EMBARRASSINGLY PARALLEL STATISTICS AND ITS APPLICATIONS: DIVIDE & RECOMBINE METHODS FOR PARALLEL COMPUTATION OF QUANTILES AND CONSTRUCTION OF K-D TREES FOR BIG-DATA

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Aritra Chakravorty

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. William S. Cleveland, Chair
School of Statistics

Dr. Patrick J. Wolfe
School of Statistics

Dr. Mark D. Ward
School of Statistics

Dr. Ryan Hafen
School of Statistics

Approved by:

Dr. Hao Zhang
Head of the School Graduate Program

To my parents.

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor Prof. William S. Cleveland for guiding me through my research. I'm extremely grateful to my committee members Dr. Patrick J. Wolfe, Dr. Mark D. Ward and Dr. Ryan Hafen for their support and valuable contribution to my research. I thank my parents Mr. Dibakar Chakravorty and Mrs. Reeta Chakraborty for being supportive and encouraging me throughout this journey. They have been the kind guiding force behind all of my academic and personal achievements. I have to specially thank Mr. Dave LeFevre and Mr. Douglas Crabill from Purdue IT support for helping me in my need. I am in debt to almost every faculty members and office staffs of Purdue Statistics for their help and kindness. I thank all the past and present student in Prof. Cleveland's research group. They have been extremely kind and helpful. I'm very grateful to my professors, seniors, friends and juniors from the extended ISI Kolkata family in Purdue, to my amazing roommates in Faith-West from PBFI and my friends in Crosswalk from SLCF in Purdue, to all amazing people from my days in Purdue Philharmonic and Symphony Orchestra, to all my senors and friends from Purdue University Tagore Society and all my friends and family. They made my stay at Purdue very memorable

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

SYMBOLS

$\mathbb{R}$	Set of real-numbers
$\mathbb{C}$	Set of complex-numbers
$\mathbb{N}$	Set of integers
$\mathbb{Z}^+$	Set of positive natural numbers
$L^2(S)$	Set of L^2 integrable functions on set S
$L^p(S)$	Set of L^p integrable functions on set S
i	Unit of complex numbers
$\{\{A_{i,j}\}\}$	A matrix whose (i,j)th entry is $A_{i,j}$
A^t	Transpose of the matrix A
$A_{i,.}$	ith row of A.
$A_{.,j}$	jth column of A.
$\|H\|$	Norm of H.
$\langle G, H\rangle$	Inner-product of G and H.
$E_P(.)$	Expectation wrt measure P.

ABBREVIATIONS

D&R	Divide and Recombine
EPS	Embarrassingly Parallel Statistics
SEP	Strongly Embarrassingly Parallel
WEP	Weakly Embarrassingly Parallel
SOT	Sum of Transformations
EPF	Embarrassingly Parallel Functions
STF	Sum of Transformation Functions
FAS	Finitely Additively Separable
EP-FQ	Embarrassingly Parallel Fourier Quantile
EP-FQCh	Embarrassingly Parallel Fourier Quantile with Chebyshev's transformation
EP-FKD	Embarrassingly Parallel Fourier KD-tree

ABSTRACT

Chakravorty, Aritra PhD, Purdue University, December 2018. Embarrassingly Parallel Statistics and its Applications: Divide & Recombine Methods for Parallel Computation of Quantiles and Construction of K-D Trees for Big-Data . Major Professor: William S. Cleveland.

In Divide & Recombine (D&R), data are divided into subsets, analytic methods are applied to each subset independently, with no communication between processes; then the subset outputs for each method are recombined. For big data, this provides almost all of the analytic tasking needed when data are analyzed. It also provides high computational performance because typically most of the computation is embarrassingly parallel, the simplest parallel computation.

Another kind of tasking must address computational performance and numeric accuracy: the computing of functions of all of the data, or "statistics". For data big and small, it is often important to compute such statistics for all of the data, which can be summaries of the data, such as sample quantiles of continuous variables, or can process the data into a form that helps analysis, such as dividing the data into representative subsets. Development of computational methods to compute these statistics can be challenging.

D&R can be a very effective framework for computing statistics. To support this, we introduce the concept of embarrassingly parallel (EP) statistics, both weak and strong. The concept of EP statistics is not entirely new, but has had little development. The existing methodology is mainly sums of sums. For example, this is done when computing the necessary statistics for least squares where sums of products and cross productions are carried out on subsets then summed across subsets. Our treatment of EP statistics has taken the concept much further. The outcome is ability to use EP statistics in conjunction with the use a Fourier series to approximate an

optimization criteria. The series terms, which are strongly EP statistics, are summed across subsets, and the result is optimized. These are EP-F computational methods.

We have so far developed two EP-F computational methods for two widely used statistic computations. EP-F-Quantile is for quantiles of big data, and EP-F-KDtree is for KD-trees. Speed and accuracy of EPF-Quantile are compared with that of the well-known binning method, which also can be formulated in terms of EP statistics. EPF-KDtree is the first parallel KD-tree computational method of which we are aware. EP and EPF computational methods have potentially many other applications to computing statistics.

1. INTRODUCTION:

Divide and Recombine(D&R) as discussed in [1] is a statistical approach to big data. A statistical division method divides the data into subsets that are written to disk with the same data structures Then, each of a collection of analytic methods is applied to each subset, and the outputs are data structures that are the same as one another. The subset analytic computations have no communication among them making them embarrassingly parallel, the simplest parallel computation. Finally, a recombination method is applied to the outputs for each method to get a final D&R result. The recombination step does have communication between the processes, but typically has a component of embarrassingly parallel computation.

1.1 D&R implementation with Map-Reduce:

DeltaRho software (www.deltarho.org) implements D&R. At the front end, a data analyst programs in the R language.The *DeltaRho* front end *R* package, *datadr* makes programming D&R easy. It can run on a multicore machine and manage the back-end parallel computations and database management. It can also run on a cluster and use the Hadoop parallel, distributed computational environment to manage the computations with *Map-Reduce* and the database with the *Hadoop Distributed File System (HDFS)*. Hadoop keeps track of subsets and outputs and runs the *R/datadr* code for division, applications of methods, and recombinations. Communication between *datadr* and *Hadoop* is achieved by the *DeltaRho RHIPE* software, the *R* and *Hadoop* Integrated Programming Environment.

D&R with DeltaRho does not require the highly limiting requirement that the data reside memory to get high computational performance. In fact, data-sets can have a memory size bigger than the physical memory size. The methods presented in

this paper, including EP-FQ, are implemented as part of an R package, *drEP*, which uses *datadr* extensively.

1.2 Numerical accuracy in D&R and EPS

D&R is extremely fast and efficient in dealing with large data. But there is an issue regarding numerical accuracy in D&R output, depending on the analytic method we need to perform during the analytic stage of D&R. We are comparing the D&R output to the the output value we had obtained if we had chosen to apply the analytic method on the entire data in memory without dividing. For some analytic methods, the D&R output can be numerically very inaccurate compared to the in-memory output. The accuracy also depends on the way we choose to partition data into subsets.

Let us illustrate with an easy example, suppose the data is $\{Y, X\}$: 100 observations of the pair of variables $\{y, x\}$. For D&R method, we divide it in two subsets $\{Y_1, X_1\}$ and $\{Y_2, X_2\}$, each subset having 50 observations. If the analytic method is to compute mean of Y, for D&R output we take the average of subset means $\bar{Y}_1$ and $\bar{Y}_2$. Note that we have $\bar{Y} = \frac{\bar{Y}_1+\bar{Y}_2}{2}$, so, in this specific situation D&R is numerically accurate for computing means. Observe that the D&R output is invariant w.r.t. the way we divide the data into subsets

But if the analytic method is to calculate the coefficient of logistic regression of y given x. If the analytic method is applied on entire data $\{Y, X\}$, we get an estimator of logistic regression coefficient $\hat{\beta}_L(X, Y)$. If we perform D&R, the analytic method is applied to the subset and we get an estimator of logistic regression coefficient for each subset: $\hat{\beta}_L(X_1, Y_1)$ and $\hat{\beta}_L(X_2, Y_2)$. Here, the recombination step gets difficult if we want numeric accuracy. In general, $\hat{\beta}_L(X, Y) \neq \frac{\hat{\beta}_L(X_1,Y_1)+\hat{\beta}_L(X_2,Y_2)}{2}$. In fact, we can't find a recombination function f that will satisfy $\hat{\beta}_L(X, Y) = f(\frac{\hat{\beta}_L(X_1,Y_1)+\hat{\beta}_L(X_2,Y_2)}{2})$ for any $\{X, Y\}$. So, whatever recombination method we choose, it will not be numerically

completely accurate, it will also depend on the way of partitioning the data. We may ensure statistical accuracy with small random error by carefully choosing f.

In this thesis, we try to identify exactly numerically accurate analytic methods by introducing the concept of Embarrassingly Parallel statistics. Also we present algorithms that systematically controls the numeric error for analytic methods that do not belong to this set. Two algorithms: EP-FQ and EP-FKD are presented in later sections illustrating this mechanism.

