Experimental and Exploratory Analysis of Programming Languages Popularity According to the PYPL Index

Oleksandr Mediakov [1], Bohdan Korostynskyi [1], Victoria Vysotska [1,2], Oksana Markiv [1] and Sofia Chyrun [1]

[1] *Lviv Polytechnic National University, S. Bandera Street, 12, Lviv, 79013, Ukraine*
[2] *Osnabrück University, Friedrich-Janssen-Str. 1, Osnabrück, 49076, Germany*

Abstract
This article dwells upon procedure for primary and exploratory analysis of data concerning programming languages popularity according to the PYPL index. Classical, but flexible methods of cluster and correlation analysis, and mathematical statistics have been used in data analysis. Examples of various researches in the field of popularity and development of programming languages have been provided, and based on open questions in this area, set of graphic results has been constructed by data analysis, which logically has resulted in conclusions regarding the selection of models and methods for further data forecasting. Integral part of the report information content consists of executable code examples in the R programming language, which can be used to conduct similar studies.

Keywords [1]
Cluster analysis, correlation analysis, programming languages, PYPL index, data forecasting, data analysis, mathematical statistics

1. Introduction

Achievements and development of modern technologies are the result of long-term people intellectual activity with aim of solving any problems that cause discomfort or impose restrictions on the desires, interests or the process of society and/or an individual flourishing. Computer systems capable of performing a huge number of arithmetic, logical and derivative operations in very short periods are vivid examples of this development achievements.

Logically, in order to use the capabilities of these systems, it is necessary to have a tool for managing them, which is programming language. The spread, diversity, and popularity of information technologies have led to the existence of a wide class of people who use programming languages. Because of this, the question of specific programming languages popularity in different periods is widely discussed in the media and among the community of IT representatives.

However, the question of popularity can be considered from different angles, the most logical of which is the statistical one. Primary statistical analysis of data on the popularity of various "living" languages will allow to see patterns and trends among developers in general. Also, mathematical processing of data will allow to build relative indicators of growth or decline in the popularity of languages.

To achieve correct results, it is necessary to define clearly the tasks that will be considered in the work. Using the usual mathematical approach, it can be argued that it is necessary to conduct data analysis with: construction of statistics of various programming languages and in general for all,

MoMLeT+DS 2022: 4th International Workshop on Modern Machine Learning Technologies and Data Science, November, 25-26, 2022, Leiden-Lviv, The Netherlands-Ukraine
EMAIL: Oleksandr.Mediakov.sa.2019@lpnu.ua (O. Mediakov); bohdan.korostynskyi.sa.2019@lpnu.ua (B. Korostynskyi); victoria.a.vysotska@lpnu.ua (V. Vysotska); oksana.o.markiv@lpnu.ua (O. Markiv)
ORCID: 0000-0002-2580-3155 (O. Mediakov); 0000-0002-0545-6264 (B. Korostynskyi); 0000-0001-6417-3689 (V. Vysotska); 0000-0002-1691-1357 (O. Markiv); 0000-0002-2829-0164 (S. Chyrun)

research of regularities in the distribution of time data, detection of the presence or absence of linear relationships between the popularity of various programming languages, and also to distinguish groups programming languages depending on the dynamics of changes in their popularity [1-3]. For a high-quality presentation of the results or the correct choice of the method, it is necessary to visualize the intermediate and final results.

2. Related works

Since the topic of programming and programming languages is broad, popular and interesting for society, analysis of the chosen topic is carried out in various forms - scientific articles and studies, journalistic articles, own investigations of corporations, communities, etc. Question of language popularity is raised to trace trends in language paradigms, settings, even syntax.

First of all, it is necessary to mention the availability of ratings that publicize current state of the "market" of programming languages. For example, one of the most famous indices as ratings, TIOBE programming community index [3]. This rating is made exclusively for programming languages (Turing complete), based on the number of mentions in various search engines. It is obvious that this approach has significant drawbacks, because it takes into account unpopular articles and literature. Thus, this rating counts only the "noise" of the programming language in the network [1-5].

In contrast to this rating, more qualitative one has emerged - PYPL Popularity of Programming Language. This rating is formed by analyzing the number of searches for programming language courses in Google [2]. Data from this rating has more respect and authority among users. It is from this rating that the data that will be analyzed in the following parts of the work was formed.

It is logical that such platforms as GitHub or StackOverflow can keep certain statistics on programming languages.

Based on data from GitHub, the company creates an annual Github report, where it displays data on the number of repositories, pull requests, etc., by programming language. With the help of this information, it is possible to see a real picture of live projects and the technologies they use.

In turn, several important ratings are formed from the data of StackOverflow, among them: popularity of programming languages, most favorite programming languages by developers, most willing languages. Based on these ratings, media representatives, specialists and users of programming languages create articles and blogs. For example, a rather popular magazine in IT – Medium published a general collection of analysis of ratings of programming languages [1]. This work reveals a lot of interesting information about the use and development of trends among developers and IT specialists, their professional preparation, formation of requirements for job vacancies or participation in project teams, recruiting and formation of requirements for IT projects [6-27]. However, many popular articles and works describe general aspects, so it is possible to highlight the place of this work among others – the work will offer a qualitative statistical analysis of the temporal evolution the popularity of programming languages for data forecasting based on machine learning [28-72], which will allow assessing the growth and decline of each language separately in a relative and absolute projection.

3. Methods

For the correct solution of the given problems, it is necessary correctly to select or create algorithms of methods. For primary statistical analysis, it is possible to build several different approaches to each problem. It is necessary to consider the algorithms of methods that will be used in the work in the future.

The simplest methods used in the work are the calculation of time series statistics. For their calculation, the usual formulas and methods of mathematical statistics are used, which do not require special improvements for specific data.

However, there is a cluster of algorithms and methods that need to be modernized or refined for specific data, or the methods have several options, and it is necessary to choose one of them.

Several such methods belong to correlation analysis. For example, there is a need to present the algorithm for forming a correlation matrix with its visualization in the form of a heat map.

S- algorithm for filling the correlation matrix and its visualization

S1. Initialize unit matrix of the order of the number of attributes

S2. Cycle through the cells of the matrix with incremental parameters i, j.

S2.1. If $i \neq j$ belongs to the matrix - in cell i, j assigns the value of the Pearson correlation coefficient [4] between attributes i and j.

S2.2. If $i=j$ skip step.

S3. Find the maximum and minimum of the matrix.

S4. Create linear color change function from minimum to maximum.

S5. Display table with colors according to the matrix cell values.

However, work with matrices is not only found in correlation analysis, the use of matrix theory is also necessary in cluster analysis. One of the main tasks when performing clustering is to determine the next union node, which mathematically means finding the minimum among the elements of the lower (or upper) part of the matrix without taking into account the main diagonal. The algorithm that performs this task is shown in Fig. 1.

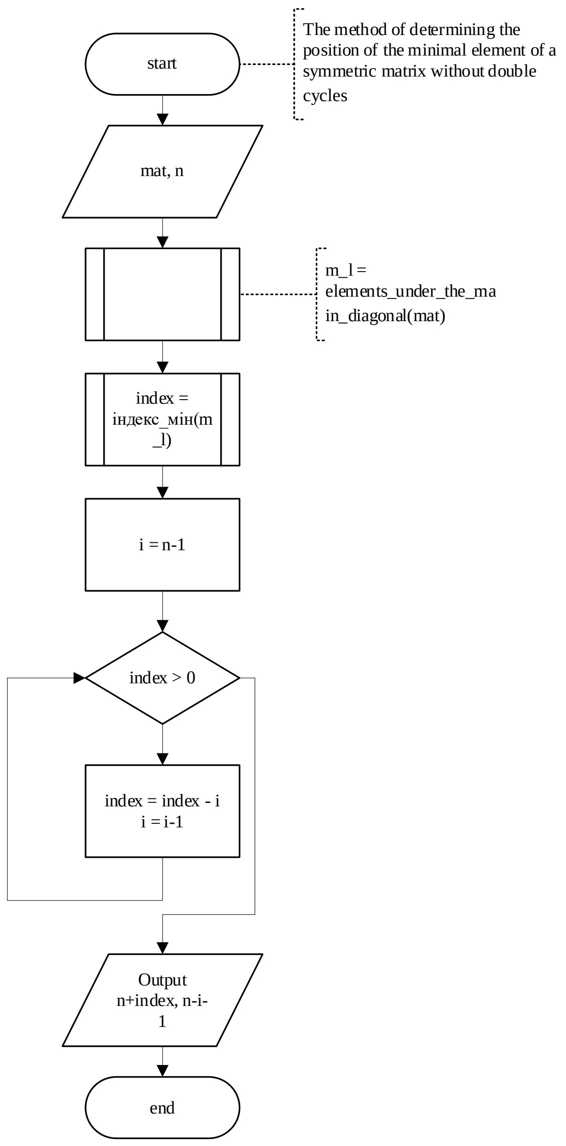

start
The method of determining the position of the minimal element of a symmetric matrix without double cycles
mat, n
m_l = elements_under_the_main_diagonal(mat)
index = індекс_мін(m_l)
i = n-1
index > 0
index = index - i
i = i-1
Output n+index, n-i-1
end

Figure 1: Block diagram of algorithm for finding the position of minimum element in symmetric matrix

In this method, the use of double loops with conditions is enabled, and the overall estimate of the asymptotics of the algorithm is $O(n)$, where n is the number of diagonal elements. However, for the general case, let m be the order of the matrix, then the complexity of the algorithm is estimated at $O\left(\left\lfloor\frac{n^2-n}{2}\right\rfloor\right)$. More general picture of the clustering method with the formation of a hierarchical structure involves the creation of a dendogram, but there is a problem of remembering the results of the clustering steps in the form of a certain data structure. For this purpose, we present a diagram of activities of the clustering algorithm with memorization of intermediate results (Fig. 2).

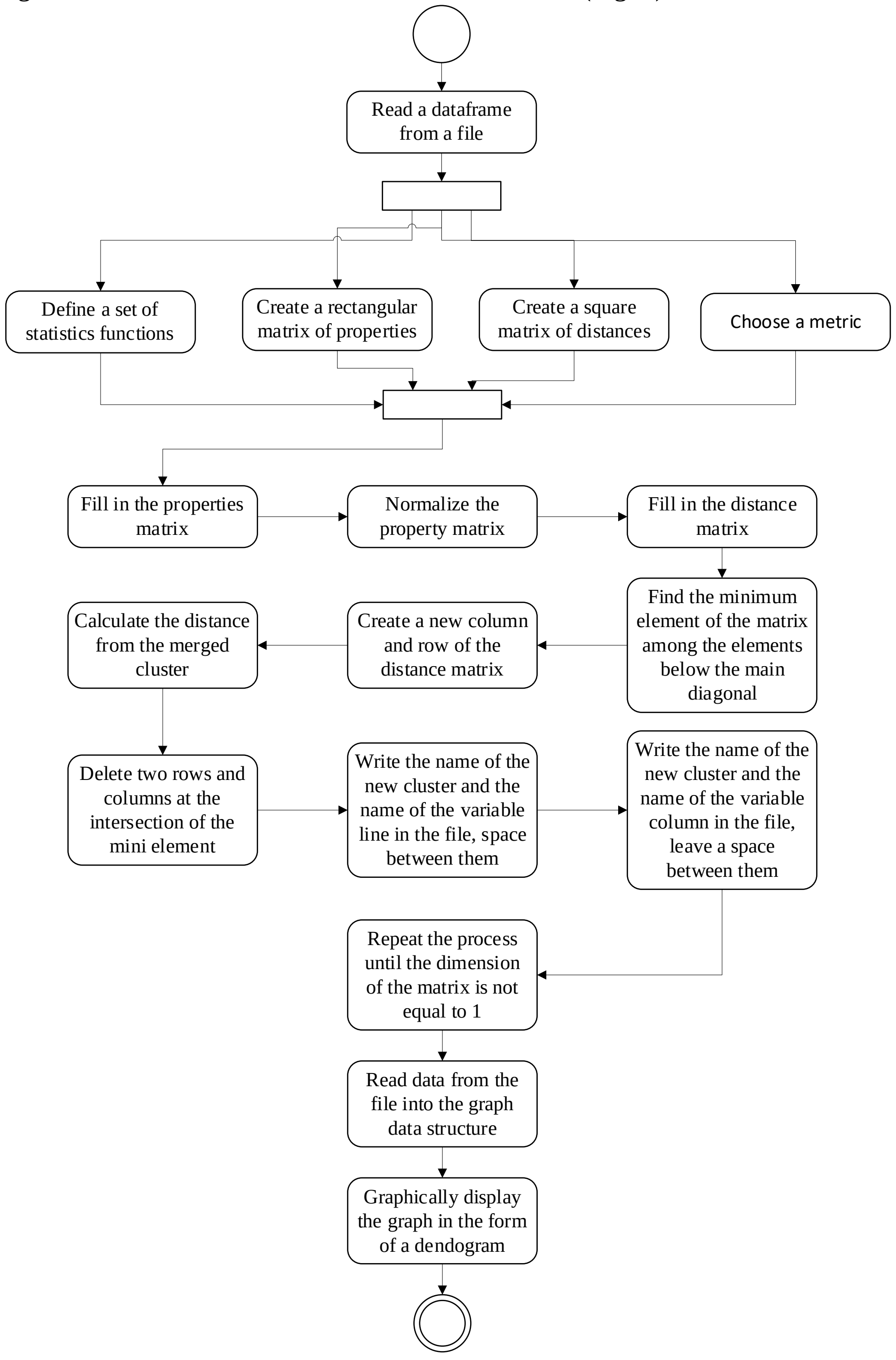
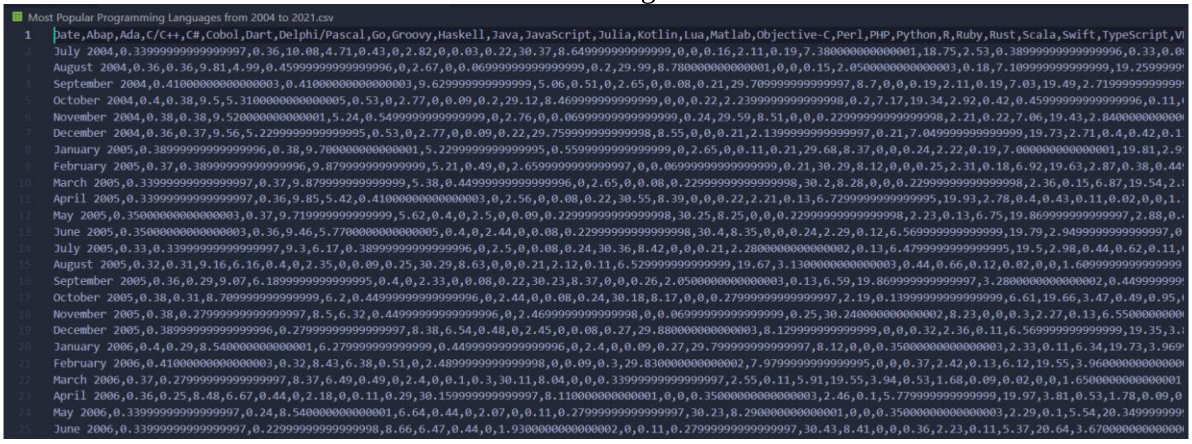

Read a dataframe from a file
Define a set of statistics functions
Create a rectangular matrix of properties
Create a square matrix of distances
Choose a metric
Fill in the properties matrix
Normalize the property matrix
Fill in the distance matrix
Find the minimum element of the matrix among the elements below the main diagonal
Create a new column and row of the distance matrix
Calculate the distance from the merged cluster
Delete two rows and columns at the intersection of the mini element
Write the name of the new cluster and the name of the variable line in the file, space between them
Write the name of the new cluster and the name of the variable column in the file, leave a space between them
Repeat the process until the dimension of the matrix is not equal to 1
Read data from the file into the graph data structure
Graphically display the graph in the form of a dendogram

Figure 2: UML activity diagram for the hierarchical clustering method

4. Experiments

Experimental data for researching the popularity of programming languages is the time sequence of the PYPL index value for 28 programming languages. PYPL [1-5] index was mentioned in previous parts of the work, but the data itself was collected by Kaggle users, the corresponding access mode is https://www.kaggle.com/muhammadkhalid/most-popular-programming-languages-since-2004.

