Analytic for Data-Driven Decision-Making in Complex High-Dimensional Time-to-Event Data

A Dissertation Presented
by
Keivan Sadeghzadeh
to
The Department of Mechanical and Industrial Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Industrial Engineering

Northeastern University
Boston, Massachusetts
Aug 2015
To the best wife, my love

*Niloo far*,

and to my shining star, my son

*Kian*.
Acknowledgments

I wish to thank everyone who made this dissertation possible.

I would like to extend my deepest appreciation to my research advisor, Professor Nasser Fard, for his technical and editorial advice which was essential for bringing this research to its current form. I am especially grateful for his excellent direction and the enthusiasm he expressed for my work.

This research greatly benefited from the useful suggestions and comments from the members of my exceptional doctoral committee, Professors Vinod Sahney and Tucker Marion. I wish to thank them for all their contributions.

I want to express my sincere gratitude to my beloved wife, Niloofar Montazeri, for providing me with motivation, understanding, support and encouragement at every step of this pursuit.

Many thanks go to Kian as well, the best child a father could have, for giving me the opportunity to get this work done. I am extremely grateful for having him and already proud of him.
Abstract

In the era of big data, analysis of complex and huge data expends time and money, may cause errors and misinterpretations. Consequently, inaccurate and erroneous reasoning could lead to poor inference and decision-making, sometimes irreversible and catastrophic events. On the other hand, proper management and utilization of valuable data could significantly increase knowledge and reduce cost by preventive actions. In many areas, there are great interests in time and causes of events. Time-to-event data analysis is a kernel of risk assessment and has an inevitable role in predicting the probability of many events occurrence. In addition, variable selection and classification procedures are an integral part of data analysis where the information revolution brings larger datasets with more variables and it has become more difficult to process the streaming high-dimensional time-to-event data in traditional application approaches, specifically in the occurrence of censored observations. Thus, in the presence of large-scale, massive and complex data, specifically in terms of variables, applying proper methods to efficiently simplify such data is desired. Most of the traditional variable selection methods involve computational algorithms in a class of non-deterministic polynomial-time hard (NP-hard) that makes these procedures infeasible. Although recent methods may operate faster, involve different estimation methods and assumptions, their applications are limited, their assumptions cause restrictions, their computational complexities are costly, or their robustness is not consistent.

This research is motivated by the importance of the applied variable reduction in complex high-dimensional time-to-event data to avoid aforementioned difficulties in decision-making and facilitate time-to-event data analysis. Quantitative statistical and computational methodologies using combinatorial heuristic algorithms for variable selection and classification are proposed. The purpose of these methodologies is to reduce the volume of the
explanatory variables and identify a set of most influential variables in such datasets.

In Chapter 1, an introduction to this research, problem definition and approach outline are provided through literature review.

In Chapter 2, comprehensive review of time-to-event data analysis dealing with censoring, survival and hazard function, and categories of this type of data is presented. This review is completed with an applied definition of decision-making process. Next, prominent data analysis tools and techniques is presented including discretization and randomization processes, and data reduction methods. Finally accelerated failure time model is introduced and defined.

Chapter 3 provides Methodology I: Nonparametric Re-sampling Methods for Kaplan-Meier Estimator Test. In this methodology, an analytical model for logical transformation of the explanatory variable dataset is proposed and a class of hybrid nonparametric variable selection methods and algorithms through variable inefficiency recognition are designed. The experimental results and comparison of them with well-known methods and simulation patterns are presented next.

Chapter 4 describes Methodology II: Heuristic Randomized Decision-Making Methods through Accelerated Failure Time Model. The methodology, including analytical model for the normal transformation of the explanatory variable dataset as well as heuristic randomized methods and algorithms are proposed. Finally, the simulation experiment, result and analysis of variable selection decision-making algorithms are presented.

In Chapter 5, Methodology III: Hybrid Clustering and Classification Methods using Normalized Weighted K-Mean is proposed. In this methodology, two different methods in order to cluster and classify variables are designed. Result and analysis of variable clustering and classification are included as
In Chapter 6, concluding remarks, a summary of the proposed methods and their advantages is presented. In addition, an overview for further studies will be discussed.

The computer package used in this research is the MATLAB® R2011b programming environment.
# Contents

List of Figures ................................. 6

List of Tables ................................. 8

1 Introduction ................................. 10
   1.1 Overview .................................. 10
   1.2 Literature Review .......................... 12
   1.3 Problem Definition and Solution Approach ... 15
   1.4 Big Data Analytics .......................... 17
   1.5 Specialty and Application ................. 20
      1.5.1 Volume ................................ 21
      1.5.2 Variety ................................ 22
      1.5.3 Velocity ................................. 23
      1.5.4 Veracity ................................ 24
      1.5.5 Variability ............................... 24
      1.5.6 Visualization ............................. 25
      1.5.7 Value .................................. 25

2 Preliminaries, Concepts and Definitions .... 27
   2.1 Overview .................................. 27
   2.2 Time-to-Event Data Analysis ................ 28
      2.2.1 Censoring ............................... 28
      2.2.2 Survival and Hazard Function .......... 30
      2.2.3 Categories of Time-to-Event Data ...... 33
      2.2.4 Kaplan–Meier Estimator ................. 34
2.3 Accelerated Failure Time Model ............................................. 35
2.4 Decision-Making Process .................................................... 38
2.5 Data Analysis Tools and Techniques ................................. 40
  2.5.1 Discretization Process ................................................. 40
  2.5.2 Randomization Process ................................................. 42
  2.5.3 Data Reduction Methods .............................................. 43
  2.5.4 Feature Extraction and Feature Selection ......................... 45
  2.5.5 Principal Component Analysis ...................................... 46

3 Methodology I: Nonparametric Re-sampling Methods for Kaplan–Meier Estimator Test ................................................................. 51
  3.1 Overview ................................................................. 51
  3.2 Analytical Model I ...................................................... 52
    3.2.1 Model Validation .................................................. 55
  3.3 Methods and Algorithms for Variable Selection .................. 59
    3.3.1 Singular Variable Effect ......................................... 61
    3.3.2 Nonparametric Test Score ....................................... 62
    3.3.3 Splitting Semi-Greedy Clustering ............................. 65
    3.3.4 Weighted Time Score ............................................. 67
  3.4 Experimental Results and Analysis for Right-Censored Data ... 68
  3.5 Experimental Results and Analysis for Uncensored Data ...... 75

4 Methodology II: Heuristic Randomized Decision-Making Methods through Weighted Accelerated Failure Time Model ............................................. 80
  4.1 Overview ................................................................. 80
  4.2 Analytical Model II ...................................................... 81
  4.3 Methods and Algorithms for Variable Classification ............ 84
    4.3.1 Ranking Classification .......................................... 87
    4.3.2 Randomized Resampling Regression ......................... 89
  4.4 Simulation Experiment, Results and Analysis for RC Method .. 91
    4.4.1 Simulation Design and Experiment ............................ 91
    4.4.2 Result and Analysis ............................................. 93
  4.5 Simulation Experiment, Results and Analysis for 3R Method .. 95
## List of Figures

1.1 Big Data Analytics Options Plotted by Potential Growth and Commitment [45]  .................................................. 19

1.2 Seven dimensions of large-scale data. .................................. 21

1.3 Volume reduction flow diagram ($n' < n$ and $p' < p$). .............. 22

1.4 Binarization and Normalization through transformation models. 23

1.5 Sample visualization. ...................................................... 26

2.1 Probability density, failure, survival, and hazard functions. ....... 33

2.2 Kaplan-Meier estimate. .................................................... 35

2.3 Data mining categories. .................................................... 44

2.4 Schema of PCA transformation. ....................................... 48

3.1 Comparison of covariate correlations in the original and the transformed dataset. .................................................. 56

3.2 Empirical survival function (solid blue), Cox PHM baseline survival function for the original dataset (black dashed), and Cox PHM baseline survival function for transformed logical dataset (red dotted). .................................................. 58

3.3 Scree plot of the original PBC dataset including 17 variables and 276 observations. .................................................... 70

3.4 Empirical transformed logical uncensored PBC dataset survival function (solid red) with 99% confidence bounds (red dotted). Subset of $k$ variable with failed to reject Wilcoxon rank sum test result (gray), and subset of $k$ variable with rejected Wilcoxon rank sum test result (black). .................................................. 72
List of Tables

3.1 Schema for high-dimensional time-to-event dataset. ............ 53
3.2 Modified k-means clustering algorithm result for the transformed logical uncensored PBC dataset. Number of clusters is equal to $\| \frac{\mathbf{x}}{\mathbf{k}} \|$. .................................................. 71
3.3 Selected less efficient variables in all proposed methods and comparison to RSF, ADD, and LS method results. ............. 73
3.4 Selected less efficient variables in all proposed methods and comparison to simulation defined pattern. ......................... 73
3.5 Performance of the proposed methods based on six simulation numerical experiment. $m$ is integer average number of selected inefficient variable and $p$ is performance of method based on 100 replications. .................................................. 75
3.6 Selected inefficient variables in all proposed methods and comparison to NTS, RSF, ADD, and LS method results. ........... 77
3.7 Selected inefficient variables in all proposed methods and comparison to NTS results and simulation defined pattern. ........ 79
4.1 Performance of the proposed methods based on four numerical simulation experiments with 200 replications. ............. 94
4.2 Performance of the proposed methods based on three numerical simulation experiments with 200 replications. ............. 99
5.1 High-dimensional time-to-event dataset ............................. 102
5.2 Sample of time-based binning based on three techniques. ... 109
5.3 Clustering by cost function ........................................... 111
Chapter 1

Introduction

1.1 Overview

Advancement in technology has led to accessibility of massive and complex data in many fields. Proper management and utilization of valuable data could significantly increase knowledge and reduce cost by preventive actions, whereas erroneous and misinterpreted data could lead to poor inference and decision-making. As an analytical approach, decision-making is the process of finding the best option from all feasible alternatives. The application of decision-making process in economics, management, psychology, mathematics, statistics and engineering is obvious and this process is an important part of many science-based professions.
In many areas, there are great interests in time and causes of events. Birth in demography and death in medical sciences, hospitalization in sociology and arrest in criminology, promotion in management and bankruptcy in business, revolution in political science and divorce in psychology, eclipse in astronomy and failure in engineering science, metamorphosis in biology and earthquake in geology, all are samples of an event which can occur at specific time points which make change in some quantitative variable. The information of such observations including event time, censoring indicator, and explanatory variables which are collected over a period of time is called time-to-event data. This type of data is the outcome of many scientific investigations, experiments and surveys. In the field of data and decision analysis, it has become more difficult to process the streaming high-dimensional time-to-event data in traditional application approaches, specifically in the presence of censored observations.

Variable selection is a necessary step in a decision-making process dealing with a large-scale data. There is always uncertainty when researchers aim to collect most important variables specifically in the presence of big data. Variable selection for decision-making in many fields is mostly guided by expert opinion [10]. The computational complexity of all the possible combinations of the $p$ variables from size 1 to $p$, could be overwhelming, where the total number of combinations are $2^p - 1$. For example, for a dataset of 20 explana-
tory variables, the number all possible combinations is $2^{20} - 1 = 1048575$.

This research presents a class of multipurpose practical methods to analyze complex high-dimensional time-to-event data to reduce redundant information and facilitate practical interpretation through variable efficiency and inefficiency recognition. In addition, numerical experiments and simulations are developed to investigate the performance and validation of the proposed methods.

1.2 Literature Review

Analytics data-driven decision-making can substantially improve management decision-making process. This process is increasingly based on the type and size of data, as well as analytic methods. It has been suggested that new methods to collect, use and interpret data should be developed to increase the performance of the decision makers [7, 35]. Data collection and analysis in a data-driven decision-making plays crucial roles. In many cases data are collected from different sources such as financial reports and marketing data and they are combined for more informative decision-making. Consequently, determining effective explanatory variables, specifically in complex and high-dimensional data provides an excellent opportunity to increase efficiency and reduce costs.
Time-to-event data such as failure or survival times have been extensively studied in engineering, economics, business and medical science. In addition, by the advent of modern data collecting technologies, a huge amount of this type of data includes high-dimensional covariates. This massive amount of data are increasingly accessible from various sources such as transaction-based information, information-sensing devices, remote sensing technologies, machines and logistics statistics, wireless sensor networks as well as quality analytics in engineering, manufacturing, service operations and many other segments. Unlike traditional datasets with few explanatory variables, analysis of datasets with high number of variables requires different approaches. In this situation, variable selection techniques could be used to determine a subset of variables that are significantly more valuable to analyze high-dimensional time-to-event datasets. If a data is compiled and processed correctly, it can enable informed decision-makings [6, 13, 21, 37, 41, 54, 60].

The opportunity for manufacturing and services in the era of data is to analyze their performance to enhance the quality. Quality dimensions of products and services, even defined by quality experts [16, 58] or perceived by customers [4] is summarized as performance, availability, reliability, maintainability, durability, serviceability, conformance, warranty, and price as well as aesthetics and reputation. Access to valuable data for sophisticated analytics can substantially improve management decision-making process. In the field of reliability, analyzing the collected data from different sources
such as customer reports as well as testing laboratories, and consequently determining effective features and variables in the failure time, specifically in complex, high-dimensional and censored time-to-event data provide an excellent chance for manufacturers to reduce costs, improve efficiency and ultimately improve the quality of their products by detecting failure causes faster [37, 43].

Many management decision-making process involves complicated problems, including multi-criteria and multi-variable, as well as risk and uncertainty that require mathematical and statistical techniques, data analysis tools, combinatorial algorithms, and quantitative computational methods. As an analytical approach, decision-making is the process of finding the best option from all feasible alternatives [10, 51] and advanced analytics are among the most popular techniques used in massive data analysis and decision-making process [45]. In many professional areas and activities, decision-making is increasingly based on the type and size of data, rather than on experience and intuition [7, 35]. In a decision-making process, variable selection and variable reduction are necessary procedures in dealing with a large-scale data. There is always uncertainty when one aims to collect most important variables, specifically in the presence of big data.
1.3 Problem Definition and Solution Approach

Variable selection procedures are an integral part of data analysis. The information revolution brings larger datasets with more variables. The demand for variable selection as a fundamental strategy for data analysis is increasing. Although a wide variety of variable selection methods have been proposed, there is still plenty of work to be done. Many of the recommended procedures have given only a narrow theoretical motivation, and their operational properties need more systematic investigation before they can be used with confidence. The problem of variable selection and subset selection in large-scale datasets arises when it is desired to model the relationship between a response variable and a subset of explanatory variables but there is uncertainty about which subset to use.

Most of the traditional variable selection methods such as Akaike information criterion (AIC) [3] or Bayesian information criterion (BIC) [53] involve computational algorithms in a class of non-deterministic polynomial-time hard (NP-hard) that makes these procedures infeasible. Although recent methods may operate faster, involve different estimation methods and assumptions, such as Cox proportional hazard model [11], accelerated failure time [31], Buckley-James estimator [8], random survival forest [27], additive risk models [36], weighted least squares [24] and classification and regression tree (CART) [5], their applications are limited, their assumptions cause restrictions, their computational complexity are costly, or their robustness are not
consistent.

Since the results obtained from NP-hard problems are not robust, then heuristic approaches are the only viable options for a variety of complex optimization problems such as variable selection in complex high-dimensional time-to-event data which need to be solved in real-world applications. The criteria for deciding whether to use a heuristic approach for solving a given problem include optimality, completeness, accuracy, and execution time which should be evaluated by experts. In mathematical optimization, a heuristic is a technique is designed for a quick solution for a problem when classical methods are slow, or for finding an approximate solution when classical methods fail to find any robust solution. As a shortcut, this is achieved by trading optimality, completeness, accuracy, or precision for speed. The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving a problem. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution, but it does not require a prohibitively long time.

This study is motivated by the importance of above-mentioned variable selection issue. The objective of this study is to design and propose combinational procedures and methodologies including nonparametric and heuristic methods for variable selection, classification and reduction in complex high-dimensional time-to-event data through determining variable efficiency and
inefficiency. The purpose of proposed analytic models and heuristic methods is to reduce the volume of the explanatory variables and identify a set of most influential variables on the time-to-event.

Proposed methodologies in this dissertation are (1) Nonparametric Re-sampling Methods for Kaplan-Meier Estimator Test, including a class of hybrid non-parametric variable selection methods, (2) Heuristic Randomized Decision-Making Methods through Weighted Accelerated Failure Time Model for variable classification and reduction, and (3) Hybrid Clustering and Classification Methods using Normalized Weighted K-Mean which is designed to cluster and classify high number of variables. The achievements are to reduce limitations in applications, relax restrictions caused by assumptions, decrease the cost of computational procedure, and increase the accuracy and robustness.

1.4 Big Data Analytics

There are more than one trillion connected devices in the world and over 15 petabytes (15 × 10^{15} bytes) of new data are generated each day [25]. As of 2012, overall created data in a single day were approximated at 2.5 etabytes (2.5 × 10^{18} bytes) [26, 39].

The size and dimension of data sets expand because they are increasingly being gathered by information-sensing mobile devices, remote sensing tech-
Technologies, genome sequencing, cameras, microphones, RFIDs, wireless sensor networks, internet search as well as finance logs [13, 21, 54]. The world’s technological per-capita capacity to store information has approximately doubled every 40 months [22, 39] while annually worldwide information volume is growing at a minimum rate of 59 percent [15].

In 2012 total software, social media, and IT services spending related to big data and analytics reported over $28 billion worldwide by the IT research firm, Gartner, Inc. This amount was forecast to drive $34 billion in 2013. Also it has been predicted that IT organizations will spend $232 billion on hardware, software and services related to Big Data through 2016. In business, economics and other fields, decisions will increasingly be based on data and analysis rather than on experience and intuition. In recent years, researchers suggest that businesses should discover new ways to collect and use data every day, develop the ability to interpret data to increase the performance of their decision makers, where data-driven decision-making (DDDM) methods emerge. According to MIT research in 2011, among 330 large publicly-traded companies in the study, those adopting this method achieved productivity gains 5 to 6 percent higher than others [7, 35].

As a vast survey published in 2011, based on responses from organizations for techniques and tool types using in big data analysis, advanced analytics (e.g., mining, predictive) and data visualization are among the most popular
Figure 1.1: Big Data Analytics Options Plotted by Potential Growth and Commitment [45].
ones in terms of potential growth and commitment [45]. See Fig. 1.1. Accordingly, the strongest commitment among options for big data analytics is to advanced analytics where closely related options such as predictive analytics, data mining, and statistical analysis have a similar commitment. On the other hand, the strongest potential growth among options for big data analytics is projected for advanced data visualization (ADV).

1.5 Specialty and Application

The application of the designed methodologies in this research is focused on reliability and failure in manufacturing and business presenting applied heuristic procedures for variable subset selection and classification in high-dimensional, large-scale and complex time-to-event dataset. In this dissertation, a class of applied data reduction and appropriate variable selection and classification in such dataset is proposed which enables to avoid difficulties in decision-making and facilitate time-to-event and failure analysis. To demonstrate the significance and advantages of proposed mythologies in this study over previous ones, seven dimensions of large-scale data are considered as Fig. 1.2.
1.5.1 Volume

The vast amount of data generated every year. This makes most datasets too large to store and analyze using traditional technologies. High-dimensional time-to-event data is a special dataset including event time and explanatory variables which needs innovative methods to be analyzed when the number of variables is more than traditional problems. In comparison to other methods, designed algorithms decrease the size of dataset in terms of explanatory variables and observations in first stage of analysis using clustering concept as a preliminary process to volume reduction, shown in Fig. 1.3.

Figure 1.2: Seven dimensions of large-scale data.
Figure 1.3: Volume reduction flow diagram ($n' < n$ and $p' < p$).

1.5.2 Variety

Years ago, all data that was created was structured data fitted in columns and rows. Data today comes in many different complex formats and most of the data that is generated is unstructured data. The wide variety of data requires a different approach as well as different techniques to analyze where each of different types of data require different types of analyses or different tools to use. Proposed models in this study regulate and integrate complex formats of explanatory variables as an unstructured time-to-event datasets including different types of quantitative and qualitative data mixed with censored and missed values, and transformed this set into a simple and easy-to-analyze dataset using binarization and normalization as Fig. 1.4:

The advantages of transformation models in this study are:

1. Transformed data is understood and interpreted easily.

2. Enable experts to define explanatory variables in a simple format be-
3. In comparison to other relevant techniques, transformed data is processed and analyzed faster and cheaper in terms of calculation.

4. The accuracy of mathematical calculation in data mining algorithms in next stages is higher based on covariance reduction.

1.5.3  Velocity

The velocity or speed refers to how fast the data is generated, stored, analyzed and visualized. Data processors required time to process the data and update the databases. In the era of big data era, real-time data is generated commonly which is a challenge. The proposed methods and algorithms
analyze and process high-dimensional time-to-event datasets faster than traditional method to obtain variable selection results.

1.5.4 Veracity

Incorrect data can cause a lot of problems. Data users need to be ensured about the accuracy of the data and the performance of the data analyses. One of the biggest problems with big data is the tendency for errors. User entry errors, redundancy and corruption all affect the value of data. Accountability and trust play major role in data science specifically in big data problems. The veracity is defined in three domains; Source, nature, and process of data.

In other words, veracity is the quality and understandability of the data. Veracity isn’t just about data quality, it’s about data understandability. An advantage of proposed methods is understandability where these methods are potentially become tools which is user-friendly end-user interface.

1.5.5 Variability

There are four commonly used measures of variability: range, mean, variance and standard deviation. In order to perform a proper variability analyses, algorithms need to be able to understand the context and be able to decipher the exact meaning of data. Transformation models which designed for early
stage of data analysis in this study play this role perfectly and make the big dataset of time-to-event accessible for any variability analysis which has not done before in this fashion.

1.5.6 Visualization

Visualization refers to making the vast amount of data comprehensible in a manner that is easy to understand and interpreted. Raw data can be put to use with the right visualizations. The best tools for visualization are complex graphs that can include many variables of data while still remaining understandable. The designed output of each part of all methodologies in this study is an understandable hybrid graph where telling a complex data analysis in a graph is very difficult but also extremely crucial. As a sample, Fig. 1.5 depicts the visualization process:

1.5.7 Value

According to McKinsey, potential annual value of large-scale data to the US Health Care is $300 billion. All available data will create a lot of value but data in itself is not valuable at all. The value is in the analyses and in how the data is turned into information and eventually turning it into knowledge. The value relies on insights derived from data analyses for decision-making. Proposed analytics in data-driven decision-making in complex high-dimensional
time-to-event data enable users to extract valuable information from such data through a class of novel methodologies.
Chapter 2

Preliminaries, Concepts and Definitions

2.1 Overview

An introduction to time-to-event data analysis, decision-making process, as well as a review of prominent data analysis tools and techniques, and accelerated failure time model are presented in this chapter.
2.2 Time-to-Event Data Analysis

Time-to-event data analysis methods consider the time until the occurrence of an event. This time can be measured in days, weeks, years, etc. This analysis is used widely in engineering, sometimes is called life data or failure time analysis since the main focus is in modeling the time it takes for a components to fail. In social sciences as economics, event history analysis is the used alternative where is also known as survival analysis in medical sciences.

In time-to-event data, subjects are usually followed over a specified time period. Study of time-to-event data focuses on predicting the probability of survival or failure. Examples of time-to-event analysis data are the lifetime of mechanic devices, electronic components, or complex systems as well as insurance compensation claims, worker’s promotions, and company bank ruptcies [31,33,59].

2.2.1 Censoring

Generally, data censoring occurs when some information available for a variable but is not complete and analyzing a censored variable requires specific procedures. In time-to-event data analysis there are usually some observations with no experience of the event during the study and the time-to-event is incomplete for them. It is known that if the event happen to these cases, the time of the event will be greater than the length time of the study/observer.
vation. In addition sometimes censoring occurs when there is no information about over demand for a limited product or service which is discontinued. Another censoring might happen due to lack of refinement in the measurement using experimental or industrial equipment.

One approach to deal with censored data is to set the censored observations to missing or replace the unobserved value by mean value, minimum, maximum, or a randomly assigned value from the range when the number of censoring observation is negligible otherwise these solutions can cause serious bias in estimates and discard potentially important information. There are two categories of right censoring:

1- Singly censored data, which is included (a) time termination where study/observe is limited to a fixed period of time due to some restrictions such as time and cost, and (b) failure termination which study/observe continue until a fixed portion of failures. In time termination, Survival time recorded for the failed subject is the times from the start of the experiment to its failure which is called exact or uncensored observation. The survival time of any remain subject is not known exactly but is recorded as at least the length of the study period and called censored observation. In type I censoring, if there are no accidental losses, all censored observations equal the length of the study period. In failure termination each censored observation is equal to the largest uncensored observation.
2- Progressively censored data, also called random censoring, which is the period of study/observe is fixed and subjects enter the study at different times during that period.

Some aspects such censoring and non-linearity create difficulty in analyzing the data by traditional statistical models. Regression models cannot effectively perform in the presence of the censored observations [32, 33]. The right censoring is most commonly seen in time-to-event data. When an experiment is censored from right, the subjects do not undergo full duration of an experimental or study time. This could be due to many reasons, such as a termination of the study, or incidents. Analyzing right-censored high-dimensional time-to-event data to find appropriate model or distribution is time consuming, not economical, and most likely no theoretical distribution adequately extracted. In this situation, nonparametric methods are considered to apply.

2.2.2 Survival and Hazard Function

By definition, the probability of an event occurring at time $t$ is

$$f(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t)}{\Delta t}$$  \hspace{1cm} (2.1)

In time-to-event or survival analysis, information on an event status and
follow up time is used to estimate a survival function $S(t)$, which is defined as the probability that an object survives at least until time $t$:

$$S(t) = P(\text{an object survives longer than } t) = P(T > t) \quad (2.2)$$

From the definition of the cumulative distribution function (or failure function):

$$S(t) = 1 - P(\text{an object survives longer than } t)$$
$$= 1 - P(T \leq t) = 1 - F(t) \quad (2.3)$$

Accordingly survival function is calculated by probability density function as:

$$S(t) = \int_t^\infty f(u)\,du \quad (2.4)$$

The survival function has the following properties: $S(t = 0) = 0$, $S(t) \to 0$ as $t \to 0$, and $S(u) \leq S(t)$ for $u \geq t$. In most of the applications, the survival function is shown as a step function rather than a smooth curve.

Hazard function defines as the probability that if subject survive to time $t$, and will succumb to the event in the next instant, as below:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \quad (2.5)$$

All four functions, probability density $f(t)$, cumulative distribution function or failure function $F(t)$, survivorship or survival function $S(t)$ and hazard
function $h(t)$, mathematically have relations:

1- Hazard function from probability density survival functions:

$$ h(t) = \frac{f(t)}{S(t)} \quad (2.6) $$

2- Probability density function from Survival function:

$$ f(t) = -\frac{dS(t)}{dt} \quad (2.7) $$

3- Probability density function from Survival function:

$$ f(t) = h(t)e^{\left(-\int_0^t h(u)du\right)} \quad (2.8) $$

4- Survival function from hazard function:

$$ S(t) = e^{\left(-\int_0^t h(u)du\right)} \quad (2.9) $$

5- Survival function from hazard function:

$$ h(t) = -\frac{d}{dt}\log S(t) \quad (2.10) $$

Fig. 2.1 simply illustrated four aforementioned functions:
2.2.3 Categories of Time-to-Event Data

Methods to analyze time-to-event data are parametric; which are based on survival function distributions such as exponential, semi-parametric; which do not assume knowledge of absolute risk and estimates relative risk, and non-parametric; which are useful when the underlying distribution of the problem is unknown. For moderate- to high-dimensional covariates, it is difficult to apply parametric or semi-parametric methods [24]. Nonparametric methods are used to describe survivorship in a population or for comparison of two or more populations.