2. EMBARRASSINGLY PARALLEL STATISTICS

To understand the intuition behind Embarrassingly Parallel Statistics, recall the example we discussed in last section. When the analytic method for D&R is: computing mean of a variable, the D&R output stays invariant w.r.t. the choice of partitioning the data. So, irrespective of how we divide the data into subsets, in this process, the collection of subset outputs of the analytic method, are mapped to a singular value for a given data. Keeping this property of D&R computation of mean, we now formally define our concept in focus.

2.1 Definitions and Examples

Let $\mathbf{X}$ be an $N \times d$ matrix whose rows are N i.i.d. observations (realizations) of a d-dimensional random variable $\tilde{\mathbf{x}}$. We divide the observations into R disjoint subsets with M_r observations for $r = 1, \ldots, R$. So the rth subset is an $M_r \times d$ matrix $\mathbf{X_r}$.

Let $T(.)$ be a numeric scalar or vector-valued statistic that is applied to $\mathbf{X}$ and $\mathbf{X_r}$. $T(.)$ is defined to be a Strongly Embarrassingly Parallel(SEP) statistic if there exists a function $f(.)$ such that $T(\mathbf{X}) = f(T(\mathbf{X}_1), T(\mathbf{X}_2), \cdots, T(\mathbf{X}_R))$.

$T(.)$ is defined to be a Weakly Embarrassingly Parallel(WEP) statistic if it is not a SEP statistic, but can be written as a function of one or more SEP statistics.

D&R has associated with it a notion of statistical accuracy. If an analytic method is applied to each of the $\mathbf{X_r}$ and then recombined, we want to chose division and recombination methods that provide a statistical accuracy as high as possible and as close as possible to the accuracy that we could have gotten had we been able to apply the method to $\mathbf{X}$ directly. Statistical accuracy is closely related to numeric accuracy and it is the concept for EPS. We want to compute statistics from $\mathbf{X}$, for example, quantiles, and seek to use operations based on $\mathbf{X_r}$ to get approximations of the values

for $\mathbf{X}$ whose numeric accuracy is as high as possible. The reason is that quantiles serve as a numeric description of the properties of observed values of a variable. Now a statistic that serves as a summary where numeric accuracy is important, can also be used for tasks that require its statistical accuracy to be considered. We now give few examples with $d = 1$ to illustrate EP statistic.

Example 2.1.1. Let $N(.)$ be a statistic whose output is the sample sizes of different set of observations. Then we have $N(\mathbf{X}) = \sum_{r=1}^{R} T(\mathbf{X_r}) = f(M_1, \ldots, M_R)$, where $f(.)$ is a function of numeric variables that sums its arguments. So $N(.)$ is a SEP statistic.

Example 2.1.2. If $T(.)$ is the summing function $f(.)$ in Example 2.1.1, then clearly $T(.)$ is a SEP statistics where, $T(\mathbf{X}) = f(T(\mathbf{X}_1), \ldots, T(\mathbf{X}_R))$.

Example 2.1.3. Let $T(.)$ be the sample maximum, then $T(\mathbf{X}) = f(T(\mathbf{X_1}), \ldots, T(\mathbf{X_R}))$, so $T(.)$ is a SEP statistic and $f(.) = T(.)$.

Example 2.1.4. Let $\mathbf{X}_r$ and $\mathbf{Y}_r$ denote data in rth subset for $r = 1, 2, \cdots, R$; $\mathbf{X}$ and $\mathbf{Y}$ denote the combined data. Consider the linear regression model: $\mathbf{y} = \mathbf{x}\beta' + \epsilon, \epsilon \sim N(0, \sigma^2)$. We know the LS estimate for β is $\tilde{\beta} = (\mathbf{X}\mathbf{X}')^{-1}\mathbf{X}\mathbf{Y}'$. Now this is WEP statistics. Because if you consider the statistics $T_1(\mathbf{X}, \mathbf{Y}) = \mathbf{X}\mathbf{Y}'$ and $T_2(\mathbf{X}) = \mathbf{X}\mathbf{X}'$, then, $T_1(\mathbf{X}, \mathbf{Y}) = \mathbf{X}\mathbf{Y}' = \sum_{r=1}^{R} \mathbf{X}_r\mathbf{Y}_r' = f(T_1(\mathbf{X}_1, \mathbf{Y}_1), \ldots, T_1(\mathbf{X}_R, \mathbf{Y}_R))$ and $T_2(\mathbf{X}) = \mathbf{X}\mathbf{X}' = \sum_{r=1}^{R} \mathbf{X}_r\mathbf{X}_r' = f(T_2(\mathbf{X}_1), \ldots, T_2(\mathbf{X}_R))$, again $f(.)$ is the function that sums its arguments. This makes both $T_1(\mathbf{X}, \mathbf{Y})$ and $T_2(\mathbf{X})$ SEP statistics, and since $\hat{\beta}$ can be expressed in terms of $T_1(\mathbf{X}, \mathbf{Y})$ and $T_2(\mathbf{X})$, it is a WEP statistic.

Example 2.1.5. Let $T(\mathbf{X})$ be the sample median. It is not SEP or WEP statistic, because we cannot write $T(\mathbf{X}) = f(T(\mathbf{X}_1), T(\mathbf{X}_2), \ldots, T(\mathbf{X}_r))$, for any function $f(.)$, neither can we express $T(.)$ in terms of finitely many SEP statistics. The subset medians do not contain sufficient information to determine the median of all of the data.

2.2 The Class of EP statistics

SEP and WEP statistics are candidates for easy, fast computation using D&R. Identification of cases and classes of SEP statistics and WEP statistics provides information about the use of D&R EP statistics in practice.Suppose the random variable $\tilde{\mathbf{x}}$ has a probability density $p(\tilde{\mathbf{x}};\beta)$ where β is a finite number of parameters. Suppose the rows of $\mathbf{X}$ are observed values of independent realizations of $\tilde{\mathbf{x}}$.

Theorem 2.2.1. *Any Minimal Sufficient Statistic for a parametric model is SEP statistic.*

(See Appendix A for proof.)

A class of distributions is an L-parameter exponential family if the densities have the form $p(\tilde{\mathbf{x}};\beta) = h(\tilde{\mathbf{x}})exp\big(\sum_{l=1}^{L}\eta_l(\beta)\tau_l(\tilde{\mathbf{x}}) - A(\beta)\big)$. If the rows of $\mathbf{X}$ are observed values of $\tilde{\mathbf{x}}$ having density $p(\tilde{\mathbf{x}};\beta)$ and $\{\eta_1(\beta),\eta_2(\beta),\ldots,\eta_L(\beta)\}$ is a linear independent set, then the L dimensional statistic: $T_m(\mathbf{X}) = \big(\sum_{\mathbf{x}\in\mathbf{X}}\tau_1(\mathbf{x}),\ldots,\sum_{\mathbf{x}\in\mathbf{X}}\tau_L(\mathbf{x})\big)$ is minimal sufficient statistic for β. Observe that the individual co-ordinate statistics have a general form $T(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}}\tau(\mathbf{x})$, for a transformation $\tau(.)$.

A statistic $T(.)$ is defined to be a Sum of Transformations (SOT) statistic if there exists a transformation $\tau(.)$ such that $T(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}}\tau(\mathbf{x})$. We say $T(.)$ is induced by $\tau(.)$.

Theorem 2.2.2. *An SOT statistic is a SEP statistic.*

(See Appendix A for proof.)

Clearly, the class of SOT statistics is a subset of the class of SEP statistics, minimal sufficient statistics for any exponential family distribution belongs to this set. But it doesn't include all possible SEP statistics. For example, let $\mathbf{x}$'s are i.i.d. $U(0,\beta)$, which doesn't belong to exponential family of distributions. An application of theorem 2.2.1 implies that the minimal sufficient statistics for β, $T_m(\mathbf{X}) = \max(\mathbf{X})$ is SEP statistic, which was verified in example 2.1.3, but it is not SOT. In the next section we generalize the concept of EP statistics to functions.

2.3 Function Spaces

Now, we extend the concept of EP statistics to complex valued functions involving a variable β. Assume a general data-point $\mathbf{x}$ is an observation of the random variable $\tilde{\mathbf{x}}$. This random variable has a sample space $\mathbb{X}$ and let $\mathbb{P}(\mathbb{X})$ be the power set of $\mathbb{X}$. $\mathbb{P}(\mathbb{X})$ is the set of samples of $\tilde{\mathbf{x}}$. Also assume $\tilde{\mathbf{x}}$ is related to a parameter-variable β belonging to the parameter space Θ. Consider a function $F(\mathbf{X}, \beta)$ having both $\mathbf{X}$ and β as arguments, $F(\mathbf{X}, \beta) : \mathbb{P}(\mathbb{X}) \times \Theta \mapsto \mathbb{C}$.

A function $F(\mathbf{X}, \beta)$ is defined to be an Embarrassingly Parallel Function(EPF) if given β, $F(\mathbf{X}, \beta)$ is a SEP statistics, i.e., there exists a function $f_\beta()$ (which may depends on β), such that $F(\mathbf{X}, \beta) = f_\beta(F(\mathbf{X}_1, \beta), F(\mathbf{X}_2, \beta), \cdots, F(\mathbf{X}_r, \beta))$, for any β.