The initial view of data in csv format is shown in Fig. 3.

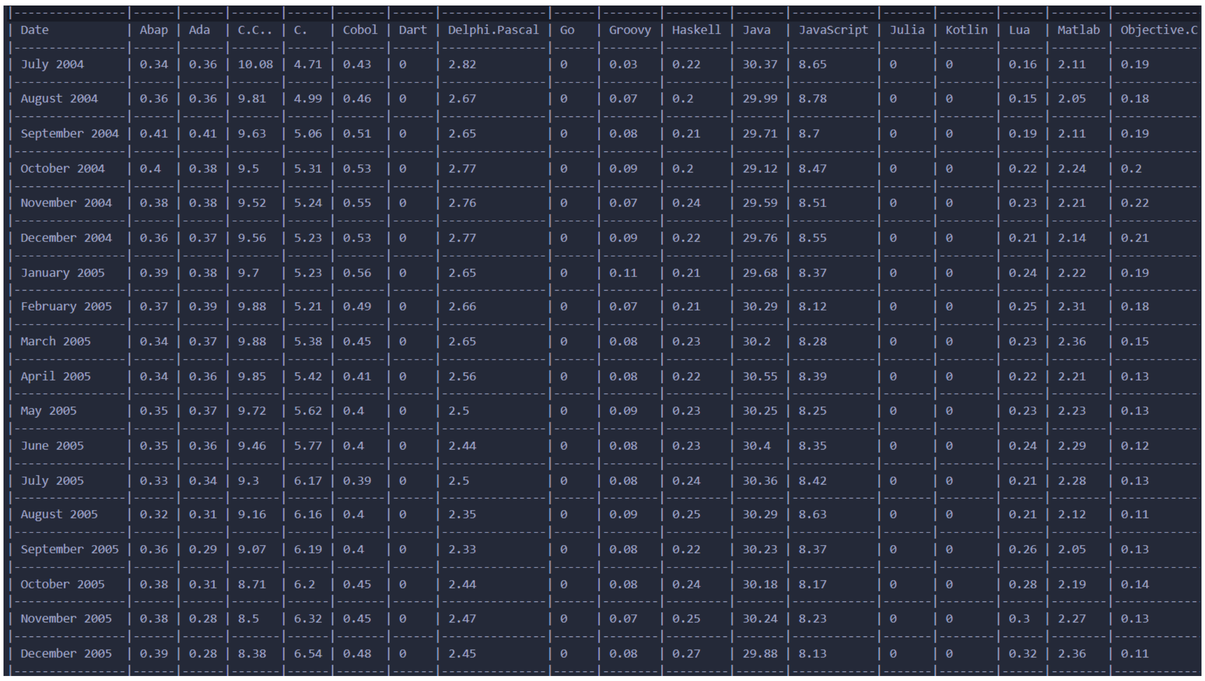


Most Popular Programming Languages from 2004 to 2021.csv

```
Date,Abap,Ada,C/C++,C#,Cobol,Dart,Delphi/Pascal,Go,Groovy,Haskell,Java,JavaScript,Julia,Kotlin,Lua,Matlab,Objective-C,Perl,PHP,Python,R,Ruby,Rust,Scala,Swift,TypeScript,V
July 2004,0.33999999999999997,0.36,10.08,4.71,0.43,0,2.82,0,0.03,0.22,30.37,8.649999999999999,0,0,0.16,2.11,0.19,7.380000000000001,18.75,2.53,0.38999999999999996,0.33,0.0
August 2004,0.36,0.36,9.81,4.99,0.45999999999999996,0,2.67,0,0.06999999999999999,0.2,29.99,8.780000000000001,0,0,0.15,2.0500000000000003,0.18,7.109999999999999,19.2599999
September 2004,0.41000000000000003,0.41000000000000003,9.629999999999999,5.06,0.51,0,2.65,0,0.08,0.21,29.709999999999997,8.7,0,0,0.19,2.11,0.19,7.03,19.49,2.71999999999999
October 2004,0.4,0.38,9.5,5.3100000000000005,0.53,0,2.77,0,0.09,0.2,29.12,8.469999999999999,0,0,0.22,2.2399999999999998,0.2,7.17,19.34,2.92,0.42,0.45999999999999996,0.11,
November 2004,0.38,0.38,9.520000000000001,5.24,0.5499999999999999,0,2.76,0,0.06999999999999999,0.24,29.59,8.51,0,0,0.22999999999999998,2.21,0.22,7.06,19.43,2.84000000000000
December 2004,0.36,0.37,9.56,5.2299999999999995,0.53,0,2.77,0,0.09,0.22,29.759999999999998,8.55,0,0,0.21,2.1399999999999997,0.21,7.049999999999999,19.73,2.71,0.4,0.42,0.1
January 2005,0.38999999999999996,0.38,9.700000000000001,5.2299999999999995,0.5599999999999999,0,2.65,0,0.11,0.21,29.68,8.37,0,0,0.24,2.22,0.19,7.000000000000001,19.81,2.9
February 2005,0.37,0.38999999999999996,9.879999999999999,5.21,0.49,0,2.6599999999999997,0,0.06999999999999999,0.21,30.29,8.12,0,0,0.25,2.31,0.18,6.92,19.63,2.87,0.38,0.44
March 2005,0.33999999999999997,0.37,9.879999999999999,5.38,0.44999999999999996,0,2.65,0,0.08,0.22999999999999998,30.2,8.28,0,0,0.22999999999999998,2.36,0.15,6.87,19.54,2.
April 2005,0.33999999999999997,0.36,9.85,5.42,0.41000000000000003,0,2.56,0,0.08,0.22,30.55,8.39,0,0,0.22,2.21,0.13,6.7299999999999995,19.93,2.78,0.4,0.43,0.11,0.02,0,0,1.
May 2005,0.35000000000000003,0.37,9.719999999999999,5.62,0.4,0,2.5,0,0.09,0.22999999999999998,30.25,8.25,0,0,0.22999999999999998,2.23,0.13,6.75,19.869999999999997,2.88,0.
June 2005,0.35000000000000003,0.36,9.46,5.7700000000000005,0.4,0,2.44,0,0.08,0.22999999999999998,30.4,8.35,0,0,0.24,2.29,0.12,6.569999999999999,19.79,2.9499999999999997,0
July 2005,0.33,0.33999999999999997,9.3,6.17,0.38999999999999996,0,2.5,0,0.08,0.24,30.36,8.42,0,0,0.21,2.2800000000000002,0.13,6.4799999999999995,19.5,2.98,0.44,0.62,0.11,
August 2005,0.32,0.31,9.16,6.16,0.4,0,2.35,0,0.09,0.25,30.29,8.63,0,0,0.21,2.12,0.11,6.529999999999999,19.67,3.1300000000000003,0.44,0.66,0.12,0.02,0,0,1.6099999999999999
September 2005,0.36,0.29,9.07,6.1899999999999995,0.4,0,2.33,0,0.08,0.22,30.23,8.37,0,0,0.26,2.0500000000000003,0.13,6.59,19.869999999999997,3.2800000000000002,0.449999999
October 2005,0.38,0.31,8.709999999999999,6.2,0.44999999999999996,0,2.44,0,0.08,0.24,30.18,8.17,0,0,0.27999999999999997,2.19,0.13999999999999999,6.61,19.66,3.47,0.49,0.95,
November 2005,0.38,0.27999999999999997,8.5,6.32,0.44999999999999996,0,2.4699999999999998,0,0.06999999999999999,0.25,30.240000000000002,8.23,0,0,0.3,2.27,0.13,6.550000000000
December 2005,0.38999999999999996,0.27999999999999997,8.38,6.54,0.48,0,2.45,0,0.08,0.27,29.880000000000003,8.129999999999999,0,0,0.32,2.36,0.11,6.569999999999999,19.35,3.
January 2006,0.4,0.29,8.540000000000001,6.279999999999999,0.44999999999999996,0,2.4,0,0.09,0.27,29.799999999999997,8.12,0,0,0.35000000000000003,2.33,0.11,6.34,19.73,3.969
February 2006,0.41000000000000003,0.32,8.43,6.38,0.51,0,2.4899999999999998,0,0.09,0.3,29.830000000000002,7.9799999999999995,0,0,0.37,2.42,0.13,6.12,19.55,3.9600000000000
March 2006,0.37,0.27999999999999997,8.37,6.49,0.49,0,2.4,0,0.1,0.3,30.11,8.04,0,0,0.33999999999999997,2.55,0.11,5.91,19.55,3.94,0.53,1.68,0.09,0.02,0,0,1.6500000000000001
April 2006,0.36,0.25,8.48,6.67,0.44,0,2.18,0,0.11,0.29,30.159999999999997,8.110000000000001,0,0,0.35000000000000003,2.46,0.1,5.779999999999999,19.97,3.81,0.53,1.78,0.09,0
May 2006,0.33999999999999997,0.24,8.540000000000001,6.64,0.44,0,2.07,0,0.11,0.27999999999999997,30.23,8.290000000000001,0,0,0.35000000000000003,2.29,0.1,5.54,20.349999999
June 2006,0.33999999999999997,0.22999999999999998,8.66,6.47,0.44,0,1.9300000000000002,0,0.11,0.27999999999999997,30.43,8.41,0,0,0.36,2.23,0.11,5.37,20.64,3.670000000000
```

Figure 3: Part of the file with output data

Unfortunately, the data readability is minimal in this form, but it can be improved by changing the view to the appropriate Fig. 4. It can be seen that the variables, that is, the columns, are each programming language and the date column, respectively, each line is a record of the date and the value of the PYPL index for that date. Accordingly, the size of the data is 207 by 29. In general, the dataset includes data from July 2004 to September 2021. And the programming languages considered in the data are: Abap, Ada, C/C++, C#, Cobol, Dart, Delphi.Pascal, Go , Groovy , Haskell, Java, JavaScript, Julia, Kotlin, Lua, Matlab, Objective.C, Perl, PHP, Python, R, Ruby, Rust, Scala, Swift, TypeScript, VBA, Visual.Basic. Such a data structure also corresponds to certain properties, for example, the sum of all values in one row equals 100, this is because the PYPL index itself displays data in 100% language breakdown.

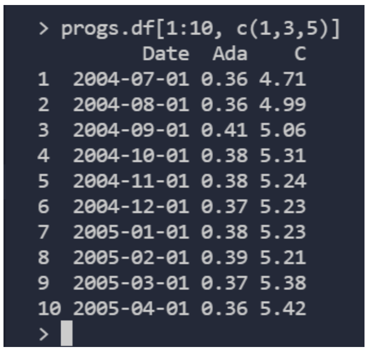


Date	Abap	Ada	C.C..	C.	Cobol	Dart	Delphi.Pascal	Go	Groovy	Haskell	Java	JavaScript	Julia	Kotlin	Lua	Matlab	Objective.C
July 2004	0.34	0.36	10.08	4.71	0.43	0	2.82	0	0.03	0.22	30.37	8.65	0	0	0.16	2.11	0.19
August 2004	0.36	0.36	9.81	4.99	0.46	0	2.67	0	0.07	0.2	29.99	8.78	0	0	0.15	2.05	0.18
September 2004	0.41	0.41	9.63	5.06	0.51	0	2.65	0	0.08	0.21	29.71	8.7	0	0	0.19	2.11	0.19
October 2004	0.4	0.38	9.5	5.31	0.53	0	2.77	0	0.09	0.2	29.12	8.47	0	0	0.22	2.24	0.2
November 2004	0.38	0.38	9.52	5.24	0.55	0	2.76	0	0.07	0.24	29.59	8.51	0	0	0.23	2.21	0.22
December 2004	0.36	0.37	9.56	5.23	0.53	0	2.77	0	0.09	0.22	29.76	8.55	0	0	0.21	2.14	0.21
January 2005	0.39	0.38	9.7	5.23	0.56	0	2.65	0	0.11	0.21	29.68	8.37	0	0	0.24	2.22	0.19
February 2005	0.37	0.39	9.88	5.21	0.49	0	2.66	0	0.07	0.21	30.29	8.12	0	0	0.25	2.31	0.18
March 2005	0.34	0.37	9.88	5.38	0.45	0	2.65	0	0.08	0.23	30.2	8.28	0	0	0.23	2.36	0.15
April 2005	0.34	0.36	9.85	5.42	0.41	0	2.56	0	0.08	0.22	30.55	8.39	0	0	0.22	2.21	0.13
May 2005	0.35	0.37	9.72	5.62	0.4	0	2.5	0	0.09	0.23	30.25	8.25	0	0	0.23	2.23	0.13
June 2005	0.35	0.36	9.46	5.77	0.4	0	2.44	0	0.08	0.23	30.4	8.35	0	0	0.24	2.29	0.12
July 2005	0.33	0.34	9.3	6.17	0.39	0	2.5	0	0.08	0.24	30.36	8.42	0	0	0.21	2.28	0.13
August 2005	0.32	0.31	9.16	6.16	0.4	0	2.35	0	0.09	0.25	30.29	8.63	0	0	0.21	2.12	0.11
September 2005	0.36	0.29	9.07	6.19	0.4	0	2.33	0	0.08	0.22	30.23	8.37	0	0	0.26	2.05	0.13
October 2005	0.38	0.31	8.71	6.2	0.45	0	2.44	0	0.08	0.24	30.18	8.17	0	0	0.28	2.19	0.14
November 2005	0.38	0.28	8.5	6.32	0.45	0	2.47	0	0.07	0.25	30.24	8.23	0	0	0.3	2.27	0.13
December 2005	0.39	0.28	8.38	6.54	0.48	0	2.45	0	0.08	0.27	29.88	8.13	0	0	0.32	2.36	0.11

Figure 4: Part of data represented in the form of table

R language was used for data experiments and graphing, as well as such packages as dplyr, ggplot2 (in fact, most of the libraries from tidyverse), reshape2, zoo, etc. (Fig. 6-7). To begin with, the data must be read correctly, and the problem arises that the format of the date column does not correspond to the one available in R, so it must be converted to a new one, for example, October 2020 will be converted into 10 2020, which in turn will be converted into a Date data type with a value of 01.10. 2020. The general function of reading data from a file and primary transformation:

```
library(dplyr)
library(ggplot2)
library(reshape2)
get_data_frame <- function() {
    # correspondence of months in English by numbers
    months_u_e <- c("January" = "1", "February" = "2", "March" = "3", "April" = "4", "May"
= "5", "June" = "6",
    "July" = "7", "August" = "8", "September" = "9", "October" = "10", "November" = "11",
"December" = "12")
    # opening a file in a dataframe
    progs.df <- read.csv("Most Popular Programming Languages from 2004 to 2021.csv")
    progs.df$Date %>% stringr::str_replace_all(pattern = months_u_e) -> progs.df$Date
    rm(months_u_e)
    # converting data string to a Date type
    progs.df$Date <- zoo::as.Date(zoo::as.yearmon(progs.df$Date, "%m %Y"))
    names(progs.df)[4] <- "C++"
    names(progs.df)[5] <- "C"
    progs.df %>% return
}
progs.df <- get_data_frame()
```

