Speeding up calibration of Latent Dirichlet Allocation model to improve topic analysis in Software Engineering

by

Jorge Arturo López

Bachelor of Applied Mathematics and Computer Science, Universidad Nacional Autónoma de México, Mexico, 1991

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2017

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Speeding up calibration of Latent Dirichlet Allocation model to improve topic analysis in Software Engineering

Master of Science 2017

Jorge Arturo López

Computer Science

Ryerson University

Abstract

Extraction of topics from large text corpuses helps improve Software Engineering (SE) processes. Latent Dirichlet Allocation (LDA) represents one of the algorithmic tools to understand, search, exploit, and summarize a large corpus of data (documents), and it is often used to perform such analysis. However, calibration of the models is computationally expensive, especially if iterating over a large number of topics. Our goal is to create a simple formula allowing analysts to estimate the number of topics, so that the top X topics include the desired proportion of documents under study. We derived the formula from the empirical analysis of three SE-related text corpuses. We believe that practitioners can use our formula to expedite LDA analysis. The formula is also of interest to theoreticians, as it suggests that different SE text corpuses have similar underlying properties.

Acknowledgements

I would like to express my gratitude to whom made possible the making of this thesis. Foremost, to my supervisor Dr. Andriy Miranskyy. During my studies at Ryerson, Dr. Miranskyy was an exemplary teacher and advisor. I also thank the examination committee for their valuable comments and suggestions.

Thanks to NSERC for the financial support received through my supervisor. My gratitude extends to RC4 for using the computer hardware that they provide to the program. Also I would like to thank IBM Canada for the opportunity to showcase my research.

Finally, I would like to thank my family for all their support and comprehension to achieve this goal.

Dedication

To my family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Latent Dirichlet Allocation (LDA) is a probabilistic model that can uncover common topics within a group of documents (such as support tickets[1] or defect descriptions), based on the analysis of words extracted from these documents. LDA is employed in a wide variety of areas, ranging from music [38] to robot learning [32]. LDA is gaining popularity in the Software Engineering (SE) community, where it is used in a number of areas including software maintenance [1], traceability [8, 49], a posteriori requirements identification [38], and program comprehension [38]. LDA is attractive because it is unsupervised, requiring no a priori annotations.

1.1 Motivation

Analysis of various *SE-related text corpuses* aids in Software (SW) development. For example, analysis of artifacts created during development can improve SW traceability [9], while mining text of defects and support tickets can help to understand which particular features are causing users the most grief [1]. The former improves code comprehension; the latter enhances business risk analysis, and prioritization of maintenance tasks. My

[1] Reports on specific problems in an issue tracking system, including their statuses and additional significant data that are logged in a call center or help desk.

work is motivated by my industrial experience in SW maintenance, resolving product issues. A representative example is as follows. The software product was returning a cryptic error message if it could not connect to the central data repository (which happened frequently). This error message triggered more than a 1,000 calls from confused clients, overloading support personnel. This problem came to attention a month later, when an analyst read (mined) thousands of support tickets opened within the last month and identified a cluster of tickets associated with the error message. Fixing the problem (thereby satisfying customers and relieving support personnel) was simple: the message text was improved, and documentation associated with the message was enhanced. However, identifying the problem was laborious and time consuming. Other practitioners report similar challenges [1]. To diagnose such problems, the analyst often needs to identify a small set of topics containing a large portion of the documents, so that s/he can recognize "key" topics: e.g., the analyst from our example sifted through thousands of support tickets to find priority problem areas. Fortunately, automatic topic analysis techniques, such as LDA [14] can expedite the process.

1.2 Research Statement

Selecting the optimal number of topics to be identified in the LDA model determines greatly its performance and outcome expected [39]. Thus, the problem can be described as follows: an analyst needs to pick K, such that top X topics would contain a certain fraction F of the documents. For example, pick the value of K, such that top 5 topics would contain 80% of all the support tickets under study. This outcome can be achieved by iterating over the values of K until reaching the desired outcome. This process, however, is resource-intensive.

Probabilistic Topic Models (TMs), such as LDA, require to be provided with the value

of number of topics K. Internally, the model also needs two additional parameters: α and β, that are calculated automatically for each parameter K [8]. It is known that the processing time, necessary to find the optimal values of K by iterating, grows exponentially [85]. Therefore, this thesis addresses the following **research question**: How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?

1.3 Novelty and contribution

We are proposing a formula to quickly estimate K in LDA by means of X, F, and N, which would be invaluable in the realm of software system development and maintenance. For example, practitioners can use the formula to determine a short-list of the most frequently reported product field failures, which, in turn, can inform their decisions regarding corrective product maintenance; businesses can identify the most frequently reported user issues in order to inform their decisions regarding product evolution, and risk analysis; management can obtain a more definitive answer to the question "how many problematic areas does the product contain?", rather than an answer such as "somewhere between 5 and 100 depending on the granularity".

1.4 Organization

The chapters are organized as follows:

1. In Chapter 2, we provide the background for progressing into the subject matter of this thesis.

2. In Chapter 3, we perform a review of the available information that we deem to be more relevant to embrace TMs and LDA.

3. In Chapter 4, we present the foundations of the LDA, its evolution and inner workings.

4. In Chapter 5, we show how the LDA model works using data from a Q&A repository, and we also propose the development of a formula to calibrate it. This involves readily calculating K (number of topics to be extracted with LDA) using this formula.

5. Finally, in Chapter 6, we outline the ideas of this thesis and identify directions of future work.

Chapter 2

Background

This chapter serves as a preamble for further discussing TMs. Here, we will review basic but important concepts that will allow us to better understand the focus of this thesis. The review does not intend to be exhaustive, but only at a level of detail that help us to understand further discussions. These concepts vary from the term frequency - inverse document frequency (tf-idf) scheme to mixture models.

2.1 Document Representation

One method for representing a document is simply as a vector of word counts, for example, let us suppose that we have a document containing just one phrase. After we perform stop word removal and word stemming, we annotate underneath of each word its count (tf) (See Figure 2.1). This concept was introduced in the 50's [56] followed by a term weighting function called (inverse document frequency (idf)), introduced in the 70's [69].

Commonly, documents are described in terms of the topics they represent. For example, documents on databases would contain several words, such as "table", "create", or "select". The purpose of the tf-idf scheme is to find important words in the text corpora. Salton and McGill introduced this scheme in 1983 [72]. They defined tf-idf as the measure of how significant a term in a text corpus is. This significance can be understood as

Document (D) = sara likes both books and politics but in the past she only read books, while robert liked both books and nature but now he only reads books

TERM	sara	like	book	polit	past	read	robert	natur
tf	1	2	2	1	1	2	1	1

D={1,2,2,1,1,2,1,1}

Figure 2.1: Representation of a document as a vector of tf's

the concentration rate of the term into relatively short quantity of documents.

The idf [25] is defined by equation (2.1), where N is the total number of documents in the corpus with $N = |D|$. The term $\{d \in D : t \in d\}$ represents the number of documents where the term t appears (i.e., $tf(t, d) \neq 0$). The tf-idf is given by tf×idf. The higher the tf-idf score, the more important this word is for describing the document.

$$idf(t, D) = \log \frac{N}{\{d \in D : t \in d\}}. \tag{2.1}$$

The applications of the tf-idf scheme are numerous. Ramos [67], exemplifies the use of tf-idf for deciding which terms are more favorable to include in a query. Terms that possesses a high tf-idf indicate that they have a strong relationship with the document where they are located. Therefore, this may imply that if those terms are shown in the query would catch the attention of the user.

Berger et al. [11] proposed several tf-idf based algorithms. For instance, one includes enhancements to improve tf-idf performance and another one implements statistical translation.

Even though it was widely used, tf-idf didn't include sufficient reduction in description of terms length and provided limited document information on statistical structure.

2.2 Document clustering

Clustering textual documents refers to grouping related documents. Unlike a multiclass classification problem (which is a supervised learning task, where we would like to label what a new article is about, and labels are provided in the training examples), the problem of clustering is an unsupervised learning task that consists in uncovering the cluster structure from data input alone [34].

One of the simplest algorithms for clustering is K-means. Here a dataset is divided in K different and non-overlapping clusters. Let us consider $C_1 \dots C_K$ clusters, then two conditions must be met:

1. $C_1 \cup C_2 \cup \dots \cup C_K = 1, \dots, n$ meaning that each observation belongs to at least one of the C_K,
2. $C_K \cap C_{K'} = 0, \forall K \neq K'$.

In this algorithm, the data points are scored by determining its distance from the center of the cluster in question [50].

As documents can be represented as vectors (X_i) we can consider to cluster a collection of them (Z_j). Supposing that our vocabulary consists of only two words for simplicity, then the collection of documents can be clustered as Figure 2.2 indicates. Let us call these clusters, topics. I.e., we can say that Topic 1 is cluster 1, Topic 2 – cluster 2, and so on. Moreover, we may label them according to the semantic contents of the topic; in this case we may say (not the algorithm) that topic 1 is about "Computer Science", topic 2 is about "Genetics", etc.

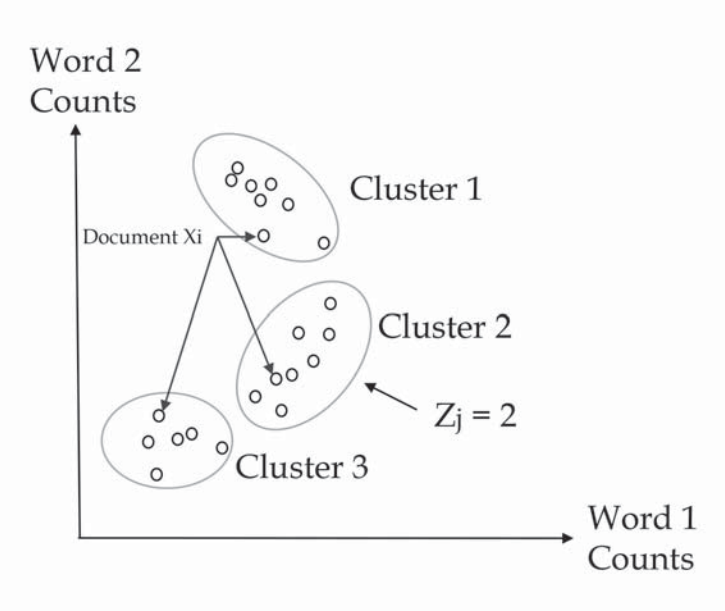


Figure 2.2: Clustering of three documents with a vocabulary of two words, adapted from [34]

2.3 Mixture models

In order to illustrate the reason for using a probabilistic model, let us exemplify the concept of uncertainty in cluster assignments using the K-means algorithm. In Figure 2.3, it is unclear if observation x_i belongs to cluster 1 or cluster 3. The overlapping among the clusters represent uncertainty areas, where we are not positive about the assignment of an observation x_i to a particular cluster.

A mixture model is a probabilistic model that in our case represents the topics of a document as a combination of weighted Gaussian distributions. For instance, let us create a mixture of Gaussian distributions with the respective topic probabilities: π_1=0.55, π_2=0.27, and π_3=0.18 (chosen arbitrarily), represented by the vector $\pi = [0.55, 0.27, 0.18]$ for K=3, where $0 \leq \pi_i \leq 1$ and $\sum_{i=1}^{K} \pi_i = 1$ [34, 42]. This mixture is shown graphically in Figure 2.4.

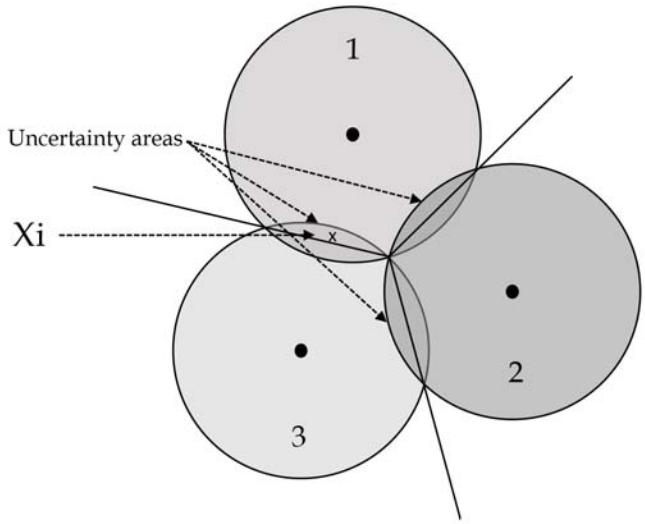


Figure 2.3: Overlapping Clusters, adapted from [34]

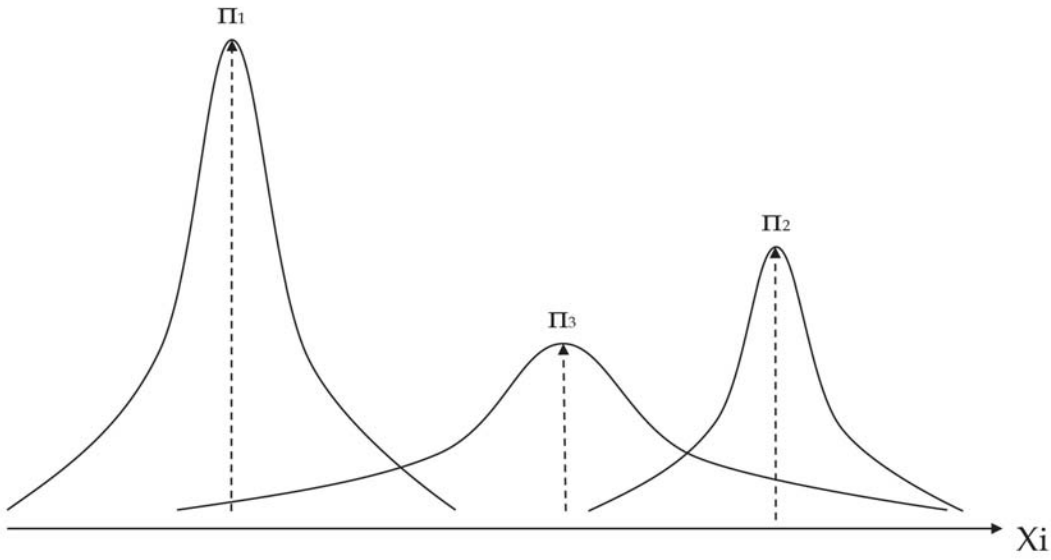


Figure 2.4: Mixture of Gaussian probabilities representing topic proportions, adapted from [34]

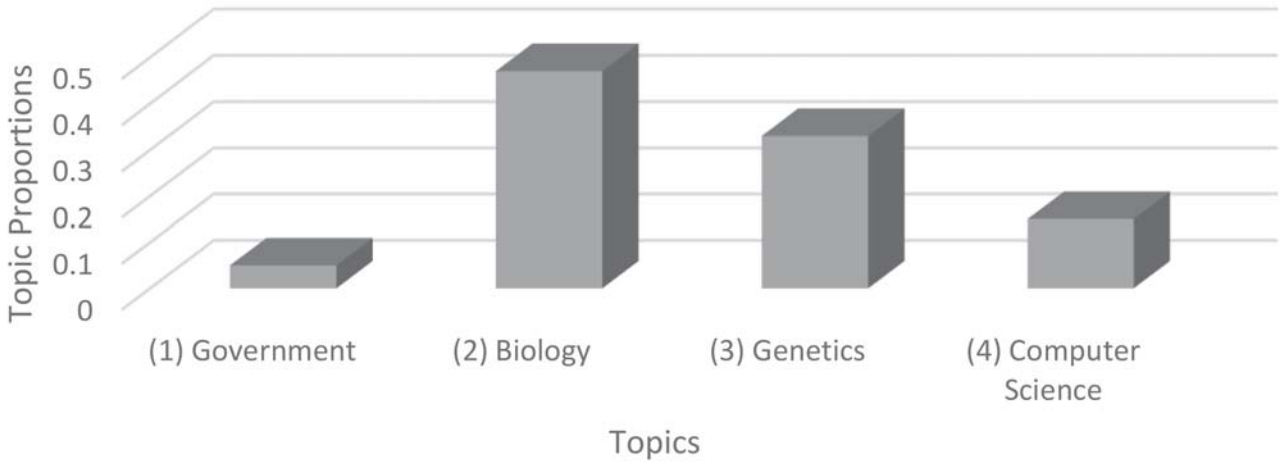


Figure 2.5: Mixture of topics in a document, adapted from [34]

2.4 Mixed Membership Modeling

In the context of document analysis, we have seen that clustering captures the topics that permeate a text corpus in which every document is assigned only to one topic (hard assignment). However, very often, the case is that a document represents more than one topic. We need models that are able to capture the uncertainty when a document may belong to several topics (soft assignment), see Figure 2.5. In that case, the document has membership in topics 1,2,3 and 4, also the relative proportion of occurrence for each topic is captured.

Such models that allow to associate any document with a set of topics are known as *mixed membership models* [18].

2.4.1 Bag of words

Let us consider the case where we extract all the words that are present in a given document. The order of these words are unimportant. Given that this collection of

words may contain multiple occurrences of a unique word we call this a multiset. This representation of a document is called a Bag-of-words (BoW) [88].

2.4.2 Prior probability

We need a clustering model for this BoW document representation. This is where the prior probability (prior) comes into play, as we do need to specify the prior that a given document is related to a specific topic. We express the prior in Equation 2.2 below. It poses the question: what is the probability that the observed word is from topic K (without observing the document content)?

$$p(z_i = K) = \pi_K, \tag{2.2}$$

where z_i is the topic assignment vector, K is the topic number, and $\pi = [\pi_1, \pi_2, \ldots, \pi_K]$ represents the corpus-wide topic prevalence.

In the BoW model representation the words of each document in the collection are going to be assigned a probability of occurrence within each uncovered topic. In this way, we will obtain topic probability vectors over words. For example, if K=4, $z_1 = (0.6, 0.25, 0.15)$, $z_2 = (0.8, 0.12, 0.08)$, $z_3 = (0.9, 0.08, 0.02)$ and, $z_4 = (0.5, 0.33, 0.17)$. Each vector component corresponds to the probability of occurrence of the w_i in z_K. Note that the vectors elements are ordered in decreasing order of probability. For details see [34].

Moreover, we can ask: what is the likelihood of seeing x_i, given that the observation $x_i \in$ topic K? This can be expressed as: $p(x_i|z_i = K, \mu_K, \Sigma_K) = N(x_i|\mu_K, \Sigma_K)$, where $N(\cdot)$ is the Normal distribution with parameters μ_K (mean) and Σ_K (covariance matrix). For more information on statistical and probability concepts, see [71].

Chapter 3

Literature Review

Information Retrieval (IR) using Probabilistic Topic Models, and in particular LDA, is a relative new area of research in machine learning [17]. In the following sections, we review the concept of TMs and their evolution from the tf-idf scheme to the LDA model. We also discuss the selection of number of topics in LDA and take a look into its applications, inside and outside the realm of SE.

3.1 Probabilistic Topic Models

The basic idea behind TMs is that, given a large unstructured text, the hidden topics are discovered. TMs are directed to answer the question: "what is this text talking about?". This concept is illustrated in Figure 3.1 using a toy example (as the text analyzed may contain billions of words [74]). Here, each of the words from the text is assigned to a topic, i.e., a list of related words on the same subject. The output of a TM is the most likely words per each of the most likely topics. We will see later that the number of topics and the number of words are parameterizable.

A more formal definition of TMs, as per [14, 78], is that they are a set of machine learning techniques aimed to extract the thematic structure in a massive collection of documents. These are IR / Data Mining (DM) techniques that use the texts latent

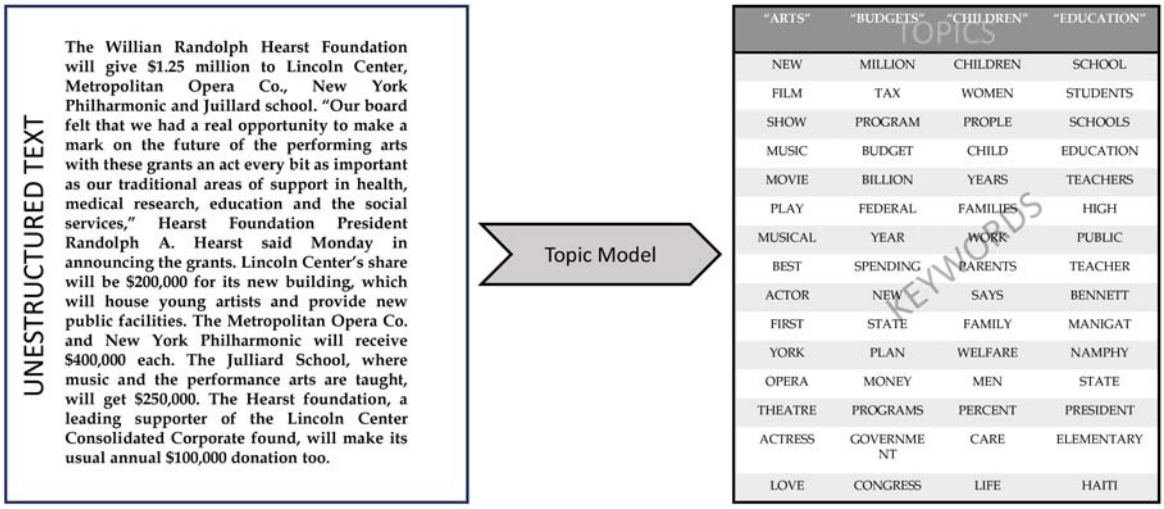


Figure 3.1: A fitting of a small piece of text processed with a Topic Model, adapted from [14]

information for clustering the documents; i.e., grouping them according to the similarity found amongst them.

The above mentioned techniques are algorithms that represent statistical methods to discover, and annotate hidden thematic structure of high volumes of information. All of them follow the same essential concept: a document is a mixture of topics and a topic is a probability distribution over words. A TM represents a generative model for documents [78], given that it stipulates a probabilistic procedure that generates documents. In order to generate a new document, we select a distribution over topics. Following this action, for each of the terms in that document, we select a topic randomly in accordance with this distribution, and then we choose a term from that topic. By applying statistical methods (such as statistical inference) we are able to reverse this process, and deduce the collection of topics that generated the set of documents. In Table 3.1, we show a real inference of five topics that we extracted from the DBA Q&A forum in stackexchange.com using the LDA TM. At the time of extraction, this collection consisted of over 50,000

Table 3.1: Example of obtaining 5 topics and 20 keywords using LDA from the dba.stackexchange.com corpus

TOPIC 1		TOPIC 2		TOPIC 3		TOPIC 4		TOPIC 5	
keyword	**prob.**	**keyword**	**prob.**	**keyword**	**prob.**	**keyword**	**prob.**	**keyword**	**prob.**
server	0.022	index	0.021	tabl	0.044	select	0.023	server	0.018
databas	0.021	row	0.021	null	0.025	date	0.016	databas	0.016
mysql	0.018	queri	0.018	key	0.017	name	0.014	sql	0.014
user	0.013	tabl	0.015	select	0.014	dbo	0.014	use	0.013
use	0.013	select	0.014	creat	0.013	end	0.011	tabl	0.013
sql	0.012	time	0.012	name	0.013	valu	0.011	file	0.013
connect	0.012	use	0.011	column	0.013	sys	0.009	data	0.012
log	0.011	order	0.008	valu	0.012	sql	0.008	innodb	0.012
error	0.011	join	0.008	use	0.011	join	0.008	log	0.011
can	0.010	column	0.007	user	0.010	error	0.008	size	0.010
file	0.009	read	0.007	insert	0.010	set	0.007	can	0.010
master	0.009	product	0.006	row	0.009	object	0.007	queri	0.009
replic	0.008	can	0.006	int	0.009	null	0.006	mysql	0.008
set	0.008	data	0.005	can	0.008	declar	0.006	run	0.007
run	0.007	plan	0.005	data	0.008	type	0.006	will	0.007
creat	0.007	cost	0.005	queri	0.007	count	0.005	transact	0.006
oracl	0.007	date	0.005	default	0.007	procedur	0.005	page	0.006
slave	0.006	scan	0.005	varchar	0.007	varchar	0.005	buffer	0.006
tri	0.006	one	0.005	type	0.007	max	0.005	read	0.006
data	0.006	updat	0.005	primari	0.007	order	0.004	time	0.005

Q&A entries. Here we selected the first 20 keywords (terms) for 5 topics in which they have the highest probability of occurrence under each topic. The keywords of the topics are all related to databases. For instance, for topic 1, the arrangement of the output keywords suggests that the main issue expressed by the users, which posted in this forum (approximately 3,500 posts/ documents in topic 1 for year 2014), is related to how they can connect to the Database Management System (DBMS).

With topic models, we are able to represent a text corpus as a collection of topics that are individually interpretable. These topics provide a probability distribution over terms that discerns clusters of coherent words [20, 78].

Figure 3.2 shows the general output of an inference performed with a TM (such as LDA), which is a matrix $M_{k,t}$, where K are the keywords (rows) and t are the topics

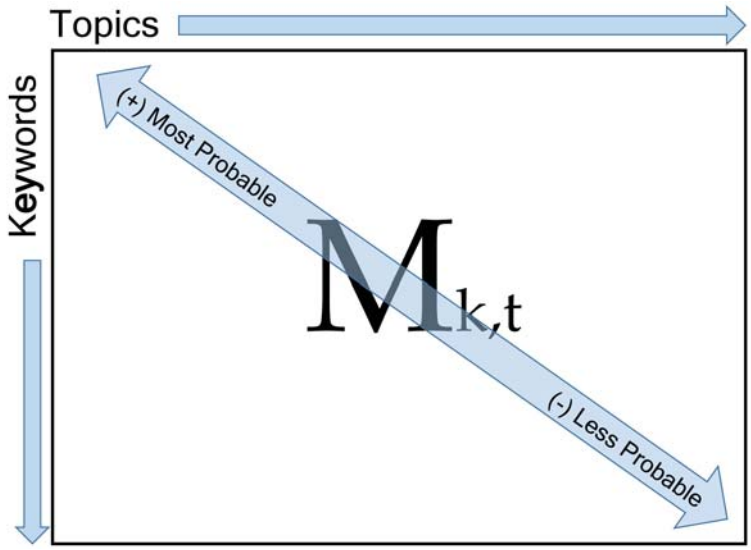


Figure 3.2: Output of an LDA inference

(columns), where $p(K_i) \geq p(K_{i+1})$ and $p(t_i) \geq p(t_{i+1})$.

3.2 Generative Models

A probabilistic topic model is a stochastic model that indicates how it is possible to generate keywords in documents based on latent (i.e. random) variables [78]. The purpose of performing an inference is to uncover the hidden variables that are able to explain the observed ones.

In Figure 3.3 (a) our toy example explains how documents can be generated using a TM. We have only two topics related to 1 (birds) and 2 (machinery). The containers represent BoWs [31]. The first drawing represents three documents that express a different distribution over words. For instance, the documents generated (1 and 3) were sampled exclusively from topic 1 and topic 2 and then the keywords from these documents have $p = 1.0$ [1] of coming from these topics, however, document 2 was generated from a mix-

[1]The value of p is chosen at random, and for illustration purposes

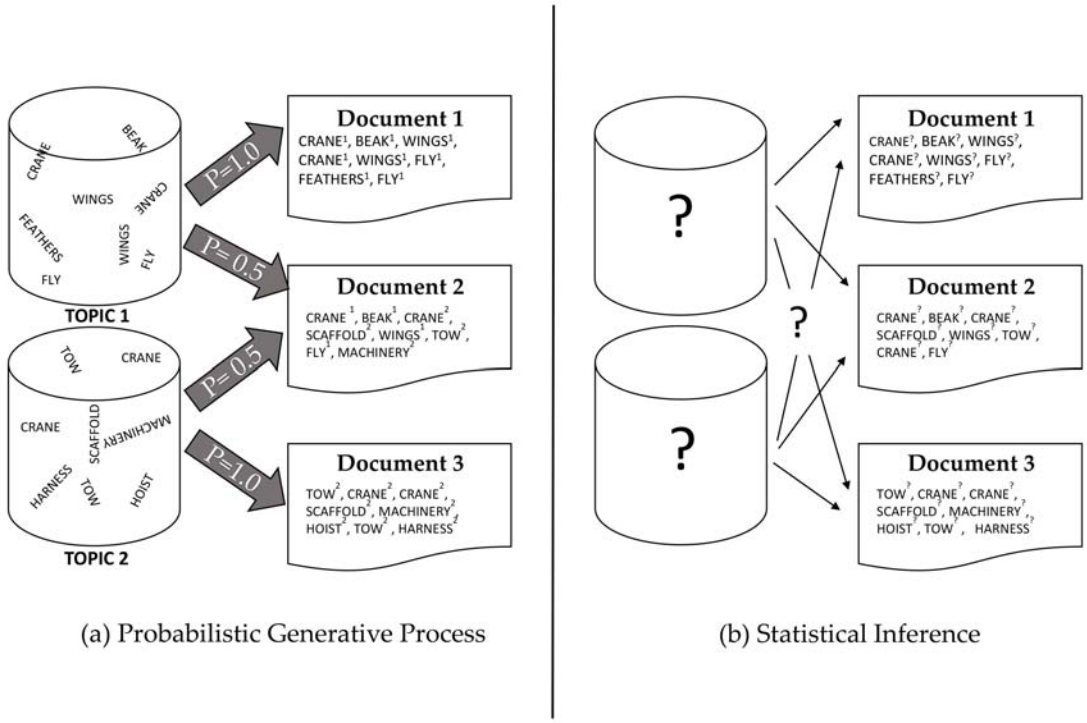


Figure 3.3: Representation of a generative process and the problem of statistical inference, adapted from [78]

ture of topic 1 and topic 2. This means that the keywords generated in each document had $p = 0.5$ footnote[1] of coming from any of the two topics. In Figure 3.3 (b), we are concerned with knowing what are the topics that may have generated the documents, given the observed terms. This is the problem of statistical inference: how to derive the probability distribution over words for each topic, the distribution over topics with each topic, and additionally determining the topics that generated each of the terms? Note that the numbers above the words in the documents indicate what is the topic that sampled the word. TMs are able to capture *polysemy*. For example in Figure 3.3, the term crane can refer to a bird or to a machinery, and both topics can assign high probability to them.

3.3 Evolution of Topic Models

In this section we discuss the progression of TMs from the tf-idf scheme, to the Latent Semantic Analysis (LSI), to the pLSI, and to the LDA model.

- **Latent Semantic Analysis (LSI)** was presented by S. Deerwester [28] in 1990. It is a well-known method for indexing and retrieving the hidden latent semantic structure of terms in a text corpus. LSI was created as a response to deal with the deficiencies of the tf-idf scheme.

  This model handles the problem of *synonymy* and *polysemy*, by collecting information on the semantic content of the documents, not solely on their words. LSI achieves this by implementing the Singular Value Decomposition (SVD) [53, 83] of the large matrix containing term-by-document elements that represents a document collection in an Information Retrieval system.

  The *latent space of topics* in the corpus (i.e., the vector space where the topics are found by the TM) is discovered by using word co-occurrence (i.e., the degree of semantic similarity in documents of the text corpus) [52]. For instance, consider a fragment of the Internet indexed by Google where the term "Database Design" will be close in the latent space for a pair of documents that contain related terms on the subject. However, within an intranet of a software development company two documents will be related exclusively if they share several terms.

  Even though LSI represents an advance in the probabilistic modeling of text, it still does not take into account the probabilistic model at the level of corpus.

- **Probabilistic Latent Semantic Indexing (pLSI)** was introduced by Hoffman [24, 46, 47, 64] in 1999[2]. The pLSI differs from the LSI in that it offers a sound

[2] pLSI model is also known as the *aspect model*.

statistical foundation. This means that it is based on the likelihood principle [11], which details a generative model for generating the words in the documents of the text corpora.

pLSI models each term in a document as a sample that comes from a mixture model. The mixture is represented by the topics (random multinomial variables) modelled as a probability distribution of a fixed collection of topics.

pLSI represents a good method for text analysis. However, it possesses two deficiencies [14]: 1) it consists of a large number of parameters that grow with the size of the corpus (and may lead to another major problem: overfitting), 2) it is not well known how to assign probability to a document that is not contained in the training set.

3.4 LDA model

LDA is the most popular modern TM and is the focus of our discussion. LDA was developed by Blei, Ng, and Jordan in 2002 [6, 13, 14].

LDA is a generative probabilistic model. The ideas behind it are as follows:

- Treat the data from observations arriving from some sort of general probabilistic process, including the structure that we want to find in the data (i.e., hidden variables) that for documents reflect the thematic structure of the collection to which we do not have access.
- Infer the hidden structure by using posterior inference. This refers to the calculation of the conditional distribution of the hidden variables, given the documents under study.
- Allows situating new data into the estimated model.

Therefore, topic models such as LDA, help us to discover topics in a corpus. A topic is a distribution over terms. For example, let us consider an article (i.e. corpus), the different words that compose this corpus are related to a set of topics. Each topic represents a distribution over terms in the vocabulary, and different topics have different words with different probabilities. LDA is used in a broad range of applications covering audio, imaging, computer code, and social networks (please see section 3.7 for more information). In addition, to supply the corpus that we want to analyze with the model, we also need to provide K, which is the number of top topics to be identified.

3.5 Dynamic Topic Models

Unlike static LDA, where it is assumed that the order of the documents to be modeled does not matter, the Dynamic Topic Model (DTM) approach assumes that topics change over time. In this model, the order of the documents is respected, and instead of being a single distribution over words, topics become a sequence of distributions over words. With DTM it is possible to track how a particular topic changes over time [17, 22]. An illustration of a DTM can be seen in Figure 3.4.

3.6 Selection of the number of topics

Selecting the number of topics (K) in the LDA model determines greatly its performance and outcome expected [39], hence the problem identified is as follows: An IT practitioner needs to discover a minor collection of issues including a broad fraction of documents. This need can be formally expressed as determining the value K, in such a way that the top X topics include a portion of the documents. This is challenging, because optimal K values may vary by research domain, and topic specificity increases with K.

Probabilistic topic models, such as LDA, require to be provided with the value of

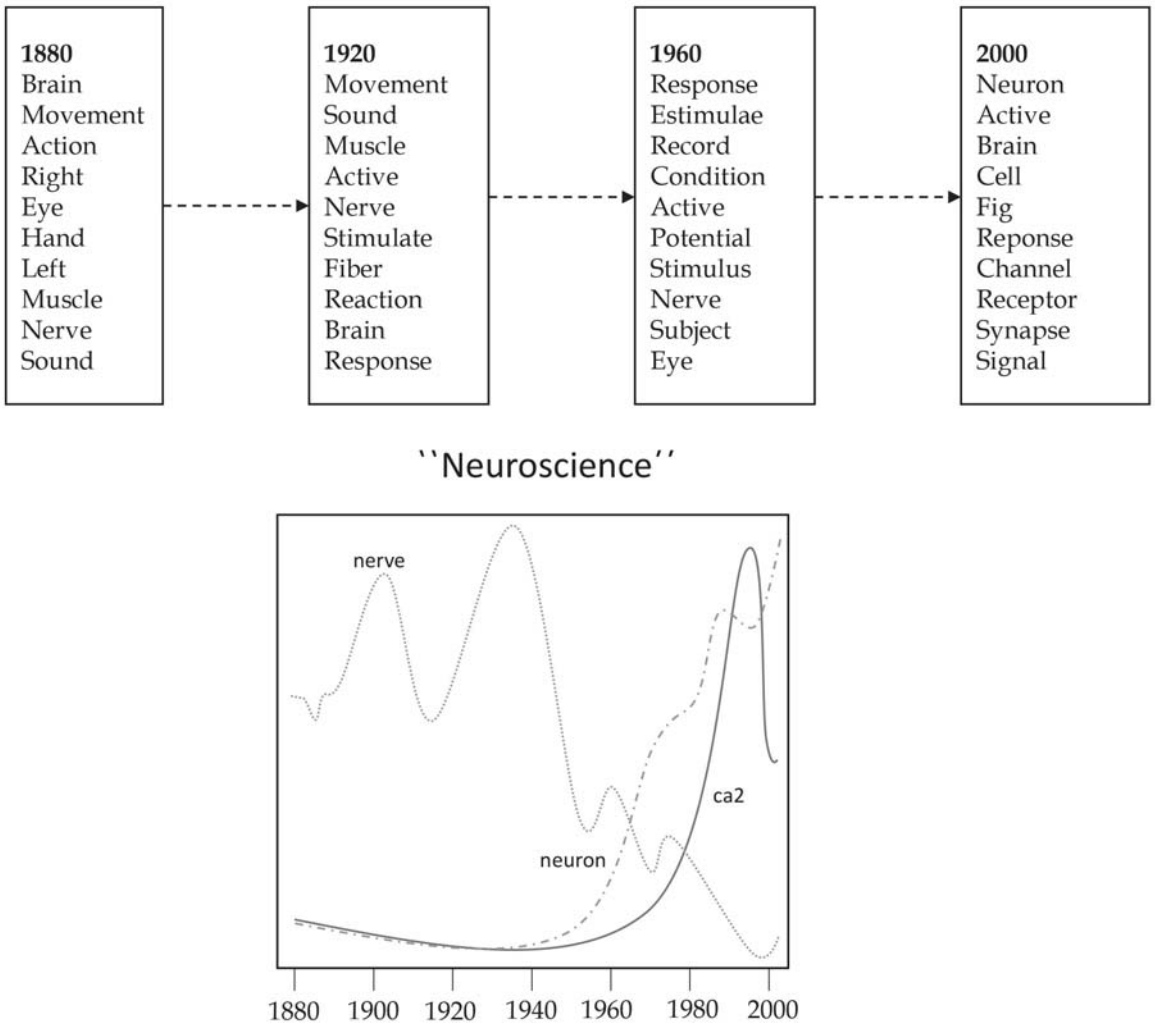


Figure 3.4: Example of one topic extracted from the issues of Science journal published between 1880 and 2000 (every 10 years as shown in [22], but shown every 40 years here). Top panel shows the tracking of ten top words for this topic for a given year; and bottom panel shows how the inferred probability of terms associated with this topic (that was labeled with "neuroscience"by an analyst) varies throughout the years. Adapted from [22].

number of topics K. Internally the model also needs two additional parameters α and β that need to be calculated for each parameter K [14]. There is limited work made on the selection of the number of topics. Many authors, such as Grant and Cordy [38] and Taddy [79], recognize this challenge. Wang et al. [85] point out that the processing time necessary to find the optimal values of this parameter by iterating may be very large.

Taddy [79] identifies three methods for learning the number of topics from data. The first one is Cross-Validation (CV), the second one is non-parametric model such as Hierarchical Dirichlet Process (HDP), the third one is marginal likelihood. Cross Validation (CV) is considered to be the most common method selected by researchers. More specifically, n-Fold CV is a technique to evaluate the performance of the prediction model. Here, the data is split into n segments of about the same size for training while reserving a portion of data (held-out data) for testing purposes. Then the model is repeatedly learned on the training sets data, and evaluated based on their goodness-of-fit to held-out test data, where fitness is based upon a statistical measure, such as *perplexity* or likelihood. Models are evaluated for several K values, and the one with the best fit is selected [2, 45]. However, CV is not a large scale method and the statistical/error information that it provides may be difficult to understand [42]. In the non-parametric technique, hierarchical Dirichlet processes [80], infer the number of topics automatically [45]. In marginal likelihood, the problem of selecting the optimal K becomes an optimization problem to maximize the log likelihood of the data, addressed by using a variational expectation-maximization procedure [14].

Additional techniques that researchers have developed include: Arun et al. [7] who propose a measure for selecting the correct K based on viewing LDA as a matrix factorization mechanism and computing the Kullback-Leibler divergence (KL), and Greene et al. [40] who propose a term-centric stability analysis strategy.

Heuristics is another way to determine the number of topics. Here, researchers identify K with "prior knowledge" of the domain [19]. For example, Zhao et al. [92] proposes a method based on the analysis of variation of statistical *perplexity* [51] during topic modeling.

In practice, researchers typically employ prior knowledge or CV to select K. Some using prior knowledge do not elaborate on their rationale for choosing K, as in [86], where K= 5 topics is used to categorize developer interactions. Others provide some justification; for example, to study topics in the Q&A Website Stack Overflow, Barua et al. [10] chose K= 40 to provide topic sets with medium granularity which implies that the topics represent the general tendency in the data set, while staying different from each other. Yet others report using a similar rationale, but experimenting with several values for K before settling on the optimal value. Some determine the optimal value based on manual inspection from among the experimental K values, as having the best quality topic set [1]. Others employ CV to determine the optimal K. For example, to predict customer behaviours, values of K from 5 to 18 are used on two datasets, with 3- and 4-fold CV, respectively [59]. Each fold's optimal K is identified using the log-likelihood, and one final K for a dataset is calculated as the average over its folds.

In some situations, the optimal K is identified using an oracle. This occurs when the topic model itself is being studied by evaluating its ability to uncover topics and relationships which are known in advance. In such work, e.g. [37, 63, 82], models are created for various values of K, and an optimal model is selected as the one being most accurate according to the oracle.

CV is the most advocated means of model selection, and thus determination of optimal K value [45]. In fact, Arcuri and Fraser argue that n-fold CV is a precondition of achieving scientifically-sound results [5]. For the researcher wishing to use LDA, perform-

ing n-fold CV is often impractical: it is resource-intensive, and the functionality is not available in any toolkit. Acruri and Frasier note that few SE researchers attempt CV; most simply select the model (and thus K value) based on the entire pool of data, calling into question the validity of the results [5] . Without a practical means of performing CV, the researcher must rely on guidelines and heuristics. Such guidance for selecting K is crucial for LDA, since its performance is particularly sensitive to the value of K chosen [45], and there is no single value of K that is appropriate in all situations and all datasets [38, 84].

Selecting quickly the value for K is the objective that we would like to tackle in this thesis.

3.7 Applications of LDA

LDA is ubiquitous and is used in many research fields. For example, for finding patterns in music [44]. In population genetics it is leveraged to discover ancestral populations [17, 33]. In computer vision, for classification and organization of images, where they are analyzed to became collections of "visual words" [17] including analysis of video [87, 88]. In bioinformatics, where the job of the TM is focused on three major tasks: biological data clustering analysis, biological data classification, and biological feature extraction [54]. We will now focus on applications of LDA in the area of SE. We shall note that the application categories explained may overlap between each other (i.e., both Feature Location Technique and Software Traceability have a goal to automate the comprehension of SW artifacts).

3.7.1 Bug localization

This is one of the most common tasks in SE. Here the developers use information about a bug in order to identify the faulty code where the bug resides. Once this have been achieved, they can work on its resolution. There have been previous efforts in IR-based bug localization. For instance LSI-based techniques have been used in source code retrieval that include clustering and feature location [60], however it has been proved that LDA-based technique is more effective [57]. Lukins et al. [58] have proposed a LDA-based technique that performs automated bug localization on Agile software. Essentially, a model is created using LDA by inputting the issue descriptions in the Issue Tracking System (ITS). With the model created, a query (made by the developer debugging the code) based on the issue description of the bug that needs to be fixed, inquires the model and provides the probable location of the piece of code that needs to be fixed. We should note that the query for finding the bug is as effective as the developer debugging the code that creates it. An experienced developer will yield to better results that will outline the bug in a query. Bug localization automation may decrease the costs associated with maintaining SW (i.e., developer effort).

3.7.2 Software traceability

With the growing complexity of SW systems, automated SW traceability has become an essential function in SE. In large companies SW projects often comprise of thousands of SW artifacts (i.e, source code, analysis and design documents, test plans and test cases, bug reports, etc.). The objective of Software Traceability is to identify the relationships amongst these artifacts. Basically, SW traceability creates a cross-reference (by means of links) between the SW and the artifact (e.g., linking source code to use cases). Asuncion et al. [9] identify two categories of SW traceability: retrospective traceability (that

takes place retroactively in static sets of artifacts) and prospective traceability (that takes place on-line, at the moment the artifacts are being produced). They propose a prospective traceability technique based on LDA. Gethers et al. [37] propose an approach that combines orthogonal Information Recovery techniques to recover traceability links in software artifacts.

3.7.3 Feature Location

LDA-based Feature Location Technique (FLT) has the purpose of finding the source code that implements certain functionality. SW systems suffer continuous code changes to adapt to new functionality. Typically, a developer — who is new to the SW — will struggle trying to comprehend it, so s/he can identify the part of the source code where the change needs to be implemented. Given that by using FLT the effort that the developer spent in this task is decreased, the SW cost will also do so. A study of LDA-based FLT is provided by Biggers et al. [12], in which they measure the performance effects of implementing FLT on several Java systems. For a detailed analysis and categorization study of FLTs see [30].

3.7.4 Source code labeling

It is intended to assist the developer in understanding source code with labels, for instance, software visualization tools or a collection of representative terms. Labels provide developers with a "quick look" information to focus on the software component areas which they need to analyze in details to make a decision. These labels can be gathered using an IR method, such as LDA. Panichella et al. [65] investigates the way in which the source code artifact labelling is implemented using LDA.

