UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Stenver Jerkku

Paralell Wilcoxon Signed-rank tests

Bachelor's Thesis (6 ECTS)

Supervisor: Sven Laur, PhD

Author: ..... "....." 2012

Supervisor: ..... "....." 2012

Allowed to defence
Professor: ..... "....." 2012

Tartu 2012

Contents

Abstract

Wilcoxon Signed-rank test is a paired statistical test that is used when measurement values are not normally distributed. The test is used by BIIT(Bioinformatics, Algorithmics and Data mining group) research group for gene regulation, gene expression data analysis, biological data mining and others. BIIT is joint research group between the Department of Computer Science (University of Tartu), Quretec, and the Estonian Biocenter.

The current implementations of the Wilcoxon Signed-rank tests are slow and unoptimized. This project will look into the foundations of wilcoxon signed-rank test, its current implementations and how to optimize it. In order to make the implementation more accurate, the relationship between Wilcoxon statistic and Guassian approximate will be observed. In order to make the implementation faster, some dynamic programming methods will be used to save computation time. The goal of optimizing is to make it more accurate and speed up the test running.

The end goal of this project is to create an accurate and fast wilcoxon test implementation in C++ shared library. In the scope of this project, the library will be integrated with two tools -command line and Cron-R. Due to the nature of shared library, it will be easy integrate the library with any other tools one might desire.

1 Introduction

The BIIT research group has biologist who conduct a variety of experiments on a control group. They try to measure, compare and test different attributes of a living organism. These attributes might be, but not limited to blood pressure, proteine amount in the blood, RNA amount, purity of urine, brainwaves etc. These attributes can be repeadedly measured on the same subject and the results are never exactly the same. There is always a little noise and varience between the samples. Biologist often experiment on the control group and try to affect these observed attributes. The take samples before and after simulating the observable attribute with some stimulant. This is called the case-control study.

However, since even without external stimulation the two results are never the same. Also, different experiments can have wildly different assumptions, so you cannot simply use a constant error margin on all your tests. For example, the blood pressure measuring before and after jogging could have a lot bigger changes then brainwave changes. Therefore the biologist need to use statistical tests to find out which case-control studies are actually significant and which are simply noise.

There are many case-control tests available. For example paired Students t-test, t-test for matched pairs and Wilcoxon signed-rank test. The Wilcoxon signed-rank test is used when the measurement values are not normally distributed when the experimental conditions are fixed. In practise, this means that the histogram of measurements is either asymmetrical or it does not resemble the bell shape.

It is common to assume that properly scaled gene expression measurements follow Gaussian distribution, while quantities of proteins and metabolites are assumed to have non-gaussian distribution.

1.1 NetCDF file format

The BIIT group holds statistical data in NetCDF file format. NetCDF is an open source standard of a set of data formats, programming interfaces, and software libraries that help read and write scientific data files. `http://www.unidata.ucar.edu/software/netcdf/docs/what_is_netcdf.html`[2] Data can be held in a structured manner in a NetCDF file. You can define dimensions, name them, put variables in the dimensions and later retrieve or change them. This is useful, for example, to hold matrix like data in a file and have fast lookups. In addition, you can define helper dimensions for the matrix for extra information. NetCDF file format is used by the BIIT researh group to hold gene data and will be given as an input file for the program.

Currently the BIIT group holds the data in the NetCDF file in the following format:

Data matrix

- double data(m, n) - the data matrix of m genes (variables) and n samples (experiments, patients, ...)
- string gene[m] - gene IDs
- string array[n] - sample IDs

Non-track metadata

- string __FileFormat - EVDF format identifier string. Current value: "ExpressView 1.1".
- string __DatasetType - A string identifying the dataset type. This is used to extract data-specific parameters from a global configuration file. See datatypes.conf (ExpressViewConfigFiles) for allowed values.
- string[n] Organism - Organism (per column)
- string[n] MetadataOrder - Sometimes metadata (including column tracks) may be in a different order than columns in data. This variable is a permutation of the array variable, specifying the order of column variables. This field is deprecated and all column metadata is expected to be in array order in new datasets.
- string InvestigationTitle - short title for the dataset
- string ExperimentDescription - long description for the dataset
- string DatasetID - Optional Source-specific dataset identifier
- string DatasetLink - Optional A URL associated with the dataset
- string __VariableLabel - Optional Custom variable type name for the dataset. Overriden by variable_label in project configuration. (ExpressView) Names of any other (optional, non-track) metadata variables must be prepended by two underscores. Additional variables may be mandated by a data type specification, see ExpressViewConfigFiles. Known specific variables:
- string __Organism - ArrayExpress chip organism (ArrayExpress; as opposed to the source of biological material stored in Organism)

Tracks

Tracks are all other variables whose name begins with an uppercase character, that are arrays of either string or double, with the size of their first dimension either m or n. The former are referred to asrow tracks and the latter as column tracks. These variables encode some information about either the rows or columns of the data matrix; e.g. ExpressView displays column tracks above the heatmap.

Examples:

string Chemotherapy[n] string Local Relapse[n] double RelativeVariance[m]

1.2 Current BIIT procedures

Currently the BIIT group has 3 ways of analysing its data.

1.2.1 GNU R

They can use GNU R project to make custom and complext analyses. For that they need to manually read NetCDF file to the R and then call the R built in function wilcox.test to use Wilcoxon test. The problem with R wilcoxon test is that it is slow. If you run thousands of tests in a row, the results might come after hours of computer processing. One of the goals of this paper and project is to make Wilcoxon test run on command line in mere seconds.

1.2.2 Web interface

They can use a web interface which can take in certain arguments and command line script. It then will use whatever command line tools it has available and visualizes the output.

1.2.3 Command line

They can also use the command line directly which has a variety of tools installed. The wilcoxon test will ultimately call R wilcoxon function.

So in the end - no matter how the biologist analyse their data, they will always use wilcoxon test in the GNU R project.

2 Wilcoxon signed-rank test

In this section, we discuss a particular statistical test called Wilcoxon signed-ran test. Wilcoxon test can be used to make sure that something has an effect on the measured samples. For example, biologist could measure 3 patients white blood cell levels, create some external stimulation on the patients and measure the white blood cell levels again. The wilcoxon signed-ranked test shows wether the external stimulation had any statistical effect on the white blood cell levels.

2.1 Statistical Tests

The statistical tests are usually done, because initial research has raised a hypothesis about the data. It is desireable to know if the hypothesis is true or not.

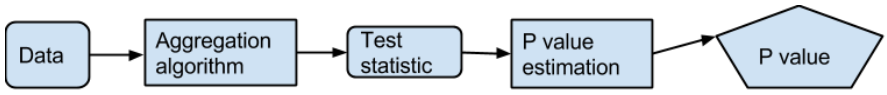

Figure 1: Graphical represenation of statistical test flow

Statistical tests are usually built by first gathering the necessary raw data. For example, make 80 persons run for 30 minutes and measure blood pressure before and after running. The data is then organized by some kind of aggregation algorithm which also might filter out samples that are not related to the hypothesis in question. For example, sorting and grouping the blood pressure results by gender, weight and in ascending order and then filter out any group that had too few samples. Then a test statistic is chosen and calculated. After that a p-value can be estimated which tells if the observable feature is interesting or not. For example, if running plays a role in raising a persons blood pressure.