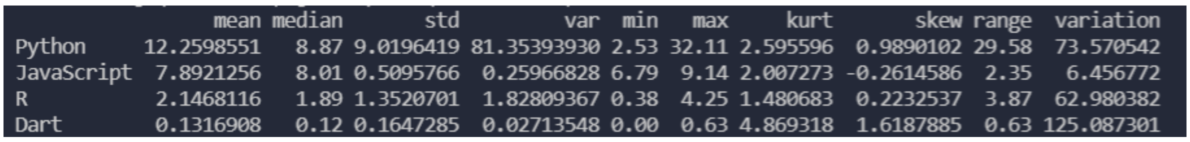


```
> progs.df[1:10, c(1,3,5)]
         Date  Ada    C
1  2004-07-01 0.36 4.71
2  2004-08-01 0.36 4.99
3  2004-09-01 0.41 5.06
4  2004-10-01 0.38 5.31
5  2004-11-01 0.38 5.24
6  2004-12-01 0.37 5.23
7  2005-01-01 0.38 5.23
8  2005-02-01 0.39 5.21
9  2005-03-01 0.37 5.38
10 2005-04-01 0.36 5.42
>
```

Figure 5: Output of a fragment of the data window after it is loaded

The first task set before the work involves the construction of statistics for each of the programming languages. It is necessary to go through the data columns and create a new dataframe with statistical parameters.

```
library(moments)
source("libs_config.r")
# returns a set of statistical characteristics for the vector x
statistical_characteristics <- function(x) {
    c("mean" = mean(x), "median" = median(x),
    "std" = sd(x), "var" = var(x), "min" = min(x),
    "max" = max(x), "kurt" = kurtosis(x), "skew" = skewness(x)) %>% return
}
# use of function for all columns except date
apply(progs.df[, 2:29], MARGIN = 2, FUN = statistical_characteristics) %>%
t() %>% data.frame() %>% mutate(range = (max - min), variation = std/mean*100) %>%
arrange(desc(mean)) -> stats.df
```



```
                 mean median       std         var  min   max     kurt       skew range  variation
Python     12.2598551   8.87 9.0196419 81.35393930 2.53 32.11 2.595596  0.9890102 29.58  73.570542
JavaScript  7.8921256   8.01 0.5095766  0.25966828 6.79  9.14 2.007273 -0.2614586  2.35   6.456772
R           2.1468116   1.89 1.3520701  1.82809367 0.38  4.25 1.480683  0.2232537  3.87  62.980382
Dart        0.1316908   0.12 0.1647285  0.02713548 0.00  0.63 4.869318  1.6187885  0.63 125.087301
```

Figure 6: Interactive console text with basic language statistics

Although the R language has many built-in functions, it is possible to build your own more flexible functions, for example a simple histogram construction would look like this:

```
ggplot(progs.df$R, aes(x = R)) + geom_hist()
```

However, in order to achieve flexibility, it is necessary to correctly determine the number of columns (orthe step of the column), as well as experiment with the visual display, therefore, the following functions were created to build the histogram and the probability distribution function:

```r
source("task_1_stat\\create_statistics.r")
# function of dividing a vector into intervals
create_stat <- function(data.vec) {
    r <- max(data.vec) - min(data.vec)
    m <- ceiling(1+log2(length(data.vec)))
    bin <- r/m
    breaks <- seq(min(data.vec), max(data.vec), bin)
    data.vec %>% sort %>% cut(by = bin, breaks = breaks) -> data.vec
    list(data.vec, breaks) %>% return
}
# function of creating a histogram as a ggplot object
create_histogram <- function(data.stats) {
    (ggplot() +
    geom_col(aes(x = head(data.stats[[2]], -1L),
        y = as.integer(table(data.stats[[1]])), fill = levels(data.stats[[1]]))) +
    labs(x = "R popularity", y = "count", fill = "Intervals") +
    scale_fill_brewer(palette = 3, type = "div"))%>% return
}
# function of creating distribution function in the form of ggplot object
create_distr_plot <- function(data.stats) {
    breaks <- data.stats[[2]]
    data.stats <- data.stats[[1]]
    counts <- as.integer(table(data.stats))
    pdf <- rep(0,(length(counts)+1))
    for( i in seq(2,length(levels(data.stats)))) {
        pdf[i] <- sum(counts[1:i])/length(data.stats)
    }
    pdf[length(counts)+1] <- 1
    (ggplot(data.frame(x = breaks, y = pdf), aes(x = x, y = y)) +
    geom_line(size = 1.5, alpha = 0.7, col = "#72007c") +
    geom_point(size = 2.5, alpha = 0.8, col = "#470068") +
    theme_light() + labs(x = "R popularity", y = "Probability")) %>% return
}
progs.df$R %>% create_stat -> stat
stat %>% create_histogram -> hist.plot
stat %>% create_distr_plot -> pdf.plot
```

With this approach, it is possible to adjust easy the number and colors of the columns, for example as in Fig. 7.

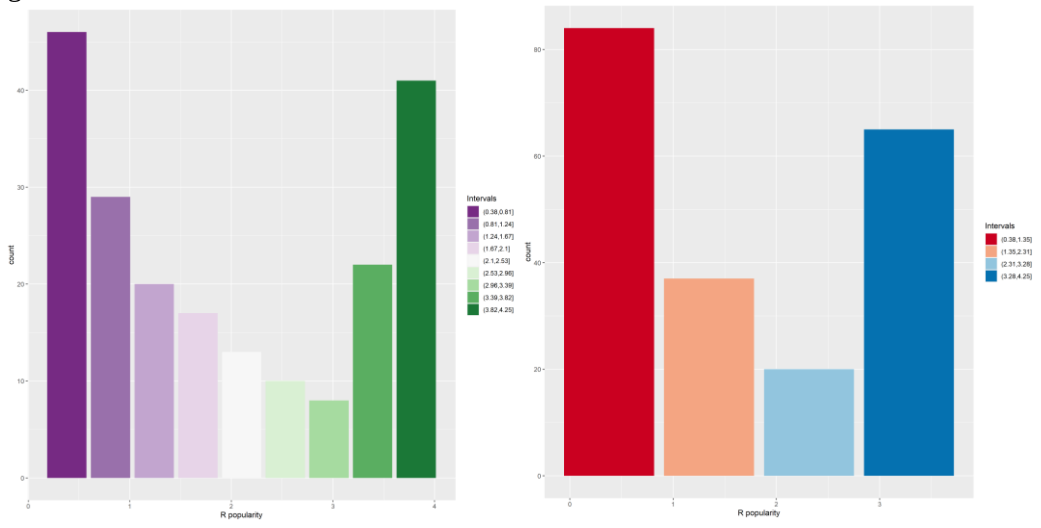


Figure 7: Different histograms for one time series

In addition, it is possible to adjust the graphs of the time series themselves, their combination and presentation, a script has been developed for this:

```
progs.df[, c("Date", row.names(stats.df))] -> progs.df
# returns  time series chart for languages indexed a through b
gen_plot <- function(a,b, polar = F) {
    progs.df[, c(1, a:b)] %>% melt(id = "Date") -> data.df
    if(!polar) {
        (ggplot(data = data.df, aes(x = Date, y = value, fill = variable, col = variable))+
        geom_line()) %>% return
    }
    else {
        (ggplot(data = data.df, aes(x = Date, y = value, fill = variable, col = variable))+
        geom_line(size= 1.5) + coord_polar()) %>% return
    }
}
lang_top1.plot <- gen_plot(2,5) + labs(col = "Programming Lang", y = "Popurality")
lang_top2.plot <- gen_plot(6,9) + labs(col = "", y = "Popurality", x = "")
lang_top3.plot <- gen_plot(10,13) + labs(col = "", y = "", x = "")
lang_top4.plot <- gen_plot(14,17) + labs(col = "", y = "Popurality", x = "")
lang_top5.plot <- gen_plot(18,21) + labs(col = "", y = "", x = "")
lang_top6.plot <- gen_plot(22,25) + labs(col = "", y = "Popurality", x = "Date")
lang_top7.plot <- gen_plot(26,29) + labs(col = "", y = "", x = "Date")
(lang_top1.plot /
(lang_top2.plot | lang_top3.plot) /
(lang_top4.plot | lang_top5.plot) /
(lang_top6.plot | lang_top7.plot)) -> main.plot
```

When studying the dynamics of time series, the smoothing mechanism can be used, for example, for exponential and median filtering, the functions will look simple:

```
# exponential smoothing
expsmooth <- function(data, alpha){
  # smoothing factor 0.1 <= a <= 0.3
  l = length(data)
  y0 = data[1]
  res <- c(alpha*data[1]+(1-alpha)*y0)
  for(n in 2:l){
    res <- append(res, alpha*data[n]+(1-alpha)*res[n-1])
  }
  return(res)
}
# median filtering
medianfiltering <- function(data){
  l = length(data)
  res <- c(data[1])
  for(n in 2:(l-1)){
    res <- append(res, max(c(min(c(data[n-1],
data[n]))),c(min(c(data[n],data[n+1]))),c(min(c(data[n-1],data[n+1])))))
  }
  res <- append(res, data[l])
  return(res)
}
```

But, for linear Kendel smoothing, there is a problem of redundancy in the construction of the smoothing function, for example, using all the parameters, the function will look like this:

```
test_w7 <- function(data){
  if(length(data)<7){
    print("The array is too small")
    return(NA)
  }
  l = length(data)
  #y(1,2,3)
  y1 = (13*data[1]+10*data[2]+7*data[3]+4*data[4]+data[5]-2*data[6]-5*data[7])/28
  y2 = (5*data[1]+4*data[2]+3*data[3]+2*data[4]+1*data[5]+0*data[6]-1*data[7])/14
  y3 = (7*data[1]+6*data[2]+5*data[3]+4*data[4]+3*data[5]+2*data[6]+data[7])/28
  res <- c(y1,y2,y3)
  for(val in 4:(l-3)){
    res <- append(res, (data[val-3]+data[val-2]+data[val-
1]+data[val]+data[val+1]+data[val+2]+data[val+3])/7)
  }
  #y(n, n-1, n-2)
  yn2 = (data[l-6]+2*data[l-5]+3*data[l-4]+4*data[l-3]+5*data[l-2]+6*data[l-
1]+7*data[l])/28
  yn1 = (-1*data[l-6]-0*data[l-5]+1*data[l-4]+2*data[l-3]+3*data[l-2]+4*data[l-
1]+5*data[l])/14
  yn = (-5*data[l-6]-2*data[l-5]+data[l-4]+4*data[l-3]+7*data[l-2]+10*data[l-
1]+13*data[l])/28
  res <- append(res, yn2)
  res <- append(res, yn1)
  res <- append(res, yn)
  return(res)
}
```

However, as can be seen, the matrix of coefficients is a mirror image. It can be simply “reflected” and a function can be constructed for different w, with the first half of the matrix and the division coefficients known:

```
# w = 7
wequal7 <- function(data){
  p = 7
  l = length(data)
  if(l<p){
    print("The array is too small")
    return(NA)
  }
  res <- c()
  # values
  values <- c(
    13,10,7,4,1,-2,-5,
    5,4,3,2,1,0,-1,
    7,6,5,4,3,2,1
  )
  addvalues <- c(28,14,28,7,28,14,28)
  # values
  mvalues <- matrix(values, nrow = p, byrow = F)
  mvalues <- valuemirror(mvalues)
  for(n in 1:((p-1)/2)){
    vres = 0
    for(v in 1:p){
      vres <- vres + mvalues[v,n] * data[v]
    }
    res <- append(res, vres/addvalues[n])
  }
  for(n in ((p+1)/2):(l-(p-1)/2)){
    vres = 0
    for(v in 1:p){vres <- vres + mvalues[v,(p+1)/2] * data[n-1-(p-1)/2+v]}
    res <- append(res, vres/addvalues[(p+1)/2])
  }
  for(n in (l+1-(p-1)/2):l){
    vres = 0
    for(v in 1:p){vres <- vres + mvalues[v,p-l+n] * data[l-p+v]}
    res <- append(res, vres/addvalues[(p-l+n)])
  }
  return(res)
}
```

From Fig. 8, it can be seen that the results are exactly the same, but the second option allows parameterizing the coefficients of Kendel's formula for any known w.

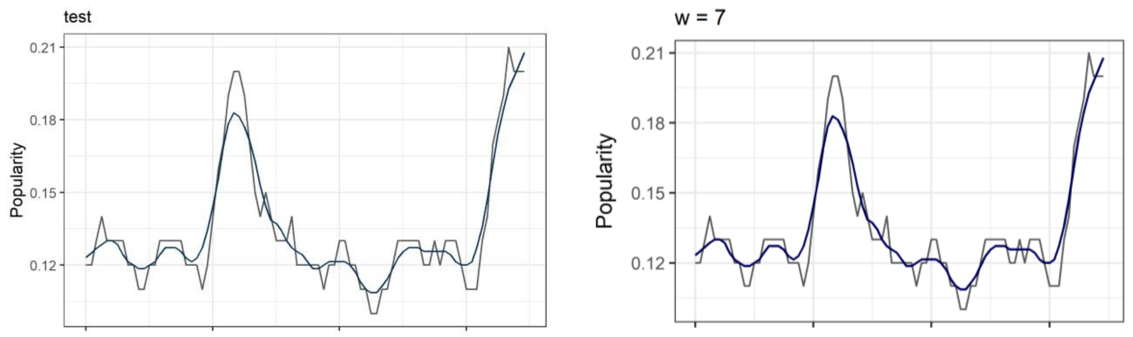


Figure 8: Comparison of two smoothing graphs

When conducting correlation analysis of time series, it is often necessary to build correlation matrices and display them in the form of heat maps, etc. However, the primary task is to construct a correlation field or dot plot, for example with ggplot2 it can be constructed as follows:

```
source("libs_config.r")
# кор. поле з лінією регресії для мов R та Julia
gplot_cor_1 <- ggplot(progs.df, aes(x = R, y = Julia)) +
    geom_point(aes(x = R, y = Julia), color="black", fill="#49ddd6",
    shape=21, alpha=0.5, size=4, stroke = 1) +
    geom_smooth(method=lm , color="#686868", fill="#0011f81f", se=F)
```

This option allows to add an automatically calculated regression line to the graph using the lm() function. To build heat maps, it is necessary first to determine the correlation matrix, and then set the graphic display. It is important to choose the right color scheme in order quickly to determine high correlation coefficients. It is also possible to adjust the display of the text. Examples of the correlation matrix are shown in Fig. 9.

```
progs.df[, c("Python", "Java", "C", "Dart", "Julia", "R")] %>% cor() -> cor.matrix_mini
cor.matrix %>% melt() -> heat.df
cor.matrix_mini %>% melt() -> heat.df_mini
heat.map_mini <- ggplot(data = heat.df_mini, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") #+
  scale_fill_gradient2(low = "#ca0033", high = "#0043fa", mid = "#ececec",
    midpoint = 0, limit = c(-1,1), space = "Lab",
    name="Cor coef") + labs(x = "", y = "") + geom_text(aes(label = round(value, 3)))
```