3.7.5 Test case prioritization

It is recognized that software developers are not always able to execute large sets of tests for every single source code change they have, considering the amount of time and effort that this action would involve. Then, test case prioritization becomes a solution to save time and effort in this process. Thomas et al. [81] propose a technique that uses linguistic data in the test cases (i.e., identifier names, string literals, and comments) to map functionality to each test case. The test cases can then be prioritized for execution.

3.8 Prerequisites of LDA

In order to implement LDA, it is necessary to perform a preparation over the dataset under study. This preparation refers to basically removing "stop-words" (such as "the", "of", or "and"). Otherwise they will pervade the topics that have been learned and may cover semantic word patterns that can be important for us [27].

3.9 Limitations of LDA

Despite being the most popular Topic Model in use, LDA presents the following disadvantages, that some authors have tried to mitigate:

1. Number of topics (K). As discussed above, it is a pre-requisite to provide K by the user in order to implement LDA. As we have mentioned, with a K too small the topics produced would be very wide, whereas a too large K would produce topics that are difficult to differentiate. To overcome this problem, a non-parametric model called Hierarchical Dirichlet processes [80] can be used to learn the optimal K. However, this process is very onerous computationally. Choosing K is a central discussion in this thesis.

2. Labeling. LDA does not provide labelling for the topics inferred. It is up to the user to describe what is the label of the topic that the keywords are referring to. For example, in the inferred topic keywords "school, teacher, blackboard, president, book", it should be labeled as "education". In this regard there are efforts that try to provide labels to topics automatically [61].

3. BoW assumption. Words that need to be inferred together in the same topic (as one word), such as "Royal Canadian Mounted Police", are disaggregated in their component words and assigned to different topics. This is because the BoW assumption allows it. This assumption can be relaxed using LDA extension implementation, such as [15].

Chapter 4

Methodology

We have reviewed the basic concept of topic models including LDA. In this chapter we will develop the LDA model by explaining its mathematical foundations.

4.1 Notation and terminology

In order to start explaining the models, we need to formally define a common terminology, such as words, documents, and text corpus [14]. Our discussion will involve the following:

w – will refer to a term or word in the vocabulary (V). It is the basic unit of discrete data, and $w_i \in V$.

d – will refer to a document defined as a sequence of N words, described by $d = \{w_1, w_2, \ldots, w_N\}$, where w_n represents the nth word in the sequence.

M – will refer to the number of documents in the text corpus, denoted by D, consisting of M documents: $D = \{d_1, d_2, \ldots, d_M\}$.

K – will refer to the selected number of topics.

z - will refer to an unobserved topic that pervades the corpus (there would be K unobserved topics in total).

θ – it is a multinomial distribution used to model the topic proportions.

ϕ – it is a multinomial distribution of z.

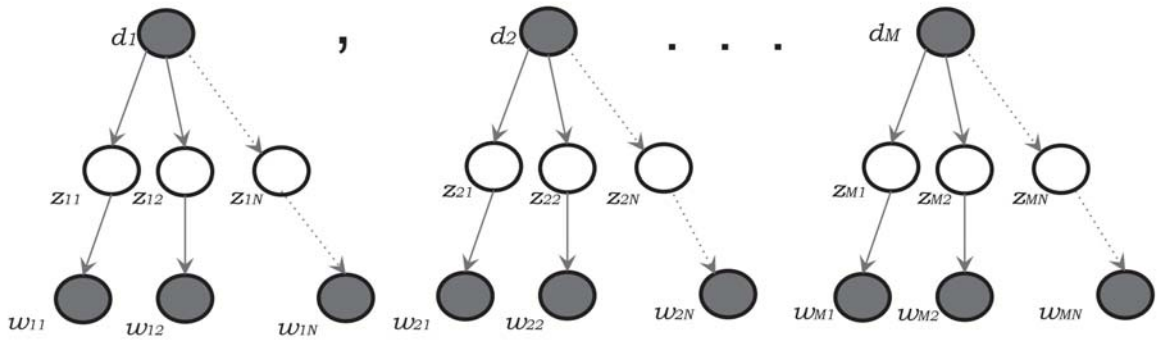


Figure 4.1: Diagram of the pLSI model without using plates notation

α and β are the hyperparameters of the model; α controls the topic distributions per document, while β controls the terms' distribution per topic.

4.2 From pLSI to LDA

The starting point, in order to understand how LDA works, is to continue discussing pLSI. We present in Figure 4.1 the pLSI model in graphical form. Here the nodes represent random variables, the solid ones are observed variables, because we know their outcome. The arrows symbolize dependency, this means that the outcome of the target variable depends on the value of the origin variable [24].

To simplify representation of the models we are going to introduce the concept of rectangular plates to denote multiple repetition in our models. With them we can discard the use of repeating nodes and use indexes, greatly simplifying diagramming of the models. For additional information on plate notation, see [90]. In Figure 4.2 we present the same diagram of pLSI using "plate notation".

The pLSI model establishes that a document d and a word w_n are conditionally inde-

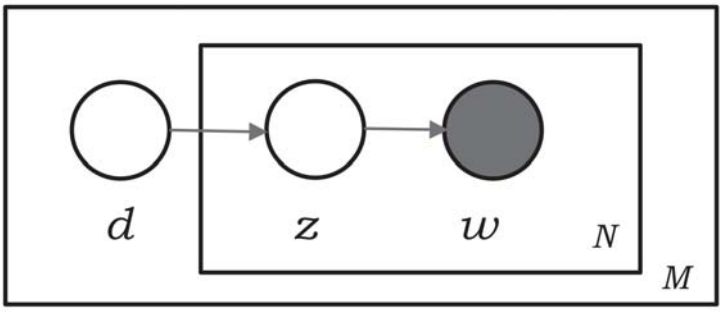


Figure 4.2: Diagram of the pLSI model using plates notation

pendent, given an unobserved topic z, and can be represented as:

$$p(d, w_n) = p(d) \sum_z p(w_n|z)p(z|d). \tag{4.1}$$

pLSI has two major issues [14]:

- Lacks of clarity on how to generate probability of not yet seen documents.
- The number of parameters grows linearly in M, which may lead to overfitting.

LDA addresses these two problems by introducing a K-parameter (K-topic), instead of using a large collection of parameters related to the training set, therefore, eliminating the possibility of overfitting.

4.3 Graphical Representation of LDA and DTM

Now, let us illustrate the graphical models for LDA and DTM.

4.3.1 LDA graphical model

Figure 4.3 illustrates the LDA model in the graphical form. As in the pLSI model, the nodes represent random variables and the solid ones are observed variables. The arcs

also represent dependency. θ's are topic proportions, the z's are the topic assignments and the w's, are the observed words. Note that α and β are called hyperparameters. α governs the amount of smoothing of the topic distributions per document [78], and β determines the amount of smoothing of the word distributions in every topic [78].

The same graphical model can be rewritten in plates notation, as illustrated in Figure 4.4.

4.3.2 DTM graphical model

In Figure 4.5 we can appreciate the DTM graphical model, here as we have reviewed, the model keeps track of topics changing over time [22]. Contrary to static LDA that makes the assumption that documents are exchangable (i.e., their order is unimportant for determining the topics that pervade the documents, and the disaggregation of every document into those topics), it is a sin equa non condition for a DTM that the corpus be organized as a sequential collection of documents. In this Figure each LDA model represents a slice of time and $\beta_{K,t}$ represents a vector for topic K in slice t. In order to make the model work for sequential modeling, the static LDA models are connected (i.e., $\beta_{K,1} \rightarrow \beta_{K,2} \rightarrow \ldots \rightarrow \beta_{K,T}$) and each slice t depends on the previous slice $t-1$.

4.4 Dirichlet distribution

As described in Chapter 3, Blei et al. [14] broaden the pLSI model by bringing in the LDA model and expanding it with a Dirichlet prior on θ. The probability density of a K-dimensional Dirichlet distribution is:

$$Dir(\alpha_1, \alpha_2, \ldots, \alpha_K) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}, \tag{4.2}$$

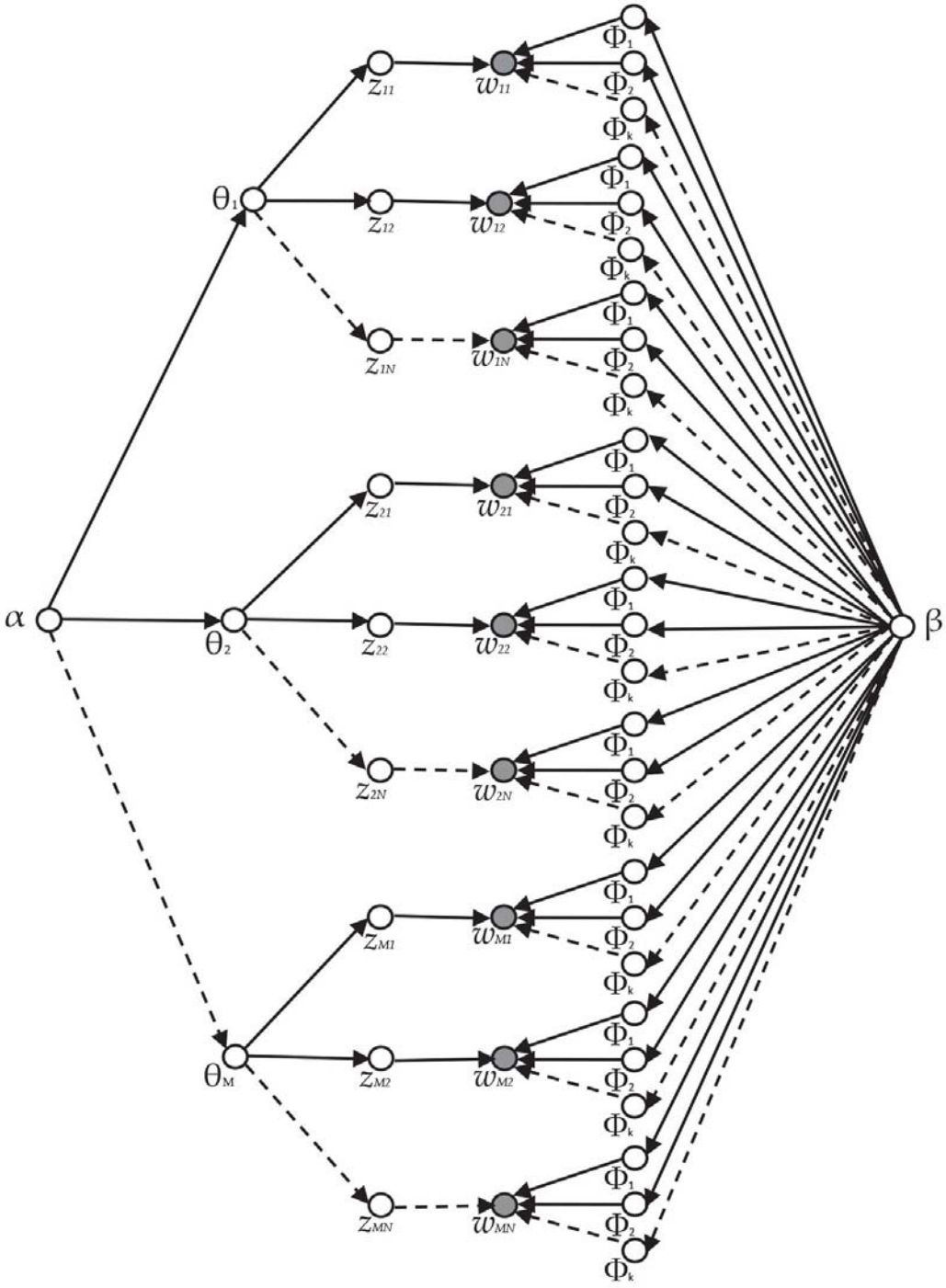


Figure 4.3: LDA graphical model without using plates notation.

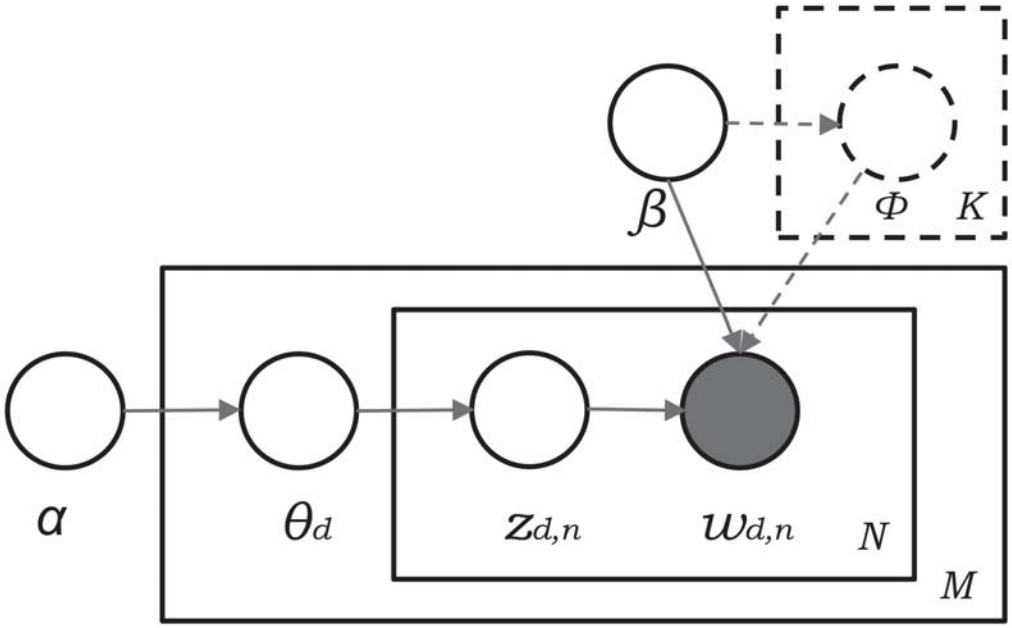


Figure 4.4: LDA graphical model using plates notation, adapted from [14]. The components with the dashed lines refers to the smoothed LDA model. ϕ is a multinomial distribution that is sampled from a Dirichlet distribution with parameter β. This sampling occurs repeatedly for each topic until K topics have been produced.

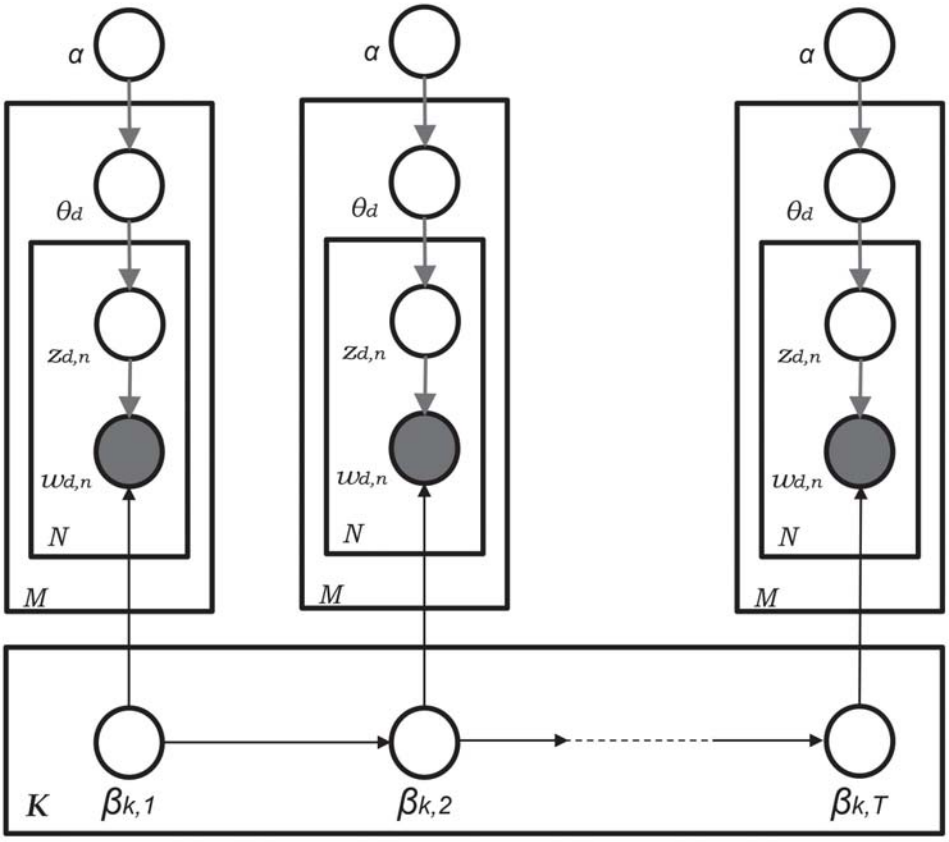


Figure 4.5: Graphical model representation of the DTM, adapted from [13].

where Γ denotes the Gamma function and the α is a K-dimensional vector with elements $\alpha_i > 0$.

Moreover, the generative process of the LDA model can be described in terms of the following joint distribution that includes a topic mixture θ, a set of N topics z, and a collection of N words w [14, 16, 17]:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{K} p(z_n|\theta)p(w_n|z_n, \beta). \tag{4.3}$$

As we have seen, LDA is a generative probabilistic model, where the documents are shown as random mixtures over latent topics and each document is a distribution over words. Generating each document w in a corpus D is done as per Algorithm 1 [14, 89].

Data: words $w \in$ document d
Result: topic assignments z
begin
  foreach *topic z from a Dirichlet distribution with parameter* β **do**
    Choose a multinomial distribution ϕ_z
  end
  foreach *document w* **do**
    Pick a multinomial distribution θ_d from the Dirichlet distribution with parameter α
  end
  foreach *of the N words* w_n **do**
    Choose a topic $z \in 1, 2, \ldots, K$ from the multinomial distribution θ_d;
    Choose a word w from the multinomial distribution ϕ_z;
  end
end

Algorithm 1: LDA algorithm, adapted from [14].

4.5 The Simplex

The Latent Space where the Dirichlet distribution "lives" is also known as the simplex[1]. The Dirichlet variable θ is defined in the $(K-1)$-simplex, θ exists in this space if the following conditions are met: $\theta_i \geq 0, \sum_{i=1}^{K} \theta_i = 1$ [15].

In order to explain this concept, we will start by exemplifying using only words, and

[1] A generalization of the concept of triangles to an arbitrary number of dimensions.

then using both words and topics. Considering only three words (e.g., a, b, and c). For convenience, we will call the $(K-1)$-simplex as the simplex; and in this case it would be a 2-dimensional simplex (or 2-D triangle), with each of the vertices representing each of the words[2]. The outcomes of this multinomial distribution are represented by a vector P of probabilities of the elements a, b and c that sum to 1.

We can start by looking at a probability distribution that occurs in this 2-D Simplex. Let us suppose that a pd of a word situates at vertex of word a (which will denote by A to differentiate a word from a vertex), then $P = \{1, 0, 0\}$. If we situate the pd at the middle of the edge $A - B$, then $P = \{\frac{1}{2}, \frac{1}{2}, 0\}$, if the pd is situated at the centroid of the triangle, then $P = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$, which implies that all three words have the same probability of occurrence. Figure 4.6 illustrates this concept[3].

One of the parameters of the Dirichlet distribution is α, in Figure 4.7 we present several variations of the shape of the distribution, based on different values of this parameter. We can see that in Figure 4.7(a) the distribution places some probability in words B and C, in Figure 4.7(b) it places probability on words A and C, while in Figure 4.7(c) places less probability in A, C and in figure 4.7(d) places probability in A, B. Observe that the peakiness of the distribution is determined by the value of α, if its value is small it means that the Dirichlet is spread out, while if its value is greater means that the distribution is more peaky. Values of $\alpha < 1$ imply increased sparsity, placing pd at the corners of the simplex [16] (see example in Figure 4.8).

Now that the concept of word simplex has been illustrated, we proceed to exhibit the combined representation of probabilities of words and topics in the simplex. LDA contemplates that the word simplex contains K points that constitutes a sub-simplex

[2]If all of the probability density resides in one of the corners, say corner A, then almost certainly (i.e., with probability 1.0) the outcome would be A.

[3]It is not possible for a term in a Dirichlet distribution to have $p = 0$ or $p = 1$, however, for illustration purposes we will allow that.

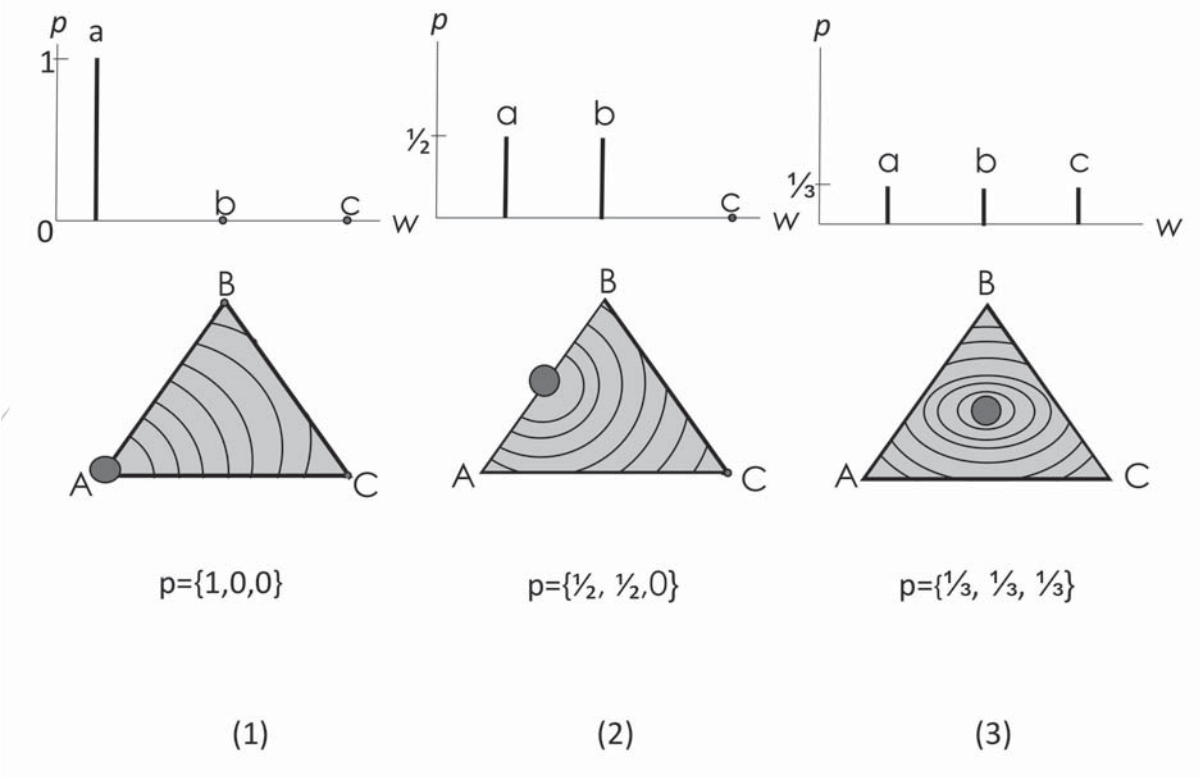


Figure 4.6: Contour plots of different pd's when the weight of the distribution (dark big point) is centred in a particular point. Adapted from [16]

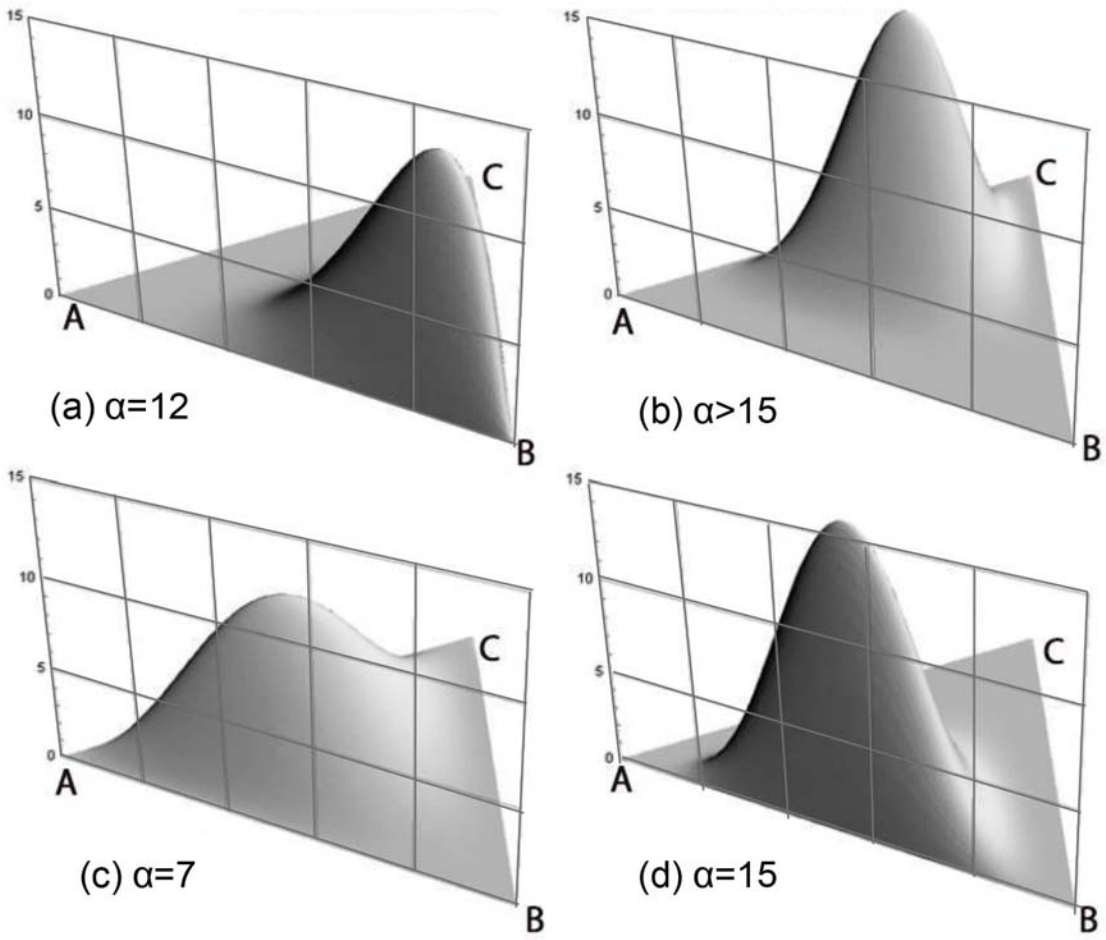


Figure 4.7: Dirichlet distribution with variations of α when weight of the distribution is distributed on the plane between the three words, adapted from [16]

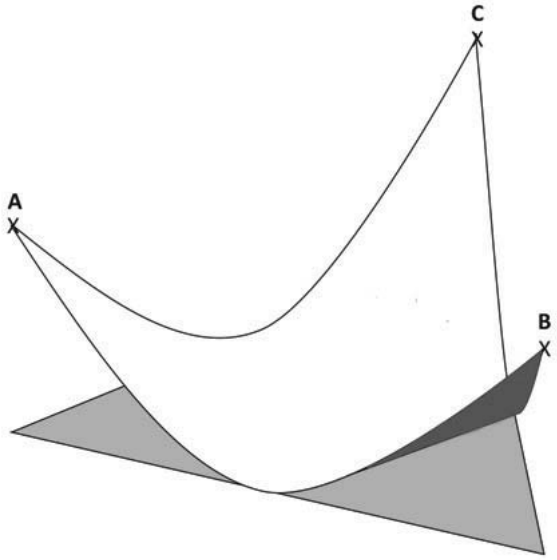


Figure 4.8: Dirichlet distribution with $\alpha < 1$, increased pd at the corners of the simplex

which is known as the topic simplex. Therefore any point in the topic simplex represents a point in the word simplex too. In LDA the words of the observed and unseen documents are generated at random by selecting a topic that is sampled from the smooth distribution with a parameter selected at random in the topic simplex. The LDA states that every word is represented as a distribution of words over a distribution of topics. This is portrayed in Figure 4.9.

4.6 Posterior calculation for LDA

By applying the Bayes' rule to the LDA model, the direct approach for calculating the topic inference equation becomes:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)}, \tag{4.4}$$

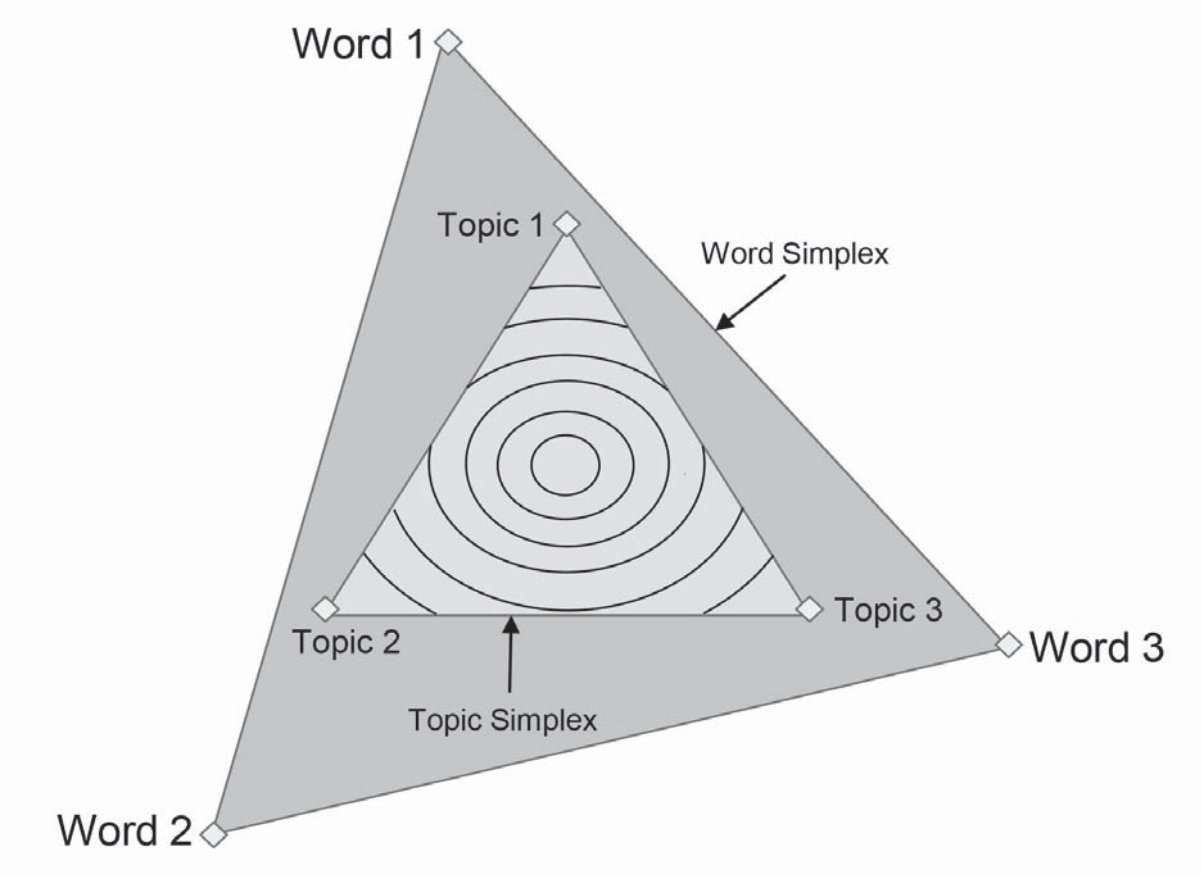


Figure 4.9: Word and topic simplex embedded, adapted from [14]. This geometrical representation of the simplex considers 3 words and 3 topics. Looked perpendicularly from above, the contoured lines represent a smooth distribution placed by LDA. The vertices of the word simplex represent distributions with $p = 1$ while the ones of the topic simplex mean different distributions over words.

where

$$p(w|\alpha,\beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_j)} \int (\prod_{i=1}^{K} \theta_i^{\alpha_i - 1})(\prod_{n=1}^{N} \sum_{i=1}^{K} \prod_{j=1}^{V} (\theta_i \beta_i j)^{w_n^j} d\theta \tag{4.5}$$

is the marginal probability of the observations. Equation 4.5 is defined as the posterior distribution (posterior), and resolving it has become a fundamental problem of statistical inference. However, there is a limitation for computing this distribution: the denominator is intractable. Given this constraint, it is possible to approximate the resolution by using sampling based algorithms and variational algorithms. In the following section we will discuss these techniques.

4.7 Approximation techniques

Here we will discuss two of the available methods[4] for approximating the posterior: CGS and variational inference.

4.7.1 Collapsed Gibbs Sampling

CGS [1, 8, 27, 41, 66] represents a sampling based technique for approximating the posterior of LDA. CGS belongs to the family of algorithms of Markov Chain Monte Carlo (MCMC) [3, 36], and it is straightforward to be implemented, being able to extract from a large corpus a collection of topics. Building a Markov chain using MCMC algorithms, CGS will converge after a number of iterations are executed [27]. It is called "collapsed" because θ and ϕ are marginalized out, and only the latent variables z is sampled [66]. This has the effect of reducing drastically the space that is being explored by the algorithm, and therefore it will have a better performance.

The multinomial parameter for topics in a document θ_j means which topics emerge in document j, while the multinomial parameter for words in a topic ϕ_K indicates which

[4]Note that the LDA R package (that we will discuss in Chapter 5) implements Collapsed Gibbs Sampling (CGS).

words are important for topic K [41].

We would like to use CGS to approximate the variables ϕ and θ, however it is not a direct approximation. These variables can be estimated by using approximations of z (Equation 4.6) [66]. CGS is an iterative process: the start of the process is called "burning" phase, where the Gibbs samples are poor estimates. In this phase, the samples are rejected and the consecutive ones start to estimate the distribution of interest. For more details see [23, 68, 78].

$$p(z_{ij} = K | z^{\neg ij}, x, \alpha, \beta) = \frac{1}{Z} a_{Kj} b_{wK}, \tag{4.6}$$

where $a_{Kj} = N_{Kj}^{\neg ij} + \alpha$, $b_{wK} = \frac{N_{wK}^{\neg ij} + \beta}{N_{K}^{\neg ij} + V\beta}$, and Z is the normalization constant $Z = \sum_K a_{Kj} b_{wK}$.

The superscript $\neg ij$ means that it is omitted from $\sum_j NwKj$. $N_{Kj} = \sum_w N_{wKj}$ and $N_{wK} = \sum_j N_{wKj}$ are counts of the number of times a word w has been assigned to topic K, and the number of times a word in document j has been assigned to topic K, respectively.

One iteration of the CGS consists in choosing a sample for z_{ij} as stated by (4.6) for each term i in each document j.

Taking into account the value sampled for z_{ij} the counts N_{Kj}, N_K, N_{wK} are brought up to date, and with this sample we are able to approximate $\hat{\theta}_j$ and $\hat{\phi}_K$:

$$\hat{\phi}_w K = \frac{N_{wK} + \beta}{N_K + V\beta} \tag{4.7}$$

$$\hat{\theta}_K j = \frac{N_{Kj} + \alpha}{N_j + K\alpha} \tag{4.8}$$

Implementation of CGS for LDA is shown in pseudocode of Porteous et al. [66] in Al-

gorithm 2. After the CGS algorithm is done, the variables ϕ and θ can be approximated. In the algorithm, N is a count variable for keeping track of the number of times any word is assigned to topic K; w is the representation of the text corpus; and z is used to handle the current topic assignment for each of the N words in w. An additional representation of CGS is given by Gao et al. in [35].

begin
  foreach $i = 1$ *to* N **do**
    $u \leftarrow$ draw from Uniform[0,1];
    foreach $k = 1$ *to* K **do**
      $P[k] \leftarrow P[k-1] + \frac{(N_{kj}^{\neg ij}+\alpha)(N_{x_{ij}k}^{\neg ij}+\beta)}{N_k^{\neg ij}+V\beta}$;
    end
    foreach $k = 1$ *to* K **do**
      if $u < P[k]/P[K]$ **then**
        $z_{ij} = k$, STOP
      end
    end
  end
end

Algorithm 2: LDA Gibbs sampling, adapted from [66].

4.7.2 Variational inference

Variational inference is another available technique to infer the posterior [1, 14, 21]. In this case this inference is performed through optimization (which we will discuss below).

Equation 4.9 is often called the variational model, which assumes independence among all the latent variables and it is the result of simplifying the graphical model of LDA, where only the nodes θ and z have been left, and nodes γ and ϕ (variational parameters) have been inserted (see Figure 4.10).

$$q(Z, \theta|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(Z_n|\phi_n), \tag{4.9}$$

where γ is a Dirichlet parameter and $\phi = (\phi_1, \ldots, \phi_n)$ are the multinomial parameters.

The problem of variational inference becomes a problem of optimizing the variational

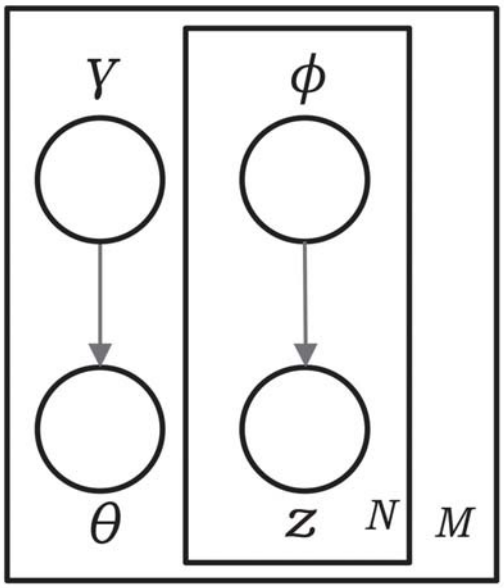


Figure 4.10: Diagram of the LDA variational model, adapted from [1]

parameters γ and ϕ. This can be achieved by minimizing the distance between the true model, $p(\theta, Z|\alpha, \beta)$, and the variational model (for instance using the KL [43]):

$$\min_{\gamma,\theta} \text{KL}[q(\theta, Z|\gamma, \phi) \parallel p(\theta, Z|\alpha, \beta)]. \tag{4.10}$$

This optimization problem is not tractable, however it can be estimated iteratively [1] by using the following two equations:

$$\gamma_i = \alpha_i + \sum i =_1^K \phi_{ni}, \tag{4.11}$$

$$\phi_{ni} \propto \beta_{iw_n} exp[\Psi(\gamma_i)], \tag{4.12}$$

where ψ is the bi-gamma function.

We present a variational inference procedure in Algorithm 3 as per [14].

initialize $\Phi_{ni}^0 \longleftarrow 1/K$ for all i and n;
initialize $\gamma_i \leftarrow \alpha_i + N/K$ for all i;
repeat
  for $n = 1$ *to* N **do**
    for $i = 1$ *to* K **do**
      $\Phi_{ni}^{t+1} \longleftarrow \beta_{iw_n} \exp(\Psi(\gamma_i))$;
    end
    normalize Φ_n^{t+1} to sum to 1;
  end
  $\gamma^{t+1} \longleftarrow \alpha + \sum_{n=1}^{N} \Phi_n^{t+1}$;
until *convergence*;

Algorithm 3: Variational inference algorithm, adapted from [14].

Chapter 5

Implementation and Calibration

In this chapter we will perform analysis in the following freely available datasets from the Questions and Answers (Q&A) forums of stackoverflow.com: android, dba and salesforce. StackOverflow consists of the StackExchange network of Q&A sites, which are forums for professional and enthusiast programmers that can post questions and answers on a forum associated with a particular product(s). First, we will start by exemplifying the usage of LDA in the dba dataset only, for the sake of brevity. Second, we will assume the problem of finding a formula to quickly select the number of topics K for implementing LDA. The programming was done using the LDA implementation in the R language package, with the exception of one script programmed in Perl.

5.1 Processing of datasets

All our datasets will undergo a common processing comprised by the following steps (see Figure 5.1) which include:

Figure 5.1: Process of data for performing a LDA inference

1. Data Extraction. The text corpus used was the dba forum of StackExchange (`https://dba.stackexchange.com`). The dataset was extracted from `https://archive.org/download/stackexchange` [4].

2. Data Conversion. The file from the repository was in Extensible Markup Language (XML) format and for performing the LDA inference we needed the data in Comma Separated Value file (CSV) format; therefore, a Perl script for doing the conversion was created: "xml2csv.pl" (see Appendix B.1). This script keeps the text for all the questions but eliminates those answers that have less than three votes and were not officially accepted as an answer (by the person who asked the question).

3. Data Cleansing. This step involved the following actions:

   (a) Converting to lowercase all words.

   (b) Removal of punctuation signs.

   (c) Removing stopwords (i.e., most common words in English, such as "and", "the", or "a").

   (d) Stemming applicable words (i.e., reducing words to their word stem, such as "computers" and "computing" to "comput").

   (e) White-space char elimination.

(f) Removing terms which have tf-idf smaller than the median of all the tf-idf values. I.e., we eliminate the terms with low "importance".

4. LDA inference. Here is where the LDA is applied to the data. This processing is particular to what our objective is. In our case, we have two types of processing, the first one is to illustrate the usage of LDA by generating a matrix of topic-keywords (similar to Figure 3.2), and the second one is to generate term frequencies to perform analysis and deduce the formula for calibrating the model.

5.2 Mining of Q&A forums

The volume of Q&A in stackexchange.com forums varies from forum to forum. In the case of database-administrators-site (extracted from `https://dba.stackexchange.com`) [76], there were approximately 50,000 entries at the time we downloaded the data. The file that we extracted included several periods by year and month. This repository contains unstructured data for both questions and answers. One of the managers in charge of the product might be wondering what the users are talking about it. Then, LDA can be applied and its output provide the necessary analytics to help responding this question.

All LDA-related computations are performed using R-package 'topicmodels' [48] (which in turn uses the C code implementation of LDA by Blei et al. [14]).

For performing the analysis to illustrate the usage of LDA, we follow the processing of the dataset as depicted in Figure 5.1. In this case the "LDA Inference" step was accomplished by the developed program "analysis_db2_idf.R" (see Appendix B.2) which involves these specific tasks:

1. Extraction of data will be by year (2015) and by their respective months.
2. The number of topics to be extracted will be 5, and the number of keywords per topic will be 20 (specifically for this example).

3. The probability per topic/keyword will be outputted.

4. The idf will be calculated.

The output of this LDA inference for each of the five topics would produce a list of keywords, their probability of belonging to a given topic, number of documents, and idf. Example of an output for topic 1 is given in Table 5.1.

Table 5.1: Topic 1 of the LDA inference of 5 topics and 20 keywords

KEYWORD	PROBABILITY	#doc	idf
server	0.022	1582	0.214
databas	0.021	1497	0.238
mysql	0.018	710	0.562
user	0.013	948	0.436
use	0.013	1631	0.200
sql	0.012	1151	0.352
connect	0.012	874	0.471
log	0.011	714	0.559
error	0.011	959	0.431
can	0.010	1582	0.214
file	0.009	829	0.494
master	0.009	407	0.803
replic	0.008	481	0.731
set	0.008	921	0.449
run	0.007	1036	0.397
creat	0.007	902	0.458
oracl	0.007	393	0.818
slave	0.006	288	0.953
tri	0.006	1018	0.405
data	0.006	847	0.485

For better appreciation of the data, let us render the data in this table in the graphical format. Figure 5.2 shows the probability of occurrence in descending order for each of the keywords in topic 1. Figure 5.3 shows the number of documents[1], where each of the keywords appears. The last plot in this sequence is Figure 5.4; it shows that the closer the value of idf to 0 is for a term, the more widespread the term is, and vice versa.

Now, let us plot the keywords of topic 1 to show its idf and probability as Figure 5.5

[1]The overall number of documents for topic 1 is 2587 (N).

