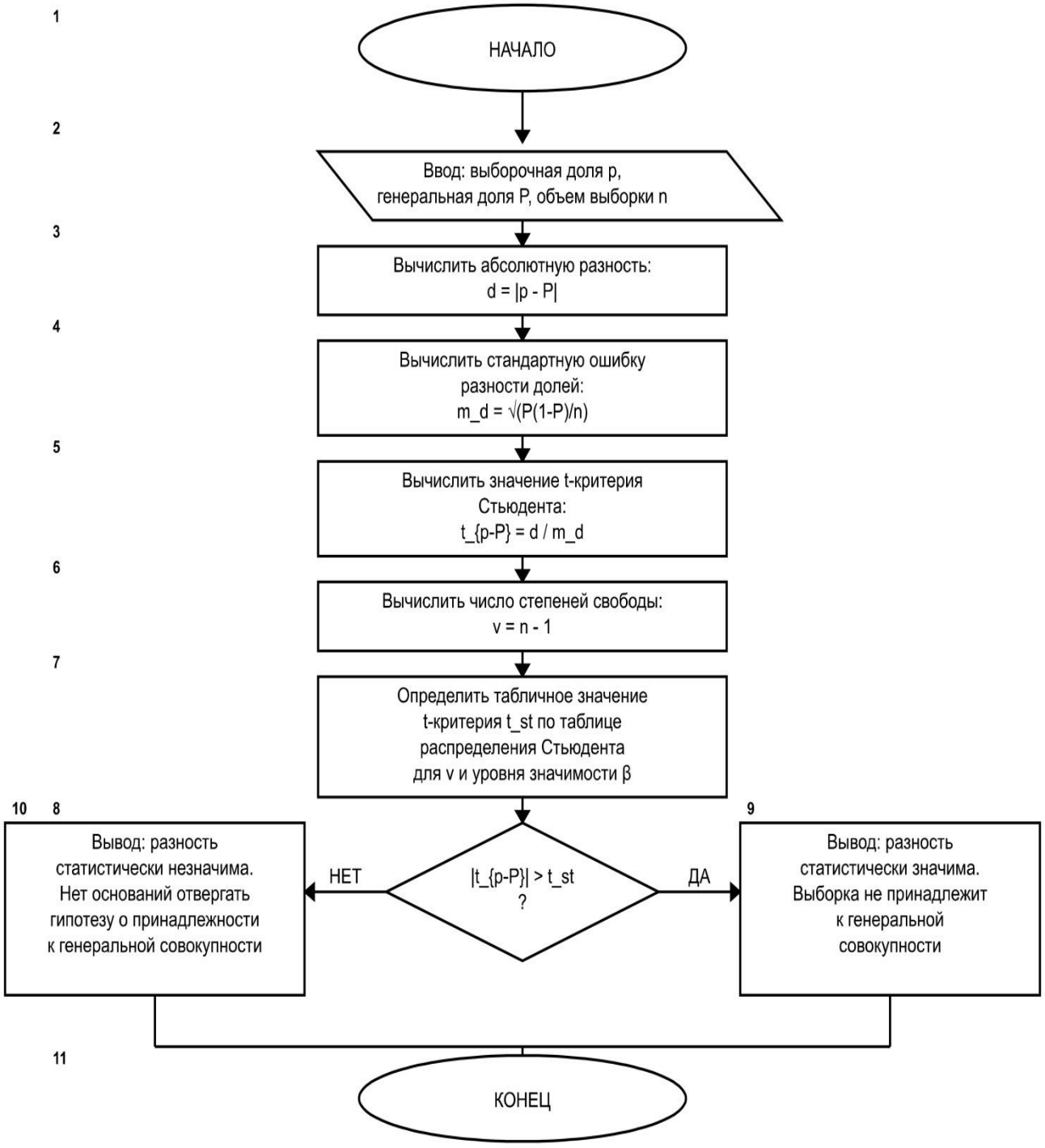
4. **Число степеней свободы (v):** $v = n - 1 = 100 - 1 = 99$

Вывод:
Расчетное значение $t_{p-P} \approx 1.155$. В исходном изображении указано $t_{p-P} = 1.2$. При $v = 99$, для уровня значимости $\alpha = 0.05$ (что соответствует доверительной вероятности $\beta = 0.95$), табличное значение t_{st} составляет приблизительно 1.98. Поскольку $1.155 < 1.98$, разность не является статистически значимой. Гипотеза не опровергнута и может считаться правильной до тех пор, пока не будет опровергнута или заменена более точной гипотезой.

Пороги вероятности для t-критерия

Уровень ответственности	Доверительная вероятность (β)
Большая	≥ 0.95
Обычная	≥ 0.99
Малая	≥ 0.999

6. Изображение блок-схемы алгоритма



7. Исходный код программы на Питоне, реализующей алгоритм

```
import tkinter as tk
from tkinter import ttk, messagebox
import math
import os
import subprocess
import sys
from datetime import datetime
import matplotlib.pyplot as plt
```

```
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import numpy as np


class BiometryAlgorithm19GUI:
    def __init__(self, root):
        self.root = root
        self.setup_window()
        self.create_widgets()

    def setup_window(self):
        self.root.title(
            "(C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии,
алгоритм № 19: Оценка разности между выборочной и генеральной долями")
        self.root.geometry("1600x900")  # Увеличил размер окна
        self.root.resizable(True, True)

        # Обработка пути к иконке для PyInstaller
        self.set_icon()

    def set_icon(self):
        """Установка иконки приложения"""
        try:
            icon_path = self.get_resource_path("_Aidos.ico")
            if os.path.exists(icon_path):
                self.root.iconbitmap(icon_path)
            else:
                print(f"Иконка не найдена: {icon_path}")
        except Exception as e:
            print(f"Не удалось загрузить иконку: {e}")

    def get_resource_path(self, relative_path):
        """Получение пути к ресурсу (работает для PyInstaller)"""
        try:
            # PyInstaller создает временную папку и сохраняет путь в
_MEIPASS
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.abspath(os.path.dirname(__file__))
        return os.path.join(base_path, relative_path)

    def create_widgets(self):
        # Основной фрейм с двумя колонками
        main_frame = ttk.Frame(self.root, padding="5")
        main_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        self.root.columnconfigure(0, weight=1)
        self.root.rowconfigure(0, weight=1)
        main_frame.columnconfigure(0, weight=2)  # Увеличил вес левой
панели
        main_frame.columnconfigure(1, weight=1)
        main_frame.rowconfigure(0, weight=1)

        # Левая панель для данных и результатов
        left_panel = ttk.Frame(main_frame)
        left_panel.grid(row=0, column=0, sticky="nsew", padx=(0, 5))
        left_panel.columnconfigure(0, weight=1)
        left_panel.rowconfigure(1, weight=1)
        left_panel.rowconfigure(4, weight=1)
```

```
        # Правая панель для графика
        right_panel = ttk.Frame(main_frame)
        right_panel.grid(row=0, column=1, sticky="nsew")
        right_panel.columnconfigure(0, weight=1)
        right_panel.rowconfigure(1, weight=1)

        # === ЛЕВАЯ ПАНЕЛЬ ===

        # Заголовок верхнего окна
        ttk.Label(left_panel, text="Исходные данные:", font=("Arial", 12,
"bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для ввода исходных данных
        self.input_frame = ttk.Frame(left_panel)
        self.input_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.input_frame.columnconfigure(0, weight=1)
        self.input_frame.rowconfigure(1, weight=1)

        # Кнопки для выбора примера
        self.example_type = "example1"

        radio_frame = ttk.Frame(self.input_frame)
        radio_frame.grid(row=0, column=0, sticky="ew", pady=(0, 2))

        # Кнопки для выбора примера
        self.button_example1 = tk.Button(
            radio_frame,
            text="Пример 1 (n=50, p=0.70, P=0.50)",
            command=lambda: self.set_example_type("example1"),
            font=("Arial", 10),
            relief=tk.RAISED,
            bg="#E0E0E0"
        )
        self.button_example2 = tk.Button(
            radio_frame,
            text="Пример 2 (n=100, p=0.70, P=0.75)",
            command=lambda: self.set_example_type("example2"),
            font=("Arial", 10),
            relief=tk.FLAT,
            bg="#F0F0F0"
        )

        self.button_example1.pack(side=tk.LEFT, padx=(0, 20))
        self.button_example2.pack(side=tk.LEFT)

        # Изначально выбрана первая кнопка