Figure 2.1: Probability density, failure, survival, and hazard functions.
2.2.4 Kaplan–Meier Estimator

The Kaplan–Meier (KM) is the most commonly used nonparametric method for the survival function and has clear advantages since it does not require an approximation that results the division of follow-up time assumption [23, 33]. Nonparametric estimate of $S(t)$ according to this estimator for distinct ordered event times $t_1$ to $t_n$ is:

$$\hat{S}(t) = \prod_{i=1}^{t} (1 - \frac{d_i}{n_i})$$ (2.11)

Where at each event time $t_j$ there are $n_j$ subjects at risk and $d_j$ is the number of subjects which experienced the event. Let $c_i$ denote the number of subjects censored between $t_i$ and $t_{i+1}$. Then the likelihood function takes the form:

$$L = \prod_{i=1}^{t} ([S(t_{i-1}) - S(t_i)]^{d_i}[S(t_i)]^{c_i})$$ (2.12)

For the conditional probability of surviving, if we define $\pi_i = \frac{S(t_i)}{S(t_{i-1})}$, then the maximum-likelihood estimation of $\pi_i$ is:

$$\hat{\pi}_i = 1 - \frac{d_i}{n_i}$$ (2.13)

Graphically, the Kaplan–Meier estimate is a step function with discontinuities or jumps at the observed failure times as shown in Fig. 2.2. It has been shown [44] that the K–M estimator is consistent. Completely nonparametric nature of this estimator assures little or no loss in efficiency using it in practice.
Accelerated failure time model is presented next.

### 2.3 Accelerated Failure Time Model

In time-to-event data analysis, when distribution of the event time as a response variable is unknown, especially in presence of censored observations,
estimations and inferences become challenging and the traditional statistical methods are not practical. Cox proportional hazards model (PHM) [11] and the accelerated failure time (AFT) [31] are frequently used in these circumstances. For a comprehensive comparison see [40]. Similar to a multiple linear regression model, AFT assumes a direct relation between the logarithm of the time-to-event and the explanatory variables.

As defined in Section 2.2, the event time may not always be observable. This is known as censored data. In this situation the response variable is defined as:

$$ T_i = \min(Y_i, C_i), \quad \delta_i = \begin{cases} 1, & Y_i \leq C_i \\ 0, & Y_i > C_i \end{cases} \quad i = 1 \ldots n $$

(2.14)

The censoring time $C$ is independent of the event time $Y$. Time-independent explanatory variable $X_j, j = 1 \ldots p$ corresponding ith observation effects multiplicatively on the time-to-event $Y_i$ or additively on $T_i$ where the relation between the explanatory variables and the response variable is considered linear. The AFT model generally is written as:

$$ S(t|x) = S_0\left(\frac{t}{\psi(x)}\right)S_0(.) $$

(2.15)

is the baseline survival function and $\psi(x)$ is an acceleration factor defined as follows:
$$\psi(x) = e^{\beta X}$$  \hspace{1cm} (2.16)

The coefficient vector $\beta$ is of length-$p$. The corresponding log-linear form of the AFT model with respect to time is given by

$$\ln (T) = X\beta + \epsilon$$  \hspace{1cm} (2.17)

Nonparametric maximum likelihood estimation does not work properly in the AFT model [62], where in many studies least squares method has been used to estimate parameters. To account for censoring, weighted least squares method is commonly used, as ordinary least squares does not work for censored data in the AFT model. According to Stute [55] to obtain estimators using weighted least squares, a general hypotheses is under the assumption that vector $\epsilon$ consists of random disturbances with zero mean so that $E [\epsilon|X] = 0$. In many studies the error distribution is unspecified [29]. Also for this vector the assumption of homoscedasticity is supported which assumes a constant variance for the errors. In addition, regardless of the values of the coefficients and explanatory variables, the log form assures that the predicted values of $T$ are positive.

Next, a review of decision-making process is presented.
2.4 Decision-Making Process

Decision-making is a process of making choices by setting objectives, gathering information, and assessing alternative choices. This process includes seven steps:

1. Identifying a problem or an opportunity
2. Gathering relevant information
3. Analyzing the problem and information
4. Establishing several possible options and alternative solutions
5. Evaluating alternatives for feasibility, acceptability and desirability.
6. Selecting a preferred alternative for future possible adverse consequences.
7. Implementing and evaluating the solution

Decision-making theories are classified based on two attributes: (a) Deterministic, which deals with a logical preference relation for any given action or Probabilistic, which postulate a probability function instead, and (b) Static, which assume the preference relation or probability function as time-independent or Dynamic which assume time-dependent events [9]. Historically, the Deterministic-Static decision-making is more popular decision-making process specifically under uncertainty. The assumption of decision-making in this study falls in this category as well.
A major part of decision-making involves the analysis of a finite set of alternatives described in terms of evaluative criteria. The mathematical techniques of decision-making are among the most valuable factors of this process, which are generally referred to as realization in the quantitative methods of decision-making [51]. With the increasing complexity and the variety of decision-making problems due to the huge size of data, the process of decision-making becomes more valuable. In most decision-making problems, the multiplicity of criteria for evaluating the alternatives is pervasive [7, 51].

The application of decision-making process in economics, psychology, management, mathematics, statistics and engineering is obvious and this process is an important part of all science-based professions. In this study, a decision-making problem is assumed to be (1) deterministic, which deals with a logical preference relation for any given action and (2) static, which consider the preference relation or probability function as time-independent. A brief review of time-to-event data analysis and survival function is following.

The performance of the decision-making process may be affected by a set of different factors includes data size, the number of variables, and the presence of special cases such as censored or missing data.

A review of commonly used data analysis tools and techniques in this study is following.
2.5 Data Analysis Tools and Techniques

Discretization process, randomization process, and data reduction methods are reviewed briefly.

2.5.1 Discretization Process

Variables in a dataset potentially are a combination format of different data types such as:

- **Dichotomous (Binary)**; that occur in one of two possible states, often labelled zero and one. E.g., improved: yes/no or failed: yes/no.

- **Nominal**; that the values can be assigned a code in the form of a labels number which is countable but not be ordered or measured. E.g., male and female (coded in order as 0 and 1)

- **Ordinal**; that can be ranked or rated which could be even counted or ordered but not measured. E.g., anxiety scale: none, mild, moderate, and severe, with numerical values of 0, 1, 2, 3.

- **Categorical**; that the value indicates membership in one of several possible non-overlapping categories. E.g., color: black, brown, red, etc. If the categories are assigned numerical values used as labels then it will be synonym for nominal.
• **Discrete**: that have only integer values. E.g., the number of a specific component in warehouse.

• **Continuous (Interval)**: that is not restricted to particular values and may take any value within a finite or infinite interval. E.g., days of hospital stay or human reaction or temperature.

There are many advantages of using discrete values over continuous as discrete variables are easy to understand and utilize, more compact and more accurate. Quantizing continuous variables is called discretization process.

In the splitting discretization methods, continuous ranges are divided into sub-ranges by the user specified width considering range of values or frequency of the observation values in each interval, respectively called equal-width and equal-frequency. A typical algorithm for splitting discretization process which quantifies one continuous feature at a time generally consists of four steps: (1) sort the feature values, (2) evaluate an appropriate cut-point, (3) split the range of continuous values according to the cut-point, and (4) stop when a stopping criterion satisfies.

In this study, discretization of explanatory variables of time-to-event dataset assumed unsupervised, static, global and direct in order to reach a top-down splitting approach and transformation of all types of variables in dataset into a logical (binary) format. Briefly, static discretization is dependent of
classification task, global discretization uses the entire observation space to discretize, and direct methods divide the range of k intervals simultaneously. For a comprehensive study of discretization process, see [34].

2.5.2 Randomization Process

Randomization is the process to select or assign subjects with the same chance. Many procedures and methods have been proposed for the random assignment such as simple, replacement, block, stratified, and covariate adaptive randomization. To select an appropriate method to produce interpretable and valid results in specific study, knowledge of advantages and disadvantages of each method is necessary. In this study we briefly review simple and block randomization [56].

Simple randomization keeps complete randomness of the assignment of a subject to a particular group based on a single sequence of random assignments. This randomization approach is simple and easy to implement. Simple randomization may be conducted with or without replacement. In simple replacement randomization, some selection criteria is beneficial to use such as subject stopping criteria which guarantees that one subject could not be selected more than a specific number.

Block randomization is designed to randomize subjects into equal sample
sizes groups. This method is used to ensure a balance in sample size across groups over time. Blocks are balanced with predetermined group assignment, which keeps the numbers of subjects in each group similar where the block size is determined by the researcher as well. Sometimes it is desired to assign a group (cluster) of subjects together in a cluster randomized trial. The procedures for block randomizing clusters are identical to those described above.

2.5.3 Data Reduction Methods

Knowledge Discovery in Database (KDD) is defined as a process of acquiring knowledge from raw data by selection, cleaning, processing, transformation, data mining and evaluation. Data mining as an analysis step of KDD is the computational process of discovering patterns in large datasets and covers the classes of data selection, learning, clustering, classification, prediction and summarizing. Categories of data mining are shown in Fig. 2.3.

These categories are grouped in two general analysis set: Descriptive and Predictive. Clustering is the process of organizing objects into groups with similar members according to one or more criterion. This process is a descriptive analysis stage of data mining which deals with finding a structure in a collection of unlabeled data. Therefore, a cluster is a collection of data objects that are similar to one another. Cluster analysis has many applications which can be used in the data mining process as a preprocessing step.
Data reduction techniques are categorized in three main strategies, including dimensionality reduction, numerosity reduction, and data compression [56, 57].

Dimensionality reduction as the most efficient strategy in the field of large-scale data deals with reducing the number of random variables or attributes in the special circumstances of the problem. Dimensionality reduction methods are mainly wavelet transforms and principal components analysis (PCA) [1, 30] which is also known as singular value decomposition (SVD) and Karhunen-
Loeve transform (KL V) [18] for transformation and projection of the original data for elimination of a subset of the original data in terms of variables covariance.

Numerosity reduction includes techniques substitute the original data by a smaller alternative set. The methods are parametric which mainly retains only data estimated parameters instead of original data such as regression and nonparametric which stores reduced representations of the data such as histograms, sampling and clustering.

Data compression strategy deals with transformations of data to reach the compressed representation of the original data. These methods, depends on the applied algorithm, may cause loose of data during the process of reconstruction of the data.

### 2.5.4 Feature Extraction and Feature Selection

All dimensionality reduction strategies and techniques are also classified as feature extraction and feature selection approaches.

Feature extraction is defined as transforming the original data into a new lower dimensional space through some functional mapping such as PCA and SVD [2, 42]. Most unsupervised dimensionality reduction techniques
are closely related to PCA, but this technique is not applicable to large datasets [61].

Feature selection is denoted as electing a subset of the original data (features) without a transformation based on some criteria in order to filter out irrelevant or redundant features, such as filter methods, wrapper methods and embedded methods [19, 52].

Although a broad understanding of all methods is not easy, general knowledge of the area categorizations is recommended in this study. Utilization of any mathematical model, algorithm and technique requires a complete knowledge of the purposes of solving the problem. For instance, selecting a proper class of dimensionality reduction such as feature selection (PCA, SVD, etc.) or feature extraction (filtering, wrapper and embedded) needs appropriate characterizing of the problem. Similarly, determining whether the data in the problem need a supervised or unsupervised method is a major part of analysis process which leads one to use appropriate techniques.

2.5.5 Principal Component Analysis

Principal Component Analysis (PCA) is one of oldest and most well-known, appropriate and applicable multivariate analysis techniques which is widely used to reduce dimensionality of data, enable one to transforms data, even
linear or nonlinear, from higher to lower dimension as needed. This reduction, if performs correctly, eliminates noise, redundant and undesirable data, retains the most of the useful data, saves resources such as time and memory used in data processing, as well as generating new significant features, variables and attributes [1, 30].

Basically, the principal function of PCA is rotational transformation the axes of data space along lines of maximum variance, as shown in Fig. 2.4, when the new axes are called principal component in order, based on their variance magnitude. The dimensionality reduction could be made by selecting first few principal components considering the fact of keeping necessary data.

According to Jolliffe [30] “The central idea of principal component analysis is to reduce the dimensionality of a dataset in which there are a large number of interrelated variables, while retaining as much as possible of the variation present in the dataset. This reduction is achieved by transforming to a new set of variables, the principal components, which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables.”

In the literature, principal components analysis (PCA), singular value decomposition (SVD) and Karhunen-Loeve transform (KLT) are mostly used equivalently. In the field of multivariate statistical analysis, these three terms
are very similar under certain circumstances but not necessarily identical [19]. Some other relevant terms with PCA in different fields are Hotelling transformation, independent component analysis (ICA), factor analysis (FA), empirical orthogonal functions (EOF), proper orthogonal decomposition (POD) eigenvalue decomposition (EVD), empirical eigenfunction decomposition (EED), empirical component analysis (ECA), spectral decomposition, and empirical modal analysis.

Some of suggested widespread methods to determine the appropriate number of components are [63]:

1) Bartlett’s Test; which is an analogous statistical test for component analysis if the remaining eigenvalues are equal. This method should no longer be employed.
2) Kaiser Criteria (K1); is the most commonly used method suggests to retain the components whose eigenvalues are greater than 1.0 and recommends to remove the components with eigenvalue of less than the average of the eigenvalues. Kaiser's method tended to severely overestimate the number of components.

3) Scree Test; is based on a graph of the eigenvalues. The scree test is simple to apply. The eigenvalues are plotted, a straight line is fit through the smaller values and those falling above the line are retained. In this method some complications may occur such no obvious break point or more than one break point. The scree test has been is effective when strong components are present and it this test has been found to be most accurate with larger samples and strong components in non-complex data structure.

4) Minimum Average Partial (MAP); a method based on the application of PCA and in the subsequent analysis of partial correlation matrices. The method seeks to determine what components are common, and is proposed as a rule to find the best factor solution, rather than to find the cutoff point for the number of factor. The MAP method was generally quite accurate and consistent when the component saturation was high or the component was defined by more than six variables.

5) Parallel Analysis (PA), involves a comparison of the obtained, real data
eigenvalues with the eigenvalues of a correlation matrix of the same rank
and based upon the same number of observations but containing only ran-
dom uncorrelated variables. This method is an adaptation of the K1 rule.
The general application of the PA method is difficult to recommend because
programs needed for its application are not widely available.
3.1 Overview

In this chapter, first proposed analytical model for transformation of the explanatory variable dataset to reach the logical representation as a sort of binary variables is presented. Next, in order to select most significant variables in terms of inefficiency, designed variable selection methods and heuristic clustering algorithms are introduced [46–48].
3.2 Analytical Model I

A multipurpose and flexible model for a type of time-to-event data with a large number of variables when the correlation between variables is complicated or unknown is presented. The objective is to simplify the original covariate dataset into a logical dataset by transformation lemma. Next, we show the validation of this designed logical model by correlation transformation [47, 48]. The analytic model [47] and its following methods and algorithms are potentially applicable solutions for many problems in a vast area of science and technologies.

The random variables $Y$ and $C$ represent the time-to-event and censoring time, respectively. Time-to-event data is represented by $(T, \Delta, U)$ where $T = \min(Y, C)$ and the censoring indicator $\Delta = 1$ if the event occurred, for instance failure is observed, otherwise 0. The observed covariate $U$ represents a set of variables. Let denote any observations by $(t_i, \delta_i, u_{ij}), \ i = 1 \ldots n, \ j = 1 \ldots p$. It is assumed that the hazard at time $t$ only depends on the survivals at time $t$ which assures the independent right censoring assumption [31].

The original time-to-event dataset may include any type of explanatory. Many time-independent variables in $U$ are binary or interchangeable with a binary variable such as dichotomous variable. Also, interpretation of bi-
nary variable is simple, understandable and comprehensible. In addition, the model is appropriate for fast and low-cost calculation. The general schema of high-dimensional event history dataset includes $n$ observations with $p$ variables as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Obs. #</th>
<th>Time to Event</th>
<th>Var. 1</th>
<th>Var. 2</th>
<th>...</th>
<th>Var. $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$t_1$</td>
<td>$u_{11}$</td>
<td>$u_{12}$</td>
<td>...</td>
<td>$u_{1p}$</td>
</tr>
<tr>
<td>2</td>
<td>$t_2$</td>
<td>$u_{21}$</td>
<td>$u_{22}$</td>
<td>...</td>
<td>$u_{2p}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$n$</td>
<td>$t_n$</td>
<td>$u_{n1}$</td>
<td>$u_{n2}$</td>
<td>...</td>
<td>$u_{np}$</td>
</tr>
</tbody>
</table>

Each array of $p$ variables vectors will take only two possible values, canonically 0 and 1. As discussed in Chapter 2, discretization method is applied to values by dividing the range of values for each variable into 2 equally sized parts. We define $w_{ij}$ as an initial splitting criterion equal to arithmetic mean of maximum and minimum value of $u_{ij}$ for $i = 1 \ldots n, \ j = 1 \ldots p$.

$$w_j = \frac{\max\{u_{ij}\} + \min\{u_{ij}\}}{2} \quad i = 1 \ldots n, \ j = 1 \ldots p \quad (3.1)$$

For any array $u_{ij}$ in the $n$-by-$p$ dataset matrix $\mathbf{U} = [u_{ij}]$, then assign a substituting array $v_{ij}$ as:
\[ v_{ij} = \begin{cases} 
0, & u_{ij} < w_{ij} \\
1, & u_{ij} \geq w_{ij} 
\end{cases} \] (3.2)

The criteria \( w_{ij} \) could be defined by expert using experimental or historical data as well. The proposed model assumes any array with a value of 1 as desired for expert and 0 otherwise. In other words, \( v_{ij} = 0 \) represent the lack of the \( j \)th variable in the \( i \)th observation. The result of the transformation is an \( n \)-by-\( p \) dataset matrix \( V = [v_{np}] \) which will be used in the following methods and algorithms. Also, we define time-to-event vector \( T = [t_n] \) including all observed event times. The logical model initially could be satisfied by proper design of data collection process by based on Boolean logic to generate binary attributes.

Therefore, the result of the transformation is an \( n \)-by-\( p \) dataset matrix \( V = [v_{np}] \) which will be used in the following methods and algorithms. Also, we define time-to-event vector \( T = [t_n] \) including all observed failure times, and \( S = [s_n] \) as survival function. The logical model initially could be satisfied by proper design of data collection process by based on Boolean logic to generate binary attributes.
3.2.1 Model Validation

Correlation Transformation

To validate the robustness of this model, first we show that the change of correlation between variables before and after transformation is not significant and the logical dataset has followed the same pattern and behavior as the original; in terms of correlation of covariates. We define correlation matrix for each of original and transformed dataset based on Pearson product-moment correlation coefficient; \( M = [m_{ij}] \) and \( N = [n_{ij}] \) where \( i = 1 \ldots n, \ j = 1 \ldots p \), where \( n_{ij} \) and \( m_{ij} \) denote covariance of variables \( i \) and \( j \) for original and transformed dataset respectively as follows:

\[
n_{ij} = \frac{1}{n - 1} \sum_{k=1}^{n} (u_{ik} - \bar{u}_i)(u_{jk} - \bar{u}_j) \quad i = 1 \ldots p, \ j = 1 \ldots p \quad (3.3)
\]

\[
m_{ij} = \frac{1}{n - 1} \sum_{k=1}^{n} (v_{ik} - \bar{v}_i)(v_{jk} - \bar{v}_j) \quad i = 1 \ldots p, \ j = 1 \ldots p \quad (3.4)
\]

where \((u_{ik}, v_{ik})\) and \((\bar{u}_i, \bar{v}_i)\) represent value of variable \( i \) in observation \( k \) and mean of variable \( i \) in each dataset respectively, and similarly the second parenthesis in equations 3.3 and 3.4 are defined for variable \( j \).

The experimental fitted line for the scatter plot of \( m_{ij} \) and \( n_{ij} \) for any dataset is \( y = a + bx \) where \( b \) is positive small and \( a \) is not significant. For instance, Fig. 3.1 shows the primary biliary cirrhosis (PBC) dataset (Section 3.2) for an experimental result of an uncensored data with the fitted line of \( y = \ldots \)
Figure 3.1: Comparison of covariate correlations in the original and the transformed dataset.

$0.0116 + 0.6356x$.

The proposed logical model validation and verification of the robustness were presented comprehensively in [46, 48].

Cox Proportional Hazards Model (PHM) Baseline

The next validation is to compare the estimates of the survival and hazard functions of the original and the transformed dataset to verify the proposed
model. Theoretically, a statistical test such as Wilcoxon rank sum test and log-rank test determine whether we can assume two sample data are equal or in other words, are collected from identical population. Based on Cox PHM baseline, if the survival or hazard functions of the original and the transformed dataset are similar, it proves that both datasets are not in-equivalent in term of hazard rate and the transformation does not change the effect of complete set of variables on survival function.

Although proportional hazards regression, as well as other multiple regression models will face several calculation errors dealing with high-dimensional data especially in with a huge number of covariates, computational methods show that the variation in the approximation of PHM baselines for equivalent time-to-event datasets is negligible. Comparing empirical product limit estimate for the survival function of the original with the PHM baseline survival function of the original dataset and transformed logical dataset in many tested sample datasets in this study shows that the transformation of the dataset does not change the nature of the covariates in terms of the survival function and the hazard function.

Fig. 3.2 depicts the empirical survival function, Cox PHM baseline survival function for original dataset, and Cox PHM baseline survival function for the transformed logical uncensored PBC dataset (Section 3.3). The Wilcoxon test score of comparing nonparametric baseline survival function of the origi-
inal dataset and transformed logical dataset is 0.9704 and according to the test concept, it shows that these two functions are not unequal. This result proves that we should expect no significant change in the covariate correlations due to the proposed transformation. We applied similar validation to many datasets in this research to verify this analytic model.

![Empirical survival function](image)

**Figure 3.2:** Empirical survival function (solid blue), Cox PHM baseline survival function for the original dataset (black dashed), and Cox PHM baseline survival function for transformed logical dataset (red dotted).

In order to select the most significant variables in terms of inefficiency, designed methods and heuristic algorithms are presented next.
3.3 Methods and Algorithms for Variable Selection

We design a class of methods applying on proposed logical model to select inefficient variables in a high-dimensional event history datasets. The major assumption to design appropriate methods for this purpose is that the variable which is completely inefficient solely can provide a significant performance improvement when engaged with others, and two variables that are inefficient by themselves can be efficient together [19]. Based on this assumption, we design three methods and heuristic algorithms to select inefficient variables in event history datasets with high-dimensional covariates. We use Kaplan-Meier estimator in this study to estimate survival probabilities as a function of time. A nonparametric test could be used to test a null hypothesis that whether two samples are drawn from the same distribution, as compared to a given alternative hypothesis. Among many nonparametric tests for comparing survival functions for aforementioned propose, we use log-rank test in our methods as a best fit for comparison of two nonparametric distributions. This test is the most commonly used for a typical study under different models for the relationship between the groups [24, 32].

The $n$-by-$p$ matrix $V$ is the prepared transformed logical dataset according
to Section 3.1, where $n$ is the number of observations, $p$ is the number of variables, and $k$ is the estimated subset size to select for calculation parts in the algorithms.

Recalling $V$ which is constructed by $k$ observation vectors corresponding to each of the variables, $D = [d_{kp}]$ as a $k$-by-$p$ matrix is a selected subset of $V$ and $k$ is defined as the number of observations in any subset of $V$, where $k \leq n$. For any variable $i$, we define vector $O^i$ as a time-to-event vector which includes failure times of any observation $j$ the value of $v_{ij}$ is one. Similarly, we define vector $Z^i$ including failure times of any observation $j$ where the value of $v_{ij}$ is zero. The vectors $R$ and $S$ are defined as follow:

$$r_i = \begin{cases} t_i, & \sum di. \geq 0 \\ 0, & Otherwise \end{cases} \quad i = 1 \ldots n \quad (3.5)$$

$$s_i = \begin{cases} t_i, & \prod di. = 1 \\ 0, & Otherwise \end{cases} \quad i = 1 \ldots n \quad (3.6)$$

Vector $R$ is constructed by all non-zero arrays $r$ and similarly vector $S$ is constructed by all non-zero arrays $s$.

A class of methods and algorithms to select inefficient variables are following:
3.3.1 Singular Variable Effect

The objective of Singular Variable Effect (SVE) method is to determine the efficiency of a variable by analyzing the effect of the presence of any variable singularly in comparison with its absence in a transformed logical dataset.

For p variable, we aim to set vector $\Delta = [\delta_i]$ where $i = 1 \ldots p$ to rank the efficiency of the variables. The preliminary step for the highest efficiency in this method is to initially clustering the variables based on the correlation coefficient matrix of original dataset, $M$, and choose a representative variable from each highly correlated cluster and eliminate the other variables from the dataset. For instance, for any given dataset, if three variables are highly correlated, only one of them is selected randomly and the other two are eliminated from the dataset. The result of this process assures that the remaining variables for applying methods and heuristic algorithms are not highly correlated.

As an outcome of the SVE procedure, if one hopes to reduce the number of variables in the dataset for further analysis, could eliminate less efficient identified variables or if aims to concentrate on a reduced number of variables, could choose a category of more efficient identified variables as well.

Heuristic algorithm for SVE method is:

\[
\text{for } i = 1 \text{ to } p \text{ do}
\]
Calculate $O^i$ and $Z^i$ for variable $i$ observation vector in dataset $V$

Compare $T$ and $O^i$ with Wilcoxon rank sum test

Save the test score for variable $i$ as $\alpha_i$

Compare $T$ and $O^i$ with Wilcoxon rank sum test

Save the test score for variable $i$ as $\beta_i$

Calculate $\delta_i = \alpha_i - \beta_i$

end for

Return $\Delta = [\delta_p]$ as the variable efficiency vector

3.3.2 Nonparametric Test Score

The log-rank test score variable selection method is applied for selecting a subset of the best and the worst variables in terms of efficiency. The Nonparametric Test Score (NTS) method is a variable clustering technique which selects set of size $k$ variables from the transformed logical dataset $V$ and calculates the score of each variable in two levels. The first level is to determine the priority of the variable efficiency via the score reached by the frequency of the presence of each variable from the rejected subsets in comparison with the original time-to-even vector $T$. We code this level of calculation with letter F. The second level rates the variables by the cumulative score of each variable from comparisons of selected subsets of all nonparametric test results with the original time-to-even vector $T$. This level acts as a searching
procedure to detect the less efficient variables. The code which this level is
denoted by is the letter C. Randomization (RN) algorithm randomly chooses
a defined \(l\) subset of \(k\) from the \(V\), transformed logical dataset of \(p\) vari-
able. We define a randomization dataset matrix \(\Psi = [\psi_{lk}]\) where each row
is formed by \(k\) variable identification numbers in any selected subsets for
overall \(l\) subsets. Heuristic algorithm of NTS method level \(F\) is:

\[
\textbf{for } i = 1 \text{ to } q \text{ do}
\]
\[
\begin{align*}
&\text{Compose the dataset } D_i \text{ for variable set } i \text{ in } \Psi \text{ including variables} \\
&\psi_{ij} \text{ where } j = 1 \text{ to } k \\
&\text{Calculate } R_i \text{ over the dataset } D_i \\
&\text{Compare } T \text{ and } R_i \text{ with log-rank test} \\
&\text{Save the test score for variables in subset } i \text{ as } \psi_{i(k+1)}
\end{align*}
\]
\[
\textbf{end for}
\]
\[
\text{Eliminate rows of } \Psi \text{ where the array } k+1 \text{ of the row has a value of more}
\text{than 0.05, and call this new set } \Psi^*.
\]
\[
\text{Count the contribution (presence) of each variable } h \text{ based on its}
\text{identification number in the}
\]
\[
\Psi^* \text{ first } k \text{ columns and save as } \gamma_h
\]
\[
\text{Return } \Gamma = [\gamma_p] \text{ as the variable efficiency vector}
\]
Heuristic algorithm of NTS method level C is presented is:

\begin{verbatim}
for i = 1 to l do
    Compose the dataset \( D_i \) for variable set \( i \) in \( \Psi \) including variables \( \psi_{ij} \) where \( j = 1 \) to \( k \)
    Calculate \( R_i \) over the dataset \( D_i \)
    Compare \( T \) and \( R_i \) with log-rank test
    Save the test score for variables in subset \( i \) as \( \psi_{i(k+1)} \)
end for

Assume \( \Omega = [\omega_p] \) as the reverse variable efficiency vector where initially each array as the cumulative contribution score corresponding to a variable is zero

for i = 1 to l do
    for j = 1 to k do
        Add the value of \( \psi_{i(k+1)} \) to the cumulative contribution score \( \omega_p \) of the variable \( i \) based on its identification number = \( \psi_{ij} \)
    end for
end for

Return \( \Omega = [\omega_p] \) as the variable inefficiency vector
\end{verbatim}
3.3.3 Splitting Semi-Greedy Clustering

Splitting Semi-Greedy (SSG) method to select an inefficient variable subset is proposed. A clustering procedure through randomly splitting approach to select the best local subset according to a defined criterion incorporated. In this method we use block randomization which is designed to randomize subjects into equal sample sizes groups. A nonparametric test is used to test a null hypothesis that whether two samples are drawn from the same distribution, as compared by a given alternative hypothesis. Wilcoxon rank sum test is used in this method.