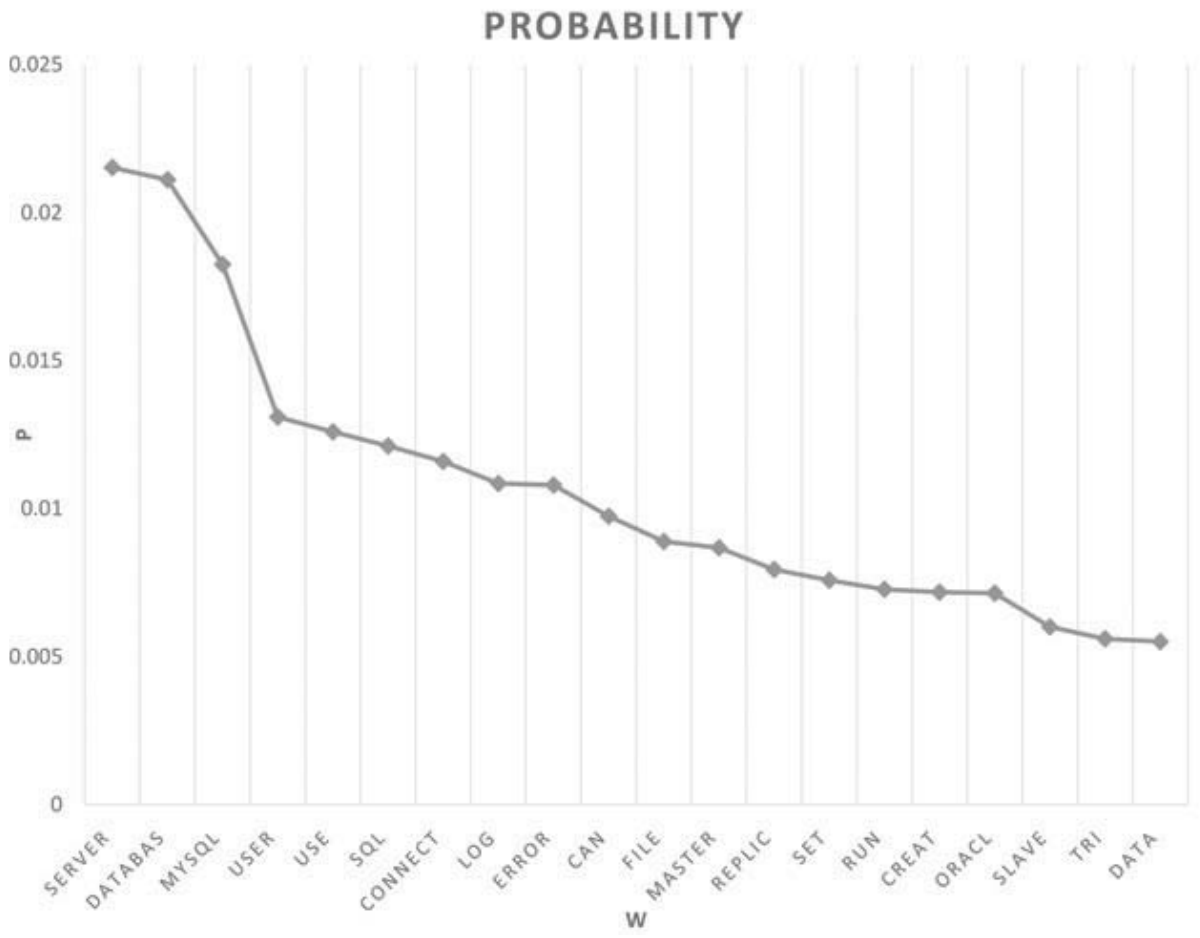


Figure 5.2: Probability per term for Topic 1

indicates. The figure shows that the most important and widespread words are ‘server‘ and ‘databas‘ and the the least widespread ones are ‘master‘, ‘oracl‘, ‘slave‘. Eyeballing the overall set of keywords, we may conjecture that this topic aggregates Q&A related to master/slave – replication connect/error – how to connect to db? etc.

We would like to mention that a manual validation of the subsets of documents has taken place.

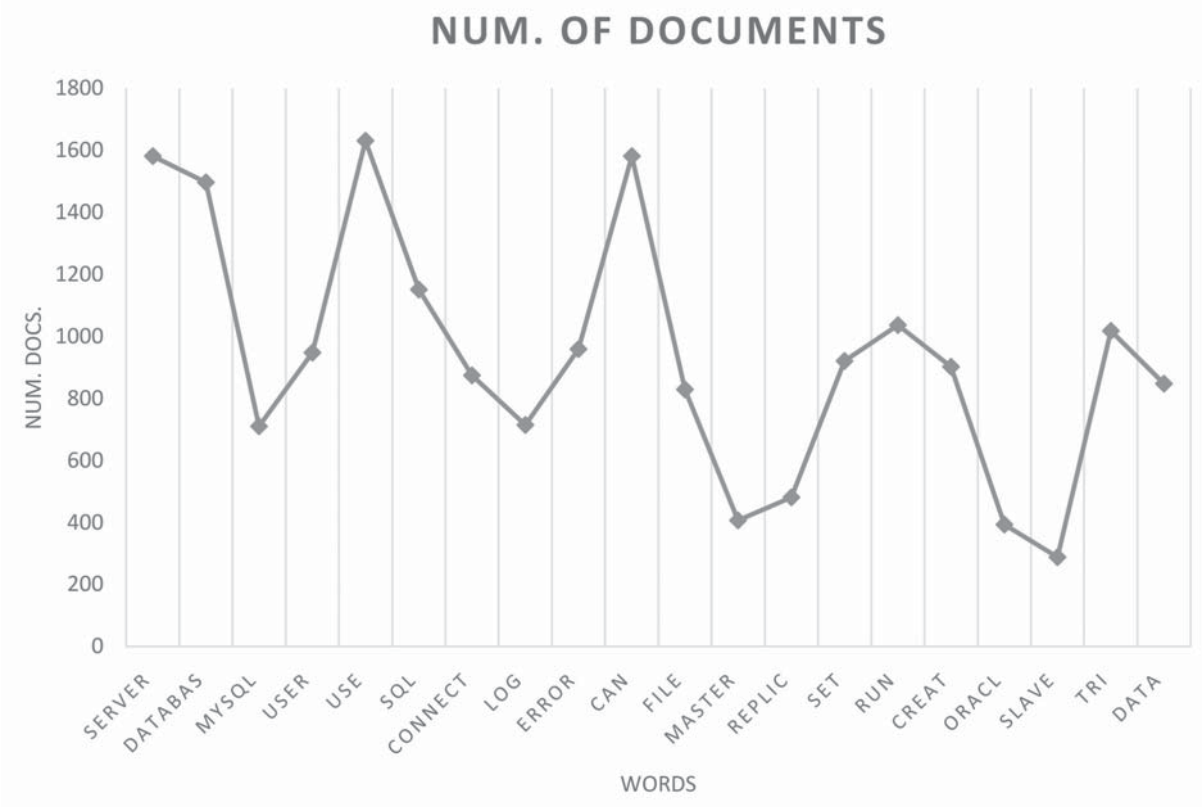


Figure 5.3: Document frequency per term for topic 1

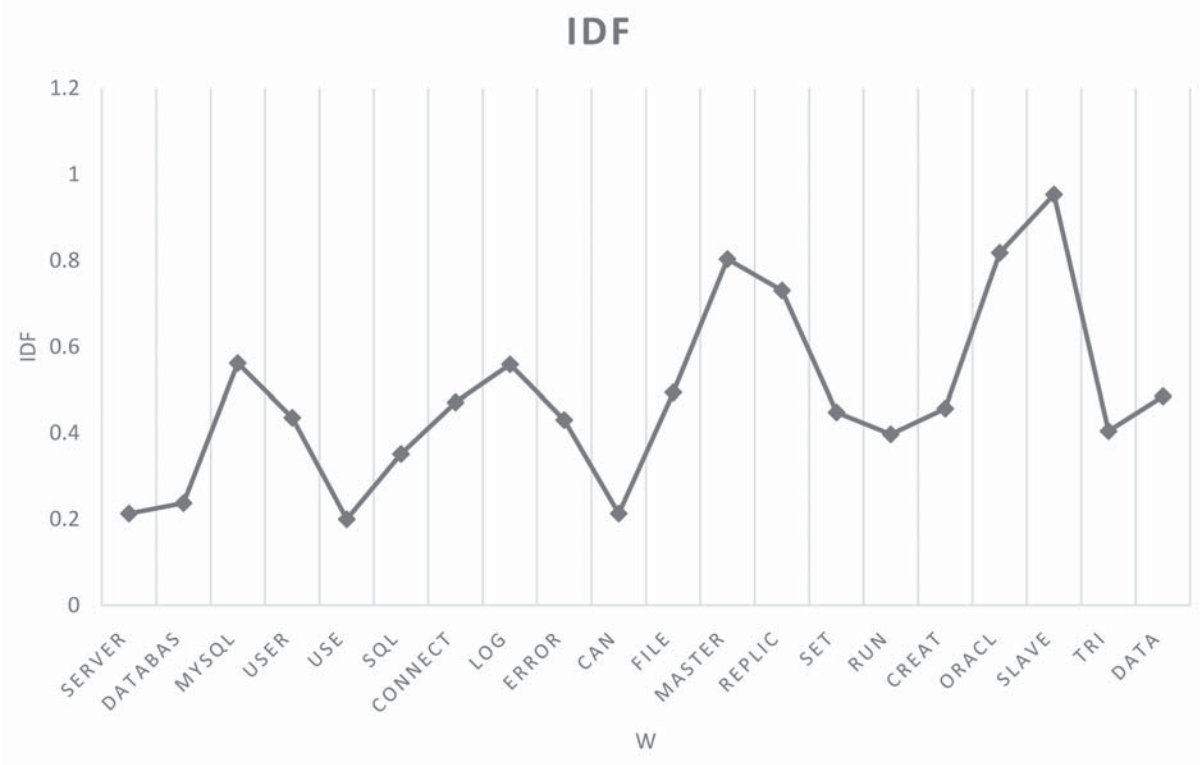


Figure 5.4: Keywords vs idf for topic 1.

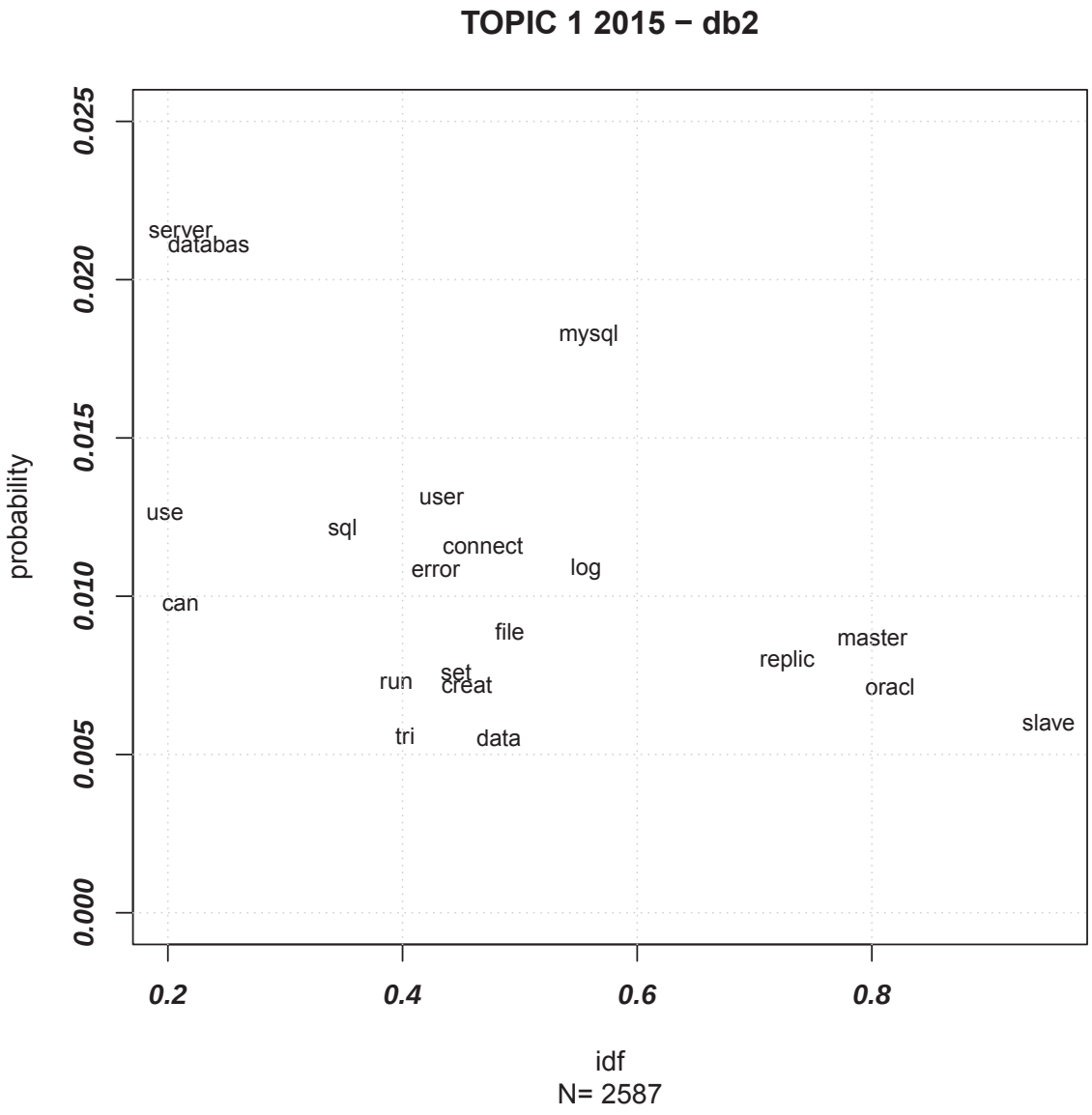


Figure 5.5: LDA inference for topic 1 showing terms plotted against its idf and probability.

5.3 Calibration

In this section we will develop our proposed model for calculating in a straightforward way the number of topics for implementing LDA. Calibration of LDA can consume a large amount of computing resources. Steyvers and Griffits [78], agree on the idea that selecting the number of topics influences greatly the understandability of the results produced by LDA. A K which is too small can produce very broad topics, and one which is too large may affect the interpretability of the topics. Our aim is to create a simple formula for readily calculating the number of topics, therefore addressing our research question:

How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?

Note that we do not speed up computation of a particular LDA model. Rather, we drastically reduce the number of LDA models that have to be computed and calibrated. In the worst-case scenario, one will need to compute up to N models, while usage of our formula reduces the number of models to, at best, one. In practice, given that our approximation is not perfect, one may have to compute a couple ofLDA models to obtain the desired value of F.

5.3.1 Datasets

We will use for our study the three downloaded datasets from stackoverflow.com. They are:

1. android (android.stackoverflow.com)[75]
2. database administration(dba.stackoverflow.com)[76]
3. salesforce (salesforce.stackoverflow.com)[77]

All of them can be downloaded at `https://archive.org/download/stackexchange` [4]. In Table 5.2 we show a summary of these datasets.

Table 5.2: Datasets considered

Dataset	Selected data from-to (period)	Number of questions
android	2014-2015 (monthly)	15765
dba	2012-2014 (monthly and quarterly)	39174
salesforce	2014-2015 (monthly)	25965

As we plan to process several periods for these datasets, the expanded table of processed datasets is as Table 5.3 illustrates.

Table 5.3: Expanded list of number of subsets processed by dataset. Year refers to the yearly data that is available to be extracted from the corpus.

Dataset	Periodicity	Year	Num. of files	Num. of subsets
android	monthly	2014	12	588
android	monthly	2015	12	588
dba	monthly	2012	12	588
dba	monthly	2013	12	588
dba	monthly	2014	12	588
dba	quarterly	2012	4	196
dba	quarterly	2013	4	196
dba	quarterly	2014	4	196
salesforce	monthly	2014	12	588
salesforce	monthly	2015	12	588
		Totals	**96**	**4704**

5.3.2 Processing of datasets for model calibration

In order to generate the information to perform our analysis for deducing the formula to calibrate the model, we will execute the processing of datasets as depicted in Section 5.1. The specific step corresponds to the LDA inference consists in processing each considered dataset (android, dba, and salesforce) for several periods: years-months/quarters, comprising 96 data files. Moreover we consider each of these yearly files to be divided into

temporal subsets (i.e., 12 for monthly and 4 for quarterly subsets) times the number of top X we computed (49, i.e. $X = 2, 3, \ldots, 50$). We will produce for each dataset/period the topic frequency for the words contained in them. For each dataset, we will calculate the distribution of the number of posts per topic, with $K = 2, 3, \ldots, N$. For each K and for each top topics $X = 2, 3, \ldots, 50$, we will calculate the empirical value of F as follows:

$$F = P/N, \tag{5.1}$$

where P is the overall number of documents in top X topics. We achieve this, by running the developed programs: "do_lda_analysis_topics.R" (See Appendix B.4) to process by year-monthly and the program "do_lda_quarter_inc.R" (See Appendix B.5) to process by year-quarterly.

Once we have generated the frequency files for each dataset-year-period (monthly/quarterly), we proceed to bind all of these files in one file, we achieve this by running the developed program "convVar_k200.R" (See Appendix B.7)

After doing this we commence our analysis for inferring our formula. The developed scripts for supporting this analysis are: "verifyFit_k200.R" (See Appendix B.8) and – "verifyFit_k200.R_analysis.R" (See Appendix B.9) for generating the performance plots.

5.3.3 Time consumption to iterate over K

Figure 5.6 illustrates the processing time to iterate over K in order to determine the number of topics for the model. Here, we have plotted the times needed to compute LDA model for a given K, while $K = 2, 3, \ldots, N$. We arbitrarily chose the Android dataset, for January of 2014. This data subset has $N = 614$. Notice that time to compute the LDA model grows (non-monotonically) with the increase of K. The maximum time for computing the model (when $K = 602$) is ≈ 6690 seconds; the overall time to compute

all 613 models for this subset is $\approx$ 1276576 seconds (or $\approx$ 15 days). The times shown here are for one subset; however, the behaviour is similar for all the subsets under study. Thus, it is beneficial to reduce the number of calibrations of LDA models.

5.3.4 Creating fitted values

We notice that the graph of $\log(F)$ against $\log(K)$ for a given X forms a straight line (e.g., see Figure 5.7), suggesting Power Law relationship between F and K [55], [62]. The law takes form

$$F = aK^b, \tag{5.2}$$

where a and b are some constants. Log transformation of Equation 5.2 yields:

$$\log(F) = \log(a) + b\log(K). \tag{5.3}$$

Eyeballing Figure 5.7 suggests that the values of $\log(a)$ and b vary (at least with the change in X). The question then becomes: can we estimate the values of a and b for a given dataset based on some attributes of the dataset?

Given that N and X are (almost) readily available to us, let us assume that a and b are governed by N and X. In order to find $a(X, N)$ and $b(X, N)$, we will compute empirical values of a and b, denoted by $\hat{a}$ and $\hat{b}$, respectively, by fitting linear regression model (Equation 5.3) to the data points for each dataset and for each value of X (we set $X = 2, 3, \ldots, 50$).

We will consider two models: 1) a more flexible one, where a and b are independent of each other and 2) a more constrained one, where $a = X^{-b}$. The second model is obtained via ansatz (an educated guess).

Further exploratory analysis shows that if $K > 0.75N$ or if $K > 200$, then the linear relation between $\log(F)$ against $\log(K)$ breaks (example is shown in Figure 5.8). From

practical perspective, an analyst would rarely be interested in a large number of topics K. Thus, if we filter out data for large values of K, then we may improve the fit of linear models while preserving the models' practicality. We will try two different filters, hence the two datasets:

1. Dataset 1: retain the data if $K \leq 0.75N$;
2. Dataset 2: retain the data if $K \leq 0.75N$ and $K \leq 200$.

We discuss in the following sections the fit of the flexible and the constraint models. Comparison of the models is given in Section 5.4.

Flexible Model

In the flexible case, Equation 5.2 becomes

$$F = a(X, N)K^{b(X,N)}. \tag{5.4}$$

Solving it for K yields:

$$K = \left[\frac{F}{a(X, N)}\right]^{1/b(X,N)}. \tag{5.5}$$

The value of N can be easily extracted from the dataset. The values of F and X are provided by an analyst. However, we still need to define the functional form of $a(X, N)$ and $b(X, N)$ to compute Equation 5.5.

To compute $a(X, N)$ we examined a number of relations between $\hat{a}$ and X, N. A regression of the form

$$\hat{a} = \alpha_1 + \alpha_2 \ln(X) + \alpha_3 \ln(N) \tag{5.6}$$

yields good results. For both datasets, the regression model is valid and statistically significant. In the case of Datasets 1 and 2 it explains most of the variability ($R^2 \approx 0.87$ and $R^2 \approx 0.96$, respectively). As expected, the fit to Dataset 2 is better than to Dataset

1, as we filter larger number of "anomalous" data points. Details of validity and the values of coefficients are given in Table 5.4.

We also defined a more complex (yet statistically significant relation between $\hat{a}$ and X, N, namely

$$\hat{a} = \alpha_1 + \alpha_2 \ln(X) + \alpha_3 \ln(N) + \alpha_4 \ln(X)^2 + \alpha_5 \ln(N)^2. \tag{5.7}$$

The model is valid and statistically significant as shown in Table 5.4. In the case of Datasets 1 and 2 the model explains more variability than Equation 5.6 ($R^2 \approx 0.91$ and $R^2 \approx 0.98$, respectively). However, the improvement is not dramatic.

Similarly, we explored relations between $\hat{b}$ and X, N to compute $b(X, N)$. A regression of the form

$$\hat{b} = \alpha_1 + \alpha_2 X + \alpha_3 N \tag{5.8}$$

yields adequate results. As in the above case (Eq. 5.6), the model is valid and statistically significant, but it cannot explain as much variability: $R^2 \approx 0.54$ for Datasets 1 and $R^2 \approx 0.78$ Dataset 2. Details of validity and the values of coefficients are given in Table 5.5.

A more complex relation between $\hat{b}$ and X, N is given by

$$\hat{b} = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 N^2 + \alpha_5 \ln(N) + \alpha_6 \ln(X). \tag{5.9}$$

Table 5.5 shows that the model is valid and statistically significant[2]. In the case of the Datasets 1 and 2, the model explains more variability than Equation 5.8 ($R^2 \approx 0.60$ and $R^2 \approx 0.83$, respectively). However, it still cannot explain as much variability as the

[2]In the case of Dataset 1, p-value of the N^2 term is > 0.05. We keep it for consistency – addition of the term does not degrade the R^2 value. To make sure that inclusion of the N^2 term does not bias the final result, we also executed the final validation (discussed in Section 5.4) while using the formula with and without the N^2 term. For Dataset 1, we obtained the same result for both formulas; for Dataset 2, the result was marginally better for the formula without the N^2 term. However, the models calibrated on Dataset 1 prevailed over the models calibrated on Dataset 2. Thus, from practical perspective, it is acceptable to keep the N^2 term (assuming that one will favor better performing models).

models for the intercept term (defined in Equations 5.6 and 5.7).

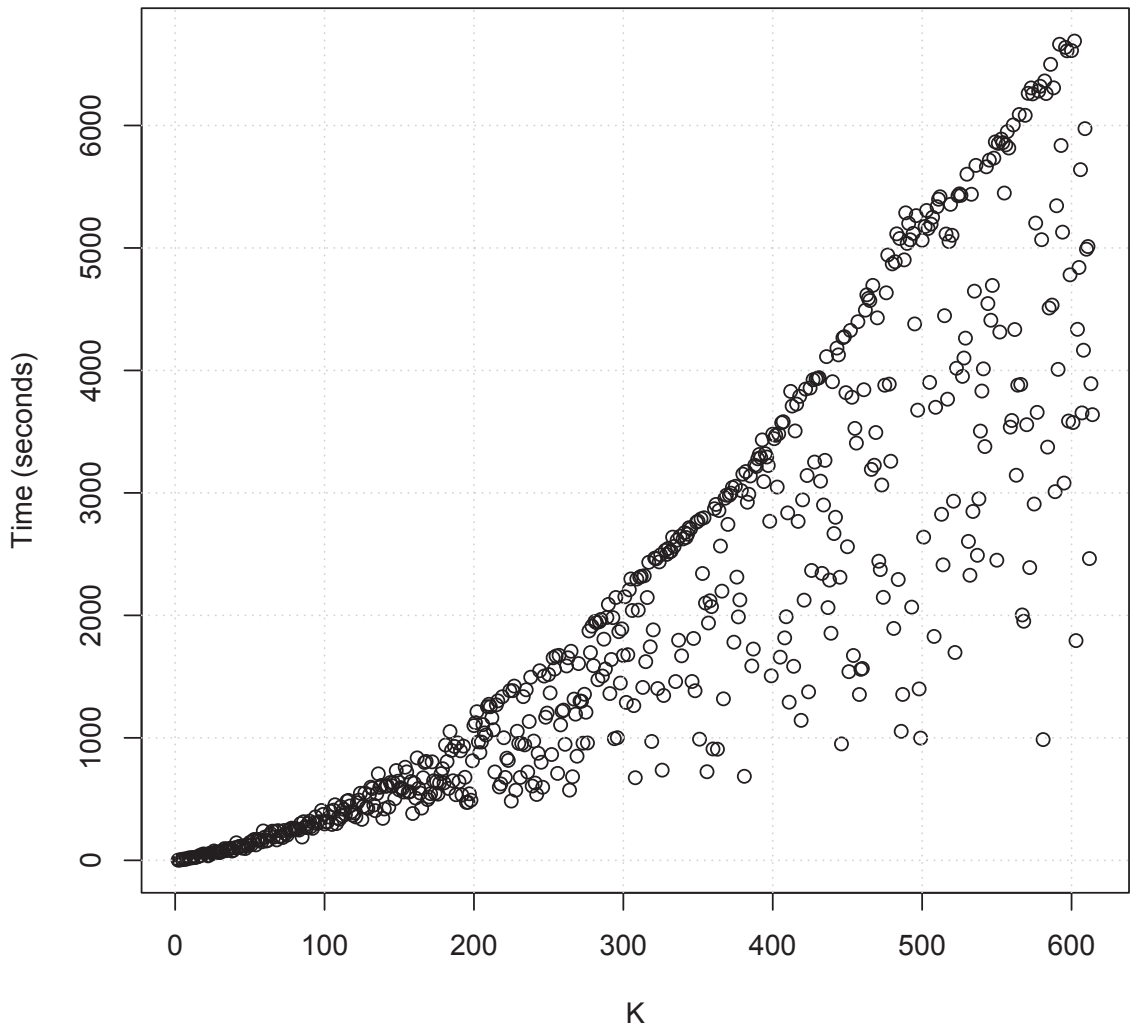


Figure 5.6: Time needed to compute (calibrate) LDA model for a given number of topics K (the input data are Android subset, January 2014).

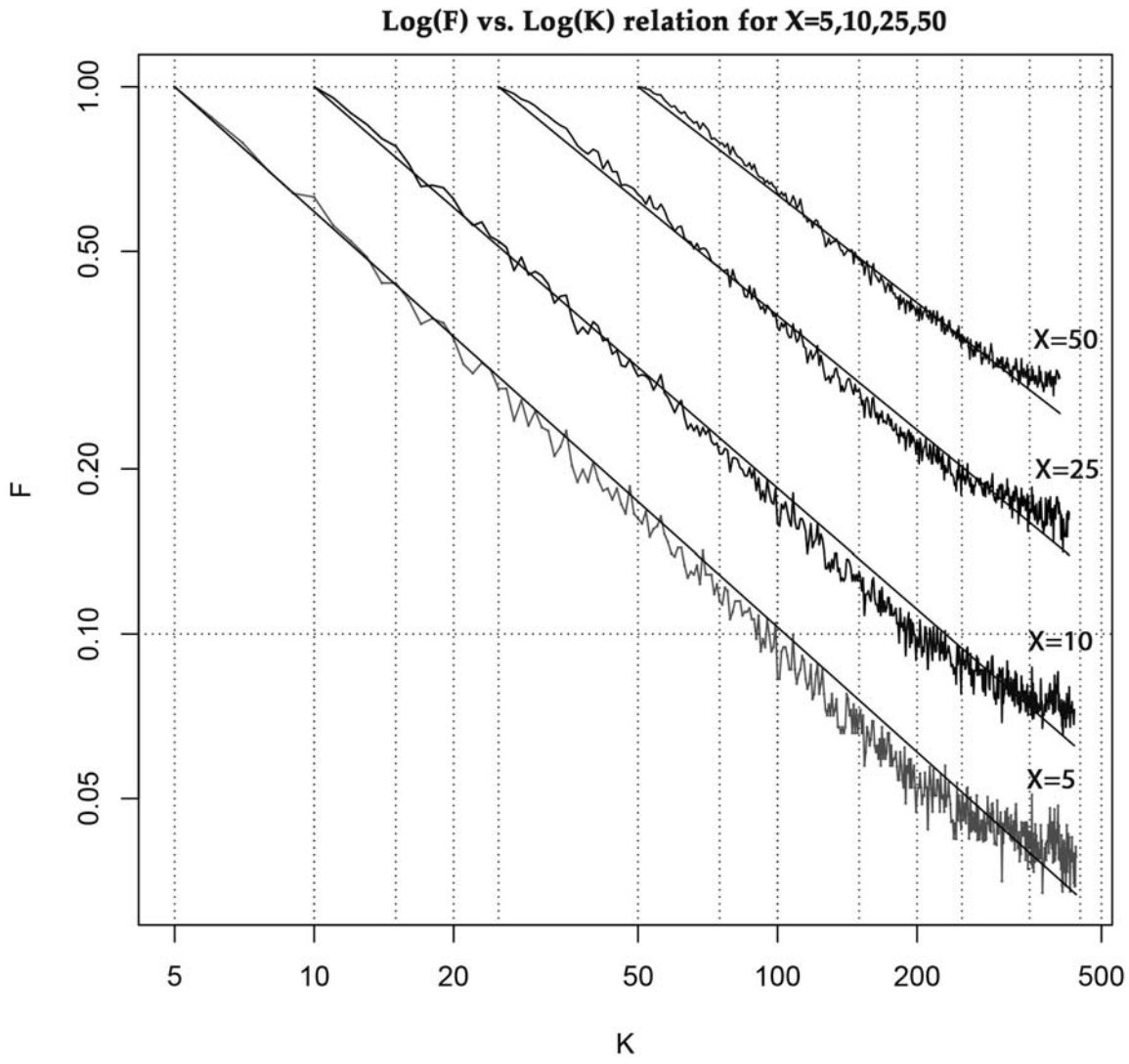


Figure 5.7: Fitted empirical data for $X = 5, 10, 25, 50$

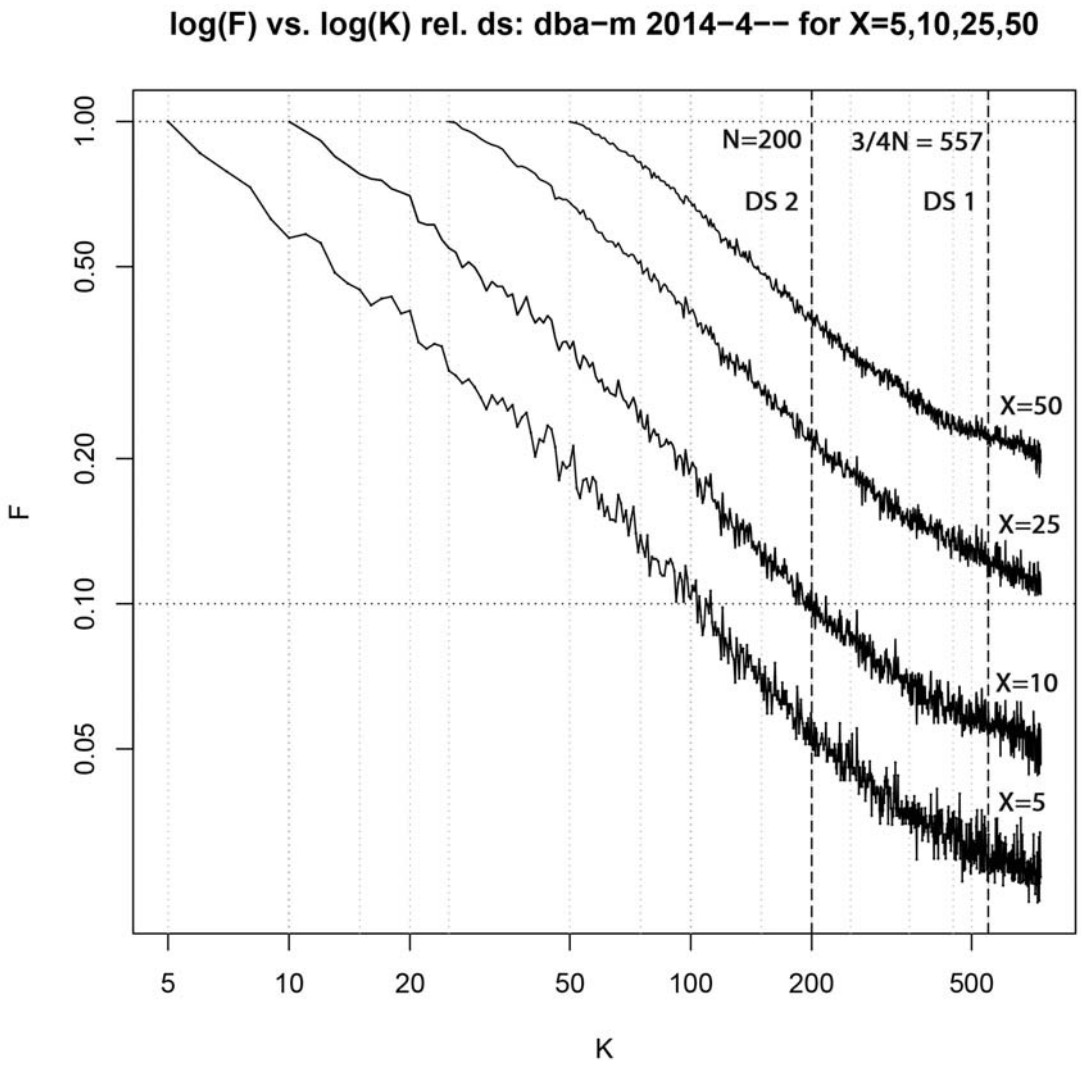


Figure 5.8: Linear relation between $log(F)$ and $log(K)$ breaks approximately at $K > 200$, example for X=5,10,25, and 50

Table 5.4: Linear regression models for intercept term $\hat{a}$ (flexible model: Equations 5.6 and 5.7). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 5.6	Eq. 5.7	Eq. 5.6	Eq. 5.7
$\ln(X)$	0.613^{***} (0.607, 0.618)	1.057^{***} (1.029, 1.085)	0.649^{***} (0.646, 0.652)	1.081^{***} (1.067, 1.095)
$\ln(N)$	0.053^{***} (0.043, 0.063)	-4.386^{***} (-4.628, -4.143)	-0.116^{***} (-0.122, -0.110)	2.694^{***} (2.569, 2.818)
$\ln(X)^2$		-0.086^{***} (-0.091, -0.080)		-0.083^{***} (-0.086, -0.081)
$\ln(N)^2$		0.319^{***} (0.302, 0.337)		-0.202^{***} (-0.211, -0.193)
Constant	-0.144^{***} (-0.215, -0.073)	14.716^{***} (13.875, 15.557)	1.171^{***} (1.128, 1.214)	-9.047^{***} (-9.480, -8.615)
Observations	4,704	4,704	4,704	4,704
R^2	0.874	0.906	0.956	0.976
Adjusted R^2	0.874	0.906	0.956	0.976
Residual Std. Error	0.182 (df = 4701)	0.157 (df = 4699)	0.109 (df = 4701)	0.081 (df = 4699)
F Statistic	$16{,}277.930^{***}$ (df = 2; 4701)	$11{,}341.730^{***}$ (df = 4; 4699)	$51{,}006.570^{***}$ (df = 2; 4701)	$47{,}647.760^{***}$ (df = 4; 4699)
Note:				*p<0.1; **p<0.05; ***p<0.01

Table 5.5: Linear regression models for slope term $\hat{b}$ (flexible model:Equations 5.8 and 5.9). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 5.8	Eq. 5.9	Eq. 5.8	Eq. 5.9
X	0.002^{***} (0.002, 0.002)	0.002^{***} (0.002, 0.002)	0.002^{***} (0.002, 0.002)	0.002^{***} (0.002, 0.002)
N	-0.00001^{***} (-0.00002, -0.00001)	-0.0001^{***} (-0.0002, -0.0001)	0.00003^{***} (0.00003, 0.00003)	0.0003^{***} (0.0003, 0.0004)
N^2		5.21e−09 (−2.44e−09, 1.29e−08)		$-4.83e{-}08^{***}$ (−5.25e−08, −4.41e−08)
$\ln(X)$		0.014^{***} (0.011, 0.016)		0.007^{***} (0.006, 0.009)
$\ln(N)$		0.116^{***} (0.095, 0.137)		-0.202^{***} (-0.213, -0.190)
Constant	-0.744^{***} (-0.746, -0.741)	-1.450^{***} (-1.567, -1.334)	-0.841^{***} (-0.842, -0.840)	0.276^{***} (0.212, 0.340)
Observations	4,704	4,704	4,704	4,704
R^2	0.538	0.599	0.782	0.825
Adjusted R^2	0.537	0.598	0.782	0.824
Residual Std. Error	0.032 (df = 4701)	0.030 (df = 4698)	0.018 (df = 4701)	0.016 (df = 4698)
F Statistic	$2{,}732.760^{***}$ (df = 2; 4701)	$1{,}401.956^{***}$ (df = 5; 4698)	$8{,}443.119^{***}$ (df = 2; 4701)	$4{,}419.474^{***}$ (df = 5; 4698)
Note:				*p<0.1; **p<0.05; ***p<0.01

Constrained Model

In the constrained case, Equation 5.2 becomes

$$F = a(X, N)K^{b(X,N)} = X^{-b(X,N)}K^{b(X,N)}. \tag{5.10}$$

Solving it for K yields:

$$K = \left[\frac{F}{X^{b(X,N)}}\right]^{1/b(X,N)}. \tag{5.11}$$

To find a relation between $\hat{b}$ and X, N we explore two regression models. A simple regression model is given by:

$$b(X, N) = \alpha_1 + \alpha_2 X + \alpha_3 N, \tag{5.12}$$

and a more complex one by:

$$b(X, N) = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 \ln(X). \tag{5.13}$$

Both models are valid and statistically significant. Model's details are given in Table 5.6. Simple model (Eq. 5.12) explains 79% of variability ($R^2 \approx 0.79$), while a more complex model, given by Eq. 5.13, explains 83% of variability ($R^2 \approx 0.83$).

Table 5.6: Linear regression models for $\hat{b}$ (constrained model: Equations 5.12 and 5.13). The values in brackets represent 90% confidence interval of the coefficients.

	Dataset 1		Dataset 2	
Coefficient at	Eq. 5.12	Eq. 5.13	Eq. 5.12	Eq. 5.13
X	0.002*** (0.002, 0.002)	0.001*** (0.001, 0.001)	0.003*** (0.003, 0.003)	0.001*** (0.001, 0.001)
N	0.00002*** (0.00001, 0.00002)	0.00002*** (0.00001, 0.00002)	0.00003*** (0.00003, 0.00003)	0.00003*** (0.00003, 0.00003)
$\ln(X)$		0.031*** (0.030, 0.032)		0.035*** (0.033, 0.037)
Constant	−0.767*** (−0.768, −0.766)	−0.819*** (−0.821, −0.817)	−0.777*** (−0.778, −0.775)	−0.836*** (−0.839, −0.833)
Observations	4,704	4,704	4,704	4,704
R^2	0.817	0.873	0.787	0.834
Adjusted R^2	0.817	0.873	0.787	0.834
Residual Std. Error	0.016 (df = 4701)	0.013 (df = 4700)	0.021 (df = 4701)	0.019 (df = 4700)
F Statistic	10,510.100*** (df = 2; 4701)	10,802.240*** (df = 3; 4700)	8,695.582*** (df = 2; 4701)	7,864.944*** (df = 3; 4700)
Note:				*p<0.1; **p<0.05; ***p<0.01

5.4 Analysis of results

In order to estimate goodness of fit of our approximation of a and b, we compute the Root-Mean-Square Error (RMSE) (defined by Equation 5.14) between the actual value of F and the ones given to us by Equation 5.4 for each dataset and for each value of X (where $X = 2, 3, \ldots, 50$). Summary statistics for various models is given in Tables 5.7 and 5.8 and Figures 5.9 and 5.10.

$$RMSE = \sqrt{\frac{\sum (y - \hat{y})^2}{n}}, \tag{5.14}$$

where y is an actual value of F, $\hat{y}$ is the predicted value of F, and n is the number of values to predict.

By looking at the Tables 5.7 and 5.8 we can appreciate that the performance of the model varies based on various factors, discussed in the following paragraphs.

Table 5.7: Summary statistics for the RMSE: Dataset 1

Statistic	n	Mean	St. Dev.	Min	Max
Flex-simple	4,704	0.028	0.013	0.006	0.072
Flex-complex	4,704	0.019	0.005	0.006	0.042
Constrained-simple	4,704	0.018	0.006	0.006	0.040
Constrained-complex	4,704	0.018	0.006	0.004	0.039
Individual fit-flex	4,704	0.018	0.005	0.005	0.032
Individual fit-constrained	4,704	0.017	0.006	0.004	0.038

Table 5.8: Summary statistics for the RMSE: Dataset 2

Statistic	n	Mean	St. Dev.	Min	Max
Flex-simple	4,704	0.034	0.020	0.010	0.226
Flex-complex	4,704	0.022	0.007	0.009	0.071
Constrained-simple	4,704	0.023	0.006	0.009	0.046
Constrained-complex	4,704	0.023	0.006	0.007	0.044
Individual fit-flex	4,704	0.018	0.005	0.006	0.038
Individual fit-constrained	4,704	0.021	0.006	0.005	0.041

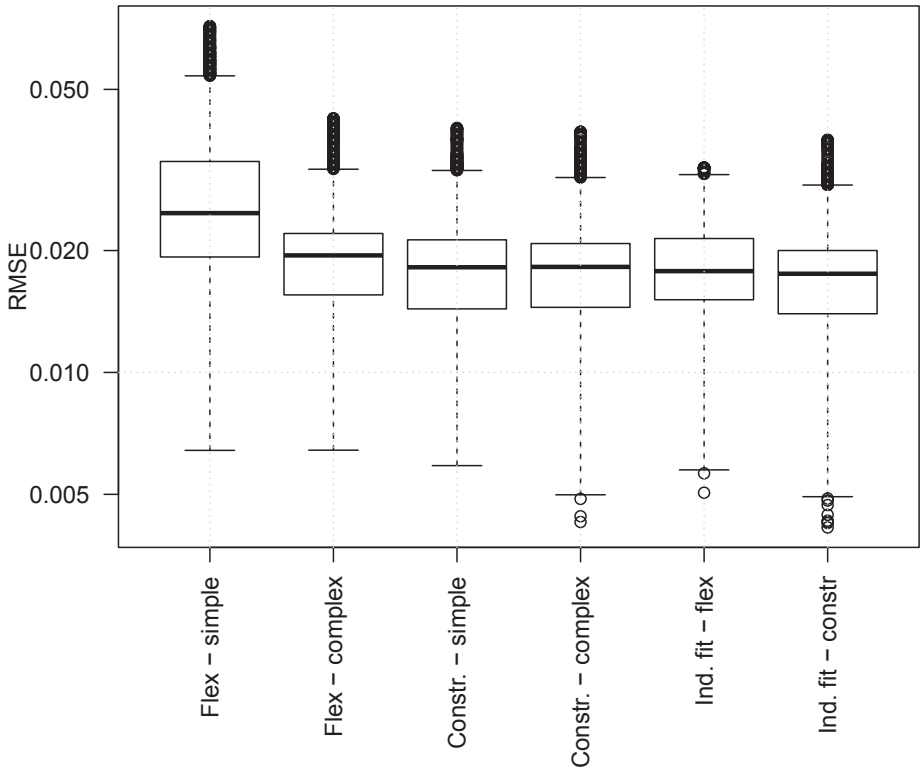


Figure 5.9: Summary statistics for the RMSE: Dataset 1. 'Flex - simple' uses Equations 5.6 and 5.8; 'Flex - complex' uses Equations 5.7 and 5.9. 'Constr - simple' uses Equation 5.12; 'Constr - complex' uses Equation 5.13. 'Ind. fit - flex' shows RMSE for flexible linear model (Eq. 5.4 fitted individually to every data subset), while 'Ind. fit - constr' depicts RMSE for constraint linear model (Eq. 5.10 fitted individually to every data subset).

5.4.1 Selection of the best performing model

In order to select the best performance model, we focus on the RMSE stats (provided in Tables 5.7 and 5.8) and select the model that minimizes the mean RMSE.