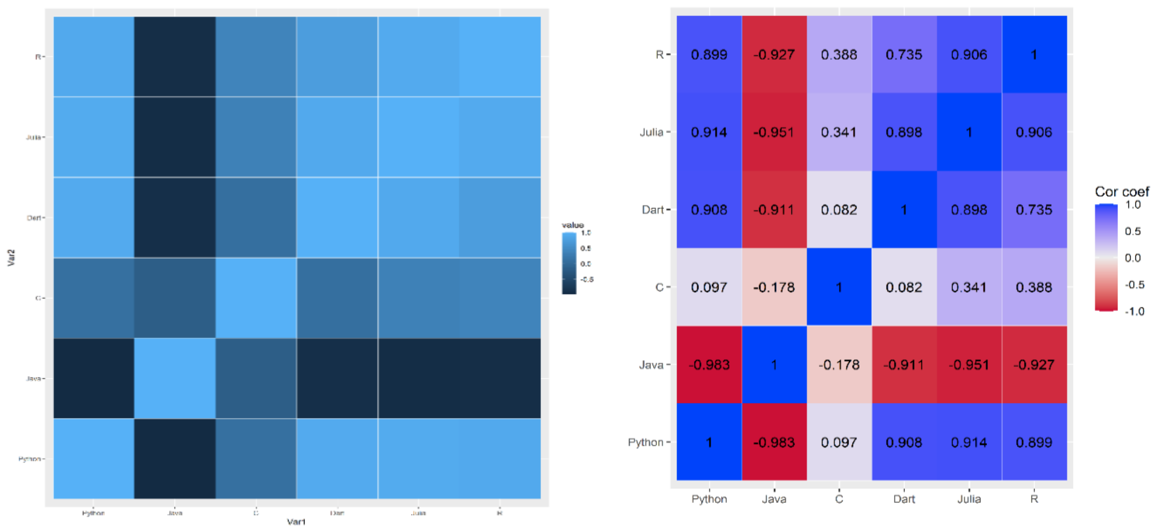


Figure 9: Correlation heat maps

The last element that must be completed according to the tasks set in the work is hierarchical clustering. To build a tree-like clustering, it is possible to use the built-in capabilities of the language:

```
source("task_1_stat\\create_statistics.r")
stats.df %>% apply(MARGIN = 2, FUN = function(x){return((x-min(x))/(max(x)-min(x)))}) ->
stats.df
stats.df %>% dist(method = "minkowski", p = 2) -> dist.m
hclust(dist.m) -> hc
plot(hc)
```

As a result, it is possible to get a dendogram from Fig. 10.

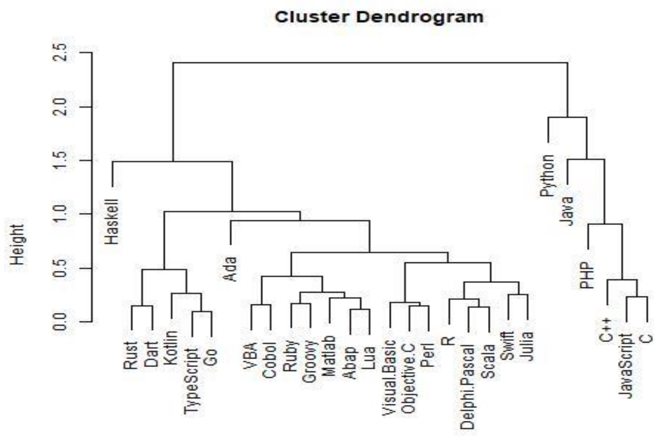


Figure 10: Generated clustering dendogram using the hclust function

However, there is a certain number of different methods of combining two clusters, and to achieve flexibility in the construction of clusters, it is necessary to define new clustering functions. So, the following functions have been developed for this work:

```
# conversion of the minimum number into matrix indices
convert_index <- function(index, n) {
    i <- n-1
    while(index > 0) {
        index <- index - i
        i <- i-1
    }
    c(n+index, n-i-1) %>% return
}
# property matrix normalization
normalize <- function(data.df) {
    data.df %>% apply(MARGIN = 2, FUN = function(x){return((x-min(x))/(max(x)-min(x)))})
%>% return
}
# returns the distance matrix according to the Minkowski metricgenerate_dist <-
function(data.df, p = 2) {
    data.df %>% dist(method = "minkowski", p = p) %>% as.matrix() %>% return
}
# the function of combining a new cluster with the deletion of two previous ones
syntez_cluster <- function(dist.matrix, ij, alpha_i, alpha_j, beta, gamma, path) {
    dist.matrix <- as.matrix(dist.matrix)
    n <- dim(dist.matrix)[1]
    i <- ij[1]
    j <- ij[2]
    cluster.name <- paste(rownames(dist.matrix)[i], colnames(dist.matrix)[j], sep = "|")
    #paste(paste(rownames(dist.matrix)[i], colnames(dist.matrix)[j], sep=" + "),
cluster.name, sep = " = ", dist.matrix[i,j]) %>% print
    new.names <- c(colnames(dist.matrix), cluster.name)
    dist.matrix %>% cbind(0) %>% rbind(0) -> dist.matrix
    colnames(dist.matrix)[n+1] <- cluster.name
    rownames(dist.matrix)[n+1] <- cluster.name
    first <- strsplit(cluster.name, "[|]")[[1]][1]
    write.table(data.frame(from = cluster.name, to = rownames(dist.matrix)[i], value =
dist.matrix[i,j], origin = first),
    file = path, col.names = F, row.names = F, append = T, sep = ",")
    write.table(data.frame(from = cluster.name, to = rownames(dist.matrix)[j], value =
dist.matrix[i,j], origin = first),
    file = path, col.names = F, row.names = F, append = T, sep = ",")
    dist.matrix %>%
    calc_cluster_dist(i = i, j = j, n = n+1, alpha_i = alpha_i, alpha_j = alpha_j, beta =
beta, gamma = gamma) %>%
    return
}
# distance calculation function for a new cluster
calc_cluster_dist <- function(dist.matrix, i, j, n, alpha_i, alpha_j, beta, gamma) {
    # in general case, it is necessary that i > j, however, we work with a lower triangular
matrix, where the condition is always fulfilled
    #if (i < j) i <- (j - i) + (j <- i);
    for (h in seq(n)[-i][-j]) {
        dist.matrix[n, h] <-
        dist.matrix[h, n] <-
        sum(c(alpha_i, alpha_j, beta, gamma) * c(dist.matrix[h,i], dist.matrix[h,j],
dist.matrix[i,j], abs(dist.matrix[h,i] - dist.matrix[h,j])))
    }
    dist.matrix[-i,-i][-j,-j] %>% return
}
```

With this approach, it is possible to combine capabilities of libraries for working with graphs and get more informative dendograms. In addition, circular dendograms can be constructed. For example, the results of one clustering can be presented in two graphs from Fig. 11, while the construction code looks like this:

```
library(igraph)
library(ggraph)
library(dplyr)
# construction of dendogram with clustering
create_dendogram <- function(dendo_df_file, dendo_name) {
  dendo.df <- read.csv(dendo_df_file)
  label.df <- data.frame(name = unique(c(dendo.df$from, dendo.df$to)))
  label.df %>% mutate(origin = NA, dist = NA, programming = NA) -> label.df
  for(i in seq_len(dim(label.df)[1])) {
    if (label.df$name[i] %in% dendo.df$from) {
      label.df$dist[i] = dendo.df$value[dendo.df$from == label.df$name[i]][1]
    }
    if (label.df$name[i] %in% dendo.df$to) {
      label.df$origin[i] = dendo.df$origin[dendo.df$to == label.df$name[i]][1]
      if(length(strsplit(label.df$name[i], "[|]")[[1]]) == 1) {
        label.df$programming[i] = label.df$name[i]
      }
    }
  }
  dendo.g <- graph_from_data_frame(dendo.df, vertices = label.df)
  dendogram.plot <- ggraph(dendo.g, layout = 'dendrogram', circular = F) +
  geom_edge_elbow(aes(colour = origin), alpha=0.8, edge_width = 2) +
  theme_graph(background = "white", plot_margin = margin(5, 5, 5, 5), base_size = 16) +
  geom_node_point(aes(size = dist)) +
  geom_node_point(aes(filter = leaf), alpha = 0.7, col = "black", size = 2) +
  ylim(-1.6, NA) +
  geom_node_text(aes(label = programming, filter = leaf, color = origin), angle=90 ,
hjust=1, nudge_y = -0.04)
  dendogram.circ <- ggraph(dendo.g, layout = 'dendrogram', circular = T) +
  geom_edge_elbow(aes(colour = origin), alpha=0.8, edge_width = 2) +
  theme_graph(background = "white", plot_margin = margin(5, 5, 5, 5), base_size = 15) +
  geom_node_point(aes(size = 1/dist), color = "#050022") +
  geom_node_point(aes(filter = leaf, ), alpha = 0.7, col = "black", size = 3) +
  ylim(-1, NA) +
  geom_node_text(aes(label = programming, filter = leaf, filter = leaf, angle = angle),
angle = 0, hjust = 1.1, vjust = 0.5)
```

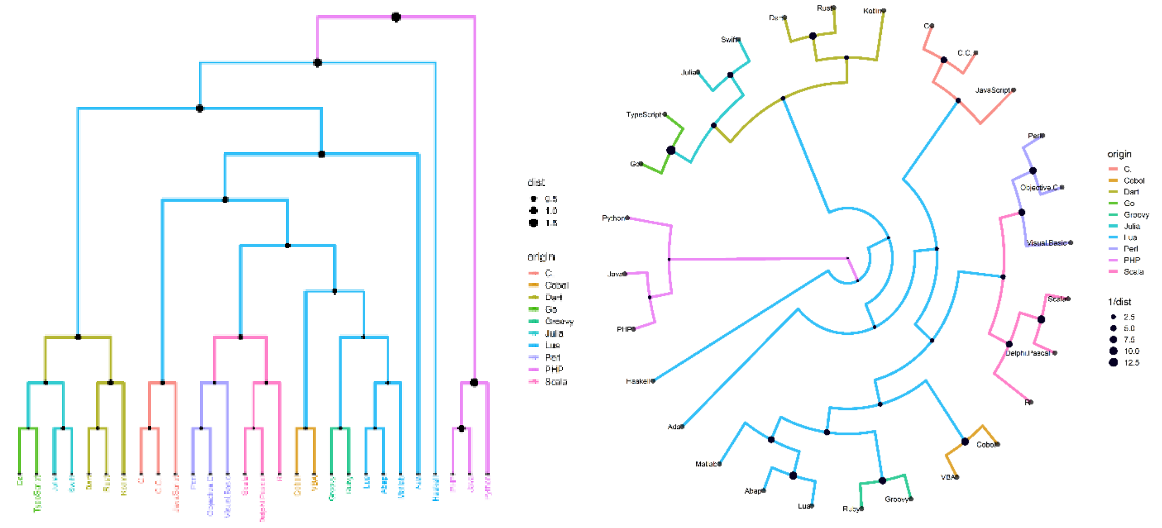


Figure 11: Normal and circular dendogram created by the script

5. Results

For the correct use of data, its filtering, sorting or clustering, it is necessary to conduct a classical intelligence analysis. If to build a graph with the data of all available programming languages, then the obtained result will not have a meaningful load, but in order to obtain a certain evolution of intelligence analysis, such a graph is shown in Fig. 12.

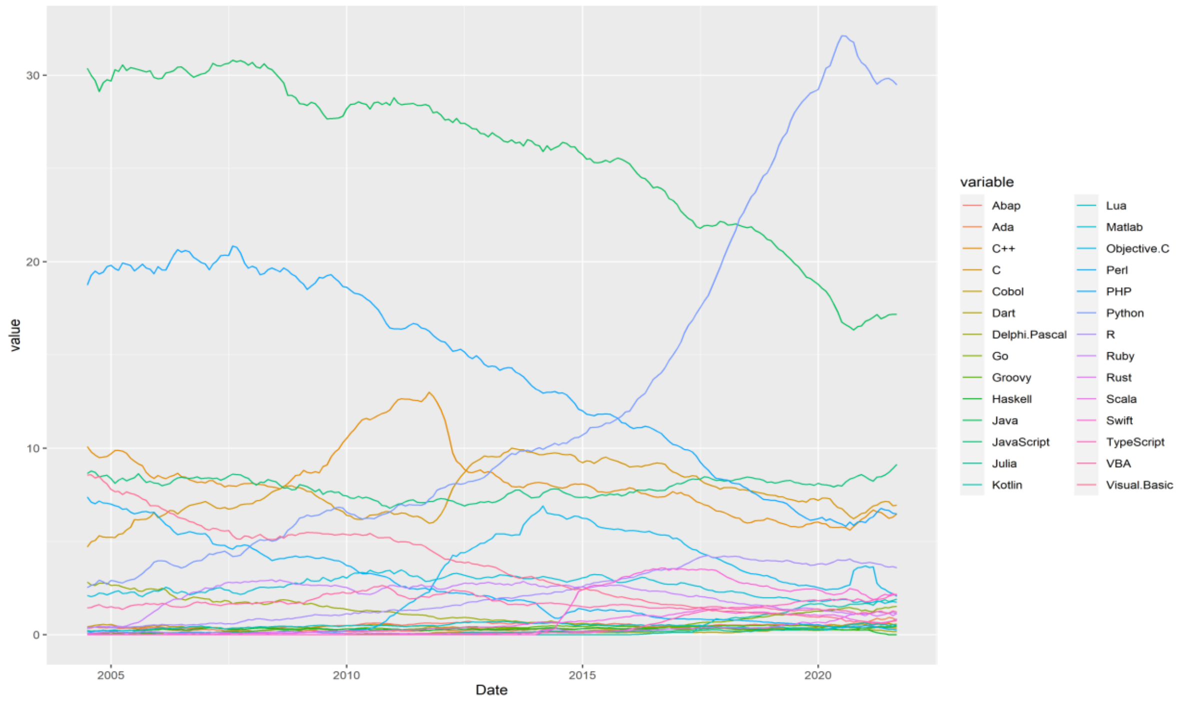
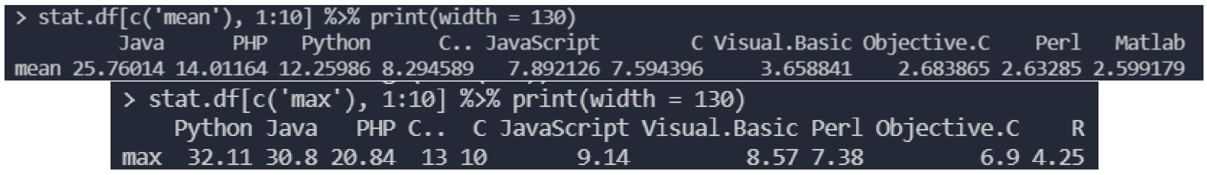


Figure 12: Graph of time series of popularity of programming languages

To improve the appearance and meaning of graphical display of data, it is necessary to divide languages into certain groups. It is best to do this according to statistical parameters. The following features are selected as the main statistical characteristics for these series: mean value, dispersion, standard deviation, median, minimum, maximum, range, coefficient of variation, skewness, and kurtosis. After sorting the data window by the average value, there is a possibility already to plot a set of graphs (Fig. 13-15) and get certain conclusions:

- The most popular language today is Python, the value of the index for September 2021 is 29.48, that is, almost a third of all queries refer to this language, in addition, the average popularity level is 12.26, and the maximum of 32.11 was reached in July 2020.
- On average, the most popular language during the observation period is Java - the average value is 25.76, and the value of the index for today is 17.18.