Note that the function $f_{\mathbf{X},F}()$, defined by $f_{\mathbf{X},F}(\beta) = F(\mathbf{X}, \beta)$ for $\beta \in \Theta$ is a member of $\mathbb{C}^\Theta$: Set of functions from Θ to $\mathbb{C}$, and for a fixed F, the set $S_F = \{f_{\mathbf{X},F}() : \mathbf{X} \in \mathbb{X}\}$ is a function space. We consider functional operators defined on this function-space and conditions that would make them WEP statistics.

A function $F(\mathbf{X}, \beta)$ is defined to be a Sum of Transformation(STF) Function if there exists a function $H(\mathbf{x}, \beta) : \mathbb{X} \times \Theta \mapsto \mathbb{C}$ such that $F(\mathbf{X}, \beta) = \sum_{\mathbf{x} \in \mathbf{X}} H(\mathbf{x}, \beta)$. We say $F(\mathbf{X}, \beta)$ is induced by $H(\mathbf{x}, \beta)$.

Then, we have:

Theorem 2.3.1. *A STF is an EPF.*

(See Appendix A for proof.)

Example 2.3.2. Consider the case where $\mathbf{x}$ s are i.i.d. observations of random variable $\tilde{\mathbf{x}}$ with density $p(\tilde{\mathbf{x}}, \beta)$.The log-likelihood function $l(\mathbf{X}, \beta)$ is STF as $l(\mathbf{X}, \beta) = \sum_{\mathbf{x} \in \mathbf{X}} l(\mathbf{x}, \beta)$. Here $H(\mathbf{x}, \beta) = l(\mathbf{x}, \beta) = \log p(\mathbf{x}, \beta)$.

For the next 3 examples, let all $\mathbf{x}$'s are i.i.d. observations of real random variable $\tilde{\mathbf{x}}$, related to β, assume $\mathbb{X} \subseteq \mathbb{R}$ and $\Theta \subseteq \mathbb{R}$.

Example 2.3.3. Assume that $\Theta = (a, b) \subset \mathbb{R}$. Let $H(\mathbf{x}, \beta) = \mathbf{1}_{\{\mathbf{x} \leq \beta\}}$, so that $F(\mathbf{X}, \beta) = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{1}_{\{\mathbf{x} \leq \beta\}}$. Note that $F(\mathbf{X})$ is maximized by any member $\hat{\beta} \in [\max(\mathbf{X}), b)$. So if we consider the functional operator on S_F: $\mathbb{T}(f_{\mathbf{X},F}) = \inf_{\hat{\beta} \in \Theta}\{\hat{\beta} : f_{\mathbf{X},F}(\hat{\beta}) = \max_{\beta \in \Theta} f_{\mathbf{X},F}(\beta)\}$, we can see that $\mathbb{T}(f_{\mathbf{X},F}) = \max(\mathbf{X})$. It was previously shown to be SEP statistics.

Example 2.3.4. Assume that $\Theta = \mathbb{R}$ and again let $H(\mathbf{x}, \beta) = \mathbf{1}_{\{\mathbf{x} \leq \beta\}}$, so that $F(\mathbf{X}, \beta) = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{1}_{\{\mathbf{x} \leq \beta\}}$. Suppose $\mathbf{X}$ has $N(\mathbf{X})$ observations and if we consider the functional operator on S_F: $\mathbb{T}(f_{\mathbf{X},F}) = \inf_{\hat{\beta} \in \Theta}\{\hat{\beta} : \frac{f_{\mathbf{X},F}(\hat{\beta})}{N(\mathbf{X})} = p\}$, we can see that $\mathbb{T}(f_{\mathbf{X},F}) = Q_p(\mathbf{X})$ which is the pth sample-quantile. It is not WEP statistics, we can't sort $\mathbf{X}$ and compute quantiles by D&R.

Example 2.3.5. Assume that $\Theta = \mathbb{R}$ and let $H(\mathbf{x}, \beta) = |\mathbf{x} - \beta|$, so that $F(\mathbf{X}, \beta) = \sum_{\mathbf{x} \in \mathbf{X}} |\mathbf{x} - \beta|$. Consider the functional operator on S_F: $\mathbb{T}(f_{\mathbf{X},F}) = \inf_{\hat{\beta} \in \Theta}\{\hat{\beta} : f_{\mathbf{X},F}(\hat{\beta}) = \min_{\beta \in \Theta} f_{\mathbf{X},F}(\beta)\}$, we can see that $T(\mathbf{X}) = \mathbb{T}(f_{\mathbf{X},F})$ =median($\mathbf{X}$). Median is a quantile and non-WEP statistics.

From these examples we can see that in general we can't tell if any arbitrary functional operator on the function space S_F will be a WEP statistics. But if we assume some restrictions on the structure of F, we can actually ensure that an arbitrary functional operator on the function space S_F to be WEP statistics. We first introduce a new class of functions.

A function $H(\mathbf{x}, \beta)$ is defined to be an Finitely Additively Separable (FAS) if there exists J pair of maps $\{f_1(\mathbf{x}), g_1(\beta)\}, \cdots, \{f_J(\mathbf{x}), g_J(\beta)\}$, here $f_j(\mathbf{x}) : \mathbb{X} \mapsto \mathbb{C}, g_j(\beta) : \Theta \mapsto \mathbb{C}$, for $j = 1, \cdots, J$ and J constants $\eta_1, \cdots, \eta_J$ such that $H(\mathbf{x}, \beta) = \sum_{j=1}^{J} \eta_j f_j(\mathbf{x}) g_j(\beta)$.

Example 2.3.6. Assume that $\mathbb{X} = \mathbb{R}$ and $\Theta = \mathbb{R}$ and let $H(\mathbf{x}, \beta) = (\mathbf{x} - \beta)^T$, so that $H(\mathbf{x}, \beta) = \sum_{t=0}^{T} \binom{T}{t} \mathbf{x}^t \beta^{T-t}$.Take $J = T + 1$, $f_j(\mathbf{x}) = \mathbf{x}^{j-1}$, $g_j(x) = (-\beta)^{T-j+1}$ and $\eta_j = \binom{T}{j-1}$, so that we have, $H(\mathbf{x}, \beta) = \sum_{j=1}^{J} \eta_j f_j(\mathbf{x}) g_j(\beta)$ and it is FAS.

Theorem 2.3.7. *Any arbitrary functional operator on the function space S_F for a STF $F(\mathbf{X}, \beta)$ induced by a FAS is WEP statistics.*

(See Appendix A for proof.)

In most situations $H(\mathbf{x}, \beta)$ is not Finitely Additively Separable, suppose in such a situation, we approximate $H(\mathbf{x}, \beta)$ by a FAS function $H_J(.,.)$ with respect to some distance. Here $H_J(.,.)$ can be written as sum of product of no less than J number of pairs $\{f_j(\mathbf{x}), g_j(\beta)\}$. Let $F(\mathbf{X}, \beta)$ is induced by $H(\mathbf{x}, \beta)$ and $F_J(\mathbf{X}, \beta)$ is induced by $H_J(\mathbf{x}, \beta)$ for any J. Then we propose to approximate the functional operator $\mathbb{T}(f_{\mathbf{X},F})$ by the functional operator $\mathbb{T}(f_{\mathbf{X},F_J})$.

In these paper, we will consider L^p approximation of $H(\mathbf{x}, \beta)$ by $H_J(\mathbf{x}, \beta)$. Specifically we will investigate the L^2 convergence properties of $H_J(\mathbf{X}, \beta)$ to $H(\mathbf{x}, \beta)$, because for any interval $S \subseteq \mathbb{R}$, $L^2(S)$ is the only Hilbert-Space among $L^p(S)$ spaces and has mathematical properties that we can exploit.

