ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

Сравнение эффективности работы в клиентах системы версионного контроля Git

Выполнили
Студент группы 152 ПИ
Р. Р. Абузяров
Студентка группы 152 ПИ
К. А. Третьякова

Москва 2018

Содержание

1 Введение

Для сравнения были выбраны два интерфейса, позволяющие использовать систему версионного контроля Git: Git Command Line Interface и официальный клиент от компании GitHub - GitHub Desktop.
Цель эксперимента - узнать, какой из выбранных клиентов позволяет пользователю работать более эффективно, и быстрее выполнять поставленные задачи.

2 План эксперимента

1. Для участия в эксперименте набираем 12 испытуемых, имеющих базовые представления о работе с системой Git.

2. В соответствии с within-subject design разбиваем участников эксперимента на две группы.

3. Предлагаем участникам пройти набор заданий последовательно на двух клиентах, порядок выполнения определяется принадлежностью к одной из двух групп.

4. Во время выполнения задания производится запись экрана устройства испытуемого.

5. Из полученных записей экрана определяется время выполнения заданий.

6. Также мы просим пользователей оценить их опыт работы с каждым клиентом по шкале от 0 до 3.

7. Оцениваем различие в скорости работы в разных клиентах с помощью t-критерия для двух независимых выборок.

8. Считаем корреляцию Пирсона для оценки влияния опыта на скорость работы.

3 Гипотеза

Нулевая гипотеза – различие в скорости работы в разных клиентах Git статистически не значимо ($M_{GUI} = M_{CLI}$).
Альтернативная гипотеза – клиент GitHub Desktop позволяет выполнять поставленные задачи быстрее, чем Git Command Line Interface ($M_{GUI} < M_{CLI}$).

4 Переменные

Независимые переменные – клиент, в котором выполняется задание (GitHub Desktop или Git CLI).
Зависимые переменные – время выполнения набора заданий в секундах, опыт работы в том или ином клиенте по шкале от 0 до 3.
Контрольные переменные – уровень подготовки, участники должны иметь базовые представления о работе с системой Git.

5 Процедура

5.1 Перед началом работы

1. Перед началом выполнения задания создайте папку с названием `git-compare` и перейдите в нее командой `cd path-to-git-compare`.

2. Создавать файлы можно любым наиболее удобным для вас способом (в терминале это можно сделать с помощью команды `touch file.txt`, которая создаст файл в текущей директории).

3. В процессе выполнения задания можно гуглить, при этом не нужно останавливать запись экрана.

4. Порядок использования клиентов вам сообщён отдельно, придерживайтесь его, так как это важно для статистического исследования.

5. Задания направлены не на или проверку навыков работы с системой Git, а на сравнение эффективности работы в разных клиентах.

6. Не прерывайте выполнение задания (между клиентами можно) и не выполняйте его больше одного раза, важен первый результат.

5.2 Задание

1. Создайте локальный репозиторий с названием `compare-git-gui`, если вы выполняете задание через клиент от GitHub, или репозиторий с названием `compare-git-terminal`, если выполняете задание через терминал.

2. Создайте файл с названием `index.txt` в этом репозитории.

3. Скопируйте в этот файл следующий текст:

   ```
   This is my initial commit
   ```

4. Закоммитьте изменения с описанием коммита `Initial commit`.

5. Создайте новую ветку `feature-1` и перейдите в нее.

6. Дополните файл `index.txt` следующей строкой:

   ```
   This is my feature commit
   ```

   После этого он должен иметь такой вид:

   ```
   This is my initial commit
   This is my feature commit
   ```

7. Создайте файл `file.txt` и скопируйте в него текст:

   ```
   This is a new file
   ```

8. Закоммитьте файл `index.txt` с описанием коммита `Index changes`.

9. Закоммитьте файл `file.txt` с описанием коммита `New file created`.

10. Перейдите в ветку `master` и вмержите ветку `feature-1` в `master`.

11. Добавьте в файл `index.txt` следующую строку:

    ```
    c00L mast3r c0mmit
    ```

    После этого он должен иметь такой вид:

    ```
    This is my initial commit
    This is my feature commit
    c00L mast3r c0mmit
    ```

12. Закоммитьте изменения с описанием коммита `Added c0oL stuff`.