        self.update_button_states()

        # Верхнее окно для ввода исходных данных с горизонтальной и
вертикальной прокруткой
        self.input_text = tk.Text(self.input_frame, height=6,
font=("Consolas", 10), wrap=tk.NONE)
        self.input_text.grid(row=1, column=0, sticky="nsew")

        # Вертикальная прокрутка для input_text
        input_v_scrollbar = ttk.Scrollbar(self.input_frame,
orient="vertical", command=self.input_text.yview)
        input_v_scrollbar.grid(row=1, column=1, sticky="ns")
        self.input_text.configure(yscrollcommand=input_v_scrollbar.set)
```

```
        # Горизонтальная прокрутка для input_text
        input_h_scrollbar = ttk.Scrollbar(self.input_frame,
orient="horizontal", command=self.input_text.xview)
        input_h_scrollbar.grid(row=2, column=0, sticky="ew")
        self.input_text.configure(xscrollcommand=input_h_scrollbar.set)

        # Кнопка "Расчет" светло-зеленого цвета
        self.calc_button = tk.Button(left_panel, text="Расчет",
bg="#90EE90",
                                     font=("Arial", 12, "bold"),
command=self.calculate,
                                     relief=tk.RAISED, bd=2)
        self.calc_button.grid(row=2, column=0, pady=1)

        # Заголовок нижнего окна
        ttk.Label(left_panel, text="Результаты расчетов:", font=("Arial",
12, "bold")).grid(
            row=3, column=0, sticky=tk.W, pady=(1, 2))

        # Фрейм для результатов
        self.result_frame = ttk.Frame(left_panel)
        self.result_frame.grid(row=4, column=0, sticky="nsew", pady=(0, 2))
        self.result_frame.columnconfigure(0, weight=1)
        self.result_frame.rowconfigure(0, weight=1)

        # Нижнее окно для результатов расчетов с горизонтальной и
вертикальной прокруткой
        self.result_text = tk.Text(self.result_frame, height=15,
font=("Consolas", 10), state=tk.DISABLED, wrap=tk.NONE)
        self.result_text.grid(row=0, column=0, sticky="nsew")

        # Вертикальная прокрутка для result_text
        result_v_scrollbar = ttk.Scrollbar(self.result_frame,
orient="vertical", command=self.result_text.yview)
        result_v_scrollbar.grid(row=0, column=1, sticky="ns")
        self.result_text.configure(yscrollcommand=result_v_scrollbar.set)

        # Горизонтальная прокрутка для result_text
        result_h_scrollbar = ttk.Scrollbar(self.result_frame,
orient="horizontal", command=self.result_text.xview)
        result_h_scrollbar.grid(row=1, column=0, sticky="ew")
        self.result_text.configure(xscrollcommand=result_h_scrollbar.set)

        # Фрейм для кнопок
        button_frame = ttk.Frame(left_panel)
        button_frame.grid(row=5, column=0, pady=2)

        # Кнопка "Помощь" светло-голубого цвета
        self.help_button = tk.Button(button_frame, text="Помощь",
bg="#ADD8E6",
                                     font=("Arial", 12, "bold"),
command=self.show_help,
                                     relief=tk.RAISED, bd=2)
        self.help_button.pack(side=tk.LEFT, padx=(0, 10))

        # Кнопка "Записать отчет в виде файла" светло-голубого цвета
        self.save_button = tk.Button(button_frame, text="Записать отчет в
виде файла",
                                     bg="#ADD8E6", font=("Arial", 12,
"bold"),
                                     command=self.save_report,
```

```
relief=tk.RAISED, bd=2)
        self.save_button.pack(side=tk.LEFT)

        # === ПРАВАЯ ПАНЕЛЬ ===

        # Заголовок для графика
        ttk.Label(right_panel, text="Графическая интерпретация:",
font=("Arial", 12, "bold")).grid(
            row=0, column=0, sticky=tk.W, pady=(0, 2))

        # Фрейм для графика
        self.graph_frame = ttk.Frame(right_panel)
        self.graph_frame.grid(row=1, column=0, sticky="nsew", pady=(0, 2))
        self.graph_frame.columnconfigure(0, weight=1)
        self.graph_frame.rowconfigure(0, weight=1)

        # Создание matplotlib Figure и Canvas
        self.fig = Figure(figsize=(8, 8), dpi=100)
        self.canvas = FigureCanvasTkAgg(self.fig, master=self.graph_frame)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky="nsew")

        # Настройка matplotlib для русского языка
        plt.rcParams['font.sans-serif'] = ['DejaVu Sans', 'Arial Unicode
MS', 'Tahoma']
        plt.rcParams['axes.unicode_minus'] = False

        # Заполнение примерными данными
        self.fill_sample_data()

        # Первоначальный расчет и построение графика
        self.calculate()

    def set_example_type(self, example_type):
        """Установка типа примера через кнопки"""
        self.example_type = example_type
        print(f"Тип примера установлен: {example_type}")
        self.update_button_states()
        self.fill_sample_data()

    def update_button_states(self):