Let us consider the spaces $L^2(\mathbb{X}), L^2(\Theta)$ and $L^2(\mathbb{X} \times \Theta)$ simultaneously. Note that $fg \in L^2(\mathbb{X} \times \Theta)$ whenever $f \in L^2(\mathbb{X})$ and $g \in L^2(\Theta)$. This is the reason the classes of functions $\zeta_0 \subseteq \zeta_1 \subseteq \zeta_2 \subseteq \cdots$ defined by:

$$\zeta_J = \Big\{ \sum_{j=1}^{J} f_j g_j \;\Big|\; f_j \in L^2(\mathbb{X}) \text{ and } g_j \in L^2(\Theta),\ 0 \le j \le J \Big\} \tag{2.1}$$

form a family of subsets in $L^2(\mathbb{X} \times \Theta)$. For convenience, we also consider the one element set ζ_0 containing the zero function of $L^2(\mathbb{X} \times \Theta)$. The following theorem shows the existence of a L^2 approximation:

Theorem 2.3.8. *Suppose that $H \in L^2(\mathbb{X} \times \Theta)$ is a non-zero function (i.e.$H \notin \zeta_0$) and put*

$$\omega = \begin{cases} J & if\ H \in \zeta_J/\zeta_{J-1} \\ \infty & if\ H \notin \zeta_J\ for\ any\ J \in \mathbb{N} \end{cases} \tag{2.2}$$

Then there exist two orthogonal systems $\{u_j\}_{j=1}^{\omega} \subset L^2(\mathbb{X}), \{v_j\}_{j=1}^{\omega} \subset L^2(\Theta)$ and a non-increasing sequence $\{\eta_j\}_{j=1}^{\omega}$ of positive reals such that

$$H(\mathbf{x}, \beta) = \sum_{j=1}^{\omega} \eta_j \bar{u}_j(\mathbf{x}) v_j(\beta) \quad for\ almost\ all\ (\mathbf{x}, \beta) \in (\mathbb{X} \times \Theta). \tag{2.3}$$

The subtotals of the extension form above are the best L_2-approximation of the function H in the sense that

$$\|H - \sum_{j=1}^{J} \eta_j \bar{u}_j v_j\| = \rho_J(H) = \sqrt{\|H\|^2 - \sum_{j=0}^{J} \eta_j^2} \quad for\ any\ n < \omega \tag{2.4}$$

Finally, the numbers η_j satisfy

$$\sum_{j=1}^{\omega} \eta_j^2 = \|H\|^2 \tag{2.5}$$

This result also gives a mechanism to approximate a general $H(\mathbf{x}, \beta)$ by sum of separable functions if $H(\mathbf{x}, \beta) \in L^2(\mathbb{X} \times \Theta)$, we may consider any orthogonal systems in $\mathbb{X}$ and in Θ and see if $H(\mathbf{x}, \beta)$ can be expressed as 2.3. For example, the functions $\{U_j(z) = e^{i\ 2\pi jz} : j \in \mathbb{Z}^+\}$ is an orthonormal basis of $L^2(0, 1)$, this is the basis for Fourier series expansion. We now give an example that will finaly lead us to EP-FQ algorithm:

Example 2.3.9. Recall example 2.3.5, we know $\sum_{\mathbf{x}\in\mathbf{X}} |\mathbf{x} - \beta|$is minimized for $\hat{\beta} =$ median($\mathbf{X}$). Suppose $\mathbf{x}$ is scaled to the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$ by a linear transformation, so the sample-space becomes $\mathbb{X} = (-\frac{\pi}{2}, \frac{\pi}{2})$. Now since the median also lies in the same interval, we have $\Theta = (-\frac{\pi}{2}, \frac{\pi}{2})$. Since both $-\frac{\pi}{2} < \mathbf{x} < \frac{\pi}{2}$ and $-\frac{\pi}{2} < \beta < \frac{\pi}{2}$, we have, $-\pi < |\mathbf{x} - \beta| < \pi$. Remember, if $-\pi < z < \pi$, then we have the Fourier expansion of $|z|$ as $|z| = \frac{\pi}{2} - \frac{4}{\pi}\sum_{j=1}^{\infty} \frac{\cos\left((2j-1)z\right)}{(2j-1)^2}$. If we replace z with $\mathbf{x} - \beta$ in this expression, we get: $|\mathbf{x} - \beta| = \frac{\pi}{2} - \frac{4}{\pi}\sum_{j=1}^{\infty} \frac{\cos\left((2j-1)(\mathbf{x}-\beta)\right)}{(2j-1)^2}$. Let $H(\mathbf{x}, \beta) = \frac{\pi}{2} - |\mathbf{x} - \beta|$, then we have, $H(\mathbf{x}, \beta) = \frac{4}{\pi}\sum_{j=1}^{\infty} \frac{cos(\{2j-1\}(\mathbf{x}-\beta)}{(2j-1)^2}$. Since, H is bounded in $\mathbb{X} \times \Theta$, $H \in L^2(\mathbb{X} \times \Theta)$. H is not finitely separable, hence $\omega = \infty$ for H. By theorem 2.3.8, we should be able to find orthognal systems $\{u_j\}_{j=1}^{\infty} \subset L^2(\mathbb{X}), \{v_j\}_{j=1}^{\infty} \subset L^2(\Theta)$ and non-increasing sequence $\{\eta_j\}_{j=1}^{\infty}$ such that 2.3 holds for H. Let $\{u_{2j-1}(\mathbf{x}) = \frac{e^{-i(2j-1)\mathbf{x}}}{\sqrt{\pi}}, u_{2j}(\mathbf{x}) = \frac{e^{i(2j-1)\mathbf{x}}}{\sqrt{\pi}}\}$; $\{v_{2j-1}(\mathbf{x}) = \frac{e^{-i(2j-1)\beta}}{\sqrt{\pi}}, v_{2j}(\mathbf{x}) = \frac{e^{i(2j-1)\beta}}{\sqrt{\pi}}$ for all $j \in \mathbb{Z}^+$. Then by the lemma in Appendix A, $\{u_j(\mathbf{x}) : j \in \mathbb{Z}^+\}$ and $\{v_j(\beta) : j \in \mathbb{Z}^+\}$ are orthonormal basis in $L^2(\mathbb{X})$ and $L^2(\Theta)$ respectively. Now, we have $\bar{u}_{2j-1}(\mathbf{x})v_{2j-1}(\beta) + \bar{u}_{2j}(\mathbf{x})v_{2j}(\beta) = \frac{e^{i(2j-1)\mathbf{x}}}{\sqrt{\pi}}.\frac{e^{-i(2j-1)\beta}}{\sqrt{\pi}} + \frac{e^{-i(2j-1)\mathbf{x}}}{\sqrt{\pi}}.\frac{e^{i(2j-1)\beta}}{\sqrt{\pi}} = \frac{2cos(\{2j-1\}(\mathbf{x}-\beta)}{\pi}$. If we replace $\frac{2cos(\{2j-1\}(\mathbf{x}-\beta)}{\pi}$ by

$\bar{u}_{2j-1}(\mathbf{x})v_{2j-1}(\beta)+\bar{u}_{2j}(\mathbf{x})v_{2j}(\beta)$ in the Fourier expression of $|\mathbf{x}-\beta|$, we have $H(\mathbf{x},\beta) = \sum_{j=1}^{\infty}\eta_j\bar{u}_j(\mathbf{x})v_j(\beta)$ for all $(\mathbf{x},\beta) \in (\mathbf{X}\times\Theta)$; here $\eta_{2j-1} = \eta_{2j} = \frac{2}{(2j-1)^2}$, also, observe that η_j's are positive and non-increasing.

Observe that for the Hilbert Space $L^2(\mathbb{X}\times\Theta)$, the inner-product $\langle\cdot,\cdot\rangle$ between two elements $H_1(\mathbf{x},\beta)$ and $H_2(\mathbf{x},\beta)$ is defined as: $\langle H_1,H_2\rangle = \int_{\frac{\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{\pi}{2}}^{\frac{\pi}{2}} H_1(\mathbf{x},\beta)\bar{H}_2(\mathbf{x},\beta)d\mathbf{x}d\beta$ and $\|H\|^2 = \langle H,H\rangle$. So, the partial sums of Fourier Series of $|\mathbf{x}-\beta|$ provides the best L^2 approximation to it, and the L^2 error goes to 0 if we increase the number of Fourier terms. We know that Fourier Series of $|z|$ uniformly converges to its limit. So the L^∞ error also goes to 0 if we increase the number of Fourier terms. This is a much stronger result. We have the following result as a corollary of 2.3.8

Corollary 2.3.10. *Consider the situation in theorem 2.3.8, if equation 2.3 holds for all* $(\mathbf{x},\beta) \in \mathbb{X}\times\Theta$*, then we have* $\|H - \sum_{j=1}^{J}\eta_j\bar{u}_jv_j\|_\infty \to 0$ *if*
(a)$u_j(\mathbf{x})$ *and* $v_j(\beta)$ *are uniformly bounded* $\forall j$*.*
(b)$\sum_{j=1}^{\infty}\eta_j < \infty$*.*

3. THE EP-FOURIER-QUANTILE ALGORITHM

For big data, the simple expedient of computing quantiles by sorting all of the values of a continuous variable, which is used for small data, can be impractically expensive or simply not feasible. Using an approach that computes quantiles on subsets and then recombines the quantile outputs creates accuracy problems. In this section we will describe an application of the concept of EP statistics: Embarrassingly-Parallel Fourier-Quantile or EP-FQ algorithm to get fast and accurate approximate quantiles for Big-Data.

EP-FQ uses a Fourier series to approximate an optimization criterion for quantiles. The series terms, which are strongly EP, are summed across subsets, and the result is optimized. Binning, another D&R method, is widely used for computing quantiles. It, too, can be formulated in terms of EP statistics. It creates equal-length intervals spanning the range of all of the data. The subset computation is to count the numbers of values in each bin. The recombination is to add up counts for each interval across subsets; then each quantile is computed by interpolation. In Section 3.7, the computational performance and accuracy of these two quantile methods are compared. In *datadr* binning is implemented by the function drQuantile(), and EP-FQ is implemented in the R package *drEP*. There are number of other quantile computational methods that are discussed in Section 3.8.