2.1.1 Test Statistic

The test statistic is a scalar function of all the observations, which summarizes the data by a single number. This value is used to estimate a p-value and accept or reject a null hypothesis. Commonly, the test statistic is either one-sample, two-sample or paired statistic, depending on the tests.

A one-sample test tests whether the data collectively satisfies some hypothesis, e.g. the data is generated by gaussian distribution.

A two-sample test compates two sub-populations of data, e.g. persons with high blood persons and persons with low blood pressures.

A paired test characterises the change of before and after measurements, e.g. does some new medical drug treatment work. The differences between measured

observable case and control experiment pairs comes from the same distribution. The distribution of differences are also symmetrical. A positive difference x is equally likely as a negative difference $-x$.

2.2 Hypothesis

2.2.1 Null hypothesis

In statistics, a null and alternative hypothesis are raised to find out if the test bears any interesting results. A null hypothesis in general terms means that the test results are boring or that there is no effect, or in more formal terms, general position. It means there is no relationship between the two observed features. For example, when measuring blood pressure before and after running, the null hypothesis would state that blood pressure will not change when running.

The null hypothesis also defines the data distribution. For example, it might define that data is generated by a Gaussian distribution. Since null hypothesis is the general position, then generally the data will be distributed according to the null hypothesis. Since the null hypothesis defines the data distribution, it will also define the test statistic distribution. Finally, since the null hypothesis defines the data distribution and test statistic distribution, then it will also play a role in estimating the p-value for the test statistic.

For example, if the observations $x_1, x_2, ..., x_n$ are generate by coin flip, then the test statistic $T = x_1 + x_2 + .. + x_n$ has a binominal distribution $Binominal_{n,\frac{1}{2}}$. If test statistic is defined as $T = x_1 * (1 - x_1) + x_2 * (1 - x_2) + ... + x_n * (1 - x_n)$, then it is constantly zero. The exact form of the test statistic determines the properties of the test.

2.2.2 Alternative hypothesis

The alternative hypothesis is the opposite to the null hypothesis - in simple terms, one could think of it as true or interesting result. It means that the two measured samples have a relationship between them, either in direct or indirect ways. A good example to explain these two is the court verdict. If null hypothesis is kept, then the suspect has not been found quilty. There was not enough evidence. On the contrary, if null hypothesis was rejected, the suspect was found quilty, because there was enough evidence to make that decision. Although the alternative hypothesis could be anything that is not a null hypothesiss, some alternatives to the null hypothesis are easier to distinguish from the null hypothesis.

When the null hypothesis is true, then the test statistic has a well-defined probability distribution. This means that one can predict pretty well the possible outcome of a measurable subset of samples. For example, the alternative

hypothesis can be separated from the null hypothesis if the distribution of the test statistic is very different from the null hypothesis. The alternative hypothesis corresponding to the null hypothesis causes the test statistic to be at the edges of a probability distributions, which makes them hard to predict.

2.2.3 P-value

P-value is estimated from the test statistic. It is used to find out if the test data is interesting. A certain threshold, called significance level (usually 0.05) is taken for the p-value. If the p-value is within significance level, then null hypothesis is rejected. If the p-value is not within significance level, then null hypothesis is kept and the data falls into distribution as defined by null hypothesis. A variety of algorithms have been thought of for p-value, depending on the type of data, goals of the tests and even the amount of data. Statistics need to make the right choice between the alorithms by taking all these assumptions into an account.

The p-value can be either one sided or two sided. One-sided p-value is the probability that you get a certain value T that is larger or equal to the computed test statistic from the observed data in the distribution that the null-hypothesis defines. This can be seen visually in Figure 2.

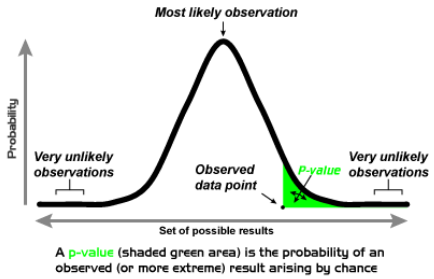

Figure 2: One sided p-value representation (Author Repapetilto, Wikipedia Foundation Inc)

Two-sided p-value shows the probability that you get a certain value T that is either larger or smaller than the computed test statistic from the observed data in the distribution that the null-hypothesis defines. This can be seen visually in the Figure 3.

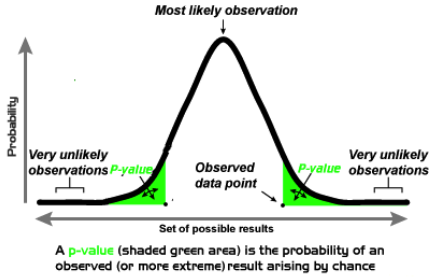

Figure 3: Two sided p-value representation (Modified by author, source from Wikipedia Foundation inc, Repapetilto)

2.3 Wilcoxon test

The Wilcoxon signed-rank test is a statistical hypothesis test that is used when comparing two related samples, matched samples, or repeated measurements on a single sample. This means that with Wilcoxon test you can look at some measurable feature, e.g. blood pressure. If two or more experiments in different conditions have been made on it,e.g. gene experiments and their confirmation results, Wilcoxon test can be used to see if the changes of the measurement levels are relevant. The null hypothesis H_0 means that the difference between pairs is zero. The alternative H_1 means it is not.

It was popularized by Sidney Siegel in his book "Nonparametric statistics - for the behavioral sciences". Sidney used the symbol T, to denote test statistic. Because of this, the test is sometimes referred to as the Wilcoxon T test and the test statistic is reported as a value of T.

2.4 Wilcoxon test Assumptions

1. Two measurements can always be compare and thus ordered. For example, measurement is a real value or ordered categorical values

2. Measurements can be naturally paired into case-control or before-after experiments. For example, measure patients white blood cell levels before and after treatment.

3. All measurement pairs are independent from other pairs. For example, measurement of one individual does not influence the other. In medicine patients in the study are sampled randomly. There is no evident procedure in selecting patients or no planned action to organize a study so one could get the "desired" result.

2.5 How to compute

Let N be the sample size, the number of pairs. Then we can use the following variable to compute the W test. For $i = 1, ..., N$, let $x_{1,i}$ and $x_{2,i}$ denote the measurements. Let W be the wilcoxon test statistic. Let $z-score$ be the wilcoxon test standard score. Let N_0 be the number of pairs from which onward one can approximate the p-value. This value is currently undefined and is the focus of this paper. It will be discussed in the future chapters.

The computation of the statistic W is organised into five steps. The last step diverges into two possible paths and the choice of the path that is chosen depends wether $N >= N_0$.

1. For $i = 1, .., N$, calculate $|x_{2,i} - x_{1,i}|$ and $\text{sign}(x_{2,1} - x_{1,i})$, where $\text{sign}(x)$ is the sign function.

2. Order the N pairs from smallest absolute difference to largest absolute difference, $|x_{2,i} - x_{1,i}|$.

3. Rank the pairs, starting with the smallest as 1. Ties receive a rank equal to the average of the ranks they span. Let R_i denote the rank.

4. Calculate the test statistic W, the absolute value of the sum of the signed ranks.
$$W = |\sum_{i=1}^{N} [\text{sign}(x_{2,i} - x_{1,i}) * R_i]| \tag{1}$$

5. As N increases, the sampling distribution of W converges to a gaussian distribution. Where N_0 depends on how accurate you want your results to be.