The concept of this method is inspired by the semi-greedy heuristic [14, 20] and tabu search [17]. Criterion of this search is similar to The Nonparametric Test Score (NTS) method [48] which is to collect the most inefficient variable subset via Wilcoxon rank sum test score. At each of $l$ trials, all $p$ variables from the transformed logical dataset $V$ are randomly clustered into subsets of size $k$ variables, where one cluster possibly contains less than $k$ variables and the number of clusters is equal to $\lceil p/k \rceil$. To calculate score summation for each variable over all trials, a randomization dataset matrix $\Xi = [\xi_{lk}]$ where each row is formed by $k$ variable identification numbers in any selected subsets for all $l$ trials. Comprehensive experimental results for validation of the proposed methods by comparison with similar methods are presented in Sections 3.4 and 3.5. Heuristic algorithm for SSG method is:
for $i = 1$ to $l$ do

Split the data into equally sized subsets

Compose the dataset $D$ for each subset

Calculate $R$ over the $D$ for each subset

Compare $T$ and $R$ with Wilcoxon rank sum test and save the test score for each subset one by one

Select a subset with the highest test score

Save the test score for variables in the selected subset as $ξ_{i(k+1)}$

end for

Assume $Θ = [θ_p]$ as the reverse variable efficiency vector

where initially each array as the cumulative contribution score corresponding to a variable is zero

for $i = 1$ to $l$ do

for $j = 1$ to $k$ do

Add the value of $ξ_{i(k+1)}$ to the cumulative contribution score $θ_p$

of the variable $i$ based on its identification number = $ξ_{ij}$

end for

end for

Return $Θ = [θ_p]$ as the variable inefficiency vector
3.3.4 Weighted Time Score

The Weighted Time Score (WTS) method is a variable clustering technique which selects set of size $k$ variables from the transformed logical dataset $V$ and calculates the score of each variable. The first step is to determine the observations in a selected subset which all $k$ variables are 1 for that observation and eliminate other observation from subset. Cumulative time score over the vector $T$ credit each of variables in the subset. Final score of all variables is reached by aggregation of those credits in $l$ trials. Randomization algorithm randomly chooses a defined $l$ subset of $k$ from the $V$, transformed logical dataset of $p$ variable. We define a randomization dataset matrix $\Psi = [\psi_{lk}]$ where each row is formed by $k$ variable identification numbers in any selected subsets for overall $l$ subsets. Heuristic algorithm for WTS method is:

\begin{verbatim}
for i = 1 to l do
    Compose the dataset $D_i$ for variable set $i$ in $\Psi$
    including variables $\psi_{ij}$ where $j = 1$ to $k$
    Calculate $S_i$ over the dataset $D_i$
    Calculate $\sum t_i$ for $S_i$ as a time score
    Save the time score for variables in subset $i$ as $\psi_{i(k+1)}$
end for

Assume $\Omega = [\omega_p]$ as the reverse variable efficiency vector
where initially each array as the cumulative contribution score
\end{verbatim}
corresponding to a variable is zero

for $i = 1$ to $l$ do
  for $j = 1$ to $k$ do
    Add the value of $\psi_{i(k+1)}$ to the cumulative contribution score $\omega_p$ of the variable $i$ based on its identification number $= \psi_{ij}$
  end for
end for

Return $\Omega = [\omega_p]$ as the variable inefficiency vector

Comprehensive experimental results for validation of the proposed methods by comparison with similar methods for right-censored dataset (Section 3.4) and uncensored dataset (Section 3.5) are presented next.

### 3.4 Experimental Results and Analysis for Right-Censored Data

To evaluate the performance of the proposed methods for right-censored data, NTS and SSG methods and algorithms under different types of datasets including collected and simulated data is investigated. In order to obtain an estimation of desired number of variables in any selected subset of these methods for calculations in hybrid algorithms, we use principal component
analysis (PCA) scree plot criterion [63]. The criterion for this evaluation is based on eigenvalue of components in a dataset. The cut-off in the scree plot is interpreted as a set of significant eigenvalues among all components which leads to determine $k$, number of efficient variables in a given dataset. We apply this technique on the original dataset to determine $k$.

First, the well-known primary biliary cirrhosis (PBC) dataset (Fleming and Harrington 1991) is considered as the sample collected dataset. These data are from a double-blinded randomized trial including 312 observations. This right-censored dataset contains 17 variables in addition to censoring information, identification number and event times for each observation. Experimental results for the 276 observations with complete records are presented (36 out of 312 observations are incomplete). For the original PBC dataset, approximate value of $k$ is 3. This value is the subset size in the algorithms applied on $V$ transformed logical dataset $n$-by-$p$ matrix, shown in Fig. 3.3.

Another set up before applying methods and algorithms is to preliminary determine which variables are more behaving similarly, using modified k-means clustering algorithm. Basically, k-mean algorithm partitions the points in the $n$-by-$p$ dataset matrix into $k$ clusters. The default of calculation is Squared Euclidean distance. Our modified k-mean algorithm changes the dimension of transformed logical dataset and returns clusters of the variables in terms of the observation. The result gives us an acceptable view of expected clus-
Figure 3.3: Scree plot of the original PBC dataset including 17 variables and 276 observations.

tering in the next parts. Number of clusters is calculated by the quotient’s nearest integer value of the number of the dataset variables $p$ divided by the number of the subset variables $k$. The result of this preliminary evaluation for the transformed logical uncensored PBC dataset is shown in Table 3.2.

To verify the performance of the proposed methods, the result of methods and algorithms for the transformed logical PBC dataset is compared with the Random Survival Forest (RSF) method results [27,28], Additive Risk Model
Table 3.2: Modified k-means clustering algorithm result for the transformed logical uncensored PBC dataset. Number of clusters is equal to $\| \frac{p}{k} \|$.

<table>
<thead>
<tr>
<th>Cluster #</th>
<th>Variable #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4, 6, 7</td>
</tr>
<tr>
<td>2</td>
<td>3, 5, 10, 17</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>8, 9, 11, 12, 13, 14, 15, 16</td>
</tr>
</tbody>
</table>

(ADD) [36], and Weighted Least Square (LS) [24] results for variable selection in PBC dataset are given in Table 3.3. A comprehensive comparison of RSF, ADD and LS performance with other relevant methods in high-dimensional time-to-event data analysis such as Cox’s Proportional Hazard Model, LASSO and PCR has been presented in [24, 28, 36].

The survival function results of NTS-RN algorithm for 100 trials on the transformed logical uncensored PBC dataset is shown in Fig. 3.4.

Each number in Tables 3.3 and 3.4 represents a specific variable in experiment dataset. For example, in Table 3.3, variable #15 is among the selected less efficient variables by methods NTS (F), SSG, RSF and LS.

From the results shown on Tables 3.3, the NTS (F) method has closest performance to the SSG. This is an advantage of these methods that more than 80% of inefficient variables which has been detected by other methods (RSF,
Figure 3.4: Empirical transformed logical uncensored PBC dataset survival function (solid red) with 99% confidence bounds (red dotted). Subset of $k$ variable with failed to reject Wilcoxon rank sum test result (gray), and subset of $k$ variable with rejected Wilcoxon rank sum test result (black).

LS and ADD) are collected by proposed algorithms at significantly short period of calculation. The robustness of this class of methods has examined for several sample collected time-to-event data with high-dimensional covariates in this study.

Continuing the validation of the proposed methods, we set $n = 400$ observations and $p = 25$ variables and simulated event times from a pseudorandom algorithm. In addition to constant and periodic binary numbers, normal and exponential distributed pseudorandom numbers are generated as independent values of explanatory variables. The censored observations were also
Table 3.3: Selected less efficient variables in all proposed methods and comparison to RSF, ADD, and LS method results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selected Less Efficient Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTS (F)</td>
<td>1, 3, 5, 6, 10, 15, 17</td>
</tr>
<tr>
<td>NTS (C)</td>
<td>2, 5, 10, 11, 13, 17</td>
</tr>
<tr>
<td>SSG</td>
<td>1, 3, 5, 10, 15, 17</td>
</tr>
<tr>
<td>RSF</td>
<td>1, 3, 5, 12, 13, 14, 15, 17</td>
</tr>
<tr>
<td>ADD</td>
<td>1, 2, 5, 12, 14</td>
</tr>
<tr>
<td>LS</td>
<td>1, 2, 3, 14, 15, 17</td>
</tr>
</tbody>
</table>

generated independent of the events. Additionally, some variables are set as a linear function of event time intentionally. We present the results of methods and algorithms applying the simulated data in Table 3.4. These results are compared with the simulation defined pattern and the comparison verifies the performance of all proposed methods and algorithms. In the simulation, defined pattern includes five inefficient variables, No. 5, 10, 15, 20, and 25.

Table 3.4: Selected less efficient variables in all proposed methods and comparison to simulation defined pattern.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selected Less Efficient Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTS (F)</td>
<td>5, 10, 25</td>
</tr>
<tr>
<td>NTS (C)</td>
<td>5, 10, 25</td>
</tr>
<tr>
<td>SSG</td>
<td>5, 10, 25</td>
</tr>
<tr>
<td>Definition</td>
<td>5, 10, 15, 20, 25</td>
</tr>
</tbody>
</table>
ables with identification number 5, 10 and 25 are detected as less efficient variables among all 25 simulated variables by NTS (F), NTS (C) and SSG methods. In this example, one could eliminate these less efficient identified variables if the objective is to reduce the number of variables in the dataset for further analysis.

In addition, a set of simulation numerical experiments are presented to evaluate performance of the proposed methods as shown in Table 3.5. For that, six different simulation models are defined. Setting of these models are as explained above for \( n = 400 \) observations and \( p = 25 \) variables following same data generating algorithm. Number of significant inefficient variables in each simulation model in order are \( m = \{4, 5, 6\} \) as presented in Definition in Table 3.5. The simulations are repeated 100 times independently. For each method, \( m \) represents integer average number of determined and selected inefficient variables and \( p \) denotes performance of the method, for 100 trails, where

\[
p = \frac{m_{\text{method}}}{m_{\text{definition}}} \tag{3.7}
\]

and \( p \) is the average of performance of each method. The adjusted censoring rates for all simulation numerical experiments are \( 50 \pm 5\% \).

Graphical tool for interpretation of the results for proposed methods, hybrid scattering for variable inefficiency for three criteria as (a) NTS (F) score, (b) NTS (C) score and (c) SSG score for PBC data experiment results is displayed.
Table 3.5: Performance of the proposed methods based on six simulation numerical experiment. $m$ is integer average number of selected inefficient variable and $p$ is performance of method based on 100 replications.

<table>
<thead>
<tr>
<th>Method</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$p$</td>
<td>$m$</td>
<td>$p$</td>
<td>$m$</td>
<td>$p$</td>
<td>$p$</td>
</tr>
<tr>
<td>NTS (F)</td>
<td>3</td>
<td>0.75</td>
<td>3</td>
<td>0.75</td>
<td>4</td>
<td>0.80</td>
<td>4</td>
</tr>
<tr>
<td>NTS (C)</td>
<td>3</td>
<td>0.75</td>
<td>3</td>
<td>0.75</td>
<td>4</td>
<td>0.80</td>
<td>5</td>
</tr>
<tr>
<td>SSG</td>
<td>2</td>
<td>0.50</td>
<td>3</td>
<td>0.75</td>
<td>3</td>
<td>0.60</td>
<td>4</td>
</tr>
<tr>
<td>Definition</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

in Fig. 3.5. Each variable is exposed by a circle icon where two axes represent the standardized criterion (a) and (b) and the diameter of each circle demonstrate the standardized criterion (c). These inefficiency score criteria are standardized in the range of $[0, 25]$ and for clear graphics presentation, plot range is $[-5, 25]$. Therefore, it is interpreted from the hybrid scatter plot that each variable with larger diameter and more distance from the center has less efficiency and is an ideal candidate to remove from dataset if it is desired.

3.5 Experimental Results and Analysis for Uncensored Data

To evaluate the performance of the designed methods for uncensored data, first PBC dataset is considered as the sample collected dataset. The purpose of this section is to examine and validate the proposed methods in
Figure 3.5: Hybrid scatter plot - Upper right variables have less efficiency.

this chapter with uncensored time-to-event data. These dataset includes 111 uncensored complete observations and 17 explanatory variables. To verify the performance of the proposed methods for uncensored data, the result of these methods and algorithms for the transformed logical uncensored PBC dataset including 111 uncensored complete observations and 17 explanatory variables is compared with the results of Nonparametric Test Score (NTS) method [48], Random Survival Forest (RSF) method [27, 28], Additive Risk Model (ADD) [36], and Weighted Least Square (LS) method [24] for similar
dataset variable selection, given in Table 3.6.

Each number in Tables 3.6 and 3.7 represents a specific variable in experiment dataset. For example, in Table 3.6, variable #1 is determined from the results as an inefficient variable by all methods.

Table 3.6: Selected inefficient variables in all proposed methods and comparison to NTS, RSF, ADD, and LS method results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selected Inefficient Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVE</td>
<td>1, 3, 5, 10, 13, 17</td>
</tr>
<tr>
<td>SSG</td>
<td>1, 3, 5, 10, 15, 17</td>
</tr>
<tr>
<td>WTS</td>
<td>1, 3, 5, 10, 15, 17</td>
</tr>
<tr>
<td>NTS</td>
<td>1, 3, 5, 6, 10, 15, 17</td>
</tr>
<tr>
<td>RSF</td>
<td>1, 3, 5, 12, 13, 14, 15, 17</td>
</tr>
<tr>
<td>ADD</td>
<td>1, 2, 5, 12, 14</td>
</tr>
<tr>
<td>LS</td>
<td>1, 2, 3, 14, 15, 17</td>
</tr>
</tbody>
</table>

From the results shown on Tables 1, the SSG and WTS methods have a same performance. More than 80% of inefficient variables which has been detected by other methods (NTS, RSF, LS and ADD) are collected by proposed algorithms at significantly shorter calculation period, where the robustness of this class of methods has examined for several sample datasets.

To show variable inefficiency through three designed methods SVE, SSG, and WTS, graphical representation for the experiment results for uncensored PBC dataset is depicted in Fig. 3.6. Each variable with larger radius
and more distance from the center is less efficient and an ideal candidate to remove from dataset if it is desired.

Figure 3.6: Radar plot of inefficient variables: Normalized inefficiency results from the transformed logical uncensored PBC dataset by SVE algorithm (red), SSG algorithm (green), and WTS algorithms (yellow).

A simulation is designed to examine and validate the proposed methods for uncensored data. We set \( n = 400 \) observations and \( p = 15 \) variables and simulated event times from a pseudorandom algorithm. We also set first five variables inefficient, where first two are absolutely inefficient. Some variable
vectors are set as a linear function of event time data in addition to constant and periodic binary numbers as well as normal and exponential distributed pseudorandom numbers as independent values of explanatory variables. The results of methods and algorithms applying the simulated data are presented in Table 3.7. These results are compared with the results from NTS method. From the simulation defined pattern as shown in Table 3.7, the comparison verifies the performance of all proposed methods.

Table 3.7: Selected inefficient variables in all proposed methods and comparison to NTS results and simulation defined pattern.

<table>
<thead>
<tr>
<th>Method</th>
<th>Selected Inefficient Variables (No.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVE</td>
<td>1, 2, 3, 10</td>
</tr>
<tr>
<td>SSG</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>WTS</td>
<td>1, 2, 3, 5, 12</td>
</tr>
<tr>
<td>NTS</td>
<td>1, 2, 5</td>
</tr>
<tr>
<td>Definition</td>
<td>1, 2, 3, 4, 5</td>
</tr>
</tbody>
</table>

Inefficiency analysis results for the simulation experiment shows that variables with identification number 1, 2 and 3 are detected as inefficient variables by all proposed methods. To reduce the number of variables in the dataset for further analysis, these explanatory variables are the best candidates to be eliminated from the dataset.
Chapter 4

Methodology II: Heuristic Randomized Decision-Making Methods through Weighted Accelerated Failure Time Model

4.1 Overview

This chapter presents an analytical model for normalization the explanatory variable dataset to reach an appropriate representative dataset. Next, designed heuristic randomized decision-making methods and algorithms
to select most significant variables in terms of efficiency through accelerated failure time model are introduced. Then numerical simulation experiment is developed to investigate the performance and validation of the proposed methods [12, 49].

4.2 Analytical Model II

A simple analytic model to analyze complicated right-censored time-to-event data with high-dimensional explanatory variables in order to facilitate variable selection procedure in decision-making procedure is developed. Transforming data values into appropriate computational form is a commonly used approach. There are many advantages of using discrete values over continuous; as discrete variables are easy to understand and utilize, more compact, and more accurate. In this study, for a time-to-event dataset including discrete values, a normalization analytical model is defined.

As explained in Chapter 2, the random variables $Y$ and $C$ represent the time-to-event and censoring time, respectively. Time-to-event data is represented by $(T, \Delta, U)$ where $T_i = \min(Y_i, C_i)$ and the censoring indicator $d_i = 1$ if the event occurs, otherwise $d_i = 0$. The observed covariate $U$ represents a set of variables. Let denote any observations by $(t_i, d_i, u_{ij}), i = 1 \ldots n, j = 1 \ldots p$. It is assumed that the hazard at time $t$ only depends on the survivals at time $t$ which assures the independent right
Data normalization is a method used to scale data and standardize the range of independent variables or features of data to allow the comparison of corresponding normalized data, also known as feature scaling. As a simple and well-known rescaling method, unity-based normalization scales in \([0, 1]\) range. One problem in this process is the appearance of outliers that takes extreme values and a solution to remove the outliers before normalization could be using a threshold as below algorithm:

1. Arrange data in order
2. Calculate first quartile \(Q_1\), third quartile \(Q_3\), and the interquartile range \(IQR = Q_3 - Q_1\)
3. Compute the range of \(Q_3 + 1.5(IQR)\) and \(Q_1 - 1.5(IQR)\)
4. Remove any value outside this range as an outlier

The original dataset may include any type of explanatory data as binary, continuous, categorical, or ordinal data. For any \(n\)-by-\(p\) dataset matrix \(U = [u_{ij}]\), there are \(n\) independent observations and \(p\) variables. To normalize the dataset, we define minimum and maximum value of each explanatory variable value in vector \(U_j\) for \(j = 1 \ldots p\) as:
For any array $u_{ij}$, we allocate a normalized substituting array $v_{ij}$ as:

$$v_{ij} = \frac{u_{ij} - u_{minj}}{u_{maxj} - u_{minj}} \quad i = 1 \ldots n, \quad j = 1 \ldots p \quad (4.2)$$

The result of the transformation is an $n$-by-$p$ dataset matrix $V = [v_{np}]$ which will be used in the designed heuristic methods and algorithms. Also, we define time-to-event vector $T = [t_n]$ including all observed event times. Application of a robustness validation method proposed by [12,46–48], indicates that the change of correlation between variables before and after transformation is not significant.

As a brute-force search algorithm outcome, the “all subset models” method is the simplest and computationally consuming which generates all possible subsets including combinations of $p$ explanatory variables, totally $2^p - 1$ subset. This exhaustive method is not suitable for large numbers of variables. Another approach is to develop models including combinations of the $k$ variables, called selective, where $1 \leq k < p$. Consequently the total number of models is given by:
\[
\frac{p!}{k!(p-k)!} \ll 2^{p-1} \tag{4.3}
\]

For instance, considering a problem where \( p = 100 \) and \( k = 3 \), the total number of subsets in exhaustive and selective models are 1048575 and 1114 respectively. Therefore finding a reasonable \( k \) is beneficial [38]. For a comprehensive discussion for evaluation a desired number of variables in any selected subset, \( k \), see [48]. In any given dataset and before transformation, the suggested criterion for this evaluation is based on eigenvalue of components determined from principal component analysis (PCA) method. The evaluated \( k \) is applied in heuristic randomized methods and algorithms to select the most significant efficient variables in order to regulate decision-making process which are presented next. The following methods utilize random subset selection concepts to evaluate coefficient score of each variable through AFT model by applying weighed least square technique.

### 4.3 Methods and Algorithms for Variable Classification

The preliminary step for the highest efficiency in proposed methods is to cluster the variables based on the correlation coefficient matrix of original dataset \( \mathbf{M} = [m_{ij}] \), and choose a representative variable from each highly correlated cluster, then eliminate the other variables from the dataset. Let
Denote covariance of variables $i$ and $j$. Pearson product-moment correlation coefficient matrix is defined as

$$m_{ij} = \frac{1}{n - 1} \sum_{k=1}^{n} (v_{ik} - \bar{v}_i)(v_{jk} - \bar{v}_j) \quad i = 1 \ldots p, j = 1 \ldots p \quad (4.4)$$

$v_{ik}$ and $\bar{v}_i$ in order represent value of variable $i$ in observation $k$ and mean of variable $i$, and similarly second parenthesis is defined for variable $j$. For instance, for any given dataset if three variables are highly correlated in $M$, only one of them is selected randomly and the other two are eliminated from the dataset. The outcome of this process assures that the remaining variables for applying methods are not highly correlated.

A method is developed for the proposed normalization model to select efficient variables in right-censored time-to-event datasets with high-dimensional explanatory variables. The basic assumption to design appropriate methods for this purpose is that the variable which is completely inefficient solely can provide a significant performance improvement when engaged with others, and two variables that are inefficient by themselves can be efficient together [44]. In these methods we use block randomization which is designed to randomize subjects into equal sample sizes groups. Generally randomization is the process to select or assign subjects with the same chance. Many randomization procedures and methods have been proposed for the random assignment; such as simple, replacement, block, and stratified [34].
To obtain variable coefficients estimation, let $w_{in}$ be the KM weights. For $i = 1 \ldots n$ these weights can be calculated using the expression

$$w_{in} = \hat{S}(\ln T_i) - \hat{S}(\ln T_{i-1})$$

(4.5)

Considering the definition of KM estimator $\hat{S}$ as equation (2.11), $w_{in}$ is obtained [32] by

$$w_{1n} = \frac{d_1}{n}, \quad w_{in} = \frac{d_i}{n - i + 1} \prod_{j=1}^{i-1} \left(\frac{n - j}{n - j + 1}\right)^{d_i}, \quad i = 2 \ldots n$$

(4.6)

Recall AFT model 2.17, the estimator of $\beta$ is

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} w_{in}[(\ln T_i) - \beta X_i]^2$$

(4.7)

The matrix calculation leads to an estimator solution as

$$\hat{\beta} = (X'W X)^{-1} X' W \ln T$$

(4.8)

where $X'$ is transpose of $X$. The equation 4.8 is used to estimate variable coefficients in heuristic decision making methods where $X \equiv V$.

Determining $k$ as a number of observations in any subset of transformed normalized dataset $V$, as described in Section 4.2. Where $k \leq n$, then $k$-by-$p$ sub-dataset matrix $D = [d_{kp}]$ is defined as a selected subset of $V$. Also vector $B = [b_k]$ will be utilized in following methods as a coefficient estimator vector corresponding to the explanatory variables for sub-dataset $D$. 

86
In continue, this chapter provides heuristic randomized decision-making algorithms for implementing variable selection through determining variable efficiency recognition procedure based on defined coefficient score in right-censored high-dimensional time-to-event data. The proposed methods utilize random subset selection concept to evaluate coefficient score of each variable through accelerated failure time model by applying weighed least square technique.

4.3.1 Ranking Classification

The concept of Ranking Classification (RC) method is inspired by the semi-greedy heuristic [14, 20]. The objective of this method is to introduce efficient variables in terms of correlation with survival time for an event time through randomly splitting approach by selecting the best local subset using a coefficient score. This method is a ranking classification using semi-greedy randomized procedure to split all explanatory variables into subset of predefined size $k$ using equal-width binning technique. The number of bins is equal to

$$c = \lceil p/k \rceil \quad (4.9)$$

Note that if $p$ is not divisible by $k$, then one bin contains less than $k$ variables. At each of $l$ trials where $l$ is a defined large reasonable number, a local coefficient score is calculated for any variable cluster at any bin. For $i$'th

87
cluster, this coefficient score is given by

\[ \tilde{\beta}_i = \left( \sum_{j=1}^{k} (\hat{\beta}_j)^2 \right)^{1/2} \]  

(4.10)

\( \hat{\beta}_j \) is obtained by weighted least squares estimation using 4.8 where \( X \equiv D \).

As a semi-greedy approach, the largest calculated \( \beta^i \) in each trial gives the selected cluster including \( k \) variables and the coefficient of these variables will be counted into a cumulative coefficient score for each of the cluster variables respectively. To calculate this score summation for each variable over all trials, a randomization dataset matrix \( \Xi = [\xi_{ck}] \) is set where each row as a bin is filled by \( k \) randomly selected variables' identification numbers in each trial. Ranking classification will be applied at the end of \( l \) trials based on accumulation of each variable score by vector \( \Theta = [\theta_p] \). Heuristic algorithm for the proposed method is following.

Let \( \Theta = [\theta_p] \) be the variable efficiency rank vector where initially each array as the cumulative contribution score corresponding to a variable is zero

for \( i = 1 \) to \( l \) do

Randomly split the dataset \( V \) into equally sized \( c \) subsets

Compose \( \Xi \) by \( c \) subsets

for \( j = 1 \) to \( c \) do

Compose the sub-dataset \( D^j \) for variable subset \( j \) in \( \Xi \) including
variables $\xi_{jq}$ for $q = 1$ to $k$

Calculate $W^j$ over the sub-dataset $D^j$

Calculate coefficient vector $B^j$ over the sub-dataset $D^j$ using $4.8$

Calculate coefficient score $\tilde{\beta}^j$ corresponding $B^j$ over the sub-dataset $D^j$ using $4.10$

end for

Select a subset with the highest $\tilde{\beta}^j$, call this new subset $D^*$

for $j = 1$ to $k$ do

Add the value of $b^j_i$ corresponding the variable $\xi_{ij}$ to the cumulative coefficient score $\theta_p$ of the variable $q$ based on its identification number $= \xi_{ij}$

end for

end for

Normalize $\Theta = [\theta_p]$

Return $\Theta$ as variable efficiency cumulative coefficient score for ranking

---

4.3.2 Randomized Resampling Regression

The Randomized Resampling Regression (3R) is a classification method inspired by resampling methods which randomly selects a subset of size $k$ out of $p$ variables from the transformed normalized dataset $V$, and estimates regres-
sion coefficients of subset variables using weighted least squares estimation described in Section 2.3. Then repeating the procedure for a defined large reasonable number of trials, \( l \)[48], the accumulation of each variable coefficient over all trials gives a final estimation score which leads to rank variables. Cumulative coefficient of an explanatory variable is an applicable estimate over a high-dimensional time-to-event dataset. We define a randomization dataset matrix \( F = [f_{ik}] \) where each row is formed by \( k \) variables’ identification numbers in any selected subsets and for overall \( l \) subsets. Heuristic algorithm for the 3R method is:

---

Let \( \Omega = [\omega_p] \) be the variable efficiency rank vector where initially each array as the cumulative contribution score corresponding to a variable is zero

\[
\text{for } i = 1 \text{ to } l \text{ do}
\]

\[
\begin{align*}
&\text{Compose the sub-dataset } D^i \text{ for variable subset } i \text{ in } F \text{ including variables } f_{ij} \text{ for } j = 1 \text{ to } k \\
&\text{Calculate } W^i \text{ over the sub-dataset } D^i \\
&\text{Calculate coefficient vector } B^i \text{ over the sub-dataset } D^i \text{ using equation } 4.10 \\
&\text{for } j = 1 \text{ to } k \text{ do} \\
&\text{Add the value of } b^i_j \text{ corresponding the variable } f_{ij} \text{ to the cumulative coefficient score } \omega_p \text{ of the variable } q \text{ based on its identification number } = f_{ij}
\end{align*}
\]
The simulation experiment, results and analysis for RC method are following in next section.