For Dataset 1 (DS1), among four models, constrained-complex model yields the lowest mean, min, and max RMSE values, comparable with the performance of individually-

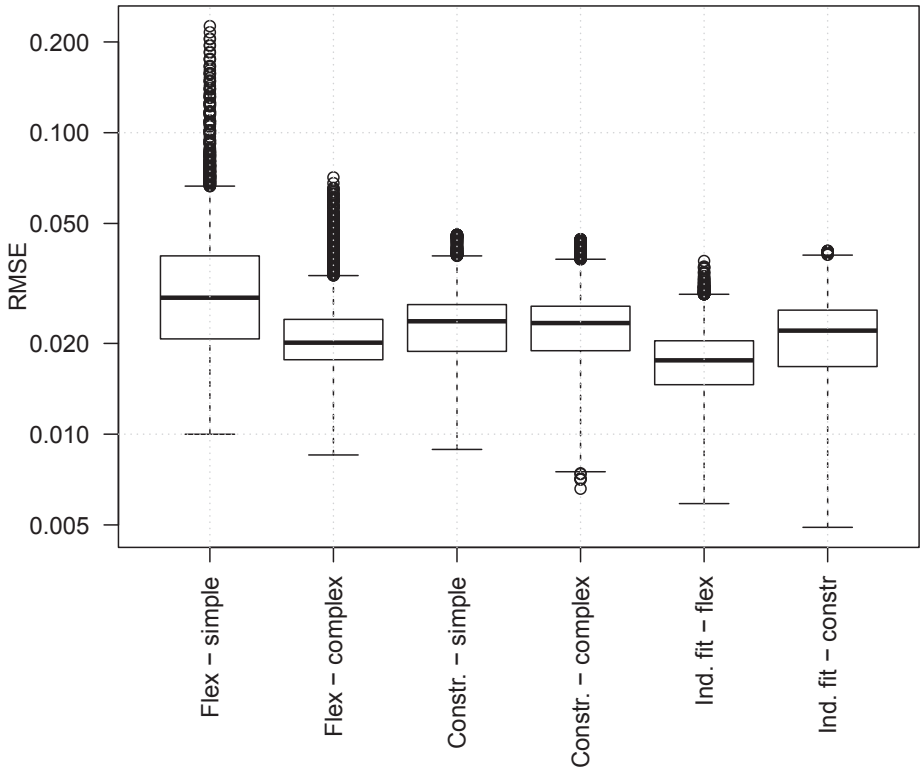


Figure 5.10: Summary statistics for the RMSE: Dataset 2. Flex - simple' uses Equations 5.6 and 5.8; 'Flex - complex' uses Equations 5.7 and 5.9. 'Constr - simple' uses Equation 5.12; 'Constr - complex' uses Equation 5.13. 'Ind. fit - flex' shows RMSE for flexible linear model (Eq. 5.4 fitted individually to every data subset), while 'Ind. fit - constr' depicts RMSE for constraint linear model (Eq. 5.10 fitted individually to every data subset).

fitted lines. Thus, constrained-complex model is the winner for DS1.

In the case of Dataset 2 (DS2), RMSE of the flex-complex model is slightly lower than that of the constrained-complex model (0.022 vs. 0.032). However, the min and max values of the flex-complex are worse than that of the constrained-complex model (0.009 vs. 0.007 and 0.071 vs. 0.044, respectively). Thus, the constrained-complex can

be considered a winner for DS2 too.

5.4.2 Analysis of the best performing model

To verify robustness of the results of the the best performing (constrained-complex) model and make sure that we are not overfitting, we performed 10-fold cross validation[3]. We split 96 data subsets into ten partitions. We then fit Equation 5.13 to nine out of ten partitions. We then fit the constrained model (Equation 5.10) to the raw data of the tenth partition and compute the RMSE value. The process is repeated nine more times, alternating partitions of the test and train sets. The resulting ten values of the RMSE are shown in Figure 5.11. As we can see, for both Dataset 1 (mean ≈ 0.018) and Dataset 2 (mean ≈ 0.024) the RMSE values are comparable to those reported in Tables 5.7 and 5.8. Thus, we are not overfitting.

In Figures 5.12 and 5.13 we can observe that for small values of X the RMSE is smaller and it increases as X gets larger. The X grows fast until approximately $X = 20$ and then it starts to slow down in growth.

In the other hand, in Figures 5.14 and 5.15 we can see that the relation between N and the RMSE varies in a similar manner to X for small values of N and large values of N. Therefore, we can say that the quality of RMSE is not influenced significantly by N, but it is influenced by X in the sense that the smaller the value of X the better the predictions are.

In the case of the datasets analyzed, for the Dataset 1 the fitting from higher to lower is android, salesforce, dba monthly, and dba quarterly (see Figure 5.16) and for the Dataset 2 is android, dba quarterly, salesforce monthly and dba monthly (see Figure 5.17).

Finally, Figure 5.18 shows one plot that demonstrates the fitting achieved with the

[3]The listing for performing the validation is given in Appendix B.10

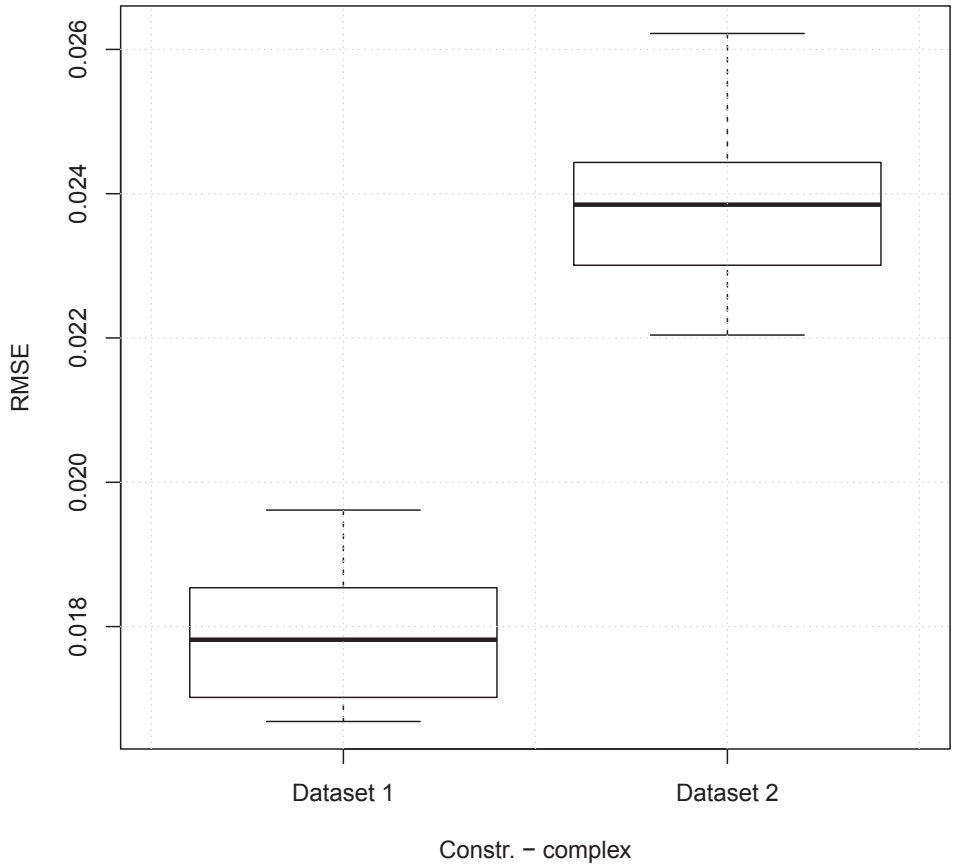


Figure 5.11: 10-fold cross validation of the fit of the 'Constrained - complex' model (Equation 5.13).

constrained-complex model for Dataset 1. By eyeballing this plot we can appreciate that starting at the left corner the fit is very good, then it goes above the data and when it gets the right side, it starts to deteriorate. The same behaviour applies to all X's shown ($X = 5, 10, 25$, and 50, these values of X were selected at random). For a larger selection of plots that show this fitting please see Appendix A.

In the case of Dataset 1, the model yields better results (i.e. lower RMSE) for data splits per-month in comparison with the data splits per-quarter, as can be seen in Fig-

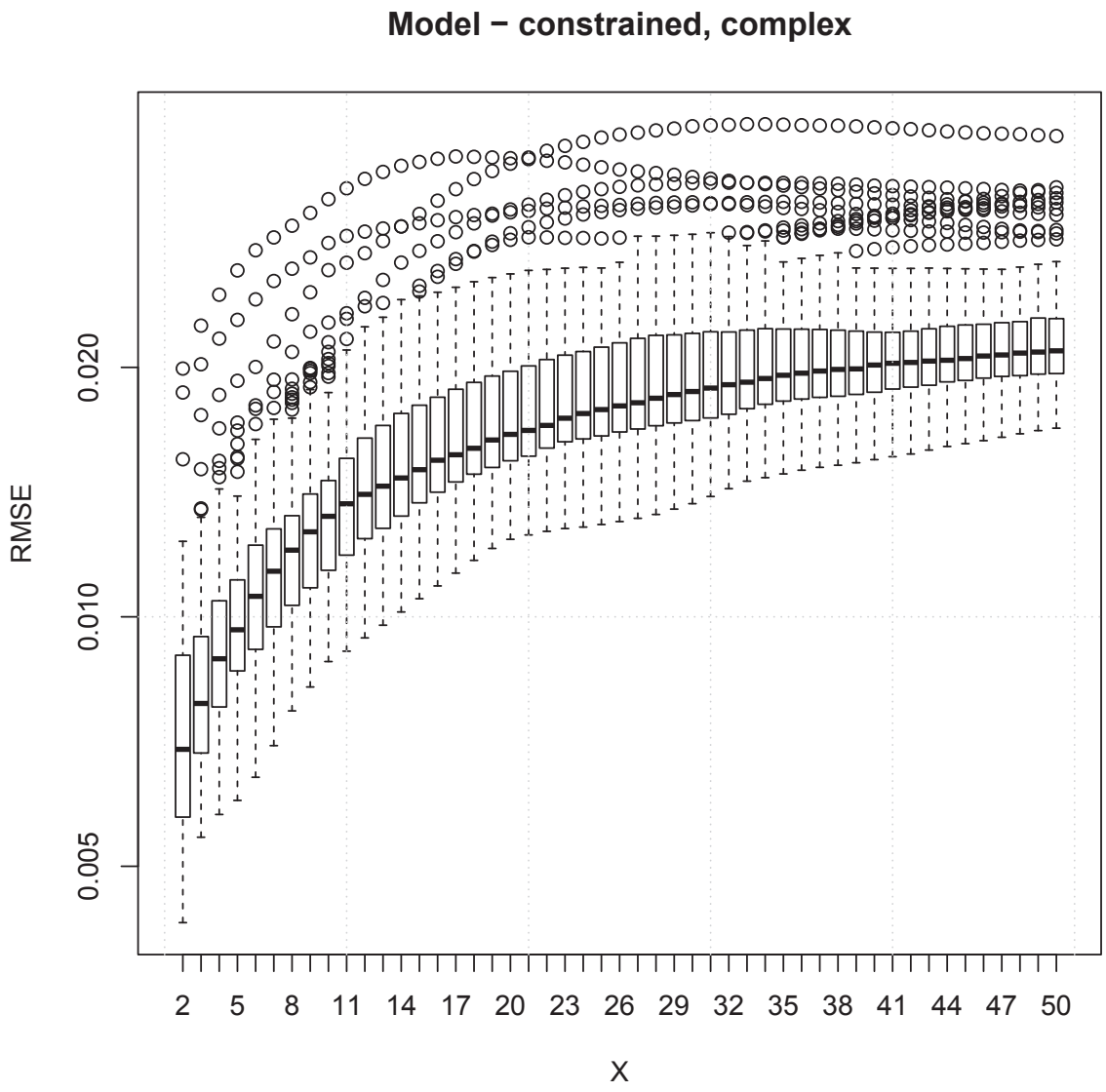


Figure 5.12: Performance of the constrained complex model for different values of X: Dataset 1.

ure 5.16. We conjecture that the larger number of documents in the quarterly subsets plays a role here: Figure 5.14 shows that RMSE increases slightly for large values of N.

However, in the case of Dataset 2 the RMSE for per-quarter data splits is on par or better than in the case of per-month splits (as can be seen in Figure 5.17). We also

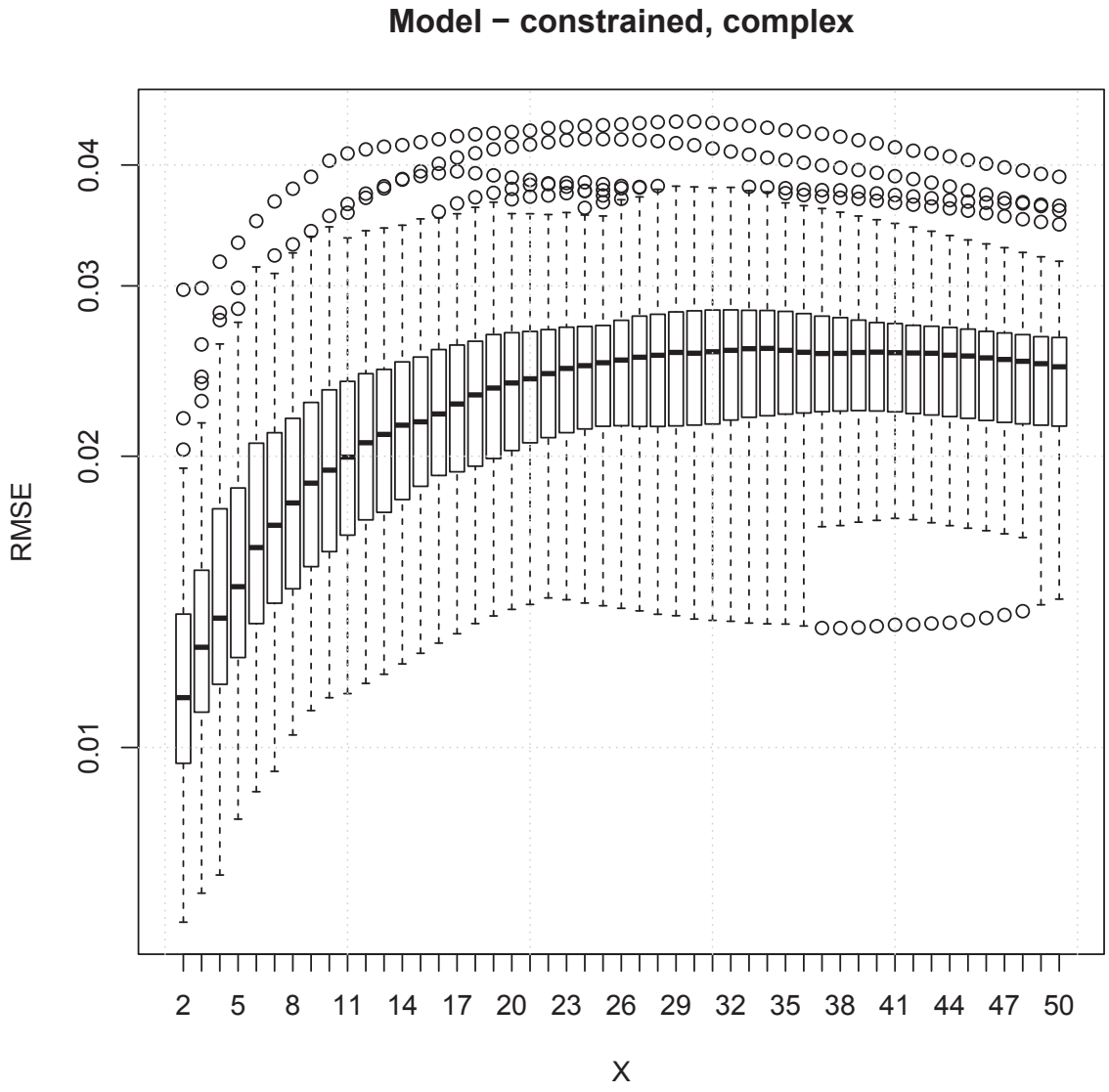


Figure 5.13: Performance of the constrained complex model model for different values of X: Dataset 2.

see that for small and large values of N the performance is comparable as shown in Figure 5.15. We conjecture that this can be explained by the fact that we truncate our K values at 200 and get rid of the error factor contributed by the larger values of K.

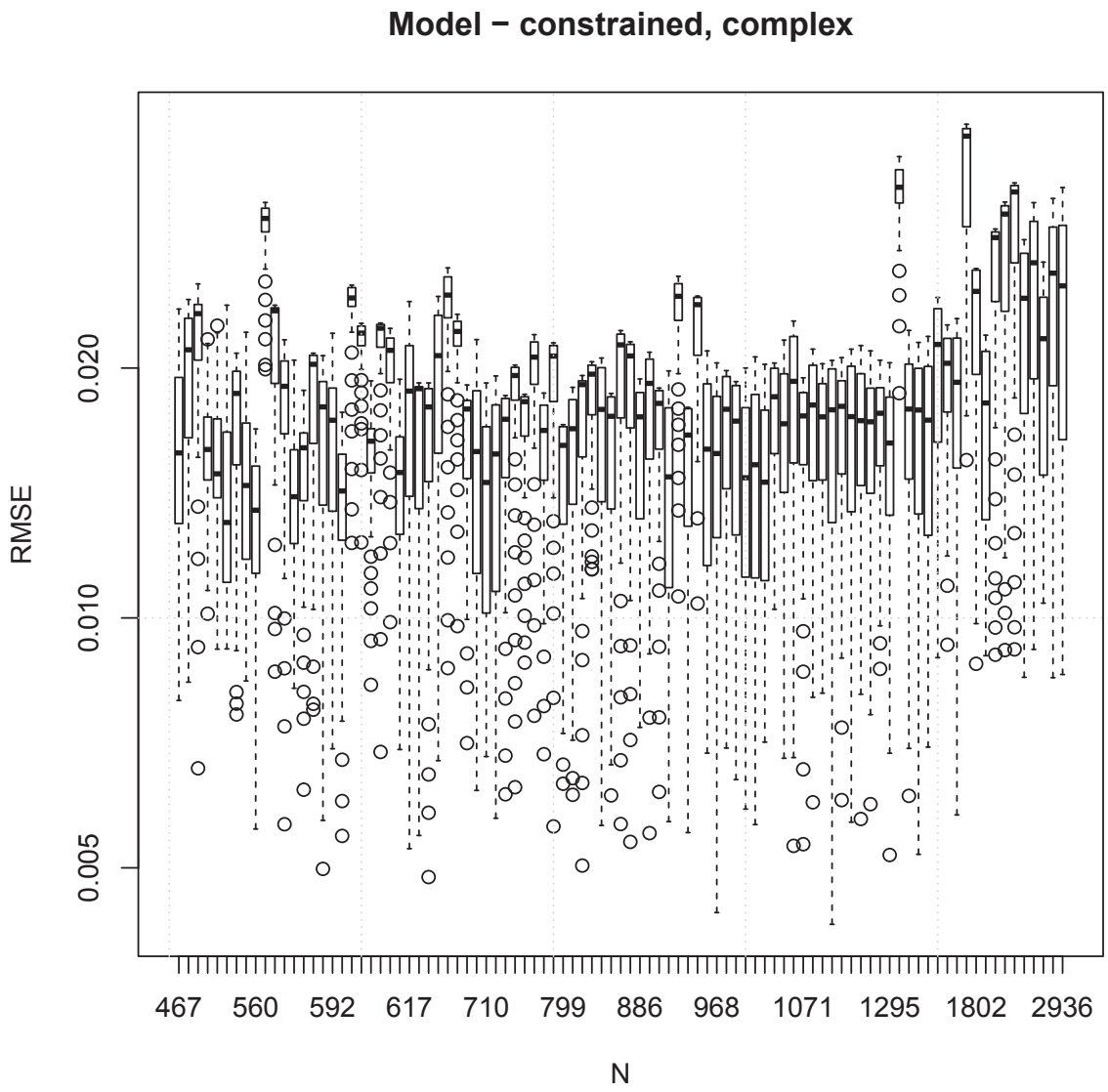


Figure 5.14: Performance of the constrained complex model for different values of N: Dataset 1.

The case when $K > 0.75N$

Based on our experiments, Equation 5.2 holds for $K \leq 0.75N$. As $K \to N$, the Power Law approximation deteriorates (see Figure 5.7 for an example). Likely, this is because

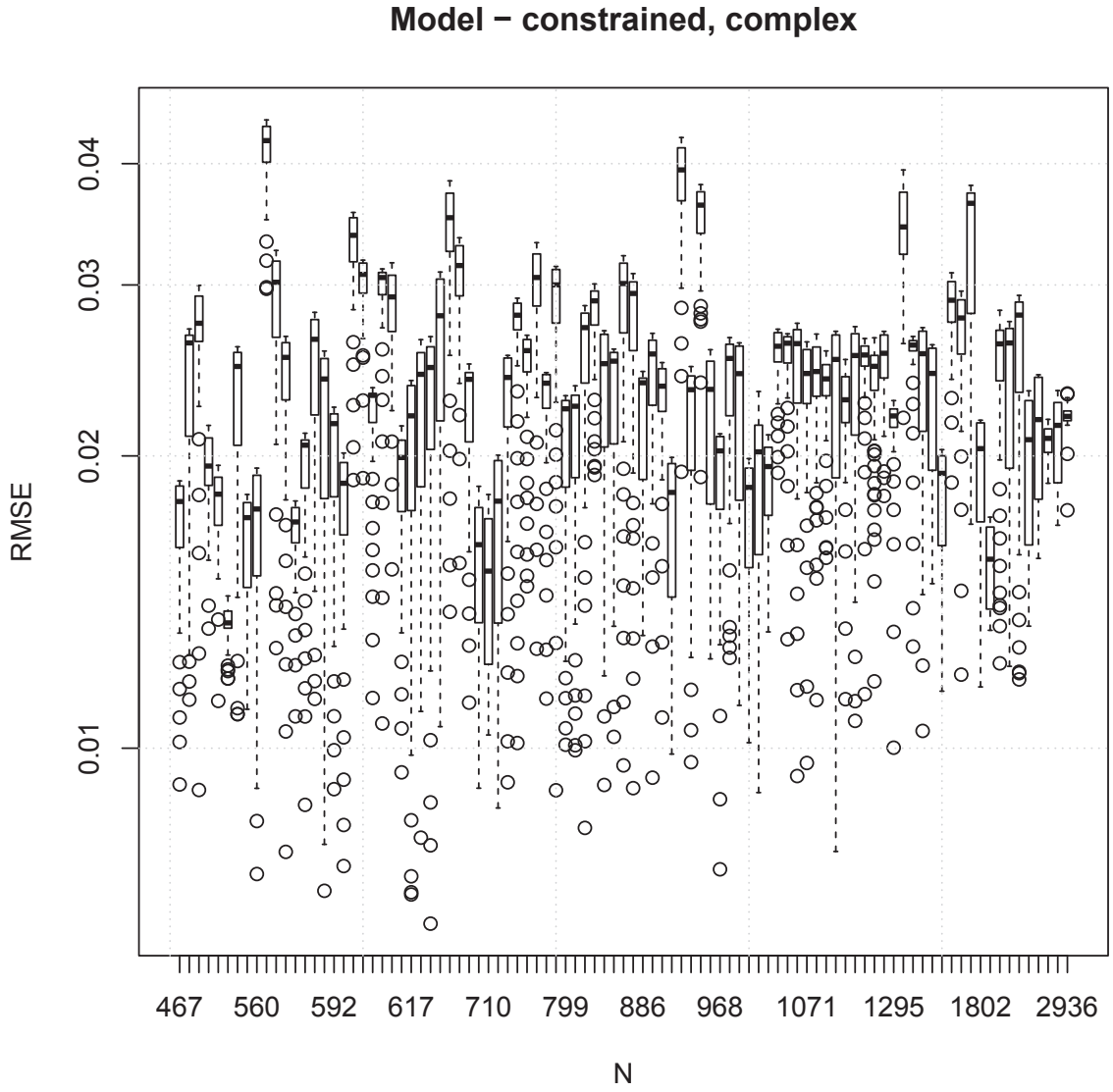


Figure 5.15: Performance of the constrained complex model model for different values of N: Dataset 2.

the average number of documents per topic becomes too small for LDA to process. It seems, empirically, that the average number of posts per topic should be 4 or greater, hence the $K \leq 0.75N$ constraint.

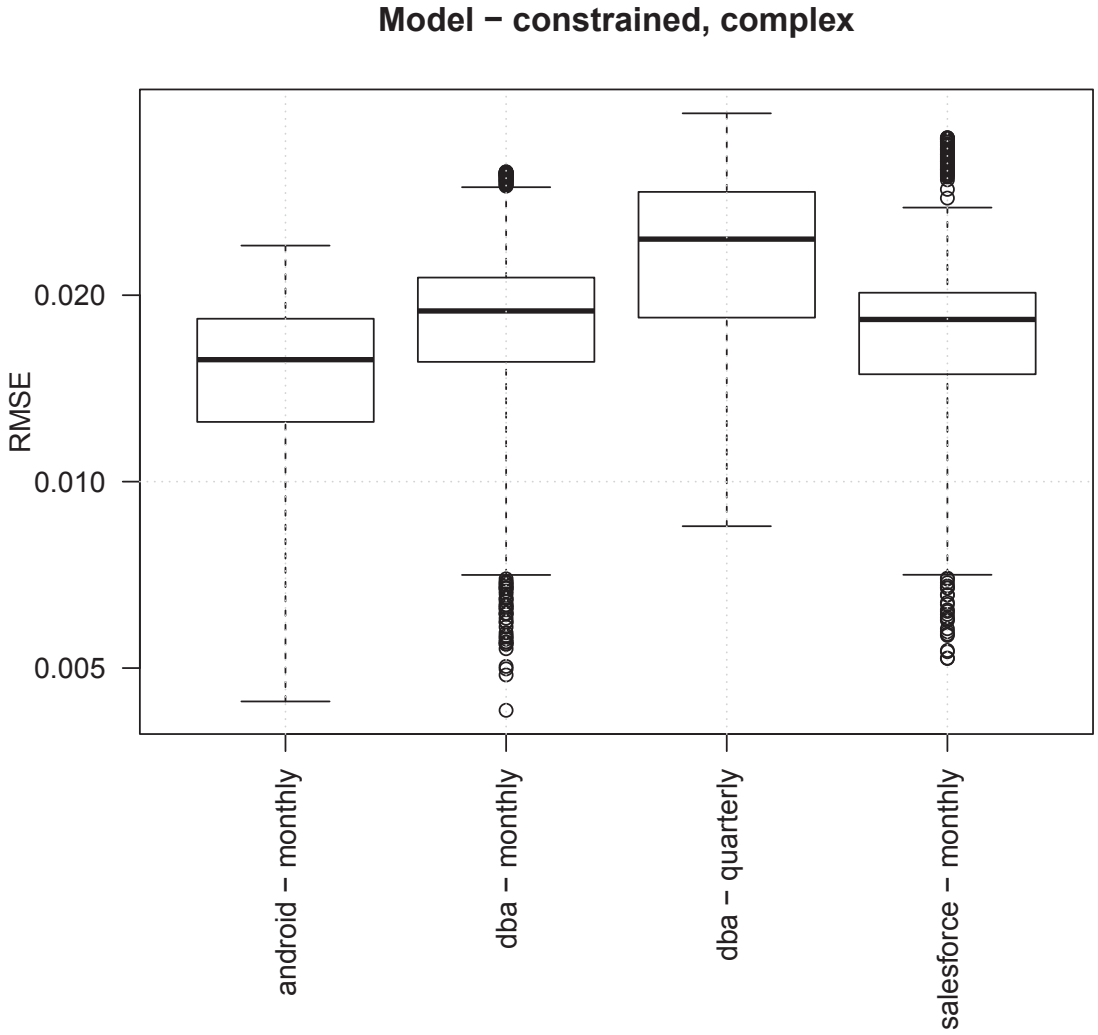


Figure 5.16: Performance of the constrained complex mode per dataset: Dataset 1.

Summary

In conclusion, we can appreciate graphically (in Figure 5.19) the "path" that leads to the best-performing model $F = X^{-b(X,N)}K^{b(X,N)}$ with the more complex prediction variable

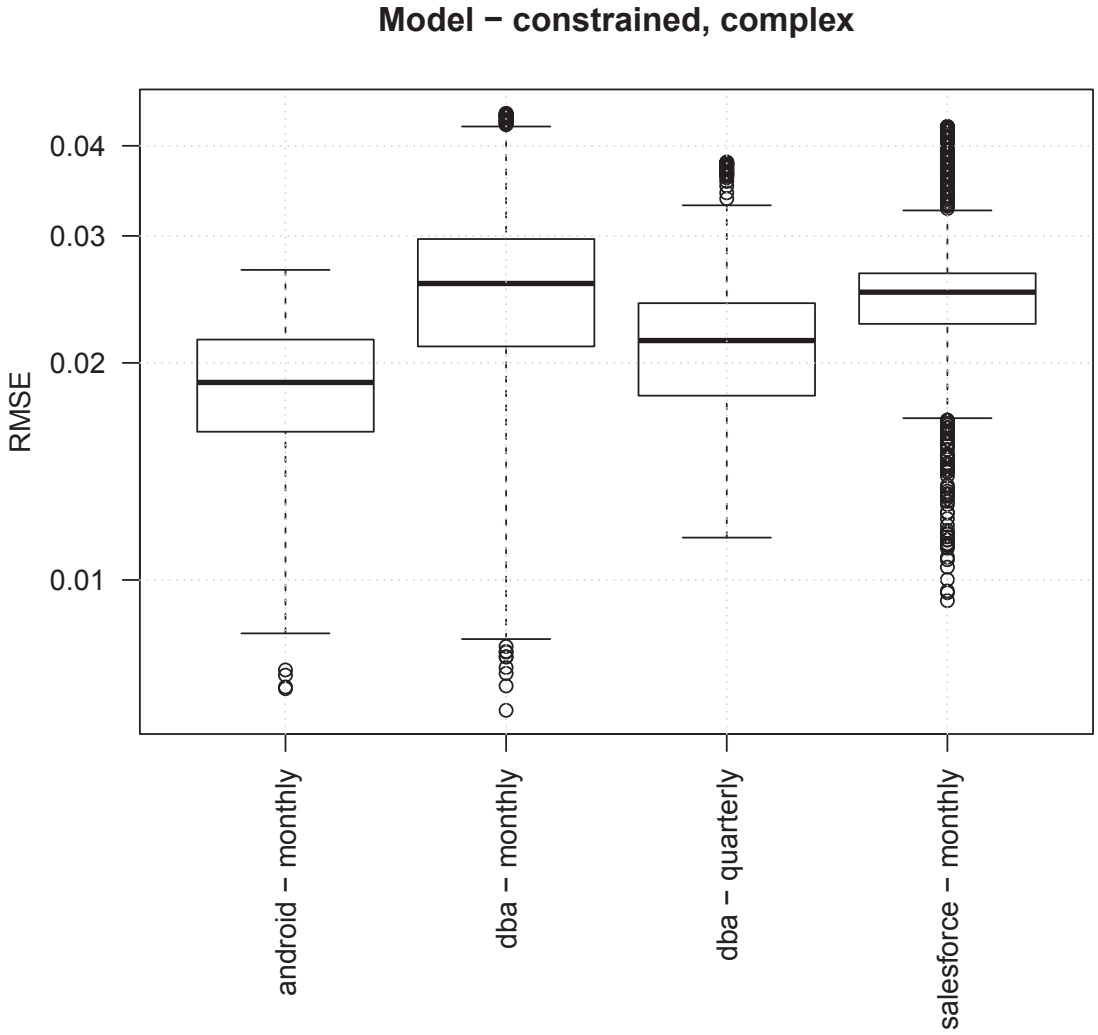


Figure 5.17: Performance of the constrained complex model per dataset: Dataset 2.

$b(X, N) = \alpha_1 + \alpha_2 X + \alpha_3 N + \alpha_4 \ln(X)$. The model performs well with the $K \leq 0.75N$ (Dataset 1) filter.

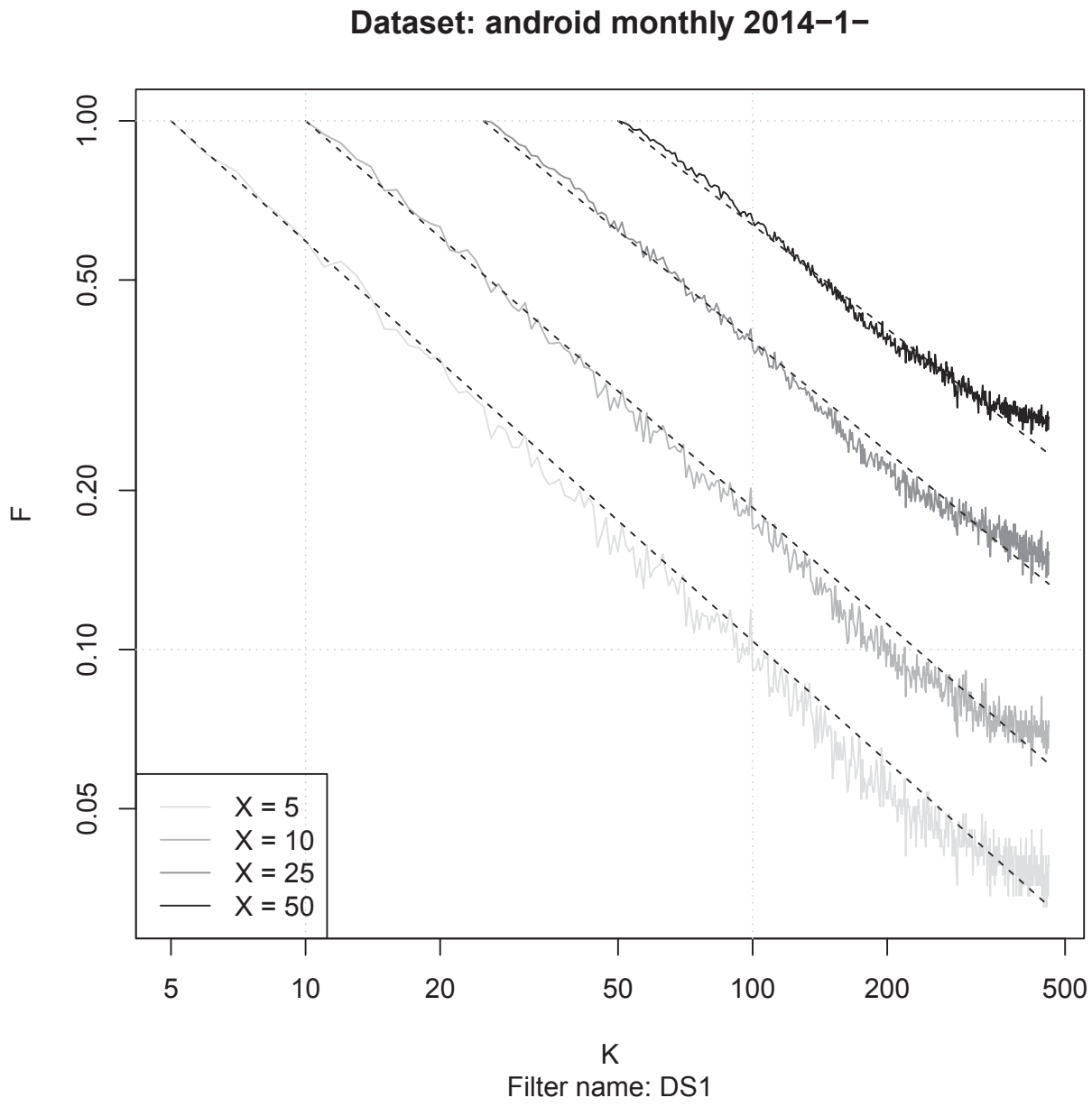


Figure 5.18: Fitting for a selection of top X using the constrained-complex model for Dataset 1

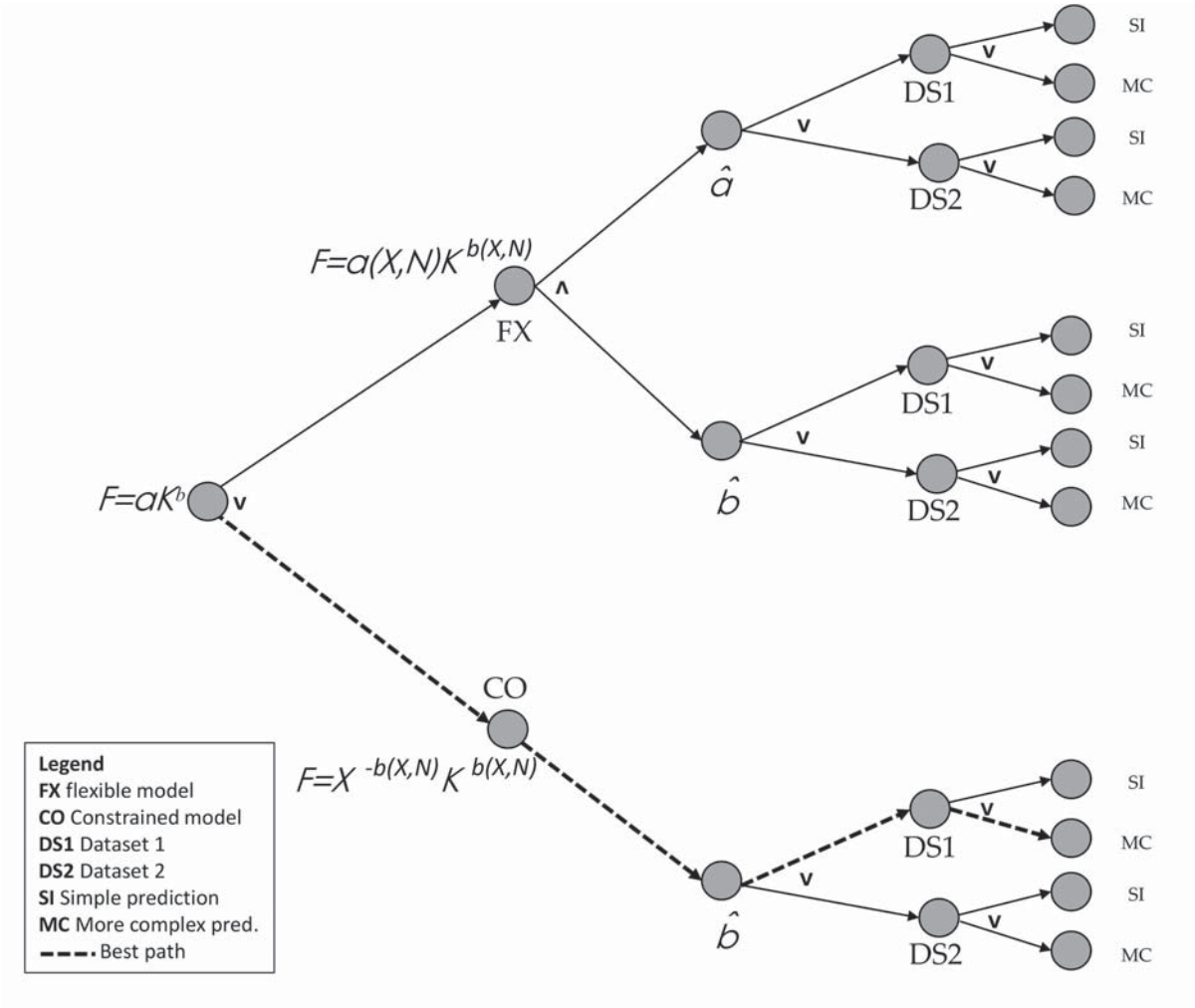


Figure 5.19: Decision tree for selecting the best fitting model. The path to the best-performing model is given by the dashed line.

5.5 Threats to Validity

In the following paragraphs we outline some of the limitations of our study and their impacts as per [26, 70].

Threats to conclusion validity are about the degree to which conclusions attained about relationships in our data are reasonable. To verify the absence of overfitting, we

performed 10-fold cross validation analysis of our best model.

Threats to construct validity involves the relationship among the concepts and theories that backs the experiment and what is measured and affected. In this regard, the experiments that we have carried out were designed based on formal knowledge. We have also used the data from three different sources, partitioning the data either monthly or quarterly.

Threats to internal validity [29] comprise potential errors in our execution of the study process, these errors may influence the accuracy of our results and the conclusions we deduce from them. In order to avoid introducing bias we used an automated procedure for data extraction and processing, that included the creation of scripts in Perl and R languages.

Threats to external validity [73] include the degree to which we can generalize our results. The subjects of our study contain three datasets for several periods comprising thousands of data points. The results cannot be generalized to other datasets, rather the design of this study is based on the concept of the critical case [91]. If the model does not work "out-of-the-box", one can re-calibrate it by following our methodology.

Chapter 6

Conclusion and Future Work

LDA is a widespread IR probabilistic topic model technique for performing analysis of text corpora. We have reviewed its roots, its basic functioning, its applications in the SE arena, exemplified its usage, and derived a formula from *SE-related text corpuses* to calibrate the model.

The problem that we have addressed in this thesis, was to find in an expedited manner the number of topics (K), which the LDA model requires to be implemented.

To answer the research question "How can we quickly select the number of topics K so that the top X topics include a certain fraction F of the N documents under study?", we created a simple, closed-form Power Law expression (5.11), estimating K with X, F, and N as input. Although Power Law occurs frequently in SE [55], to the best of our knowledge, this is the first appearance of the power law in LDA parameter calibration. Moreover, we established that LDA models and (5.11) become unstable if $K > 0.75N$.

Practitioners can accelerate LDA analysis by using (5.11) with (5.13) to suggest the number of topics required to answer a particular question (e.g., to identify customers' pain-points and to prioritize maintainers tasks). They can forego the often computationally-prohibitive current practice of iterating over all values of K to identify the optimal value. Formula (5.11) is also of interest to theoreticians as it suggests that dif-

ferent *SE-related text corpuses* (in our case a technical Q&A forum and an issue-tracking system) might have similar underlying properties.

The formula is validated on three datasets and, as discussed in Section 5.5, cannot be generalized to other datasets. However the results do hint the existence of underlying structural similarities in the datasets.

In the future, we plan to analyze additional *SE-related text corpuses* and extend our work to non-*SE-related text corpuses*. Also, we would like to find ways for improving the formula for K.