(a) For $N >= N_0$, a $z - score$ can be calculated as

$$z = \frac{W - 0.5}{\sigma_w}, \sigma_w = \sqrt{\frac{N(N+1)(2N+1)}{6}} \tag{2}$$

If $Z > Zcritical$ then reject H_0, where $Z_{critical}$ is calculated with Gaussian distribution.

(b) For $N < N_0$, W is compared to a $p - value$ that can be calculated from enumeration of all possible combinations of W given N.
If $W >= P_{critical}$, N then reject H_0

2.6 Example

Given pairs $(6, 8), ((2, -3), (-3, 3), (1, 3)$.

1. Calculate absolute values and signs:
   Absolute values $(2, -5, 6, 2)$
   Signs $(1, -1, 1, 1)$

2. Order the pairs. Ties receive therank equal to average of the ranks they span
   $(2 \Rightarrow 1.5), (5 \Rightarrow 3), (6 \Rightarrow 4), (2 \Rightarrow 1.5)$

3. Calculate the test statistic W
   $W = 1 * 1.5 + 1 * 1.5 + (-1 * 3) + 1 * 4 = 4.0$

4. Since N_r is very small, use a table to look up the p value. The calculation of p table is in a later chapter.
   $P(5, 4) = 0.3125$
   Since $0.3125 > 0.05$, reject H_1

The calculations can also be seen on Table 1 and Table 2.

i	$x_{2,1}$	$x_{1,i}$	sign	abs
1	6	8	1	2
2	2	-3	-1	5
3	-3	3	1	6
4	1	3	11	2

Table 1: Initial data

i	$x_{2,1}$	$x_{1,i}$	sign	abs	R_i	sign $* R_i$
1	6	8	1	2	1.5	1.5
4	1	3	11	2	1.5	1.5
2	2	-3	-1	5	3	-3
3	-3	3	1	6	4	4

Table 2: After ordering by absolute difference

2.7 P-value assignment

Let us consider Wilcoxon test if we have 6 measurements. Then there are three differences as shown in in Figure 4. As the test considers only the sign and order of the measurements, then only 8 configurations are possible as shown in Figure 5 and Table 3.

The formula to calculate that result was 2^n. Due to the symmetry of of the possible configurations under the distribution of the null hypothesis, all 8 configurations can be proven equally probable.

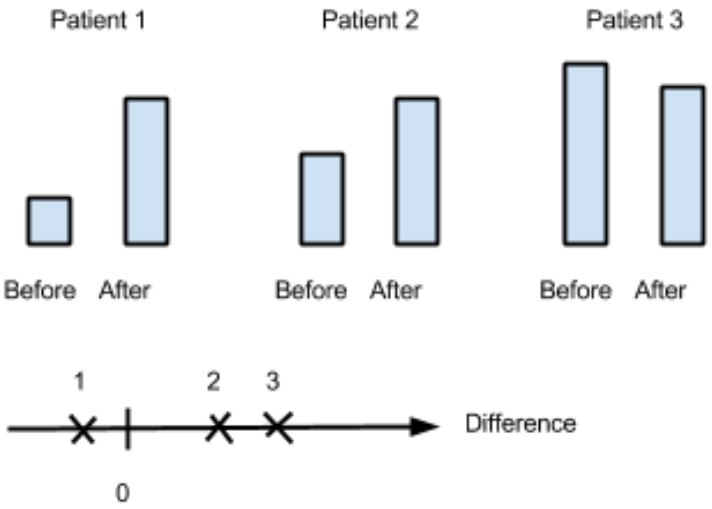

Figure 4: Patient example

Now if the W value in our data is $+2$, then there are 3 configurations that have larger or equal W value. This can be seen from the Table 3. Then the one-sided p-value is $\frac{3}{8}$. The two-sided p-value is $\frac{3}{4}$. The same procedure is needed to assign p-value.

So in the end, since Wilcoxon test neglets the actual value of the tests, then to estimate the p-value, we only need to know the W value and number of tests. No matter the actual value of the test cases, W statistic will always create the same configurations, given a certain N.

2.7.1 V and P example

Given N is 2, then ranks are $0, 1, 2$, one can combine them into $0+1+2$ or $0+1-2$ or $0-1+2$ or $0-1-2$. This gives us one way to get 3, one way to get 1, one

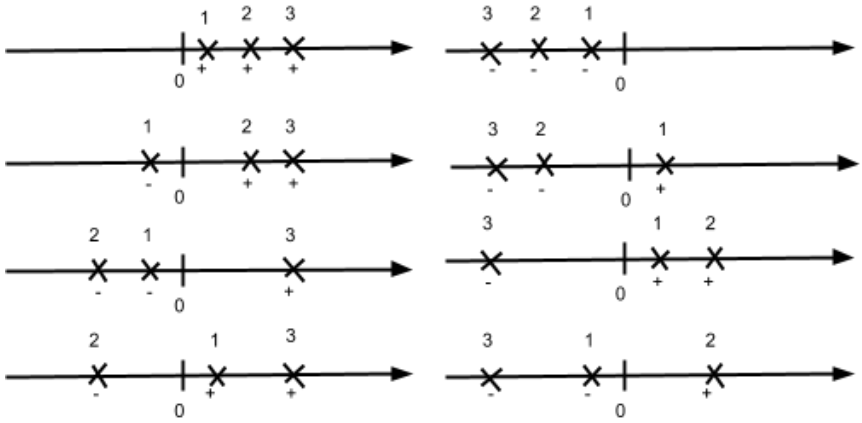

Figure 5: Two sided p-value representation

$W = 1 + 2 + 3 = 6$	$W = -1 - 2 - 3 = -6$
$W = -1 + 2 + 3 = 4$	$W = 1 - 2 - 3 = -4$
$W = -1 - 2 + 3 = 0$	$W = 1 + 2 - 3 = 0$
$W = 1 - 2 + 3 = 2$	$W = -1 + 2 - 3 = -2$

Table 3: All eight configurations. Values of the test statistic corresponding to the configuration.

way to get -1 and one way to get -3

It can now be calculated that the probability that our there is a 0.25% that our value is 3 or lower and 0.5% that our value is 1 or lower.

This distribution is the most accurate table you can compare wilcoxon test results on - it shows the exact probability that your tests falls into a certain gaussian distribution range. However, the table is expensive to calculate.

2.7.2 Exact computation of P-value

The problem of assigning a p-value can be reduced to the following problem. For example, we can find out what is the probability that $W = k$ or $W \geq k$. Given a N ranks, we assign $+$ or - sign to each rank randomly. Since the sign distribution is random, then either sign has equal probability to be assigned. For fixed number of ranks N, we can consider all 2^n possible sign assignements and formulate the number of combinations that lead to the $W \geq k$. A V-table is formed. It signifies all the possible random signed rank combination sums in an N sized ranked array,

where k shows how many times a certain sum was achived. Let $V_{N,k}$ be the number of times a certain sums was achieved.

From this table, the probabaility that $W \geq k$ can be calculated. This value will show how probable it is that the given N ranks with random signs, how probable it is that a sum k is the result. If this process of calculating values is done repeadedly for every possible N and k, then a P-table forms. The P-table shows the probability of every possible sum k appearing given N ranks with random signs. Let $P_{N,k}$ be the function to get the desired P value from that table.

If the P-table gets large enough, it will start taking the shape of Guassian distribution. Because of that, given enough ranks, a gaussian distribution can be used to approximate the P-table.

