Open-source Programming (OSP) Framework for Automatic Database Management for Discrete Event Simulation

Shahrokh Paravarzar, Mohammad Tabesh, Yashar Pourrahimian, Hooman Askari-Nasab
Mining Optimization Laboratory (MOL)
University of Alberta, Edmonton, Canada

ABSTRACT

Discrete event simulation is a simulation method has been used in forecasting and analysis the mining process and scheduling during last years. Discrete event simulation requires the pre- and post-processing of data and the results. In this paper, our focus is on pre-processing of data for simulation goal. Capturing the system behaviour and more precisely the event occurrence based on random variables and distributions is the goal of the pre-processing and data analysis in Arena simulation. The simulation process requires pre-processing steps that are time-consuming due to the huge amount of dispatching data that is providing by dispatching system during the mining process and retrieving from databases. Implementation of those stages is (semi)manual in available software. For this reason, Python open-source programming language implemented to provide an automatic procedure from database management and to retrieve data and input data preparation with the aim of discrete event simulation.

1. Introduction

System control and management are intensively software dependent process. These processes are handled mostly by commercial software. Commercial software refers to that kind of software which is produced for selling to the customers for the commercial purposes. Providing commercial software and programming is time, and labour-intensive process and control over this type of software can be achieved by copyright, contract law and software patent (Liberman, 1995). Technically control on commercial software based on the specific approach of end-users and development on the software structure is not possible most of the times; achieving to this objective, usually, clients have to spend more additional cost in addition to the original software price to having a suitable framework based on their system requirements.

Open-source Software (OSS) is the type of computer software that is released under a license that the users have a right of study, change, and distribute the software to several users and for any purposes (Laurent, 2004). Open-source software has properties of free redistribution, by means of having no restrictions for selling or giving away any component of that software or program code. The source code is available for this software and can be used for making any changes and derived works and it can be modified by all the users (Feller and Fitzgerald, 2002). The first open-source program was written by Ada Lovelace for calculating Bernoulli's Number using the Analytical Engine in earliest of the 18^{th} century. In the 1950s the first high-level programming language designed and developed by Germans to communicate instructions to the computer (Ceruzzi, 2003).

There are several programming languages for open-source programming such as Java, PHP, Python, Perl. Each language has its benefits and disadvantages from implementation and programming point

of view (RAO, 2014). Source code for these types of programs can be customized efficiently based on user needs and requirements and the bugs presented in the program can be found easily and fixed, and users from different countries and backgrounds can participate and collaborated. The codes from other languages can be translated whereas in case of licensed programming language it is not allowed (St. Amant, 2007). The source code utilization is unlimited for open-source code while for commercial software trial version is just available for few days only for it can be provided with a restriction on usage. The main drawback for open-source programming language is the version of the code which older versions may not be compatible with a new version of libraries and the original code should be updated after a while and messy lines of codes can be extremely confusing. Another weak point is that the solution for specific problems is highly related to the support provided by the user community. In the area of process and systems simulation there is a verity of simulation software such as AnyLogic, Arena, AutoMod, GPSS, Flexsim, ProModel, Simcad Pro, VisualSim that are commercial software and SimPy, Simula, Salabim, Simmer and etc. which are open-source software (Dagkakis and Heavey, 2016).

The Arena simulation commercial software package is the software provided by Rockwell software (2018) with the aim of simulation of processes and system performance analysis. This commercial software is widely used for in mine engineering with a purpose of mining process simulation. Each simulation process requires the pre- and post-processing stages. For achieving to this goal, Rockwell Company provided the third party packages as an add-on to Arena simulation software. Process analyzer is a tool for scenario analysis by execution of different simulation models and scenarios as useful tools for decision making. Output analyzer is a user interface that provides an easy and quick way for reviewing the simulation outputs. The main stage of every simulation process is providing the data from the database and the probability of their occurrence during the time that will be used in modelling the process. In discrete event simulation occurrence of the events are dependent on the probability of the occurrence of that specific event which is introduced to the simulation process by series of probability distribution function. There exist more than eighty different probability distribution functions that are shared in some properties with each other. Obtaining those functions require the process of retrieving data from the system of database and distribution fitting for each set of data or a process in a system. The input analyzer is a versatile tool that could provide the facility for doing this process. Arena takes the input file and tries to find the best fit for the data twelve different distribution model provided on input analyzer package and provide the best fit according to the least square error, chi-square test and Kolmogorov Smirnov test if they are available for the underlying dataset.

Real simulation process with high precision and detailed modelling requires a detailed analysis of the data that should be prepared from available databases or sampling on the real process. After data preparation, the files in *.csv or *.txt format take as an input for the analysis. In case of having several files, all these processes should be done manually, and all the expressions must be saved in an excel archive files and call by using VBA programming or manually to their place in the software which in case of having hundreds of data sets will be a time-consuming process.

In mining process simulation data for discrete simulation process are usually obtain from dispatching database. These databases consist of thousandths of records for several tables containing multiple columns. Processing these data and preparation is a complex task for modellers (Fig 1).

The most proper way of retrieving these data is the implementation of a database management system and link them to the input analyzer in order to have a rapid data derivation from the database management system. A relational database management system (RDBMS) usually use for this purpose. Retrieving data from RDBMS requires using the database abstraction layer (DBAL) and writing the quarries directly on the command line or indirectly on database management software (Coronel and Morris, 2016). According to DB-Engines, in June 2018, the most widely used systems were Oracle, MySQL (Free software), Microsoft SQL Server, PostgreSQL(Free software), IBM DB2, Microsoft Access, and SQLite (Free software).

There is not any coherent and direct link between database management software and input analyzer in Arena simulation software. Making this link requires the establishing the multiple links between database and Arena simulation. Stages consist of connecting to the database, running the queries, retrieving the data, data manipulation and saving them in a required format (asci, *.txt, *.csv) as an input for input analyzer and obtaining the probability distribution function the underlying variables or datasets.