Appendices

Appendix A

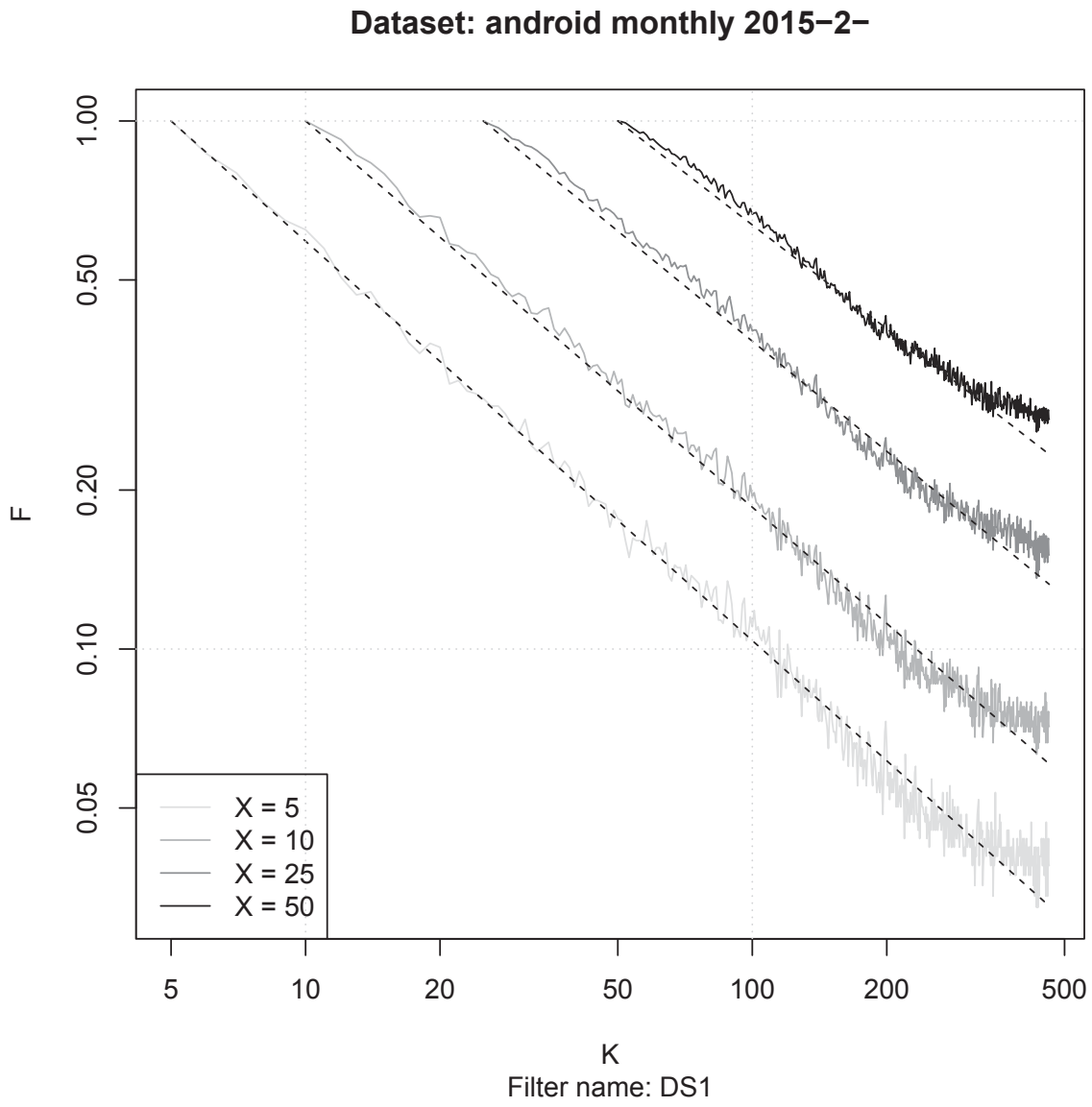
Sample Plots

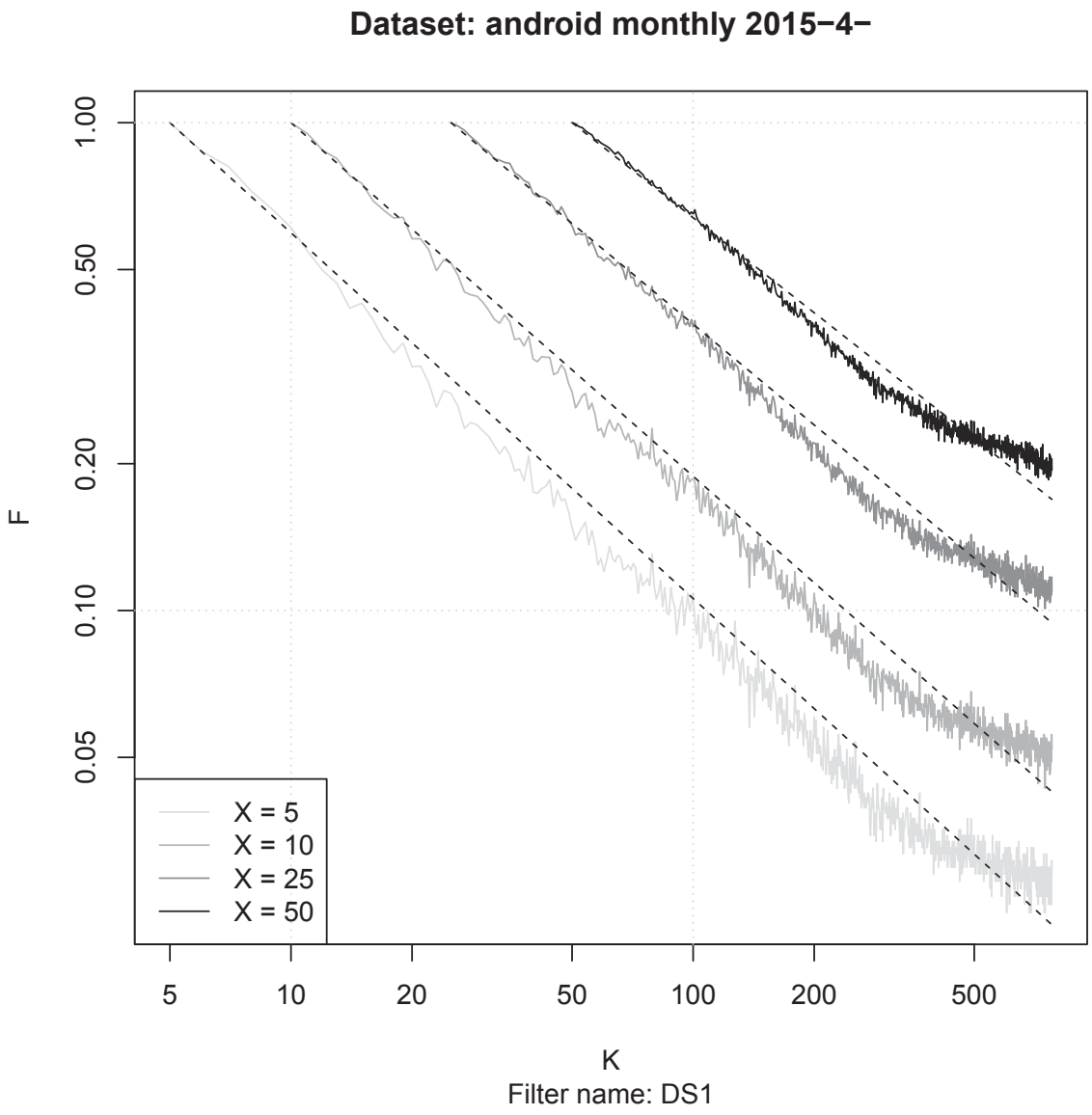
We are including in this section a selection of sample plots that show the fitting for the datasets considered (Table A.1).

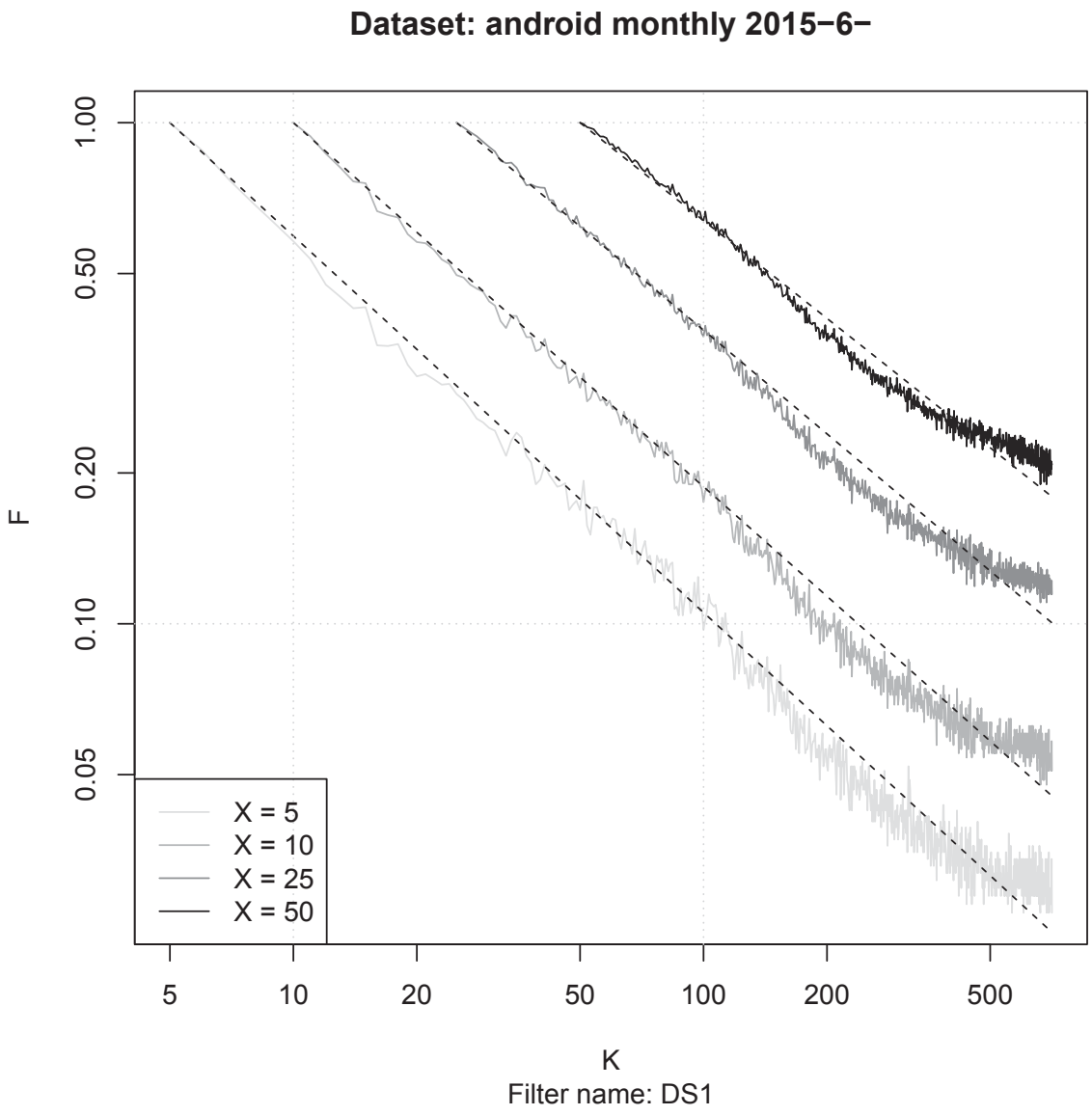
Table A.1: Sample plots included in this appendix

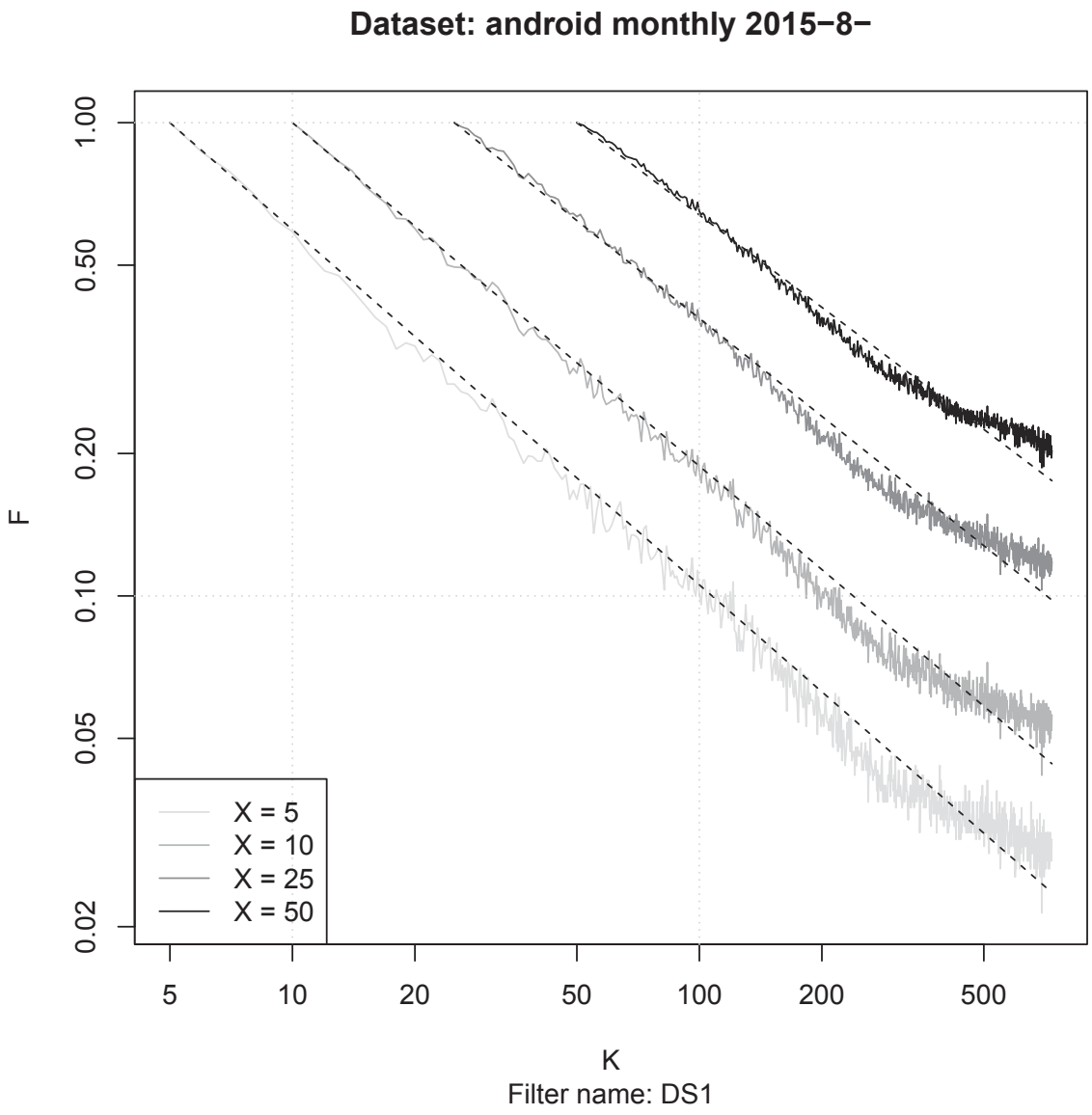
Dataset name	Period	Year	Month or Quarter number	Filter
android	monthly	2015	2,4,6,8,10,12	DS1
android	monthly	2015	2,12	DS2
dba	quarterly	2012	1,2,3,4	DS1
dba	quarterly	2012	2,4	DS2
dba	monthly	2014	2,4,6,8,10,12	DS1
dba	monthly	2014	2,12	DS2
salesforce	monthly	2015	2,4,6,8,10	DS1
salesforce	monthly	2015	2,12	DS2

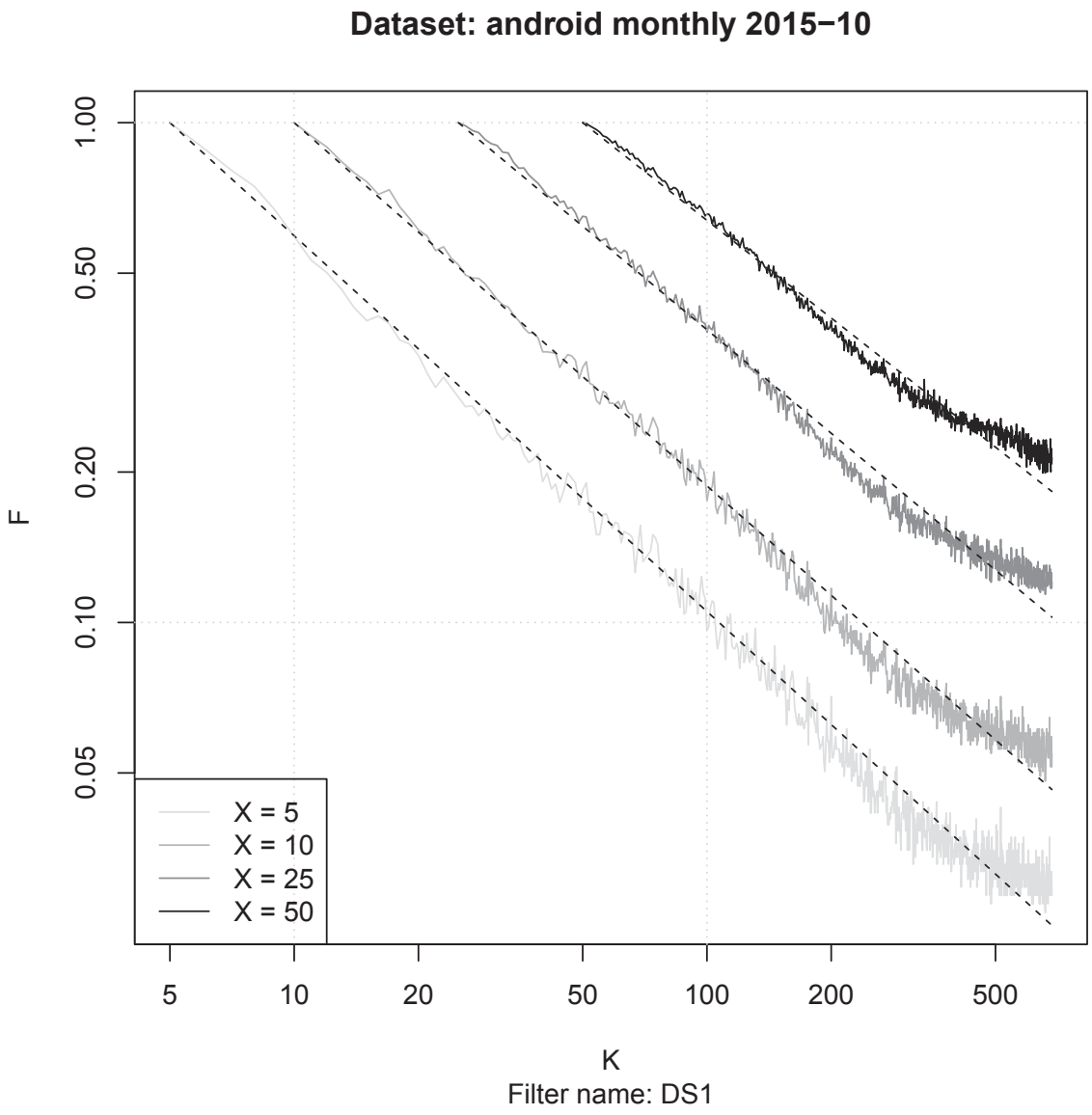
A.1 android dataset

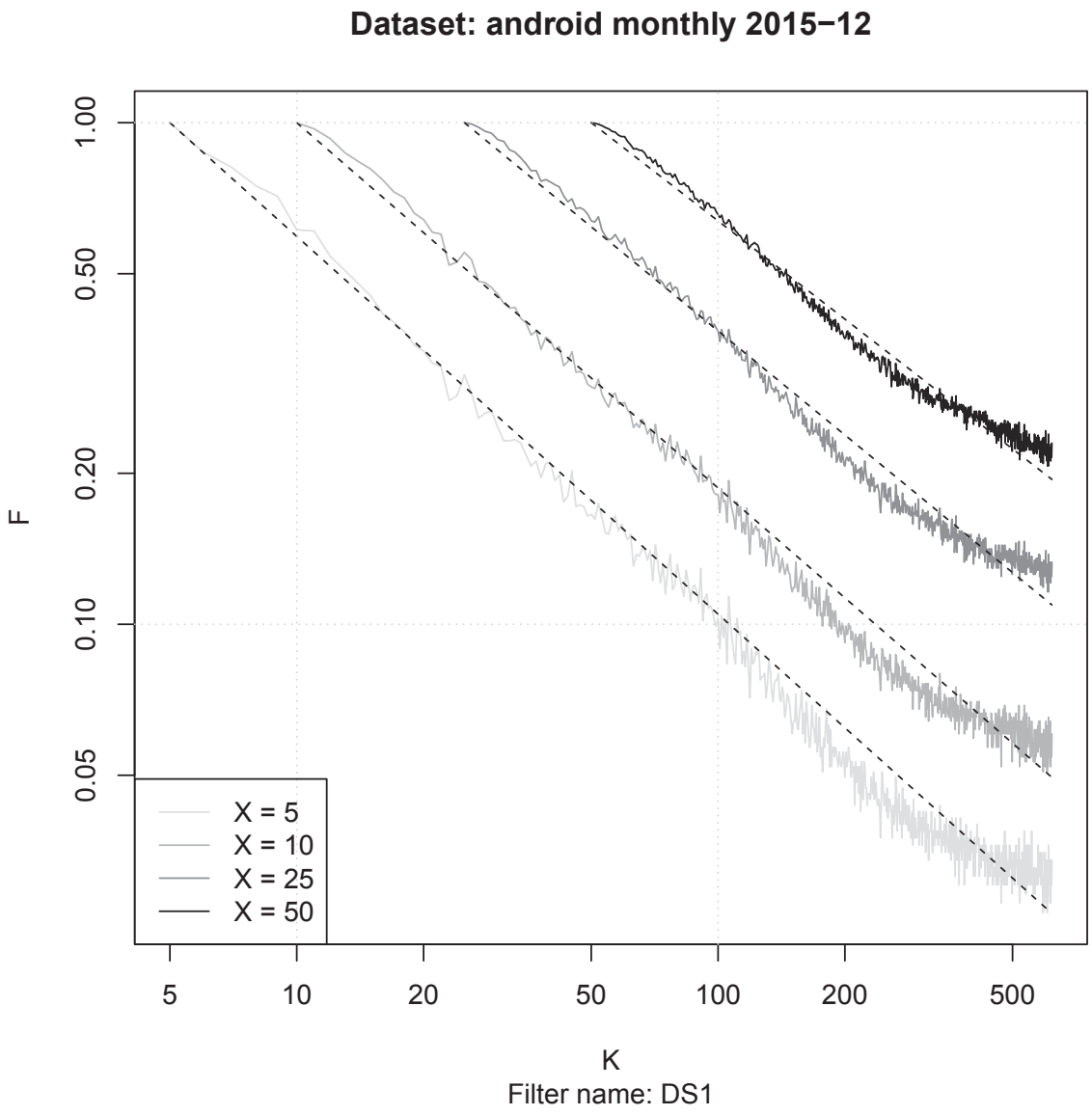


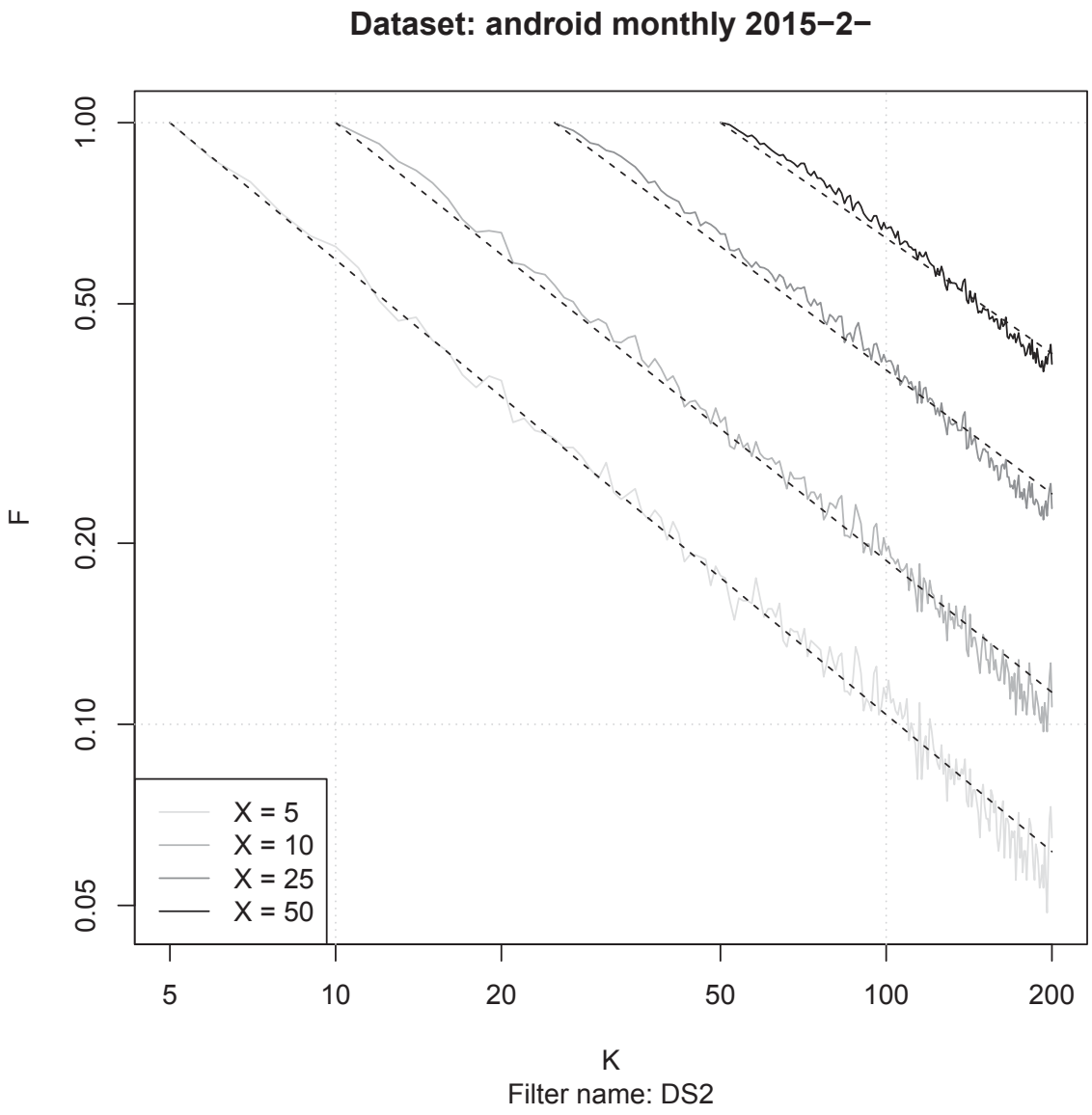

Dataset: android monthly 2015−4−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

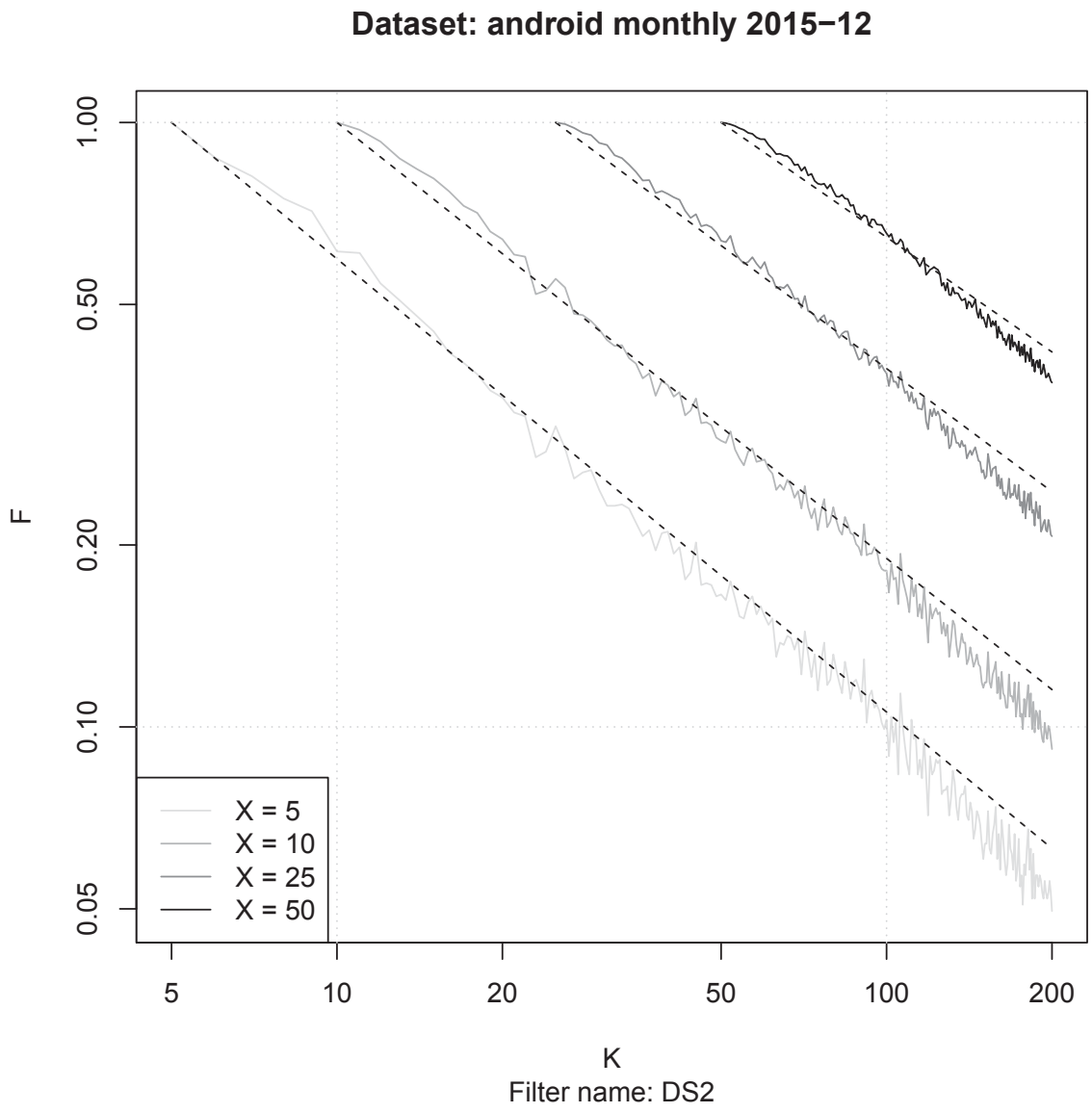

Dataset: android monthly 2015−6−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1


Dataset: android monthly 2015−8−
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

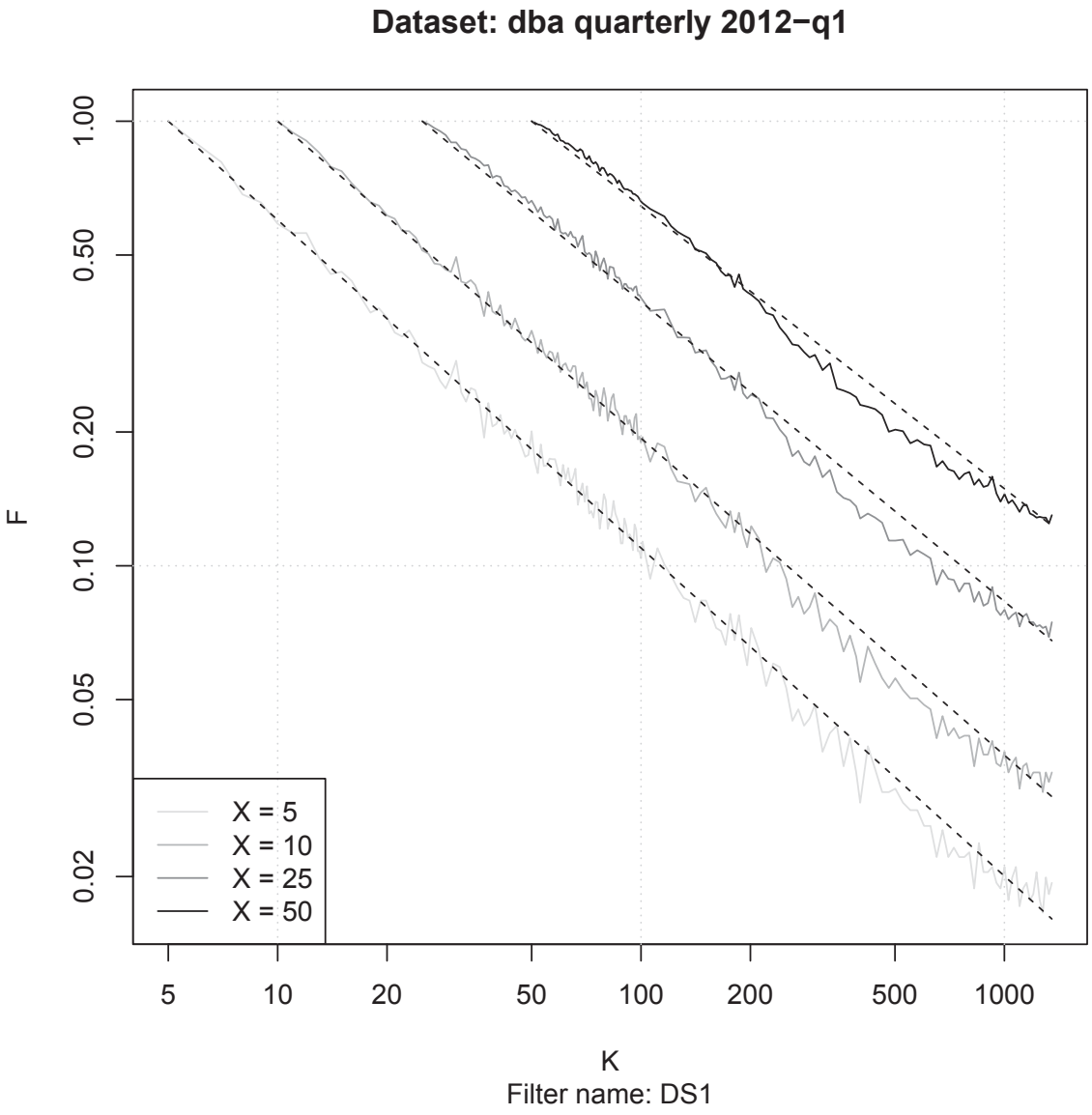

Dataset: android monthly 2015−10
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

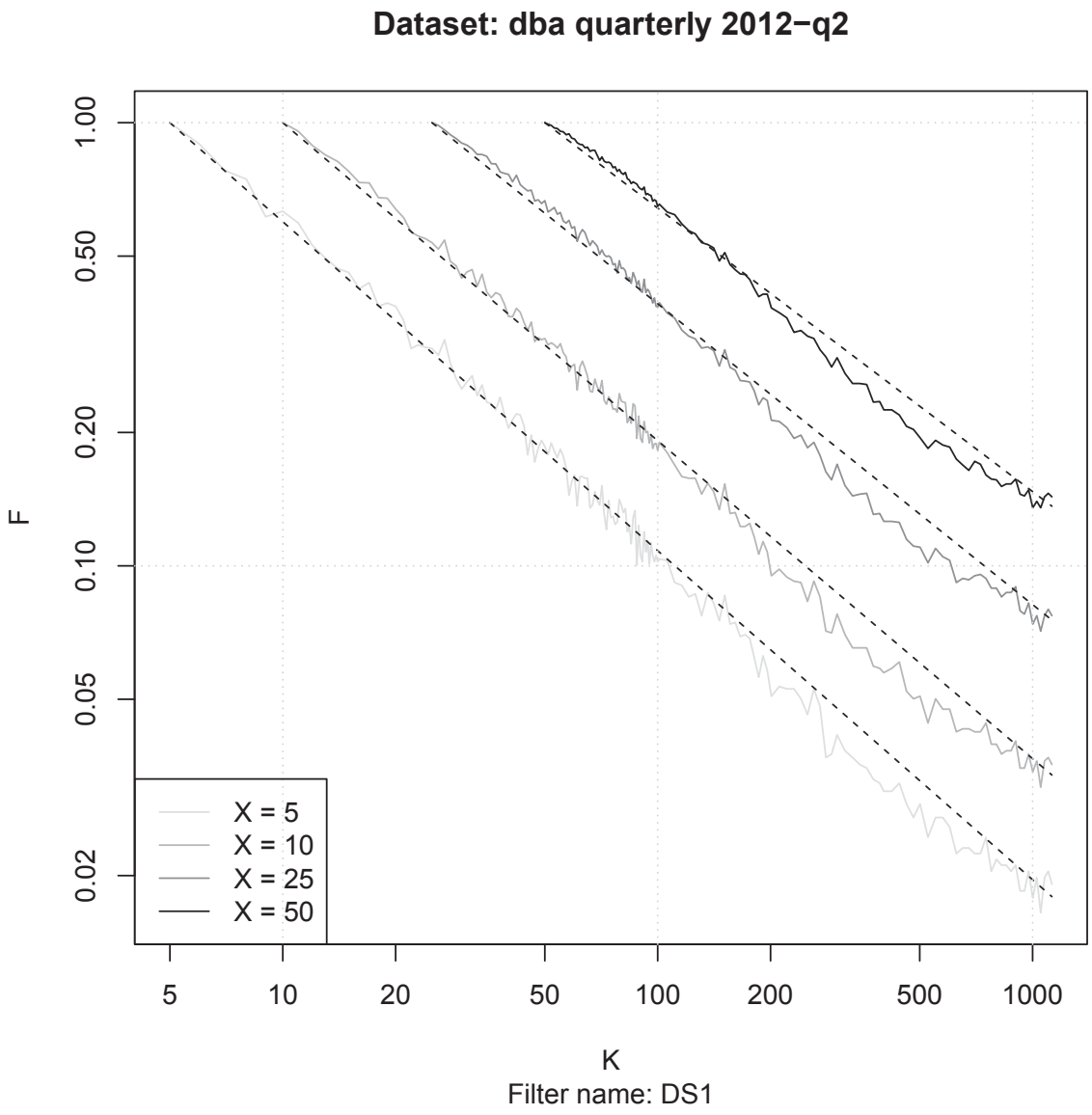

Dataset: android monthly 2015−12
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

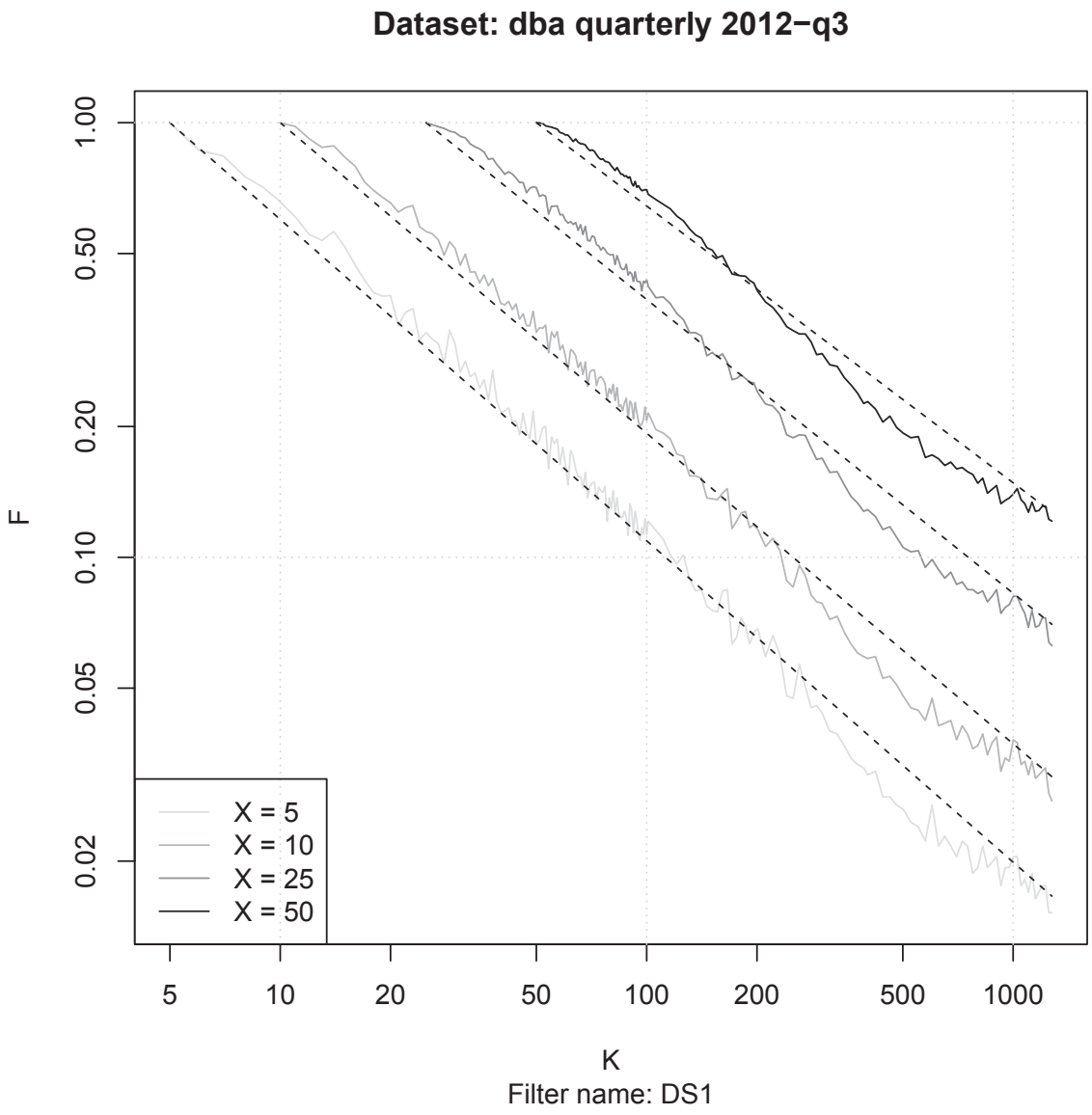

Dataset: android monthly 2015−2−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

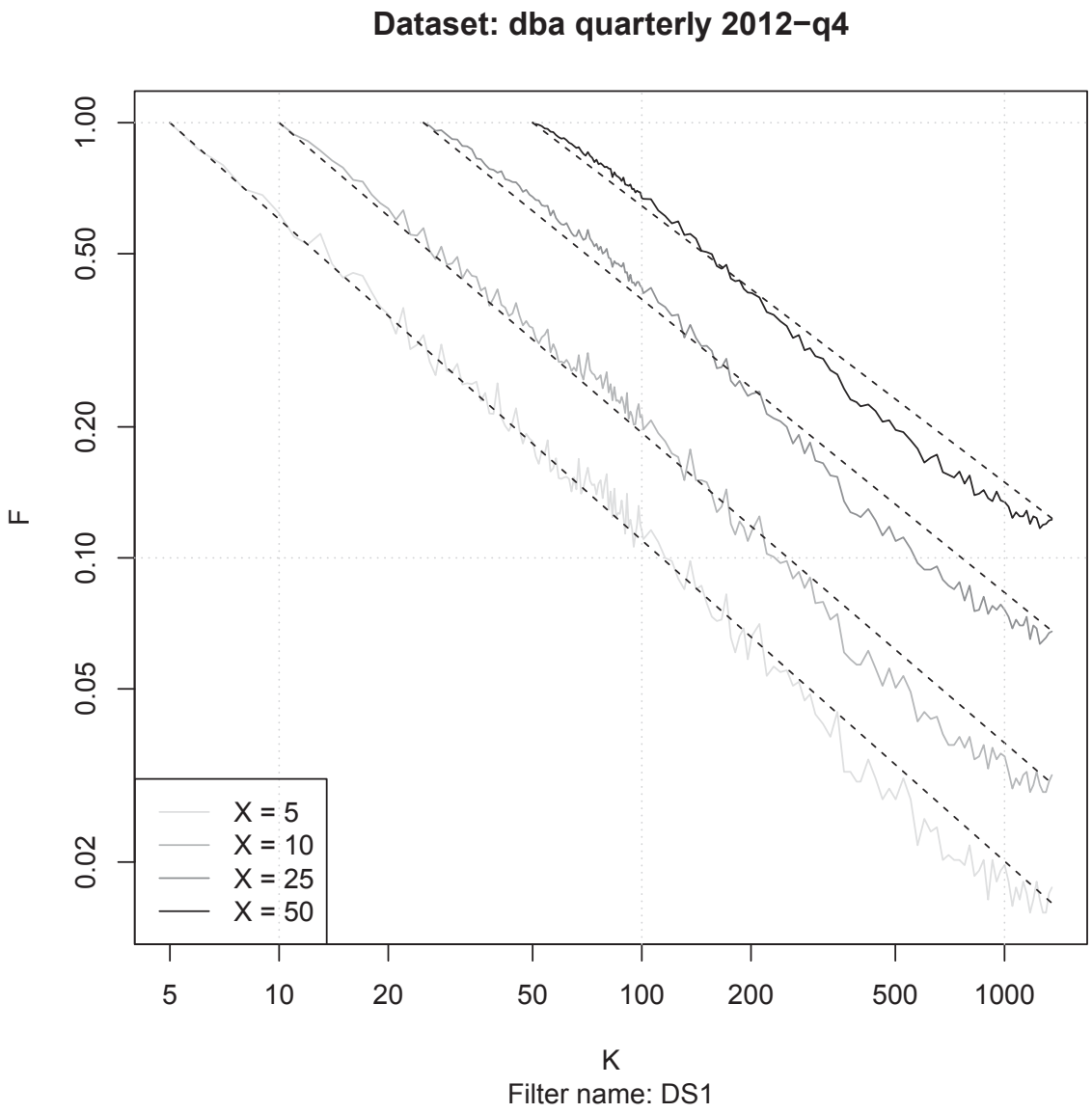

Dataset: android monthly 2015−12
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