2.7.3 The V and P table formulas

Given N which shows us the number of ranks starting from 0 and k which shows us the sum that we are interested in. Recursively apply this algorithm until you reach to the $N = 1$.

$$V_{N+1,k} = V_{N,k-1} + V_{N,k+N+1} \tag{3}$$

There are some additional conditions to the formula:

$$V_{N,k} = 0 \quad \text{if} \quad k < -N\frac{N+1}{2} \quad \text{or} \quad k > N\frac{N+1}{2} \tag{4}$$

$$V_{N,k} = 1 \quad \text{if} \quad k = -N\frac{N+1}{2} \quad \text{or} \quad k = N\frac{N+1}{2} \tag{5}$$

With these formulas, the probability that W is a certain value can be calculated. Given V table, N and k, we can calculate the P by

$$P(T) = \frac{1}{2^n}\left|\sum_{K=T}^{\infty} W_{N,K}\right| \tag{6}$$

To get the P table, we go through all N and K values that we are interested in.

2.7.4 The Gaussian approximation of P table

As mentioned above, if the number of ranks N gets large enough, then Gaussian distribution can be used. To use that, the cumulative deinsity function $F(x)$ is usually used. A simple Gaussian function function $F(x, 1)$, where x is the statistic $z - value$, eg standard score and 1 is the sigma, will give approximately the same results as the accurate P table.

2.7.5 Code sample

A python code to calculate the V table is as follows:

```
def V(n, k, vTable):
    if((n == 1 and (k == 1 or k == -1)) or (n == 0 and k == 0)):
        return 1
    if((k < -(n * (n + 1) / 2)) or (k > (n * (n + 1) / 2))):
        return 0
    n = n - 1

    kIndex =  k + kSize

    leftValue = 0 if kIndex - n - 1 < 0 or \
      kIndex - n - 1 >= len(vTable[n]) else vTable[n][kIndex - n - 1]
    rightValue = 0 if kIndex + n + 1 < 0 or \
      kIndex + n + 1 >= len(vTable[n]) else vTable[n][kIndex + n + 1]

    return leftValue + rightValue

def calculateVValues():
    vTable = []
    for n in range (nSize):
        tableRow = []
        for T0 in range (kSize * 2 + 1):
            tableRow.append(V(n, T0 - kSize, vTable))
        vTable.append(tableRow)
    return vTable
```

The python code to calculate the P table is as follows:

```
def calculatePValues(vTable):
    pTable = []
    for n in range(nSize):
        tableRow = []
        sumValue = 0
        powerValue = 2**n
        for T0 in range (kSize * 2, kSize-1, -1):
            sumValue += vTable[n][T0]
            p = sumValue / powerValue
            tableRow.insert(0, p)
```

```
        pTable.append(tableRow)
    return pTable
```

You can find a python implementation of a code sample in `https://github.com/stenver/wilcoxon-test/blob/master/WilcoxonVTable/pAccurateTable.py`

2.7.6 Bonferroni correction

If you make multiple hypothesis on a test, then you increase the risk in which you reject null hypotheses when its actually true. The Bonferroni test helps to counteract this in a simple and naive way. For each hypothesis you make on a test, you should use a significance level M times lower than before. This ensures that the the increased risk in which we can reject null hypothesis will not raise, no matter the number of hypothesis on a test. So for example, if you make M hypothesis and want want a significance level α, then you should run each test at a significance level of $\frac{\alpha}{M}$.

If you want to use Bonferron correction with Wilcoxon signed-rank test, then you need to keep in mind that the approximated P value significance level needs to be much more accurate, since the bonferron test divides it by the amount of features tested. This means higher the N_0 value, the more features you add in the test.

3 Approximating the p-value

3.1 Goal

The current implementations of Wilcoxon signed-rank test usually assume that Z can be used to calculate the P value, because mostly, the N is sufficiently large to use Z. This allows them to use Gaussian Distributions to approximate the P value and the accurate P table calculation is often left unoptimized.

In the BIIT research group, however, the N is usually not sufficiently large. This means that Gaussian distribution cannot be used. Furthermore, it is not known where exactly the N limit is, so an arbitrary number is used for that purpose. The currently suggested N is 10.

Also, since BIIT uses multiple hypothesis testing, the approximation must be good even for small P-values. As mentioned above, the Bonferroni correction will lower the P significantly.

Because of this, one focus of the paper is finding out how good the approximation of Gaussian Distribution is. The reason to find it out is, as mention, because approximation is computationally cheap and the preffered method to be used over accurate P table, which is computationally expensive.

3.2 Approximation traits

When $N = 1$, then approximation is quite unaccurate. As N increases, then the approximation becomes more and more accurate.

When the $N \to \infty$, then the approximate region will almost equal the accurate P region. This means that when $N \to \infty$, then there is no reason not to use Gaussian Distribution, as it will be completely accurate.

3.3 Optimizing

The bottleneck of the other implementations of the Wilcoxon test, when N is low, is that they calculate the P and V tables every time the test is run, thus if you run thousands of tests in a row, a lot of complex recomputation is done. This eventually becomes a performance issue.

To speed this up - this project will calculate the entire $P_{N,k}$ for every value that is possible where $N < N_0$. Then the $P_{N,k}$ will be hardcoded inside the program for quick lookup of the value.

One of the focuses of this paper is finding the N_0 value

4 Finding the N_0 value

4.1 Proving that P and P_{approx} get more similar as N increases

First we wanted to confirm that as N grows, P and P_{approx} value will get more and more similar. To do this, we took $P_{N,k}$ value for each N where $N < 50$ and $P_{N,k} = 0$ and $P_{N,k-1}! = 0$. We then compared the $P_{N,k}$ and $P_{approx}(N, k)$ for each N where: The results are on Figure 6.

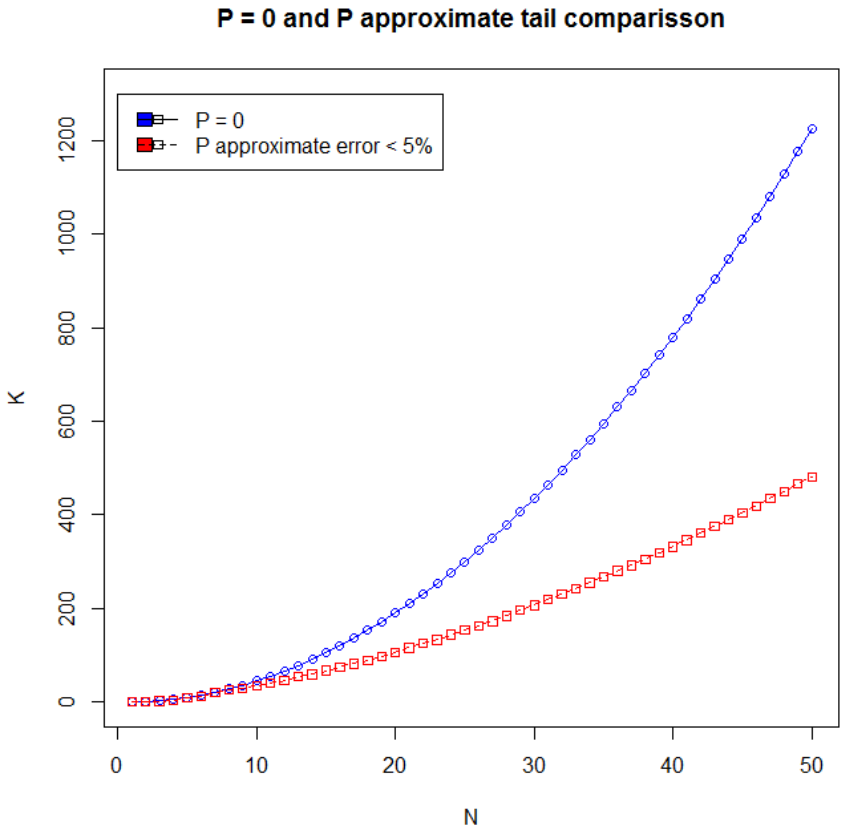

Figure 6: Approximate and accurate comparisson

Much to our surprise, as N grew, the gap between approximate and accurate P values grew bigger. This meant that as N grows, the Gaussian distribution will get more inaccurate toward the tails of the distribution. Previously we thought that with the growth of N, Gaussian would surely get more and more accurate. From Figure 7 we can see that Gaussian distribution is a little bit bigger and gets