The common process consists of generating the database from dispatching data, linking the database to Matlab database analyzer and writing the queries using a Matlab mathematical programming language and saving the files and manually import them to input analyzer. After fitting the distribution to each data set by retrieving from queries the user should save the expressions in a file for using in the simulation process. The process consists of several manual procedures in the whole of the process.

In this paper, the open-source Python programming language is used with the aim of generating a comprehensive and coherent process as a replacement for the input analyzer having the capability of processing the several datasets automatically. Our objective is to retrieve the required data from the database and directly obtain the distribution function expression file that is required for the simulation model. In this process, all the intermediate stages combine or omitted. The process is faster and the outputs can be used directly for simulation modelling process in Arena software. The comparison between the Matlab and Python process illustrated in Fig 1.

2. Methodology

Python as an open-source programming language provided different libraries for different purposes. As our goal is to design a process for retrieving the data and then obtain the most fitted distribution function to the data, it is necessary to use the following packages.

NumPy is the fundamental package for scientific computing in Python providing the arrays and matrices along with a large high-level mathematical function to operate on those arrays. This package can be replaced with the computational abilities of Matlab as their functions are C optimized. *Pandas* is a package for open-source data analysis and manipulation tools for tabular data. Dataframe and time series object is the main data structure on panda libraries. *Matplotlib* is the plotting library for the Python programming language that is synchronized with NumPy package. *Pyodbc* is an open-source Python module that provides access to the ODBC databases. These packages would be a replacement for Matlab Database explorer package. *SciPy* it is part of NumPy array object that is designed for scientific computing and technical computing. Some of the key algorithms of SciPy are hierarchical clustering, vector quantization, K-means, Discrete Fourier Transform algorithms, interpolation tools, optimization algorithms including linear programming, etc.

In our research, the SciPy statistic module is the main class that is used for our distribution fitting goal. More than eighty continuous random variable (RVs) and ten discrete random variables have been implemented in SciPy package. The process of retrieving data and distribution fitting is based on pseudo code illustrated in Table 1.

One of the major goals is to test how the obtained data from Python can be feed to Arena simulation as a distribution function as their parameters and the way to obtain them is a black box in Arena simulation as commercial software. For this reason, using the similar synthetic dataset, the fitted distribution in each case should compare with results in both Python and Arena software outputs. For this aim random variable for each distribution function generated with parameters mostly similar to the typical and standard shape of each distribution. For instance, in case of normal distribution synthetic normal data generated using Arena and then based on normal synthetic dataset normal distribution fitted on data using Python and Arena input analyzer package assuming that our generated random variable statistically accepted and match with a theoretical distribution. Parameters

for each model registered and compared with each other in order to match the parameters. Summary of the parameters and test results are shown in Table 2.

Table 1 Pseudo code for retrieving data and Python approach

Pseudo code in Python

```
Import Libraries
For i in a number of queries:
    Connect to database or server using Pyodbc
    Input SQL queries
    Execute SQL queries
    Save data in a temporary variable
    Close the Connection
    If data continuous:
      If data sufficient for draw distribution:
      Call best Fit Function:
            Generate a histogram of data obtain classes and their bin values
             Set best fit as normal and error infinity
            For j in the distribution model
                Try:
                Obtain fit parameters corresponding to the model
                Obtain theoretical PDF of the model
                 Compare Experimental Pa DF with a theoretical model
                Calculate squared error
                IF Error is less than the last one
                    Set the best distribution as the current
                    Set the parameter as the current parameter
                    Set error as current error
                Except:
      Calculate the Chi-square and Kolmogorov-Smirnov test
      Return best distribution, Errors, Test results
    elif:
       Consider Mean of data as a random exponential occurrence EXPO ()
    else:
      Calculate Discrete Empirical Function
    Save temporary model
    Plot and save the Figs
Write-Output Files
```