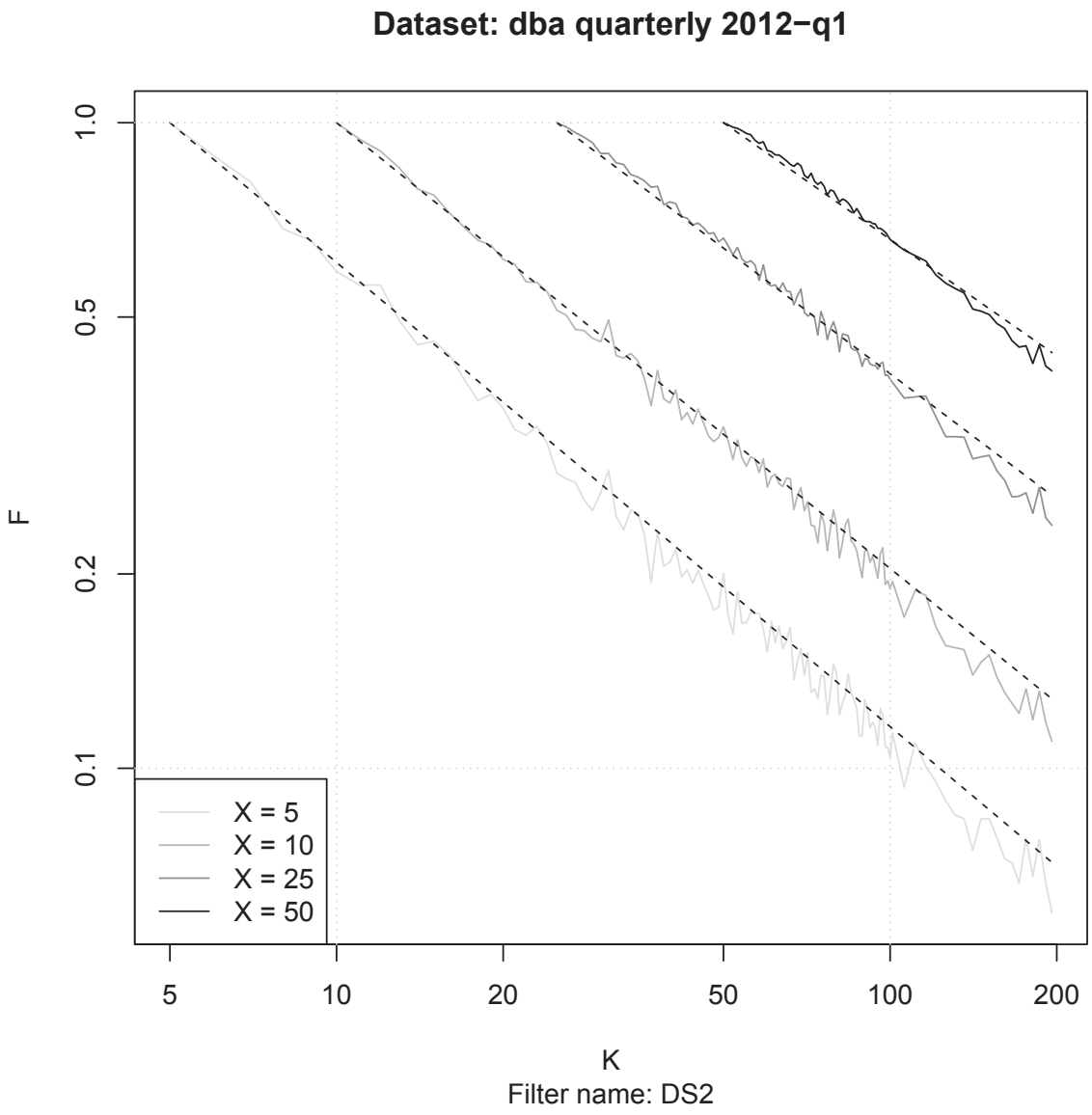
A.2 dba quarterly dataset



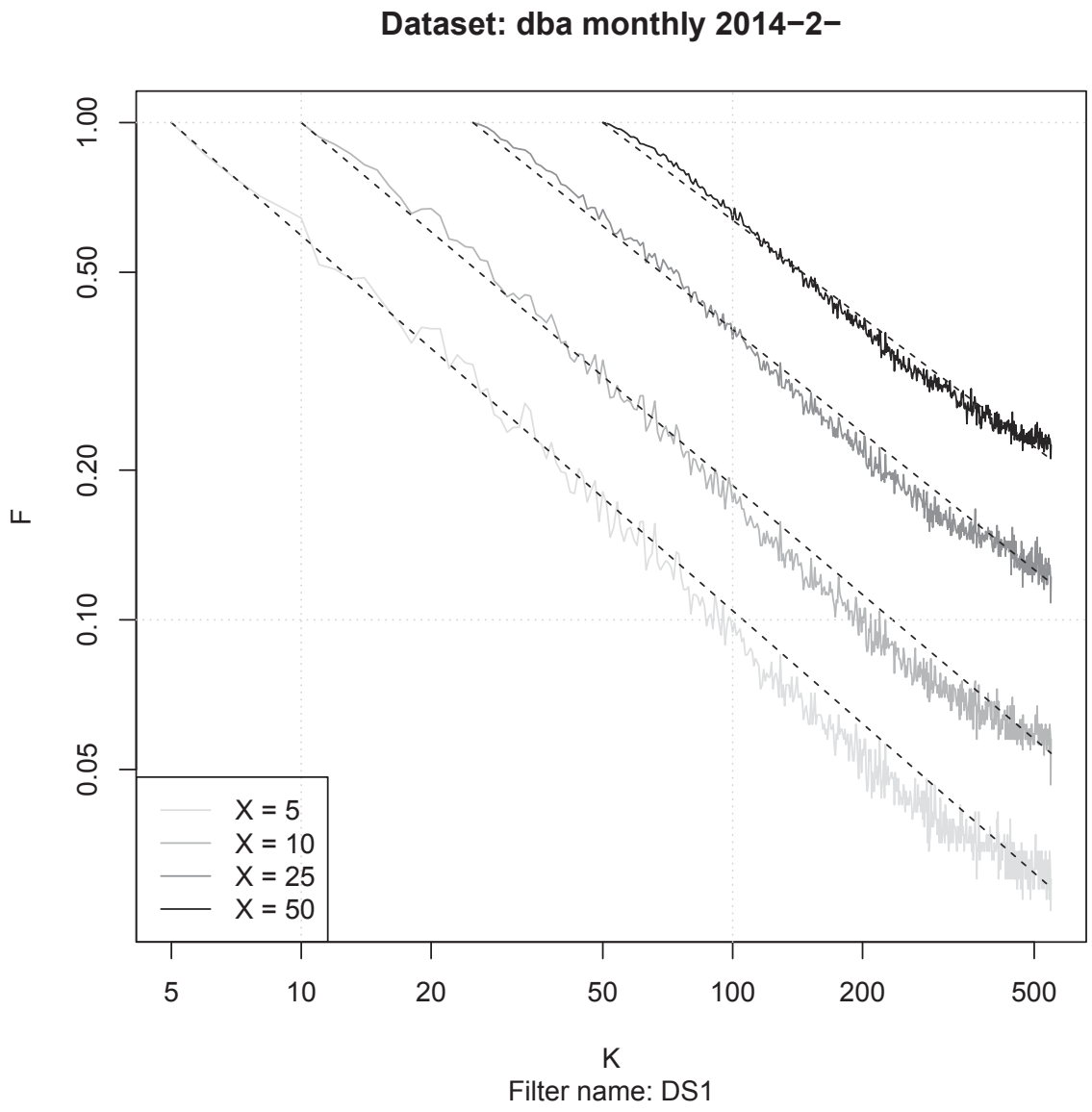

Dataset: dba quarterly 2012−q2
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1

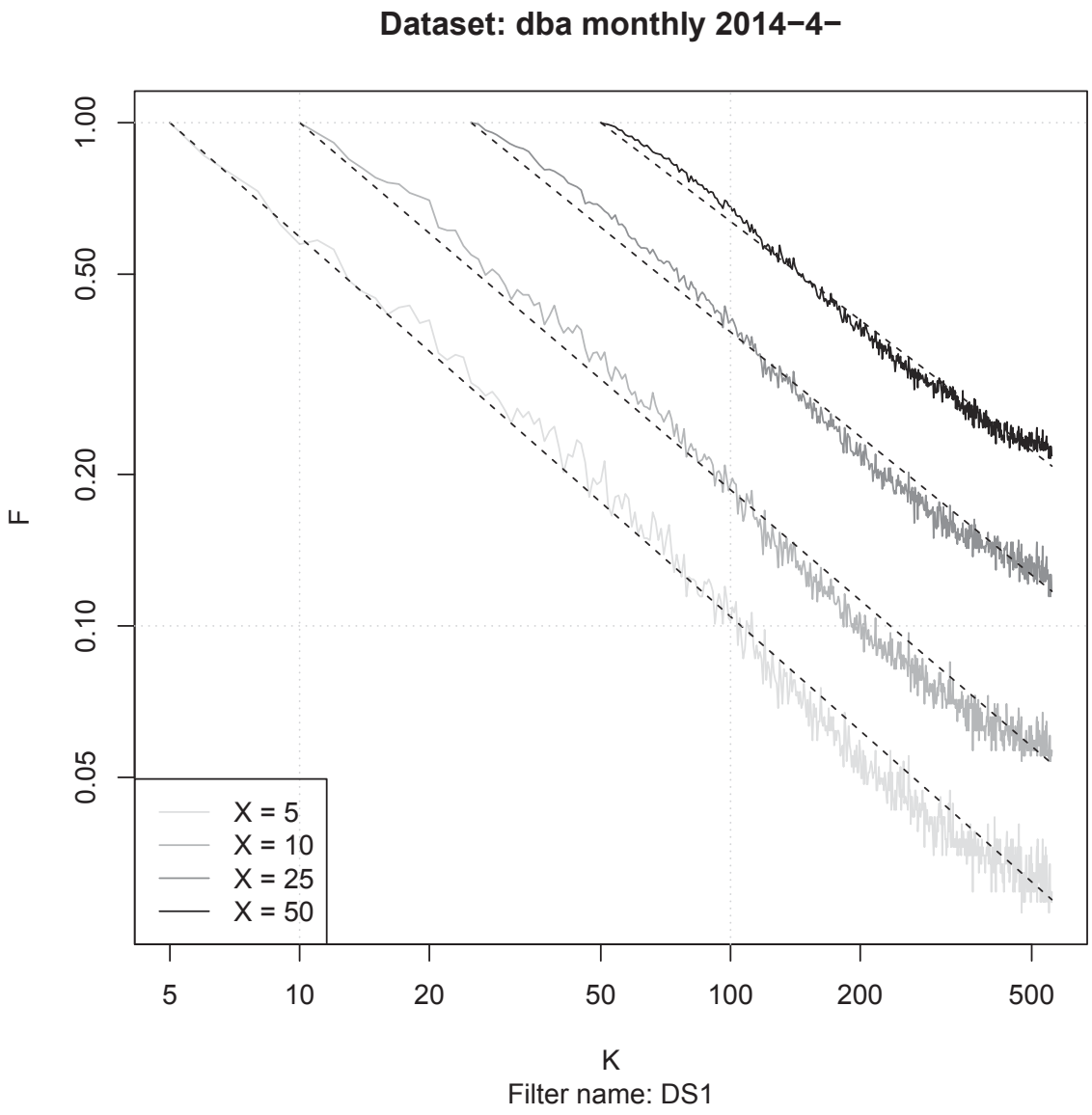

Dataset: dba quarterly 2012−q3
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1

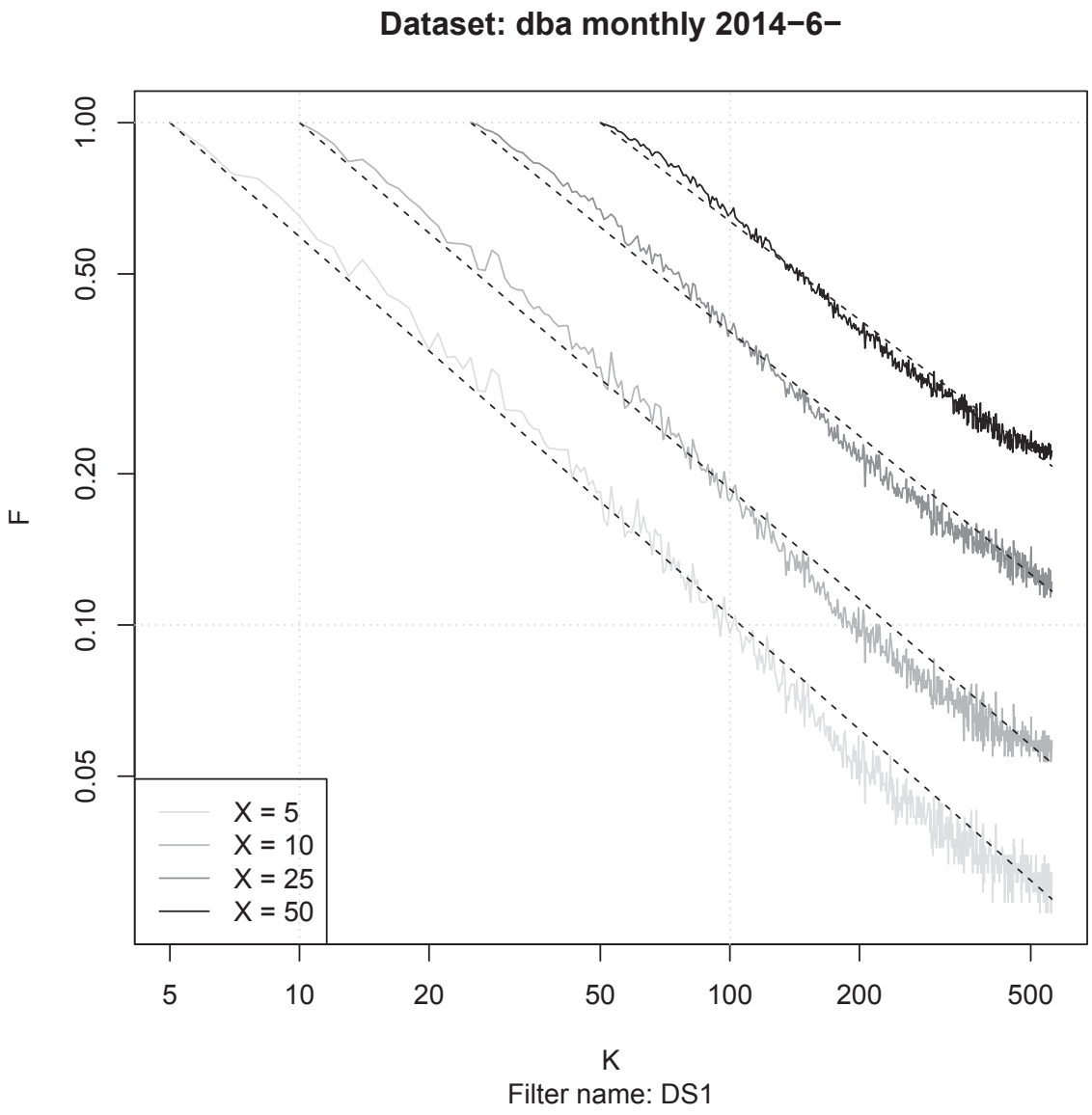

Dataset: dba quarterly 2012−q4
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1

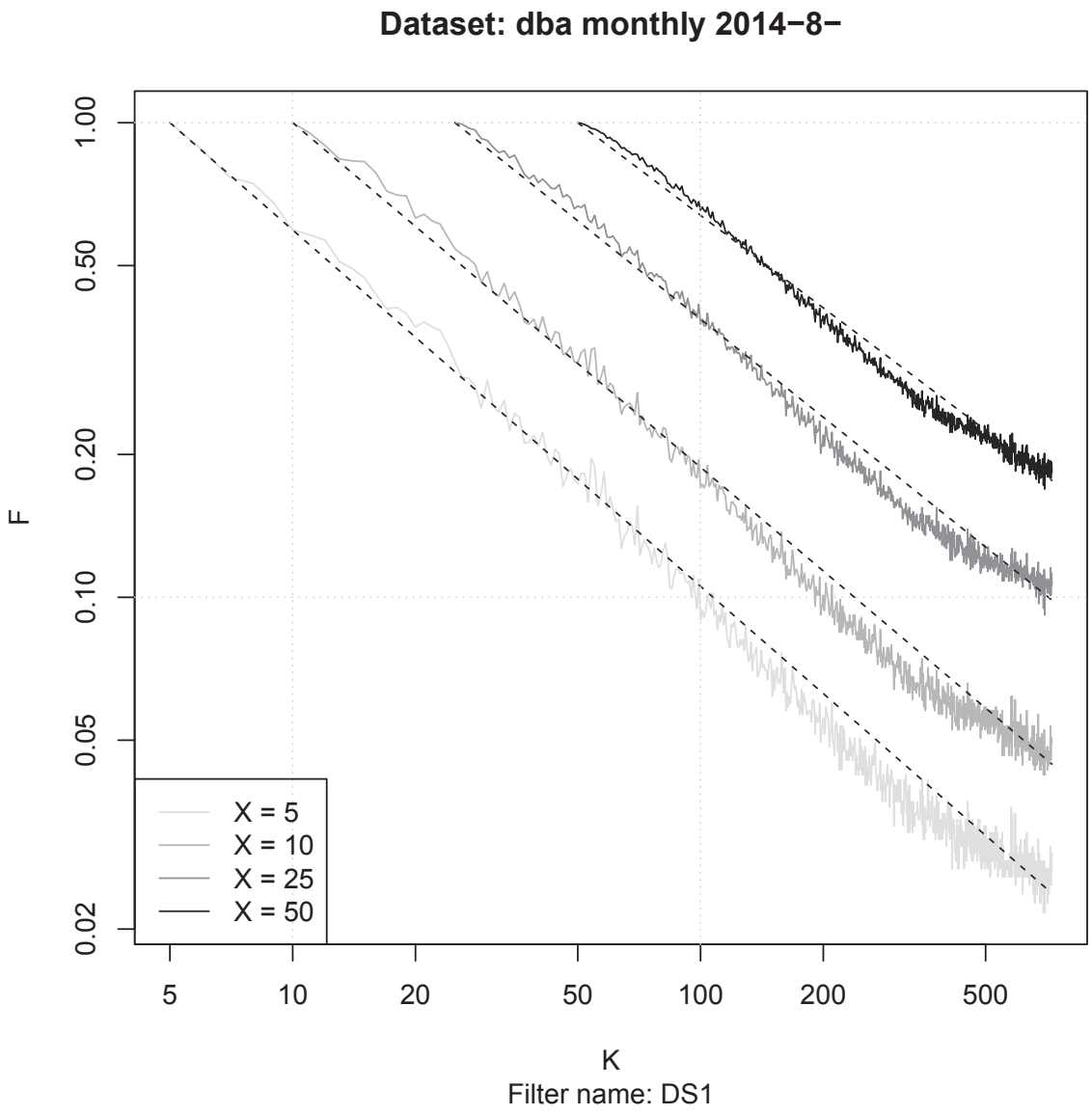

Dataset: dba quarterly 2012−q1
F
1.0
0.5
0.2
0.1
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

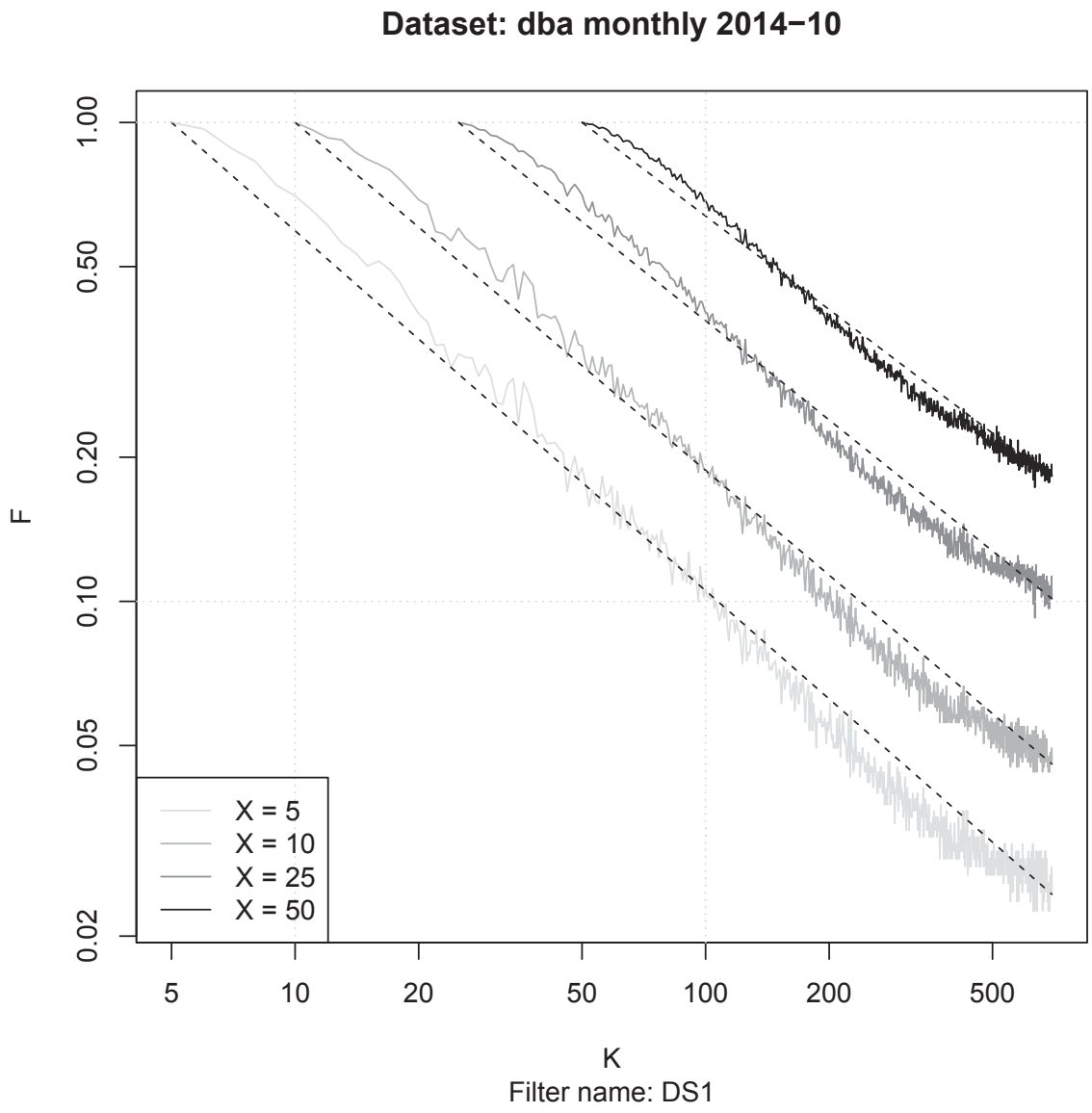
A.3 dba monthly dataset

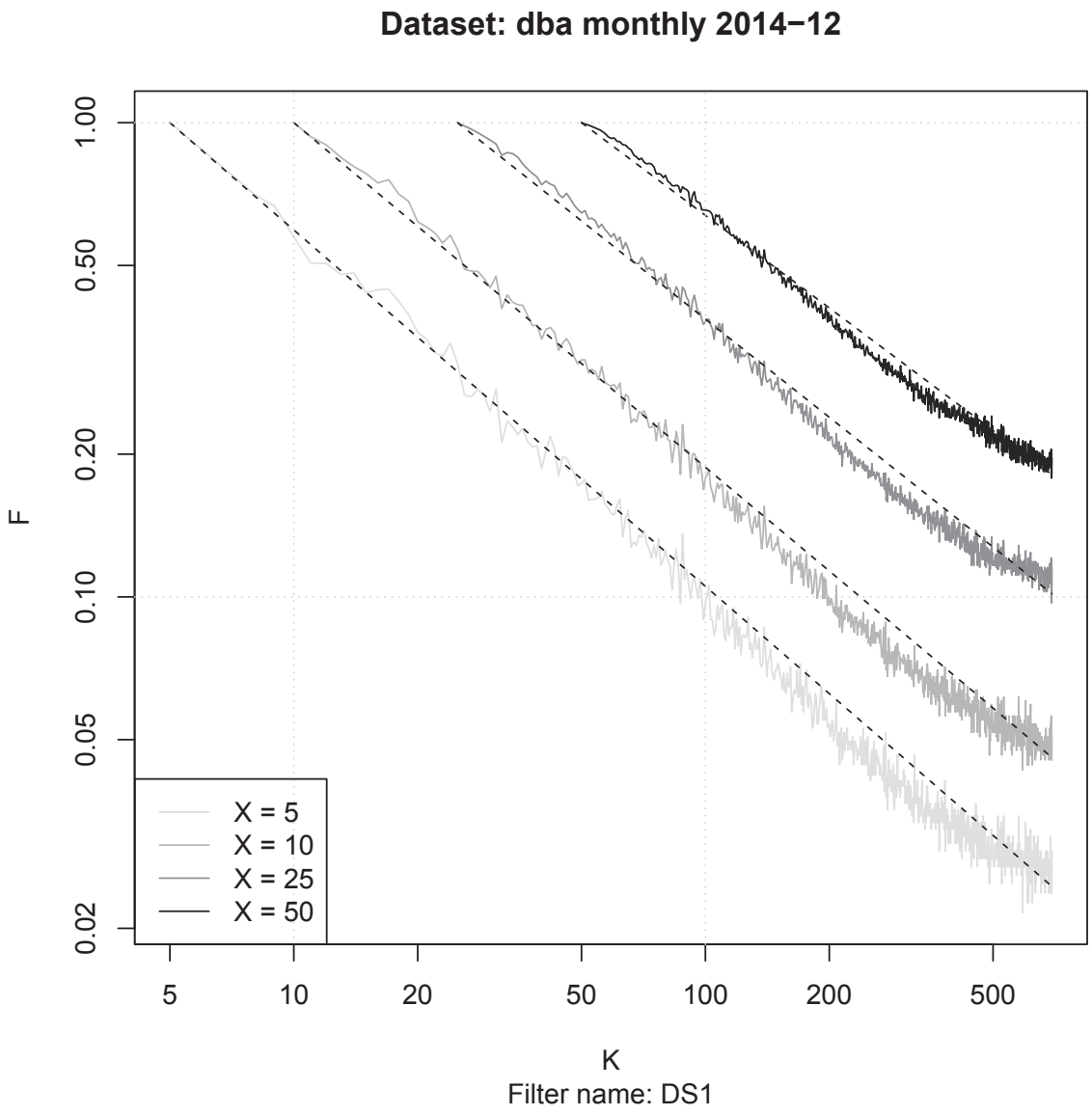


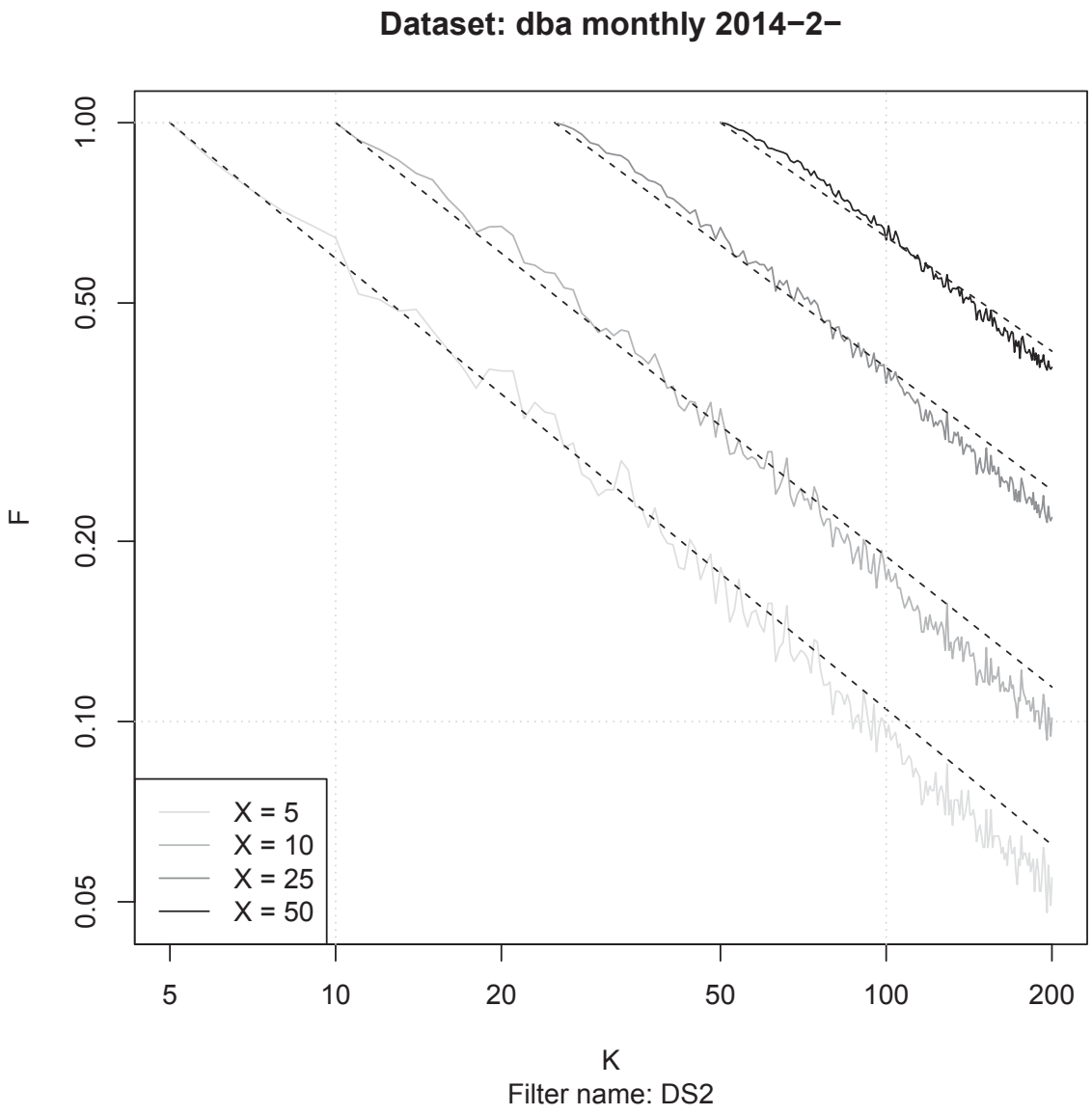

Dataset: dba monthly 2014−4−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

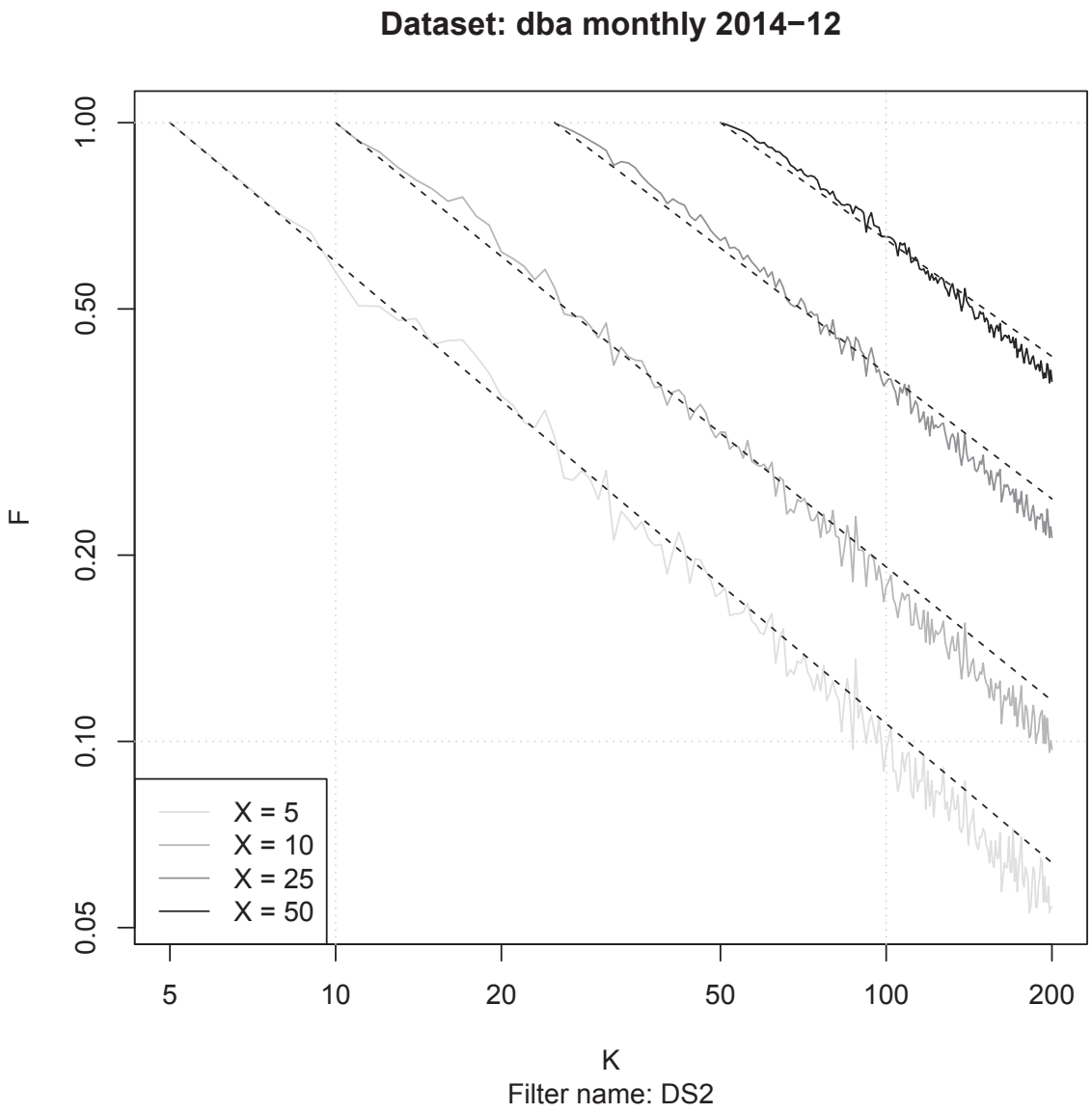

Dataset: dba monthly 2014−6−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1


Dataset: dba monthly 2014−8−
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

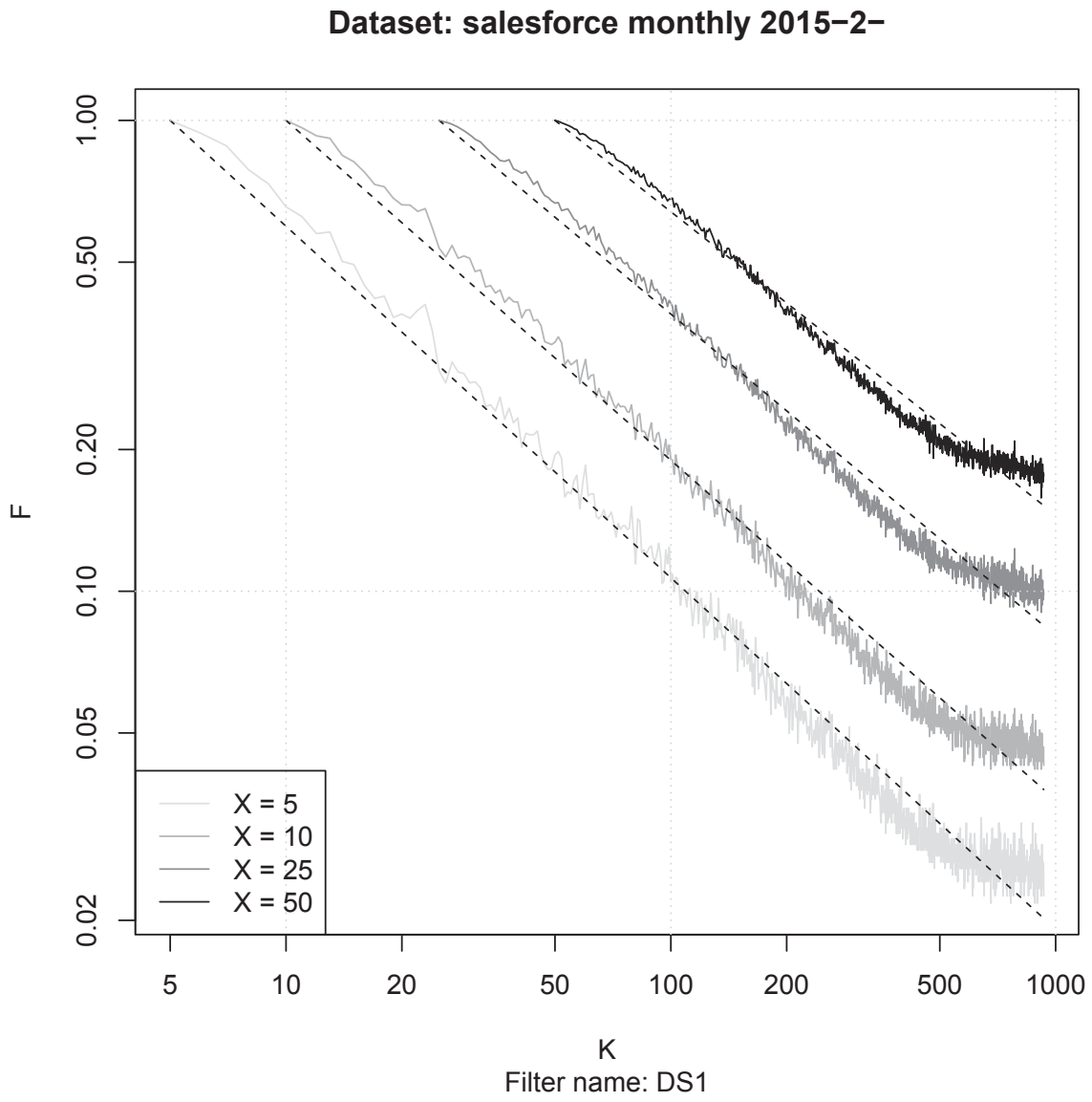

Dataset: dba monthly 2014−10
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

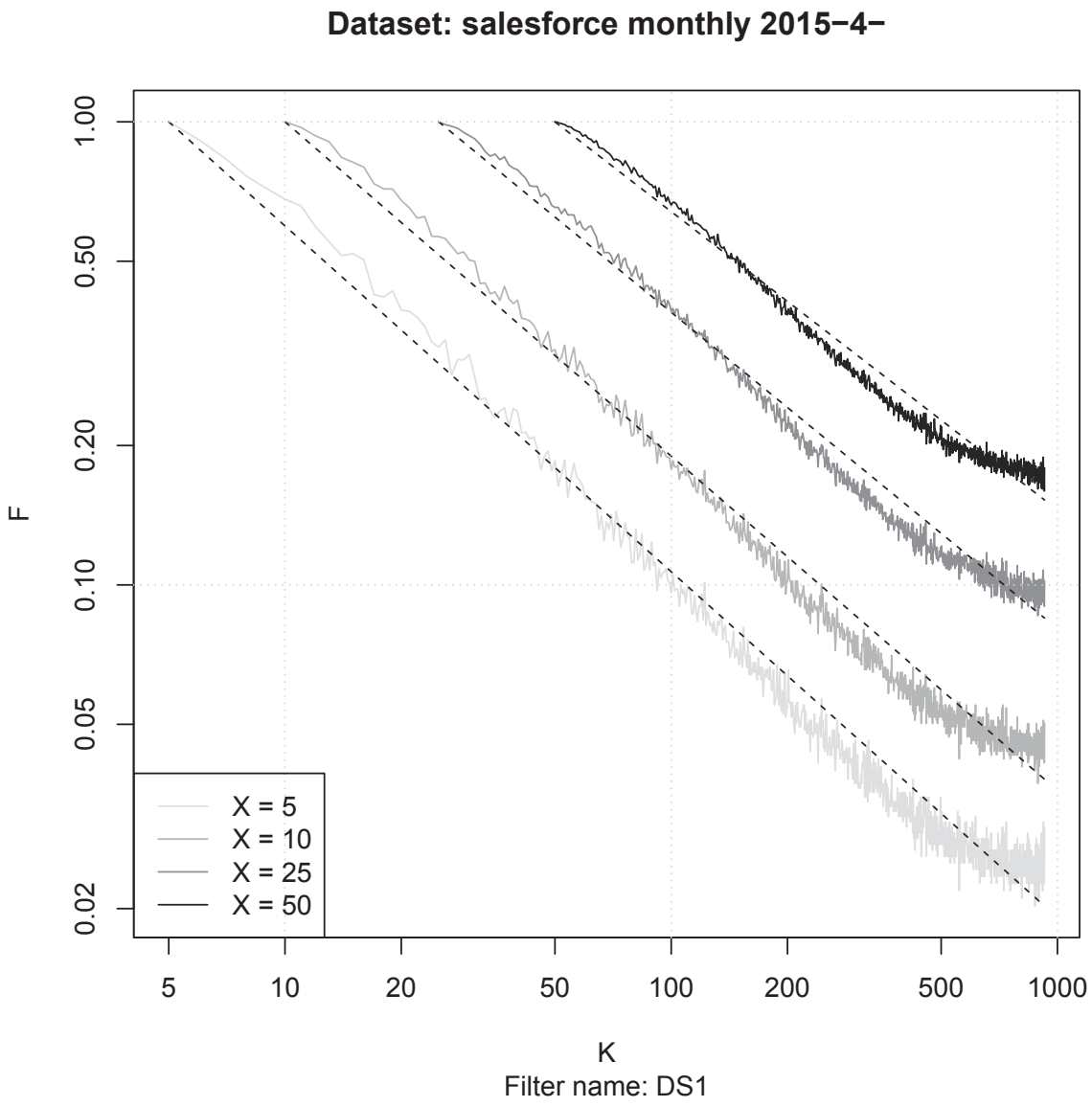

Dataset: dba monthly 2014−12
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
K
Filter name: DS1

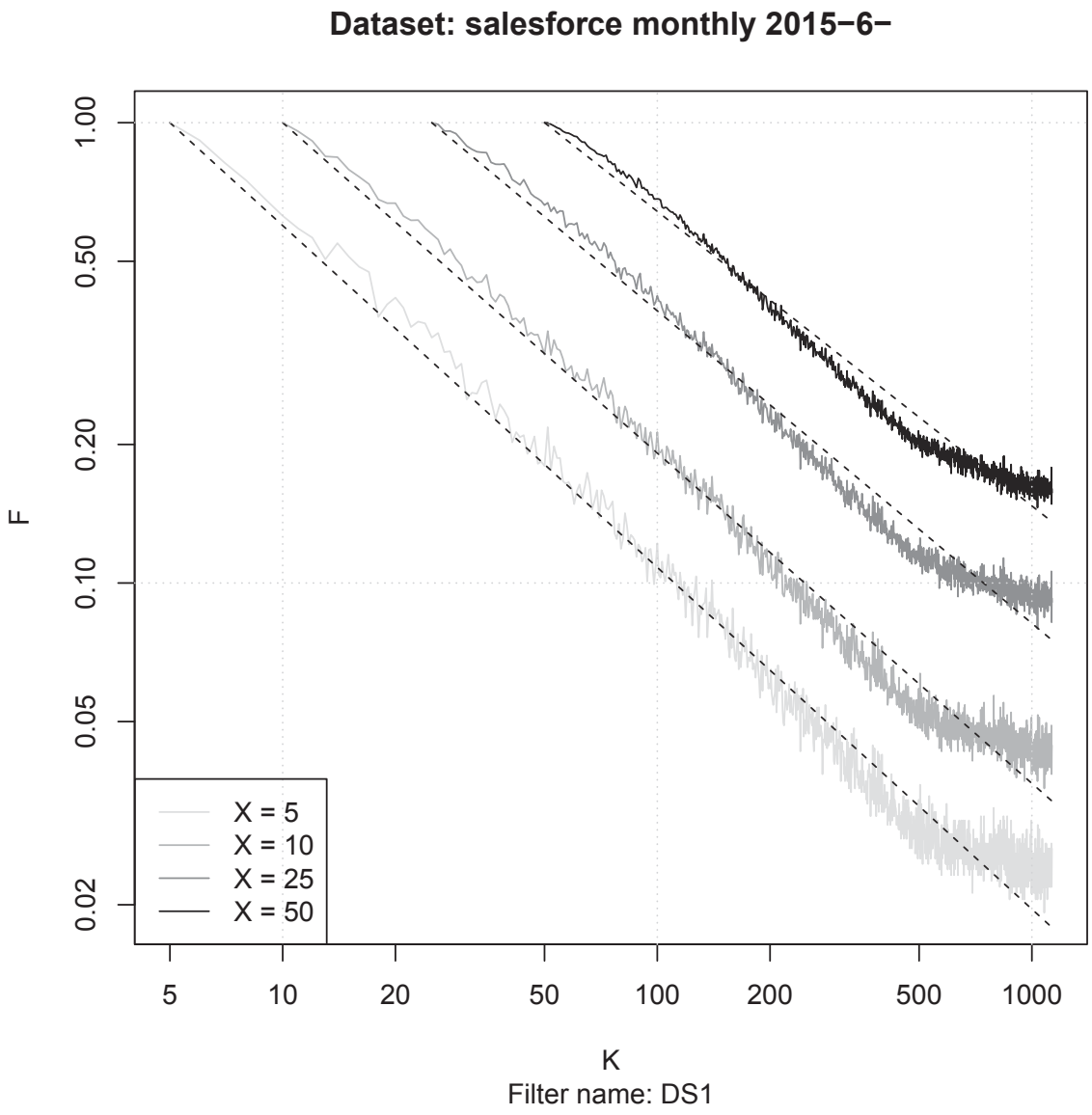

Dataset: dba monthly 2014−2−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