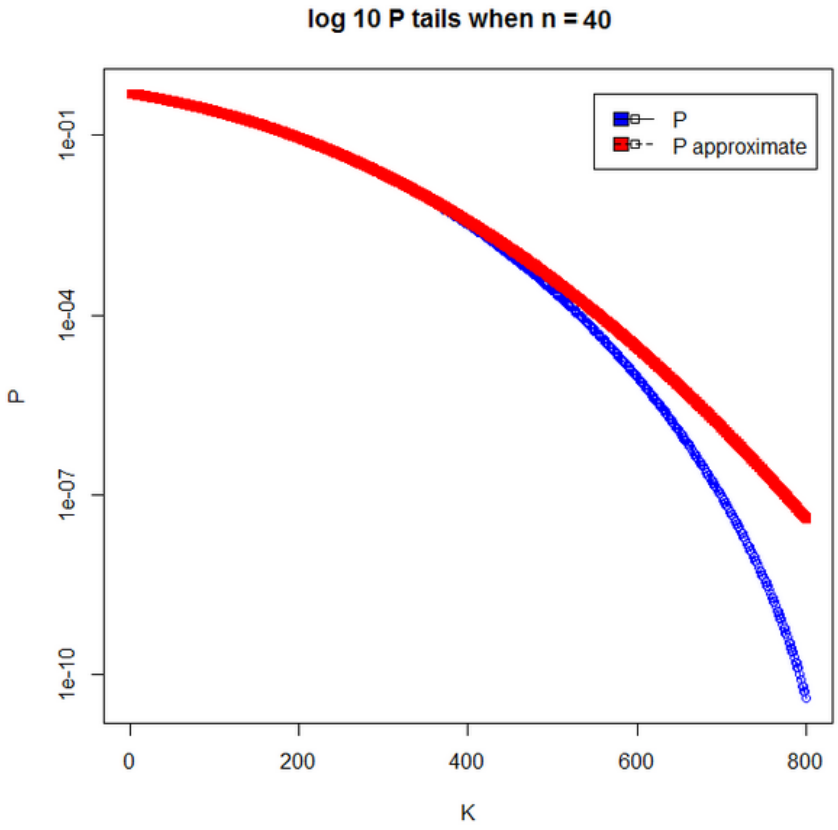

Figure 7: Relative error increasing toward edge of tail

bigger as K grows. Figure 2 illustrates the difference in position of between last accurate P_{appox} and $P = 0$. Accurate means that the relvative error is under 5%

To further investigate this finding, we decided to find out the minimum P value that you can get for each N while maintaining a certain error threshold. The thresholds chosen were $5\%, 10\%, 20\%, 50\%$. The formula used was the same as before, except the thresholds were replaced as needed.

From Figure 8 we can see that as the N grows, the graph becomes stable, meaning that the distribution becomes a Gaussian distribution at around $N > 10$ and $N < 25$. We can see that the minimum P value under error threshold does get smaller, as N increases, so this further proves that Gaussian distribution does get more accurate as N grows. This is because as N increases, you can take a P value closer to the tail of the distribution and still get an accurate value.

The Figure 9 shows us the relative error as P grows when $N = 20$. The P values are logged on this graph. There are two noteworthy things here, however. First, around $P = 0.05$, the relative error actually get a little small tip toward

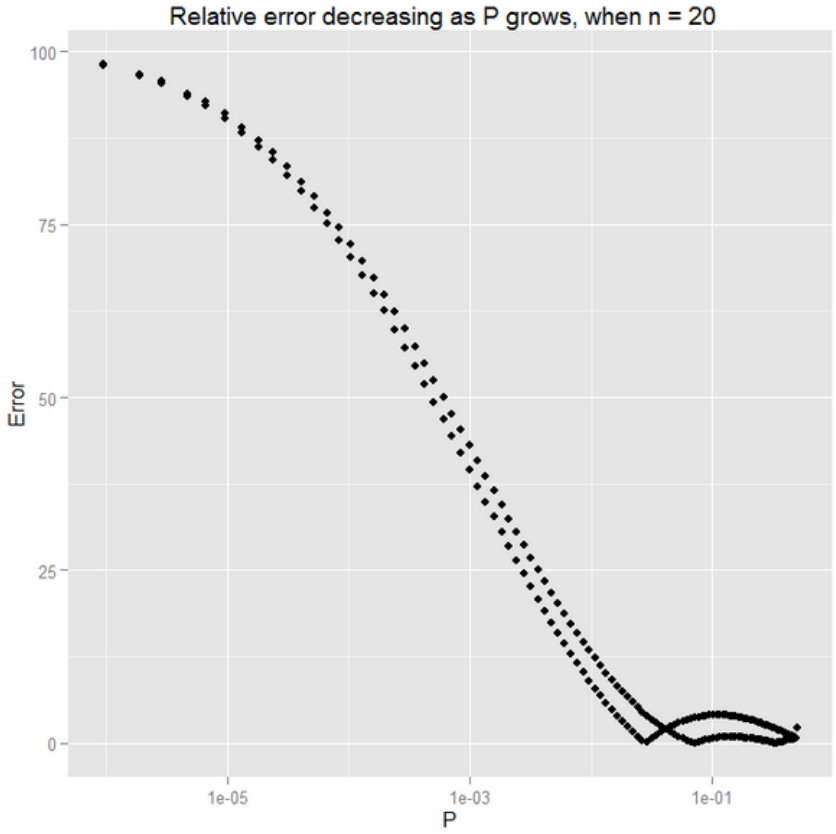

Figure 8: Comparing approximate probability relative error as sample size increses

accuracy. Second, it can be seen that there are two accuracy paths that the error takes, depending on weather in $P_{approxN,K}$ the K is even or odd number. This is a computational artefact caused by the recurrence of the $V_{N,K}$ values and we will not make any conclusions of that.

4.2 Investigating the $P_{N,k}$ and $P_{approxN,K}$ similarities

To further prove that we can start using Gaussian Distribution at a certain N, we needed to prove that Papprox will get more accurate as the N increases toward the tail of the distribution aswell.