3.1 Basic idea

Remember, sample-median or more generally any sample-quantile is not an EP statistic. In EP-FQ algorithm we approximate sample-quantiles by a sequence of WEP statistics with strong control over the approximation, in a sense that these sequence of WEP statistics are best L^2 approximations with respect to an optimization

criteria. Quantiles corresponding to any arbitrary set of f-values can be computed by a single D&R step in EP-FQ method. However the initial version of EP-FQ is slow and we modify EP-FQ multiple times in later sections, using recursive relations involving Chebyshev's polynomials to get faster version of EP-FQ.

We assume all observations in the data $\mathbf{X}$ are i.i.d. observations of a real valued random variable $\tilde{\mathbf{x}}$. Denote the Pth sample quantile of the data $\mathbf{X}$ as $Q_P(\mathbf{X})$. First, note that,$Q_P(\mathbf{X})$ is the minimizer of the objective function: $F(\mathbf{X},\beta) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} |\mathbf{x}-\beta|_P$, wrt β. Here, $|z|_P = (1-P)z^- + Pz^+$, where, z^- and z^+ are the negative and the positive part of z respectively. Now we know, $z^- = \frac{(|z|-z)}{2}$ and $z^+ = \frac{(|z|+z)}{2}$. So, we have $|z|_P = (1-P)\frac{(|z|-z)}{2} + P\frac{(|z|+z)}{2} = \frac{|z|}{2} + \frac{z}{2}(2P-1)$. We assume that, there are real numbers m and M, such that, all the observations lie inside (m, M).(we can take $m = min(\mathbf{X}) - \delta$,$M = max(\mathbf{X}) + \delta$ for a small $\delta > 0$.) Then linearly transform $\mathbf{X}$ if necessary to ensure that the transformed observations lie inside $(-1, 1)$. Without loss of generality, let $Q_P(\mathbf{X})$ is the Pth sample quantile of this transformed data(we can get back the original sample quantiles easily by inverse linear transformations). Since all observations lie inside $(-1, 1)$, we can say that $Q_P(\mathbf{X})$ will also lie in $(-1, 1)$, so that, we can obtain $Q_P(\mathbf{X})$ by minimizing $F(\mathbf{X},\beta)$ with respect to β in the interval $(-1, 1)$, i.e.

$$Q_P(\mathbf{X}) = argmin_{\beta\in(-1,1)} \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} |\mathbf{x}-\beta|_P \tag{3.1}$$

In this interval we have, $|\mathbf{x}-\beta| < \pi$ for all $\mathbf{x} \in \mathbf{X}$ (their difference can be at-most 2) and we consider Fourier series expansion of $|\mathbf{x}-\beta|$ around 0:

$$\begin{aligned}
&|\mathbf{x}-\beta| \\
=&\frac{\pi}{2} - \sum_{j=1}^{\infty} \frac{4}{\pi(2j-1)^2} cos\big((2j-1)(\mathbf{x}-\beta)\big) \\
=&\frac{\pi}{2} - \frac{4}{\pi}\sum_{j=1}^{\infty} \frac{cos\big((2j-1)\mathbf{x}\big)cos\big((2j-1)\beta\big)}{(2j-1)^2} - \frac{4}{\pi}\sum_{j=1}^{\infty} \frac{sin\big((2j-1)\mathbf{x}\big)sin\big((2j-1)\beta\big)}{(2j-1)^2}
\end{aligned} \tag{3.2}$$

This means,

$$
\begin{aligned}
&|\mathbf{x}-\beta|_P \\
=&\frac{\pi}{4}-\frac{2}{\pi}\sum_{j=1}^{\infty}\frac{cos\big((2j-1)\mathbf{x}\big)cos\big((2j-1)\beta\big)}{(2j-1)^2}-\frac{2}{\pi}\sum_{j=1}^{\infty}\frac{sin\big((2j-1)\mathbf{x}\big)sin\big((2j-1)\beta\big)}{(2j-1)^2} \\
+&\frac{(\mathbf{x}-\beta)}{2}(2P-1)
\end{aligned}
\tag{3.3}
$$

Now let us introduce the sequence of Statistics:

$$
\begin{aligned}
\bar{C}_j(\mathbf{X}) &= \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} cos(j\mathbf{x}). \\
\bar{S}_j(\mathbf{X}) &= \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} sin(j\mathbf{x}). \\
\bar{\mathbf{X}} &= \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} \mathbf{x}.
\end{aligned}
\tag{3.4}
$$

Then, from 3.3 we have the expression,

$$
\begin{aligned}
&f_{P;\mathbf{X}}(\beta) \\
=&\frac{\pi}{4}-\frac{2}{\pi}\sum_{j=1}^{\infty}\frac{\bar{C}_{2j-1}(\mathbf{X})cos\big((2j-1)\beta\big)}{(2j-1)^2}-\frac{2}{\pi}\sum_{j=1}^{\infty}\frac{\bar{S}_{2j-1}(\mathbf{X})sin\big((2j-1)\beta\big)}{(2j-1)^2} \\
+&\frac{(\bar{\mathbf{X}}-\beta)}{2}(2P-1)
\end{aligned}
\tag{3.5}
$$

In EP-FQ algorithm, we approximate $f_{P;\mathbf{X}}(.)$ with $f^{J_0}_{P;\mathbf{X}}(.)$ for a large integer J_0, here

$$
\begin{aligned}
&f^{J_0}_{P;\mathbf{X}}(\beta) \\
=&\frac{\pi}{4}-\frac{2}{\pi}\sum_{j=1}^{J}\frac{\bar{C}_{2j-1}(\mathbf{X})cos\big((2j-1)\beta\big)}{(2j-1)^2}-\frac{2}{\pi}\sum_{j=1}^{J}\frac{\bar{S}_{2j-1}(\mathbf{X})sin\big((2j-1)\beta\big)}{(2j-1)^2} \\
+&\frac{(\bar{\mathbf{X}}-\beta)}{2}(2P-1)
\end{aligned}
\tag{3.6}
$$

J_0 should be chosen for desired accuracy. From example 2.3.9 we realize that $f^{J_0}_{P;\mathbf{X}}(.)$ is the best L^2 approximation to $f_{P;\mathbf{X}}(.)$, in-fact we have for any P or $\mathbf{X}$,

$|f^{J_0}_{P;\mathbf{X}}(\beta) - f_{P;\mathbf{X}}(\beta)| < \frac{2}{\pi}\sum_{j=J_0+1}^{\infty}\frac{1}{(2j-1)^2} = O(\frac{1}{J_0})$, which is tail sum of a convergent series. So we have uniform convergence of the sequence of functions $f^{J_0}_{P;\mathbf{X}}(\beta)$ to its limit function $f_{P;\mathbf{X}}(\beta)$ for any P or $\mathbf{X}$. This is a powerful property, implying, no matter how big or skewed the data is, we have a sequence of optimization criterion that approaches the actual optimization criterion at a fixed rate and we can pick an integer J_0, large enough to guarantee an error bound for the optimization criteria.

We can minimize this infinitely differentiable function $f^{J_0}_{P;\mathbf{X}}(\beta)$, or maximize the following equivalent function $\mho^{J_0}_{P;\mathbf{X}}(\beta)$to get the quantile estimate $\hat{Q}^{J_0}_P(\mathbf{X})$ for any P.

$$\mho^{J_0}_{P;\mathbf{X}}(\beta) = \frac{2}{\pi}\sum_{j=1}^{J}\frac{\bar{C}_{2j-1}(\mathbf{X})cos\big((2j-1)\beta\big) + \bar{S}_{2j-1}(\mathbf{X})sin\big((2j-1)\beta\big)}{(2j-1)^2} + \beta P \quad (3.7)$$

We can observe that $\bar{C}_j(\mathbf{X})$ and $\bar{S}_j(\mathbf{X})$ are EP statistics and they can be computed by Divide and Recombine, also this terms are division independent. After the D&R step, for each quantile P we have to solve an optimization problem to get corresponding $\hat{Q}^{J_0}_P(\mathbf{X})$.

3.2 Exact D&R method

In this section we describe D&R implementation of EP-FQ algorithm. Recall that Map-Reduce model for distributed large data-set stored in Hadoop Distributed File System(HDFS), is a simple and fast framework to implement general D&R algorithm, and we use this model to implement EP-FQ via R package datadr. In D&R step of EP-FQ we compute statistics $\bar{C}_j(\mathbf{X})$ and $\bar{S}_j(\mathbf{X})$ for $j = 1, \cdots, J_0$, J_0 is a large integer. Note that, $\bar{C}_j(\mathbf{X}) = \frac{C_j(\mathbf{X})}{N(\mathbf{X})}$, $\bar{S}_j(\mathbf{X}) = \frac{S_j(\mathbf{X})}{N(\mathbf{X})}$, where $C_j(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}}\cos(j\mathbf{x})$, $S_j(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}}\sin(j\mathbf{x})$ and $N(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}}1$. Observe that these are all Sum of Transformation(and hence SEP) statistics.