The average value of the top 10 programming languages, as well as the top ten in terms of the maximum, are shown as slices of the data frame in Fig. 13. In addition, the dynamics of changes in the popularity of the language can be considered in the polar coordinate system, as in Fig. 14.



```
> stat.df[c('mean'), 1:10] %>% print(width = 130)
         Java      PHP   Python       C.. JavaScript         C Visual.Basic Objective.C     Perl    Matlab
mean 25.76014 14.01164 12.25986 8.294589    7.892126 7.594396     3.658841    2.683865 2.63285 2.599179
```

```
> stat.df[c('max'), 1:10] %>% print(width = 130)
    Python Java   PHP C..  C JavaScript Visual.Basic Perl Objective.C    R
max  32.11 30.8 20.84  13 10       9.14         8.57 7.38         6.9 4.25
```

Figure 13: Execution of requests to print parts of the data window

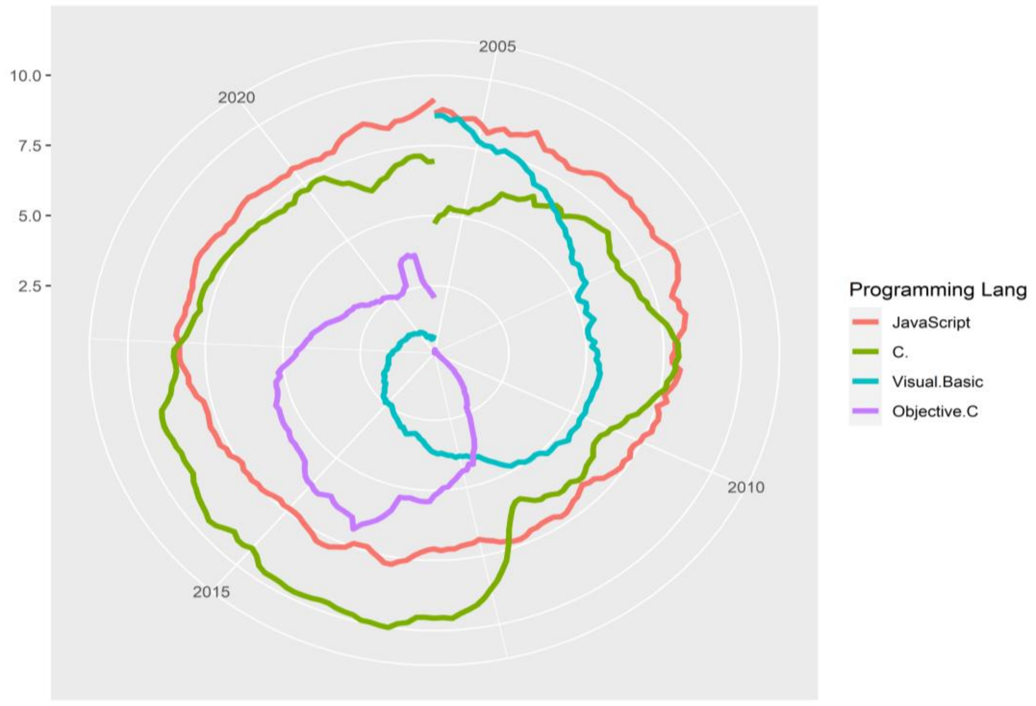


Figure 14: Time series graph of the second group of languages in the polar system

For example, this graph shows the uniformity of the distribution of the popularity of the JavaScript language. However, for correct presentation of the hypothesis about the distribution, it is necessary to use the mechanism of construction of histograms and the distribution function. To conduct the experiment, consider the programming language R. From Fig. 4 shows that the popularity of the language grew up to a certain time, and then stabilized, and it is impossible to determine whether the time series is subject to a certain distribution. As a result of using the constructed functions, a histogram in Fig. 16 is represented. This histogram shows that the popularity of R language does not lend itself to standard density functions, however, it is possible to construct an empirical probability distribution function to get a higherlevel picture of the distribution of the data. For this, function has been created that uses the same parameters as the histogram. It can be seen that the distribution function does not have a clear picture to correspond to the known distributions, perhaps this is explained by the instability of the parameters that determine the popularity of a particular programming language. When examining the behavior of some time series, there is a problem of a large range of data in local intervals, for example, for the Dart programming language, the popularity in the period after 2011 (ie, from the first non-zero values) does not have a clear or convenient trend picture.

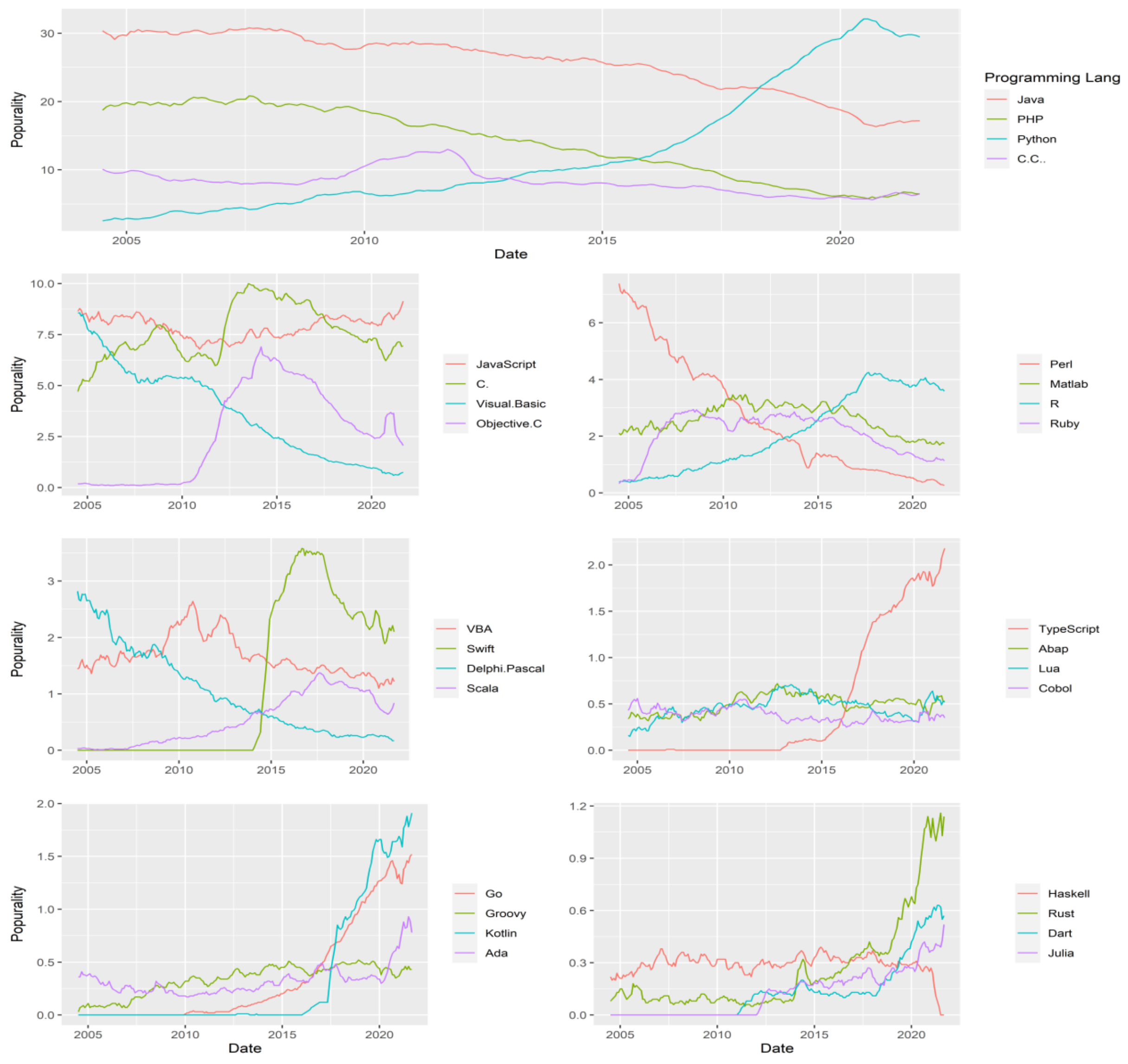
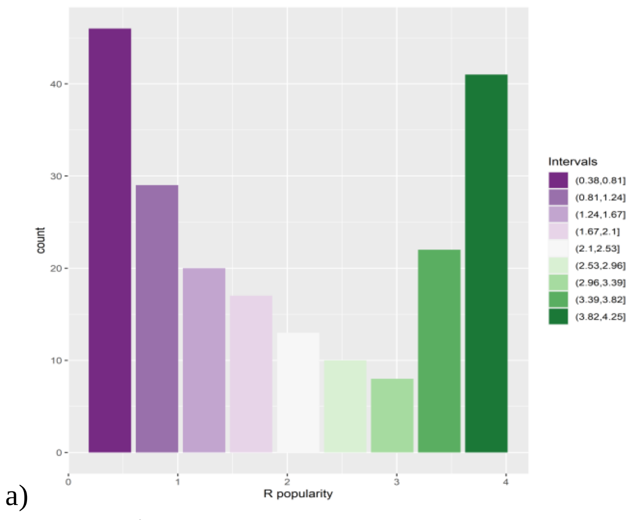


Figure 15: Set of time series graphs divided into groups of 4 by the average value.



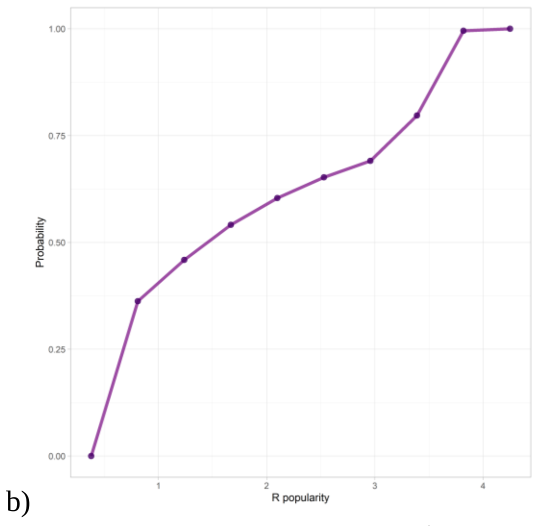


Figure 16: a) Histogram of the time series of the popularity of the R language and b) Empirical probability distribution function for a time series.

Smoothing mechanism was used to highlight the trend line close to the initial data. The following smoothing methods were used in the work (Fig. 17-19):

- Moving average: by 3, 7, 11, 13, 15 points and 7 points of the neo-oil formula;
- Exponential: at alpha 0.2 and 0.3;
- Median filtering.

The results of applying smoothing methods are shown in Fig. 17 – linear moving average, and Fig. 19 - non-linear and median. As was shown in the Experiments section, smoothing functions can be of different levels of complexity. From the graph in Fig. 17, it can be seen that when smoothing at points 11, 13, 15, a strong distortion is obtained, and when three points are used, the difference is minimal. To investigate the effectiveness of the moving average method, we calculate the correlation coefficient for real data and smoothing at w=5 and w=13. The corresponding calculations are shown in Fig. 18.

Nonlinear smoothing is shown in Fig. and has the best-fitting form, and it rejects most outliers. The worst result can be considered when using the exponential method with alpha equal to 0.2, and as can be seen from Fig. 19, the correlation coefficient is 0.83. If considering the obtained results, there is general cyclical one might say sinusoidal dynamics of development of language popularity. Two main peaks may have occurred during the release of the popular framework in the Dart language - Flatter, in general, the cyclical dynamics has a tendency to grow.

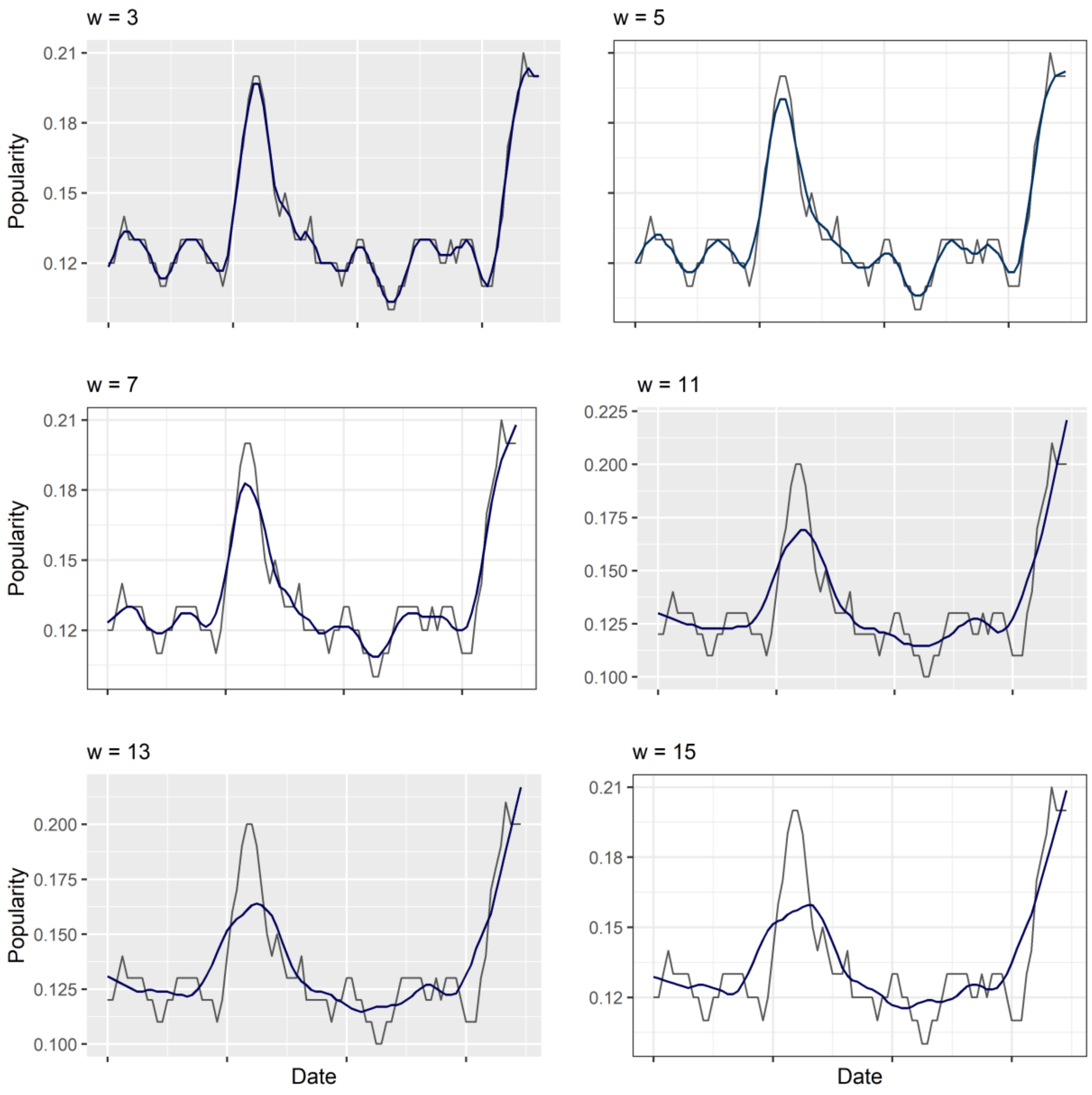
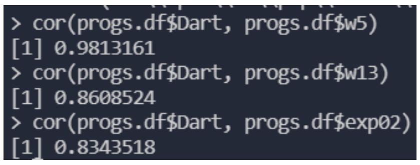