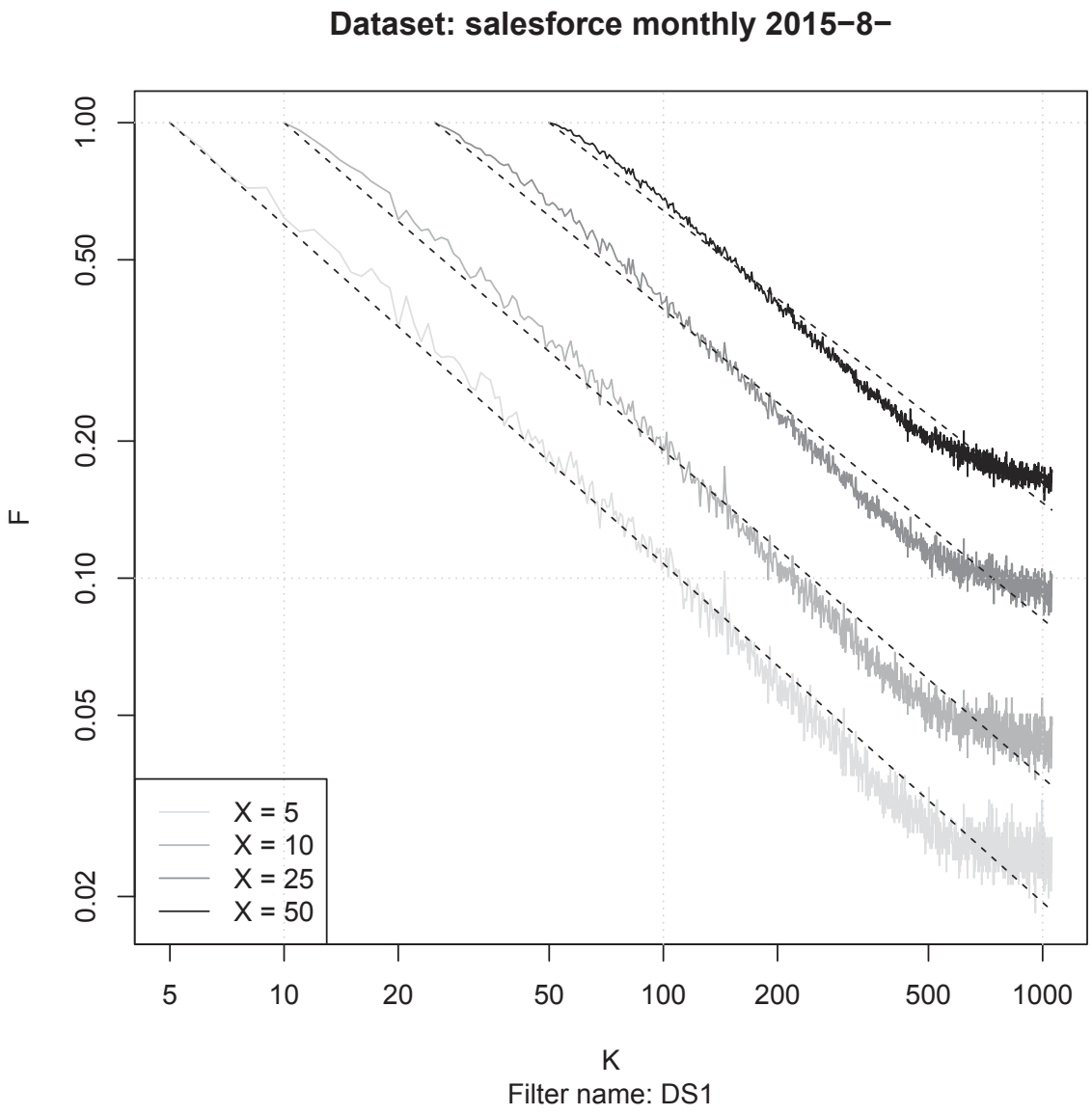

Dataset: dba monthly 2014−12
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

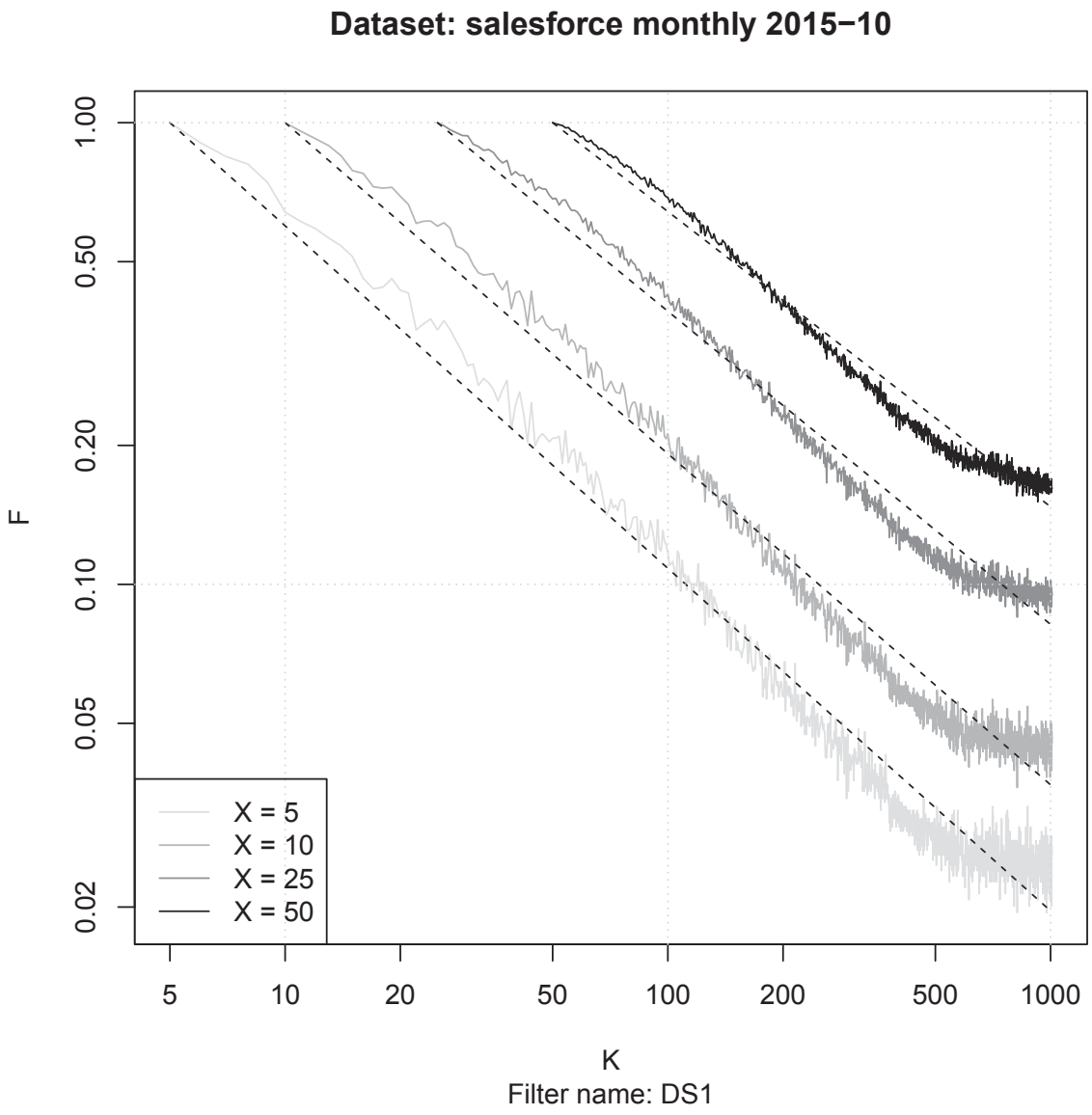
A.4 salesforce dataset

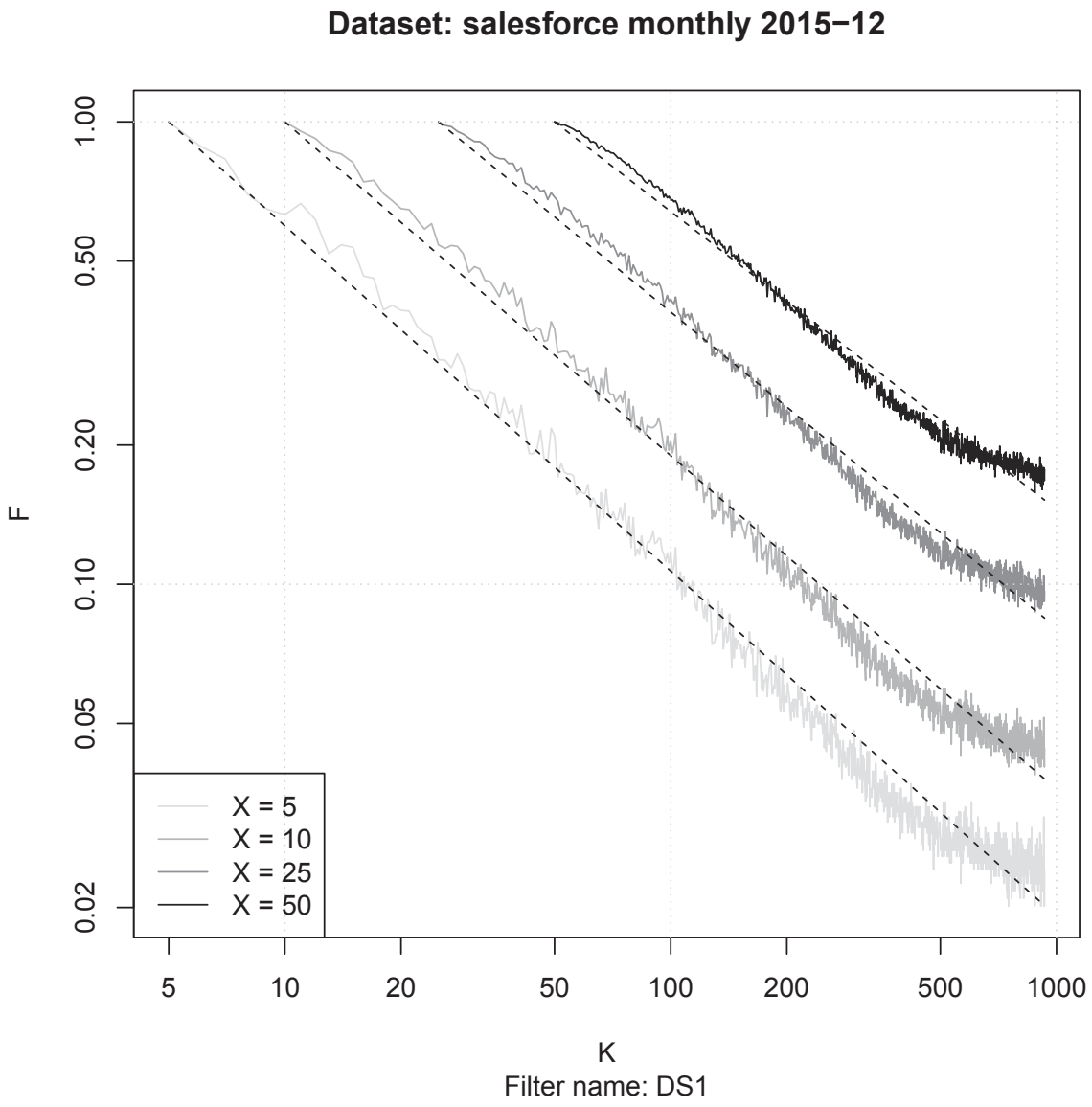


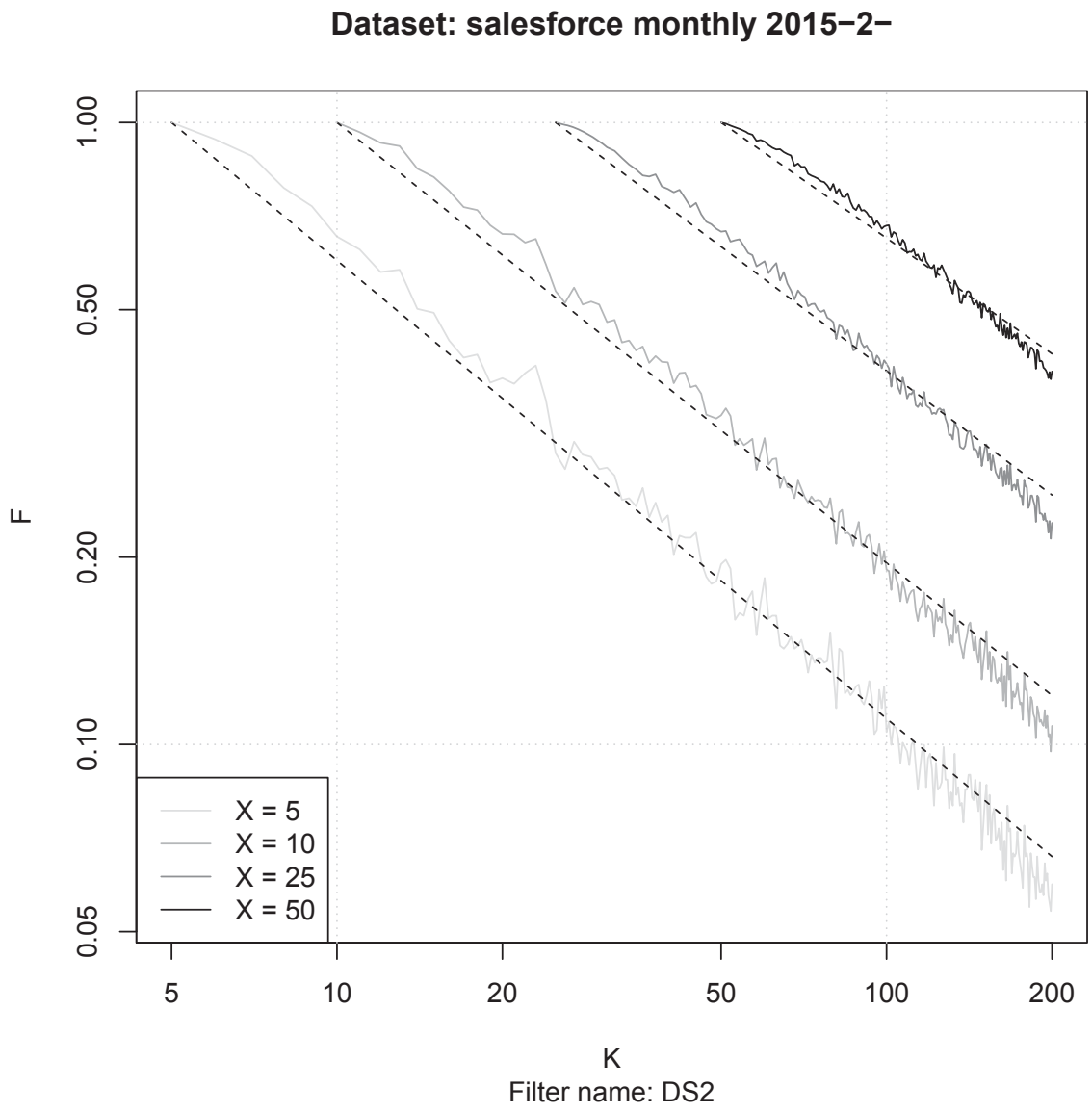

Dataset: salesforce monthly 2015−4−
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1

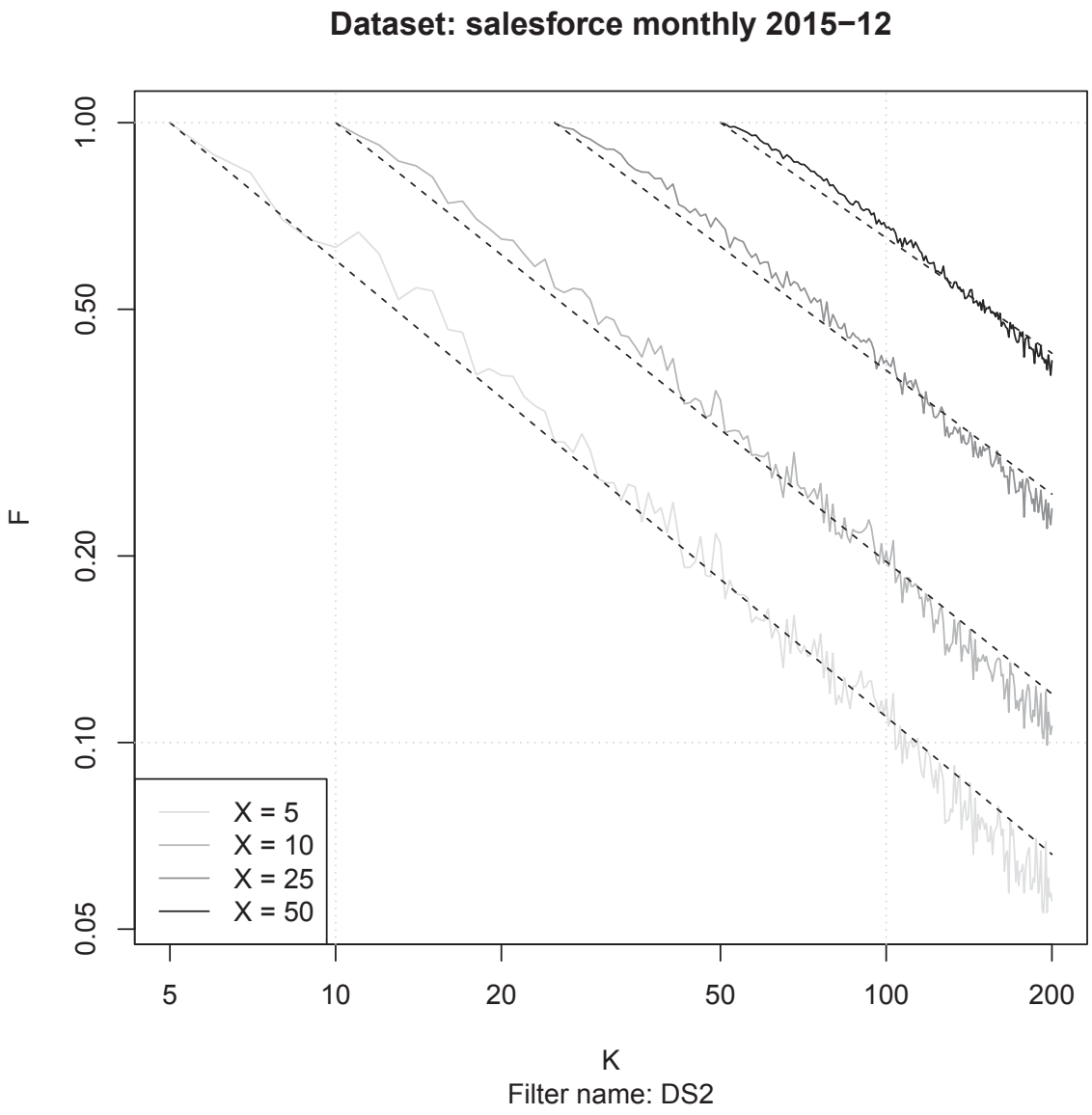

Dataset: salesforce monthly 2015−6−
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1


Dataset: salesforce monthly 2015−8−
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1


Dataset: salesforce monthly 2015−10
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1


Dataset: salesforce monthly 2015−12
F
1.00
0.50
0.20
0.10
0.05
0.02
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
500
1000
K
Filter name: DS1


Dataset: salesforce monthly 2015−2−
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2


Dataset: salesforce monthly 2015−12
F
1.00
0.50
0.20
0.10
0.05
X = 5
X = 10
X = 25
X = 50
5
10
20
50
100
200
K
Filter name: DS2

Appendix B

Code listings

The following Table (B.1) is the list of scripts developed for the implementation and calibration of LDA.

Table B.1: List of programs included in this appendix

Name	Language	Objective
xml2csv.pl	Perl	Convert format of corpus from XML to CSV
do_analysis_db2_idf.R	R	LDA inference for dba corpus with idf calculation
utilsdb2.R	”	Utility functions for dba LDA inference
do_lda_analysis_topics.R	”	LDA inference for monthly datasets
do_lda_quarter_inc.R	”	LDA inference for quarterly dataset
utils.R	”	Utility functions for dba LDA inference (monthly and quarterly)
convVar_k200.R	”	Binds topic frequency files to be analyzed with the regression model
verifyFit_k200.R	”	Iterates over datasets, and computes the RMSE of the approx. formulas
verifyFit_k200_analysis.R	”	Plots the RMSE boxplots per model and per dataset name
validate_fit.R	”	Performs validation using 10 folds

B.1 Program: xml2csv.pl

```
1 #------------------------------------------------------------------------------
2 #This script reads in XML data from StackOverflow and returns it in text format
3 #Output "title \t body", one line per post
4 #
```

```
5  #Usage example: perl xml2csv.pl Posts.xml Posts.xml.csv
6  #--------------------------------------------------------------------------------
7  use warnings;
8  use strict;
9
10 use XML::Simple;     # use XML::Twig for big files
11 use HTML::Strip;
12
13 use utf8;
14 use Text::Unidecode;     #translates unicode into ascii
15
16 my $minVoteCount = 3;    #minimum number of votes required to save an answer
17
18 sub transform;
19
20 if ( scalar(@ARGV) != 2 ) {
21  print "usage: xml2csv.pl in_file_name out_file_name\n";
22  exit;
23 }
24
25 #read XML file into memory
26 my $ref = XMLin( $ARGV[0] );
27
28 open( my $fid, ">" . $ARGV[1] )
29   or die("Cannot open $ARGV[1] : $!\n");     #export to CSV
30
31 my $hs = HTML::Strip->new();
32 print $fid "create_ts\tid\ttags\ttitle\tbody\tanswers\n";     #header
33 foreach ( @{ $ref->{row} } ) {
34  if ( $_->{PostTypeId} == "1" ) {     #if the post is the original question
35   my $creationDate = ( $_->{CreationDate} );
36   my $id           = $_->{Id};
37   my $tags         = $_->{Tags};
38   my $bestAnswerId = $_->{AcceptedAnswerId};
39
40   #remove "<" and replace ">" with " "
41
42   $tags =~ s/></^&^/g;
43   $tags =~ s/^<//g;
44   $tags =~ s/>$//g;
45
46   my $title    = transform( $_->{Title} );
47   my $body     = transform( $hs->parse( $_->{Body} ) ); #strip HTML tags
```

```
48    my $answers = getAnswersText( $id, $minVoteCount, $bestAnswerId );
49    print $fid join( "\t", ( $creationDate, $id, $tags, $title, $body, $answers ) )
          . "\n";
50   }
51  }
52  close $fid;
53  $hs->eof;
54
55  #convert unicode to ascii and make it lower case
56  sub transform {
57   my ($txt) = @_;
58   $txt =~ s/\n|\t|\r/ /g;          #replace new line or tab with space
59   $txt =~ s/[[:punct:]]/ /g;     #replace punctuation with space
60   return lc( unidecode($txt) ); #convert #convert unicode to latin and change
         words to lowercase
61  }
62
63  sub getAnswersText {
64   my ( $questionId, $minVote, $AcceptedAnswerId ) = @_;
65   my $answers = "";
66
67   foreach ( @{ $ref->{row} } ) {
68    if (
69     (
70      ( $_->{PostTypeId}    eq "2" )            #record type = answer
71      and ( $_->{ParentId} eq $questionId )     #answer to question $questionId
72      and ( ( $_->{Score} + 0 ) >= $minVote )  #minimal vote
73     )
74     or (                                          #always get accepted answer
75      defined($AcceptedAnswerId) and ( $_->{Id} eq $AcceptedAnswerId )
76     )
77      )
78    {
79     $answers .= transform( $hs->parse( $_->{Body} ) ) . " ";
80    }
81   }
82   return $answers;
83  }
```

B.2 Program: do_analysis_db2_idf.R

```
1  ##-------------------------------------------------------------------------
2  ## The script reads posts in delimited format, and generates a list of keyword
3  ## Usage: Rscript my_Script_name.R out_directory/ in_file_name yyyy
4  ## This program includes the probability per keyword
5  ##         also includes the specificity by keyword by topic
6  ##-------------------------------------------------------------------------
7  library(tm)
8  library(foreach)
9  library(doParallel)
10 library(stringr) ## added to handle string functions
11
12 memory.limit(24356)
13 memory.size(max = TRUE)
14
15 source("utilsdb2.R")
16 xURL <- "http://dba.stackexchange.com/questions/"
17 xSystime <- format(Sys.time(), "%a-%b-%d_%H-%M-%S_%Y")
18 ## following lines commented out to run in PC ##
19 ##args <- commandArgs(trailingOnly = TRUE)
20 ##outDir <- args[1]
21 ##readFrom <- paste(args[1], args[2], sep="")
22 ##year  <-  as.integer(args[3])
23
24
25 ## following lines hardcoded to run in PC ##
26 readFrom <- "nondb2.awk" ##"Posts.xml.filtered.csv.new" ##"db2.awk"
27 year <- 2015
28 ##
29 ##month <-  as.integer(args[3])
30
31 topKeywordCount <- 20 ## SET TO 20 FOR THIS RUN
32
33 ## corp <- createCorp(readFrom)
34 corp <- createCorp(readFrom, year) ## added year
35
36 # To access original id of a document in VCorp run corp[[ index_of_document_1_to_
       N ]]$meta$id
37 # To access original id of a document in DocumentTermMatrix run dtm$dimnames$Docs
       [ index_of_document_1_to_N ]
38
39 cat("DEBUG after calling createCorp!\n")
```

```
40 ## Build a Document-Term Matrix
41 dtm <- DocumentTermMatrix(corp, control = list(minWordLength = 2)) #keep words of
       lenght 2 or longer
42 cat("Before tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n")#
     dtm <- removeFrequentWords(dtm) #removing based on median tf-idf value
43 cat("After tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n"
     nlikely event)
44 cat("After removing terms appearing only in 1 document: term count =", ncol(dtm),
       ", doc count =", nrow(dtm), "\n")
45
46 #setup parallel backend to use 8 processors
47 cl<-makeCluster(8) ## change to 8 for multiprocessing
48 registerDoParallel(cl)
49
50 foreach(topicCount = c(5) #max = 1 topic per document ## changed topic count=5
51         , .packages='topicmodels' #include package
52 ) %do%
53 { #change to %dopar% for multi execution
54   cat("*RUNNING SINGLE PROCESSING*\n")
55   cat("TopicCount:", topicCount, "\n") #to screen (no screen output is parallel
       mode)
56   mdl <- LDA(dtm, topicCount) #LDA model
57   topic.keyword <- terms(mdl, topKeywordCount)
58   mdl.post <- posterior(mdl) #get posterior data
59 }
60
61 ## the following nested loop was added to append the p-value per topic.keyword
62 Ttopic.keyword <- topic.keyword ## make a copy
63
64 ## the following nested loop was added to write the p-values of topics by column
65
66 outPval <- paste("pvalues", xSystime, "_", readFrom, "_", year, "-", topicCount,
     ".txt", sep = "")
67 for (j in 1:topicCount) {
68   cat("TOPIC ", j, "\n", append = T,
69       file = outPval)
70   cat("KEYWORD,PROBABILITY\n", append = T,
71       file = outPval)
72   for (i in 1:topKeywordCount){
73
74     Ttopic.keyword[i,j] <-
75       Ttopic.keyword[i,j] <- paste(topic.keyword[i,j], ",", mdl.post$terms[[j,
           topic.keyword[i,j]]])
```

```
76      cat(Ttopic.keyword[[i,j]],"\n", append = T,
77          file = outPval)
78    }
79  }
80
81  write.csv(Ttopic.keyword,
82            file = paste("pvalues_", xSystime, "_", readFrom, "_", year, "-",
                  topicCount, ".csv", sep = "") )
83
84  ## code added to support report creation
85
86  ## Will retrieve all ids and questions
87  t <- topics(mdl)
88  len <- length(topics(mdl))
89  saveTo <- paste("Report-", xSystime, "_", readFrom, "-", year, ".txt")
90  cat("question_id\tquestion_description\ttopic\t\n", sep="\t", append = T, file =
      saveTo) # to file
91  for (i in 1:len){
92    ri <- corp[[i]]$meta$id
93    rq <- corp[[i]]$meta$quest
94    rt <- t[i]
95    r <- paste(xURL, ri, "\t", rq, "\t", rt, sep="")
96    cat(r, "\n", append = T, file = saveTo) # to file
97  }
98
99  ## The following code was added for generating the specificity per keyword
100
101 ## t is number of terms in dtm
102 ## tC is the index of topicCount
103 ## kC in the index of topKeywordCount
104 ## topKeywordCount is the number of keywords
105 ## dC is the index of number of topics
106 ## countKw is the counter of the keyword that appears in the topics
107
108 allDocuments <- nrow(dtm) ## Size in rows of dtm
109 allKeywords <- ncol(dtm)
110 countKw <- matrix(0, topKeywordCount, topicCount) ## initialize count keyword
      matrix with 0
111
112 ## code added to calculate frequencies
113 countKw <- array(0,dim=c(topKeywordCount, topicCount)) ## intermediate array to
      handle the counting of kW in allDocuments
114 t <- topics(mdl) ## documents and the topics it belongs to
```

```
115 z <- as.matrix(dtm) ## dtm as matrix
116 dfz <- as.data.frame(z) ## converts z to data frame
117 tK <- 0 ## array(0,dim=c(topKeywordCount, topicCount))
118
119 acumTo <- rep(0,topicCount) ## initialize acum of topics
120 for(i in 1:topKeywordCount){
121       acumKw<-0
122       for(j in 1:topicCount){
123        xlookup <- topic.keyword[i,j]
124        colZ <- which(colnames(z)==topic.keyword[i,j]) ## finds the column in z
               that the kW is in
125
126        for(k in 1:allDocuments){
127          if(dfz[[colZ]][k] > 0) {
128           tK <- t[[k]]  ## indek tk is the topic number
129
130            if(tK==j){
131              if(dfz[[colZ]][k] > 0){
132                  acumKw <- acumKw + 1
133              }
134            }
135          }
136        }
137
138        if(!is.null(acumKw)){
139             countKw[i,j] <- acumKw
140             acumKw <-0
141        }
142       }
143
144 }
145
146 ## Calculation of N as the number of documents per topic
147 for(i in 1:nrow(dtm)){
148   topic<-t[[i]]
149   acumTo[topic]<-acumTo[topic]+1
150 }
151
152 ## the following nested loop was added to write the topics by column
153 outIdf <- paste("idf_", xSystime, "_", readFrom, "_", year, "-", topicCount, ".
       txt", sep = "")
154 for (j in 1:topicCount) {
155   cat("TOPIC ", j, "\n", append = T,
```

```
156         file = outIdf)
157     cat("KEYWORD,PROBABILITY,N,d,idf\n", append = T,
158         file = outIdf)
159     for (i in 1:topKeywordCount){
160
161       idf<-log10(acumTo[j]/countKw[i, j])
162       cat("i j", i, " ", j, "idf= log(", acumTo[j], "/", countKw[i,j], ")\n")
163       Ttopic.keyword[i,j] <-
164         paste(topic.keyword[i,j], ",", mdl.post$terms[[j, topic.keyword[i,j]]], ","
              ,
165               acumTo[j], ",", countKw[i, j], ",", idf) ## going to work with a copy
                    of topic.keywords instead
166       cat(Ttopic.keyword[[i,j]],"\n", append = T,
167           file = outIdf)
168     }
169 }
170
171 cat("Done\n")
```

B.3 Program: utilsdb2.R

```
1  library(topicmodels)
2  ##-----------------------------------------------------------------------
3  ## This function reads the data from csv file, cleans it and returns tm text
       Corpus
4  ## readFromfileName - name of the file to read from
5  ##   year filter
6  ##   year - optional: year when docs were created, format = YYYY
7  ##-----------------------------------------------------------------------
8  createCorp <- function(readFromfileName, year){
9    cat("@ createCorp \n")
10   library(tm)
11   ## Load the data
12   Posts <- read.delim(file = readFromfileName, header = T, quote = "", sep = "\t"
         )
13
14   cat("Read", nrow(Posts), "rows from", readFromfileName, "\n")
15   cat("year to process =", year, "\n")
16
17   ## the following lines to filter by year
18   if(! missing(year)){
19     Posts$create_ts <- as.POSIXlt(Posts$create_ts)
20     Posts <- subset(Posts, Posts$create_ts$year == (year - 1900))
21     cat("Kept", nrow(Posts), "rows\n")
22   }
23
24   return( doCorpCreation(Posts) )
25 }
26
27 ## this is a private function for corpus creation
28 doCorpCreation <- function(Posts){
29
30   Posts <- data.frame(doc_id = Posts$id, question = Posts$title, txt = paste(
         Posts$title, Posts$body, Posts$answers, sep =" ")) ## added question
31   ## Build a corpus
32   posts.reader <- readTabular(mapping=list(content="txt", id="doc_id", quest="
         question")) ## added question
33
34   corp <- VCorpus(DataframeSource(Posts), readerControl=list(reader=posts.reader)
         )
35
36   ## Transform data
```

```
37    ## mc.cores = 1 to fix run time error
38
39    corp = tm_map(corp, content_transformer(tolower), mc.cores=1) #converting to
          lower case
40    corp = tm_map(corp, removeWords, stopwords('english'), mc.cores=1) # Remove
          Stopwords
41    corp = tm_map(corp, stemDocument, mc.cores=1) # Stemming
42    corp = tm_map(corp, stripWhitespace, mc.cores=1) # Eliminate whitespace char
43
44    cat ("Created corpus from", length(corp), "documents\n")
45    return (corp)
46  }
47
48  ## Removes topics from the Document Term Matrix using tf-idf approach
49  ## dtm - Document Term Matrix to process
50  ## threshold - remove words with tf-idf values smaller than threshold
51  ##     this parameter is optional: if threshold is not defined, we will remove
52  ##     most frequent words, based on the median value of tf-idf (at least 50%)
53  ##     Setting threshold = 0 will eliminate words appearing in every document
54  removeFrequentWords <- function(dtm, threshold){
55    library("slam")
56    termTfidf <- tapply(dtm$v/row_sums(dtm)[dtm$i], dtm$j, mean) *
57      log2(nDocs(dtm)/col_sums(dtm > 0))
58
59    # if no theshold is provided then set the thrshold value to median of tf-idf
60    # distribution, removing at least 50% of the words
61    if( missing(threshold) ) {
62      threshold <- median(termTfidf)
63    }
64
65    # remove terms which have tf-idf smaller than the median of all the tf-idf
          values
66    # this will shrink the dictionay by approximately 50%
67    dtm <- dtm[, termTfidf > threshold]
68    dtm <- dtm[row_sums(dtm) > 0,]  #remove docs that have no terms remaining (
          unlikely event)
69    return(dtm)
70  }
71
72  ## This function returns frequency of topics for a given LDA model
73  ## dat - Document Term Matrix to feed to the LDA model
74  ## topicCount - Number of topics in the LDA model
75  getTopicsFrequency <- function(dat, topicCount){
```

```
76    mdl <- LDA(dat, topicCount) #LDA model
77
78    mdl.alpha <- mdl@alpha
79    mdl.beta.mean <- mean(mdl@beta)
80    mdl.beta.sd <- sd(mdl@beta)
81
82    return( list(
83      mdl.alpha = mdl.alpha,
84      mdl.beta.mean = mdl.beta.mean,
85      mdl.beta.sd = mdl.beta.sd,
86      topic.frequency = as.vector(table(topics( mdl )))
87    ))
88  }
89
90  incCalc <- function(totCount) {
91    ## Function to calculate increments, 2..100, 1 101..200 5, 201..300 10 ...
          capped to 25 after totCount >= 600 to 25
92    numRanges <- ceiling(totCount/100) ## this is to ensure that if goes beyond
          100's it is taken into account
93    vecRanges = matrix(nrow = 1, ncol = numRanges)
94    vecCount <- c()
95    MRanges <- matrix(nrow = numRanges,ncol = 2)
96    ix <- 1
97
98    maxCount <- 1
99    cat("numRanges=", numRanges, "\n")
100   for(i in 1:numRanges){
101     indI <- (100*i)-100+1
102     indJ <- 100*i
103
104     MRanges[i,1] <- indI
105     MRanges[i,2] <- indJ
106
107     cat("i=", i, "from=", MRanges[i,1], "to ", MRanges[i,2])
108
109     cat ("indI=", indI, "indJ=", indJ, "i=", i, "\n")
110
111     p <- i
112
113     if (p > 1 && p < 7) {p <- 5*(i-1)} else {if (p!=1){p <-25}}
114
115     cat("p=",p,"\n")
116
```

```
117     for (j in seq(from=MRanges[i,1], to=MRanges[i,2], by=p)){
118
119       if ( j < totCount ) { ## This is to limit the calculation of intervals up
              to totCount
120         vecCount[ix] <- j
121         maxCount <- ix
122         cat("j =", j, "\n")
123         cat("vecCount[", ix, "]=", vecCount[ix], "\n")
124         ix <- ix + 1
125       }
126     }
127     cat("maxCount=", maxCount, "\n")
128   }
129   return(vecCount)
130 }
```

B.4 Program: do_lda_analysis_topics.R

```
1  ##-------------------------------------------------------------------------
2  ## The script reads posts in delimited format, and generates
3  ## distribution of LDA topics for a given month year,
4  ## saving the output in in_file_name + ".year-month.topic_frequency"
5  ## Usage: Rscript do_lda_analysis.R in_file_name year month
6  ##
7  ## This version adds a Semaphore in order to slowdown the process of writing in a
       parallel processing
8  ## includes refactoring changes (i.e. taking out semaphore code from for loop..)
9  ##-------------------------------------------------------------------------
10 library(tm)
11 library(foreach)
12 library(doParallel)
13
14 source("utils.R")
15
16 args <- commandArgs(trailingOnly = TRUE)
17 readFrom <- args[1]
18 year  <-  as.integer(args[2])
19 month <-  as.integer(args[3])
20
21 semaphoreFileName <- paste("semaf_", Sys.getpid(), sep='') # append R's process
       id to semaphore to avoid conflict
22 if( file.exists(semaphoreFileName) ){ file.remove(semaphoreFileName) }
23
24 saveTo <- paste(readFrom, ".", year, "-", month, ".topic_frequency", sep ="")
25
26 corp <- createCorp(readFrom, year, month)
27
28 ## Build a Document-Term Matrix
29 dtm <- DocumentTermMatrix(corp, control = list(minWordLength = 2)) #keep words of
       lenght 2 or longer
30 cat("Before tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n")
31 dtm <- removeFrequentWords(dtm) #removing based on median tf-idf value
32 cat("After tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n")
33 ## the following line commented out to avoid sparse error
34 dtm <- removeSparseTerms(dtm, 1 - (1.1/nrow(dtm)) )  #remove terms appearing only
       in 1 document
35 dtm <- dtm[row_sums(dtm) > 0,]  #remove docs that have no terms remaining (
       unlikely event)
36 cat("After removing terms appearing only in 1 document: term count =", ncol(dtm),
```

```
      ", doc count =", nrow(dtm), "\n")
37
38 #setup parallel backend to use 8 processors
39 cl<-makeCluster(8)
40 registerDoParallel(cl)
41
42 cat("topicCount\tmdl.alpha\tmdl.beta.mean\tmdl.beta.sd\ttime (sec)\ttopic.
       frequency\n", sep="\t", append = T
43      , file = saveTo) # to file
44
45 foreach(topicCount = 2:nrow(dtm) #max = 1 topic per document
46         , .packages='topicmodels' #include package
47 ) %dopar% { #change to %do% for sequential execution
48   startRun <- Sys.time()
49
50   val <- getTopicsFrequency(dtm, topicCount)
51
52   prefix <- paste(topicCount, val$mdl.alpha, val$mdl.beta.mean, val$mdl.beta.sd,
       difftime(Sys.time(), startRun, units = "secs"), sep="\t")
53
54   cat("TopicCount:", topicCount, "\n") #to screen (no screen output is parallel
       mode)
55
56   while (file.exists(semaphoreFileName)==TRUE) {
57         Sys.sleep(1); cat("in while sleeping 1 sec\n");
58   }
59   file.create(semaphoreFileName) ## lock file
60   for (i in 1:length(val$topic.frequency)){
61        #cat("in cat for",i,Sys.time(), file.exists(semaphoreFileName), "\n")
62        cat(prefix, val$topic.frequency[i]
63              , "\n", sep="\t", append = T
64              , file = saveTo) # to file
65   }
66   file.remove(semaphoreFileName) ## unlock file
67   #cat("Semaphore is",file.exists(semaphoreFileName), "\n")
68  }
69 cat("Saved data to", saveTo, "\n")
70 cat("Done\n")
```

B.5 Program: do_lda_quarter_inc.R

```
1  ##-------------------------------------------------------------------
2  ## The script reads posts in delimited format, and generates
3  ## distribution of LDA topics for a given quarter-year,
4  ## saving the output in in_file_name + "." + year "-q" + quarter ".topic_
       frequency"
5  ## Usage: Rscript do_lda_quarter_inc.R in_file_name year quarter
6  ##-------------------------------------------------------------------
7  library(tm)
8  library(foreach)
9  library(doParallel)
10
11 source("utils.R")
12
13 args <- commandArgs(trailingOnly = TRUE)
14 readFrom <- args[1]
15 year  <-  as.integer(args[2])
16 quarter <-  as.integer(args[3])
17
18 semaphoreFileName <- paste("semaf_", Sys.getpid(), sep='') # append R's process
       id to semaphore to avoid conflict
19 if( file.exists(semaphoreFileName) ){ file.remove(semaphoreFileName) }
20
21 saveTo <- paste(readFrom, ".", year, "-q", quarter, ".topic_frequency", sep ="")
22
23 corp <- createCorpQuarter(readFrom, year, quarter)
24
25 ## Build a Document-Term Matrix
26 dtm <- DocumentTermMatrix(corp, control = list(minWordLength = 2)) #keep words of
       lenght 2 or longer
27 cat("Before tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n")
28 dtm <- removeFrequentWords(dtm) #removing based on median tf-idf value
29 cat("After tf-idf: term count =", ncol(dtm), ", doc count =", nrow(dtm), "\n")
30 ## the following line commented out to avoid sparse error
31 dtm <- removeSparseTerms(dtm, 1 - (1.1/nrow(dtm)) )  #remove terms appearing only
       in 1 document
32 dtm <- dtm[row_sums(dtm) > 0,]  #remove docs that have no terms remaining (
       unlikely event)
33 cat("After removing terms appearing only in 1 document: term count =", ncol(dtm),
       ", doc count =", nrow(dtm), "\n")
34
35 #setup parallel backend to use 8 processors
```

```
36 cl<-makeCluster(8)
37 registerDoParallel(cl)
38
39 cat("topicCount\tmdl.alpha\tmdl.beta.mean\tmdl.beta.sd\ttime (sec)\ttopic.
       frequency\n", sep="\t", append = T
40       , file = saveTo) # to file
41
42 vecCount <- incCalc(nrow(dtm))
43
44 ## foreach(topicCount = 2:nrow(dtm) #max = 1 topic per document
45 cat("lenght(vecCount)=", length(vecCount), "\n")
46 foreach(ix = 2:length(vecCount)
47           , .packages='topicmodels' #include package
48 ) %do% { #change to %dopar% for multiprocess execution
49
50   topicCount <- vecCount[ix]
51
52   startRun <- Sys.time()
53
54   val <- getTopicsFrequency(dtm, topicCount)
55
56   prefix <- paste(topicCount, val$mdl.alpha, val$mdl.beta.mean, val$mdl.beta.sd,
         difftime(Sys.time(), startRun, units = "secs"), sep="\t")
57
58   cat("TopicCount:", topicCount, "\n") #to screen (no screen output is parallel
         mode)
59
60   while (file.exists(semaphoreFileName)==TRUE) {
61     Sys.sleep(1); cat("in while sleeping 1 sec\n");
62   }
63   file.create(semaphoreFileName) ## lock file
64   for (i in 1:length(val$topic.frequency)){
65     cat(prefix, val$topic.frequency[i]
66           , "\n", sep="\t", append = T
67           , file = saveTo) # to file
68   }
69   file.remove(semaphoreFileName)
70 }
71 cat("Saved data to", saveTo, "\n")
72 cat("Done\n")
```