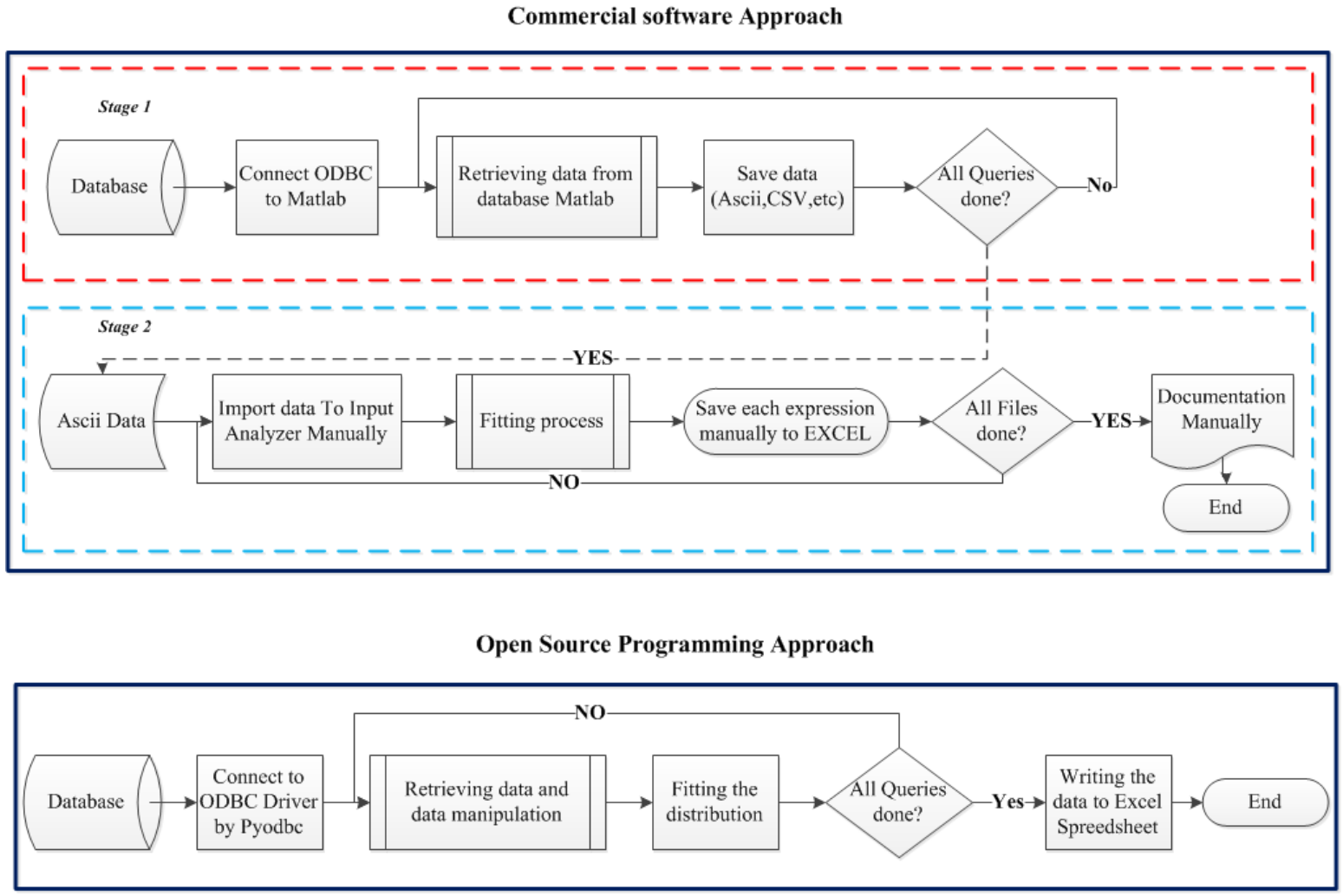

Fig 1 Comparison of data extraction and distribution model fitting for Matlab and Python

Table 2 Comparison of the distribution model output parameters for Arena input analyzer and Python

Distribution	Arena model	Python model	Link Python output to Arena
BETA	Min+Max*BETA(Alpha1, Alpha2)	beta(a, b, loc, scale)	Loc+Scale*BETA(a,b)
Erlang	Offset+Erlang(Exponential Mean, Erlang parameter k)	erlang(a, loc, scale)	Loc + erlang(scale,a)
Exponential	Offset + EXPO(Mean)	expon(loc, scale)	Loc + EXPO(scale)
Gamma	Offset + GAMM(Scale Parameter Beta, Shape Parameter Alpha)	gamma(a, loc, scale)	Loc + gamma(scale,a)
Normal	NORM(Mean, Standard Deviation)	norm(loc, scale)	NORM(loc,scale)
Triangular	TRIA(Minimum Value, Most Likely Value, Maximum Value)	triang(c, loc, scale)	TRIA(loc, loc+(c*scale), scale)
Uniform	UNIF(Minimum Value , Maximum Value)	uniform(loc, scale)	UNIF(loc,scale)
Weibull	WEIB(Scale Parameter Beta, Shape Parameter Alpha)	weibull_min(c, loc, scale)	loc + WEIB(scale,c)
Johnson	JOHN(Shape parameter 1(Gamma), Shape parameter 2(Delta),Scale Parameter Lamda, Location Parameter Xi)	johnsonsb(a, b, loc, scale)	JOHN(-2.19, 1.35, 2.60, 0.21)
Lognormal	Offset +LOGN(LogMean ,LogSTD)	Lognormal(s, scale, location)	No direct conversion

Table 3 Fitted model for normal distribution using synthetic data

Distribution	Parameter	Arena model	Arena Square Error	Python model	Python Square Error	Link Python output to Arena
Normal	Mean = 2.0, Std Dev = 10	NORM(1.71,9.98)	0.000077	NORM(1.746,9.712)	0.00006	NORM(1.746,9.712)

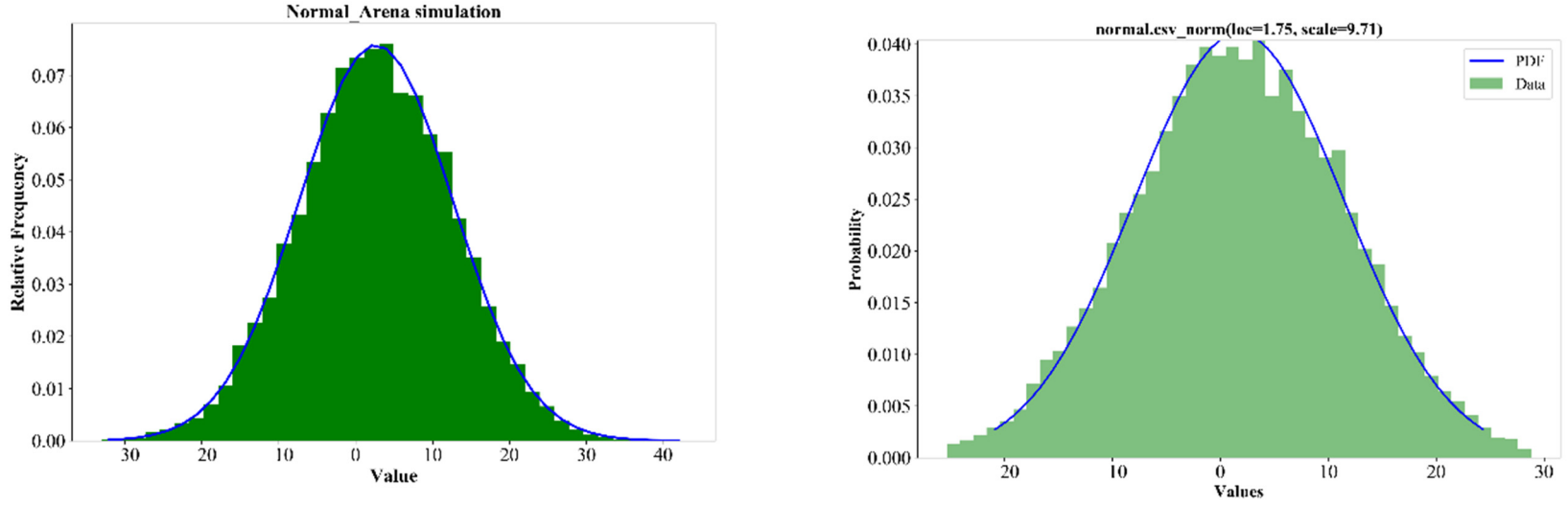

Fig 2 Comparison between input analyzer output and Python for normal distribution using synthetic data