4.4 Simulation Experiment, Results and Analysis for RC Method

This section includes RC method verification by simulation experiment, and also experimental result and analysis are presented.

4.4.1 Simulation Design and Experiment

To evaluate the performance and investigate the accuracy of the proposed methods, simulation experiments are developed. We set \( n = 1000 \) observations, \( p = 25 \) variables. Four different simulation model are generated by
pseudorandom algorithms for $i = 1 \ldots p$ as

$$u_i^1 \sim U(-1,1), \quad u_i^2 \sim N(0,1), \quad u_i^3 \sim Exp(1), \quad u_i^4 \sim Weib(2,1)$$

(4.11)

by uniform, normal, exponential, and Weibull distributions respectively, where $U_j, j = 1 \ldots 4$ is explanatory variable matrix including $p$ variable vectors represent the original dataset $U$ for $j$'th model. In each simulation, the coefficient components used in 4.9 are set as

$$\beta_{ii} = 10, \quad \beta_{jj} = 1, \quad \beta_{kk} = 0.1,$$

$$ii = 1 \ldots 5, \quad jj = 6 \ldots 15, \quad kk = 16 \ldots 25$$

(4.12)

Simulated event time vector $Y$ is obtained from 2.17 where $X \equiv Y$ and the censoring indicator is generated using a uniform distribution. To improve accuracy in calculating survival time vector $t$, an error vector is considered as $E \sim N(0,0.1)$ in each simulation. In order to obtain $k$ as an estimation of desired number of variables in any selected subset, principal component analysis (PCA) is used, which gives $k = 3$.

All Four simulation models are applied in the proposed algorithm. A sample of numerical results of simulation experiment when data is generated by the second model $U^2$ and applied in the proposed algorithm is depicted in Fig. 4.1. Variables with larger coefficient scores $\tilde{\beta}$ are more efficient and variables with smaller scores are ideal candidate to be eliminated from the dataset if it
4.4.2 Result and Analysis

A set of numerical simulation experiments are conducted to evaluate performance of the proposed method. Four defined simulation models in (4.11) are examined, and the result are presented in Table 4.1. These experiments are also designed for $n=1000$ observations and $p = 25$ variables following the

Figure 4.1: Normalized coefficient score results for the simulation dataset for 100 trials.

is desired. Each number represents a specific identity number of a variable.
same data generating algorithm. Number of significant efficient variables designed in each simulation model is set \(m^* = \{5\}\) for each experiment. Each simulation is repeated 200 times independently. For each method, \(m\) represents average number of determined and selected efficient variables as an output of the proposed heuristic algorithm and \(\tilde{p}\) denotes performance of the method, for 200 trials, where

\[
\tilde{p} = \frac{m_{\text{method}}}{m_{\text{definition}}} = \frac{m}{m^*}
\]  

(4.13)

and \(\bar{p}\) is the grand average of performance of each method. The adjusted censoring rates of \(d_i\) for all numerical simulation experiments are 30 \(\pm\) 10%.

Table 4.1: Performance of the proposed methods based on four numerical simulation experiments with 200 replications.

<table>
<thead>
<tr>
<th>Method</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(m)</td>
<td>(p)</td>
<td>(m)</td>
<td>(p)</td>
<td>(m)</td>
</tr>
<tr>
<td>RC</td>
<td>4.2</td>
<td>0.84</td>
<td>4.4</td>
<td>0.88</td>
<td>4.1</td>
</tr>
<tr>
<td>Definition</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

The robustness of this method is verified by replicated simulation experiments leading to similar results. As an outcome of proposed methods, to reduce the number of variables in the dataset for further analysis, it is recommended to eliminate those variables with lower rank. In addition, to concentrate on a reduced number of variables, a category of highest ranked variables is suggested.
Next, the simulation experiment, results and analysis for R3 method are presented.

4.5 Simulation Experiment, Results and Analysis for 3R Method

In this section, R3 method is verified by simulation experiment where experimental result and analysis are also discussed.

4.5.1 Simulation Design and Experiment

To evaluate the performance and investigate the accuracy of the proposed methods, simulation experiments are developed. We set $n = 500$ observations, $p = 20$ variables. Three different simulation model are generated by pseudorandom algorithms as

$$u_i^1 \sim U (-1, 1), \quad u_i^2 \sim N (0, 1), \quad u_i^3 \sim Exp (1), \quad i = 1 \ldots p$$ (4.14)

by uniform, normal and exponential distributions respectively, where $U_j^i, j = 1 \ldots 3$ is explanatory variable matrix represent the original dataset $U$ for $j$th
model. In each simulation the coefficient components are set as

\[ \beta_{ii} = 10, \quad \beta_{jj} = 1, \quad \beta_{kk} = 0.1, \quad \beta_{ll} = 0, \]

\[ ii = 1 \ldots 5, \quad jj = 6 \ldots 10, \quad kk = 11 \ldots 15, \quad ll = 16 \ldots 20 \] (4.15)

Simulated event time vector \( Y \) is obtained from 2.17 and the censoring indicator is generated using a uniform distribution. To improve accuracy in calculating time-to-event vector \( T \), an error vector is considered as \( E \sim N(0, 0.1) \) in each simulation. In order to obtain \( k \) as an estimation of desired number of variables in any selected subset, principal component analysis (PCA) is used, which gives \( k = 3 \).

### 4.5.2 Result and Analysis

All three simulation models are applied in R3 method. A sample of numerical results of simulation experiment when data is generated by all three models 4.14 and applied in 3R method is depicted in Fig. 4.2 and Fig. 4.3. Each variable with larger score is more efficient and each variable with smaller score is candidate to be eliminated from the dataset if it is desired. Each number represents a specific identity number of a variable.

A set of numerical simulation experiments are conducted to evaluate performance of the proposed methods and the results are presented in Table 4.2. Three defined simulation models in 4.14 are examined. Setting of these
Figure 4.2: Normalized coefficient score results in 100 trials for the simulation dataset in Normal model.

models is as explained above; for \( n = 500 \) observations and \( p = 20 \) variables following same data generating algorithm. Number of significant efficient variables in each simulation model is set \( m^* = \{5\} \) for each experiment. Each simulation is repeated 200 times independently. For each method, \( m \) represents average number of determined and selected efficient variables and \( \tilde{p} \) denotes performance of the method, for 200 trials, where

\[
\tilde{p} = \frac{m_{\text{method}}}{m_{\text{definition}}} = \frac{m}{m^*} \tag{4.16}
\]
Figure 4.3: Normalized coefficient score results in 100 trials for the simulation dataset in Uniform model (top), Normal model (middle) and Exponential model (bottom).

and $\bar{p}$ is the grand average of performance of each method. The adjusted censoring rates for all numerical simulation experiments are $30 \pm 10\%$.

The robustness of these methods is verified by replicated simulation experiments leading to similar results. As an outcome of proposed methods, to reduce the number of variables in the dataset for further analysis, it is recommended to eliminate those variables with lower rank. In addition, to concentrate on a reduced number of variables, a category of highest ranked
Table 4.2: Performance of the proposed methods based on three numerical simulation experiments with 200 replications.

<table>
<thead>
<tr>
<th>Method</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>4.3</td>
<td>0.86</td>
<td>4.4</td>
<td>0.88</td>
</tr>
<tr>
<td>3R</td>
<td>4.0</td>
<td>0.80</td>
<td>4.2</td>
<td>0.84</td>
</tr>
<tr>
<td>Definition</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

variables is suggested.
Chapter 5

Methodology III: Hybrid Clustering and Classification Methods using Normalized Weighted K-Mean

5.1 Overview

In continuation of analyzing the variable selection methods in complex high-dimensional time-to-event data, a new approach is to cluster variables into subsets in terms of similarity in order to reduce the number of variables
through (a) substitute a representative for each subset, and (b) determine the most efficient or inefficient subsets beneficial to the process of variable reduction. An efficient variables is defined in previous methodologies [50].

For that goal, two hybrid methods are proposed under the assumption that the datasets includes uncensored data with no missing array.

The matrix $A = [a_{ij}]$ has $i$ rows and $j$ columns while the column vector $a = [a_j]$ has $i$ elements. Also $a_{ij}$ denotes the element of matrix $A$ in row $i$ and column $j$, and the $j$'th column of matrix $A$ is denoted by $a_j$. Additionally, a large matrix can be composed from smaller ones, such as $A = [B \ C]$.

The original dataset contains time-to-event vector with $n$ components, $t = [t_n]$ and may include any type of explanatory variable in matrix $U = [u_{np}]$ which includes $n$ independent observations and $p$ explanatory variables.

5.2 Methods and Algorithms for Variable Clustering

Two methods are presented in this section:
5.2.1 Clustering through Cost Function

The outline of this method is to obtain the best possible clustering for variables through a hybrid heuristic algorithm by using k-mean method. In this method the cumulative results from reasonable number of trials gives an optimum solution which clusters variables considering the efficiency on the event time.

An original given dataset \( D \) as an entry of the algorithms is shown in Table 5.1.

Table 5.1: High-dimensional time-to-event dataset

<table>
<thead>
<tr>
<th>Obs. #:</th>
<th>( t )</th>
<th>( u_1 )</th>
<th>( u_2 )</th>
<th>( u_3 )</th>
<th>( u_4 )</th>
<th>( u_5 )</th>
<th>( u_6 )</th>
<th>\cdots</th>
<th>( u_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( t_1 )</td>
<td>( u_{11} )</td>
<td>( u_{12} )</td>
<td>( u_{13} )</td>
<td>( u_{14} )</td>
<td>( u_{15} )</td>
<td>( u_{16} )</td>
<td>\cdots</td>
<td>( u_{1p} )</td>
</tr>
<tr>
<td>2</td>
<td>( t_2 )</td>
<td>( u_{21} )</td>
<td>( u_{22} )</td>
<td>( u_{23} )</td>
<td>( u_{24} )</td>
<td>( u_{25} )</td>
<td>( u_{26} )</td>
<td>\cdots</td>
<td>( u_{2p} )</td>
</tr>
<tr>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>( n )</td>
<td>( t_n )</td>
<td>( u_{n1} )</td>
<td>( u_{n2} )</td>
<td>( u_{n3} )</td>
<td>( u_{n4} )</td>
<td>( u_{n5} )</td>
<td>( u_{n6} )</td>
<td>\cdots</td>
<td>( u_{np} )</td>
</tr>
</tbody>
</table>

The dataset in matrix \( D \) is defined as below, where \( t = [t_n] \) is time-to-event vector (time vector) and \( U = [u_{ij}] \) is variable matrix:

\[
D = [t \ U] \quad (5.1)
\]

Also the variable matrix \( U \) with \( p \) vectors of \( n \) components is:

\[
U = [u_1 \ u_2 \ldots u_p] \quad (5.2)
\]
Procedure for each trial is defined as below:

Step 1: The variable matrix $U$ is normalized in the first step. Minimum and maximum value of each element in vector $u_j$ for $j = 1 \ldots p$ are obtained as:

$$
\begin{align*}
umin_j &= \min \{ u_{ij} \} \\
ump_j &= \max \{ u_{ij} \}
\end{align*}
$$

Then normalized elements $v_{ij}$ for all $u_{ij}$ are calculated:

$$
\begin{align*}
v_{ij} &= \frac{u_{ij} - umin_j}{umax_j - umin_j} & i = 1 \ldots n, j = 1 \ldots p
\end{align*}
$$

The result of the normalization is a $n$-by-$p$ matrix $V = [v_{np}]$ called normalized variable matrix. The normalized dataset is presented by:

$$
D^N = [t \ V] = [t \ v_1 \ v_2 \ldots \ v_p]
$$

Step 2: In the normalized matrix $V$, vectors $v_1$ to $v_p$ are clustered randomly into $q$ clusters containing $k$ variable vectors in each cluster where $k = \lceil p/q \rceil$.

For instance, a set of 12 variables is clustered into 4 clusters of 3 variable vectors. Each cluster including $k$ variable vectors which randomly assigned to a variable vectors of matrix $V$. Fig. 5.1 depicts the randomly clustering sample in this step.

Therefore, in general, each of cluster is represented by 5.6 where $k$ is number of vector per cluster.
Superscripts in vectors show the members in each cluster and subscript denotes a cluster indicator. For example $v_3^2$ is the second vector in the third cluster and matrix $W_4$ is the fourth cluster out of $q$ cluster. Consequently clustered dataset is presented in a new format:

$$D^C = [t \ W_1 \ W_2 \ldots \ W_q]$$

(5.7)

Step 3: By defining transformation function ($f$), each cluster matrix $W_j$ will be substituted by a variable vector $w_j$ where $i$'th element of this vector is resulted from applying transformation function on elements in $i$'th row in the subject matrix $W_j$ as below for $i = 1 \ldots n$:

$$w_{ij} = f(v_{ij}^1, v_{ij}^2, \ldots, v_{ij}^k) \quad j = 1 \ldots q$$

(5.8)

This transformation is shown in Fig. 5.2:
In the first level of calculation, transformation function is defined as an arithmetic mean where for \( i = 1 \ldots n \):

\[
w_{ij} = \frac{(v_{ij}^1 + v_{ij}^2 + \cdots + v_{ij}^k)}{k} \quad j = 1 \ldots q
\]  

(5.9)

For example, first element of the second transformed vector when \( k = 3 \) is

\[
w_{12} = \frac{(v_{12}^1 + v_{12}^2 + v_{12}^3)}{3}
\]

(5.10)

This process makes the transformed dataset in the form of

\[
D^R = [t \ W] = [t \ w_1 \ w_2 \ldots \ w_q]
\]

(5.11)

as a matrix which is constructed by \( q + 1 \) vectors.

Step 4: In this step k-mean clustering algorithm is applied by selecting parameter \( m \) as the desired number of the clusters. This process is performed
to cluster observation considering time vector \( t \) into \( m \) clusters. As a result, transformed dataset \( D^R \) is split into \( m \) clusters:

\[
C_i = [t_i \, w_{1}^i \, w_{2}^i \, \ldots \, w_{q}^i] \quad i = 1 \ldots m \quad (5.12)
\]

Where \( t_i \) is \( i \)’th cluster of time sub-vector and \( w_{j}^i \) is a subset of vector \( w_i \) allocated into the \( i \)’th cluster. Fig. 5.3 shows the concept of this step.

<table>
<thead>
<tr>
<th>Obs.#</th>
<th>( t )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>\ldots</th>
<th>( w_q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>1</td>
<td>( t_1 )</td>
<td>( w_{11} )</td>
<td>( w_{12} )</td>
<td>\ldots</td>
</tr>
<tr>
<td>2</td>
<td>( t_2 )</td>
<td>( w_{21} )</td>
<td>( w_{22} )</td>
<td>\ldots</td>
<td>( w_{2q} )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>3</td>
<td>( t_3 )</td>
<td>( w_{31} )</td>
<td>( w_{32} )</td>
<td>\ldots</td>
</tr>
<tr>
<td>4</td>
<td>( t_2 )</td>
<td>( w_{41} )</td>
<td>( w_{42} )</td>
<td>\ldots</td>
<td>( w_{4q} )</td>
</tr>
<tr>
<td>5</td>
<td>( t_5 )</td>
<td>( w_{51} )</td>
<td>( w_{52} )</td>
<td>\ldots</td>
<td>( w_{5q} )</td>
</tr>
<tr>
<td>6</td>
<td>( t_6 )</td>
<td>( w_{61} )</td>
<td>( w_{62} )</td>
<td>\ldots</td>
<td>( w_{6q} )</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>( n )</td>
<td>( t_n )</td>
<td>( w_{n1} )</td>
<td>( w_{n2} )</td>
<td>\ldots</td>
<td>( w_{nq} )</td>
</tr>
</tbody>
</table>

Figure 5.3: Sample of time-based clustering.

As an example, from Fig. 5.3, \( C_1 = [t_1 \, w_{1}^1 \, w_{2}^1 \, \ldots \, w_{q}^1] \) (Shaded cluster) where \( t_1 = [t_1 \, t_2]^T \), \( w_{1}^1 = [w_{11} \, w_{21}]^T \) and so on.

Step 5: A potential function \( (y) \) with respect to each of \( m \) clusters is defined over time data points. The objective of this function is to calculate the weight of time in each cluster. In the \( i \)’th cluster \( (C_i) \) this function is defined as below where \( t_i^j \) indicates the \( j \)’th time element in the \( i \)’th cluster when \( j \) is the number of time elements in each cluster.
\[ y_t = y(t_{1i}, t_{2i}, \ldots, t_{ji}) \]  

(5.13)

In the first level we define this function as an arithmetic mean of time data points in each cluster, equal to \( y_t \).

Step 6: Cost function over a trial is defined to calculate aggregation of potential functions output \( y \)'s for all clusters in a trial as a criterion to compare all trials. The objective of this definition is to determine variance in the output of the potential functions based on time-to-event. The formula for cost function is following:

\[ Cost = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} |y_i - y_j| \]  

(5.14)

For each trial, Steps 1 to 6 gives a cost result. Comparing all costs, the greatest amount introduces the best k-mean clustering which is a function of randomly clustered variables. Finally the variable clustering of mentioned trial is selected as the optimum clustering.

### 5.2.2 Clustering through Weight Function

The concept and objective of this method is similar to the Method A. The procedure of the Method B is following:

Step 1: The original dataset \( D \) includes time-to-event vector \( t = [t_n] \) and
variable matrix \( U = [u_{np}] \) (Equation 1). The time vector \( t \) and variable matrix \( U \) are normalized similar to Step 1 in Method A. The result is a normalized time vector \( t^* = [t^*_n] \) and a normalized variable matrix \( V = [v_{np}] \). The new dataset is:

\[
D^* = [t^* V] = [t^* \ v_1 \ v_2 \ldots \ v_p]
\]  

(5.15)

Step 2: Normalized dataset \( D^* \) is split into \( m \) bins based on one of following techniques in terms of vector \( t \):

1. K-M Estimator: Based on survival time

2. Equal Bin: Based on equal time intervals

3. Equal Frequency: Based on the frequency

Fig. 5.4 represents clustering schema.

<table>
<thead>
<tr>
<th>Obs.#</th>
<th>( t^* )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>\ldots</th>
<th>( v_p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_1 )</td>
<td>1 ( t^* )</td>
<td>( v_{11} )</td>
<td>( v_{12} )</td>
<td>( v_{13} )</td>
<td>( v_{14} )</td>
<td>\ldots</td>
<td>( v_{1p} )</td>
</tr>
<tr>
<td>2 ( t^* )</td>
<td>( v_{21} )</td>
<td>( v_{22} )</td>
<td>( v_{23} )</td>
<td>( v_{24} )</td>
<td>\ldots</td>
<td>( v_{2p} )</td>
<td></td>
</tr>
<tr>
<td>( B_2 )</td>
<td>3 ( t^* )</td>
<td>( v_{31} )</td>
<td>( v_{32} )</td>
<td>( v_{33} )</td>
<td>( v_{34} )</td>
<td>\ldots</td>
<td>( v_{3p} )</td>
</tr>
<tr>
<td>4 ( t^* )</td>
<td>( v_{41} )</td>
<td>( v_{42} )</td>
<td>( v_{43} )</td>
<td>( v_{44} )</td>
<td>\ldots</td>
<td>( v_{4p} )</td>
<td></td>
</tr>
<tr>
<td>5 ( t^* )</td>
<td>( v_{51} )</td>
<td>( v_{52} )</td>
<td>( v_{53} )</td>
<td>( v_{54} )</td>
<td>\ldots</td>
<td>( v_{5p} )</td>
<td></td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>( n ) ( t^* )</td>
<td>( v_{n1} )</td>
<td>( v_{n2} )</td>
<td>( v_{n3} )</td>
<td>( v_{n4} )</td>
<td>\ldots</td>
<td>( v_{np} )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.4: Schema of time-based binning.
Therefore,

\[ B_i = [t^*_i \ V_i] = [t^*_i \ v^1_i \ v^2_i \ \ldots \ v^p_i] \]

\[ i = 1 \ldots m \] (5.16)

Where \( t^*_i \) is of time sub-vector in \( i \)'th bin and \( v^j_i \) is a variable sub-vector of vector \( v_i \) allocated into the \( i \)'th bin. \( V_i \) is variable matrix including variable elements in \( i \)'th bin \( (B_i) \) which is a sub-matrix of \( V \). For instance, in the shaded bin in Fig. 5.4, \( t^*_2 = [t^*_3, \ t^*_4]^T \), \( v^1_2 = [v^1_{31}, \ v^1_{41}]^T \).

As an example, let’s assume a sample dataset of 9 observations \( (n = 9) \) including time-to-event vector \( t = [10, 20, 30, 30, 50, 60, 70, 70, 90]^T \) and K-M survival function \( s = [1.00, \ 0.95, \ 0.80, \ 0.65, \ 0.55, \ 0.30, \ 0.25, \ 0.15, \ 0.00]^T \) as calculated, with respect to observations respectively. The results of binning into 3 bins \( (m = 3) \) based on each of above techniques are presented in Table 5.2:

**Table 5.2: Sample of time-based binning based on three techniques.**

| Bin #1 | 1, 2, 3 | 1, 2, 3, 4 | 1, 2, 3 |
| Bin #2 | 4, 5    | 5, 6      | 4, 5, 6 |
| Bin #3 | 6, 7, 8, 9 | 7, 8, 9 | 7, 8, 9 |

Step 3: To calculate the weight of time in each cluster, weight function with
respect to each of \( m \) clusters is defined over time data points in each cluster.

This function in the first level is defined as an arithmetic mean of time data points in the cluster, \( z_i \). In Fig. 5.4, weight function of shaded bin \#2 (\( B_2 \)) including \( t_3^* \) and \( t_4^* \) is calculated as:

\[
z_2 = (t_3^* + t_4^*) / 2 \tag{5.17}
\]

Step 4: By multiplication each weight function result scalar \((z_i)\) in corresponding bin variable matrix \((V_i)\), weighted variable matrix \( V^* \) is generated.

\[
V^*_i = z_i V_i \quad i = 1 \ldots m \tag{5.18}
\]

\[
V^{*T} = [V^*_1 V^*_2 \ldots V^*_m] \tag{5.19}
\]

Step 5: Clustering algorithm k-mean applies on weighted variable matrix \( V^* \) in terms of variables to cluster variable vectors in subsets.

### 5.3 Experiment and Result

Coding methods A and B by MATLAB shows robustness by tuning in parameter definition such as \( k \) and \( m \).

A sample result for Method A applied on PBC dataset including 16 selected
variables and 111 uncensored observations are given in Table 5.3 for $k = 4$ and $m = 5$, for 100 trials:

<table>
<thead>
<tr>
<th>Cluster #</th>
<th>Variable #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 13, 15</td>
</tr>
<tr>
<td>2</td>
<td>3, 5, 7, 10</td>
</tr>
<tr>
<td>3</td>
<td>8, 9, 12, 14</td>
</tr>
<tr>
<td>4</td>
<td>4, 6, 11, 16</td>
</tr>
</tbody>
</table>

The result shows the four variable clusters obtained by cost function comparison.
Chapter 6

Conclusion

Chapter 6 presents a summary of research and an overview for further research steps.

6.1 Summary

Proper management and utilization of valuable data could significantly increase knowledge and reduce cost by preventive actions, whereas erroneous and misinterpreted data could lead to poor inference and decision-making. Variable selection is necessary in a decision-making application as unimportant variables adds unnecessary noise and often it is not clear which variables would be most useful and cost-effective to collect. Therefore data collection and analysis of data in data-driven decision-making plays crucial roles. In
many cases data are collected from different sources such as financial reports and marketing data, and they are combined for more informative decision-making. Consequently, determining effective explanatory variables, specifically in complex and high-dimensional data provides an excellent opportunity to increase efficiency and reduce costs. Unlike traditional datasets with few explanatory variables, analysis of datasets with high number of variables requires different approaches.

The proposed methods are beneficial to explanatory variable selection in decision-making process to obtain an appropriate subset of variables from among a high-dimensional and large-scale time-to-event data. By using such novel methods and reducing the number of variables, data analysis and decision-making processes will be faster, simpler and more accurate. For example, in business applications, many explanatory variables in a customer survey are defined based on cause and effect analysis process data or similar analytic process outcome. In most cases, correlations of these variables are complicated and unknown, and it is important to simply understand the efficiency of each variable in the survey. These procedures potentially are applicable solutions for many problems in a vast area of science and technology.
6.2 Next Steps

Next steps in this research are to examine the algorithms by exhaustive method for robustness and use evolutionary algorithms to optimize the solution.

By definition, heuristic algorithms find approximate solutions in acceptable time and computational space. These algorithms either give nearly the right answer or provide a solution not for all instances of the problem and usually are used for problems that cannot be easily solved. An efficient heuristic algorithm class is “evolutionary algorithms” which are methods that exploit ideas of biological evolution, such as reproduction, mutation and recombination, for searching the solution of an optimization problem. They apply the principle of survival on a set of potential solutions to produce gradual approximations to the optimum. Evolutionary techniques can differ in the details of implementation and the problems to which they are applied.