To achive this, we found out the largest relative error between P and P_{approx} for each N and for each P where $P = (0.5 : 0.1), P = (0.1, 0.01), P = (0.001, 0.0001), ..., P = (10 - 14, 10 - 15)$

As can be seen from Figure 10, the relative error increases exponentially. When $P = (10-5, 10-6)$ the relative error is massive. When $N = 200$, then the error is

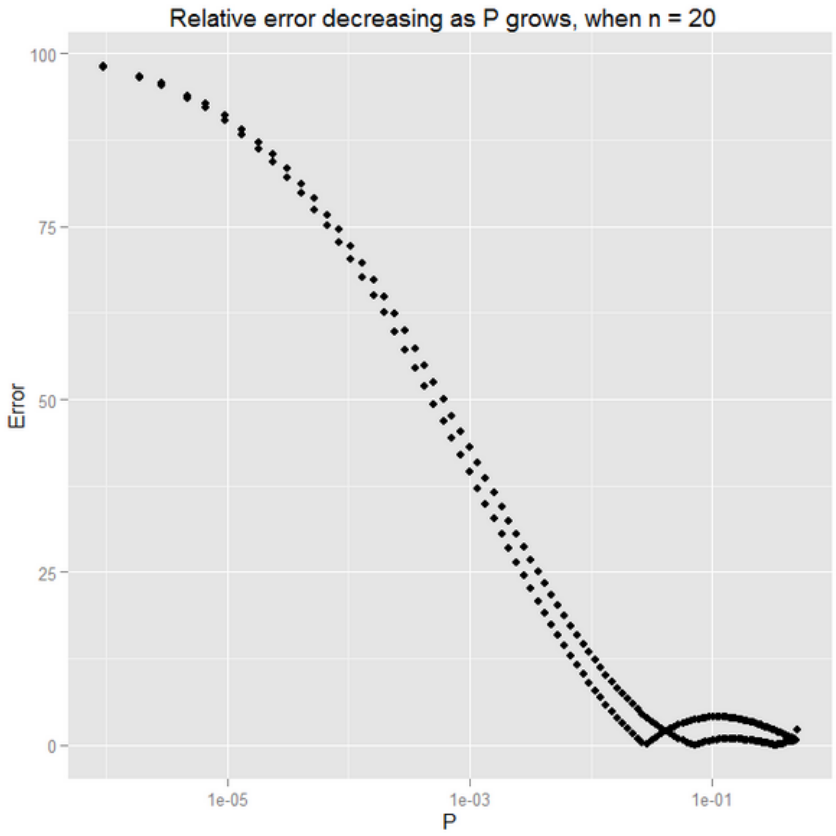

Figure 9: Showing the relative error as probabilty increases when sample size is 20

still around 40%. The bigger the N gets, the bigger the error at the tail. However, it can also be seen that the error decreases steadily as N grows for each of the P range chosen.

This means that even though the error of the distribution tail edge does increase as N grows, overall, both distributions get more and more similar to the Gaussian distribution

To further investigate the relation between $P_{N,k}$ and $P_{approxN,K}$, we wanted to see the difference between all P values when $P = 50$ and when $P = 25$

As can be seen from Figure 11 or Figure 12, when the $N = 50$ or $N = 25$, then the $P_{N,k}$ and $P_{approxN,K}$ values are almost the same with slight differences - the $P_{approximate}$ is a little bit larger. When $N = 25$, the graph is a little less smooth, which tells us that the differences are not as similar as they are when $N = 50$.

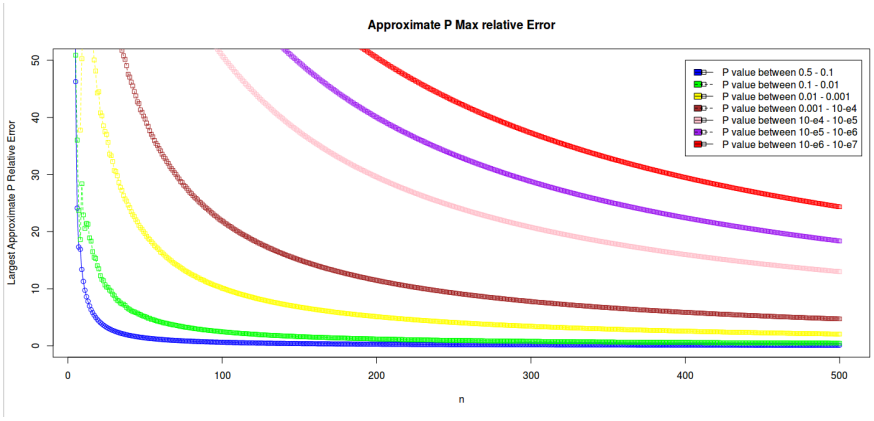

Figure 10: Showing the maximum approximate relative error in probability ranges, as sample size increases

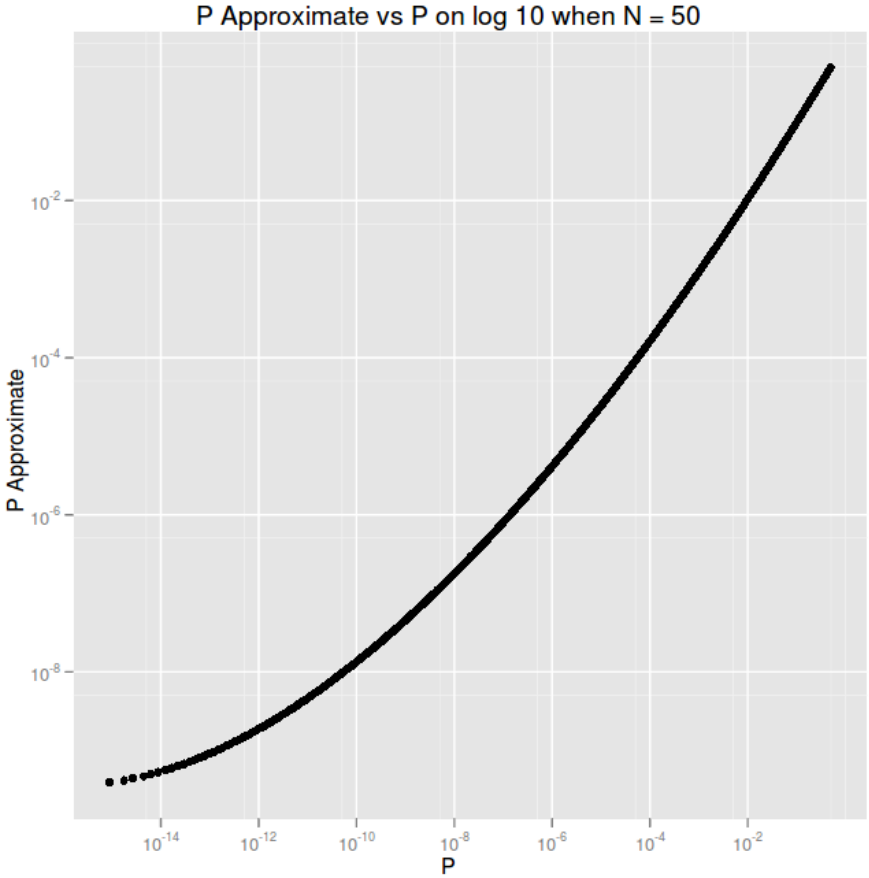

Figure 11: Relative value of all actual probabilities and approximated probabilities when sample size is 50