Figure 17: Graphic display of moving average methods with different number of points



```
> cor(progs.df$Dart, progs.df$w5)
[1] 0.9813161
> cor(progs.df$Dart, progs.df$w13)
[1] 0.8608524
> cor(progs.df$Dart, progs.df$exp02)
[1] 0.8343518
```

Figure 18: Results of correlation coefficients calculation

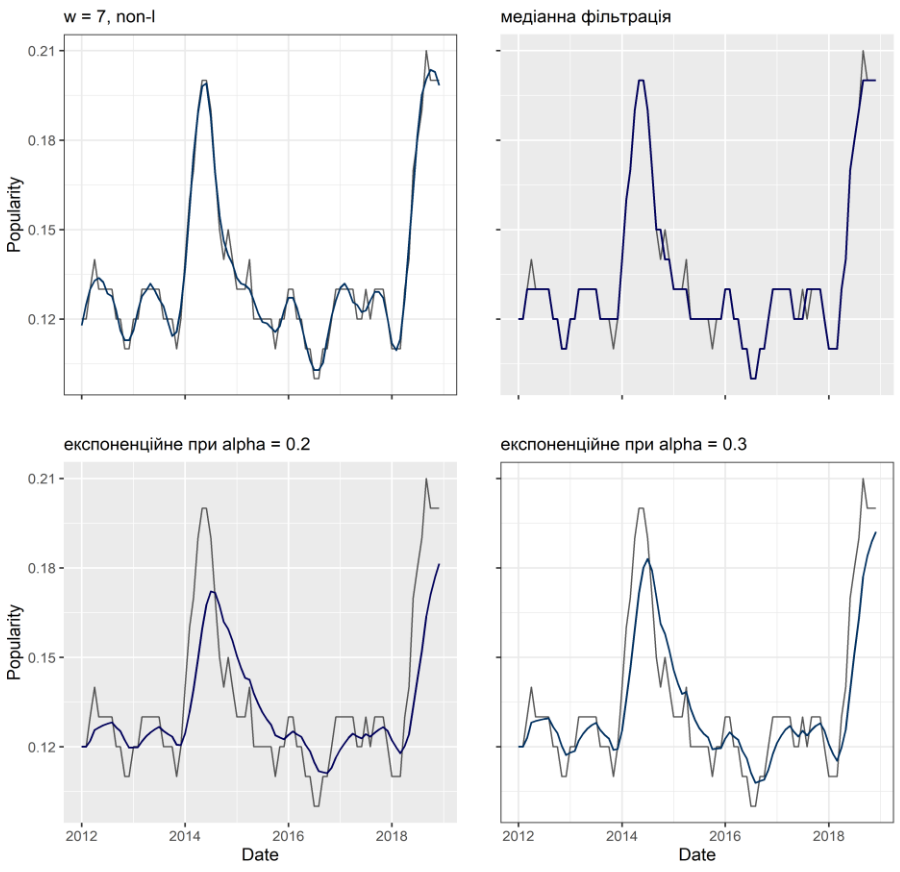


Figure 19: Graphical results of smoothing by non-linear methods

In order to consider the development of the popularity of all languages at the same time, it is first necessary to simplify the task and consider the relationship between the development of some languages (Fig. 20). To do this, first consider certain logical assumptions:

1. New programming languages (Go, Dart, Julia, Nim, etc.) usually do not affect the popularity of old languages, because new languages solve modern problems, and established languages have their permanent niche. From this it can be assumed that there is no definite linear relationship between Dart and C#.

2. Programming languages Java and Python are considered the biggest "enemies", completely different style in syntax, speed, community, etc. However, it is quite logical to assume that the growth of the popularity of the Python language has a strong influence on the popularity of Java, in particular, it should decrease.

3. Since data analysis, data science, artificial intelligence and neural networks are currently the most popular for discussion and development in IT, languages such as R or Julia are constantly increasing in popularity, and therefore there should be a direct linear relationship between the popularity of these languages, and both should grow.

If we interpret the logical assumptions in the language of correlation analysis, then the first assumption speaks about the absence of a linear relationship between the data and the correlation coefficient approaching zero, the second - there is a negative linear relationship with a correlation coefficient equal to -1, and the last one about a positive linear relationship connection with a coefficient of +1. In order to confirm these hypotheses, correlation fields were first constructed. To finally confirm the hypothesis, it is necessary to calculate the correlation coefficient.

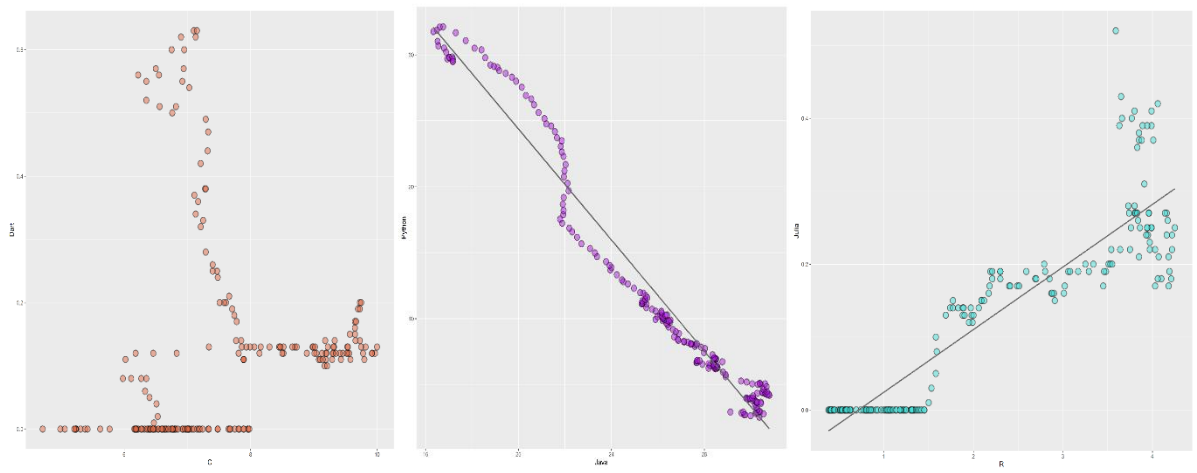
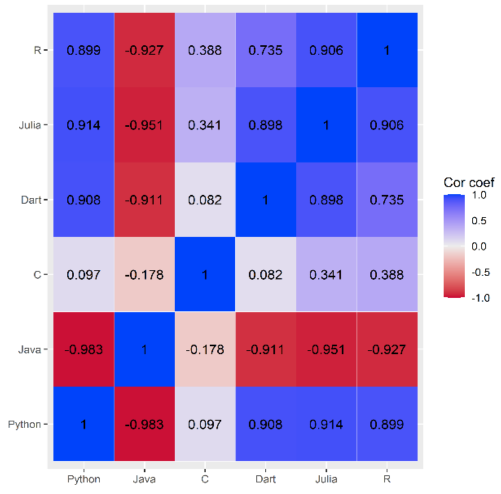


Figure 20: a) Correlation field for Dart and C languages and regression line for b) Python and Java languages and c) Julia and R languages

From the graphical representation of the dependence, it can be already seen that the initial assumptions are correct, however, to achieve accuracy, a heat map was constructed based on the correlation matrix for the languages described above. From the map in Fig. 20 it is visible that the correlation coefficient between Dart and C languages is close to zero, namely 0.082, and between Julia and R the coefficient is equal to 0.906. The most interesting thing is the behavior of the correlation coefficient for Java, it is less than zero in almost all languages, and therefore there is a tendency to decrease the popularity of the language in comparison with all other languages. However, Java and Python have the highest coefficient by module, which confirms the second hypothesis.

However, there is a certain problem in the presence of correlation relations between each language, and not all of them have a certain logic. To study the general behavior of languages with each other, it is possible to build a general heat map, which is shown in Fig. 21-22.



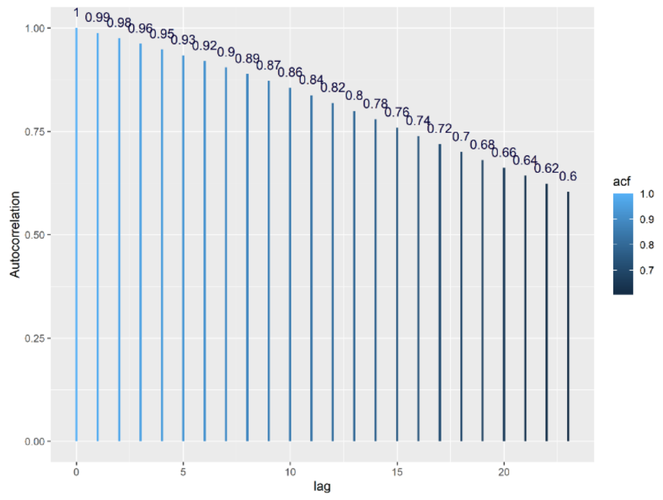


Figure 21: Heat map of the correlation matrix

After analyzing the correlation coefficients of all languages with each other, there is an assumption that, in fact, the dependencies between languages should be considered at a higher level, for example, taking into account the connections between the purposes of languages, trends in the modern IT market, etc. But it is possible individually to study the popularity dynamics of certain programming language. The growing popularity of the Python language has been mentione, and graphically it is also visible. However, there is another statistical method for checking the presence of trend in a time series - autocorrelation. Using the acf function of the R language, autocorrelation was calculated at a lag of 0-23. The result is presented in the form of a graph in Fig. 21-22. It can be seen from the correlogram that even at the last lag the coefficient is above 0.5, and it is quite obvious that the time series has a linear trend.

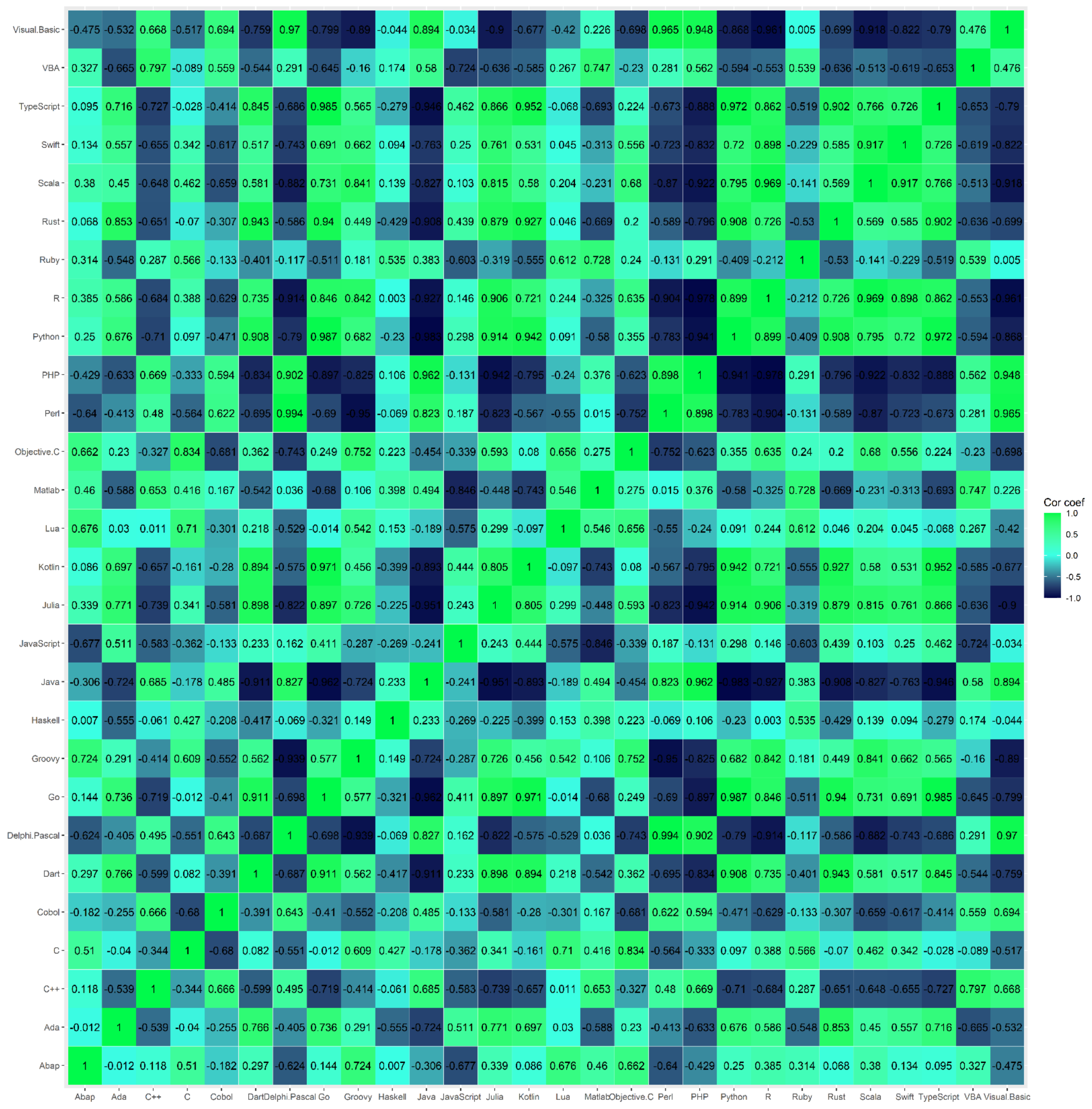


Figure 22: Correlation heatmap for all programming languages from experimental data

The logical conclusion of the intelligence analysis of data on the popularity of programming languages is to conduct a simple cluster analysis. In the course of the task, clustering was carried out with different parameters, namely with different metrics and unification strategies. In all cases, the Minkowski metric was used, with a variable parameter p. Two graphical approaches were used to display the results - conventional dendograms and circular ones. The vertices of the union are marked on ordinary dendograms proportional to the distance to the union, that is, the greater the distance, the larger the vertex, and in circular dendograms it is inversely proportional. In addition, the dendograms have a colored grouping, which is tied to the programming languages that are the initial nodes for a particular cluster.

1. Clustering at p = 2, the nearest neighbor joining strategy is a classic approach to clustering. As can be seen from the dendogram in Fig. 23, a different number of clusters can be distinguished at different levels, but those programming languages that were added at the very end - Ada, PHP, Haskell, Java and Python - are distinguished. These languages are unique enough to be clustered by the residual principle, that is, with this metric and clustering strategy, these languages have no relatives. On the other hand, some languages had a sufficiently small distance between them, for example, Go and TypeScript have a metric of about 0.09, which is why they form the first cluster. According to the colors, Go is the starting vertex for the final cluster.