Also, the another challenge in this research is to face the data streaming circumstances in which variables are time-dependent where real-time analysis of data is more complicated.
This is just the beginning ...
Bibliography


Appendices
Appendix A

Matlab Code

clc; clear; close all; commandwindow;

%% ...
========================================================================
%% part setting
%% ...
========================================================================

%% part 01: sda (survival data analysis)
%% part 02: svd (singular value decomposition)
%% part 03: pca (principal component analysis)
%% part 04: cox phm (cox proportional hazards model)
%% part 05: mlr (multiple linear regression model)
%% part 06: k-means (clustering)
%% part 07: rf (random forest)
%% part 08: on and off comparison (doe approach)
%% part 09: test score - brute-force (exhaustive) search (sda ... combination)
%% part 10: test score - random subset selection (sda ... randomization)
%% part 11: time score - brute-force (exhaustive) search (sda ... combination)
%% part 12: time score - random subset selection (sda ... randomization)
%% part 13: bisection semi-greedy clustering algorithm ... (bisection method)
%% part 14: splitting semi-greedy clustering algorithm ... (splitting method)
part 15: weighted standardized score method
part 16: aft
part 17: penalty - cost function
part 18: hybrid algorithm
part 19: variable clustering and correlation
part 20: plots and charts
part 21: fuzzy time score - random subset selection
part 22: fuzzy weighted standardized score method
part 23: MIII1
part 24: MIII2
part no.: [01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 ...
17 18 19 20 21 22 23 24]
part_code = [ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0];

%% ...
========================================================================
data setting
========================================================================

% data 01: data generator 01
% data 02: data generator 02
% data 03: data generator 03
% data 04: pbc data
% data 05: pharynx data
% data 06: actg data
% data 07: new data
% data 08: gbcs data
% data 09: va data
% data 10: sample data
% data 11: simulation data generator 01
% data 12: simulation data generator 02
% data 13: simulation data generator 02
% data 14: sample data
% data no.: [01 02 03 04 05 06 07 08 09 10 11 12 13 14]
data_code = [ 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0];
% http://vincentarelbundock.github.io/Rdatasets/datasets.html

%% ...
========================================================================
censoring
========================================================================

% data censoring 01: uncensored
% data censoring 02: censored
% data cens no.: [01 02]
data_cens_code = [ 1 1];

%% explanatory variable data type
% data type 01: logical
% data type 02: normal
% data type no.: [01 02]
data_type_code = [ 1 1];

%% calculation method
% method 01: mid-range
% method 02: mean
% method 03: median
% method 04:
% method no.: [01 02 03 04]
method_code = [ 1 0 0 0];

%% ---------- data 01: data generator 01 ...
if data_code(01) == 1
    data = [];
    data_generated = rand(100,10);
    x_km_all = rand(418,17);
    data = [data,pbc(:,1:3),x_km_all];
end % if data_code(01) == 1

%% ---------- data 02: data generator 02 ...

if data_code(02) == 1

data = [];

n = 400;
lower = 1;
upper = 5;

data(1:n,1) = 1:n;
% data(1:n,2) = sortrows(fix(1000*rand(n,1)));
data(1:100,2) = sortrows(fix(100*rand(100,1)));
data(101:200,2) = sortrows(fix(100*rand(100,1)+200));
data(201:300,2) = sortrows(fix(100*rand(100,1)+400));
data(301:400,2) = sortrows(fix(100*rand(100,1)+600));

data(1:n,3) = 1;

data(1:n,4) = 0; % data(n,4) = 0.1;
data(1:n,5) = 1; % data(n,4) = 0.9;
data(1:n,6) = 2*data(1:n,1) + 2*rand(n,1); % a(ID) + e
data(1:n,7) = -3*data(1:n,1) + -3*rand(n,1); % a(ID)^2 + e
data(1:n,8) = 4*data(1:n,2) + 4*rand(n,1); % a(T) + e
data(1:n,9) = -5*data(1:n,2) + -5*rand(n,1); % a(T)^2 + e
data(1:n,10) = sign(mod((data(1:n,1)-1 - ...% 0 and 1 in order for ... mod(data(1:n,1)-1,5)),10)); % 0 and 1 in order for ... every 5 cons. obs.
data(1:n,11) = randn(n,1); % normal random N(0,1)
data(1:n,12) = 1 + 2.*randn(n,1); % normal random N(0,1)
data(1:n,13) = normrnd(0,1,n,1); % normal random N(0,1)
data(1:n,14) = randi([lower upper],n,1); % integer ... uniform random (1 to upper)
data(1:n,15) = lower + (upper-lower).*rand(n,1); % ... uniform random (lower to upper)
data(1:n,16) = rand(n,1); % uniform random (0 to 1)
data(1:n,17) = rand(n,1)<0.1; % 0 for 90% and 1 for 10%
data(1:n,18) = rand(n,1)<0.5; % 0 for 50% and 1 for 50%
data(1:n,19) = rand(n,1)<0.9; % 0 for 10% and 1 for 90%
data(1:n,20) = random('unif',1,10,n,1); % uniform random ... U(2)
data(1:n,21) = random('norm',2,3,n,1); % normal random ... N(2,3)
data(1:n,22) = random('exp',2,n,1); % exponential random ... E(2)
data(1:n,23) = random('wbl',2,3,n,1); % weibull random ... W(2,3)

% http://www.mathworks.com/help/stats/random.html
```matlab
end % if data_code(02) == 1

% ...
========================================================================
%% ---------- data 03: data generator 03 ...
========================================================================
%%
% ...

if data_code(03) == 1
  data = [];
  n = 300;
  lower = 1;
  upper = 5;
  data(1:n,1) = 1:300;
  % data(1:n,2) = sortrows(fix(1000*rand(n,1)));
  data(1:100,2) = sortrows(fix(100*rand(100,1)));
  data(101:200,2) = sortrows(fix(100*rand(100,1)+200));
  data(201:300,2) = sortrows(fix(100*rand(100,1)+400));
  data(1:n,3) = 1;
  data(1:n,4) = 0; % data(n,4) = 0.1;
  data(1:n,5) = 1; % data(n,4) = 0.9;
  data(1:n,6) = 2*data(1:n,1) + 2*rand(n,1); % a(ID) + e
  data(1:n,7) = -3*data(1:n,1) + -3*rand(n,1); % a(ID)^2 + e
  data(1:n,8) = 4*data(1:n,2) + 4*rand(n,1); % a(T) + e
  data(1:n,9) = -5*data(1:n,2) + -5*rand(n,1); % a(T)^2 + e
  data(1:n,10) = sign(mod((data(1:n,1)-1 - ...
    mod(data(1:n,1)-1,5)),10)); % 0 and 1 in order for ...
    every 5 cons. obs.
  data(1:n,11) = randn(n,1); % normal random N(0,1)
  data(1:n,12) = 1 + 2.*randn(n,1); % normal random N(0,1)
  data(1:n,13) = normrnd(0,1,n,1); % normal random N(0,1)
  data(1:n,14) = randi([lower upper],n,1); % integer ...
    uniform random (1 to upper)
  data(1:n,15) = lower + (upper-lower).*rand(n,1); % ...
    uniform random (lower to upper)
  data(1:n,16) = rand(n,1); % uniform random (0 to 1)
  data(1:n,17) = rand(n,1)≤0.1; % 0 for 90% and 1 for 10%
  data(1:n,18) = rand(n,1)≤0.5; % 0 for 50% and 1 for 50%
  data(1:n,19) = rand(n,1)≤0.9; % 0 for 10% and 1 for 90%
  data(1:n,20) = random('unif',1,10,n,1); % uniform random ...
    U(2)
  data(1:n,21) = random('norm',2,3,n,1); % normal random ...
    N(2,3)
  data(1:n,22) = random('exp',2,n,1); % exponential random ...
```

127
    data(1:n,23) = random('wbl',2,3,n,1); % weibull random ...
    W(2,3)

% http://www.mathworks.com/help/stats/random.html
end % if data_code(03) == 1

%% ...
========================================================================
%% ---------- data 04: pbc data ...
---------------------------------------------
%% ...
=========================================================================
%%
if data_code(04) == 1

load 'pbc';

data = pbc;

% N  Case number.
% X  The number of days between registration and the ... earlier of death, liver transplantation, or study ... analysis time in July, 1986.
% D  1 if X is time to death, 0 if time to censoring
% Z1  Treatment Code, 1 = D-penicillamine, 2 = placebo.
% Z2  Age in years. For the first 312 cases, age was ... calculated by dividing the number of days between birth and ... study registration by 365.
% Z3  Sex, 0 = male, 1 = female.
% Z4  Presence of ascites, 0 = no, 1 = yes.
% Z5  Presence of hepatomegaly, 0 = no, 1 = yes.
% Z6  Presence of spiders 0 = no, 1 = Yes.
% Z7  Presence of edema, 0 = no edema and no diuretic ... therapy for edema; 0.5 = edema present for which no ... diuretic therapy was given, or despite diuretic therapy
% Z8  Serum bilirubin, in mg/dl.
% Z9  Serum cholesterol, in mg/dl.
% Z10 Albumin, in gm/dl.
% Z11 Urine copper, in mg/day.
% Z12 Alkaline phosphatase, in U/liter.
% Z13 SGOT, in U/ml.
% Z14 Triglycerides, in mg/dl.
% Z15 Platelet count; coded value is number of ...
platelets per-cubic-milliliter of blood divided by 1000.
% Z16  Prothrombin time, in seconds.
% Z17  Histologic stage of disease, graded 1, 2, 3, or 4.

end % if data_code(04) == 1

%% ...
========================================================================
%% ---------- data 05: pharynx data ...
------------------------------------------
%% ...
========================================================================

if data_code(05) == 1

load 'pharynx';

data = pharynx;

% CASE   Case Number
% TIME   Survival time in days from day of diagnosis
% STATUS 0=censored, 1=dead
% INST   Participating Institution
% SEX    1=male, 2=female
% TX     Treatment: 1=standard, 2=test
% GRADE 1=well differentiated, 2=moderately ...
      differentiated, 3=poorly differentiated, 9=missing
% AGE   In years at time of diagnosis
% COND  Condition: 1=no disability, 2=restricted ...
      work, 3=requires assistance with self care, 4=bed ...
      confined, 9=missing
% SITE 1=facial arch, 2=tonsillar fossa, ...
      3=posterier pillar, 4=pharyngeal tongue, 5=posterier wall
% T_STAGE 1=primary tumor measuring 2 cm or less in ...
         largest diameter, 2=primary tumor measuring 2 cm to 4 ...
         cm in largest diameter with minimal infiltration in depth, 3=primary ...
% N_STAGE 0=no clinical evidence of node metastases, ...
         1=single positive node 3 cm or less in diameter, not ...
         fixed, 2=single positive
         3=multiplie positive nodes or fixed positive nodes
% ENTRY_DT Date of study entry: Day of year and year, dddyy

end % if data_code(05) == 1

%% ...
========================================================================
%% ---------- data 06: actg data ...

---

if data_code(06) == 1
  load 'actg';
  data = actg;
end % if data_code(06) == 1

%% ...  

%% ---------- data 07: new data ...

---

if data_code(07) == 1
  load 'new';
  data = gbcs;
end % if data_code(07) == 1

%% ...  

%% ---------- data 08: gbcs data ...

---

if data_code(08) == 1
  load 'gbcs';
  data = gbcs;
end % if data_code(08) == 1

%% ...  

%% ---------- data 09: va data ...

---

%% ...  

---
%% if data_code(07) == 1
load 'va';
data = va;
end % if data_code(09) == 1

% ...
========================================================================
%% ---------- data 10: sample data ...
----------------------------------------
%% ...
========================================================================
%% if data_code(10) == 1
load 'sample01';
load 'sample02';
load 'sample03';
load 'sample04';
load 'sample05';
data = sample05;
end % if data_code(10) == 1

%% ---------- data 11: simulation data generator 01 ...
----------------------
%% ...
========================================================================
%% if data_code(11) == 1
data = [];
n = 1000;
lower = 1;
upper = 5;
data(1:n,1) = 1:1000;
% data(1:n,2) = sortrows(fix(1000*rand(n,1)));
data(1:n,3) = 1;
data(1:n,4) = normrnd(0,1,n,1) %+ data(1:n,1); % normal ...
data(1:n,5) = normrnd(0,1,n,1) %+ data(1:n,1); % normal ...
data(1:n,6) = normrnd(0,1,n,1) %+ data(1:n,1); % normal random N(0,1)
data(1:n,7) = normrnd(0,1,n,1) %+ data(1:n,1); % normal random N(0,1)
data(1:n,8) = normrnd(0,1,n,1) %+ data(1:n,1); % normal random N(0,1)
data(1:n,9) = normrnd(0,1,n,1) % normal random N(0,1)
data(1:n,10) = normrnd(0,1,n,1) % normal random N(0,1)
data(1:n,11) = normrnd(0,1,n,1) % normal random N(0,1)
data(1:n,12) = normrnd(0,1,n,1) % normal random N(0,1)
data(1:n,13) = normrnd(0,1,n,1) % normal random N(0,1)
data(1:n,14) = rand(n,1); % uniform random (0 to 1)
data(1:n,15) = rand(n,1); % uniform random (0 to 1)
data(1:n,16) = rand(n,1); % uniform random (0 to 1)
data(1:n,17) = rand(n,1); % uniform random (0 to 1)
data(1:n,18) = rand(n,1); % uniform random (0 to 1)
data(1:n,19) = random('exp',1,n,1); % exp random (1)
data(1:n,20) = random('exp',1,n,1); % exp random (1)
data(1:n,21) = random('exp',1,n,1); % exp random (1)
data(1:n,22) = random('exp',1,n,1); % exp random (1)
data(1:n,23) = random('exp',1,n,1); % exp random (1)
data(1:n,24) = random('wbl',2,1,n,1); % exp random (2)
data(1:n,25) = random('wbl',2,1,n,1); % exp random (2)
data(1:n,26) = random('wbl',2,1,n,1); % exp random (2)
data(1:n,27) = random('wbl',2,1,n,1); % exp random (2)
data(1:n,28) = random('wbl',2,1,n,1); % exp random (2)

for i=1:n
    beta1x = sum(data(i,4:8));
    beta2x = sum(data(i,9:13));
    beta3x = sum(data(i,14:18));
    beta4x = sum(data(i,19:28));
    data(i,2) = 10*beta1x + 1*beta2x + 0.1*beta3x + ... 0*beta4x;
end

% http://www.mathworks.com/help/stats/random.html

end % if data_code(11) == 1

%%% ... 
%-------- data 12: simulation data generator 02 ...
%--------
% ... 
%--------
% if data_code(12) == 1
data = [];  
n = 1000;  
lower = 1;  
upper = 5;  
data(1:n,1) = 1:1000;  
for i = 4:23  
    data(1:n,i) = normrnd(0,1,n,1);  
end  
for i = 1:n  
    beta1x = sum(data(i,4:8));  
    beta2x = sum(data(i,9:13));  
    beta3x = sum(data(i,14:18));  
    beta4x = sum(data(i,19:23));  
    data(i,2) = 10*beta1x + 1*beta2x + 0.1*beta3x + ...  
    0*beta4x;  
end  

% http://www.mathworks.com/help/stats/random.html  
end  

%% ...  
end % if data_code(12) == 1  

% %  
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% % ------- data 13: simulation data generator 03 ...  
% %-----------------------------------  
% % ...  
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% if data_code(13) == 1  
end % if data_code(13) == 1  

% % ...  

133
if data_code(14) == 1
data = [];
data_generated = rand(100,10);
x_km_all = rand(418,17);
data = [data,pbc(:,1:3),x_km_all];
end % if data_code(14) == 1

% general setting

% defining uncensored dataset matrix
data_unc = data; % defining uncensored dataset matrix
data_unc(find(data_unc(:,3)==0),:) = []; % remove censored data
data_unc = sortrows(data_unc,2); % sort by failure time

% defining censored dataset matrix
data_cens = data; % defining censored dataset matrix
data_cens = sortrows(data_cens,2); % sort by failure time

if data_cens_code(01) == 1
    [n p] = size(data_unc);
data_cal = data_unc; % defining calculate dataset matrix
end

if data_cens_code(02) == 1
    [n p] = size(data_cens);
data_cal = data_cens; % defining calculate dataset matrix
end

data_cal(n+1,:) = min(data_cal(1:n,:)); % calculate variable min
data_cal(n+2,:) = max(data_cal(1:n,:)); % calculate variable max

if method_code(01) == 1; data_cal(n+3,:) = (data_cal(n+1,:) + data_cal(n+2,:))/2; end % calculate variable "mid-range"
if method_code(02) == 1; data_cal(n+3,:) = ... mean(data_cal(1:n,:)); end; % calculate variable "mean"
if method_code(03) == 1; data_cal(n+3,:) = ...
    median(data_cal(1:n,:)); end; % calculate variable "median"
data_raw = data_cal(1:n,4:p); % defining raw dataset matrix
data_nor = zscore(data_raw); % defining normalized dataset ...
% logical vector
data_log = data_cal; % defining logical dataset matrix
data_fuz = data_cal; % defining logical dataset matrix

for i = 1:n
    for j = 4:p
        if data_log(i,j) < data_log(n+3,j);
            data_log(i,j) = 0;
        else
            data_log(i,j) = 1;
        end
    end
end

for i = 1:n
    for j = 4:p
        if data_fuz(n+1,j)== data_fuz(n+2,j);
            data_fuz(i,j) = data_fuz(n+1,j);
        else
            data_fuz(i,j) = ...
            (data_fuz(i,j)-data_fuz(n+1,j))/(data_fuz(n+2,j)-data_fuz(n+1,j));
        end
    end
end

if data_type_code(01) == 1
    data_time = data_log(1:n,2); % defining failure time vector
    data_var = data_log(1:n,4:p); % defining logical ...
    variable dataset matrix
end

if data_type_code(02) == 1
    data_time = data_fuz(1:n,2); % defining failure time vector
    % data_var = data_fuz(1:n,4:p); % defining logical ...
    variable dataset matrix
    data_fuzzy = data_fuz(1:n,4:p); % defining logical ...
    variable dataset matrix
end

% extra change for var 1 (switching) % data_var(:,1) = ...
    sign(data_var(:,1)-1);
var(p-3) = ...
    struct('all',[],'one',[],'zero',[],'failure',[],'surv',[]);
for i = 1:p-3
    [var(i).all(:,1)] = data_log(1:n,2); % time
    [var(i).all(:,2)] = data_log(1:n,3); % status
    [var(i).all(:,3)] = data_log(1:n,i+3); % variable (0 or 1)
    [var(i).one(:,1)] = data_log(1:n,2); % time
    [var(i).one(:,2)] = data_log(1:n,3); % status
    [var(i).one(:,3)] = data_log(1:n,i+3); % variable (only 1)
    var(i).one(find([var(i).one(1:end-1,3)]==0,:),:) = []; % ... compeleting var(i).one (eliminating 0s)
    var_one_sum(i,1) = sum(var(i).one(:,1)); % cumulative time
    var_one_sum(i,2) = sum(var(i).one(:,3)); % cumulative ones
    [var(i).zero(:,1)] = data_log(1:n,2); % time
    [var(i).zero(:,2)] = data_log(1:n,3); % status
    [var(i).zero(:,3)] = data_log(1:n,i+3); % variable (only 0)
    var(i).zero(find([var(i).zero(1:end-1,3)]==1,:),:) = []; % ... compeleting var(i).zero (eliminating 1s)
    var_zero_sum(i,1) = sum(var(i).zero(:,1)); % cumulative time
    var_zero_sum(i,2) = sum(var(i).zero(:,3)); % cumulative zeros

    % [var(i).failure(:,1),var(i).failure(:,2)] = ...
    % ecdf(var(i).one(:,1));
    [var(i).failure(:,1),var(i).failure(:,2)] = ...
    % ecdf(var(i).one(:,1),'censoring',var(i).one(:,2));
    var(i).surv(:,1) = 1 - var(i).failure(:,1); % survival ... function for single variable #i
end

% data_var_num = data_var;
data_var_num = data_fuzzy;
for i = 1:p-3
    data_var_num(n+1,i) = i;
end

% model validation - Comparison of covariate correlations ... in the original and the transformed dataset
corrcoef_raw = corrcoef(data_raw); % ...
                        +++++++++++++++++++++++++++++++++++++++++++++++
corrcoef_var = corrcoef(data_var); % ...
                        +++++++++++++++++++++++++++++++++++++++++++++++
corrplot(data_raw) % ...
for i = 1:p-3
    scatter(corrcoef_raw(i,i+1:end),corrcoef_var(i,i+1:end),'k');
    x_cor = [x_cor, corrcoef_raw(i,i+1:end)];
    y_cor = [y_cor, corrcoef_var(i,i+1:end)];
    hold on; grid on;
end
[p_cor,s_cor] = polyfit(x_cor,y_cor,1); % poly fit
z_cor = polyval(p_cor,x_cor);
plot(x_cor,z_cor,'--','Color',[0.5 0.5 0.5]);
hold on; grid on;
xlabel('original dataset','FontSize',15); ... 
ylabel('transformed dataset','FontSize',15); hold on
axis([-0.6 0.6 -0.6 0.6]); axis square;

if part_code(01) == 1
    % ...
The Wilcoxon rank sum test is equivalent to the Mann-Whitney U test.

% [P,H] = ranksum(...) returns the result of the hypothesis test, performed at the 0.05 significance level, in H.
% H=0 indicates that the null hypothesis ("medians are equal") cannot be rejected at the 5% level.
% H=1 indicates that the null hypothesis can be rejected ... at the 5% level.

y = data_unc(:,2)'; % or data_time'
y_data = data_cens(:,2)'; % or data_time'
freqy = data_unc(:,3)';
censy = data_cens(:,3)';
freqy = censy;
[f,x] = ecdf(y,'frequency',freqy);
[f_data,x_data] = ...
    ecdf(y_data,'frequency',freqy,'function','survivor');
[f_data,x_data] = ...
    ecdf(y_data,'censoring',censy,'function','survivor');
data_surv = f_data;
data_time = f_data;

figure;
ecdf(y_data,'frequency',freqy,'function','survivor'); ... % kaplan-meier plot
ecdf(y_data,'censoring',censy,'function','survivor'); % ...
kaplan-meier plot
hold on; grid on;
title('kaplan-meier survival function','FontSize',15);
xlabel('time,t','FontSize',15); ylabel('survival ... probability,s(t)','FontSize',15);
plot(x,s,'color','r'); hold on; grid on; % survival plot
h = get(gca,'children');
set(h,'color','k');

% ...

% uncensored data
y_unc = data_unc(:,2)'; % or data_time'
freqy_unc = data_cens(:,3)';
[f_unc,x_unc] = ...
    ecdf(y_data,'frequency',freqy_unc,'function','survivor');
data_surv = f_unc;

138
% figure;
edcf(y_data,'frequency',freqy_unc,'function','survivor'); ...  
% kaplan-meier plot
hold on; grid on;
title('kaplan-meier survival function ...  
(uncensored)', 'FontSize', 15);
xlabel('time,t', 'FontSize', 15); ylabel('survival ...  
probability,s(t)', 'FontSize', 15);
plot(x, s, 'color', 'r'); hold on; grid on; % survival plot
h = get(gca, 'children');
set(h, 'color', 'g');
% ...

..............................................................

[p_data, s_data] = polyfit(x_data, data_surv, 1); % poly fit
z = polyval(p_data, x_data);
plot(x, s, 'o', x, z, '-'); hold on; grid on;
plot(x_data, z, '-', 'Color', 'm');
hold on; grid on;
axis([0 data_cal(n,2) 0 1]);
slop = [];
figure;
for i = 1:p-3
    yi = var(i).one(:,1)'; % time
censyi = var(i).one(:,2)'; % status
    [fi, xi] = ...  
        ecdf(yi, 'censoring', censyi, 'function', 'survivor');
data_surv_i = fi; % survival function for single ...  
variable #1 equal to var(i).surv(:,1)

    ecdf(yi, 'censoring', censyi, 'function', 'survivor'); % ...  
        kaplan-meier plot
    hold on; grid on;
title('kaplan-meier survival function', 'FontSize', 15);
xlabel('time,t', 'FontSize', 15); ylabel('survival ...  
probability,s(t)', 'FontSize', 15); hold on;
% plot(x, data_surv, 'Color', 'k'); hold on; % survival ...  
plot
[pi, si] = polyfit(xi, data_surv_i, 1); % poly fit
z = polyval(pi, xi);
slop(i,1) = p_data(1,1) - pi(1,1);
plot(xi, z, '-', 'Color', 'g');
hold on; grid on;
% [p_rank(i), h_rank(i)] = ...  
ranksum(data_time, var(i).one(:,1)); % wilcoxon ...  
rank sum test
[p_rank(i),h_rank(i)] = ...
    ranksum(data_surv,data_surv_i); % wilcoxon rank ...
end

axis([0 data_cal(n,2) 0 1]);
xlabel('time,t','FontSize',15); ylabel('survival probability,s(t)','FontSize',15);

figure;
bar(x_axis,h_rank); % 0 or 1
hold on; grid off;
title('preliminary variable importance - wilcoxon rank ...
    test','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('importance','FontSize',15); hold on;

figure;
bar(x_axis,1-p_rank,'g'); % 0 to 1
hold on; grid on;
title('preliminary variable importance - wilcoxon rank ...
    test','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('importance','FontSize',15); hold on;

figure;
bar(x_axis,slop,'r'); % plot(slop,'r')
hold on; grid on;
title('slop of the survival function','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('slop','FontSize',15); hold on;

end % if part_code(01) == 1

%% ...
%-------------------- part 02: svd (singular value decomposition) ...
---------------------

if part_code(02) == 1

% [U,S,V] = svd(X) produces a diagonal matrix S of the ... same dimension as X, with nonnegative diagonal ... elements in decreasing order, and unitary matrices U ... and V so that X = U*S*V'.
% [U,S,V] = svd(X,0) produces the "economy size" ...
decomposition. If X is m-by-n with m > n, then svd ...
computes only the first n columns of U and S is n-by-n.
% [U,S,V] = svd(X,'econ') also produces the "economy ...
size" decomposition. If X is m-by-n with m ≥ n, it is ... equivalent to svd(X,0). For m < n, only the first m ...
columns of V are computed and S is m-by-m.
% http://www.mathworks.com/help/matlab/ref/svd.html

[U1,S1,V1] = svd(data_raw); % svd : data_raw = U*S*V'
[U2,S2,V2] = svd(data_var); % svd : data_var = U*S*V'

figure;
bar(x_axis,diag(S1));
hold on; grid on;
title('singular value decomposition','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('eigenvalue','FontSize',15);
set(gca,'FontSize',15);

figure;
pareto(diag(S1));
hold on; grid on;
title('singular value decomposition','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('eigenvalue','FontSize',15);

figure;
bar(x_axis,diag(S2));
hold on; grid on;
title('singular value decomposition','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('eigenvalue','FontSize',15);

figure;
pareto(diag(S2));
hold on; grid on;
title('singular value decomposition','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('eigenvalue','FontSize',15);

figure;
bar(x_axis,[diag(S1) diag(S2)],'stack');
hold on; grid on;
title('singular value decomposition raw - ... log','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
2','FontSize',15); hold on;
legend('criterion 1','criterion 2');
set(gca,'FontSize',15);
figure;
bar(x_axis,1:p-3,[diag(S1) diag(S2)],0.5,'grouped');
hold on; grid on;
title('singular value decomposition - raw - ...
log','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
2','FontSize',15); hold on;
legend('criterion 1','criterion 2');
set(gca,'FontSize',15);

U3 = U1; U3(:,3:111)=0;
S3 = S1; S3(:,3:17)=0;
V3 = V1; V3(3:17,:)=0;
DATA_SVD = U3*S3*V3';

eend % if part_code(2) == 1
%% ...
========================================================================
%% ---------- part 03: pca (principal component analysis) ... 
-----------------
%% ...
========================================================================

if part_code(03) == 1

% [coeff, score, latent, tsquared, explained, mu] = ... 
 pca(X,Name,Value) returns the principal component ... 
 coefficients for the n-by-p data matrix X. Rows of X ... 
 correspond to observations and columns correspond to ... 
 variables. 
% The coefficient matrix is p-by-p. Each column of coeff ... 
 contains coefficients for one principal component, and ... 
 the columns are in descending order of component ... 
 variance. By default, pca centers the data and uses ... 
 the singular value decomposition algorithm. 
% Also returns any of the output arguments in the ... 
 previous syntaxes using additional options for ... 
 computation and handling of special data ... 
 types, specified by one or more Name, Value pair arguments, 
% and returns the principal component scores in score ... 
 and the principal component variances in latent. ... 
 Principal component scores are the representations of ... 
 X in the principal component space. Rows of score ... 
 correspond to observations, and columns correspond to ... 
 components. 
% The principal component variances are the eigenvalues ... 
 of the covariance matrix of X.
In addition returns the Hotelling's T-squared statistic for each observation in X and explained, the percentage of the total variance explained by each principal component and μ, the estimated mean of each variable in X.


```matlab
[coeff_pcacov_raw,latent_pcacov_raw] = pcacov(data_raw);
[coeff_pcacov_var,latent_pcacov_var] = pcacov(data_var);

for i = 1:2
    figure;
    if i == 1;
        % eigenvalues_pcacov = latent_pcacov_raw; h = ...
        title('scree plot - pcacov - raw'); hold on;
    end
    else
        eigenvalues_pcacov = latent_pcacov_var; h = ...
        title(''); hold on;
    end
    set(h,'Color','k','FontSize',15); hold on;
    plot(eigenvalues_pcacov,'o-','color','k'); hold on;
    % title('scree plot','FontSize',15);
    xlabel('component number','FontSize',15); ...
    ylabel('eigenvalue','FontSize',15);
    % figure;
    % bar(x_axis,eigenvalues); hold on; grid on;
    % xlabel('component number','FontSize',15); ...
    ylabel('eigenvalue','FontSize',15);
end

[coeff_pca_raw,score_pca_raw,latent_pca_raw] = ... 
 pca(data_raw,'algorithm','svd');
[coeff_pca_var,score_pca_var,latent_pca_var] = ... 
 pca(data_var,'algorithm','svd');

for i = 1:2
    figure;
    if i == 1;
        eigenvalues_pca = latent_pca_raw; h = ...
        title('scree plot - pca - raw'); hold on;
    else
        eigenvalues_pca = latent_pca_var; h = ...
        title('scree plot - pca - var'); hold on;
    end
    set(h,'Color','k','FontSize',15); hold on;
    plot(eigenvalues_pca,'o-','color','m');
end
```
hold on; grid on;
% title('scree plot - ',title,'FontSize',15);
xlabel('component number','FontSize',15); ...
ylabel('eigenvalue','FontSize',15);