Table 4 Fitted model for lognormal distribution using synthetic data

Distribution	Parameter	Arena model	Arena Square Error	Python model	Python Square Error	Link Python output to Arena
Lognormal	LogMean = 2, logSTD =0.3 , Offset = 5	6+LOGN(1,0.328)	0.000770	Lognormal(s=0.16,loc=5.07,scale=1.90)	0.12866	4.258 + LOGN(2.73,0.285)

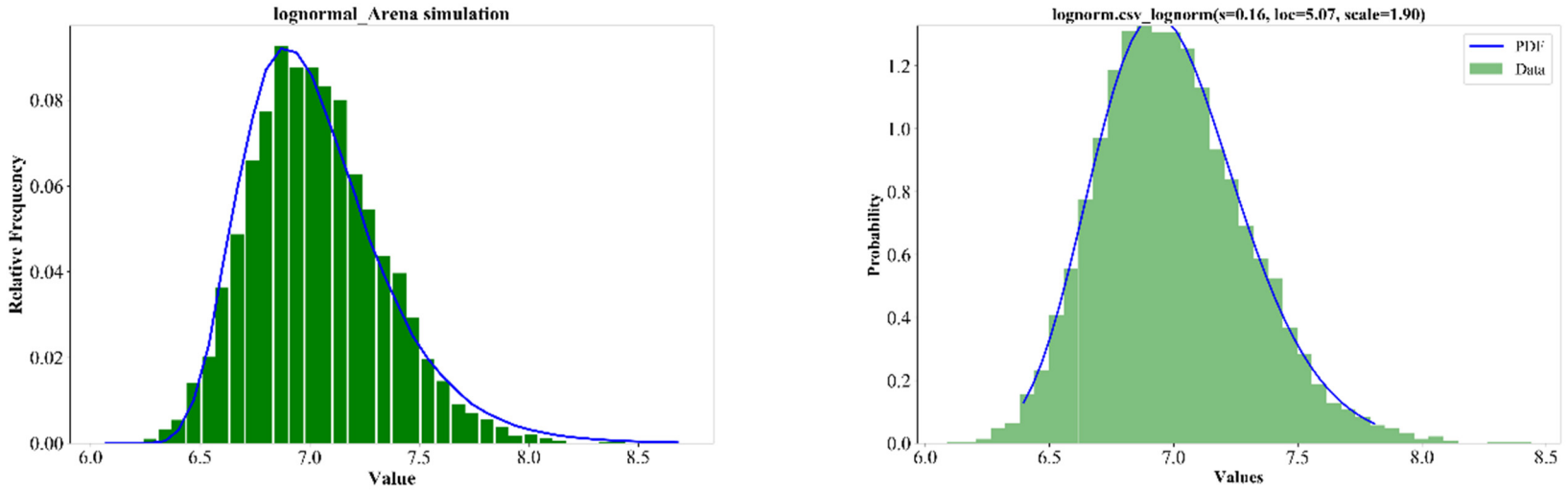

Fig 3 Comparison between input analyzer output and Python for lognormal distribution using synthetic data

Table 5 Fitted model for beta distribution using synthetic data

Distribution	Parameter	Arena model	Arena Square Error	Python model	Python Square Error	Link Python output to Arena
Beta	Alpha1= 0.5, Alpha2=0.5, Min=0, Max=2	2 * BETA(0.547,0.59)	0.001042	Beta(a=0.54,b=0.54,loc=0,scale=2.0)	1.00973	-0.002+ 2.002 * BETA(0.539,0.54)

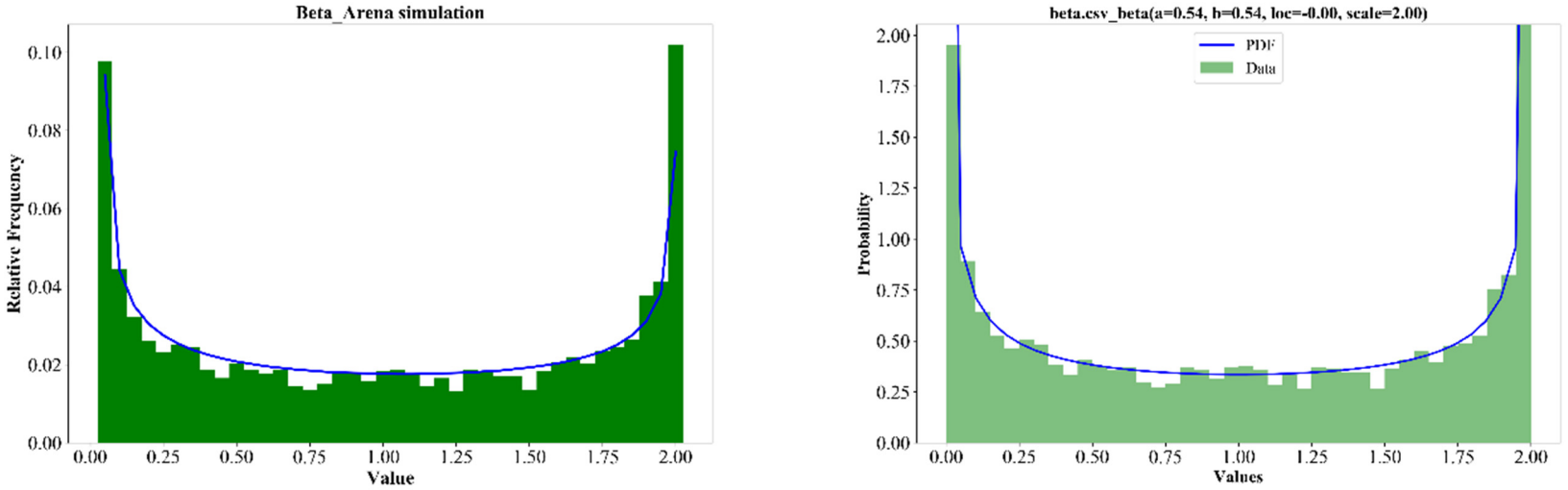

Fig 4 Comparison between input analyzer output and Python for beta distribution using synthetic data