The division method[D] divides the data in subsets and stores in HDFS. In EP-FQ we compute EP statistics independent of the division method used, and the the D&R output we get in EP-FQ would be same for any division of the data. Data $\mathbf{X}$ is divided in R subsets $\mathbf{X}_1, \cdots, \mathbf{X}_R$. The observations are all real-valued and the data

structure in each subset are real numeric vectors. For the Map-Reduce model, initial Map inputs are R number of key-value pairs $\{1, \mathbf{X}_1\}, \cdots, \{R, \mathbf{X}_R\}$.

Afterwards, an analytic method[A] is applied to subset data, this part of D&R is Embarrassingly parallel, computation is done within the subsets and without any interaction between the subsets and carried out in Map stage of Map-Reduce model. In a general rth subset $\mathbf{X}_r$, for each observation $\mathbf{x}$ of $\mathbf{X}_r$, we compute J_0 Cosines: $\cos(\mathbf{x}), \cdots, \cos\big((2J_0-1)\mathbf{x}\big)$ and J_0 Sines: $\sin(\mathbf{x}),\cdots,\sin\big((2J_0-1)\mathbf{x}\big)$ and add up these terms to get $C_1(\mathbf{X_r}), \cdots, C_{2J-1}(\mathbf{X_r})$ and $S_1(\mathbf{X_r}),\cdots,S_{2J-1}(\mathbf{X_r})$. Also we count $N(\mathbf{X}_r)$, the number of observations in $\mathbf{X}_r$. These are values for Map output key-value pair, we assign the numbers $0, \cdots, 2J_0$ as keys. So, for the rth subset, the Map output or the intermediate key-value pairs of Map-Reduce model are $\{0, N(\mathbf{X}_r)\}$, $\{1, C_1(\mathbf{X}_r)\}$, $\{2, S_1(\mathbf{X}_r)\}$, $\cdots$,$\{2J_0 - 1, C_{2J_0-1}(\mathbf{X}_r)\}$, $\{2J_0, S_{2J_0-1}(\mathbf{X}_r)\}$.

Finally, for the recombination method[R], since $\mathbf{X}$ is a disjoint union of $\mathbf{X}_1$, $\cdots$, $\mathbf{X}_R$, we have for $j = 1, \cdots, J_0$, $C_j(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \cos(j\mathbf{x}) = \sum_{\mathbf{x}\in\cup_{r=1}^{R}\mathbf{X}_r} \cos(j\mathbf{x}) = \sum_{r=1}^{R}\sum_{\mathbf{x}\in\mathbf{X}_r} \cos(j\mathbf{x}) = \sum_{r=1}^{R} C_j(\mathbf{X}_r)$. Similarly, $S_j(\mathbf{X}) = \sum_{r=1}^{R} S_j(\mathbf{X}_r)$ and $N(\mathbf{X}) = \sum_{r=1}^{R} N(\mathbf{X}_r)$. So, in the Reduce stage of Map-Reduce model, for each key $0, \cdots, 2J_0$, we add corresponding values from the set of all intermediate key-value pairs. Final Map-Reduce output is the set of key-value pairs $\{0, N(\mathbf{X})\}$,$\{1, C_1(\mathbf{X})\}$, $\{2, S_1(\mathbf{X})\}$, $\cdots$,$\{2J_0 - 1, C_{2J_0-1}(\mathbf{X})\}$, $\{2J_0, S_{2J_0-1}(\mathbf{X})\}$.

We read the Map-Reduce output key-value pairs in the front-end machine to get our D&R output $N(\mathbf{X}), C_1(\mathbf{X}), S_1(\mathbf{X}), \cdots, C_{2J_0-1}(\mathbf{X})\}, S_{2J_0-1}(\mathbf{X})\}$. After the D&R step, for each quantile p we maximize the optimization criteria in equation 3.7 to get corresponding $\hat{Q}_p^{J_0}(\mathbf{X})$.

3.3 Time-complexity of EP-FQ: Comparison with Binning

Let $N = N(\mathbf{X})$, the sample size of data $\mathbf{X}$. The exact computation of sample quantiles requires sorting entire data, an efficient sorting algorithm has $O(N \log N)$ time complexity, and can't be carried out in parallel. EP-FQ is serial $O(N)$ algorithm,

as in EP-FQ we compute Sum of Transformations. The only instance, we require interaction among observations, is when we take the the sums, which makes parallel Map-Reduce implementation feasible.

Binning method is another fast, accurate $O(N)$ algorithm to compute approximate quantiles and it can be implemented in parallel by D&R with Map-Reduce. We look at the range of data and divide it in a number of bins. Suppose we have $nBins$ bins $(b_0, b_1]$, $(b_1, b_2]$, $\cdots$, $(b_{nBins-1}, b_{nBins})$. We tabulate the number of observations in each of these bins, i.e. for each $r = 1, \cdots, nBins$, we compute SEP statistic $B_r(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} I(b_{r-1} < \mathbf{x} \leq b_r)$. We get cumulative frequency counts at the bin-points, then we approximate quantile values by interpolation.

We need to know whether EP-FQ beats Binning in run-time with comparable accuracy. The post D&R steps in both of these methods(serial optimization for EP-FQ and cumulative frequency interpolation for Binning) take fixed amount of run-time, independent of the sample size $N = N(\mathbf{X})$. So we should focus on run-time of D&R steps to compare time-complexity.

Note in D&R step, both these algorithms involve serial computation of a number of SEP Sum of Transformations. For comparable accuracy, we have $2J_0 + 1 < nBins$. EP-FQ beats Binning in Reduce step because in EP-FQ, there will be less number of intermediate key-value pairs to sum up in Reduce step than Binning.

So, we need to compare average Map-time of EP-FQ and average Map-time in Binning. To compare average Map-time we introduce concept of *in memory* computation. Suppose we assume that the data $\mathbf{X}$ is small enough to fit in memory of a CPU processor. Let, $\tilde{t}_F(\mathbf{X})$ = Run-time to compute SEP statistics $N(\mathbf{X})$, $C_{2j-1}(\mathbf{X})$ and $S_{2j-1}(\mathbf{X})$, for $j = 1, \cdots, J_0$ and $\tilde{t}_B(\mathbf{X})$ = Run-time to compute all $nBins$ SEP bin-counts $B_r(\mathbf{X})$ for $r = 1, \cdots, nBins$ in the processor. Now, given data $\mathbf{X}$, $\tilde{t}_F(\mathbf{X})$ and $\tilde{t}_B(\mathbf{X})$ are random variables, and we can stochastic-ally compare them to judge which algorithm has faster average Map-time. We will compare in R and the data and parameters we are going to use is as follows:

```
#Input-Data
x.raw <- sample(1:10^7,10^7)                                    # x.raw is the data, for
                                                                # runtime comparison we
                                                                # use the same standerd
                                                                # data x.raw: a random
                                                                # premutation of numbers
                                                                # (1,2,...,10^7).

x.range <- range(x.raw)                                         # x.raw is bounded by the
                                                                # numbers x.range[1] and
                                                                # x.range[2] (for general
                                                                # data, if it's not given
                                                                # we get x.range by a
                                                                # previous D&R step).

#Binning-Variables

nBins <- 10000                                                  # Number of bins for the
                                                                # Binning method.

delta <- diff(x.range) / (nBins - 1)                            # Fixed bin width.

cuts <- seq(x.range[1] - delta / 2, x.range[2] + delta / 2, by = delta)   # Bin boundaries.

#EP.FQ-Variables

J0 <- 100                                                       # Number of terms for
                                                                # EP.FQ.

A <- 2/(x.range[2]-x.range[1])
B <- -(x.range[2]+x.range[1])/(x.range[2]-x.range[1])           # A & B are numbers s.t.
                                                                # -1 < A*(x.raw) + B < 1.

#Final-Data for EP.FQ

X <- A*x.raw+B
```

For this $\mathbf{X}$, Binning with $nBins = 10,000$ and EP-FQ with $J_0 = 100$ terms provides comparable accuracy, if we decrease J_0, EP-FQ becomes less accurate. In the R package *datadr*, there is an implementation of Binning method in the function *drQuantile()*. In the R package *drEps*, there is an implementation of EP-FQ in the function *drQ()*. To make our comparison, we are going to use Map-expressions from scripts of these functions. See the R function *map.bin()* in Appendix C for Binning Map implementation and the R function *map.EP-FQ0()* in Appendix C for EP-FQ Map implementation. All the SEP statistics are serially computed in *C++* for loop, which is fast. Here is the run-time for these two methods in one of the run for data $\mathbf{X}$, in a *Intel Core i7CPU 2.67GHz* processor

```
> start.time <- proc.time()
> lsb <- map.bin(x.raw)
> end.time <- proc.time()
> end.time-start.time
user  system elapsed
5.092   0.404   5.496
>
> start.time <- proc.time()
> lsf0 <- map.EP.FQ0(X)
> end.time <- proc.time()
> end.time-start.time
user  system elapsed
61.324   0.000  61.329
>
```

We experiment with the same data $\mathbf{X}$ and parameters for 1000 different runs and observed that, $Mean(\frac{\tilde{t}_F(\mathbf{X})}{\tilde{t}_B(\mathbf{X})}) > 10$. So, we can say that Binning beats EP-FQ in memory. We can justify heuristically, Binning uses binary search to assign each $\mathbf{x}$ to one of the bins, if there are $nBins$ bins then average assignment takes $O(\log nBins)$ operations for each $\mathbf{x}$, where as EP-FQ requires computation of J_0 Cosines and J_0 Sines for each $\mathbf{x}$, requiring $O(J_0)$ operations. We will try to modify EP-FQ in next few sections to have comparable run-time at the Map stage.