% figure;
% bar(x_axis,eigenvalues_pca); hold on; grid on;
% xlabel('component number','FontSize',15); ... 
ylabel('eigenvalue','FontSize',15);
end

cov = cov(data_var);
[VV,DD] = eigs(cov);

figure;
bar(coeff_pcacov_raw(:,1:2),'grouped');
hold on; grid on;
title('coeff - pcacov - raw - comp 1 and 2','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coeff','FontSize',15); hold on;
legend('comp 1','comp 2');
set(gca,'FontSize',15);

figure;
bar(coeff_pcacov_var(:,1:2),'grouped');
hold on; grid on;
title('coeff - pcacov - log - comp 1 and 2','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coeff','FontSize',15); hold on;
legend('comp 1','comp 2');
set(gca,'FontSize',15);

figure;
bar(coeff_pca_raw(:,1:2),'grouped');
hold on; grid on;
title('coeff - pca - raw - comp 1 and 2','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coeff','FontSize',15); hold on;
legend('comp 1','comp 2');
set(gca,'FontSize',15);

figure;
bar(coeff_pca_var(:,1:2),'grouped');
hold on; grid on;
title('coeff - pca - log - comp 1 and 2','FontSize',15); ...
hold on;
figure;
bar(1:p-3,[coeff_pca_raw(:,1) 
    coeff_pca_var(:,1)],0.5,'grouped');
hold on; grid on;
title('singular value decomposition - raw - log - comp ... 
1','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coeff','FontSize',15); hold on;
legend('raw','log');
set(gca,'FontSize',15);

figure;
bar(1:p-3,[coeff_pca_raw(:,2) 
    coeff_pca_var(:,2)],0.5,'grouped');
hold on; grid on;
title('singular value decomposition - raw - log - comp ... 
2','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coeff','FontSize',15); hold on
legend('raw','log');
set(gca,'FontSize',15)

end % if part_code(03) == 1

%% ...
========================================================================
%% ---------- part 04: cox phm (cox proportional hazards ... 
model) ----------
%% ...
========================================================================

if part_code(04) == 1

% [b,logl,H,stats] = coxphfit(X,T,Name,Value) returns a ... 
p-by-1 vector,b,of coefficient estimates for a Cox ... 
proportional hazards regression of the observed ... 
responses in an n-by-1 vector,T,on the predictors in ... 
an n-by-p matrix X.
% The model does not include a constant term,and X ... 
cannot contain a column of 1s.
% Also returns a vector of coefficient estimates,with ... 
additional options specified by one or more ... 
Name,Value pair arguments
% and returns the loglikelihood,logl,a ...
structure, stats, that contains additional ...
statistics, and a two-column matrix, H, that contains ...
the T values in the first column and the estimated ...
baseline cumulative hazard, in the second column. You ...
can use any of the input arguments in the previous ...
syntaxes.

\[
[b1, \text{logL1}, H1, \text{st1}] = \text{coxphfit}(\text{data_raw}, \text{data_time});
\]

figure;
stairs(H1(:,1), \exp(-H1(:,2)),'linestyle','-', 'color','k');
hold on; grid on;
\% xx = linspace(0,100);

\[
[b2, \text{logL2}, H2, \text{st2}] = \text{coxphfit}(\text{data_nor}, \text{data_time});
\]
\% figure;
stairs(H2(:,1), \exp(-H2(:,2)),'linestyle','--', 'color','g');
hold on; grid on;
\% xx = linspace(0,100);

\[
[b3, \text{logL3}, H3, \text{st3}] = \text{coxphfit}(\text{data_var}, \text{data_time});
\]
\% figure;
stairs(H3(:,1), \exp(-H3(:,2)),'linestyle',':', 'color','r');
hold on; grid on;
\% xx = linspace(0,100);
\%
title(sprintf('Baseline survivor function for ...
X=%g', mean(x)));

b4 = abs(b1); \% absolute values of cox phm coefficients ...
- raw data
b5 = abs(b3); \% absolute values of cox phm coefficients ...
- logical data
edcdf(y_data, 'censoring', censy, 'function', 'survivor', 'bounds', 'on'); ...
hold on; \% kaplan-meier plot (blue graph)
title('cox phm baseline survival ...
  function', 'FontSize', 15); hold on;
xlabel('time', 'FontSize', 15); ylabel('survival ...
  probability, s(t)', 'FontSize', 15); hold on;
legend('raw data', 'standardized data', 'logical ...
  data', 'km', 'Location', 'NorthEastOutside'); hold on;
figure;
edcdf(y_data, 'censoring', censy, 'function', 'survivor', 'bounds', 'off'); ...
hold on; \% kaplan-meier plot
stairs(H1(:,1), \exp(-H1(:,2)),'linestyle','--', 'color','k'); ...
hold on; grid on;
stairs(H3(:,1),exp(-H3(:,2)),'linestyle',':','color','r'); ...
hold on; grid on;
xx = linspace(0,100);
xlabel('time','FontSize',15); ylabel('survival probability,s(t)','FontSize',15); hold on;
% title('cox phm baseline survival function','FontSize',15); hold on
[p_h1h3,h_h1h3] = ranksum(H1(:,2),H3(:,2)); % wilcoxon ...
rank sum test
logrank_h1h3 = logrank(H1(:,2),H3(:,2)); % log-rank test
figure;
bar(x_axis,h_rank,'y'); hold on; grid on;
title('slope of the survival function','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ... 
ylabel('slope','FontSize',15); hold on;
bar(x_axis,b2); hold on; grid on;
title('slope of the survival function','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ... 
ylabel('slope','FontSize',15); hold on;
figure;
data_surv(1) = subplot(3,1,1);
plot(b1,'k'); hold on; grid on; % row data 
title('cox phm coefficients','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coefficient','FontSize',15); hold on;
data_surv(2) = subplot(3,1,2);
plot(b2,'b'); hold on; grid on; % standardized data 
hold on; grid on;
title('cox phm coefficients','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coefficient','FontSize',15); hold on;
data_surv(3) = subplot(3,1,3);
plot(b3,'r'); hold on; grid on; % logical data 
title('cox phm coefficients','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('coefficient','FontSize',15); hold on;
figure;
plot(b1,'linestyle','-','color','k'); hold on; grid on; ... 
% row data 
plot(b2,'linestyle','--','color','b'); hold on; grid on; ... 
% standardized data 
plot(b3,'linestyle','-.','color','r'); hold on; grid on; ...
% logical data
plot(b4,'linestyle','-.','color','g'); hold on; grid on; ...
% abs logical data
title('cox phm coefficients','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('coefficient','FontSize',15); hold on;
legend('raw','standardized','logical','logical absolute','Location','NorthEastOutside'); hold on;
% plot(b2,'--','color',[0.1 0.1 0.1])
% set(gca,'FontSize',10); hold on

figure;
plot(b1); hold on; grid on; % row data
title('cox phm coefficients - raw data','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('coefficient','FontSize',15); hold on;

figure;
plot(b4,'r'); hold on; grid on; % row data
title('cox phm coefficients - raw data (absolute value)','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('coefficient','FontSize',15); hold on;

end % if part_code(04) == 1

%% ...
========================================================================
%% ---------- part 05: mlr (multiple linear regression model) ----------
%% ...
========================================================================
%% if part_code(05) == 1

% [b,bint,r,rint,stats] = regress(s,x) multiple linear ...
% regression using least squares
% http://www.mathworks.com/help/stats/regress.html
% lm = fitlm(s,x) linear model
% rlm = fitlm(s,x,'linear','RobustOpts','on') robust ...
% linear model
% mdl = LinearModel.fit(X,y) creates a linear model of ...
% the responses y to a data matrix X

% http://www.mathworks.com/help/stats/linearmodel.fit.html
% ...
% [b,bint,r,rint,stats] = regress(data_unc(1:n,2),data_unc(1:n,4:p));
[b,bint,r,rint,stats] = regress(data_time,data_var);
b_fu = regress(data_time,data_fuzzy);

% lm_unc = fitlm(data_unc(1:n,4:p),data_unc(1:n,2));
% rlm_unc = ...
    fitlm(data_unc(1:n,4:p),data_unc(1:n,2),'linear','RobustOpts','on');

lm_unc = fitlm(data_cens(1:n,4:p),data_cens(1:n,2));
rlm_unc = ...
    fitlm(data_cens(1:n,4:p),data_cens(1:n,2),'linear','RobustOpts','on');
lm_var = fitlm(data_var,data_log(1:n,2));

figure;
bar(x_axis,abs(b),'k'); hold on; grid on; % plot(b); ...
    hold on; grid on;
title('multiple linear regression model','FontSize',15); ...
    hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;

figure;
bar(x_axis,abs(b_fu),'b'); hold on; grid on; % plot(b); ...
    hold on; grid on;
title('fuzzy multiple linear regression ... model','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;

% figure;
% plot(lm_unc); hold on; grid on;
% figure;
% plot(lm_var); hold on; grid on;

figure;
plot((b),'k');
hold on; grid on;
title('cumulation of all variable scores - ... randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;

plot((lm_unc.Coefficients(2:end,1)),'b');
hold on; grid on;
title('cumulation of all variable scores - ... randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...

149
ylabel('y','FontSize',15); hold on;
plot((rlm_unc.Coefficients(2:end,1)),'r');
hold on; grid on;
title('cumulation of all variable scores - randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('y','FontSize',15); hold on;

plot((lm_var.Coefficients(2:end,1)),'m');
hold on; grid on;
title('cumulation of all variable scores - randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ... 
ylabel('y','FontSize',15); hold on;

end % if part_code(05) == 1

%% ...
========================================================================
%% ---------- part 06: k-means (clustering) ...
-------------------------------
========================================================================
%% ...
========================================================================
%
if part_code(06) == 1

% [IDX,C,sumd,D] = kmeans(X,k) partitions the points in ...
% the n-by-p data matrix X into k clusters,returns the ...
% k cluster centroid locations in the k-by-p matrix C,
% returns the within-cluster sums of point-to-centroid ...
% distances in the 1-by-k vector sumd and distances ...
% from each point to every centroid in the n-by-k ...
% matrix D
% This iterative partitioning minimizes the sum,over all ...
% clusters,of the within-cluster sums of ...
% point-to-cluster-centroid distances. Rows of X ... 
correspond to points,columns correspond to variables.
% kmeans returns an n-by-1 vector IDX containing the ... 
% cluster indices of each point. By default,kmeans uses ... 
% squared Euclidean distances. When X is a ... 
% vector,kmeans treats it as an n-by-1 data ... 
% matrix,regardless of its orientation.
% http://www.mathworks.com/help/stats/kmeans.html
k = 3;
opts = statset('Display','final');
[cidx_raw,ctrs_log] = ...
    kmeans(data_raw,fix((p-3)/k),'Distance','cityblock','Replicates',5,'Options',opts);

opts = statset('Display','final');
[cidx_log,ctrs_log] = ...
    kmeans(data_var,fix((p-3)/k),'Distance','cityblock','Replicates',5,'Options',opts);

figure;
% ...
    plot(data_var(cidx_log==1,1),data_var(cidx_log==1,2),'r.',data_var(cidx_log==2,1),data_var(cidx_log==2,2),'b.',ctrs_log(:,1),ctrs_log(:,2),'kx');
    scatter(10*(cidx_raw-1)+(1:n)',10*(cidx_log-1)+(1:n)');
    hold on; grid on;
    title('k-means clustering scatter','FontSize',15); hold on;
    xlabel('variable','FontSize',15); ...
    ylabel('cluster','FontSize',15); hold on;

figure;
plot([cidx_raw cidx_log]);
hold on; grid on;
figure;
bar(x_axis,cidx_raw_kmean,'k');
hold on; grid on;
figure;
bar(x_axis,cidx_log_kmean,'m');
hold on; grid on;

-------------------------------------------------------------------------------
data_raw_kmean = data_raw';
data_log_kmean = data_var';

opts = statset('Display','final');
[cidx_raw_kmean,ctrs_raw_kmean] = ...
    kmeans(data_raw',fix((p-3)/k),'Distance','cityblock','Replicates',5,'Options',opts);

opts = statset('Display','final');
[cidx_log_kmean,ctrs_log_kmean] = ...
    kmeans(data_var',fix((p-3)/k),'Distance','cityblock','Replicates',5,'Options',opts);

figure;
bar(x_axis,cidx_raw_kmean,'k');
hold on; grid on;
figure;
bar(x_axis,cidx_log_kmean,'m');
hold on; grid on;
title('variables k-means clustering - log','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('cluster','FontSize',15); hold on;
end % if part_code(06) == 1

%% ...
========================================================================
%% ---------- part 07: rf (random forest) ... ---------------------------------
========================================================================

if part_code(07) == 1

% B = TreeBagger(NTrees,X,Y) creates an ensemble B of ...
% NTrees decision trees for predicting response Y as a ...
% function of predictors X.
% By default TreeBagger builds an ensemble of ...
% classification trees.
% The function can build an ensemble of regression trees ...
% by setting the optional input argument 'method' to ...
% 'regression'.
% http://www.mathworks.com/help/stats/treebagger.html
b = TreeBagger(100,data_var,data_log(1:n,2),'oobpred','on');
figure;
plot(oobError(b)); hold on; grid on;
title('random forest','FontSize',15); hold on;
xlabel('number of grown trees'); ylabel('out-of-bag ...
classification error');
% stochastic_bosque(data_var,data_log(1:n,2))

end % if part_code(07) == 1

%% ...
========================================================================
%% ---------- part 08: on and off comparison (doe approach) ...---------------
========================================================================

if part_code(08) == 1

for i=1:p-3
[p_rank_one(i),h_rank_one(i)] = ...
end % if part_code(08) == 1
ranksum(data_surv,var(i).one(:,1)); % wilcoxon rank sum test
[p_rank_zero(i),h_rank_zero(i)] = ...
ranksum(data_surv,var(i).zero(:,1)); % wilcoxon rank sum test
∆_wilcoxon_time(i,1) = p_rank_one(i) - p_rank_zero(i);
end

figure;
bar(x_axis,var_one_sum,'y','grouped');
hold on; grid off;
title('one by one variable time sum','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;

figure;
bar(x_axis,∆_wilcoxon_time,'r','grouped');
hold on; grid off;
title('∆- wilcoxon rank test','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('importance','FontSize',15); hold on;

% axis([0 17 -1 2])
end % if part_code(08) == 1

%% ...
========================================================================
%% ---------- part 09: test score - brute-force (exhaustive) search -------
========================================================================
%% ...
========================================================================
%% ...
if part_code(09) == 1
k = 3; % no. of variable combination out of all variables
comb = [];%comb = nchoosek(data_var_num(end,:),k); % ...
[data_var_num(end,:)] is number of variables vector
[comb_row comb_col] = size(comb);
% y = data_unc(:,2)'; % (or data_time') failure time row
%freqy = data_unc(:,3)';
% from part 1 ...
=================================================================================
% [f,x] = ecdf(y,'censoring',freqy);
% s = 1-f;
% figure;
% ecdf(y,'censoring',freqy,'function','survivor'); % kaplan-meier plot
% hold on; grid on;
% title('kaplan-meier survival function','FontSize',15);
% xlabel('time,t','FontSize',15); ylabel('survival ... probability,s(t)','FontSize',15);
% % plot(x,s,'color','r'); hold on; grid on; % survival plot
% % h = get(gca,'children');
% % set(h,'color','k');
% [P0,S0] = polyfit(x,s,1); % poly fit
% z = polyval(P0,x);
% % plot(x,s,'o',x,z,'-'); hold on; grid on;
% % plot(x,z,'-', Color,'r');
% % hold on; grid on;
% % axis([0 data_cal(n,2) 0 1]);
% % slop = [];
% % end of from part 1 ...

--------------------------------------------------

% figure;

sum_limit_effect_var = 0;

for i = 1:comb_row

data_comb = []; % instructing survival data from set #i
data_comb = data_var(:,1:i:k); % for set #i
data_comb(:,k+1) = sum(data_comb,2);
% data_comb(:,k+2) = sign(data_comb(:,k+1))
% data_comb(:,k+2) = floor(data_comb(:,k+1)/k);

data_comb(1:n,k+2) = data_time; % time
% data_comb(:,k+4) = data_comb(:,k+2).*data_comb(:,k+3);
data_comb(1:n,k+3) = data_log(1:n,3); % status

var_comb = []; % instructing survival data from set ...
  % i based on element k+1 (for time t,select if at ... least "sum_limit_effect_var" variables at 1)
% var_comb = data_comb(:,k+4);
% var_comb(find([var_comb(:,1)]==0,:),:) = []; % ... remove not effective failure times from set #i of ... all combinations

var_comb(:,1:2) = ...
data_comb(find(data_comb(:,k+1)>sum_limit_effect_var),k+2:k+3); ...
% remove ineffective failure times from set #i of ...
all combinations

if isempty(var_comb) == 0;

    var_comb_surv = []; 
    [var_comb_surv(:,1),var_comb_surv(:,2)] = ...
        ecdf(var_comb(:,1),'censoring',var_comb(:,2),'function','survivor')
    % kaplan-meier plot

    [comb(i,k+1),comb(i,k+2)] = ... 
        ranksum(data_time,var_comb(:,1)); % wilcoxon ... 
        rank sum test (col k+1 and k+2) for time 
    comb(i,k+3) = logrank(data_time,var_comb(:,1)); ... 
    % logrank test (col k+3) for time 

    [comb(i,k+4),comb(i,k+5)] = ... 
        ranksum(data_surv,var_comb_surv(:,1)); % ... 
        wilcoxon rank sum test (col k+4 and k+5) for ... 
        survival 
    % comb(i,k+6) = ... 
        logrank(data_surv,var_comb_surv(:,1)); % ... 
        logrank test (col k+6) for survival 

end

% kaplan-meier plot for all ...
=======================================

plot_km_for_all = 0;

if plot_km_for_all == 1

    if comb(i,k+2)==0; color_km_all = 'b'; end; 
    if comb(i,k+2)==1; color_km_all = 'r'; end; 

    % figure; 
    [f_km_all,x_km_all] = ... 
        ecdf(var_comb(:,1),'function','survivor'); % ... 
        kaplan-meier plot 

    stairs(x_km_all,f_km_all,color_km_all);

    hold on; grid on; 
    % title('kaplan-meier survival ... 
    %.function','FontSize',15); 
    % xlabel('time,t','FontSize',15); ... 
    ylabel('survival ... 

155
\[
\text{probability, } s(t), 'Fontsize', 15);
\]
\[
\text{plot(x_km_all,1-f_km_all,'color','r'); hold on; grid on; } % \text{survival plot}
\]
\[
\text{h = get(gca,'children');}
\]
\[
\text{set(h,'color','k');}
\]
\[
[p_km_all,s_km_all] = ...
\text{polyfit(var_comb_failure(:,2),var_comb_surv(:,1),1); } ... % \text{polyfit}
\]
\[
\text{z_km_all = ...}
\text{polyval(p_km_all,var_comb_failure(:,2));}
\]
\[
\text{plot(var_comb_failure(:,2),var_comb_surv(:,1),'o',var_comb_failure(:,2),z_km_all,'-'); } ...
\text{hold on; grid on;}
\]
\[
\text{plot(var_comb_failure(:,2),z_km_all,'-', 'Color','r');}
\]
\[
\text{hold on; grid on;}
\]
\[
\text{axis([0 data_cal(n,2) 0 1]);}
\]
\[
y_km = data_cens(:,2); % or data_time
freqy_km = data_cens(:,3);'
\]
\[
[f_km,x_km] = ecdf(y_data,'censoring',freqy_km);
\]
\[
data_surv_km = 1-f_km;
\]
\[
\text{figure;}
\text{ecdf(y_km,'censoring',freqy_km,'function','survivor'); } ...% \text{kaplan-meier plot}
\]
\[
\text{title('kaplan-meier survival ...}
\text{function','Fontsize',15);}
\text{xlabel('time,t','Fontsize',15); ylabel('survival ...}
\text{probability, } s(t), 'Fontsize',15);
\]
\[
\text{plot(x,s,'color','r'); hold on; grid on; } % ...
\text{survival plot}
\text{h = get(gca,'children');}
\text{set(h,'color','k');}
\]
\[
[p_km,s_km] = polyfit(x_km,data_surv_km,1); % ... \text{polyfit}
\text{z_km = polyval(p_km,x_km);
\text{plot(x,s,'o',x,z, '-'); hold on; grid on; plot(x_km,z_km,'-', 'Color','m:');}
\text{hold on; grid on;}
\text{axis([0 data_cal(n,2) 0 1]);}
\]
\[
end
\]
\[
\% ...
\]
end

data_nptest = [];
data_nptest_cum = [];
data_nptest_cum_unique = [];
data_nptest_elem_freq = [];
data_nptest_elem = [];

data_nptest = comb;

% sortcomb = sortrows(comb,4)

% ---------- keep rejected sets (positive approach) ...

% k+2 or k+5
data_nptest(find([data_nptest(:,k+2)]==0,:), :) = [];
keep rejected sets (worst variables): eliminate sets ...
with wilcoxon rank = 0 and keep wilcoxon rank = 1 ...
(keep wilcoxon rank < 0.05)
data_nptest(find([data_nptest(:,k+1)]>0.05,:), :) = [];
keep rejected sets: eliminate sets with wilcoxon rank ...
> 0.05 and keep wilcoxon rank < 0.05

data_nptest(find([data_nptest(:,k+3)]>0.05,:), :) = [];
keep rejected sets: eliminate sets with logrank ...
0.05 and keep logrank < 0.05

% ---------- eliminate rejected sets (negative approach) ...

% k+2 or k+5
data_nptest(find([data_nptest(:,k+2)]==1,:), :) = [];
eliminate rejected sets: eliminate sets with wilcoxon ...
rank = 1 and keep wilcoxon rank = 0 (keep wilcoxon ...
rank > 0.05)
data_nptest(find([data_nptest(:,k+1)]<0.05,:), :) = [];
eliminate rejected sets: eliminate sets with wilcoxon ...
rank < 0.05 and keep wilcoxon rank > 0.05

data_nptest(find([data_nptest(:,k+3)]<0.05,:), :) = [];
eliminate rejected sets: eliminate sets with logrank ...
< 0.05 and keep logrank > 0.05

% ...

data_nptest = data_nptest(:,1:k);
[data_row data_col] = size(data_nptest);
for i = 1:data_col % put all selected var numbers in a ...
single vector
data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
data_nptest(:,i);
end

data_nptest_cum_unique = unique(data_nptest_cum); % ...
elements in data_nptest_cum
[data_nptest_elem_freq,data_nptest_elem] = ...
hist(data_nptest_cum,data_nptest_cum_unique); % ...
elements in data_nptest_cum and their frequencies

figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique)); ...
% elements in data_nptest_cum and their frequencies - ...
method 1
hold on; grid on;
title('wilcoxon rank or logrank frequencies (method ...
1)','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

figure;
bar(data_nptest_elem,data_nptest_elem_freq); % elements ...
in data_nptest_cum and their frequencies - method 2
hold on; grid on;
title('wilcoxon rank or logrank frequencies (method ...
2)','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

figure;
bar(data_nptest_elem_comb,data_nptest_elem_freq_comb); % elements ...
in data_nptest_cum and their frequencies - method 2
hold on; grid on;
title('wilcoxon rank or logrank frequencies (method ...
2)','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

for i = 1:p-3
  data_comb_nptest_cum(i,1) = i;
data_comb_nptest_cum(i,2) = 0;
data_comb_nptest_cum(i,3) = 0;
end
for i = 1:comb_row
    for j = 1:comb_col % = k
        data_comb_nptest_cum(comb(i,j),2) = ...
        data_comb_nptest_cum(comb(i,j),2) + ...
        comb(i,k+1); % cumulation of all wilcoxon ...
        rank for all variables
        data_comb_nptest_cum(comb(i,j),3) = ...
        data_comb_nptest_cum(comb(i,j),3) + ...
        comb(i,k+3); % cumulation of all logrank for ...
        all variables
    end
end

figure;
bar(data_comb_nptest_cum(:,2),'r','grouped');
hold on; grid off;
title('cumulation of all wilcoxon rank','FontSize',15);
hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
figure;
bar(data_comb_nptest_cum(:,3),'b','stacked');
hold on; grid off;
title('cumulation of all logrank','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
figure;
bar(data_comb_nptest_cum(:,2:3),'grouped');
hold on; grid off;
title('cumulation of all wilcoxon rank and ...
      logrank','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
figure;
bar(data_comb_nptest_cum(:,2:3),'stacked');
hold on; grid off;
title('cumulation of all wilcoxon rank and ...
      logrank','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
data_comb_nptest_cum_norm = ...
    9*((data_comb_nptest_cum(:,2) - ...
    min(data_comb_nptest_cum(:,2)))/(max(data_comb_nptest_cum(:,2)) ...
- min(data_comb_nptest_cum(:,2))) + 1; % 1 to 10 ...
score normalized for overal plot at the end of part 10

end % if part_code(09) == 1

%% ...
========================================================================
%% ---------- part 10: test score - random subset selection ... 
---------------
========================================================================
%
if part_code(10) == 1

m = 100; % no. of randomly selected sets
k = 3; % no. of variable combination out of all variables
select = [];

for i = 1:m
    select(i,:) = randsample(data_var_num(end,:),k,false);
end

sum_limit_effect_var = 0;

figure;

for i = 1:m % trials

    data_select = []; % instructing survival data from ...
    % set #i
    data_select = data_var(:,select(i,1:k)); % for set #i
    data_select(:,k+1) = sum(data_select,2);
    data_select(1:n,k+2) = data_time;

    var_select = [];
    % instructing survival data from ...
    % set #i based on element k+1 (for time t, select if ...
    % at least "sum_limit_effect_var" variables at 1)
    var_select = ...
    data_select(find(data_select(:,k+1)>sum_limit_effect_var),k+2); ...
    % remove ineffective failure times from set #i of ...
    % all combinations
    var_select(:,2) = 1; % just as a frequency of ...
    % failure time in order to plot kaplan-meier ...
    % survival function

    if isempty(var_select) == 0;
        var_select_surv = [];
        [var_select_surv(:,1),var_select_surv(:,2)] = ...
    end

end
ecdf(var_select(:,1),'censoring',var_select(:,2),'function','survivor'); ...
% kaplan-meier plot

select(i,k+1),select(i,k+2)] = ... ranksum(data_time,var_select(:,1)); % ... wilcoxon rank sum test (col k+1 and k+2) for time
select(i,k+3) = ... logrank(data_time,var_select(:,1)); % logrank ... test (col k+3) for time

select(i,k+4),select(i,k+5)] = ... ranksum(data_surv,var_select_surv(:,1)); % ... wilcoxon rank sum test (col k+4 and k+5) for ... survival
% select(i,k+6) = ... logrank(data_surv,var_comb_surv(:,1)); % ... logrank test (col k+6) for survival

end

% kaplan-meier plot for all ...
=======================================
plot_km_for_all = 0;

if plot_km_for_all == 1

if select(i,k+2)==0; color_km_all = [0.8 0.8 ... 0.8]; end;
if select(i,k+2)==1; color_km_all = [0.0 0.0 ... 0.0]; end;