B.6 Program: utils.R

```
1  library(topicmodels)
2  ##------------------------------------------------------------------------
3  ## The function takes term document matrix (sparse representation), tdm, and
       exports it to file
4  ## fileName in the following format "distinctWordCount wordId:WordCount ...", one
       document per
5  ## line. For example, "2 15:7 21:4" means that a document contains two distinct
       words
6  ## with ids 15 and 21, appearing 7 and 4 times, respectively.
7  ## This format is used by LDA and HDP-LDA code provided by Blei's team.
8  ##
9  ## The function saves dictionary of words, one word per line to file fileName,
       suffixed by ".dic".
10 ## Id of a given word is given by word's line number in the file fileName.dic;
11 ## line number count starts from 1.
12 ##
13 ## The function saves document "names", one name per line to file fileName,
       suffixed by ".doc".
14 ## Id of a given document is equivalent to the document's line number in the file
       fileName.doc;
15 ## line number count starts from 1.
16 ##
17 ## Usage example: export2lda_c(myTermDocMatrix, "~/foo.txt")
18 ## Note that the existing file will be overwritten
19 ##------------------------------------------------------------------------
20 export2lda_c <- function(tdm, fileName){
21   # We assume that
22   # j - term (a.k.a. word) ids
23   # i - doc ids
24   # v - word frequency
25
26   fid <- file(fileName, "w") # overwrite existing file
27   docIds <- unique(tdm$i)
28   for(docId in docIds){
29     # I am not certain if $j is guaranteed to be consecutive;
30     # hence the inefficient search using which
31     wordIndexes <- which( tdm$i == docId)
32     cat(length(wordIndexes), file = fid, sep = "", append = T)
33     for(wordIndex in  wordIndexes){
34       cat(" ", tdm$j[wordIndex], ":", tdm$v[wordIndex], file = fid, sep = "",
             append = T)
```

```
35      }
36      cat("\n", file = fid, sep = "", append = T)
37    }
38    close(fid)
39
40    # save dictionary of words
41    write(tdm$dimnames$Terms, file = paste(fileName, ".dic", sep=""), sep = "\n")
42
43    # save original document names
44    write(tdm$dimnames$Docs, file = paste(fileName, ".doc", sep=""), sep = "\n")
45  }
46
47  ## This function reads the data from csv file, cleans it and returns tm text
        Corpus
48  ## readFromfileName - name of the file to read from
49  ## Quarterly Filter (mandatory):
50  ##   year  - mandatory: year when docs were created, format = YYYY
51  ##   month - mandatory: month when docs were created
52  createCorpQuarter <- function(readFromfileName, year, quarter){
53    library(tm)
54    ## Load the data
55    Posts <- read.delim(file = readFromfileName, header = T, quote = "", sep = "\t"
          )
56
57    cat("Read", nrow(Posts), "rows from", readFromfileName, "\n")
58
59    if(! missing(year) & ! missing(quarter)){
60      Posts$create_ts <- as.POSIXlt(Posts$create_ts)
61      Posts <- subset(Posts, quarters(Posts$create_ts) == paste("Q", quarter, sep =
            "")
62                      & Posts$create_ts$year == (year - 1900))
63      cat("Kept", nrow(Posts), "rows\n")
64    }else{
65      stop("Provide year and quarter")
66    }
67
68    return( doCorpCreation(Posts) )
69  }
70
71
72  ## This function reads the data from csv file, cleans it and returns tm text
        Corpus
73  ## readFromfileName - name of the file to read from
```

```
74 ## Monthly Filter (optional):
75 ##   year  - optional: year when docs were created, format = YYYY
76 ##   month - optional: month when docs were created
77 createCorp <- function(readFromfileName, year, month){
78   library(tm)
79   ## Load the data
80   Posts <- read.delim(file = readFromfileName, header = T, quote = "", sep = "\t"
       )
81
82   cat("Read", nrow(Posts), "rows from", readFromfileName, "\n")
83
84   if(! missing(year) & ! missing(month)){
85     Posts$create_ts <- as.POSIXlt(Posts$create_ts)
86     Posts <- subset(Posts, Posts$create_ts$mon == (month - 1)
87             & Posts$create_ts$year == (year - 1900))
88     cat("Kept", nrow(Posts), "rows\n")
89   }
90
91   return( doCorpCreation(Posts) )
92 }
93
94 ## this is a private function for corpus creation
95 doCorpCreation <- function(Posts){
96   # Concatenate columns, otherwise DataframeSource gets confused
97   Posts <- data.frame(paste(Posts$title, Posts$body, Posts$answers, sep =" "))
98
99   ## Build a corpus
100  corp <- Corpus(DataframeSource(Posts))
101
102  ## Transform data
103  corp = tm_map(corp, content_transformer(tolower)) #converting to lower case
104  corp = tm_map(corp, removeWords, stopwords('english')) # Remove Stopwords
105  corp = tm_map(corp, stemDocument) # Stemming
106  corp = tm_map(corp, stripWhitespace) # Eliminate whitespace char
107
108  cat ("Created corpus from", length(corp), "documents\n")
109  return (corp)
110 }
111
112 ## Removes topics from the Document Term Matrix using tf-idf approach
113 ## dtm - Document Term Matrix to process
114 ## threshold - remove words with tf-idf values smaller than threshold
115 ##    this parameter is optional: if threshold is not defined, we will remove
```

```
116 ##     most frequent words, based on the median value of tf-idf (at least 50%)
117 ##     Setting threshold = 0 will eliminate words appearing in every document
118 removeFrequentWords <- function(dtm, threshold){
119   library("slam")
120   termTfidf <- tapply(dtm$v/row_sums(dtm)[dtm$i], dtm$j, mean) *
121     log2(nDocs(dtm)/col_sums(dtm > 0))
122
123   # if no theshold is provided then set the thrshold value to median of tf-idf
124   # distribution, removing at least 50% of the words
125   if( missing(threshold) ) {
126     threshold <- median(termTfidf)
127   }
128
129   # remove terms which have tf-idf smaller than the median of all the tf-idf
        values
130   # this will shrink the dictionay by approximately 50%
131   dtm <- dtm[, termTfidf > threshold]
132   dtm <- dtm[row_sums(dtm) > 0,]  #remove docs that have no terms remaining (
        unlikely event)
133   return(dtm)
134 }
135
136 ## This function returns frequency of topics for a given LDA model
137 ## dat - Document Term Matrix to feed to the LDA model
138 ## topicCount - Number of topics in the LDA model
139 getTopicsFrequency <- function(dat, topicCount){
140   mdl <- LDA(dat, topicCount) #LDA model
141
142   mdl.alpha <- mdl@alpha
143   mdl.beta.mean <- mean(mdl@beta)
144   mdl.beta.sd <- sd(mdl@beta)
145
146   return( list(
147     mdl.alpha = mdl.alpha,
148     mdl.beta.mean = mdl.beta.mean,
149     mdl.beta.sd = mdl.beta.sd,
150     topic.frequency = as.vector(table(topics( mdl )))
151     ))
152 }
153 incCalc <- function(totCount) {
154   ## Function to calculate increments, 2..100, 1 101..200 5, 201..300 10 ...
        capped to 25 after totCount >= 600 to 25
155   numRanges <- ceiling(totCount/100) ## this is to ensure that if goes beyond
```

```
        100's it is taken into account
156   vecRanges = matrix(nrow = 1, ncol = numRanges)
157   vecCount <- c()
158   MRanges <- matrix(nrow = numRanges,ncol = 2)
159   ix <- 1
160
161   maxCount <- 1
162   cat("numRanges=", numRanges, "\n")
163   for(i in 1:numRanges){
164     indI <- (100*i)-100+1
165     indJ <- 100*i
166
167     MRanges[i,1] <- indI
168     MRanges[i,2] <- indJ
169
170     cat("i=", i, "from=", MRanges[i,1], "to ", MRanges[i,2])
171
172     cat ("indI=", indI, "indJ=", indJ, "i=", i, "\n")
173
174     p <- i
175
176     if (p > 1 && p < 7) {p <- 5*(i-1)} else {if (p!=1){p <-25}}
177
178     cat("p=",p,"\n")
179
180     for (j in seq(from=MRanges[i,1], to=MRanges[i,2], by=p)){
181
182       if ( j < totCount ) { ## This is to limit the calculation of intervals up
              to totCount
183           vecCount[ix] <- j
184           maxCount <- ix
185           cat("j =", j, "\n")
186           cat("vecCount[", ix, "]=", vecCount[ix], "\n")
187           ix <- ix + 1
188       }
189     }
190     cat("maxCount=", maxCount, "\n")
191   }
192   return(vecCount)
193 }
```

B.7 Program: convVar_k200.R

```
1  ##-------------------------------------------------------------------------
2  ### This program reads the *.topic.frequency files for each dataset gets the top
       frequency for x=2..50, and merges them up together
3  ### for them to be analyzed later with a regression model
4  ##-------------------------------------------------------------------------
5  readFile <- function(fName) {
6    dat <- read.delim(file = fName, header = T, fill=T, sep = "\t",
7                      row.names=NULL)
8    colnames(dat)<-c(colnames(dat)[-1],"x")
9    dat$x<-NULL
10   dat<-as.data.frame(dat)
11   return(dat)
12 }
13
14 processFile <- function(dat){
15   topicCountList <- sort(unique(strtoi(dat$topicCount))) ## making sure the list
       is ordered
16
17   timeframe_type = vector()
18   timeframe = vector()
19   dataset_name = vector()
20   topicCount = vector()
21   postFraction = vector()
22   topXX = vector()
23   documentCount = vector()
24
25   topX <- data.frame(timeframe_type = vector(), timeframe = vector(), dataset_
       name = vector(), topicCount = vector(), postFraction = vector(), topXX =
       vector(), documentCount = vector())
26
27   for(x in 2:50){#top X topics
28     for( t in topicCountList ){ ## strtoi(topicCount)
29       if (t < x){ ## to avoid writing NA's
30         next
31       }
32       else
33       {
34         d <- dat [which(dat$topicCount == t), ]
35         freq <- sort(d$topic.frequency, decreasing = T)
36         documentCount <- sum (freq) #number of docs in corpus
37         postFraction = sum(freq[1:x]) / documentCount # this is an F value for a
```

```
            given top X
38          if (is.na(postFraction)) {
39            cat("========== postfraction is NA program skips
                  interaction t=", t, "==========\n")
40            cat("x=", x, " documentCount=", documentCount, "\n")
41            next
42          }
43        topX <- rbind(topX, data.frame(timeframe_type = Xtimeframe_type[i_dsName
              ], timeframe = Xtimeframe[i_listf], dataset_name = Xdataset_name[i_
              dsName], topicCount = t, postFraction = postFraction, topXX = x,
              documentCount = documentCount))
44      }
45    }
46  }
47  ## writes output file everytime finishes processing one frequency file
48
49  fName <- paste(outPath, "topX", ".csv", sep = "")
50
51  if(firstime){
52    cat("** WRITING OUTPUT FILE FIRSTIME**", firstime, "\n")
53    cat(dir(outPath),"\n")
54    ## cat("FILE EXIST NOT NEGATED:", fName, " is ", file.exists(fName), "\n")
55    ## cat("FILE EXIST NEGATED:", fName, " is ", !file.exists(fName), "\n")
56    write.table(topX, file = fName, sep = ",", row.names=FALSE, col.names=TRUE,
          append = T)
57    firstime <<- FALSE
58    print(file.info(fName))
59  } else {
60    cat("** WRITING OUTPUT FILE SUCCESIVE **", firstime, "\n")
61    write.table(topX, file = fName, sep = ",", row.names=FALSE, col.names=FALSE,
          append = T)
62    print(file.info(fName))
63  }
64 } ## EOF
65
66
67 ######################################################################
68 #####                          main                              #####
69 ######################################################################
70
71 XSystime <<- format(Sys.time(), "%a-%b-%d_%H-%M-%S_%Y")
72 dsName <- c("salesforce-m-", "dba-m-", "dba-q-", "android-m-") ## to handle the
       ds names
```

```
73
74  ### test dsName <- c("test-q-") ## to handle the ds names
75
76  Xtimeframe_type <<- c("monthly", "monthly", "quarterly", "monthly")
77  Xdataset_name <<- c("salesforce", "dba", "dba", "android")
78
79  FKdir <- c()
80  outListf <- c() ### this is the output directory of the files to be plotted, may
        not be used
81  ffDir <- c()
82  i_dsName <<- 1
83  firstime <<- TRUE ## this flag was placed after first, for so it wrote the hdrs
        each time ds changed!
84  for (i_dsName in 1:length(dsName)) {
85    cat("DEBUG: Processing dataset=", dsName[i_dsName], " at:", XSystime, "\n")
86
87    ## former FKdir[i_dsName] <- file.path("~", "Thesis", "Fitting", paste(dsName[i
          _dsName],"FK", sep=""), "files-FK")
88    ## former ffDir[i_dsName] <- file.path("~", "Thesis", "Fitting", paste(dsName[i
          _dsName],"FK", sep=""), "files-frequencies")
89
90    FKdir[i_dsName] <- file.path("/media", "data", "thesis", "Thesis", "Fitting",
          paste(dsName[i_dsName],"FK", sep=""), "files-FK")
91    ffDir[i_dsName] <- file.path("/media", "data", "thesis", "Thesis", "Fitting",
          paste(dsName[i_dsName],"FK", sep=""), "files-frequencies")
92
93
94    setwd(ffDir[i_dsName])
95    cat("DEBUG: ffDir[", i_dsName, "]=", ffDir[i_dsName], "\n")
96
97    listf <- list.files(pattern="Posts.xml.csv")
98
99    ## former outPath <<- file.path("~", "Thesis", "Fitting","/")
100   outPath <<- file.path("/media", "data", "thesis", "Thesis", "Fitting","/")
101   ##} ## ends here for debugging purposes next code deactivated:
102   ##  for (i_dsName in 1:length(dsName)) {
103
104   for (i_listf in 1:length(listf)) {
105     readFromfileName <- listf[i_listf]
106     cat("listf[", i_listf, "]=", listf[i_listf],"\n")
107     Xtimeframe <<- vector()
108     Xtimeframe[i_listf] <- substr(gsub("\\.","-",readFromfileName),15,21)
109     readFromfileName <- gsub(" ", "", readFromfileName, fixed = TRUE)
```

```
110       cat("DEBUG: Reading=", readFromfileName, " Xtimeframe=", Xtimeframe[i_listf],
              "\n")
111       dat <- readFile(readFromfileName)
112                   processFile(dat)
113     }
114   }
115   cat("** END OF EXECUTION **\n")
```

B.8 Program: verifyFit_k200.R

```
1  ##------------------------------------------------------------------------
2  ## This program iterates over the specified datasets, and computes the RMSE of
       the approximation formulas
3  ##------------------------------------------------------------------------
4
5  #set to True if you want to enable filtering of K > 200, else set to False
6  data_filter_200 <- T
7
8  readFromfileName = "topX.csv.original.zip"
9
10 if(data_filter_200){
11   model_file_name <- "./models/models_fitting.remove_top_25_percent_and_values_gt
         _200.rda"
12   models_performance_file_name <- "./models/models_performance.remove_top_25_
         percent_and_values_gt_200.csv"
13 }else{
14   model_file_name <- "./models/models_fitting.remove_top_25_percent.rda"
15   models_performance_file_name <- "./models/models_performance.remove_top_25_
         percent.csv"
16 }
17
18 load(model_file_name)
19 ##############################################################################
20 #####                 define approximation formulas                      #####
21 ##############################################################################
22
23 # The "flexible" model a(X, N) * K^b(X,N)
24 # K == topic count
25 # X == the top-X
26 # N == number of documents
27 model_two_param_simple <- function(K, X, N){
28
29   val <- data.frame(X = X, N = N)
30   a <- predict(dat.lm.intercept, val) # 1.171 + 0.649 * log(X) - 0.116 * log(N)
31   b <- predict(dat.lm.slope, val)  # -0.841 + 0.002 * X + 0.00003 * N
32
33   exp(a  + b * log(K))
34 }
35
36 model_two_param_complex <- function(K, X, N){
37
```

```
38    val <- data.frame(X = X, N = N)
39    a <- predict(dat.lm.intercept.complex, val)
40    b <- predict(dat.lm.slope.complex, val)
41
42    exp(a  + b * log(K))
43  }
44
45  # The "constrained" model X^-b(X, N) * K^b(X,N)
46  # K == topic count
47  # X == the top-X
48  # N == number of documents
49
50  model_one_param_simple <- function(K, X, N){
51    val <- data.frame(X = X, N = N)
52    b <- predict(dat.lm.b, val)  #-8.408351E-01 + 2.110667E-03 * X + 3.397901E-05 *
          N
53
54    X^-b * K^b
55  }
56
57  model_one_param_complex <- function(K, X, N){
58    val <- data.frame(X = X, N = N)
59    b <- predict(dat.lm.b.complex, val)
60
61    X^-b * K^b
62  }
63
64
65  ##############################################################################
66  #####                          helpers                                   #####
67  ##############################################################################
68
69  # root-mean-square error
70  rmse <- function(actual, expected){
71    sqrt(mean((expected-actual)^2))
72  }
73
74  ##############################################################################
75  #####                          main                                      #####
76  ##############################################################################
77
78  dat <- read.csv(unz(readFromfileName, "topX.csv.original"), header = T, sep = ","
        , row.names = NULL) ## NOT EXCEL
```

```
79  validGroups <- unique(paste(dat$timeframe_type, dat$timeframe, dat$dataset_name,
        sep = " "))
80
81  dat.stats <- data.frame()
82  for(iG in 1:length(validGroups)) {
83    s <- strsplit(validGroups[iG], " ")
84    sdf <- as.data.frame(s)
85    p1 <- as.character(sdf[1,]) ## timeframe_type
86    p2 <- as.character(sdf[2,]) ## timeframe
87    p3 <- as.character(sdf[3,]) ## dataset_name
88
89    cat("p1=", p1, " p2=", p2, " p3=", p3, " iG=", iG, "\n")
90    for(itopXX in 2:50) {
91      ds <- dat[dat$timeframe_type ==  p1 & dat$timeframe == p2 & dat$dataset_name
            == p3 & dat$topXX == itopXX,]
92      #cat("p1=", p1, " p2=", p2, " p3=", p3, " itopXX=", itopXX, " iG=", iG, "\n")
93
94      if(data_filter_200){ #if True -- keep only values of K <= 200
95        ds <- ds[ds$topicCount <= 200, ]
96      }
97      ds <- ds[ds$topicCount < 0.75 * max(ds$documentCount), ]
98
99
100     #fit indvidual flex model
101     ds.lm.flex <- lm(log(postFraction) ~ log(topicCount), data = ds )
102
103     #fit indvidual constrained model
104     ds.lm.constr <- nls(ds$postFraction ~ ds$topXX^(-b) * ds$topicCount^b, data =
            ds, start = list( b = -1))
105
106
107     F.ds.lm.flex <- exp(predict(ds.lm.flex)) #keep in mind that we are operating
            on log-transform data here, need to convert it back
108     F.ds.lm.constr <- predict(ds.lm.constr)
109
110     F.model_two_param_simple <- model_two_param_simple(ds$topicCount, ds$topXX
            [1], ds$documentCount[1])
111     F.model_one_param_simple <- model_one_param_simple(ds$topicCount, ds$topXX
            [1], ds$documentCount[1])
112     F.model_two_param_complex <- model_two_param_complex(ds$topicCount, ds$topXX
            [1], ds$documentCount[1])
113     F.model_one_param_complex <- model_one_param_complex(ds$topicCount, ds$topXX
            [1], ds$documentCount[1])
```

```
114
115     #compute rmse and save stats
116     dat.stats <- rbind(dat.stats,
117       data.frame(
118         dt_partition = p1,
119         time_frame = p2,
120         dataset_name = p3,
121         X = ds$topXX[1],
122         N = ds$documentCount[1],
123         rmse_flex_model_simple = rmse(ds$postFraction, F.model_two_param_simple),
124         rmse_constraint_model_simple = rmse(ds$postFraction, F.model_one_param_
              simple),
125         rmse_flex_model_complex = rmse(ds$postFraction, F.model_two_param_complex
              ),
126         rmse_constraint_model_complex = rmse(ds$postFraction, F.model_one_param_
              complex),
127         rmse_tailored_flex_lm = rmse(ds$postFraction, F.ds.lm.flex),
128         rmse_tailored_constr_lm = rmse(ds$postFraction, F.ds.lm.constr)
129       )
130     )
131   }
132 }
133
134 write.csv(dat.stats, models_performance_file_name, row.names = FALSE)
```

B.9 Program: verifyFit_k200_analysis.R

```
#-------------------------------------------------------------------------
#set to True if you want to enable filtering of K > 200, else set to False
#plots the RMSE boxplots per model and per dataset name
#-------------------------------------------------------------------------
data_filter_200 <- F

if(data_filter_200){
  model_file_name <- "./models/models_fitting.remove_top_25_percent_and_values_gt_200.rda"
  models_performance_file_name <- "./models/models_performance.remove_top_25_percent_and_values_gt_200.csv"
  rmse_per_model_file_name <- "./models/models_performance__remove_top_25_percent_and_values_gt_200.pdf"
  rmse_per_dataset_file_name <- "./models/models_performance__per_dataset__remove_top_25_percent_and_values_gt_200.pdf"
  rmse_per_x_file_name <- "./models/models_performance__per_x__remove_top_25_percent_and_values_gt_200.pdf"
  rmse_per_n_file_name <- "./models/models_performance__per_n__remove_top_25_percent_and_values_gt_200.pdf"
}else{
  model_file_name <- "./models/models_fitting.remove_top_25_percent.rda"
  models_performance_file_name <- "./models/models_performance.remove_top_25_percent.csv"
  rmse_per_model_file_name <- "./models/models_performance__remove_top_25_percent.pdf"
  rmse_per_dataset_file_name <- "./models/models_performance__per_dataset__remove_top_25_percent.pdf"
  rmse_per_x_file_name <- "./models/models_performance__per_x__remove_top_25_percent.pdf"
  rmse_per_n_file_name <- "./models/models_performance__per_n__remove_top_25_percent.pdf"

}


dat.stats <- read.csv(models_performance_file_name)

#let's visualize performance of the models
library(tidyr)
dat.stats.long <- gather(dat.stats, model_name, rmse, rmse_flex_model_simple:rmse_tailored_constr_lm, factor_key=TRUE)
```

```
30
31 #reorder for box-plot
32 dat.stats.long$model_name <- factor(dat.stats.long$model_name, levels=c("rmse_
       flex_model_simple", "rmse_flex_model_complex", "rmse_constraint_model_simple"
       , "rmse_constraint_model_complex", "rmse_tailored_flex_lm", "rmse_tailored_
       constr_lm" ))
33
34 pdf(rmse_per_model_file_name)
35 boxplot(rmse ~ model_name,
36          data = dat.stats.long,
37          log = "y",
38          #xlab = "Model Name",
39          las = 2,
40          ylab = "RMSE",
41          par(mar = c(12, 5, 4, 2)+ 0.1),
42          names = c("Flex - simple", "Flex - complex", "Constr. - simple", "Constr.
                - complex", "Ind. fit - flex", "Ind. fit - constr")
43 )
44 grid()
45 dev.off()
46
47 pdf(rmse_per_dataset_file_name)
48 dat.stats$ds_merged_name <- as.factor(paste(dat.stats$dataset_name, "-", dat.
       stats$dt_partition))
49 boxplot(rmse_constraint_model_complex ~ ds_merged_name,
50          data = dat.stats,
51          log = "y",
52          #xlab = "Model Name",
53          las = 2,
54          ylab = "RMSE",
55          par(mar = c(12, 5, 4, 2)+ 0.1)
56          #names = c("Flex - simple", "Flex - complex", "Constr. - simple", "Constr
                . - complex", "Ind. fit - flex", "Ind. fit - constr")
57 )
58 grid()
59 dev.off()
60
61 pdf(rmse_per_dataset_file_name)
62 dat.stats$ds_merged_name <- as.factor(paste(dat.stats$dataset_name, "-", dat.
       stats$dt_partition))
63 boxplot(rmse_constraint_model_complex ~ ds_merged_name,
64          data = dat.stats,
65          log = "y",
```

```
66          #xlab = "Model Name",
67          main = "Model - constrained, complex",
68          las = 2,
69          ylab = "RMSE",
70          par(mar = c(12, 5, 4, 2)+ 0.1)
71          #names = c("Flex - simple", "Flex - complex", "Constr. - simple", "Constr
                . - complex", "Ind. fit - flex", "Ind. fit - constr")
72  )
73  grid()
74  dev.off()
75
76  pdf(rmse_per_x_file_name)
77  boxplot(rmse_constraint_model_complex ~ X,
78          data = dat.stats,
79          log = "y",
80          xlab = "X",
81          main = "Model - constrained, complex",
82          #las = 2,
83          ylab = "RMSE"
84          #par(mar = c(12, 5, 4, 2)+ 0.1)
85          #names = c("Flex - simple", "Flex - complex", "Constr. - simple", "Constr
                . - complex", "Ind. fit - flex", "Ind. fit - constr")
86  )
87  grid()
88  dev.off()
89
90  pdf(rmse_per_n_file_name)
91  boxplot(rmse_constraint_model_complex ~ N,
92          data = dat.stats,
93          log = "y",
94          xlab = "N",
95          main = "Model - constrained, complex",
96          #las = 2,
97          ylab = "RMSE"
98          #par(mar = c(12, 5, 4, 2)+ 0.1)
99          #names = c("Flex - simple", "Flex - complex", "Constr. - simple", "Constr
                . - complex", "Ind. fit - flex", "Ind. fit - constr")
100 )
101 grid()
102 dev.off()
103
104
105 #summary stats
```

```
106 library(stargazer)
107 stargazer(dat.stats[,6:11])
```

B.10 Program: validate_fit.R

```
#-------------------------------------------------------------------------
# set to True if you want to enable filtering of K > 200, else set to False
# performs validation using 10 folds
#-------------------------------------------------------------------------
data_filter_200 <- F

# root-mean-square error
rmse <- function(actual, expected){
  sqrt(mean((expected-actual)^2))
}

# one param complex prediction model
model_one_param_complex <- function(K, X, N){
  val <- data.frame(X = X, N = N)
  b <- predict(dat.lm.b.complex, val)

  X^-b * K^b
}


################################################################################
#####                    get raw data                                      #####
################################################################################

readFromfileName = "topX.csv.original.zip"
dat.raw <- read.csv(unz(readFromfileName, "topX.csv.original"), header = T, sep =
    ",", row.names = NULL)

#add key column
dat.raw$data_subset_name <- paste(dat.raw$timeframe_type, dat.raw$timeframe, dat.
    raw$dataset_name)

################################################################################
#####                    get the individual fit of b-values                #####
################################################################################

if(data_filter_200){
  to_fit_file_name <- "to_fit.remove_top_25_percent_and_values_gt_200.csv"
}else{
  to_fit_file_name <- "to_fit.remove_top_25_percent.csv"
}
```

```
40 dat <- read.csv(to_fit_file_name)
41
42 #add key column
43 dat$data_subset_name <- paste(dat$time_interval, dat$time_frame, dat$dataset_name
       )
44
45
46
47 ##############################################################################
48 #####                    do run 10-fold validation                       #####
49 ##############################################################################
50
51 #split 96 datasets into 10 folds
52 unique_data_subsets <- unique(dat$data_subset_name)
53 require(caret)
54 folds <- createFolds(c(1:length(unique_data_subsets)), k = 10, list = TRUE,
       returnTrain = FALSE)
55
56 rmse_per_fold <- c()
57 for(f in 1:10){
58   train_subsets <- unique_data_subsets[-folds[[f]]]
59   test_subsets  <- unique_data_subsets[ folds[[f]]]
60
61
62   #let's select the b values associated with the data subsets in the train set
63   train_data <- dat[ dat$data_subset_name %in% train_subsets, ]
64
65   #complex coanstrained model: compute the model for the value of b
66   dat.lm.b.complex <- lm(b ~ X + N + log(X) , data = train_data)
67
68   #let us see how the model for b value fits the raw data  for the test set
69   dat.raw.test <- dat.raw[dat.raw$data_subset_name %in% test_subsets, ]
70
71   #apply DS1 or DS2 filter
72   if(data_filter_200){ #if True -- keep only values of K<= 200
73     dat.raw.test <- dat.raw.test[dat.raw.test$topicCount <= 200, ]
74   }
75   dat.raw.test <- dat.raw.test[dat.raw.test$topicCount < 0.75 * dat.raw.test$
       documentCount, ]
76
77
78   fitted_values <- model_one_param_complex(dat.raw.test$topicCount, dat.raw.test$
       topXX, dat.raw.test$documentCount)
```

```
79    #compute and save rmse per fold
80    rmse_per_fold <- c(rmse_per_fold, rmse(dat.raw.test$postFraction, fitted_values
          ))
81  }
82
83
84  ##############################################################################
85  #####                    show summary stats                                  #####
86  ##############################################################################
87
88  summary(rmse_per_fold)
89
90
91  if(data_filter_200){
92    figure_file_name <- "./figures/one_param_complex_10-fold_validation_DS2.pdf"
93    rmse.df.ds2 <- data.frame(fiter_name = "Dataset 2", rmse = rmse_per_fold)
94  }else{
95    figure_file_name <- "./figures/one_param_complex_10-fold_validation_DS1.pdf"
96    rmse.df.ds1 <- data.frame(fiter_name = "Dataset 1", rmse = rmse_per_fold)
97  }
98
99  pdf(figure_file_name)
100 boxplot(rmse_per_fold,
101         ylab = "RMSE",
102         xlab = "Constr. - complex"
103         )
104 grid()
105 dev.off()
106
107 #need to run the script twice with data_filter_200 = T and data_filter_200 = F to
        get the figure below correctly
108 pdf("./figures/one_param_complex_10-fold_validation_DS_both.pdf")
109 rmse.df.ds_both <- rbind(rmse.df.ds1, rmse.df.ds2)
110 plot(rmse.df.ds_both$fiter_name, rmse.df.ds_both$rmse,
111         ylab = "RMSE",
112         xlab = "Constr. - complex"
113 )
114 grid()
115 dev.off()
```

References

[1] Ayushi Aggarwal, Gajendra Waghmare, and Ashish Sureka. Mining issue tracking systems using topic models for trend analysis, corpus exploration, and understanding evolution. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, RAISE 2014, pages 52–58, New York, NY, USA, 2014. ACM.

[2] Edoardo M Airoldi, Elena A Erosheva, Stephen E Fienberg, Cyrille Joutard, Tanzy Love, and Suyash Shringarpure. Reconceptualizing the classification of PNAS articles. *Proc. Natl. Acad. Sci. U. S. A.*, 107(49):20899–20904, 7 December 2010.

[3] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[4] archive.org. Internet archive. `https://archive.org/download/stackexchange`. Accessed: 2015-9-NA.

[5] Andrea Arcuri and Gordon Fraser. On parameter tuning in search based software engineering. In *Search Based Software Engineering*, pages 33–47. Springer, Berlin, Heidelberg, 10 September 2011.

[6] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE, 2012.

[7] Rajkumar Arun, Vommina Suresh, C E Veni Madhavan, and M N Narasimha Murthy. On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Advances in Knowledge Discovery and Data Mining*, pages 391–402. Springer, Berlin, Heidelberg, 21 June 2010.

[8] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. *Uncertainty in Articial Intelligence*, 2009.

[9] Hazeline U Asuncion, Arthur U Asuncion, and Richard N Taylor. Software traceability with topic modeling. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, pages 95–104. ACM, 1 May 2010.

[10] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empir. Softw. Eng.*, 19(3):619–654, 1 June 2014.

[11] James O Berger, Robert L Wolpert, M J Bayarri, M H DeGroot, Bruce M Hill, David A Lane, and Lucien LeCam. The likelihood principle. *Lect. Notes Monogr. Ser.*, 6:iii–199, 1988.

[12] Lauren R Biggers, Cecylia Bocovich, Riley Capshaw, Brian P Eddy, Letha H Etzkorn, and Nicholas A Kraft. Configuring latent dirichlet allocation based feature location. *Empir. Softw. Eng.*, 19(3):465–500, 1 June 2014.

[13] David Blei, Lawrence Carin, and David Dunson. Probabilistic topic models: A focus on graphical model design and applications to document and image analysis. *IEEE Signal Process. Mag.*, 27(6):55–65, 1 November 2010.

[14] David M Blei. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[15] David M Blei. *Probabilistic models of text and images*. PhD thesis, University of California, Berkeley, 2004.

[16] David M Blei. Topic models, machine learning summer school. Lecture, 2009.

[17] David M Blei. Introduction to probabilistic topic models. Princeton University, 2011.

[18] David M Blei. Mixed membership models. *Princeton University*, 2011.

[19] David M Blei and Jonathan Chang. Hierarchical Relational Models For Document Networks. *Ann. Appl. Stat.*, 4(1):124–150, 2010.

[20] David M Blei, Sean Gerrish, Chong Wang, Jordan L Boyd-graber, and Jonathan Chang. Reading tea leaves: How humans interpret topic models. In Y Bengio, D Schuurmans, J D Lafferty, C K I Williams, and A Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 288–296. Curran Associates, Inc., 2009.

[21] David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Bayesian Anal.*, 1(1):121–143, March 2006.

[22] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM.

[23] George Casella and Edward I George. Explaining the gibbs sampler. *Am. Stat.*, 46(3):167–174, 1 August 1992.

[24] Aggarwat Charu and Chengxiang Zhai. Dimensionality reduction and topic modeling. In *Mining Text Data*, pages 140–148. Springer, 2012.

[25] Kenneth Church and William Gale. Inverse document frequency (IDF): A measure of deviations from poisson. In Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, Text, Speech and Language Technology, pages 283–295. Springer Netherlands, 1999.

[26] Reidar Conradi and Alf Inge Wang. *Empirical methods and studies in software engineering: experiences from ESERNET*, volume 2765. Springer, 2003.

[27] William M Darling. A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 642–647. ailab.chonbuk.ac.kr, 2011.

[28] Scott Deerwester, Susan Dumais, Thomas Landauer, and George Furnas. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[29] Oscar Dieste, Anna Grim, Natalia Juristo, Himanshu Saxena, et al. Quantitative determination of the relationship between internal validity and bias in software engineering experiments: Consequences for systematic literature reviews. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 285–294. IEEE, 2011.

[30] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. Feature location in source code: a taxonomy and survey. *J. Softw. Evol. and Proc.*, 25(1):53–95, 1 January 2013.

[31] Charles Elkan. Text mining and topic models. *University of California at San Diego (Lecture notes)*, pages 1–13, 2014.

[32] Felix Endres, Christian Plagemann, Cyrill Stachniss, and Wolfram Burgard. Unsupervised discovery of object classes from range data using latent dirichlet allocation. In *Robotics: Science and Systems*, volume 2, pages 113–120. pdfs.semanticscholar.org, 2009.

[33] Daniel Falush, Matthew Stephens, and Jonathan K Pritchard. Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies. *Genetics*, 164(4):1567–1587, August 2003.

[34] Emily B. Fox and Carlos Guestrin. Machine learning: Clustering & retrieval on-line course, university of washington. `https://coursera.org`, 2016. Accessed: 2017-5-21.

[35] Yang Gao, Jianfei Chen, and Jun Zhu. Streaming gibbs sampling for LDA model. 6 January 2016.

[36] James E Gentle. *Computational Statistics*. Springer Science & Business Media, 28 July 2009.

[37] Malcom Gethers, Rocco Oliveto, Denys Poshyvanyk, and Andrea De Lucia. On integrating orthogonal information retrieval methods to improve traceability recovery. In *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, pages 133–142. ieeexplore.ieee.org, 2011.

[38] Scott Grant and James R Cordy. Estimating the optimal number of latent concepts in source code analysis. In *2010 10th IEEE Working Conference on Source Code Analysis and Manipulation*, pages 65–74, 2010.

[39] Scott Grant, James R Cordy, and David B Skillicorn. Using heuristics to estimate an appropriate number of latent topics in source code analysis. *Science of Computer Programming*, pages 1673–1678, 2013.

[40] Derek Greene, Derek OCallaghan, and Pádraig Cunningham. How many topics? stability analysis for topic models. In *Machine Learning and Knowledge Discovery in Databases*, pages 498–513. Springer, Berlin, Heidelberg, 15 September 2014.

[41] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proc. Natl. Acad. Sci. U. S. A.*, 101 Suppl 1:5228–5235, 6 April 2004.

[42] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, 2009.

[43] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–317–IV–320. ieeexplore.ieee.org, April 2007.

[44] Matthew D Hoffman, David M Blei, and Perry R Cook. Content-Based musical similarity computation using the hierarchical dirichlet process. In *ISMIR*, pages 349–354. books.google.com, 2008.

[45] Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *J. Mach. Learn. Res.*, 14(1):1303–1347, 2013.

[46] Thomas Hoffman. Probabilistic latent semantic indexing. *in Proceedings of the Twenty-Second Annual International SIGIR Conference*, 1999.

[47] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42:177–196, 2001.

[48] Kurt Hornik and Bettina Grün. topicmodels: An R package for fitting topic models. *J. Stat. Softw.*, 40(13):1–30, 2011.

[49] Ritika Jain, Smita Ghaisas, and Ashish Sureka. SANAYOJAN: A framework for traceability link recovery between use-cases in software requirement specification and regulatory documents. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, RAISE 2014, pages 12–18, New York, NY, USA, 2014. ACM.

[50] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with applications in R*. Springer, 2015.

[51] Hayato Kobayashi. Perplexity on reduced corpora. In *ACL (1)*, pages 797–806, 2014.

[52] Franco Lancia. Word co-occurrence and theory of meaning. *Retrieved August*, 18:2007, 2005.

[53] Kenneth Lange. Singular value decomposition. In *Numerical Analysis for Statisticians*, Statistics and Computing, pages 129–142. Springer New York, 2010.

[54] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. An overview of topic modeling and its current applications in bioinformatics. *Springerplus*, 5(1):1608, 20 September 2016.

[55] Panagiotis Louridas, Diomidis Spinellis, and Vasileios Vlachos. Power laws in software. *ACM Trans. Softw. Eng. Methodol.*, 18(1):2:1–2:26, October 2008.

[56] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[57] Stacy K Lukins. *Source Code Retrieval for Bug Localization using Latent Dirichlet Allocation and its relationship to stability of Agilely developed Software*. PhD thesis, The University of Alabama in Huntsville, 2009.

[58] Stacy K Lukins, Nicholas A Kraft, and Letha H Etzkorn. Bug localization using latent dirichlet allocation. *Information and Software Technology*, 52(9):972–990, 2010.

[59] Duc Minh Luu, Ee-Peng Lim, and Freddy Chong Tat Chua. On modeling brand preferences in item adoptions. In *ICWSM*. pdfs.semanticscholar.org, 2014.

[60] Adrian Marcus, Andrey Sergeyev, Vaclav Rajlich, and others. An information retrieval approach to concept location in source code. *2004. Proceedings. 11th* , 2004.

[61] Qiaozhu Mei, Xuchua Shen, and Chengxiang X Zhai. Automatic labeling of multinomial topic models. *Proceedings of the 13th ACM SIGKDD*, 2007.

[62] Mej Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, 1 September 2005.

[63] Rocco Oliveto, Malcom Gethers, Denys Poshyvanyk, and Andrea De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *2010 IEEE 18th International Conference on Program Comprehension*, pages 68–71. ieeexplore.ieee.org, June 2010.

[64] Dan Oneata. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty*, pages 1–7, 1999.

[65] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International Conference on Software Engineering (ICSE)*, 2013.

[66] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In

Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 569–577. ACM, 24 August 2008.

[67] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning, Rutgers University*, volume 242, pages 133–142, 2003.

[68] Philip Resnik and Eric Hardisty. Gibbs sampling for the uninitiated. Technical report, University of Maryland, 2010.

[69] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[70] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

[71] Prasanna Sahoo. Probability and Mathemaical Statistics. pages 214–221. University of Louisville, 2015.

[72] Gerard Salton and Michael Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[73] Janet Siegmund, Norbert Siegmund, and Sven Apel. Views on internal and external validity in empirical software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 9–19. ieeexplore.ieee.org, May 2015.

[74] Vivek Kumar Rangarajan Sridhar. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of NAACL-HLT*, pages 192–200, 2015.

[75] stackexchange.com. Android enthusiasts questions and answers. `http://android.stackexchange.com`. Accessed: 2015-9-NA.

[76] stackexchange.com. Database administrators questions and answers. `http://dba.stackexchange.com`. Accessed: 2015-9-NA.

[77] stackexchange.com. Salesforce questions and answers. `https://salesforce.stackexchange.com`. Accessed: 2015-9-NA.

[78] Mark Steyvers and Tom Griffiths. Handbook of latent semantic analysis, chapter 21: Probabilistic topic models, 2007.

[79] Matthew Taddy. On estimation and selection for topic models. In *AISTATS*, pages 1184–1193, 2012.

[80] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *J. Am. Stat. Assoc.*, 101(476):1566–1581, 1 December 2006.

[81] Stephen W Thomas, Hadi Hemmati, Ahmed E Hassan, and Dorothea Blostein. Static test case prioritization using topic models. *Empir. Softw. Eng.*, 19(1):182–212, 1 February 2014.

[82] Kai Tian, Meghan Revelle, and Denys Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 163–166. ieeexplore.ieee.org, May 2009.

[83] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In Daniel P Berrar, Werner Dubitzky, and Martin Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109. Springer US, 2003.

[84] Hanna M Wallach, David M Mimno, and Andrew McCallum. Rethinking LDA: Why priors matter. In Y Bengio, D Schuurmans, J D Lafferty, C K I Williams, and A Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1973–1981. Curran Associates, Inc., 2009.

[85] Biao Wang, Yang Liu, Zelong Liu, and Maozhen Li. Topic selection in latent dirichlet allocation. In *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 756–760. ieeexplore.ieee.org, August 2014.

[86] Shaowei Wang, David Lo, and Lingxiao Jiang. An empirical study on developer interactions in StackOverflow. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1019–1024, New York, NY, USA, 2013. ACM.

[87] Xiaogang Wang, Xiaxu Ma, and Eric Grimson. Unsupervised activity perception by hierarchical bayesian models. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. ieeexplore.ieee.org, June 2007.

[88] Yang Wang, Payam Sabzmeydani, and Greg Mori. Semi-Latent dirichlet allocation: A hierarchical model for human action recognition. In *Human Motion – Understanding, Modeling, Capture and Animation*, pages 240–254. Springer, Berlin, Heidelberg, 2007.

[89] Xing Wei and W Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 178–185, New York, NY, USA, 2006. ACM.

[90] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. Data mining: Practical machine learning tools and techniques. page 371. Morgan Kaufmann, 2016.

[91] Robert K. Yin. *Case study research: Design and methods*. Sage publications, 5th edition, 2013.

[92] Weizhong Zhao, James J Chen, Roger Perkins, Zhichao Liu, Weigong Ge, Yijun Ding, and Wen Zou. A heuristic approach to determine an appropriate number of topics in topic modeling. *BMC Bioinformatics*, 16(13):S8, 1 December 2015.

Glossary

F word frequency. 3, 54, 56–58, 68

K Refers to the selected number of topics. vi, ix, x, 2–4, 7, 8, 11, 14, 19, 21–23, 26, 28, 30, 31, 33, 35, 36, 42, 43, 46, 54, 56, 58, 61, 66, 82, 83

M Refers to the number of documents in the text corpus, denoted by D, consisting of M documents: $D = \{d_1, d_2, \ldots, d_M\}$. 28

N total number of documents in corpus. 3, 6, 54, 57, 58

X top X topics. 3, 54, 56–58, 68

α Hyperparameter of the model, controls the topic distributions per document. 29

β Hyperparameter of the model, controls the term distributions. 29

ϕ Multinomial distribution on z. 28

θ Multinomial distribution used to model the topic proportions. 28

d Refers to a document defined as a sequence of N words, described by $d = \{w_1, w_2, \ldots, w_N\}$ where w_N represents the nth word in the sequence. 28

w Refers to a term or word in the vocabulary (V). It is the basic unit of discrete data, and $w_i \in V$. 11, 28

SE-related text corpuses Any text corpus associated with software, processes, products and projects. 1, 82, 83

perplexity measurement of how well a probability distribution or probability model predicts a sample. 21, 22

polysemy when a word may have more than one meaning (i.e., crane, bank, etc.). 16, 17

synonymy when one word meaning can be expressed by many words (i.e., defect and bug). 17

Acronyms

BoW Bag-of-words. 11, 15, 27

CGS Collapsed Gibbs Sampling. 41–43

CSV Comma Separated Value file. 47

CV Cross-Validation. 21–23

DBMS Database Management System. 14

DM Data Mining. 12

DTM Dynamic Topic Models. vi, ix, 19, 30, 31, 34

FLT Feature Location Technique. 25

HDP Hierarchical Dirichlet Process. 21

idf inverse document frequency. 5, 6, 49

IR Information Retrieval. 12, 24, 82

ITS Issue Tracking System. 24

KL Kullback-Leibler divergence. 21, 44

LDA Latent Dirichlet Allocation. vi, viii–x, 1, 3, 4, 14, 18, 19, 23–31, 33, 35, 36, 39–42, 46–49, 54, 55, 76, 82, 115

LSI Latent Semantic Analysis. 17, 24

MCMC Markov Chain Monte Carlo. 41

pd probability density. ix, 36, 37, 39

pLSI Probabilistic Latent Semantic Analysis. ix, 17, 18, 29–31

posterior posterior distribution. 41, 43

prior prior probability. 11

Q&A Questions and Answers. 46, 48, 50

RMSE Root-Mean-Square Error. 68, 69, 71

SE Software Engineering. 1, 23, 24, 82

SVD Singular Value Decomposition. 17

SW Software. 1, 2, 23–25

tf term frequency. ix, 5, 6

tf-idf term frequency - inverse document frequency. 5, 6, 12, 17, 48

TMs Topic Models. 2, 3, 5, 16

XML Extensible Markup Language. 47