Table 6 Fitted model for Gamma distribution using synthetic data

Distribution	Parameter	Arena model	Arena Square Error	Python model	Python Square Error	Link Python output to Arena
Gamma	Alpha = 2 beta=4 offset=5	5 + GAMM(4.04, 1.98)	0.000121	Gamma(a=1.97, loc=5.01,scale=4.05)	0.000725	5.011 + GAMM(4.055,1.971)

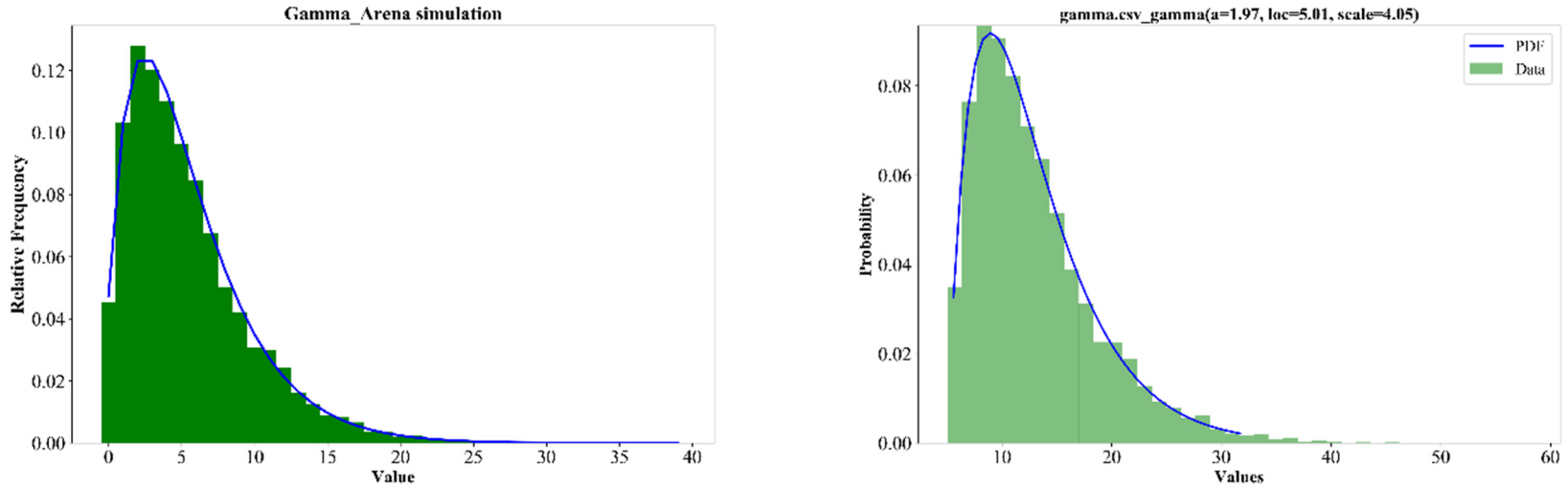

Fig 5 Comparison between input analyzer output and Python for gamma distribution using synthetic data

2.1. Model accuracy and goodness of fit

The goodness of fit is the statistical model provide the numbers for expert's judgment to assess how good our distribution fitted model is matched with the theoretical model. These values show the discrepancy between the observed values and the values obtained from the theoretical model. There are several ways of performing these tests. In Arena simulation, the tests provided by software are a square error, Chi-square test, Kolmogorov-Smirnov method. In the next section, a brief description of these tests is provided. According to the Arena's manual, the distribution fitting (curve fitting) is based on maximum likelihood estimators, and the beta, triangular and uniform distributions are exceptions. The beta distribution is fitted first with maximum likelihood estimators and then moments, and then the best result among them will accept as the main result. Triangular and uniform distributions use empirical rules to fit distributions on data. According to SciPy reference fitting distributions is based on maximizing a log-likelihood function and the outlier's values receive the penalty. The answer is not globally optimal, and it is only locally optimal or optimization may fail altogether.

2.1.1. Square error measurement

The quality of the distribution fitting primarily is based on the square error criterion. This condition can be defined as a sum of the square difference between the relative frequency of the data for ith interval and relative frequency of the fitted probability distribution function. This relationship can be defined as follows:

$$LSQ = \left[f_i - f(x_i)\right]^2 \tag{1}$$

Where f_i is relative frequency of data on interval ith and $f(x_i)$ is the relative frequency of the fitted probability distribution. $f(x_i)$ is obtained from cumulative distribution function by the difference of the two intervals $F(x_i) - F(x_{i-1})$ in a case that the cumulative distribution function is not available the $f(x_i)$ will be determine by numerical integration (Corder and Foreman, 2014; Bethea, 2018).

2.1.2. Kolmogorov-Smirnov Goodness-of-fit test

The Kolmogorov-Smirnov test is used to decide that if the sample under study comes from the same population for a specific distribution or not. The test is based on the empirical distribution function. The Kolmogorov-Smirnov test can be defined according to the following formulation

H_0: the data follow a specified distribution

H_1: the data do not follow the specific distribution

The Kolmogorov-Smirnov test statistic is defined as:

$$D = \max_{1 \le i \le N} \left(F(Y_i) - \frac{i-1}{N}, \frac{i}{N} - F(Y_i)\right) \tag{2}$$

Where F is the theoretical cumulative distribution function of the underlying data which should be in the form of the continues data and N is the number of samples. The test is a nonparametric test which Gaussian distribution assumption is not required (Corder and Foreman, 2014; Bethea, 2018).