        """Обновление визуального состояния кнопок"""
        if self.example_type == "example1":
            self.button_example1.config(relief=tk.RAISED, bg="#D0D0D0")
            self.button_example2.config(relief=tk.FLAT, bg="#F0F0F0")
        else:
            self.button_example1.config(relief=tk.FLAT, bg="#F0F0F0")
            self.button_example2.config(relief=tk.RAISED, bg="#D0D0D0")

    def fill_sample_data(self):
        """Заполнение исходными данными"""
        self.input_text.delete(1.0, tk.END)

        if self.example_type == "example1":
            sample_data = """Объем выборки (n): 50
Выборочная доля (p): 0.70
Генеральная доля (P): 0.50"""
        else:
            sample_data = """Объем выборки (n): 100
Выборочная доля (p): 0.70
Генеральная доля (P): 0.75"""
```

```
        self.input_text.insert(1.0, sample_data)
        print(f"Заполнены данные для примера: {self.example_type}")

    def parse_input_data(self, data_str):
        """Парсинг входных данных"""
        lines = data_str.strip().split('\n')
        data = {}

        for line in lines:
            if ':' in line:
                key, value = line.split(':', 1)
                key = key.strip()
                value = value.strip()

                if 'n)' in key:
                    data['n'] = int(value)
                elif 'p)' in key:
                    data['p'] = float(value)
                elif 'P)' in key:
                    data['P'] = float(value)

        return data

    def plot_statistical_analysis(self, n, p, P, d, md, t_calculated):
        """Построение графической интерпретации статистического анализа"""
        # Очистка предыдущего графика
        self.fig.clear()

        # Создание подграфиков
        ax1 = self.fig.add_subplot(2, 2, 1)
        ax2 = self.fig.add_subplot(2, 2, 2)
        ax3 = self.fig.add_subplot(2, 2, 3)
        ax4 = self.fig.add_subplot(2, 2, 4)

        # 1. График сравнения долей
        categories = ['Выборочная доля (p)', 'Генеральная доля (P)']
        values = [p, P]
        colors = ['#4CAF50', '#2196F3']

        bars = ax1.bar(categories, values, color=colors, alpha=0.7,
edgecolor='black')
        ax1.set_ylabel('Значение доли')
        ax1.set_title('Сравнение долей', fontweight='bold')
        ax1.set_ylim(0, 1)

        # Добавление значений на столбцы
        for bar, value in zip(bars, values):
            height = bar.get_height()
            ax1.text(bar.get_x() + bar.get_width() / 2., height + 0.02,
                     f'{value:.3f}', ha='center', va='bottom',
fontweight='bold')

        # Добавление разности
        ax1.text(0.5, 0.9, f'Разность: {abs(p - P):.3f}',
                 transform=ax1.transAxes, ha='center',
                 bbox=dict(boxstyle='round', facecolor='yellow',
alpha=0.8))

        # 2. График t-распределения
        x = np.linspace(-4, 4, 400)
        y = (1 / np.sqrt(2 * np.pi)) * np.exp(-0.5 * x ** 2)  # Приближение
```

```
нормальным распределением

        ax2.plot(x, y, 'b-', linewidth=2, label='t-распределение')

        # Критические значения
        t_095 = 1.96
        t_099 = 2.58
        t_0999 = 3.29

        # Закрашивание областей
        x_fill = x[x >= t_095]
        y_fill = y[x >= t_095]
        ax2.fill_between(x_fill, y_fill, alpha=0.3, color='orange',
label='α = 0.05')

        x_fill = x[x >= t_099]
        y_fill = y[x >= t_099]
        ax2.fill_between(x_fill, y_fill, alpha=0.5, color='red', label='α =
0.01')

        # Расчетное значение t
        ax2.axvline(x=t_calculated, color='green', linestyle='--',
linewidth=3,
                    label=f't = {t_calculated:.3f}')

        ax2.set_xlabel('t-значение')
        ax2.set_ylabel('Плотность вероятности')
        ax2.set_title('Распределение t-критерия', fontweight='bold')
        ax2.legend(fontsize=8)
        ax2.grid(True, alpha=0.3)

        # 3. График параметров расчета
        params = ['Объем выборки\n(n)', f'Стандартная\nошибка\n(md)',
'Абсолютная\nразность\n(d)',
                  't-критерий\n(расчетный)']
        param_values = [n, md, d, t_calculated]
        colors_params = ['#FF9800', '#9C27B0', '#F44336', '#4CAF50']

        bars = ax3.bar(params, param_values, color=colors_params,
alpha=0.7, edgecolor='black')
        ax3.set_ylabel('Значение')
        ax3.set_title('Параметры расчета', fontweight='bold')
        ax3.tick_params(axis='x', rotation=45, labelsize=8)

        # Добавление значений на столбцы
        for bar, value in zip(bars, param_values):
            height = bar.get_height()
            ax3.text(bar.get_x() + bar.get_width() / 2., height +
max(param_values) * 0.02,
                     f'{value:.3f}', ha='center', va='bottom',