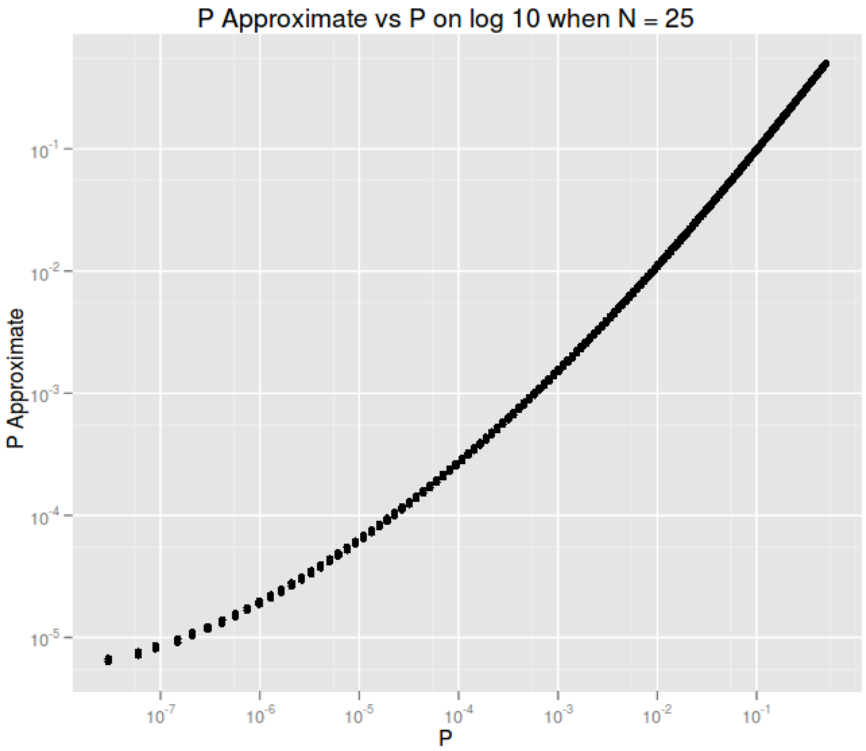

Figure 12: Relative value of all actual probabilities and approximated probabilities when sample size is 25

4.3 Determining the N_0 value

Judging by all the data that has been gathered, it be can clearly see that as N grows, $P_{N,k}$ and $P_{approxN,K}$ tail edges grow apart but overall the distribution gets more and more similar. Determining the N_0 value depends on the amount of data we have and the accuracy of the result we want. The most helpful graphs to help us determine the necessary N_0 value is Figure 10. Table 4 illustrates the number of samples(N) needed for a certain relative accuaracy. So the N_0 value depends on the number of measurements and the required accuracy where $N_0 = N$:

Table 4: Showing the minimal required sample size for a certain relative error and number of measurements

Measurements	Error	Required N
10	5%	25
100		250
1000		500
10000		> 500
100000		> 500
10	10%	80
100		200
1000		> 500
10000		> 500
100000		> 500
10	20%	50
100		100
1000		280
10000		500
100000		> 500
10	50%	25
100		50
1000		100
10000		150
100000		200

As pointed out in the beginning, when the N grows, the $P_{N,k}$ table size grows exponentially.

Since even when $N = 80$, the plain text file was already 2.5mb, the need to optimize further arised.

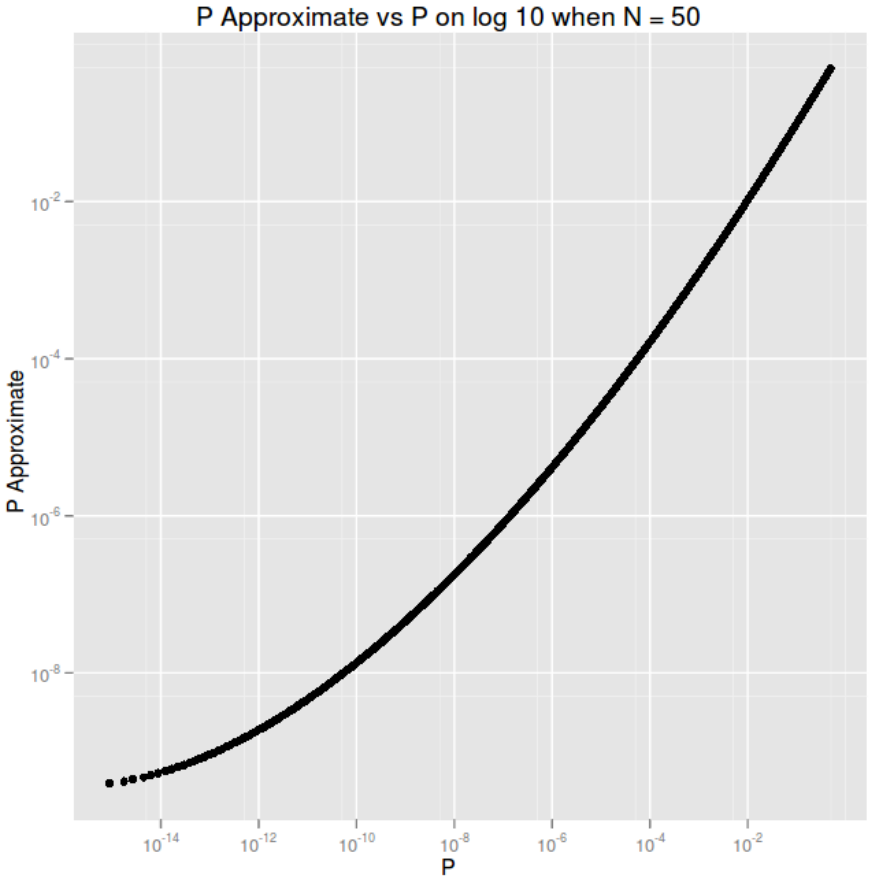

Figure 13: P range with DRC algorithm when N=500

5 Further optimizations

To further optimize the library, it was needed to approximate the accurate P table. Two algorithms were chosen for that matter dose-response curve and linear approximation.

5.0.1 dose-response curve

When using the dose-reponse curve algorithm, the data range as shown in Figure 13 changes sharply toward the edge of the range. Further tests also showed that the algorithm was completly inaccurate at the end of the far edge, so unfortunately this algorithm could not be used.

5.1 Linear approximation

A more straight forward algorithm to use was linear approximation. The tests showed that using relative P values between the P tables gave less approximated data points, then using straight forward accurate P table.

5.1.1 Relative values between P tables

The algorithm to get a relative value for every position between the P tables was pretty straight forward. In Python it looked like this:

```
def getRelativeValuesInRow(n, pTable, gaussianPTable):
    pRelativeValues = []
    for j in range(len(pTable[n])):
      relativeValue = pTable[n][j] / gaussianPTable[n][j]
      pRelativeValues.append(relativeValue)
      if(pTable[n][j] == 0):
          break
    return pRelativeValues

def calculateRelativeTable(pTable, gaussianPTable):
  relativeTable = []
  for n in range(len(pTable)):
    relativeTable = getRelativeValuesInRow(n, pTable, gaussianPTable)
  return relativeTable
```

This returned a new table which had the same number of elements as the accurate P table or Guassian P rable, but for every position there was a relative p-value.

5.1.2 Linear approximation algorithm

The linear approximation algorithm works by starting from data point in the data range and then one by one moving to the next data point, until the points between them cannot be approximated with enough accuracy. Then the last accurate point is saved, picked as the next starting point and the process is repeated until end of range has been reached. This guarantees approximated data range within error margin.