% if select(i,k+2)==0; color_km_all = ... [c_km_all_code c_km_all_code c_km_all_code]; end
% if select(i,k+2)==1; color_km_all = ... [c_km_all_code c_km_all_code c_km_all_code]; end

% figure;
[f_km_all,x_km_all] = ... ecdf(var_select(:,1),'function','survivor'); ... % kaplan-meier plot

stairs(x_km_all,f_km_all,'color',color_km_all);
hold on; grid on;
% title('kaplan-meier survival ... function','FontSize',15);
xlabel('time','FontSize',15); %
ylabel('survival probability','FontSize',15);

[p_km_all,s_km_all] = polyfit(var_select_failure(:,2),var_select_surv(:,1),1); % poly fit

z_km_all = polyval(p_km_all,var_comb_failure(:,2)); %

plot(var_comb_failure(:,2),var_comb_surv(:,1),'o',var_comb_failure(:,2),z_km_all,'-'); %
hold on; grid on;

plot(var_comb_failure(:,2),z_km_all,'-','Color','r'); %
hold on; grid on;
axis([0 data_cal(n,2) 0 1]);

[p_km,s_km] = polyfit(x_km,data_surv_km,1); % poly fit

figure;
% ...% kaplan-meier survival function,'FontSize',15);
xlabel({'time, t','FontSize',15}); %
ylabel({'survival probability, s(t)}','FontSize',15);

plot(x,s,'color','r'); hold on; grid on; % ...
survival plot

h = get(gca,'children');% set(h,'color','k');

[p_km,s_km] = polyfit(x_km,data_surv_km,1); % ...

\[ z_{km} = \text{polyval}(p_{km}, x_{km}); \]
\[
\% \text{plot}(x, s, 'o', x, z, '-'); \text{hold on; grid on};
\]
\[
\% \text{plot}(x_{km}, z_{km}, 'm:');\text{hold on; grid on;}
\]
\[
\text{axis([0 data_cal(n,2) 0 1]);}
\]
\[
\text{end}
\]
\[
\% \ldots
\]
=================================================================
\[
\text{end}
\]
\[
\text{data_nptest} = [];\]
\[
\text{data_nptest\_cum} = [];\]
\[
\text{data_nptest\_cum\_unique} = [];\]
\[
\text{data_nptest\_elem\_freq} = [];\]
\[
\text{data_nptest\_elem} = [];\]
\[
\text{data_nptest} = \text{select};\]
\[
\% \text{sortcomb} = \text{sortrows(comb,4)}\]
\[
\% \text{--------- keep rejected sets (positive approach) ...}\]
\[
\text{-----------}
\]
\[
% k+2 or k+5
\]
\[
\text{data_nptest}(\text{find([data_nptest(:,k+2)]==0),:}) = []; \% \ldots
\]
\[
\text{keep rejected sets: eliminate sets with wilcoxon rank ...}
\]
\[
\text{= 0 and keep wilcoxon rank = 1 (keep wilcoxon rank < ... 0.05)}
\]
\[
% \text{data_nptest}(\text{find([data_nptest(:,k+1)]>0.05),:}) = []; \% \ldots
\]
\[
\text{keep rejected sets: eliminate sets with wilcoxon rank ...}
\]
\[
\text{> 0.05 and keep wilcoxon rank < 0.05}
\]
\[
% \text{data_nptest}(\text{find([data_nptest(:,k+3)]>0.05),:}) = []; \% \ldots
\]
\[
\text{keep rejected sets: eliminate sets with logrank > ...}
\]
\[
\text{0.05 and keep logrank < 0.05}
\]
\[
% \text{--------- eliminate rejected sets (negative approach) ...}
\]
\[
\text{----------}
\]
\[
% \text{data_nptest}(\text{find([data_nptest(:,k+2)]==1),:}) = []; \% \ldots
\]
\[
\text{eliminate rejected sets: eliminate sets with wilcoxon ...}
\]
\[
\text{rank = 0 and keep wilcoxon rank = 1 (keep wilcoxon ...}
\]
\[
\text{rank > 0.05)}
\]
\[
% \text{data_nptest}(\text{find([data_nptest(:,k+1)]<0.05),:}) = []; \% \ldots
\]
\[
\text{eliminate rejected sets: eliminate sets with wilcoxon ...}
\]
\[
\text{rank > 0.05 and keep wilcoxon rank > 0.05}
\]
% data_nptest(find([data_nptest(:,k+3)]<0.05),:) = []; % ... 
eleminate rejected sets: eleminate sets with logrank ... 
> 0.05 and keep logrank > 0.05

% ...

---------------------------------------------------------------------

data_nptest = data_nptest(:,1:k);
[data_row data_col] = size(data_nptest);

for i = 1:data_col % put all selected var numbers in a ... 
single vector
    data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ... 
    data_nptest(:,i);
end

data_nptest_cum_unique = unique(data_nptest_cum); % ... 
elements in data_nptest_cum
[data_nptest_elem_freq,data_nptest_elem] = ... 
    hist(data_nptest_cum,data_nptest_cum_unique); % ... 
elements in data_nptest_cum and their frequencies

figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique)); ... 
% elements in data_nptest_cum and their frequencies - ... 
    method 1
    hold on; grid off;
    title('wilcoxon rank or logrank frequencies (method ... 
1)','FontSize',15); hold on;
    xlabel('variable','FontSize',15); ... 
    ylabel('censoring','FontSize',15); hold on;

figure;
bar(data_nptest_elem,data_nptest_elem_freq); % elements ... 
in data_nptest_cum and their frequencies - method 2
    hold on; grid off;
    title('wilcoxon rank or logrank frequencies (method ... 
2)','FontSize',15); hold on;
    xlabel('variable','FontSize',15); ... 
    ylabel('censoring','FontSize',15); hold on;

data_nptest_elem_select = data_nptest_elem; % for overal ... 
plot at the end of part 10

data_nptest_elem_freq_select = 9*((data_nptest_elem_freq ... 
- min(data_nptest_elem_freq))/(max(data_nptest_elem_freq) ... 
- min(data_nptest_elem_freq))) + 1; % 1 to 10 score ... 
normalized for overal plot at the end of part 10
% ----------- cumulative test scores for all variables ...

-----------------

data_select_nptest_cum = []; 

for i = 1:p-3 
    data_select_nptest_cum(i,1) = i;  
    data_select_nptest_cum(i,2) = 0;  
    data_select_nptest_cum(i,3) = 0;  
end; 

for i = 1:m 
    for j = 1:k 
        data_select_nptest_cum(select(i,j),2) = ...  
        data_select_nptest_cum(select(i,j),2) + ...  
        select(i,k+1); % cumulation of all wilcoxon rank for all variables  
        data_select_nptest_cum(select(i,j),3) = ...  
        data_select_nptest_cum(select(i,j),3) + ...  
        select(i,k+3); % cumulation of all logrank for all variables  
    end 
end

figure; 
bar(data_select_nptest_cum(:,2),'r','grouped'); 
hold on; grid off; 
title('cumulation of all wilcoxon rank','FontSize',15); 
hold on; 
xlabel('variable','FontSize',15); 
ylabel('y','FontSize',15); hold on;

figure;  
bar(data_select_nptest_cum(:,3),'b','stacked'); 
hold on; grid off; 
title('cumulation of all logrank','FontSize',15); hold on;  
xlabel('variable','FontSize',15); 
ylabel('y','FontSize',15); hold on;

figure; 
bar(data_select_nptest_cum(:,2:3),'grouped'); 
hold on; grid off;  
title('cumulation of all wilcoxon rank and logrank','FontSize',15); hold on;  
xlabel('variable','FontSize',15); 
ylabel('y','FontSize',15); hold on;

figure;
bar(data_select_nptest_cum(:,2:3),'stacked');
hold on; grid off;
title('cumulation of all wilcoxon rank and ...
   logrank','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;

data_select_nptest_cum_norm = ... 
   9*((data_select_nptest_cum(:,2) - 
      min(data_select_nptest_cum(:,2)))/(max(data_select_nptest_cum(:,2)) 
      - min(data_select_nptest_cum(:,2)))) + 1; % 1 to 10 ...
score normalized for overall plot at the end of part 10

figure;
subplot(2,1,1);
bar(data_nptest_elem_comb,data_nptest_elem_freq_comb,'FaceColor',[0.1 ... 
   0.1 0.1]); % wilcoxon
hold on; grid off;
bar(data_nptest_elem_select,data_nptest_elem_freq_select,0.4,'FaceColor',[1.0 ... 
   0.6 0.8]); % wilcoxon
hold on; grid off;
% title('...','FontSize',15); hold on
xlabel('variable','FontSize',15); ...
ylabel({'normalized';'efficiency ... score'},'FontSize',15); hold on;
axis([0 18 0 10]);
subplot(2,1,2);
bar(data_comb_nptest_cum_norm,'FaceColor',[0.8 0.8 ... 
   0.8]); % wilcoxon
hold on; grid off;
bar(data_select_nptest_cum_norm,0.4,'FaceColor',[0.0 0.3 ... 
   0.2]); % wilcoxon
hold on; grid off;
% title('...','FontSize',15); hold on
xlabel('variable','FontSize',15); ...
ylabel({'normalized';'inefficiency ... score'},'FontSize',15); hold on;
axis([0 18 0 10]);

end % if part_code(10) == 1

%% ...
=========================================================================
%% ---------- part 11: time score - brute-force (exhaustive) ...
% search -------
=========================================================================

if part_code(11) == 1
k = 3; % no. of variable combination out of all variables
comb = [];  
comb = nchoosek(data_var_num(end,:),k);  
[comb_row comb_col] = size(comb);  
sum_limit_effect_var = k-1;  
for i = 1:comb_row
    data_comb = []; % instructing survival data from set #i
    data_comb = data_var(:,comb(i,1:k)); % for set #i
    data_comb(:,k+1) = sum(data_comb,2);
    data_comb(1:n,k+2) = data_time;
    var_comb = []; % instructing survival data from set ...
    % i based on element k+1 (for time t, select if at ...
    % least "sum_limit_effect_var" variables at 1)
    var_comb = ...  
    data_comb(find(data_comb(:,k+1)==sum_limit_effect_var),k+2); ...  
    % remove ineffective failure times from set #i of ...
    % all combinations
    var_comb(:,2) = 1; % just as a frequency of failure ...
    % time in order to plot kaplan-meier survival function
    var_comb_sum = [];  
    var_comb_sum = sum(var_comb(:,1)); % score of set = ...
    % sum of failure times includes all variables in ...
    % set #i
    comb(i,k+1) = var_comb_sum; % add score to last col ...
    % of set #i
end  

data_comb_time_cum = []; % cumulative scores for all ...
variables  
for i = 1:p-3
    data_comb_time_cum(i,1) = i;  
    data_comb_time_cum(i,2) = 0;  
end;  

for i = 1:comb_row
    for j = 1:comb_col % = k
        data_comb_time_cum(comb(i,j),2) = ...  
        data_comb_time_cum(comb(i,j),2) + ...  
        comb(i,k+1); % cumulation of all scores for ...
        all variables
end
end
figure;
bar(data_comb_time_cum(:,2),'k','grouped');
hold on; grid off;
title('cumulation of all variable scores - ... combination','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
end % if part_code(11) == 1

%% ...
========================================================================
%% ---------- part 12: time score - random subset selection ...---------------
========================================================================
%% ...
========================================================================
%% if part_code(12) == 1
m = 100; % no. of randomly selected sets
k = 3; % no. of variable combination out of all variables
select = [];
for i = 1:m
    select(i,:) = randsample(data_var_num(end,:),k,false);
end
sum_limit_effect_var = k;
for i = 1:m % trials
    data_select = []; % instructing survival data from ...
    set #i
    data_select = data_var(:,select(i,1:k)); % for set #i
    data_select(:,k+1) = sum(data_select,2);
    data_select(1:n,k+2) = data_time;
    var_select = []; % instructing survival data from ...
    set #i based on element k+1 (for time t,select if ...
    at least "sum_limit_effect_var" variables at 1)
    var_select = ...
    data_select(find(data_select(:,k+1)==sum_limit_effect_var),k+2); ... % remove ineffective failure times from set #i of ...
    all combinations
    var_select(:,2) = 1; % just as a frequency of ...
    failure time in order to plot kaplan-meier ...
survival function

```matlab
var_select_sum = []; 
var_select_sum = sum(var_select(:,1)); % score of ... 
set = sum of failure times includes all variables ... 
in set #i 
select(i,k+1) = var_select_sum; % add score to last ... 
col of set #i
end

data_select_time_cum = []; % cumulative scores for all ... 
variables 
for i = 1:p-3 
data_select_time_cum(i,1) = i; 
data_select_time_cum(i,2) = 0; 
end; 
for i = 1:m 
for j = 1:k 
data_select_time_cum(select(i,j),2) = ... 
data_select_time_cum(select(i,j),2) + ... 
select(i,k+1); % cumulation of all scores for ... 
all variables 
end 
figure; 
bar(data_select_time_cum(:,2),'r','grouped'); 
hold on; grid off; 
title('cumulation of all variable scores - ... 
randomization','FontSize',15); hold on; 
xlabel('variable','FontSize',15); ... 
ylabel('y','FontSize',15); hold on; 
end % if part_code(12) == 1

%% ...
========================================================================
%% ---------- part 13: bisection greedy clustering algorithm ... 
========================================================================

if part_code(13) == 1 
m = 100; % no. of trials
```
clus_limit = 3; % min no. of cluster members
var_num = p-3;
var_vector = [];
var_vector = data_var_num(end,:);
var_rand = [];

data_nptest = [];
data_nptest_cum = [];
data_nptest_cum_unique = [];
data_nptest_elem_freq = [];
data_nptest_elem = [];

for i = 1:m % trials
    greedy_cluster_num = var_num/2;
    var_rand = [];
    greedy_a_var_num = [];
    greedy_b_var_num = [];
    var_rand = var_vector(randperm(length(var_vector)));
    % random sort of var numbers
    while greedy_cluster_num ≥ clus_limit
        k = greedy_cluster_num;
        sum_limit_effect_var = greedy_cluster_num-2;
        greedy_a_var_num = var_rand(1:greedy_cluster_num);
        greedy_b_var_num = ...
            var_rand(greedy_cluster_num+1:end);
        % cluster a
        greedy_a = [];
        data_greedy_a = [];
        data_greedy_a = ... 
            data_var(:,greedy_a_var_num(1,1:k)); %...
                instructing cluster a
        data_greedy_a(:,k+1) = sum(data_greedy_a,2);
        data_greedy_a(1:n,k+2) = data_time;
        var_greedy_a = [];
        var_greedy_a = ...
            data_greedy_a(find(data_greedy_a(:,k+1)>sum_limit_effect_var),k+2);
            % remove ineffective failure times from ... cluster a
        var_greedy_a(:,2) = 1; % just as a frequency of ...
failure time in order to plot kaplan-meier ... survival function

[greedy_a(1,1),greedy_a(1,2)] = ...
    ranksum(data_time,var_greedy_a(:,1)); % ...
    wilcoxon rank sum test (col k+1 and k+2)
greedy_a(1,3) = ...
    logrank(data_time,var_greedy_a(:,1)); % ...
    logrank test (col k+3)

% cluster b

greedy_b = [];
data_greedy_b = []; % cluster b
data_greedy_b = ... 
    data_var(:,greedy_a_var_num(1,1:k)); % ...
    instructing cluster b 
data_greedy_b(:,k+1) = sum(data_greedy_b,2);
data_greedy_b(1:n,k+2) = data_time;

var_greedy_b = []; % instructing survival data ...
    from cluster b based on element k+1 (for time ...
    t, select if at least "sum_limit_effect_var" ... variables at 1)
var_greedy_b = ... 
data_greedy_b(find(data_greedy_b(:,k+1)>sum_limit_effect_var),k+2);
    % remove ineffective failure times from ... 
    cluster b 
var_greedy_b(:,2) = 1; % just as a frequency of ...
    failure time in order to plot kaplan-meier ...
    survival function

[greedy_b(1,1),greedy_b(i,2)] = ...
    ranksum(data_time,var_greedy_b(:,1)); % ...
    wilcoxon rank sum test (col k+1 and k+2)

    logrank(data_time,var_greedy_b(:,1)); % ...
    logrank test (col k+3)

var_rand = [];

if greedy_a(1,1) > greedy_b(1,1) 
    var_rand = greedy_a_var_num;
    var_rand_test = greedy_a_var_num;
    var_rand_test(1,4) = greedy_a(1,1);
else
    var_rand = greedy_b_var_num;
    var_rand_test = greedy_a_var_num;
    var_rand_test(1,4) = greedy_b(1,1);
end
greedy_cluster_num = greedy_cluster_num/2;
end % while

data_nptest = [data_nptest; var_rand];
end % for i = 1:m % trials
[data_row data_col] = size(data_nptest);
for i = 1:data_col % put all selected var numbers in a ... single vector
data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
data_nptest(:,i);
end
data_nptest_cum_unique = unique(data_nptest_cum); % ... elements in data_nptest_cum
[data_nptest_elem_freq,data_nptest_elem] = ...
hist(data_nptest_cum,data_nptest_cum_unique); % ... elements in data_nptest_cum and their frequencies
figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique),'m'); ...
% elements in data_nptest_cum and their frequencies - ... method 1
hold on; grid off;
title('wilcoxon rank or logrank frequencies (method ... 1)',['FontSize',15]); hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
figure;
bar(data_nptest_elem,data_nptest_elem_freq,'c'); % ... elements in data_nptest_cum and their frequencies - ... method 2
hold on; grid off;
title('wilcoxon rank or logrank frequencies (method ... 2)',['FontSize',15]); hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
end % if part_code(13) == 1

%%% ...

========================================================================
%% ---------- part 14: splitting semi-greedy clustering ...
algorithm --------------
if part_code(14) == 1

m = 100; % no. of trials
k = 3; % no. of cluster members
var_num = p-3;
CLUS_NU = ((p-3) - mod(p-3,k))/k;
clus_nu = fix((p-3)/k); %

var_vector = [];
var_vector = data_var_num(end,:);
var_rand = [];

data_nptest = [];
data_nptest_cum = [];
data_nptest_cum_unique = [];
data_nptest_elem_freq = [];
data_nptest_elem = [];

for i = 1:m % trials

var_rand = [];
greedy(clus_nu) = ... 
struct('var_num',[],'data',[],'var',[],'test',[]);

var_rand = ... 
var_vector(randperm(length(var_vector))); % ... 
random sort of var numbers

var_rand = randperm(p-3); % random sort of var numbers

sum_limit_effect_var = 0;

for j = 1:clus_nu

greedy(j).var_num = var_rand(k*(j-1)+1:k*j);
end

% greedy(clus_nu).var_num = ... 
var_rand(k*(clus_nu-1)+1:end)

for j = 1:clus_nu % cluster #j
[greedy_var_num_row greedy_var_num_col] = ... 
size(greedy(j).var_num);
kk = greedy_var_num_col; % is equal to k for ... 
regular clusters and it would be less than k ...
if put remains is an extra cluster

greedy(j).data = []; % cluster #j

greedy(j).data = ...
data_var(:,greedy(j).var_num(1,1:kk)); % ... instructing cluster #j (var_num(1,1:k))
greedy(j).data(:,kk+1) = sum(greedy(j).data,2);
greedy(j).data(1:n,kk+2) = data_time;

1834

greedy(j).var = [];
% instructing survival data from cluster #j based on element k+1 (for time t, select if at least "sum_limit_effect_var" variables at 1)

1836

greedy(j).var = ...
greedy(j).data(find(greedy(j).data(:,kk+1)>sum_limit_effect_var),kk)
% remove ineffective failure times from cluster #j

1837

greedy(j).var(:,2) = 1;
% just as a frequency of failure time in order to plot kaplan-meier ... survival function

1838

[greedy(j).test(1,1),greedy(j).test(1,2)] = ...
ranksum(data_time,greedy(j).var(:,1));
% ... wilcoxon rank sum test (col k+1 and k+2)
greedy(j).test(1,3) = ...
logrank(data_time,greedy(j).var(:,1));
% ... logrank test (col k+3)

1840

end

1841

1842

greedy_result(1,j) = j;
greedy_result(2,j) = greedy(j).test(1,1);

1843

1844

end % for i = 1:m % trials

1845

1846

[max_num,max_index] = max(greedy_result(2,:));

1847

var_rand = greedy(greedy_result(1,max_index)).var_num;
data_nptest = [data_nptest; var_rand];

1849

1850

end % for i = 1:m % trials

1851

1852

[data_row data_col] = size(data_nptest);

1853

for i = 1:data_col % put all selected var numbers in a ... single vector
    data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
    data_nptest(:,i);
end

1858

1859

end

1860

data_nptest_cum_unique = unique(data_nptest_cum); % ... elements in data_nptest_cum

1861

174
break % elements in data_nptest_cum and their frequencies - ... method 1
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 1)'),'FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique),'FaceColor',[0.0,0.4,0.4],'EdgeColor',[0.0,0.1,0.1]); ...
% elements in data_nptest_cum and their frequencies - ... method 2
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 2)'),'FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
figure;
bar(data_nptest_elem,data_nptest_elem_freq,'FaceColor',[0.4,0.4,0.4],'EdgeColor',[0.1,0.1,0.1]); ...
% elements in data_nptest_cum and their frequencies - ... method 2
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 2)'),'FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

% survival comparision od sets (instead of time)
end % if part_code(14) == 1

%% ...
========================================================================
%% ---------- part 15: weighted standardized score method ... ----------------
========================================================================
%% ...
if part_code(15) == 1
for i= 1:p-3
    weighted_score(:,i) = data_time.*data_nor(:,i);
end
weighted_score_sum(1,:) = abs(sum(weighted_score(:,:)));
figure;
bar(x_axis,weighted_score_sum,'FaceColor',[0.0,0.0,0.4],'EdgeColor',[0.1,0.1,0.1]); ...
hold on; grid off;
title('one by one comparison weighted score','FontSize',15);
xlabel('variable','FontSize',15); ...
ylabel('importance','FontSize',15); hold on;
axis([0 17 -1 2])
m = 100; % no. of randomly selected sets
k = 3; % no. of variable combination out of all variables
weight = [ ];

for i = 1:m
    weight(i,:) = randsample(data_var_num(end,:),k,false);
end

% weight = nchoosek(data_var_num(end,:),k);
[weight_row weight_col] = size(weight);

for i = 1:weight_row
    data_weight = [ ]; % instructing survival data from ... set #i
    data_weight = data_var(:,weight(i,1:k)); % for set #i
data_weight(:,k+1) = sum(data_weight,2);
data_weight(1:n,k+2) = data_time;
data_weight(:,k+3) = ... data_weight(:,k+1).*data_weight(:,k+2); % weight each observation of set #i
    weight(i,k+1) = sum(data_weight(:,k+3)); % weighted score of set #i
end

data_weight_cum = [ ];

for i = 1:p-3
    data_weight_cum(i,1) = i;
    data_weight_cum(i,2) = 0;
end;

for i = 1:weight_row
    for j = 1:weight_col % = k
        data_weight_cum(weight(i,j),2) = ...
        data_weight_cum(weight(i,j),2) + ...
        weight(i,k+1); % cumulation of all weighted scores for all variables
    end
end

figure;
bar(x_axis,data_weight_cum(:,2),'FaceColor',[0.2,0.2,0.2],'EdgeColor',[0.1,0.1,0.1],hold on; grid off;
title('combination weighted score','FontSize',15); hold on;
xlabel('variable','FontSize',15); ylabel('combination ... weighted score','FontSize',15); hold on;
end % if part_code(15) == 1

%% ...

========================================================================
%% ---------- part 16: clustering aft regression ...
--------------------------

%% ...
========================================================================

if part_code(16) == 1

k = 3; % no. of variable combination out of all variables
comb_reg = [];
comb_reg = nchoosek(data_var_num(end,:),k); % ...
data_var_num(end,:) is number of variables vector
[comb_reg_row comb_reg_col] = size(comb_reg);

for i = 1:comb_reg_row

  % raw data ...
  data_reg_comb_raw = []; % instructing survival data ...
  from set #i
  data_reg_comb_raw = data_raw(:,comb_reg(i,1:k)); % ...
  for set #i
  data_reg_comb_raw(:,k+1) = sum(data_reg_comb_raw,2);
data_reg_comb_raw(1:n,k+2) = data_time; % time
  data_reg_comb_raw(1:n,k+3) = data_log(1:n,3); % status
  reg_raw = regress(data_time,data_reg_comb_raw(:,1:k));

  % ...
  data_reg_comb_var = []; % instructing survival data ...
  from set #i
  data_reg_comb_var = data_var(:,comb_reg(i,1:k)); % ...
  for set #i
  data_reg_comb_var(:,k+1) = sum(data_reg_comb_var,2);
data_reg_comb_var(1:n,k+2) = data_time; % time
  data_reg_comb_var(1:n,k+3) = data_log(1:n,3); % status
  reg_var_a = regress(data_time,data_reg_comb_var(:,1:k));
  reg_var_b = fitlm(data_reg_comb_var(:,1:k),data_time);
  reg_var_c = ...
  fitlm(data_reg_comb_var(:,1:k),data_time,'linear','RobustOpts','on');

end % if part_code(15) == 1

end % if part_code(16) == 1
comb_reg(i,k+1:2*k) = abs(reg_var_a');
% comb_reg(i,2*k+1:3*k) = ...
    abs(reg_var_b.Coefficients(2:end,1));
% comb_reg(i,3*k+1:4*k) = ...
    abs(reg_var_c.Coefficients(2:end,1));
end

data_reg_cum = []; for i = 1:p-3
    data_reg_cum(i,1) = i;
    data_reg_cum(i,2) = 0;
    % data_reg_cum(i,3) = 0;
    % data_reg_cum(i,4) = 0;
end;
for i = 1:comb_reg_row
    for j = 1:comb_reg_col % = k
        data_reg_cum(comb_reg(i,j),2) = ...
            data_reg_cum(comb_reg(i,j),2) + ...
                comb_reg(i,1*k+j); % cumulation of all ... regression type 1
        % data_reg_cum(comb_reg(i,j),3) = ...
            data_reg_cum(comb_reg(i,j),3) + ...
                comb_reg(i,2*k+j); % cumulation of all ... regression type 2
        % data_reg_cum(comb_reg(i,j),4) = ...
            data_reg_cum(comb_reg(i,j),4) + ...
                comb_reg(i,3*k+j); % cumulation of all ... regression type 3
    end
end
figure;
bar(data_reg_cum(:,2),'k','grouped'); hold on; grid off;
title('cumulative clustering aft ...
    regression','FontSize',15); hold on;
xlabel('variable','FontSize',15); xlabel('y','FontSize',15); hold on;
figure;
bar(data_reg_cum(:,3),'b','grouped'); hold on; grid off;
title('cumulative clustering aft ...
    regression','FontSize',15); hold on;
xlabel('variable','FontSize',15); xlabel('y','FontSize',15); hold on;
data_comb_nptest_cum_norm = 9*((data_comb_nptest_cum(:,2) - min(data_comb_nptest_cum(:,2)))/(max(data_comb_nptest_cum(:,2)) - min(data_comb_nptest_cum(:,2)))) + 1; % 1 to 10 ... score normalized for overall plot at the end of part 10

[b, bint, r, rint, stats] = regress(data_time, data_var);

% lm_unc = fitlm(data_unc(1:n,4:p),data_unc(1:n,2));
% rlm_unc = ...
% fitlm(data_unc(1:n,4:p),data_unc(1:n,2),'linear','RobustOpts','on');

lm_unc = fitlm(data_cens(1:n,4:p),data_cens(1:n,2));
rlm_unc = ...
fitlm(data_cens(1:n,4:p),data_cens(1:n,2),'linear','RobustOpts','on');

lm_var = fitlm(data_var, data_log(1:n,2));
ylabel('y','FontSize',15); hold on;
% figure;
% plot(lm_unc); hold on; grid on;
% figure;
% plot(lm_var); hold on; grid on;
figure;
plot((b),'k');
hold on; grid on;
title('cumulation of all variable scores - randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
plot((lm_unc.Coefficients(2:end,1)),'b');
hold on; grid on;
title('cumulation of all variable scores - randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ...
ylabel('y','FontSize',15); hold on;
plot((rlm_unc.Coefficients(2:end,1)),'r');

end % if part_code(16) == 1

% ... ====
% ------- part 19: aft ...
---
% ... ====
% ...