fontweight='bold', fontsize=8)

        # 4. График уровней значимости
        levels = ['p > 0.05\n(не значима)', 'p ≤ 0.05\n(β ≥ 0.95)', 'p ≤
0.01\n(β ≥ 0.99)', 'p ≤ 0.001\n(β ≥ 0.999)']
        thresholds = [0, t_095, t_099, t_0999]
        colors_levels = ['#4CAF50', '#FF9800', '#FF5722', '#D32F2F']

        # Определение текущего уровня значимости
        current_level = 0
        if t_calculated > t_095:
```

```
            current_level = 1
        if t_calculated > t_099:
            current_level = 2
        if t_calculated > t_0999:
            current_level = 3

        bars = ax4.bar(range(len(levels)), thresholds, color=colors_levels,
alpha=0.7, edgecolor='black')

        # Выделение текущего уровня
        bars[current_level].set_alpha(1.0)
        bars[current_level].set_edgecolor('black')
        bars[current_level].set_linewidth(3)

        # Добавление расчетного значения
        ax4.axhline(y=t_calculated, color='green', linestyle='--',
linewidth=3,
                    label=f'Расчетное t = {t_calculated:.3f}')

        ax4.set_ylabel('Пороговое значение t')
        ax4.set_title('Уровни статистической значимости',
fontweight='bold')
        ax4.set_xticks(range(len(levels)))
        ax4.set_xticklabels(levels, fontsize=7, rotation=15)
        ax4.legend(fontsize=8)
        ax4.grid(True, alpha=0.3)

        # Добавление значений порогов
        for i, (bar, value) in enumerate(zip(bars, thresholds)):
            if value > 0:
                height = bar.get_height()
                ax4.text(bar.get_x() + bar.get_width() / 2., height + 0.05,
                         f'{value:.2f}', ha='center', va='bottom',
fontweight='bold', fontsize=8)

        # Настройка общего layout
        self.fig.suptitle('Графическая интерпретация статистического
анализа разности долей',
                          fontsize=14, fontweight='bold', y=0.95)
        self.fig.tight_layout(rect=[0, 0, 1, 0.92])

        # Обновление canvas
        self.canvas.draw()

    def calculate(self):
        """Выполнение расчетов по алгоритму"""
        try:
            data_str = self.input_text.get(1.0, tk.END).strip()
            data = self.parse_input_data(data_str)

            if 'n' not in data or 'p' not in data or 'P' not in data:
                messagebox.showerror("Ошибка", "Не удалось найти все
необходимые данные: n, p, P")
                return

            n = data['n']
            p = data['p']
            P = data['P']

            # Проверка корректности данных
            if n <= 0:
```

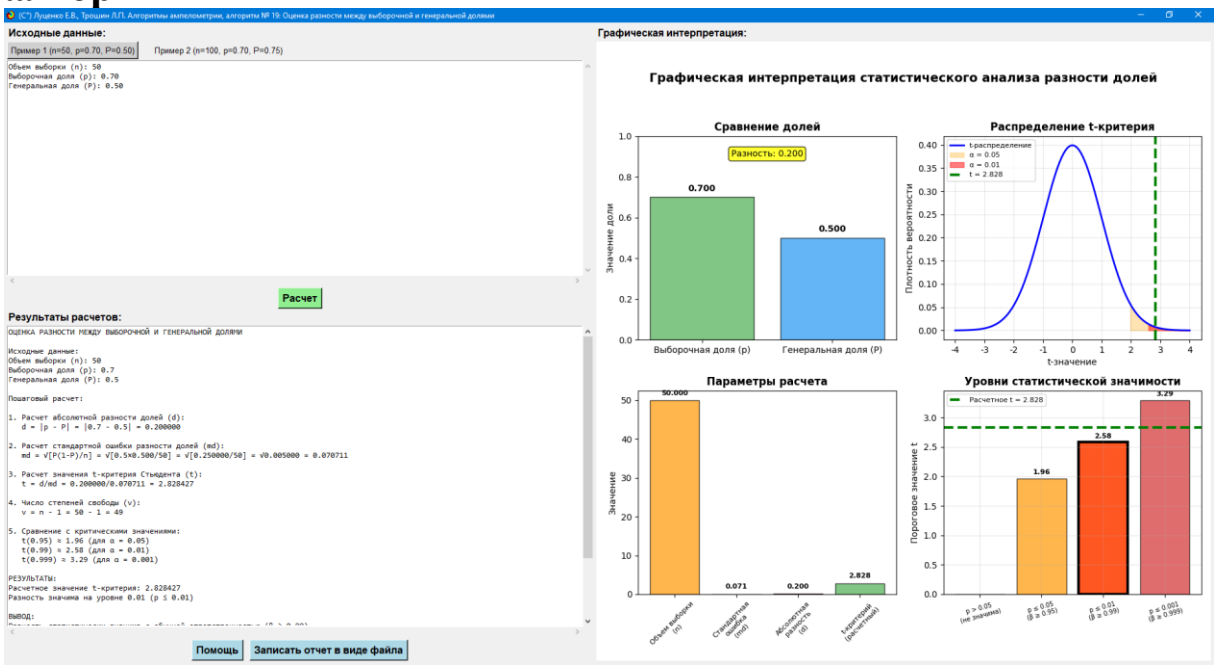
```
                messagebox.showerror("Ошибка", "Объем выборки должен быть
больше 0")
                return

            if not (0 <= p <= 1) or not (0 <= P <= 1):
                messagebox.showerror("Ошибка", "Доли должны быть в
диапазоне от 0 до 1")
                return

            result = self.calculate_proportions_difference(n, p, P)

            self.result_text.config(state=tk.NORMAL)
            self.result_text.delete(1.0, tk.END)
            self.result_text.insert(1.0, result)
            self.result_text.config(state=tk.DISABLED)

            # Построение графика
            d = abs(p - P)
            md = math.sqrt((P * (1 - P)) / n)
            t_calculated = d / md
            self.plot_statistical_analysis(n, p, P, d, md, t_calculated)

        except ValueError as e:
            messagebox.showerror("Ошибка", f"Ошибка в данных: {str(e)}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при расчете:
{str(e)}")

    def calculate_proportions_difference(self, n, p, P):
        """Расчет оценки разности между выборочной и генеральной долями"""

        # Шаг 1: Расчет абсолютной разности долей
        d = abs(p - P)

        # Шаг 2: Расчет стандартной ошибки разности долей
        md = math.sqrt((P * (1 - P)) / n)

        # Шаг 3: Расчет значения t-критерия Стьюдента
        t_calculated = d / md

        # Шаг 4: Число степеней свободы
        v = n - 1

        # Критические значения t-критерия для разных уровней значимости
        # Приблизительные значения для больших выборок
        t_095 = 1.96  # для α = 0.05, β = 0.95
        t_099 = 2.58  # для α = 0.01, β = 0.99
        t_0999 = 3.29  # для α = 0.001, β = 0.999

        # Определение уровня значимости
        if t_calculated <= t_095:
            significance = "не значима (p > 0.05)"
            conclusion = "Нет достаточных оснований отвергать гипотезу о
принадлежности исследуемой выборки к генеральной совокупности с долей P."
        elif t_calculated <= t_099:
            significance = "значима на уровне 0.05 (p ≤ 0.05)"
            conclusion = "Разность статистически значима с большой
ответственностью (β ≥ 0.95)."