The to approximate the data points in python look like this

```
def linearInterpolate(approximatePoint,
                      endPoint,
                      startPoint,
                      endValue,
                      startValue):
  return startValue +
          ((endValue - startValue) * (approximatePoint - startPoint) /
          (endPoint - startPoint))

def canApproximate(actualPoints, currentPoint, nextPoint):
  startValue = actualPoints[currentPoint]
  endValue = actualPoints[nextPoint]

  for i in range(1, nextPoint - currentPoint):
    approximatePoint = currentPoint + i
    trueValue = actualPoints[approximatePoint]
    approximateValue = linearInterpolate(approximatePoint,
                                         currentPoint,
                                         nextPoint,
                                         startValue,
                                         endValue)
    if approximateValue == 0:
      return True

    relativeAccuaracy = trueValue / approximateValue

    if relativeAccuaracy < 0.9 or relativeAccuaracy > 1.1:
      return False
  return True

def getNextApproximatedPoint(actualPoints, currentPoint):
  nextPoint = currentPoint + 1

  while(canApproximate(actualPoints, currentPoint, nextPoint)):
    if(nextPoint == len(actualPoints) - 1):
      break
    nextPoint += 1
  return nextPoint

def calculateApproximateRow(actualPoints):
```

```
  approximatedPoints = []
  currentPoint = 0
  approximatedPoints.append([currentPoint, actualPoints[currentPoint]])

  while(currentPoint != len(actualPoints) - 1):
    nextPoint = getNextApproximatedPoint(actualPoints, currentPoint)
    approximatedPoints.append([nextPoint, actualPoints[nextPoint]])
    currentPoint = nextPoint
  return approximatedPoints
```

5.1.3 Linear interpolation

To reverse the algorithm, a very simple formula - linear interpolation is used. Given the approximate x, start point x_1, end point x_2, start point y_1, end point y_2, you can easily approximate the y of that point.

$y = y_1 + \frac{(y_2 - y_1) * (x - x_1)}{(x_2 - x_1)}$

In python the code looks like this:

```
def linearInterpolate(approximatePoint,
                      endPoint,
                      startPoint,
                      endValue,
                      startValue):
  return startValue +
          ((endValue - startValue) * (approximatePoint - startPoint) /
          (endPoint - startPoint))
```

5.2 Optimization summary

In the end, thanks to linear approximation, the approximated table, when $N = 500$ could be created. The error margin for the chosen approximated table was chosen to be 20%, since that was deemed as accurate enough. Its still a lot more accurate then Gaussian P table at the edges of the table.

The approximated table is only 1.6mb large. Smaller then when $N = 80$ the accurate P table was.

As for the library itself - it could run over 20000 wilcoxon tests with 120 samples in under 0.7 seconds. Compared to the vanilla R implementation which took many seconds to run 20 tests, this is a significant improvement.

5.3 Further optimizations

A lot of further optimizations could be done on this subject, for example

1. The algorithm to approximate accurate P table could be improved.
2. The plain text file could be compressed.
3. The shared library algorithm implementations could be improved and expanded.

6 The implementaion

Repository that contains everything about our findings can be found in
`https://bitbucket.org/stenver/wilxoni-astaku-test/src/870cc6112d0d?at=default` repository

The repository contains a number of folders.

6.1 RcppWilcoxonTest

An interface that connects our implementation of the optimized Wilcoxon algorithm to the R. Note that you must have installed our implementation to use it. The interface must be compiled with separate R commands from the command line. You need Rcpp packages for R to use it. It can be installed in R console by running:

```
$install.packages("Rcpp")
```

After that, the package can be installed to R by running the command in the command line in the folder:

```
$R CMD INSTALL .
```

You can now load our library in R by calling the

```
>library('RcppWilcoxonTest')
```

and you invoke the function by calling

```
>RcppWilcoxonTest::WilxTest(dataMatrix, testIndexes, controlIndexes)
```

6.2 TerminalWilcoxonTest

An interface that connects our implementation of the optimized Wilcoxon algorithm to the R. Note that you must have installed our implementation to use it. The interface can be compiled by going to the folder, compiling and running help for further help.

```
$make
WilcoxonTest    help
```

Currently only supports NetCDF file as input data.

6.3 WilcoxonTestLibrary

Implementation for the optimized wilcoxon test. The library can be installed by going to the folder and running

```
make install
```

6.4 WilcoxonVTable

Python program that can calculate V, P and approximated P tables, print them, create files of the tables and create a number of graphs on the tables.

6.5 Seminar_paper

The folder that contains this paper and all images attached to it. It also contains R programs to create those images.

7 Conclusion

It was found out that Gaussian distribution tail edge grows larger apart from the $P_{N,k}$ as N increases while the overall distribution grows more similar. In addition, it was confirmed that if thousands of parallel tests are ran, then using approximation is not reliable and instead an accurate table of p-values must be used. Calculating that table is very expensive and thus it is advised to precalculate the table for the program. However, the research shows that if 10 000 measurements are run under 5% relative error, then the hardcoded N should be over 1000. Table of that size would take many gigabytes of space and is impractical to use.

To get around that, approximation of the accurate P table was used. This allowed to significantly reduce the size of the file and thus increase the hardcoded N that was implemented inside the library.

For the time being, the chosen N was 500, with 20% error margin on the approximation. This is still significantly more accurate then Gaussian approximated table at the edges of the distribution. It will allow to make 1000 parallel measurement with 5% error margin.

The library itself could run over 20000 parallel tests with 100 samples in under a second. It is a shared C++ library with currently implemented interfaces in Cron R and Terminal.

Overall the research included some surprising results and can be considered a success.

8 References

1. Wilcoxon, F. (1945) "Individual Comparisons by Ranking Methods.", *Biometrics Bulletin, Vol. 1*, p. 80-83.

2. Siegel, S. (1956) "Nonparametric statistics for the behavioral sciences", p. 75-83.

3. Rew, R., (2011) Davis, G., Emmerson, S. and Daview, H. "The NetCDF Users Guide", p. 5-7, p. 17-23.

4. Abdi, H. (2008) "The Bonferonni and ÅăidÃąkCorrections for Multiple Comparisons" *Encyclopedia of Measurement and Statistics*, p. 1-7

5. Lowry, R. (2011) "Concepts & Applications of Inferential Statistics" *http://vassarstats.net/textbook/ch12a.html* , ch.12a.

6. Triola, M (2001). "Elementary statistics (8 ed.)", p. 388.

7. Ritz, C. and Strebig, J. (2013) "Dose-reponse curve" (`http://cran.r-project.org/web/packages/drc/index.html`), p. 111.

8. DebRoy, S. and Trapletti, A. (2014) "Writing R extensions" (`http://cran.r-project.org/doc/manuals/R-exts.html`), ch 5-6.

9. Eddelbuettel, D. and Francois, R. (2014) "Seamless R and C++ integration" (`http://cran.r-project.org/web/packages/Rcpp/vignettes/Rcpp-introduction.pdf`), 3-15.

10. Wikipedia Foundation Inc, z-score (`http://en.wikipedia.org/wiki/Standard_score`), 16 January 2014

11. Wikipedia Foundation Inc, linear interpolation (`http://en.wikipedia.org/wiki/Linear_interpolation`), 10 January 2014

12. Wikipedia Foundation Inc, Gaussian function (`http://en.wikipedia.org/wiki/Gaussian_function`), 8 January 2014

13. Wikipedia Foundation Inc, Test Statistic (`http://en.wikipedia.org/wiki/Test_statistic`), 10 January 2014