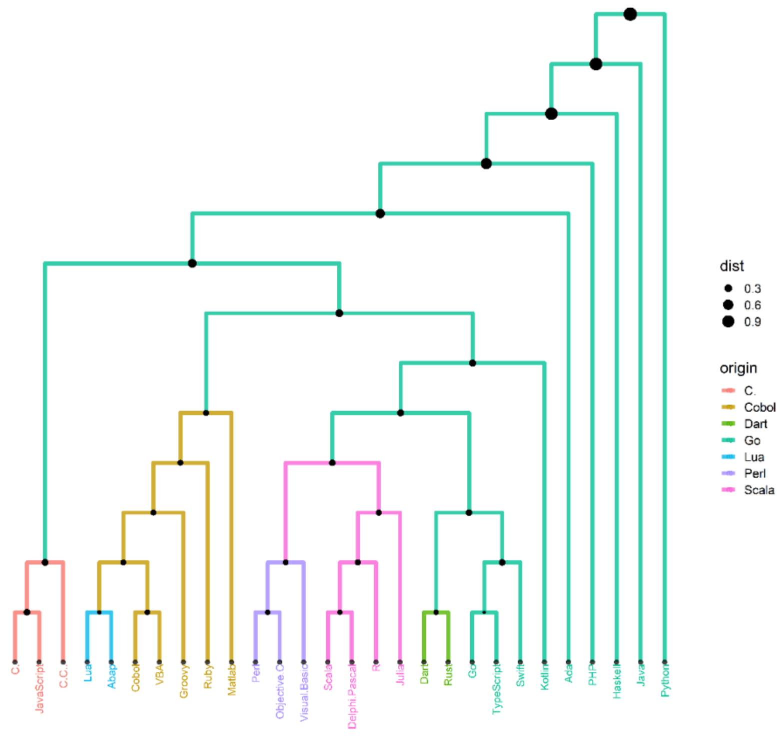


Figure 23: Dendogram of clustering 1

2. Clustering at p = 2, the unification strategy is flexible at αi = αj = 0.3535, β= 0.293, γ = 0. When using another unification strategy, we have a slightly different clustering, which in general does not have a fundamental difference from the previous one. The main difference is the appearance of PHP and Java language integration node before entering the common terminal cluster. Another difference is the new initial vertices, if in the previous configuration it was the Go language, then in this case Groovy (for less popular languages) and PHP (for the most popular) dominate.

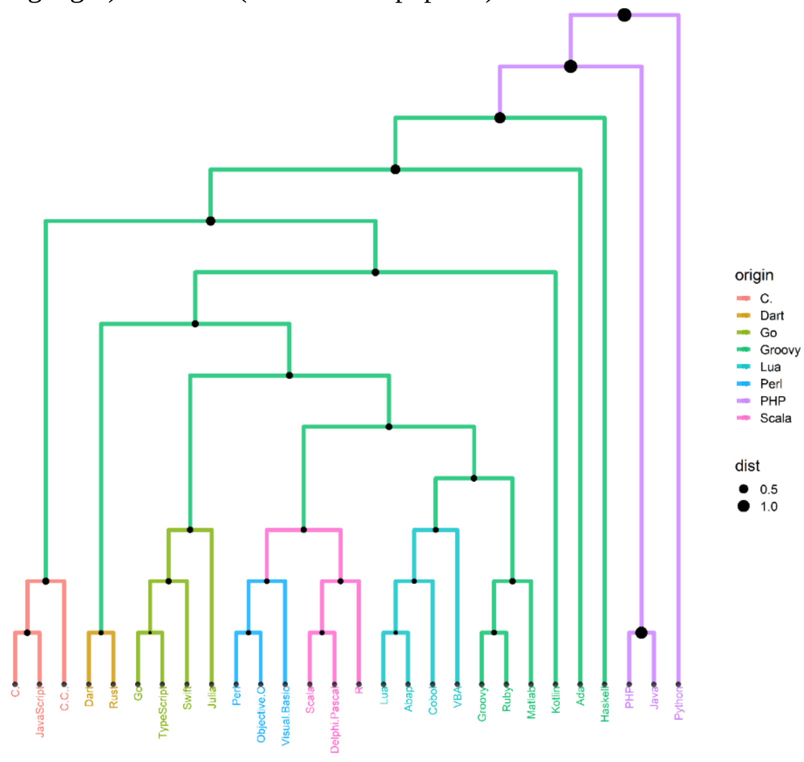


Figure 24: Dendogram of clustering 2

3. Clustering at p = 3, pooling strategy of the far-neighbor. With such parameters, different picture was obtained as a result of clustering. First, cluster of PHP, Java and Python languages was formed, which is the top 3 in terms of average value. Kotlin was added to the cluster with Dart, Rust, TypeScript, Go, which was not presented in this cluster with other parameters. In addition, Lua is now the main root for clusters. In addition, with such a configuration, the tendency to form clusters at the middle level remains, for example, languages C/C++, C# and JavaScript stick to one cluster almost always.

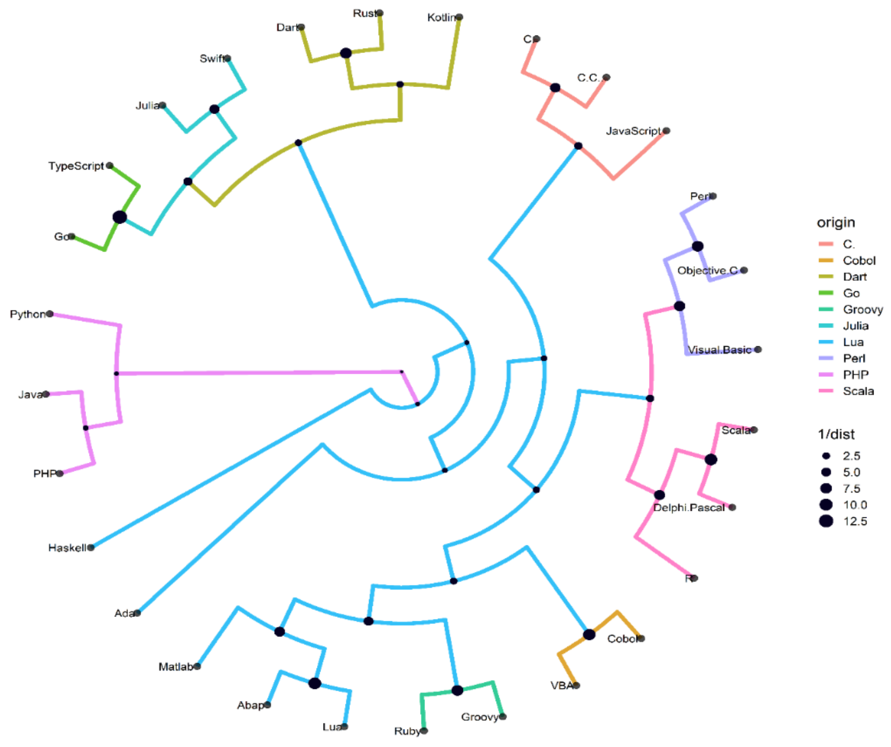


Figure 25: Circular dendogram of clustering 3

4. Clustering at p = 0.5, pooling strategy of the nearest neighbor. This clustering is interesting because of the parameter p, because it is less than 1.

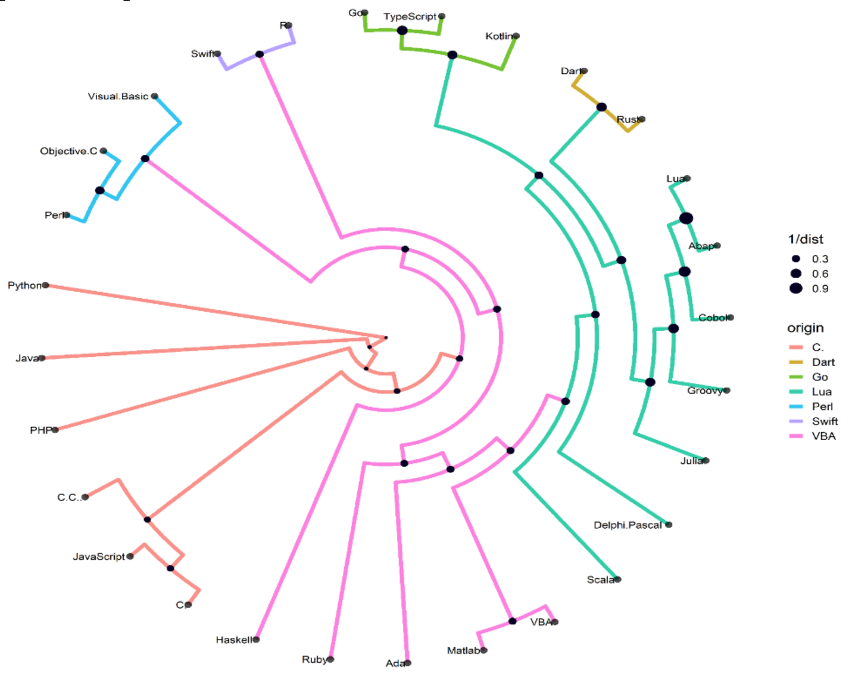


Figure 26: Circular dendogram of clustering 4

From the diagram it can be seen that separate cluster of C, C++ and JavaScript languages, a large cluster of Scala, Julia, Cobol and Lua languages can be distinguished, and again a cluster of languages that are far from all others - Python, Java and PHP, which will be added at the end

6. Discussions

The first important note that characterizes all data in general is the complete diversity of data for each language. The dynamics of the development of the popularity of each language during one period are maximally diversified, but at the same time interconnected in a certain way. When examining the results, it was shown that it is possible to obtain the appearance of linear relationships between the popularity of different languages that have logical connections. However, some languages are not related to each other, their development and use are quite different, but there is a correlation between them. If we consider the heat map from Fig. 18, it can be seen that the popularity of Swift and R and Scala languages have a very high correlation, but in fact the languages are not related to each other, they operate in different niches. Correlation of languages speaks of simultaneous growth in popularity, and therefore can indicate general trends in the IT market, but not the specifics of each language. The question arises of building such a model that can take into account the dynamics of all languages at the same time, but at the same time the input data is only one variable - the date. Such a problem with the one-dimensionality of the input data gives rise to the need for a large amount of training data.

To begin with, conducting exploratory analysis, it was proven that the data is complete, qualitative, suitable for some statistical tests. When constructing statistics, it can be seen that the data have various asymmetries and excesses, and do not lend themselves to normal distributions, in addition, most graphs of time series have obvious non-constancy. Therefore, it is impossible to select statistical parameters for the data that will fully describe them. In addition, it is necessary to maximally correspond to the limitations of the data, for example, the sum of all values by rows is equal to the constant 100. This means that approaches such as networks with marginalization or Bayesian neural networks are not suitable. In addition to the quality, but at the same time chaotic data, there is another problem - a small number of records. This problem rejects the possibility of using multilayer neural networks.

However, it is possible to distinguish certain approaches to describe the data, but for this it is necessary to find a mathematical counterpart of the function that will describe the current state of the data. The first and main option is a vector function. For example, a parametric vector function with three coordinates $\mathrm{v(t)} = [\sin \mathrm{t}, \mathrm{t}^2, \cos \mathrm{t}]$ For our data, such a function will be 28-dimensional. The date will act as the parameter t in the data.

Such a function will be able to accurately describe the current state of the data if we choose a function depending on t for each language, such as an approximation or interpolation polynomial. Smoothing functions can be used to reduce the number of input points.

When constructing such a function, it is possible to conduct experiments with the search for dependencies, derivatives of different orders to study the function. However, if there is a problem of extrapolation, it is immediately possible to assert a large absolute error.

The first way to reduce the number of calculations and the error is to use the clustering results. Some clustering languages have such small distances that they can be treated as a single variable. Then it is possible to get a function with fewer dimensions. A reduction in error will occur when clustering is used to generate interpolation nodes.

However, it is still not possible to achieve a high predicted accuracy. Therefore, the next option is neuron-by-neuron of the network. Such neural networks differ from most in that they do not have a defined architecture at the beginning, but form it during the learning process, by adding a neuron or a layer. Such networks require many times less input data, but are not suitable for multidimensional results. To build a network that can produce vector of index values for the future with a small amount of input data, it is possible to use the approach of graph neural networks together with clustering. Dendogram is good initial network graph, and the distances between clusters can be used as coefficients when passing messages. When combining methods of graph and neuron-by-neuron networks, it is possible to get the smallest error when predicted. So, as a result of exploratory and cluster analysis, it was concluded that the data on the popularity of programming languages are suitable for research, but the methods of their analysis should be the most modern.

7. Conclusions

As a result of experimental and exploratory analysis of data concerning the programming languages popularity according to the PYPL index since 2004, the presence of various trends in the dynamics of the popularity of programming languages was revealed and the relationship between two different languages was established. In the case of a complex consideration of time series, a hypothesis was created regarding the complexity and absolute nonlinearity of the general trend using programming languages, which affects the choice of models for forecasting. Some programming languages have a sufficiently unique popularity that distinguishes them when constructing hierarchical cluster structures. Such languages are highly popular, have the most diverse fields of application and their own niches. An equally important result is the fact of conducting data research as such, because it indicates the correctness, integrity and informativeness of the data, due to which such data can be used for educational purposes.

8. References

[1] Devathon. Top 10 Best Programming Language Rankings. URL: https://medium.com/@devathon_/top-10-best-programming-language-rankings-e49a0def796c

[2] PYPL PopularitY of Programming Language. URL: https://pypl.github.io/PYPL.html.

[3] TIOBE Index for October 2021. URL: https://www.tiobe.com/tiobe-index/.

[4] Victor A. Bloomfield. Using R for Numerical Analysis in Science and Engineering. Minneapolis, USA: University of Minnesota, 2014.

[5] Most Popular Programming Languages Since 2004. URL: https://www.kaggle.com/muhammadkhalid/most-popular-programming-languages-since-2004.

[6] I. Rishnyak, Y. Matseliukh, T. Batiuk, L. Chyrun, O. Strembitska, O. Mlynko, V. Liashenko, A. Lema, Statistical Analysis of the Popularity of Programming Language Libraries Based on StackOverflow Queries, CEUR Workshop Proceedings Vol-3171 (2022) 1351-1379.

[7] M. Fedorov, A. Berko, Y. Matseliukh, V. Schuchmann, I. Budz, O. Garbich-Moshora, M. Mamchyn, Decision Support System for Formation and Implementing Orders Based on Cross Programming and Cloud Computing, CEUR Workshop Proceedings Vol-2917 (2021) 714-748.

[8] D. Koshtura, V. Andrunyk, T. Shestakevych, Development of a Speech-to-Text Program for People with Haring Impairments, CEUR Workshop Proceedings Vol-2917 (2021) 565-583.

[9] S. Tetiana, The method of education format ascertaining in program system of inclusive education support, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 2017, pp. 279-283.

[10] M.O. Medykovskyi, V.M. Teslyuk, O.B. Shunevych, The use of dynamic programming for the problem of the uniform use of wind plants, Technical Electrodynamics 4 (2014) 135-137.

[11] M. Medykovskyy, V. Teslyuk, O. Shunevych, Optimization of wind power stations structure by the dynamic programming method, Actual Problems of Economics, 152(2) (2014) 508-515.

[12] V. Litvinenko, P. Bidyuk, J. Bardachov, V. Sherstjuk, A. Fefelov, Combining clonal selection algorithm and gene expression programming for time series prediction, in: Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2005, pp. 133–138.