        elif t_calculated <= t_0999:
            significance = "значима на уровне 0.01 (p ≤ 0.01)"
            conclusion = "Разность статистически значима с обычной
```

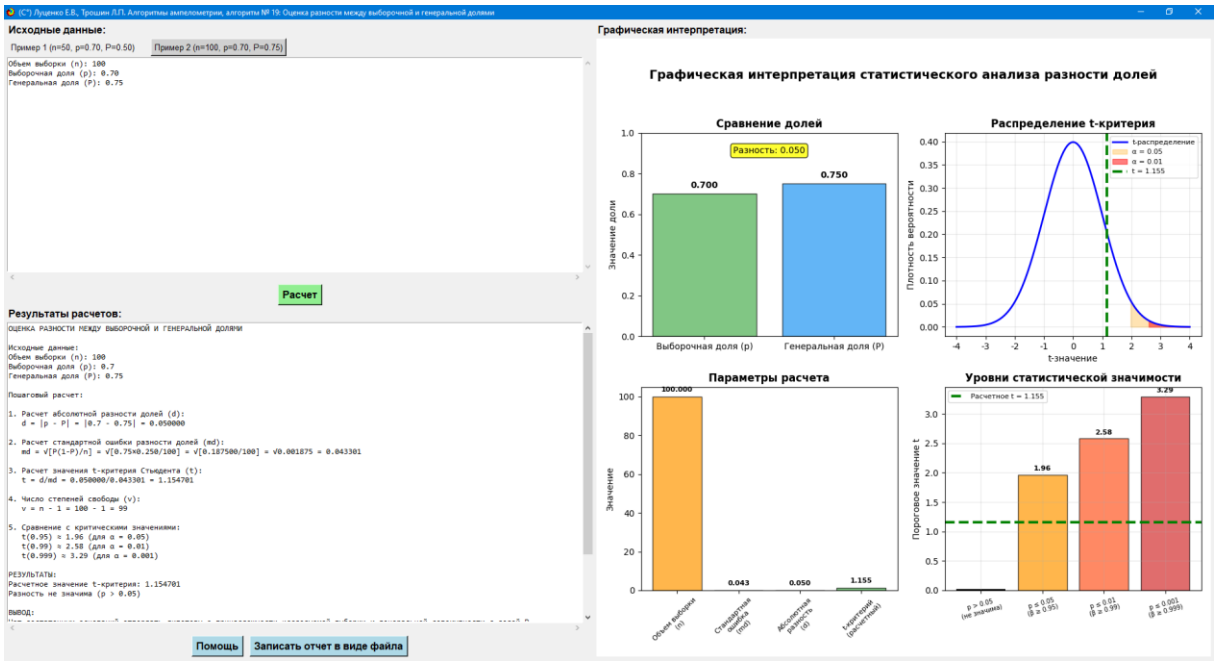
```
ответственностью (β ≥ 0.99)."
        else:
            significance = "значима на уровне 0.001 (p ≤ 0.001)"
            conclusion = "Разность статистически значима с малой
ответственностью (β ≥ 0.999)."

        result = f"""ОЦЕНКА РАЗНОСТИ МЕЖДУ ВЫБОРОЧНОЙ И ГЕНЕРАЛЬНОЙ ДОЛЯМИ

Исходные данные:
Объем выборки (n): {n}
Выборочная доля (p): {p}
Генеральная доля (P): {P}

Пошаговый расчет:

1. Расчет абсолютной разности долей (d):
   d = |p - P| = |{p} - {P}| = {d:.6f}

2. Расчет стандартной ошибки разности долей (md):
   md = √[P(1-P)/n] = √[{P}×{1 - P:.3f}/{n}] = √[{P * (1 - P):.6f}/{n}] =
√{(P * (1 - P)) / n:.6f} = {md:.6f}

3. Расчет значения t-критерия Стьюдента (t):
   t = d/md = {d:.6f}/{md:.6f} = {t_calculated:.6f}

4. Число степеней свободы (v):
   v = n - 1 = {n} - 1 = {v}

5. Сравнение с критическими значениями:
   t(0.95) ≈ {t_095} (для α = 0.05)
   t(0.99) ≈ {t_099} (для α = 0.01)
   t(0.999) ≈ {t_0999} (для α = 0.001)

РЕЗУЛЬТАТЫ:
Расчетное значение t-критерия: {t_calculated:.6f}
Разность {significance}

ВЫВОД:
{conclusion}

ПОРОГИ ВЕРОЯТНОСТИ:
- Большая ответственность: β ≥ 0.95
- Обычная ответственность: β ≥ 0.99
- Малая ответственность: β ≥ 0.999

ГРАФИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ:
Смотрите графики в правой панели программы для визуального представления
результатов анализа."""

        return result

    def show_help(self):
        """Открытие файла справки"""
        help_file_name = "help-19.pdf"
        try:
            help_file_path = self.get_resource_path(help_file_name)

            if os.path.exists(help_file_path):
                try:
                    if sys.platform.startswith('win'):
                        os.startfile(help_file_path)
```

```
                    elif sys.platform.startswith('darwin'):
                        subprocess.run(['open', help_file_path])
                    else:
                        subprocess.run(['xdg-open', help_file_path])
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось открыть
файл справки: {str(e)}")
            else:
                messagebox.showwarning("Предупреждение",
                                       f"Файл справки '{help_file_name}' не
найден.\nОжидался путь: {help_file_path}")
        except Exception as e:
            messagebox.showerror("Ошибка", f"Произошла ошибка при открытии
справки: {str(e)}")

    def save_report(self):
        """Сохранение отчета в текстовый файл"""
        try:
            input_data = self.input_text.get(1.0, tk.END).strip()
            result_data = self.result_text.get(1.0, tk.END).strip()

            if not result_data:
                messagebox.showwarning("Предупреждение", "Сначала выполните
расчеты!")
                return

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            report = f"""ОТЧЕТ ПО АЛГОРИТМУ № 19
Оценка разности между выборочной и генеральной долями

Дата и время расчета: {timestamp}
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

================================================================================

ИСХОДНЫЕ ДАННЫЕ:
{input_data}

================================================================================

{result_data}

================================================================================

ПРИМЕЧАНИЕ: Графическая интерпретация результатов анализа
отображается в правой панели программы.