13. Перейдите в ветку `feature-1` и добавьте в файл `index.txt` следующую строку:

    ```
    Boring feature commit.
    ```

    После этого он должен иметь такой вид:

```
This is my initial commit
This is my feature commit
Boring feature commit.
```

14. Закоммитьте изменения с описанием коммита `Added boring stuff`.

15. Вмержите ветку `master` в ветку `feature-1` и разрешите конфликт так, чтобы файл `index.txt` выглядел следующим образом:

```
This is my initial commit
This is my feature commit
Boring feature commit.
c00L mast3r c0mmit
```

16. Закоммитьте изменения.

6 Результаты

6.1 t-критерий, основная выборка

В результате эксперимента, в котором было задействовано 12 человек, были собраны следующие данные:

Время GUI (сек)	**Время CLI (сек)**	**Опыт GUI (0-3)**	**Опыт CLI (0-3)**
348	335	2	3
587	919	1	2
450	993	2	1
452	921	2	3
456	440	2	3
863	613	2	2
978	858	1	3
420	840	2	2
832	1401	2	1
493	340	2	3
420	261	1	3
332	276	1	3

Таблица 1. Основная выборка

	Минимум	**Максимум**	**Среднее**	**Отклонение**
GUI	332	978	553	217
CLI	261	1401	683	360

Таблица 2. Агрегированные данные основной выборки

Посчитаем t-статистику для проверки гипотезы по формуле

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}},$$

где $\overline{X}$ – среднее выборочное,
s^2 – несмещенная оценка дисперсии выборки,
n – количество элементов в выборке;
получим

$$t = 1.0760531414398307$$

Критическое значение на уровне значимости 0.05

$$T_{0.05,22} = 2.0738730679040147$$

Рассчитанное значение t меньше критического, поэтому делаем выбор в пользу нулевой гипотезы.
Используя вычислительные средства рассчитаем p-значение

$$p = 0.293562912282017$$

Значит нулевая гипотеза будет отвергнута на уровне значимости большем, чем 0.293562912282017.

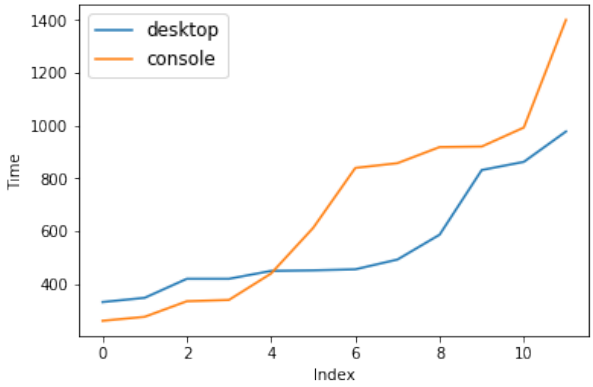

Рис. 1. Отсортированные данные

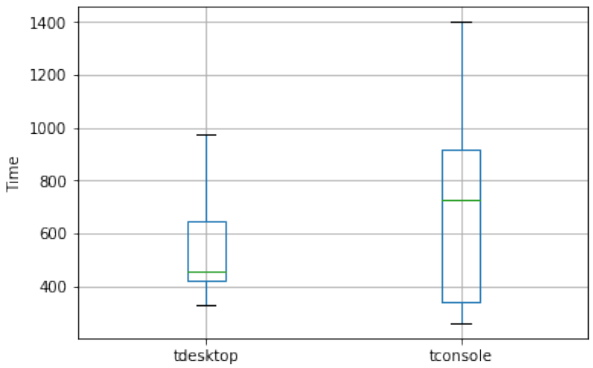

Рис. 2. Диаграмма размаха

6.2 t-критерий, альтернативная выборка

Используя данные об опыте работы участников эксперимента с разными клиентами системы Git, сформируем выборку, учитывая только участников, у которых разница в опыте между клиентами не превышает 1 балла. Полученная выборка содержит 9 участников:

Время GUI (сек)	Время CLI (сек)	Опыт GUI (0-3)	Опыт CLI (0-3)
348	335	2	3
587	919	1	2
450	993	2	1
452	921	2	3
456	440	2	3
863	613	2	2
420	840	2	2
832	1401	2	1
493	340	2	3

Таблица 3. Альтернативная выборка

	Минимум	Максимум	Среднее	Отклонение
GUI	348	978	545	183
CLI	335	1401	756	354

Таблица 4. Агрегированные данные альтернативной выборки

Посчитаем t-статистику для проверки гипотезы

$$t = 1.5893693291646498$$

Критическое значение на уровне значимости 0.05

$$T_{0.05,18} = 2.1199052992210112$$

Рассчитанное значение t также меньше критического, однако взглянем на p-значение

$$p = 0.1315394589256398$$

То есть нулевая гипотеза будет отвергнута на уровне значимости 0.1315394589256398, что гораздо ниже, чем для основной выборки, но различия во времени прохождения заданий все еще статистически не значимы.

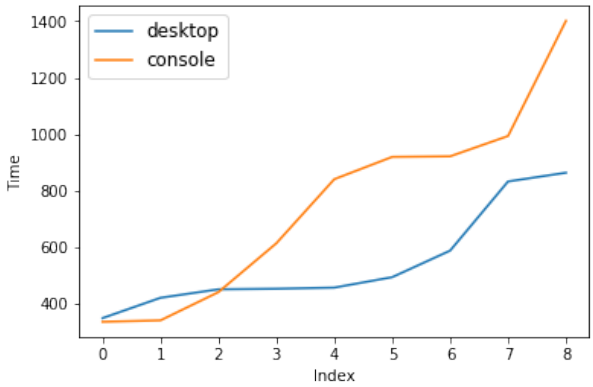

Рис. 3. Отсортированные данные

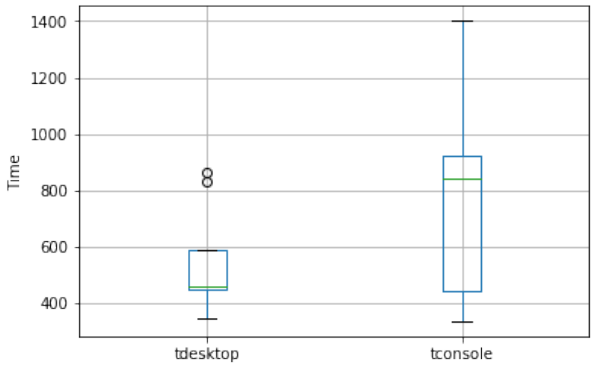

Рис. 4. Диаграмма размаха

6.3 Корреляция Пирсона

Вычислим матрицу корреляции Пирсона для основной выборки:

	tdesktop	**tconsole**	**expdesktop**	**expconsole**
tdesktop	1	0.445065	**-0.086964**	-0.509855
tconsole	0.445065	1	-0.172809	**-0.791743**
expdesktop	0.086964	-0.172809	1	0.1
expconsole	-0.509855	-0.791743	0.1	1

Таблица 5. Матрица корреляции Пирсона

Обратим внимание на выделенные коэффициенты. Можем заметить, что опыт работы в клиенте влияет на скорость выполнения заданий в этом клиенте (отрицательная корреляция обозначает, что чем больше опыт, тем меньше времени на выполнение заданий). Также заметим, что опыт работы в CLI имеет гораздо большее влияние на скорость прохождения заданий, чем опыт работы в GUI, что может свидетельствовать о том, что с увеличением опыта работы в CLI можно достичь гораздо большей эффективности.

7 Заключение

Таким образом, различия в скорости работы между клиентами системы Git нельзя считать статистически значимыми. Однако можно сделать вывод о том, Git Command Line Interface более чувствителен к опыту пользователя, опытные пользователи могут достичь большей скорости и эффективности работы с его помощью. Также стоит упомянуть, что многие функции системы Git (такие как rebase, stash и другие) доступны только в Git Command Line Interface.