2.1.3. Chi-square test

The goal of the test is same as the Kolmogorov-Smirnov test; this test can be applied to any univariate distribution if the cumulative distribution function existed. In our case, the test applied to the binned data which is calculated from the histogram of the experimental data. So the value that is obtained for the test depends on the number of binned data, and for this reason, sufficient data should be available in order to the chi-square test be approximately valid. This test can be applied to discrete distribution.

H_0: the data follow a specified distribution

H_a: the data do not follow the specific distribution

The test is performed on the data that are divided into k bins a, d the test is defined as **Error! Reference source not found.**):

$$\chi^2 = \sum_{i=1}^{k} (o_i - \mathrm{E}_i)^2 / \mathrm{E}_i \tag{3}$$

Where o_i is the observed frequency for bin i_{th} and $\mathrm{E}_i$is the expected frequency for bin i^{th} and the expected frequency is calculated from the interval between the lower and upper bound of i^{th} bin and n is number of data. As explained before the test is sensitive to number of bins and there is not any optimal value for the test output (Bethea, 2018).

3. Case studies

3.1. Synthetic data

In order to test the compatibility of using SciPy as a replacement of input analyzer synthetic data generated to performing the tests and comparing the results. Both input analyzer and SciPy have an ability to generating the random distribution data based on specific distribution. For our goal, the synthetic data generated using the input analyzer and for each specific distribution typical parameters assigned in order to generate the typical distribution shapes for each specific distribution types. After generating the data fitting and corresponding shape parameters obtained and the fitting tools forced to fit the corresponding model that is directly related to the data. The same process performed by SciPy package. On the next step, the shape parameters compared with those which obtained from an input analyzer. The results show the difference for lognormal distribution as the Arena software's conversion for mean and standard deviation is different from SciPy libraries. According to SciPy documentation, the parametrization of random variable X with mean μ and standard deviation of σ is expressed as a $expo(Y) = X$ then the S as a shape parameter is equal to σ and the scale is $scale =$ $\exp(\mu)$. On the other hand, in Arena input analyzer conversions are as **Error! Reference source not found.**) to **Error! Reference source not found.**):

$$LogMean = \mu \, , LogStd = \sigma \tag{4}$$

$$\mu = \ln\left(\frac{\mu^2}{\sqrt{(\sigma^2 + \mu^2}}\right) \tag{5}$$

and

$$\sigma = \sqrt{\frac{\ln(\sigma^2 + \mu^2)}{\mu^2}} \tag{6}$$

Solving the system of equations, the values obtained by SciPy can be converted to Arena formulation as **Error! Reference source not found.**):

$$\sigma = \sqrt{e^{2S} e^{2scale} (e^{s2} - 1)} \tag{7}$$

$$\mu = \sqrt{\mathrm{e}^{\mathrm{s}2} \, \mathrm{e}^{2\mathrm{scale}}} \tag{8}$$

3.2. Mining real dataset

Real dispatching data from an open pit mining system used as a database. Dispatching database contains a table for production, Activity and Shovel_In_Opereation and Trucks_In_Operation retrieved from the Production table which contains 1,468,959 number of rows and retrieving final necessary data from those tables. The logic for the database is illustrated in Fig 6.

Data and queries write for two different seasons (summer and winter) based on shovels and trucks group consist of six shovel types and six truck types. Number of passes for each truck and shovel, average bucket tonnage for trucks, average loading cycle for shovels, the total cycle time for trucks are the parameters which are retrieving from the database.

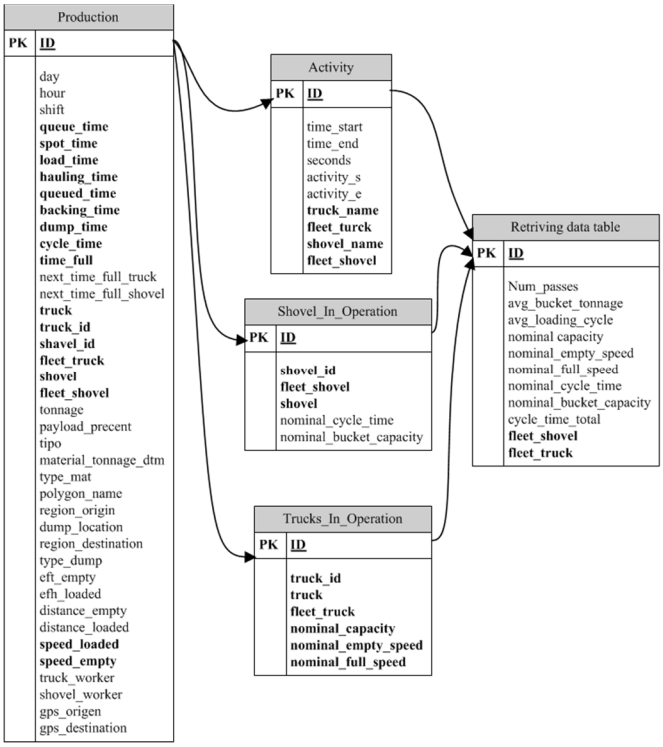

Fig 6 database diagram and relationships between each table for mining database

From the database and running related queries more than 270 series of data obtained that are showing the behaviour of shovel and trucks and their related parameters in two different seasons. The simulation model will recognize these behaviours based on a random distribution function. So it would be necessary to find the best-fitted distribution function model for the data. The process using Arena input analyzer is not time efficient, so data retrieving from database analyzed with the code that is provided by Python library, and final results write into CSV file. The fitted parameters acceptable as an input for Arena software are shown in Table 7 for discrete and continues variables with a related fitted distribution on Figs 7 and 8.