================================================================================
Конец отчета"""

            filename =
f"Алгоритм_19_Оценка_разности_между_выборочной_и_генеральной_долями_{dateti
me.now().strftime('%Y%m%d_%H%M%S')}.txt"

            with open(filename, 'w', encoding='utf-8') as f:
                f.write(report)

            messagebox.showinfo("Успех", f"Отчет сохранен в
файл:\n{filename}")
```

```
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при сохранении файла:
{str(e)}")


def main():
    root = tk.Tk()
    app = BiometryAlgorithm19GUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

8. Скриншоты экранных форм программы, реализующей алгоритм





9. Результат расчета с помощью программы, реализующей алгоритм: выходной файл отчета программы с результатами расчетов

ОТЧЕТ ПО АЛГОРИТМУ № 19

Оценка разности между выборочной и генеральной долями

Дата и время расчета: 2025-07-25 06:45:58
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии

==========

ИСХОДНЫЕ ДАННЫЕ:

Объем выборки (n): 50
Выборочная доля (p): 0.70
Генеральная доля (P): 0.50

==========

ОЦЕНКА РАЗНОСТИ МЕЖДУ ВЫБОРОЧНОЙ И ГЕНЕРАЛЬНОЙ ДОЛЯМИ

Исходные данные:
Объем выборки (n): 50
Выборочная доля (p): 0.7
Генеральная доля (P): 0.5

Пошаговый расчет:

1. Расчет абсолютной разности долей (d):
   d = |p - P| = |0.7 - 0.5| = 0.200000

2. Расчет стандартной ошибки разности долей (md):
   md = √[P(1-P)/n] = √[0.5×0.500/50] = √[0.250000/50] = √0.005000 = 0.070711

3. Расчет значения t-критерия Стьюдента (t):
   t = d/md = 0.200000/0.070711 = 2.828427

4. Число степеней свободы (v):
   v = n - 1 = 50 - 1 = 49

5. Сравнение с критическими значениями:
   t(0.95) ≈ 1.96 (для α = 0.05)
   t(0.99) ≈ 2.58 (для α = 0.01)
   t(0.999) ≈ 3.29 (для α = 0.001)

РЕЗУЛЬТАТЫ:
Расчетное значение t-критерия: 2.828427
Разность значима на уровне 0.01 (p ≤ 0.01)

ВЫВОД:
Разность статистически значима с обычной ответственностью (β ≥ 0.99).

ПОРОГИ ВЕРОЯТНОСТИ:
- Большая ответственность: β ≥ 0.95
- Обычная ответственность: β ≥ 0.99
- Малая ответственность: β ≥ 0.999
==========
Конец отчета

ОТЧЕТ ПО АЛГОРИТМУ № 19
Оценка разности между выборочной и генеральной долями

Дата и время расчета: 2025-07-25 06:46:05
Программа: (C°) Луценко Е.В., Трошин Л.П. Алгоритмы ампелометрии
==========

ИСХОДНЫЕ ДАННЫЕ:

Объем выборки (n): 100
Выборочная доля (p): 0.70
Генеральная доля (P): 0.75
==========

ОЦЕНКА РАЗНОСТИ МЕЖДУ ВЫБОРОЧНОЙ И ГЕНЕРАЛЬНОЙ ДОЛЯМИ

Исходные данные:
Объем выборки (n): 100
Выборочная доля (p): 0.7
Генеральная доля (P): 0.75
Пошаговый расчет:

1. Расчет абсолютной разности долей (d):
   d = |p - P| = |0.7 - 0.75| = 0.050000

2. Расчет стандартной ошибки разности долей (md):
   md = √[P(1-P)/n] = √[0.75×0.250/100] = √[0.187500/100] = √0.001875 = 0.043301

3. Расчет значения t-критерия Стьюдента (t):
   t = d/md = 0.050000/0.043301 = 1.154701

4. Число степеней свободы (v):
   v = n - 1 = 100 - 1 = 99

5. Сравнение с критическими значениями:
   t(0.95) ≈ 1.96 (для α = 0.05)
   t(0.99) ≈ 2.58 (для α = 0.01)
   t(0.999) ≈ 3.29 (для α = 0.001)

РЕЗУЛЬТАТЫ:

Расчетное значение t-критерия: 1.154701
Разность не значима (p > 0.05)

ВЫВОД:

Нет достаточных оснований отвергать гипотезу о принадлежности исследуемой выборки к генеральной совокупности с долей P.

ПОРОГИ ВЕРОЯТНОСТИ:

- Большая ответственность: $\beta \geq 0.95$
- Обычная ответственность: $\beta \geq 0.99$
- Малая ответственность: $\beta \geq 0.999$

==========

Конец отчета

10. Инструкция пользователю по программе "Алгоритм № 19: Оценка разности между выборочной и генеральной долями"

Версия: 1.0

Авторы: (C°) Луценко Е.В., Трошин Л.П.

Дата: 2025 год

---

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программа предназначена для автоматизации расчетов по алгоритму № 19 "Оценка разности между выборочной и генеральной долями". Данный статистический метод позволяет:

- Оценить статистическую значимость разности между выборочной долей (p) и предполагаемой генеральной долей (P)
- Проверить гипотезу о принадлежности исследуемой выборки к определенной генеральной совокупности
- Использовать t-критерий Стьюдента для долей

2. СИСТЕМНЫЕ ТРЕБОВАНИЯ

- **Операционная система:** Windows 7/8/10/11 (32-bit или 64-bit)
- **Оперативная память:** не менее 256 МБ
- **Свободное место на диске:** не менее 50 МБ
- **Дополнительные требования:** Установка Python НЕ требуется (программа скомпилирована)

3. УСТАНОВКА И ЗАПУСК

3.1 Установка

1. Распакуйте архив с программой в любую папку на вашем компьютере
2. Убедитесь, что в папке с программой присутствуют следующие файлы:
    - Algorithm19.exe (исполняемый файл программы)
    - _Aidos.ico (иконка программы)
    - help-19.pdf (файл справки)

3.2 Запуск программы

1. Дважды щелкните по файлу Algorithm19.exe
2. Программа откроется в новом окне

4. ОПИСАНИЕ ИНТЕРФЕЙСА

4.1 Главное окно программы

Главное окно программы содержит следующие элементы:

Верхняя часть:

- Заголовок: "Исходные данные:"
- Кнопки выбора примера:
    - "Пример 1 (n=50, p=0.70, P=0.50)"
    - "Пример 2 (n=100, p=0.70, P=0.75)"
- Окно ввода исходных данных

Средняя часть:

- Кнопка "Расчет" (светло-зеленого цвета)

Нижняя часть:

- Заголовок: "Результаты расчетов:"
- Окно вывода результатов
- Кнопки управления:
    - "Помощь" (светло-голубого цвета)
    - "Записать отчет в виде файла" (светло-голубого цвета)

4.2 Окно ввода исходных данных

В верхнем окне необходимо ввести исходные данные в следующем формате:

Объем выборки (n): [значение]
Выборочная доля (p): [значение]
Генеральная доля (P): [значение]

Пример:
Объем выборки (n): 50
Выборочная доля (p): 0.70
Генеральная доля (P): 0.50

4.3 Окно результатов расчетов
В нижнем окне отображаются результаты вычислений с подробным пошаговым расчетом и статистическими выводами.