3.4 Chebyshev's Transformation and the EP-FQCh algorithm

We know, for a large integer J_0, computation of $\bar{C}_j(\mathbf{X})$ and $\bar{S}_j(\mathbf{X})$ for $j = 1, 2, \cdots, J_0$ by D&R is very time consuming, as we need to perform J_0 Cosine and J_0 Sine operations for each $\mathbf{x}$. We propose a faster alternative in this section, but first, we need the following identities:

$$\begin{aligned} \cos\big((2j-1)\mathbf{x}\big) &= T_{2j-1}\big(\cos(\mathbf{x})\big) \\ \sin\big((2j-1)\mathbf{x}\big) &= (-1)^j T_{2j-1}\big(\sin(\mathbf{x})\big) \end{aligned} \tag{3.8}$$

Here $T_j(t)$ is the jth order Chebyshev's polynomial. In mathematics the Chebyshev polynomials [2], named after Pafnuty Chebyshev, are a sequence of orthogonal polynomials which can be defined recursively. The Chebyshev polynomials of the first kind are defined by the recurrence relation $T_0(t) = 1$; $T_1(t) = t$; and $T_{j+1}(t) =$

$2tT_j(t)-T_{j-1}(t)$ for $j>0$. Any odd degree Chebyshev's polynomial $T_{2j-1}(t)$ is a linear combination of t, t^3, $\cdots$, t^{2j-1}. For example: $T_3(t)=4t^3-3t; T_5(t)=16t^5-20t^3+5$ etc. From 3.8, we realize that, $\sum_{\mathbf{x}\in\mathbf{X}}\cos\big((2j-1)\mathbf{x}\big)$ can be written as a linear combination of $\sum_{\mathbf{x}\in\mathbf{X}}\cos(\mathbf{x})$, $\sum_{\mathbf{x}\in\mathbf{X}}\cos^3(\mathbf{x})$, $\cdots$, $\sum_{\mathbf{x}\in\mathbf{X}}\cos^{2j-1}(\mathbf{x})$. Equivalently, $\bar{C}_{2j-1}(\mathbf{X})$ can be written as a linear combination of $\bar{C}^1(\mathbf{X})$,$\bar{C}^3(\mathbf{X})$, $\cdots$, $\bar{C}^{2j-1}(\mathbf{X})$ if we define:

$$\bar{C}^j(\mathbf{X})=\frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\cos^j(\mathbf{x}) \tag{3.9}$$

For the Sine terms observe that:

$$\sin^{2j-1}(\mathbf{x})=\sin(\mathbf{x})-\sum_{j'=1}^{j-1}\sin(\mathbf{x})\cos^{2j'}(\mathbf{x}); \forall j>1 \tag{3.10}$$

So from 3.8 and 3.9, we realize that, $\sum_{\mathbf{x}\in\mathbf{X}}\sin\big((2j-1)\mathbf{x}\big)$ can also be written as a linear combination of $\sum_{\mathbf{x}\in\mathbf{X}}\sin(\mathbf{x})$, $\sum_{\mathbf{x}\in\mathbf{X}}\sin(\mathbf{x})\cos^2(\mathbf{x})$, $\cdots$, $\sum_{\mathbf{x}\in\mathbf{X}}\sin(\mathbf{x})\cos^{2j-2}(\mathbf{x})$. Equivalently, $\bar{S}_{2j-1}(\mathbf{X})$ can be written as a linear combination of $\bar{S}_1(\mathbf{X})$, $\overline{SC}^2(\mathbf{X})$, $\cdots$, $\overline{SC}^{2j-2}(\mathbf{X})$ if we define:

$$\overline{SC}^j(\mathbf{X})=\frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\sin(\mathbf{x})\cos^j(\mathbf{x}) \tag{3.11}$$

D&R step:

In the Fourier-Chebyshev method we propose to compute $2J_0$ number of SEP statistics: $\bar{C}^1(\mathbf{X})$, $\bar{C}^3(\mathbf{X})$, $\cdots$, $\bar{C}^{2J-1}(\mathbf{X})$ and $\bar{S}_1(\mathbf{X})$, $\overline{SC}^2(\mathbf{X})$, $\cdots$, $\overline{SC}^{2J-2}(\mathbf{X})$ by a single D&R step. This is fast because, for each observation $\mathbf{x}$, as opposed to J_0 Cosine and J_0 Sine operations, we need to perform only one Cosine operation (which is $\cos(\mathbf{x})$), we can get $\sin(\mathbf{x})$ from $\cos(\mathbf{x})$ as, $\sin(\mathbf{x})=\sqrt{1-\cos^2(\mathbf{x})}$ for $\mathbf{x}\in(-1,1)$, and then, we multiply both $\cos(\mathbf{x})$ and $\sin(\mathbf{x})$, by the same number $\cos^2(\mathbf{x})$, $J-1$ times in a loop to get summands for all SEP statistics. This loop multiplication is extremely fast, see the R function *map.EP-FQCh1()* in Appendix C for a map implementation of the Fourier-Chebyshev method.

Fourier-Chebyshev method is much faster than the previously discussed Fourier method. For comparison again we pick the same $\mathbf{X}$, keep $nBins = 10,000$, but increase J_0 from 100 terms to 300 terms.

Here is the run-time for these two methods:

```
> start.time <- proc.time()
> lsf1 <- map.EP.FQCh1(X)
> end.time <- proc.time()
> end.time-start.time
user  system elapsed
5.448   0.164   5.630
>
```

So that we know EP-FQCh is faster and it also beats Binning in run-time.

Now lets find out how to efficiently compute $\bar{C}_1(\mathbf{X})$, $\bar{C}_3(\mathbf{X})$, $\cdots$, $\bar{C}_{2J-1}(\mathbf{X})$ and $\bar{S}_1(\mathbf{X})$, $\bar{S}_3(\mathbf{X})$, $\cdots$, $\bar{S}_{2J-1}(\mathbf{X})$ from the D&R output $\bar{C}^1(\mathbf{X})$, $\bar{C}^3(\mathbf{X})$, $\cdots$, $\bar{C}^{2J-1}(\mathbf{X})$ and $\bar{S}_1(\mathbf{X})$, $\overline{SC}^2(\mathbf{X})$, $\cdots$, $\overline{SC}^{2J-2}(\mathbf{X})$. We will call this linear transformation procedure EP-FQCh recursion of type A, we'll see it is actually a recursive procedure.

We can derive two set of equations from the recursive relation for Chebyshev's polynomials $T_{j+1}(t) = 2tT_j(t) - T_{j-1}(t)$:

$$\begin{aligned} t^{2m}T_{2j+1}(t) &= 2t^{2m+1}T_{2j}(t) - t^{2m}T_{2j-1}(t) \\ t^{2m+1}T_{2j}(t) &= 2t^{2m+2}T_{2j-1}(t) - t^{2m+1}T_{2j-2}(t) \end{aligned} \tag{3.12}$$

Recursion for Cosine terms:

Let us define: $\gamma_j^m(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \cos^m(\mathbf{x})\ T_j\big(\cos(\mathbf{x})\big)$ for $m, j \in \mathbb{N}$. Observe that, by assumption of Chebyshev polynomials, $T_0\big(\cos(\mathbf{x})\big) = 1$ and $T_1(\cos(\mathbf{x})) = \cos(\mathbf{x})$, then, we have, $\gamma_0^m(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \cos^m(\mathbf{x}) = \bar{C}^m(\mathbf{X})$ and $\gamma_1^{m-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \cos^m(\mathbf{x}) = \bar{C}^m(\mathbf{X})$. From the output of the D&R step, we already have: $\gamma_0^{2j-1}(\mathbf{X})$ and $\gamma_1^{2j-2}(\mathbf{X})$ available to us for $j = 1, \cdots, J_0$. We collect $\bar{C}_1(\mathbf{X}) = \gamma_1^0(\mathbf{X})$ and write these statistics as 2 rows of the matrix:

$$\Gamma_0^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_0^1(\mathbf{X}) & \gamma_0^3(\mathbf{X}) & \cdots & \gamma_0^{2J-1}(\mathbf{X}) \\ \gamma_1^0(\mathbf{X}) & \gamma_1^2(\mathbf{X}) & \cdots & \gamma_1^{2J-2}(\mathbf{X}) \end{bmatrix}$$

Now in 3.12, if we replace t with $\cos(\mathbf{x})$ and take a sum over all $\mathbf{x}$, we get:

$$\begin{aligned} \gamma_{2j+1}^{2m}(\mathbf{X}) &= 2\gamma_{2j}^{2m+1}(\mathbf{X}) - \gamma_{2j-1}^{2m}(\mathbf{X}) \\ \gamma_{2j}^{2m+1}(\mathbf{X}) &= 2\gamma_{2j-1}^{2m+2}(\mathbf{X}) - \gamma_{2j-2}^{2m+1}(\mathbf{X}) \end{aligned} \tag{3.13}$$

these two equations form the basic building relations for an recursive process that we are going to call EP-FQCh type A recursion.