Table 7 Retrieved data and their fitted distribution with its parameters

Retrieved data	Best distribution	Expression	Least Square Error
Average_ Load_CycleTime _summerData_992GHL_CAT77D	gamma	-21.103 + GAMM(4.976,11.239)	0.000517865
Average_Load_CycleTime_winterData_Hit 5500EX_CAT 793D	norm	NORM(53.792,13.388)	0.000445379
backing_time_summerData_CAT 793D	lognorm	-11.024 + LOGN(24.36,6.468)	0.017501618
backing_time_winterData_CAT 793DT	lognorm	-9.064 + LOGN(24.701,7.607)	0.006585836
speed_Empty_winterData_CAT 793DT	norm	NORM(36.952,8.073)	0.000889691
speed_Empty_winterData_Cat 793B	norm	NORM(32.726,9.802)	0.000368787
SpotTime_summerData_CAT 785C_Hit 5500EX	lognorm	0.025 + LOGN(47.374,54.748)	0.000209112
SpotTime_summerData_CAT 793B_Hit 5500EX	lognorm	-0.269 + LOGN(64.171,77.815)	7.91092E-05
Avg_Bucket_Ton_summerData_Hit 5500EX_CAT 785C	lognorm	45.228 + LOGN(26.351,5.923)	0.00641712
Avg_Bucket_Ton_winterData_Hit 5500EX_CAT 793DT	norm	NORM(80.001,3.894)	0.085005238
NumPass_winterData_CAT 994F_CAT 793B	discrete cumulative	DISC(0.0,0,0.007,1,0.035,2,1.0,3)	
NumPass_summerData_Hit 2500_CAT 793D	discrete cumulative	DISC(0.005,0,0.075,1,1.0,2)	

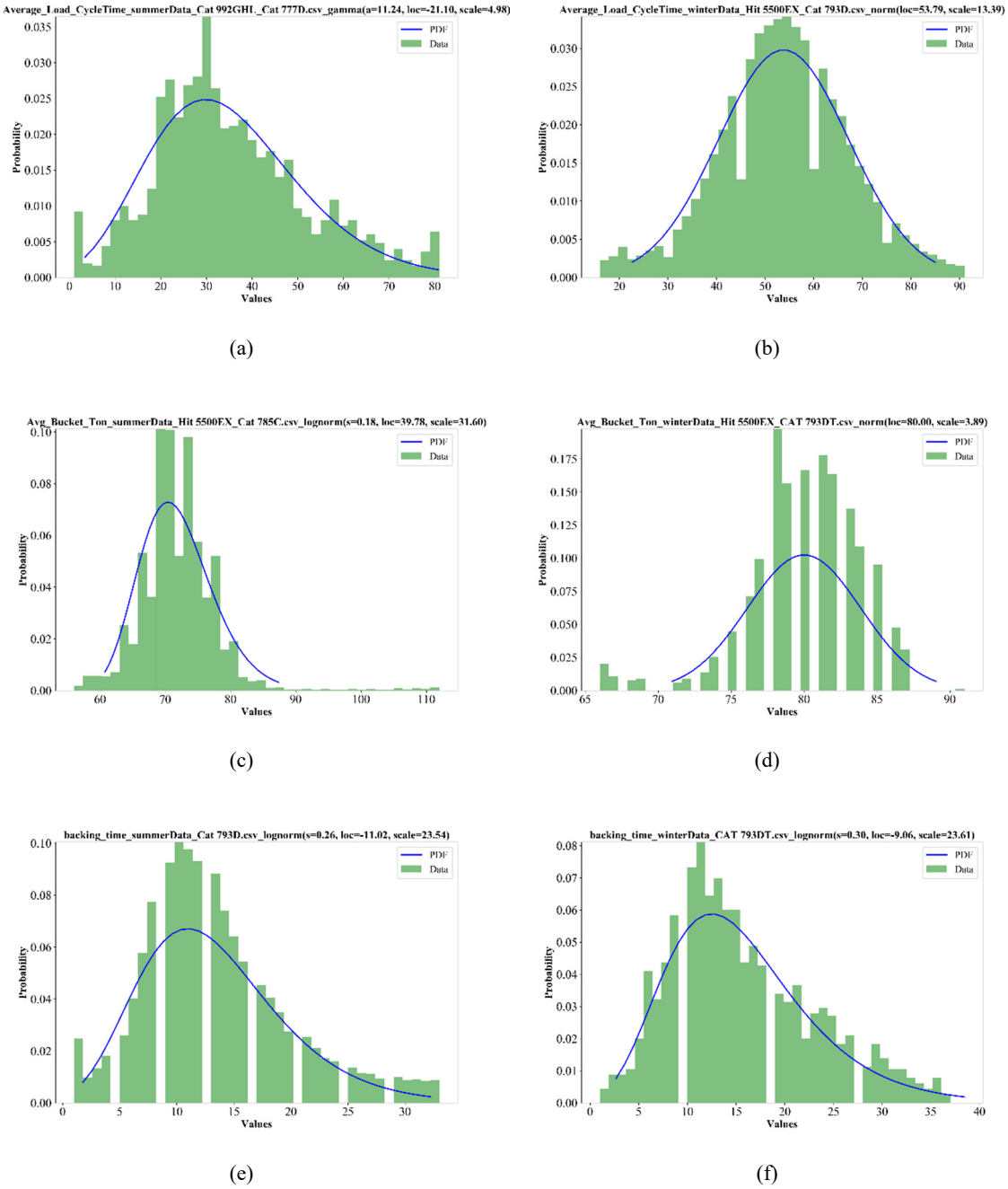

Fig 7 fitted distribution for average load cycle time for shovel CAT 992GHL and truck CAT 777D summer (a) and average load cycle time for winter data for shovel Hit 5500EX truck CAT 793D (b), average bucket tonnage for summer data shovel Hit 5500EX truck CAT 785C (c) and average bucket tonnage for winter data Hit 5500EX CAT 793DT (d), backing time for summer data truck CAT 793D (e) and backing time for winter data CAT 793DT (f).