5. РАБОТА С ПРОГРАММОЙ

5.1 Быстрый старт (использование примеров)

1. При запуске программы автоматически загружается Пример 1
2. Для переключения на Пример 2 нажмите соответствующую кнопку
3. Нажмите кнопку "Расчет"
4. Изучите результаты в нижнем окне

5.2 Работа с собственными данными

1. Очистите верхнее окно от примера
2. Введите свои данные в требуемом формате:
    - Объем выборки (n) - целое положительное число
    - Выборочная доля (p) - число от 0 до 1
    - Генеральная доля (P) - число от 0 до 1
3. Нажмите кнопку "Расчет"
4. Проанализируйте результаты

5.3 Интерпретация результатов
Программа выполняет следующие расчеты:

1. **Абсолютная разность долей (d):** $|p - P|$
2. **Стандартная ошибка разности (md):** $\sqrt{[P(1-P)/n]}$
3. **t-критерий Стьюдента:** $t = d/md$
4. **Число степеней свободы:** $v = n - 1$
5. **Сравнение с критическими значениями** и формулировка вывода

Уровни значимости:

- **Не значима** ($p > 0.05$): Нет достаточных оснований отвергать гипотезу

- **Значима на уровне 0.05**: Большая ответственность (β ≥ 0.95)
- **Значима на уровне 0.01**: Обычная ответственность (β ≥ 0.99)
- **Значима на уровне 0.001**: Малая ответственность (β ≥ 0.999)

6. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИИ

6.1 Справка

- Нажмите кнопку "Помощь" для открытия файла help-19.pdf
- В справке содержится подробное описание алгоритма с формулами и примерами

6.2 Сохранение отчета

1. После выполнения расчетов нажмите кнопку "Записать отчет в виде файла"
2. В папке с программой будет создан текстовый файл с отчетом
3. Имя файла содержит дату и время создания
4. Отчет включает:
    - Исходные данные
    - Полные результаты расчетов
    - Дату и время выполнения

7. ВОЗМОЖНЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

7.1 Ошибки ввода данных

Сообщение: "Не удалось найти все необходимые данные: n, p, P" **Причина:** Неправильный формат ввода данных **Решение:** Проверьте формат ввода, используйте двоеточие после названий параметров

Сообщение: "Объем выборки должен быть больше 0" **Причина:** Введено нулевое или отрицательное значение n **Решение:** Введите положительное целое число для объема выборки

Сообщение: "Доли должны быть в диапазоне от 0 до 1" **Причина:** Значения p или P выходят за допустимые пределы **Решение:** Введите значения долей в диапазоне от 0.0 до 1.0

7.2 Ошибки файловой системы
Сообщение: "Файл справки 'help-19.pdf' не найден" **Причина:** Отсутствует файл справки в папке с программой **Решение:** Убедитесь, что файл help-19.pdf находится в той же папке, что и программа

7.3 Системные ошибки
Проблема: Программа не запускается **Возможные причины и решения:**

- Отсутствуют системные библиотеки: обновите Windows
- Антивирус блокирует программу: добавьте в исключения
- Недостаточно прав доступа: запустите от имени администратора

8. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

8.1 Пример 1: Проверка принадлежности выборки
Задача: Проверить, принадлежит ли выборка из 50 объектов с долей успехов 0.70 к генеральной совокупности с долей 0.50.
Исходные данные:
Объем выборки (n): 50
Выборочная доля (p): 0.70
Генеральная доля (P): 0.50
Результат: Разность статистически значима, выборка не принадлежит к данной генеральной совокупности.

8.2 Пример 2: Проверка гипотезы о доле
Задача: Проверить гипотезу о том, что генеральная доля составляет 0.75, если в выборке из 100 объектов доля составила 0.70.
Исходные данные:
Объем выборки (n): 100
Выборочная доля (p): 0.70
Генеральная доля (P): 0.75
Результат: Разность статистически не значима, гипотеза не опровергнута.

9. КОНТАКТНАЯ ИНФОРМАЦИЯ
При возникновении технических вопросов или предложений по улучшению программы обращайтесь к авторам:

---

10. ПРИЛОЖЕНИЯ

Приложение А. Формулы алгоритма

Основная формула t-критерия:

$t = |p - P| / \sqrt{[P(1-P)/n]}$

Где:

- p - выборочная доля
- P - генеральная доля
- n - объем выборки
- v = n - 1 (степени свободы)

Приложение Б. Критические значения t-критерия

Уровни значимости (приблизительные значения для больших выборок):

Уровень значимости (α)	Доверительная вероятность (β)	Критическое значение t
0.05	0.95	1.96
0.01	0.99	2.58
0.001	0.999	3.29

Приложение В. Структура отчета

Автоматически генерируемый отчет содержит:

1. **Заголовок отчета** с номером и названием алгоритма
2. **Дату и время** выполнения расчетов
3. **Информацию о программе** и авторах
4. **Исходные данные** в том виде, как они были введены
5. **Подробные результаты расчетов** включая:
    - Пошаговые вычисления
    - Промежуточные результаты
    - Статистические выводы
    - Интерпретацию результатов

Пример имени файла отчета: Алгоритм_19_Оценка_разности_между_выборочной_и_генеральной_долями_20250725_143052.txt

Приложение Г. Рекомендации по использованию

Для получения корректных результатов рекомендуется:

1. **Проверка исходных данных:**
   - Убедитесь, что объем выборки достаточно велик (n > 30)
   - Проверьте, что значения долей находятся в диапазоне [0, 1]
   - Значения долей должны быть реалистичными для вашей задачи
2. **Интерпретация результатов:**
   - Обращайте внимание на уровень значимости
   - Учитывайте практическую значимость различий
   - Помните о том, что статистическая значимость не всегда означает практическую важность
3. **Документирование работы:**
   - Сохраняйте отчеты для дальнейшего анализа
   - Ведите учет различных вариантов расчетов
   - Фиксируйте предположения и ограничения

ВНИМАНИЕ: Данная программа предназначена для образовательных и научных целей. При принятии важных решений на основе статистических расчетов рекомендуется консультация со специалистами в области статистики.