data_status = data_cens(:,3);
distr = 'weibull';
init = [];
% [pars covpars SE CI Zscores pvalues gval exitflag ...
  hess] = ...
aft(data_time, data_raw, dat_status, distr, init, minimizer, options, printresults
  [pars covpars SE CI Zscores pvalues gval exitflag hess] ...
  = aft(data_time, data_raw, data_status, distr, init,1,1,1);
if part_code(20) == 1

% norm: (max - min)/range = (upper - lower)/new range

x_data = data_select_nptest_cum(:,2); % part 10
x_data_norm = 1*((x_data - min(x_data))/(max(x_data) - ...
  min(x_data))) + 0; % max = 1, min = 0
x_data_scatter = 20*((x_data - min(x_data))/(max(x_data) ...
  - min(x_data))) + 0; % max = 20, min = 0

y_data = data_select_time_cum(:,2); % part 12
y_data_norm = 1*((y_data - min(y_data))/(max(y_data) - ...
  min(y_data))) + 0; % max = 1, min = 0
y_data_scatter = 20*((y_data - min(y_data))/(max(y_data) ...
  - min(y_data))) + 0; % max = 20, min = 0

z_data = data_nptest_elem_freq'; % part 14 ...
% z_data = data_weight_cum(:,2); % part 15
z_data_norm = 1*((z_data - min(z_data))/(max(z_data) - ...
  min(z_data))) + 0; % max = 1, min = 0
z_data_scatter = 1.1*((z_data - min(z_data))/(max(z_data) ...
  - min(z_data))) + 1; % max = 1.1, min = 1

plotter_scat = ...
[x_data_scatter,y_data_scatter,z_data_scatter];
plotter_norm = [x_data_norm,y_data_norm,z_data_norm];

figure;
for i = 1:p-3
  x_data = plotter_scat(i,1);
y_data = plotter_scat(i,2);
r = plotter_scat(i,3);
% for ang = 0:0.05:2*pi;
%  xp = r*cos(ang);
%  yp = r*sin(ang);
%  plot(x+xp,y+yp,'Color',[0.1 0.1 ... 0.1],'LineWidth',2); hold on;
% end

% plot(1,1,'.r','MarkerSize',100)
px = x_data-r;
py = y_data-r;
% h = rectangle('Position',[px py r*2 ... r*2],'Curvature',[1,1],'FaceColor',[0 rand rand]);
% h = rectangle('Position',[px py r*2 ... r*2],'Curvature',[1,1],'FaceColor',[0 ... (x_data_norm(i,1) + y_data_norm(i,1))/2 ... (x_data_norm(i,1) + y_data_norm(i,1))/2]);
h = rectangle('Position',[px py r*2 ... r*2],'Curvature',[1,1],'FaceColor',[0 ... (x_data_norm(i,1) + y_data_norm(i,1))/2 (x_data_norm(i,1) + ... y_data_norm(i,1))/2]);
% h = rectangle('Position',[px py r*2 ... r*2],'Curvature',[1,1],'FaceColor',[0 ... (x_data_norm(i,1) + y_data_norm(i,1) + r_data_norm(i,1))/3 (x_data_norm(i,1) + ... y_data_norm(i,1) + r_data_norm(i,1))/3]);
daspect([1,1,1]);
% text(x+px,y+py,['\fontsize{12}','\color[rgb]{.5 .0 .0}',num2str(data_comb_nptest_cum(i,1))],'FontSize',5); ...
hold on; % 'FontWeight','bold'

hold on; grid on;
end

for i = 1:p-3
x_data = plotter_scat(i,1);
y_data = plotter_scat(i,2);
text(x_data,y_data,['\fontsize{12}','\color[rgb]{1 1 1}',num2str(data_comb_nptest_cum(i,1))],'FontSize',5); ...
hold on; % 'FontWeight','bold'

hold on; grid on;
end

axis([min(plotter_scat(:,1))-5 max(plotter_scat(:,1))+5 ... min(plotter_scat(:,2))-5 max(plotter_scat(:,2))+5]);
hold on; grid on;
% title('balloon scatter plot (a)','FontSize',15); hold on
xlabel('criterion a','FontSize',15); ylabel('criterion ... b','FontSize',15); hold on;
set(gca,'FontSize',15);
figure;
scatter(x_data_norm,y_data_norm,fix(500*(z_data_norm+.1)),rand([p-3,3]),'fill', ... % http://www.mathworks.com/help/matlab/ref/scatter.html
axis([min(plotter_norm(:,1))-0.1 ... max(plotter_norm(:,1))+0.1 min(plotter_norm(:,2))-0.1 ... max(plotter_norm(:,2))+0.1]);
hold on; grid on;
scatter_labels = cellstr(num2str((1:p-3)'));
text(x_data_norm,y_data_norm,scatter_labels); % 0.1 ... displacement so the text does not overlay the data points
title('balloon scatter plot (b)','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ... 2','FontSize',15); hold on;
set(gca,'FontSize',15);

-----------------------------------------------

figure;
% bar(x_axis,1:p-3,plotter_norm,0.5,'stack'); % ... http://www.mathworks.com/help/matlab/ref/bar.html
hold on; grid on;
title('3D bar','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ... 2','FontSize',15); hold on;
legend('criterion 1','criterion 2','criterion 3');
set(gca,'FontSize',15);

figure;
% bar3(plotter_norm,0.5);
bar3(1:p-3,plotter_norm,0.5); % ... http://www.mathworks.com/help/matlab/ref/bar3.html
hold on; grid on;
title('3D bar','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ... 2','FontSize',15); hold on;
axis([0 13 0 12 0 80])
set(gca,'FontSize',15);

figure;
% labels = strcat();
labels = cellstr(num2str((1:p-3)'));
% pie(y_data,ones(1,p-3),pie_labels); % ... http://www.mathworks.com/help/matlab/ref/pie.html
hold on; grid on;
str = {'Item A: ';'Item B: ';'Item C: '};
combinedstrings = strcat(str,percentValues); % text
and percent values

% oldExtents_cell = get(hText,'Extent'); % cell array
% oldExtents = cell2mat(oldExtents_cell); % numeric array
% set(hText,{'String'},combinedstrings);
% newExtents_cell = get(hText,'Extent'); % cell array
% newExtents = cell2mat(newExtents_cell); % numeric array
% width_change = newExtents(:,3)-oldExtents(:,3);
% signValues = sign(oldExtents(:,1));
% offset = signValues.*(width_change/2);
% textPositions_cell = get(hText,{'Position'}); % cell array
% textPositions = cell2mat(textPositions_cell); % ...
numeric array
% textPositions(:,1) = textPositions(:,1) + offset; % ...
add offset
% set(hText,{'Position'},num2cell(textPositions,[3,2])); ...
% set new position

title('pie chart','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
2','FontSize',15); hold on;
set(gca,'FontSize',15);
figure;
area(plotter_norm); % ...
http://www.mathworks.com/help/matlab/ref/area.html
hold on; grid on;
title('area plot','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
2','FontSize',15); hold on;
legend('Location','NorthWest');
set(gca,'FontSize',15);
figure;
scatter3(x_data_norm,y_data_norm,z_data_norm,100,rand([p-3,3]),'fill','MarkerEdgeColor','k','LineWidth',0.5); ...
hold on; grid on;
scatter3_labels = cellstr(num2str((1:p-3)'));
text(x_data_norm,y_data_norm,z_data_norm,scatter3_labels); ... % 0.1 displacement so the text does not overlay the ...
data points
title('3D scatter (a)','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
2','FontSize',15); zlabel('criterion ...
3','FontSize',15); hold on;
set(gca,'FontSize',15);

figure;
scatter3(x_data_norm,y_data_norm,z_data_norm,100,[x_data_norm ... 
    y_data_norm ... 
    z_data_norm],'fill','MarkerEdgeColor','k','LineWidth',0.5); ... 
hold on; grid on;
scatter3_labels = cellstr(num2str((1:p-3)'));
text(x_data_norm,y_data_norm,z_data_norm,scatter3_labels); ... 
    % 0.1 displacement so the text does not overlay the ... 
    data points

hold on; grid on;
title('3D scatter (b)','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ... 
    2','FontSize',15); zlabel('criterion ... 
    3','FontSize',15); hold on;
set(gca,'FontSize',15);

figure;
scatter3(x_data_norm,y_data_norm,z_data_norm,fix(500*(z_data_norm+.1)),[zeros(p-3,1) ... 
    (x_data_norm + y_data_norm + z_data_norm)/3 ... 
    (x_data_norm + y_data_norm + ... 
    z_data_norm)/3],'fill','MarkerEdgeColor','k','LineWidth',0.5); ... 
hold on; grid on;
scatter3_labels = cellstr(num2str((1:p-3)'));
text(x_data_norm,y_data_norm,z_data_norm,scatter3_labels); ... 
    % 0.1 displacement so the text does not overlay the ... 
    data points

hold on; grid on;
title('3D scatter (c)','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); yxlabel('criterion ... 
    2','FontSize',15); zlabel('criterion ... 
    3','FontSize',15); hold on;
set(gca,'FontSize',15);

% make a color index for the ozone levels
% nc = 16;
% offset = 1;
% c = response - min(response);
% c = round((nc-1-2*offset)*c/max(c)+1+offset);

figure;
radarplot(plotter_norm);
hold on; grid on;
title('radarplot','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ... 
    2','FontSize',15); hold on;
set(gca,'FontSize',15);
figure;
[x_data,y_data] = meshgrid(-3:.5:3,-3:.1:3); % ...
   http://www.mathworks.com/help/matlab/ref/ribbon.html
z = peaks(x_data,y_data);
ribbon(y_data,z);
hold on; grid on;
xlabel('X'); ylabel('Y');
zlabel('Z');
title('ribbon','FontSize',15); hold on;
xlabel('criterion 1','FontSize',15); ylabel('criterion ...
   2','FontSize',15); hold on;
set(gca,'FontSize',15);
colormap hsv;

pi=3.14

figure;
radar_redius = 20;
radar_segments = p-3;
radar_labels = cellstr(num2str((1:p-3)'));
for i = 1:radar_segments
   line([0 radar_redius*cos(i*2*pi/radar_segments)']
        [0 ...
        radar_redius*sin(i*2*pi/radar_segments)]','Color',[0.5,0.5,0.5]);
   text((radar_redius+2)*cos(i*2*pi/radar_segments)
        (radar_redius+2)*sin(i*2*pi/radar_segments),
        0.25 ...
        0.25)',num2str(data_comb_nptest_cum(i,1)),'FontSize',5); ...
   hold on; % 'FontWeight','bold'
end
hold on;
for i = 1:radar_segments
   for j = 2:radar_redius
      line([j*cos(i*2*pi/radar_segments) ...
            j*cos((i+1)*2*pi/radar_segments)]',
            [j*sin(i*2*pi/radar_segments) ...
            j*sin((i+1)*2*pi/radar_segments)]','Color',[0.5,0.5,0.5]);
   end
end
hold on;
for i = 1:radar_segments
   x_radar(i,:) = ...
      [radar_redius*x_data_norm(i,1)*cos(i*2*pi/radar_segments) ...
      radar_redius*x_data_norm(i,1)*sin(i*2*pi/radar_segments)];
   y_radar(i,:) = ...
      [radar_redius*y_data_norm(i,1)*cos(i*2*pi/radar_segments) ...
      radar_redius*y_data_norm(i,1)*sin(i*2*pi/radar_segments)];

radar_redius\*y_data_norm(i,1)*\sin(i*2*pi/radar_segments)];
  z_radar(i,:) = ...
  [radar_redius\*z_data_norm(i,1)*\cos(i*2*pi/radar_segments) ...
  radar_redius\*z_data_norm(i,1)*\sin(i*2*pi/radar_segments)];
end

fill(x_radar(:,1),x_radar(:,2),[1,0.4,0.6],'FaceAlpha',0.4);
hold on;
fill(y_radar(:,1),y_radar(:,2),[0,0.5,0.5],'FaceAlpha',0.4);
hold on;
fill(z_radar(:,1),z_radar(:,2),[1,0.9,0.5],'FaceAlpha',0.4);
hold on;
% (gcf,'Color',[1,0.4,0.6])
axis square;
axis off;
end % if part_code(20) == 1

% ...  
========================================================================
% ---------- part 21: fuzzy time score - random subset ...  
=========================================================================

if part_code(21) == 1

  % data_fuzzy = data_cal(1:n,4:p);
  m = 1000; % no. of randomly selected sets
  k = 3; % no. of variable combination out of all variables
  select = [ ];
  data_select_xxx = [ ];
  data_select_xxx(1:p-3,1:2) = 0;
  for i = 1:m
    select(i,:) = randsample(data_var_num(end,:),k,false);
  end
  sum_limit_effect_var = k;
  for i = 1:m % trials
    b_fuzzy = [ ];
    data_select = [ ]; % instructing survival data from ...
    data_select = data_fuzzy(:,select(i,1:k)); % for set #i
    b_fuzzy = regress(data_time,data_select);
    for j = 1:k
      data_select_xxx(select(i,j),1) = ...
  end

187
data_select_xxx(select(i,j),1) + ...
b_fuzzy(j,1); % cumulation of all scores for ...
all variables

data_select_xxx(select(i,j),2) = ...
data_select_xxx(select(i,j),2) + ...
abs((b_fuzzy(j,1))^2); % cumulation of all ...scores for all variables

data_select_xxx(select(i,j),1) + ...
b_fuzzy(j,1); % cumulation of all scores for ...
all variables

end
end

figure;
bar(zscore(data_select_xxx(:,1)),'b','grouped');
hold on; grid off;
title('fuzzy time scores - ...randomization','FontSize',15); hold on;
xlabel('variable','FontSize',15); ylabel('fuzzy time ...scores - randomization','FontSize',15); hold on;
figure;
bar( (data_select_xxx(:,2) - min(data_select_xxx(:,2)) ...)/(max(data_select_xxx(:,2)) + ...min(data_select_xxx(:,2)) ...),'FaceColor',[0.1,0.2,0.3],'EdgeColor',[0.1,0.1,0.1]);

figure;
bar( (data_select_xxx(:,2)),'FaceColor',[0.4,0.4,0.4],'EdgeColor',[0.1,0.1,0.1]);
hold on; grid off;
figure;
load 'figs';
figure;
s(1) = subplot(3,1,1);
bar( (figs(:,1) - min(figs(:,1)))/(max(figs(:,1)) + ...min(figs(:,1)) ...),'FaceColor',[0.3,0.3,0.3],'EdgeColor',[0.1,0.1,0.1]);
hold on; grid off;
xlabel('variable','FontSize',15); ylabel('normalized ...coefficient score','FontSize',15); hold on;
s(2) = subplot(3,1,2);
bar( (figs(:,2) - min(figs(:,2)))/(max(figs(:,2)) + ...min(figs(:,2)) ...),'FaceColor',[0.3,0.3,0.3],'EdgeColor',[0.1,0.1,0.1]);
hold on; grid off;
xlabel('variable','FontSize',15); ylabel('normalized ...coefficient score','FontSize',15); hold on;
s(3) = subplot(3,1,3);
bar( (figs(:,5) - min(figs(:,5)))/(max(figs(:,5)) + ...
min(figs(:,5)) ...
'), 'FaceColor', [0.3, 0.3, 0.3], 'EdgeColor', [0.1, 0.1, 0.1]);
hold on; grid off;
xlabel('variable', 'FontSize', 15); ylabel('normalized ... coefficient score', 'FontSize', 15); hold on;

end % if part_code(21) == 1

%% ...
========================================================================
%% ---------- part 22: fuzzy weighted standardized score ... method -------
========================================================================
%%
if part_code(22) == 1

m = 100; % no. of trials
k = 3; % no. of cluster members
var_num = p-3;
clus_nu = ((p-3)- mod(p-3,k))/k;
clus_nu = fix((p-3)/k); %
var_vector = [];
var_vector = data_var_num(end,:);
var_rand = [];
data_nptest = [];
data_nptest_cum = [];
data_nptest_cum_unique = [];
data_nptest_elem_freq = [];
data_nptest_elem = [];
for i = 1:m % trials
    var_rand = [];
    greedy(clus_nu) = ...       struct('var_num', [], 'data', [], 'var', [], 'test', []);
    var_rand = ...       var_vector(randperm(length(var_vector))); % ... random sort of var numbers
    var_rand = randperm(p-3); % random sort of var numbers
    sum_limit_effect_var = 0;
    for j = 1:clus_nu
        greedy(j).var_num = var_rand(k*(j-1)+1:k*j);
        ...
    ...
% greedy(clus_nu).var_num = ...
    var_rand(k*(clus_nu-1)+1:end)

for j = 1:clus_nu % cluster #j
    [greedy_var_num_row greedy_var_num_col] = ...
        size(greedy(j).var_num);
    kk = greedy_var_num_col; % is equal to k for ...
        regular clusters and it would be less than k ...
            if put remains is an extra cluster
    greedy(j).data = []; % cluster #j
    greedy(j).data = ...
        data_var(:,greedy(j).var_num(1,1:kk)); % ...
            constructing cluster #j (var_num(1,1:k))
    greedy(j).data(:,kk+1) = sum(greedy(j).data,2);
    greedy(j).data(1:n,kk+2) = data_time;

    greedy(j).var = []; % constructing survival data ...
        from cluster #j based on element k+1 (for ...
            time t, select if at least ...
                "sum_limit_effect_var" variables at 1)
    greedy(j).var = ...
        greedy(j).data(find(greedy(j).data(:,kk+1)>sum_limit_effect_var),kk) % remove ineffective failure times from ...
            cluster #j
    greedy(j).var(:,2) = 1; % just as a frequency of ...
        failure time in order to plot kaplan-meier ...
            survival function

    [greedy(j).test(1,1),greedy(j).test(1,2)] = ...
        ranksum(data_time,greedy(j).var(:,1)); % ...
            wilcoxon rank sum test (col k+1 and k+2)
    greedy(j).test(1,3) = ...
        logrank(data_time,greedy(j).var(:,1)); % ...
            logrank test (col k+3)

    greedy_result(1,j) = j;
    greedy_result(2,j) = greedy(j).test(1,1);

end

[max_num,max_index] = max(greedy_result(2,:));
var_rand = greedy(greedy_result(1,max_index)).var_num;
data_nptest = [data_nptest; var_rand];
end % for i = 1:m % trials
[data_row data_col] = size(data_nptest);
for i = 1:data_col % put all selected var numbers in a ...
    single vector
    data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
        data_nptest(:,i);
end

data_nptest_cum_unique = unique(data_nptest_cum); % ...
    elements in data_nptest_cum
[data_nptest_elem_freq,data_nptest_elem] = ...
    hist(data_nptest_cum,data_nptest_cum_unique); % ...
    elements in data_nptest_cum and their frequencies

figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique),'FaceColor',[0.0,0.4,0.4],'EdgeColor',[0.0,0.1,0.1]); ...
% elements in data_nptest_cum and their frequencies - ...
    method 1
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ...
    or logrank frequencies (method 1)','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...% survival comparision od sets (instead of time)
ylabel('censoring','FontSize',15); hold on;

figure;
bar(data_nptest_elem,data_nptest_elem_freq,'FaceColor',[0.4,0.4,0.4],'EdgeColor',[0.1,0.1,0.1]); ...
% elements in data_nptest_cum and their frequencies - ...
    method 2
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ...
    or logrank frequencies (method 2)','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

% survival comparision od sets (instead of time)
end % if part_code(22) == 1

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %------- part 23: M-III1 ...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %

if part_code(23) == 1

m = 100; % no. of trials
k = 4; % no. of cluster members
kmean_clus_num = 5;
var_num = p-3;
% clus_nu = ((p-3) - mod(p-3,k))/k;
clus_nu = fix((p-3)/k); %
data_trans = [];
kmean_result = [];
potentil_function = [];
cost_function = [];
var_rand = [];
var_vector = [];

for i = 1:m % trials
    var_rand = [];
greedy(i,clus_nu) = struct('var_num',[],'data',[]);
data_trans = [];
idx = [];
C = [];
sumd = [];
D = [];
kmean_result = [];
potentil_function = [];
cost_function(i,1) = 0;

    % var_rand = ...
    var_vector(randperm(length(var_vector))); % ...
    random sort of var numbers
    var_rand = randperm(p-3); % random sort of var numbers

    for j = 1:clus_nu
        greedy(i,j).var_num = var_rand(k*(j-1)+1:k*j);
    end

    % greedy(clus_nu).var_num = ...
    var_rand(k*(clus_nu-1)+1:end)

    for j = 1:clus_nu % cluster #j
        [greedy_var_num_row greedy_var_num_col] = ... 
        size(greedy(i,j).var_num);
        kk = greedy_var_num_col; % is equal to k for ...
        regular clusters and it would be less than k ...
        if put remains is an extra cluster
            greedy(i,j).data = [];
        % cluster #j
        greedy(i,j).data = ...
data_fuzzy(:,greedy(i,j).var_num(1,1:kk)); % ...
constructing cluster #j (var_num(1,1:k))
greedy(i,j).data(:,kk+1) = mean(greedy(i,j).data,2);
greedy(i,j).data(1:n,kk+2) = data_time;

data_trans(:,j) = greedy(i,j).data(:,kk+1);
end

[idx,C,sumd,D] = kmeans(data_trans,kmean_clus_num);
% opts = statset('Display','final');
% [cidx_raw,ctrs_log] = ... kmeans(data_trans,5,'Distance','cityblock','Replicates',5,'Options',opts);

kmean_result = horzcat(data_time,idx,data_trans);

for j = 1:kmean_clus_num
cluster_kmean = ...
kmean_result(find(kmean_result(:,2)==j,:),:);
potentil_function(j,1) = mean(cluster_kmean(:,1));
potentil_function(j,2) = mean(cluster_kmean(:,2));
end

for ii = 1:kmean_clus_num-1
for jj = ii+1:kmean_clus_num
  cost_function(i,1) = cost_function(i,1) + ...
  abs(potentil_function(ii,1)-potentil_function(jj,1));
  cost_function(i,2) = i;
end
end

end % for i = 1:m % trials

cost_function_sort = sortrows(cost_function,1);
largest_cost_trial = cost_function_sort(end,2);
greedy(largest_cost_trial,:) = ...

[data_row data_col] = size(data_nptest);

for i = 1:data_col % put all selected var numbers in a ... single vector
  data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
  data_nptest(:,i);
end

data_nptest_cum_unique = unique(data_nptest_cum); % ... elements in data_nptest_cum

[data_nptest_elem_freq,data_nptest_elem] = ...
hist(data_nptest_cum,data_nptest_cum_unique); % ...
elements in data_nptest_cum and their frequencies

figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique),'FaceColor',[0.0,0.4,0.4],'EdgeColor',[0.0,0.1,0.1]); % elements in data_nptest_cum and their frequencies - ... method 1
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 1)','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

figure;
bar(data_nptest_elem,data_nptest_elem_freq,'FaceColor',[0.4,0.4,0.4],'EdgeColor',[0.1,0.1,0.1]); % elements in data_nptest_cum and their frequencies - ...

method 2
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 2)','FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;

% survival comparison of sets (instead of time)

end % if part_code(23) == 1

%% ...
========================================================================
%% ---------- part 24: M-III2 ...
---------------------------------------------
========================================================================
%% ...
========================================================================

if part_code(24) == 1

m = 10; % no. of trials
k = 4; % no. of cluster members
kmean_clus_num = 5;
var_num = p-3;
clus_num = ((p-3) - mod(p-3,k))/k;
clus_num = fix((p-3)/k); %
data_trans = [];
kmean_result = [];
potentil_function = [];
cost_function = [];

194
var_rand = []; 
var_vector = []; 
var_vector = data_var_num(end,:); 

for i = 1:m % trials 
    var_rand = []; 
greedy(i,clus_nu) = struct('var_num',[],'data',[]); 
data_trans = []; 
idx = []; 
C = []; 
sumd = []; 
D = []; 
kmean_result = []; 
potentil_function = []; 
cost_function(i,1) = 0; 

    % var_rand = ... 
    var_vector(randperm(length(var_vector))); % ... 
    random sort of var numbers 
    var_rand = randperm(p-3); % random sort of var numbers 

    for j = 1:clus_nu 
        greedy(i,j).var_num = var_rand(k*(j-1)+1:k*j); 
    end 

    % greedy(clus_nu).var_num = ... 
    var_rand(k*(clus_nu-1)+1:end) 

    for j = 1:clus_nu % cluster #j 
        [greedy_var_num_row greedy_var_num_col] = ... 
        size(greedy(i,j).var_num); 
        kk = greedy_var_num_col; % is equal to k for ... 
        regular clusters and it would be less than k ... 
        if put remains is an extra cluster 
            greedy(i,j).data = []; % cluster #j 
            greedy(i,j).data = ... 
            data_fuzzy(:,greedy(i,j).var_num(1,1:kk)); % ... 
            constructing cluster #j (var_num(1,1:k)) 
            greedy(i,j).data(:,kk+1) = mean(greedy(i,j).data,2); 
            greedy(i,j).data(1:n,kk+2) = data_time; 
        end 
        data_trans(:,j) = greedy(i,j).data(:,kk+1); 
    end 

    [idx,C,sumd,D] = kmeans(data_trans,kmean_clus_num); 

    % opts = statset('Display','final');
% [cidx_raw,ctrs_log] = ...
  kmeans(data_trans,5,'Distance','cityblock','Replicates',5,'Options',opts);

kmean_result = horzcat(data_time,idx,data_trans);

for j = 1:kmean_clus_num
  cluster_kmean = ...
  kmean_result(find(kmean_result(:,2)==j,:),);
  potentil_function(j,1) = mean(cluster_kmean(:,1));
  potentil_function(j,2) = mean(cluster_kmean(:,2));
end

for ii = 1:kmean_clus_num-1
  for jj = ii+1:kmean_clus_num
    cost_function(i,1) = cost_function(i,1) + ...
    abs(potentil_function(ii,1)-potentil_function(jj,1));
    cost_function(i,2) = i;
  end
end
end % for i = 1:m % trials

cost_function_sort = sortrows(cost_function,1);
largest_cost_trial = cost_function_sort(end,2);
greedy(largest_cost_trial,:).var_num
[data_row data_col] = size(data_nptest);
for i = 1:data_col % put all selected var numbers in a ...
  single vector
  data_nptest_cum((i-1)*data_row+1:i*data_row,1) = ...
  data_nptest(:,i);
end

data_nptest_cum_unique = unique(data_nptest_cum); % ...
[data_nptest_elem_freq,data_nptest_elem] = ...
  hist(data_nptest_cum,data_nptest_cum_unique); % ...
  elements in data_nptest_cum and their frequencies
figure;
bar(data_nptest_cum_unique,histc(data_nptest_cum,data_nptest_cum_unique),'FaceColor',[0.0,0.4,0.4],'EdgeColor',[0.0,0.1,0.1]); ...
  elements in data_nptest_cum and their frequencies - ...
  method 1
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ...
  or logrank frequencies (method 1)','FontSize',15); ...
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
figure;
bar(data_nptest Elem, data_nptest Elem_freq,'FaceColor',[0.4,0.4,0.4],'EdgeColor',[0.1,0.1,0.1]); ...
% elements in data_nptest_cum and their frequencies - ...
method 2
hold on; grid off;
title('splitting semi-greedy clustering - wilcoxon rank ... or logrank frequencies (method 2)', 'FontSize',15); ...
hold on;
xlabel('variable','FontSize',15); ...
ylabel('censoring','FontSize',15); hold on;
% survival comparison of sets (instead of time)
end % if part_code(24) == 1
%%% ... 
=========================================================================
% ---------- end of parts ...
------------------------------------------------
=========================================================================