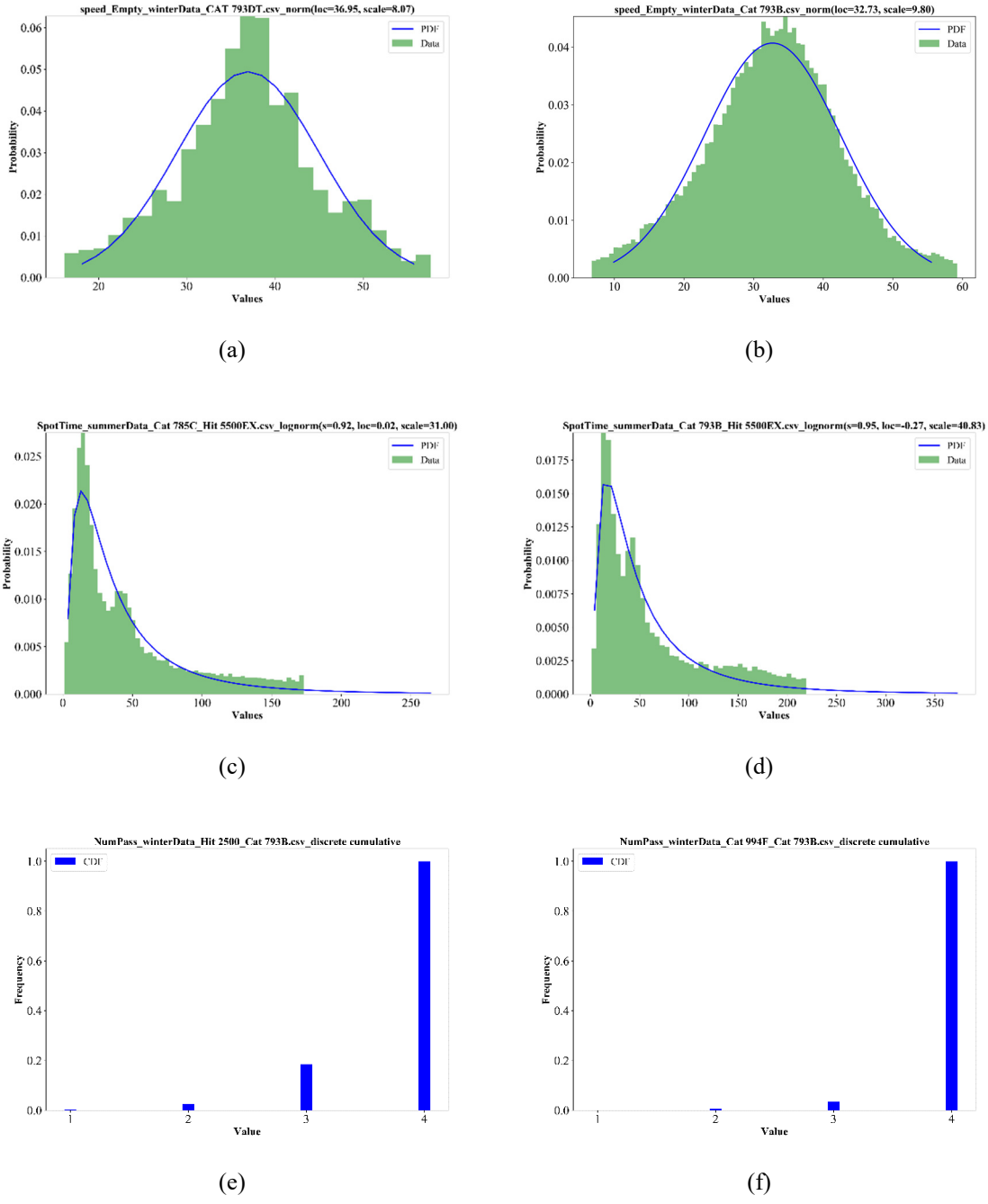

Fig 8 Fitted distribution for speed empty travel on winter for CAT 793DT (a) and for CAT 793B (b), spot time for summer data on shovel Hit 5500EX for truck CAT 785C (c) and truck CAT 793B (d), number of passes for winter data on shovel Hit 2500 truck CAT 793B (e) and shovel CAT 994F truck CAT 793B (f)

4. Conclusion

Open-source programming using Python programming language implemented for pre-processing analysis of data retrieving from dispatching database. The code provided has an ability to retrieving the data from the database and fit the distributions automatically to the data and generate the file consist of distribution expressions which are required for performing the simulation. The results show that the data manipulation such as removing outliers, duplicate values, bimodality, etc. can have a great effect on output results. Arena simulation software supports a restricted range of distributions and is not able to fit and use bimodal distributions model for the simulation process.

For this reason, after retrieving data from the database, the data should belong to the same population and check for the corresponding hypothesis. In case of not having a sufficient number of data the number of bins can change to the critical parameters as the current methods for automatically choosing the number of bins will be failed to obtain the optimum number of the bins and expert's ideas will be required and the process will be performed by trial and error for choosing the suitable number of bins. The advantage of the method, using Python versus Matlab programming language, is providing more coherent and faster process for pre-processing of the data with the lowest cost for time and money.

5. References

[1] Bethea, R. M. (2018). *Statistical methods for engineers and scientists.* Routledge,

[2] Ceruzzi, P. E. (2003). *A history of modern computing.* MIT press.

[3] Corder, G. W. and Foreman, D. I. (2014). *Nonparametric Statistics: A Step-by-Step Approach.* Wiley.

[4] Coronel, C. and Morris, S. (2016). *Database systems: design, implementation, & management.* Cengage Learning.

[5] Dagkakis, G. and Heavey, C. (2016). A review of open source discrete event simulation software for operations research. *Journal of Simulation*, *10* (3), 193-206.

[6] Feller, J. and Fitzgerald, B. (2002). *Understanding Open Source Software Development.* Addison-Wesley.

[7] Laurent, A. M. S. (2004). *Understanding Open Source and Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software.* O'Reilly Media.

[8] Liberman, M. (1995). Overreaching Provisions in Software License Agreements. *Richmond Journal of Law & Technology*, *1* (1), 4.

[9] RAO, M. N. (2014). *FUNDAMENTALS OF OPEN SOURCE SOFTWARE.* PHI Learning,

[10] Rockwell software (2018). Arena simulation. Rockwell software, Retrieved 2018 from: https://www.arenasimulation.com/

[11] St. Amant, K. (2007). *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives: Technological, Economic, and Social Perspectives.* Information Science Reference.