СОДЕРЖАНИЕ

ЛИТЕРАТУРА

1. Плохинский Н. А. Алгоритмы биометрии. Под ред. академика АН УССР Б. В. Гнеденко. М., Изд-во Моск, ун-та. 1980, 21004, 150 с., http://lc.kubagro.ru/Algorithms_of_biometrics-N_A_Plokhinsky-1980.pdf
2. Выбор эталонов при определении коэффициентов наследуемости у Vitis vinifera L. / П.Я. Голодрига, Л.П. Трошин, А.П. Петров, И.Д. Соколов // Тр. по прикл. ботанике, генетике и селекции / ВАСХНИЛ. ВИР им. Н.И. Вавилова. - Л., 1975. - Т. 54, вып. 2. - С. 128-131.
3. Голодрига П.Я., Трошин Л.П., Петров А.П. (при участии И.Д. Соколова). О связи уровня паратипической изменчивости с величиной среднего значения некоторых признаков у винограда // Цитология и генетика. - 1974. - № 3. - С. 248-252.
4. Соколов И.Д., Соколова Е.И., Трошин Л.П. и др. Введение в биометрию. – Краснодар: КубГАУ, 2016. – 245 с.
5. Соколов И.Д., Соколова Е.И., Трошин Л.П. и др. Биометрия. – Краснодар: КубГАУ, 2018. – 161 с.
6. Трошин Л.П. Введение в ампелометрию. – Краснодар: КубГАУ, 2019. – 296 с. (при последней встрече 29.09.1999 г. И.Д. подсказал эту идею).
7. Биометрия: практикум / Е.И.Соколова, И.Д.Соколов, Л.П.Трошин [и др.]. – Краснодар: КубГАУ, 2019. – 180 с.
8. Трошин, Л. П. Ампелометрия : учебное пособие / Л. П. Трошин, Р. Н. Куфанова, Е. В. Луценко. – Краснодар : Кубанский государственный аграрный университет имени И. Т. Трубилина, 2025. – 240 с. – ISBN 978-5-93856-913-3. – DOI 10.13140/RG.2.2.23509.95201.
9. Луценко, Е. В. Когнитивное моделирование в ампелографии / Е. В. Луценко, Л. П. Трошин. – Краснодар : Кубанский государственный аграрный университет им. И.Т. Трубилина (Краснодар), 2024. – 153 с. – ISBN 978-5-93856-883-9. – DOI 10.13140/RG.2.2.28319.06566. – EDN HJMPLB.
10. Луценко, Е. В. Количественное измерение сходства-различия клонов винограда по контурам листьев с применением АСК-анализа и системы "Эйдос" / Е. В. Луценко, Л. П. Трошин, Д. К. Бандык // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. – 2016. – № 116. – С. 1205-1228. – EDN VQUVXN.
11. Луценко, Е. В. Применение теории информации и когнитивных технологий для решения задач генетики (на примере вычисления количества информации в генах о признаках и свойствах различных автохтонных сортов винограда) / Е. В. Луценко, Л. П. Трошин // Политематический сетевой электронный научный журнал Кубанского

государственного аграрного университета. – 2016. – № 121. – С. 116-165. – DOI 10.21515/1990-4665-121-003. – EDN WWSKQV.

12. Луценко, Е. В. Решение задач ампелографии с применением АСК-анализа изображений листьев по их внешним контурам (обобщение, абстрагирование, классификация и идентификация) / Е. В. Луценко, Д. К. Бандык, Л. П. Трошин // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. – 2015. – № 112. – С. 862-910. – EDN UZEBTF.

13. Биометрическая оценка полиморфизма сортогрупп винограда Пино и Рислинг по морфологическим признакам листьев среднего яруса кроны / Л. П. Трошин, Е. В. Луценко, П. П. Подваленко, А. С. Звягин // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. – 2009. – № 52. – С. 181-194. – EDN JUOVHG.

14. Биометрическая оценка морфологических признаков популяции Каберне-Совиньон / А.С.Звягин, Л.П. Трошин, П.П.Подваленко, В.И.Вернигоров // Критерии и принципы формирования высокопродуктивного виноградарства. – Анапа, 2007. – С. 203-207.

15. Звягин А.С., Трошин Л.П. Биометрический анализ урожайности сортогруппы Каберне-Совиньон в Центральной зоне Кубани // Студенчество и наука. – Краснодар, 2007. - С. 167-169.

16. Звягин А.С., Трошин Л.П., Подваленко П.П. Биометрическая оценка морфологических признаков популяции Мерло // Критерии и принципы формирования высокопродуктивного виноградарства. – Анапа, 2007. – С. 165-172.

17. Трошин Л.П. Использование биометрической оценки морфологических признаков клонов для идентификации генотипов сортогруппы Мерло / Л.П. Трошин, А.С. Звягин, Д. Сидоренко // Научный журнал КубГАУ [Электронный ресурс]. – Краснодар: КубГАУ, 2008. – №04(38). – Шифр Информрегистра: 0420800012\0053. – Режим доступа: http://ej.kubagro.ru/2008/04/pdf/10.pdf.

18. Биометрическая оценка полиморфизма сортогрупп винограда Пино и Рислинг по морфологическим признакам листьев среднего яруса кроны / Л.П. Трошин, Е.В. Луценко, П.П. Подваленко, А.С. Звягин // Научный журнал КубГАУ [Электронный ресурс]. – Краснодар: КубГАУ, 2009. – № 08 (52). – Режим доступа: http://ej.kubagro.ru/2009/08/pdf/01.pdf.

19. Трошин Л.П. Рабочая тетрадь для лабораторно-практических занятий по генетике. - Краснодар: КГАУ, 2010. – 43 с.

20. Трошин Л.П. Морфометрический анализ листовой ампелографической информации // Виноделие и виноградарство. – 2011. - № 3. – С. 48-49; - № 4. – С. 47-49.

Учебное издание

Луценко Евгений Вениаминович
Трошин Леонид Петрович

АЛГОРИТМЫ АМПЕЛОМЕТРИИ

Том-2.
Алгоритмы 12-19: Репрезентативность выборочных показателей

Учебное пособие

В авторской редакции
Компьютерная верстка – Е. В. Луценко
Макет обложки – Е. В. Луценко

Подписано в печать 17.02.2025. Формат 60 х 84 1/16
Усл. печ.л. – 12,846. Уч.-изд.л. – 9,208.

Кубанский госуларственный
аграрный университет им. И.Т. Трубилина
350044, г. Краснодар, ул. Калинина, 13