[13] V. Lytvyn, V. Vysotska, A. Rzheuskyi, Technology for the Psychological Portraits Formation of Social Networks Users for the IT Specialists Recruitment Based on Big Five, NLP and Big Data Analysis, CEUR Workshop Proceedings Vol-2392 (2019) 147-171.

[14] I. Rishnyak, O. Veres, V. Lytvyn, M. Bublyk, I. Karpov, V. Vysotska, V. Panasyuk, Implementation models application for IT project risk management, CEUR Workshop Proceedings Vol-2805 (2020) 102-117.

[15] A. Rzheuskyi, O. Kutyuk, V. Vysotska, Y. Burov, V. Lytvyn, L. Chyrun, The Architecture of Distant Competencies Analyzing System for IT Recruitment, in: Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT, 2019, pp. 254-261.

[16] M. Bublyk, V. Vysotska, L. Chyrun, V. Panasyuk, O. Brodyak, Assessing Security Risks Method in E-Commerce System for IT Portfolio Management, CEUR Workshop Proceedings Vol-2853 (2021) 462-479.

[17] A. O. Karpyak, O. M. Rybytska, Cluster Analysis of Motivational Management of Personnel Support of IT Companies, CEUR Workshop Proceedings Vol-3171 (2022) 1684-1693.
[18] V. Yunchyk, A. Fedonuyk, M. Khomyak, S. Yatsyuk, Cognitive Modeling of the Learning Process of Training IT Specialists, CEUR Workshop Proceedings Vol-2917 (2021) 141-150.
[19] A. Rzheuskyi, O. Kutyuk, O. Voloshyn, A. Kowalska-Styczen, V. Voloshyn, L. Chyrun, S. Chyrun, D. Peleshko, T. Rak, The intellectual system development of distant competencies analyzing for IT recruitment, Advances in Intelligent Systems and Computing 1080 (2020) 696–720. DOI: 10.1007/978-3-030-33695-0_47.
[20] L. Chyrun, P. Kravets, O. Garasym, A. Gozhyj, I. Kalinina, Cryptographic information protection algorithm selection optimization for electronic governance IT project management by the analytic hierarchy process based on nonlinear conclusion criteria, CEUR Workshop Proceedings Vol-2565 (2020) 205–220.
[21] V. Sokol, M. Bilova, A. Druzhynin, N. Cherkun, I. Perepelytsia, An Adaptive Algorithm for Effective Selection of Training Content for IT Professionals, CEUR Workshop Proceedings, Vol-2870 (2021) 1173-1183.
[22] L. Halkiv, O. Karyy, I. Kulyniak, Y. Kis, A. Tsapulych, The National System of Higher Education and Government Procurement for Its Services as Activators of the Development of IT Entrepreneurship, CEUR Workshop Proceedings Vol-2870 (2021) 1338-1349.
[23] V. Andrunyk, V. Pasichnyk, N. Antonyuk, T. Shestakevych, A Complex System for Teaching Students with Autism: The Concept of Analysis. Formation of IT Teaching Complex, Advances in Intelligent Systems and Computing 1080 (2020) 721-733.
[24] O. Duda, N. Kunanets, O. Matsiuk, V. Pasichnyk, Cloud-based IT infrastructure for "Smart City" projects, in: Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, 2018, pp. 389-410.
[25] M. Bublyk, O. Rybytska, A. Karpiak, Y. Matseliukh, Structuring the fuzzy knowledge base of the IT industry impact factors, in: Computer sciences and information technologies, 2018, pp. 21–24.
[26] O. Veres, O. Oborska, A. Vasyliuk, Y. Brezmen, I. Rishnyak, Problems and peculiarities of the IT project managment of ontological engineering for person psychological state diagnosing, CEUR Workshop Proceedings Vol-2565 (2020) 162–177.
[27] O. Veres, et. al., Development and Operations - The Modern Paradigm of the Work of IT Project Teams, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 2019, 3, pp. 103–106.
[28] B.E. Kapustiy, B.P. Rusyn, V.A. Tayanov, Peculiarities of Application of Statistical Detection Criteria for Problem of Pattern Recognition/Journal of Automation and Information Science 37(2) (2005) 30-36.
[29] S. Babichev, et. al., A fuzzy model for gene expression profiles reducing based on the complex use of statistical criteria and Shannon entropy, Advances in Intelligent Systems and Computing 754 (2018) 545-554.
[30] M. Voronenko, et. al., Analysis of the Effectiveness of an Investment Project Using Statistical Bayesian Networks, in: International Conference on Advanced Computer Information Technologies, ACIT, 2020, pp. 408–411.
[31] S. Babichev, O. Khamula, B. Durnyak, J. Škvor, Technique of gene expression profiles selection based on sota clustering algorithm using statistical criteria and Shannon entropy, Advances in Intelligent Systems and Computing 1246 (2021) 23–38.
[32] P. Bidyuk, I. Kalinina, A. Gozhyj, Methodology of constructing statistical models for nonlinear non-stationary processes in medical diagnostic systems, CEUR Workshop Proceedings Vol-2753 (2020) 36–45.
[33] I. Gorbenko, A. Kuznetsov, Y. Gorbenko, S. Vdovenko, V. Tymchenko, M. Lutsenko, Studies on Statistical Analysis and Performance Evaluation for Some Stream Ciphers, International Journal of Computing 18(1) (2019) 82-88.
[34] R. Kaminskyi, N. Kunanets, A. Rzheuskyi, Mathematical support for statistical research based on informational technologies, CEUR Workshop Proceedings Vol-2105 (2018) 449-452.
[35] I. Sokolovskyy, N. Shakhovska, Statistical modeling of diffusion processes with a fractal structure, CEUR Workshop Proceedings Vol-2488 (2019) 145–154.

[36] M. Bublyk, V. Feshchyn, L. Bekirova, O. Khomuliak, Sustainable Development by a Statistical Analysis of Country Rankings by the Population Happiness Level, CEUR Workshop Proceedings, Vol-3171(2022) 817-837.
[37] I. Khomytska, V. Teslyuk, The Method of Statistical Analysis of the Scientific, Colloquial, Belles-Lettres and Newspaper Styles on the Phonological Level, Advances in Intelligent Systems and Computing 512 (2016) 149–163.
[38] I. Khomytska, V. Teslyuk, Authorship and Style Attribution by Statistical Methods of Style Differentiation on the Phonological Level, Advances in Intelligent Systems and Computing 871 (2019) 105–118. doi: 10.1007/978-3-030-01069-0_8.
[39] A. Vasyliuk, Y. Matseliukh, T. Batiuk, M. Luchkevych, I. Shakleina, H. Harbuzynska, S. Kondratiuk, K. Zelenska, Intelligent Analysis of Best-Selling Books Statistics on Amazon, CEUR Workshop Proceedings Vol-3171 (2022) 1432-1462.
[40] T. Shestakevych, Modeling the Process of Analysis of Statistical Characteristics of Student Digital Text, CEUR Workshop Proceedings Vol-2870 (2021) 657-669.
[41] M. Bublyk, Y. Matseliukh, Small-Batteries Utilization Analysis Based on Mathematical Statistics Methods in Challenges of Circular Economy, CEUR Workshop Proceedings Vol-2870 (2021) 1594-1603.
[42] V. Vysotska, O. Markiv, S. Teslia, Y. Romanova, I. Pihulechko, Correlation Analysis of Text Author Identification Results Based on N-Grams Frequency Distribution in Ukrainian Scientific and Technical Articles, CEUR Workshop Proceedings Vol-3171 (2022) 277-314.
[43] B.O. Kapustiy, B.P. Rusyn, V.A. Tayanov, A new approach to determination of correct recognition probability of set objects, Upravlyaushchie Sistemy i Mashiny 2 (2005) 8-12.
[44] A.B. Lozynskyy, I.M. Romanyshyn, B.P. Rusyn, Intensity Estimation of Noise-Like Signal in Presence of Uncorrelated Pulse Interferences, Radioelectronics and Communications Systems 62(5) (2019) 214-222.
[45] N. Romanyshyn, Algorithm for Disclosing Artistic Concepts in the Correlation of Explicitness and Implicitness of Their Textual Manifestation, CEUR Workshop Proceedings 2870 (2021) 719-730.
[46] V. Pasichnyk, T. Shestakevych, N. Kunanets, V. Andrunyk, Analysis of completeness, diversity and ergonomics of information online resources of diagnostic and correction facilities in Ukraine, CEUR Workshop Proceedings Vol-2105 (2019) 193-208.
[47] H. Zhengbing, V. Yatskiv, A. Sachenko, Increasing the Data Transmission Robustness in WSN Using the Modified Error Correction Codes on Residue Number System, Elektronika ir Elektrotechnika 21(1) (2015) 76-81.
[48] V. Yatskiv, N. Yatskiv, S. Jun, A. Sachenko, H. Zhengbing, The use of modified correction code based on residue number system in WSN, in: Proceedings of the International Conference on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS. 2013, pp. 513-516.
[49] O. Veres, Y. Matseliukh, T. Batiuk, S. Teslia, A. Shakhno, T. Kopach, Y. Romanova, I. Pihulechko, Cluster Analysis of Exclamations and Comments on E-Commerce Products, CEUR Workshop Proceedings Vol-3171 (2022) 1403-1431.
[50] I. Lurie, V. Lytvynenko, S. Olszewski, M. Voronenko, A. Kornelyuk, U. Zhunissova, O. Boskin, The Use of Inductive Methods to Identify Subtypes of Glioblastomas in Gene Clustering, CEUR Workshop Proceedings Vol-2631 (2020) 406-418.
[51] Y. Bodyanskiy, A. Shafronenko, I. Klymova, Adaptive Recovery of Distorted Data Based on Credibilistic Fuzzy Clustering Approach, CEUR Workshop Proceedings Vol-2870 (2021) 6-15.
[52] S. Babichev, B. Durnyak, I. Pikh, V. Senkivskyy, An Evaluation of the Objective Clustering Inductive Technology Effectiveness Implemented Using Density-Based and Agglomerative Hierarchical Clustering Algorithms, Advances in Intelligent Systems and Computing 1020 (2020) 532–553. Cham. https://doi.org/10.1007/978-3-030-26474-1_37.
[53] P. Bidyuk, A. Gozhyj, I. Kalinina, V. Vysotska, Methods for Forecasting Nonlinear Non-Stationary Processes in Machine Learning, Communications in Computer and Information Science 1158 (2020) 470-485. DOI: 10.1007/978-3-030-61656-4_32.
[54] P. Bidyuk, A. Gozhyj, I. Kalinina, V. Vysotska, M. Vasilev, R. Malets, Forecasting Nonlinear Nonstationary Processes in Machine Learning Task, in: Proceedings of the IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP, 2020, pp. 28-32. DOI: 10.1109/DSMP47368.2020.9204077.

[55] O. Soprun, M. Bublyk, Y. Matseliukh, V. Andrunyk, L. Chyrun, I. Dyyak, A. Yakovlev, M. Emmerich, O. Osolinsky, A. Sachenko, Forecasting Temperatures of a Synchronous Motor with Permanent Magnets Using Machine Learning, CEUR workshop proceedings 2631 (2020) 95–120.

[56] N. Koval, A. Tryhuba, I. Kondysiuk, I. Tryhuba, O. Boiarchuk, M. Rudynets, V. Grabovets, V. Onyshchuk, Forecasting the Fund of Time for Performance of Works in Hybrid Projects Using Machine Training Technologies, CEUR Workshop Proceedings Vol-2917 (2021) 196-206.

[57] R. Yurynets, Z. Yurynets, O. Budiakova, L. Gnylianska, M. Kokhan, Innovation and Investment Factors in the State Strategic Management of Social and Economic Development of the Country: Modeling and Forecasting, CEUR Workshop Proceedings Vol-2917 (2021) 357-372.

[58] Y. Tverdokhlib, V. Andrunyk, L. Chyrun, L. Chyrun, N. Antonyuk, I. Dyyak, O. Naum, D. Uhryn, V. Basto-Fernandes, Analysis and Estimation of Popular Places in Online Tourism Based on Machine Learning Technology, CEUR Workshop Proceedings Vol-2631 (2020) 457-470.

[59] I. Bodnar, M. Bublyk, O. Veres, O. Lozynska, I. Karpov, Y. Burov, P. Kravets, I. Peleshchak, O. Vovk, O. Maslak, Forecasting the Risk of Cervical Cancer in Women in the Human Capital Development Context Using Machine Learning, CEUR Workshop Proceedings Vol-2631 (2020) 491-501.

[60] L. Podlesna, M. Bublyk, I. Grybyk, Y. Matseliukh, Y. Burov, P. Kravets, O. Lozynska, I. Karpov, I. Peleshchak, R. Peleshchak, Optimization Model of the Buses Number on the Route Based on Queueing Theory in a Smart City, CEUR Workshop Proceedings Vol-2631 (2020) 502-515.

[61] V. Lytvynenko, et. al., Hybrid Methods of GMDH-Neural Networks Synthesis and Training for Solving Problems of Time Series Forecasting, Advances in Intelligent Systems and Computing 1020 (2020) 513–531.

[62] P. Bidyuk, A. Gozhyj, I. Kalinina, V. Gozhyj, Analysis of uncertainty types for model building and forecasting dynamic processes, Advances in Intelligent Systems and Computing 689 (2018) 66-78.

[63] P. Bidyuk, A. Gozhyj, Y. Matsuki, N. Kuznetsova, I. Kalinina, Modeling and forecasting economic and financial processes using combined adaptive models, Advances in Intelligent Systems and Computing 1246 (2021) 395–408.

[64] V. Hnatushenko, V. Hnatushenko, N. Dorosh, N. Solodka, O. Liashenko, Non-relational approach to developing knowledge bases of expert system prototype, Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu 2 (2022) 112-117. https://doi.org/10.33271/nvngu/2022-2/112.

[65] R. Sytnyk, V. Hnatushenko, V. Hnatushenko, Decentralized Information System for Supply Chain Management Using Blockchain, in: International Workshop on Intelligent Information Technologies and Systems of Information Security, CEUR Workshop Proceedings Vol-3156 (2022) 587-598.

[66] T. Basyuk, A. Vasyliuk, Approach to a subject area ontology visualization system creating, CEUR Workshop Proceedings Vol-2870 (2021) 528–540.

[67] T. Basyuk, A. Vasyliuk, Design and Construction of a Popularizing Internet Resources System Using External Ranking Factors, CEUR Workshop Proceedings Vol-3171 (2022) 437–450.

[68] A. Vasyliuk, T. Basyuk, Construction features of the industrial environment control system, CEUR Workshop Proceedings Vol-2870 (2021) 1011–1025.

[69] O. Mediakov, T. Basyuk, Specifics of Designing and Construction of the System for Deep Neural Networks Generation, CEUR Workshop Proceedings Vol-3171 (2022) 1282–1296.

[70] V. Davydov, D. Hrebeniuk, Development the resources load variation forecasting method within cloud computing systems, Advanced Information Systems 4(4) (2020) 128–135. https://doi.org/10.20998/2522-9052.2020.4.18.

[71] S. Krepych, I. Spivak, Forecasting system of utilities service costs based on neural network, Advanced Information Systems 4(4), (2020) 102–108. https://doi.org/10.20998/2522-9052.2020.4.14.

[72] S. Semenov, C. Weilin, L. Zhang, S. Bulba, Automated penetration testing method using deep machine learning technology, Advanced Information Systems 5(3) (2021) 119–127. https://doi.org/10.20998/2522-9052.2021.3.16.