In the first step of EP-FQCh type A recursion, we compute $\gamma_2^1(\mathbf{X})$, $\gamma_2^3(\mathbf{X})$, $\cdots$, $\gamma_2^{2J-3}(\mathbf{X})$ in terms of $\gamma_1^2(\mathbf{X})$, $\gamma_1^4(\mathbf{X})$, $\cdots$, $\gamma_1^{2J-2}(\mathbf{X})$ and $\gamma_0^1(\mathbf{X})$, $\gamma_0^3(\mathbf{X})$, $\cdots$, $\gamma_0^{2J-3}(\mathbf{X})$, using the $2nd$ equation of 3.13. In terms of the matrix $\Gamma_0^{(A)}(\mathbf{X})$, we shift each element in 2nd row to left and apply the equation to columns $2, \cdots, J_0$ to get new set of statistics as follows:

$$\begin{bmatrix} & \gamma_0^1(\mathbf{X}) & \gamma_0^3(\mathbf{X}) & \cdots & \gamma_0^{2J-3}(\mathbf{X}) & \gamma_0^{2J-1}(\mathbf{X}) \\ \gamma_1^0(\mathbf{X}) & \gamma_1^2(\mathbf{X}) & \gamma_1^4(\mathbf{X}) & \cdots & \gamma_1^{2J-2}(\mathbf{X}) & \\ & \downarrow & \downarrow & \cdots & \downarrow & \\ & \gamma_2^1(\mathbf{X}) & \gamma_2^3(\mathbf{X}) & \cdots & \gamma_2^{2J-3}(\mathbf{X}) & \end{bmatrix}$$

Then, we discard the first row of $\Gamma_0^{(A)}(\mathbf{X})$ and add these new set of statistics as the new row to get the matrix:

$$\Gamma_1^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_1^0(\mathbf{X}) & \gamma_1^2(\mathbf{X}) & \cdots & \gamma_1^{2J-4}(\mathbf{X}) & \gamma_1^{2J-2}(\mathbf{X}) \\ \gamma_2^1(\mathbf{X}) & \gamma_2^3(\mathbf{X}) & \cdots & \gamma_2^{2J-3}(\mathbf{X}) & \end{bmatrix}$$

First, observe that the 2nd row of $\Gamma_1^{(A)}(\mathbf{X})$ is incomplete as its J_0th element is missing. In the second step of EP-FQCh type A recursion, we compute $\gamma_3^0(\mathbf{X})$, $\gamma_3^2(\mathbf{X})$, $\cdots$, $\gamma_3^{2J-4}(\mathbf{X})$ in terms of $\gamma_2^1(\mathbf{X})$, $\gamma_2^3(\mathbf{X})$, $\cdots$, $\gamma_2^{2J-3}(\mathbf{X})$ and $\gamma_1^0(\mathbf{X})$, $\gamma_1^2(\mathbf{X})$, $\cdots$, $\gamma_1^{2J-4}(\mathbf{X})$, using the $1st$ equation of 3.13. In terms of the matrix $\Gamma_1^{(A)}(\mathbf{X})$, we apply the equation to columns $1, \cdots, J-1$ to get new set of statistics as follows:

$$\begin{bmatrix} \gamma_1^0(\mathbf{X}) & \gamma_1^2(\mathbf{X}) & \cdots & \gamma_1^{2J-4}(\mathbf{X}) & \gamma_1^{2J-2}(\mathbf{X}) \\ \gamma_2^1(\mathbf{X}) & \gamma_2^3(\mathbf{X}) & \cdots & \gamma_2^{2J-3}(\mathbf{X}) & \\ \downarrow & \downarrow & \cdots & \downarrow & \\ \gamma_3^0(\mathbf{X}) & \gamma_3^2(\mathbf{X}) & \cdots & \gamma_3^{2J-4}(\mathbf{X}) & \end{bmatrix}$$

After second step we collect $\bar{C}_3(\mathbf{X}) = \gamma_3^0(\mathbf{X})$. We again discard the first row of $\Gamma_1^{(A)}(\mathbf{X})$ and add these new elements as the new row to get the matrix:

$$\Gamma_2^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_2^1(\mathbf{X}) & \gamma_2^3(\mathbf{X}) & \cdots & \gamma_2^{2J-3}(\mathbf{X}) \\ \gamma_3^0(\mathbf{X}) & \gamma_3^2(\mathbf{X}) & \cdots & \gamma_3^{2J-4}(\mathbf{X}) \end{bmatrix}$$

These two steps make one recursion of EP-FQCh type A recursion.We now describe a general recursion. At the end of the $j-1$th recursion we have the matrix:

$$\Gamma_{2j-2}^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_{2j-2}^1(\mathbf{X}) & \gamma_{2j-2}^3(\mathbf{X}) & \cdots & \gamma_{2j-2}^{2J-2j+1}(\mathbf{X}) \\ \gamma_{2j-1}^0(\mathbf{X}) & \gamma_{2j-1}^2(\mathbf{X}) & \cdots & \gamma_{2j-1}^{2J-2j}(\mathbf{X}) \end{bmatrix}$$

In the first step of a general jth recursion, we compute $\gamma_{2j}^1(\mathbf{X})$, $\gamma_{2j}^3(\mathbf{X})$, $\cdots$, $\gamma_{2j}^{2J-2j-3}(\mathbf{X})$ in terms of $\gamma_{2j-1}^0(\mathbf{X})$, $\gamma_{2j-1}^2(\mathbf{X})$, $\cdots$, $\gamma_{2j-1}^{2J-2j}(\mathbf{X})$ and $\gamma_{2j-2}^1(\mathbf{X})$, $\gamma_{2j-2}^3(\mathbf{X})$, $\cdots$, $\gamma_{2j-2}^{2J-2j-1}(\mathbf{X})$, using the $2nd$ equation of 3.13 as follows:

$$\begin{bmatrix} & \gamma_{2j-2}^1(\mathbf{X}) & \gamma_{2j-2}^3(\mathbf{X}) & \cdots & \gamma_{2j-2}^{2J-2j-1}(\mathbf{X}) & \gamma_{2j-2}^{2J-2j+1}(\mathbf{X}) \\ \gamma_{2j-1}^0(\mathbf{X}) & \gamma_{2j-1}^2(\mathbf{X}) & \gamma_{2j-1}^4(\mathbf{X}) & \cdots & \gamma_{2j-1}^{2J-2j}(\mathbf{X}) & \\ & \downarrow & \downarrow & \cdots & \downarrow & \\ & \gamma_{2j}^1(\mathbf{X}) & \gamma_{2j}^3(\mathbf{X}) & \cdots & \gamma_{2j}^{2J-2j-1}(\mathbf{X}) & \end{bmatrix}$$

Again, we discard the first row of $\Gamma_{2j}^{(A)}(\mathbf{X})$ and add these new elements as the new row to get the matrix:

$$\Gamma_{2j-1}^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_{2j-1}^0(\mathbf{X}) & \gamma_{2j-1}^2(\mathbf{X}) & \cdots & \gamma_{2j-1}^{2J-2j-2}(\mathbf{X}) & \gamma_{2j-1}^{2J-2j}(\mathbf{X}) \\ \gamma_{2j}^1(\mathbf{X}) & \gamma_{2j}^3(\mathbf{X}) & \cdots & \gamma_{2j}^{2J-2j-1}(\mathbf{X}) & \end{bmatrix}$$

And like before, in the second step of jth recursion, we compute $\gamma^0_{2j+1}(\mathbf{X})$, $\gamma^2_{2j+1}(\mathbf{X})$, $\cdots$, $\gamma^{2J-2j-2}_{2j+1}(\mathbf{X})$ in terms of $\gamma^1_{2j}(\mathbf{X})$, $\gamma^3_{2j}(\mathbf{X})$, $\cdots$, $\gamma^{2J-2j-1}_{2j}(\mathbf{X})$ and $\gamma^0_{2j-1}(\mathbf{X})$, $\gamma^2_{2j-1}(\mathbf{X})$, $\cdots$, $\gamma^{2J-2j-2}_{2j-1}(\mathbf{X})$, using the $1st$ equation of 3.13 as follows:

$$\begin{bmatrix} \gamma^0_{2j-1}(\mathbf{X}) & \gamma^2_{2j-1}(\mathbf{X}) & \cdots & \gamma^{2J-2j-2}_{2j-1}(\mathbf{X}) & \gamma^{2J-2j}_{2j-1}(\mathbf{X}) \\ \gamma^1_{2j}(\mathbf{X}) & \gamma^3_{2j}(\mathbf{X}) & \cdots & \gamma^{2J-2j-1}_{2j}(\mathbf{X}) & \\ \downarrow & \downarrow & \cdots & \downarrow & \\ \gamma^0_{2j+1}(\mathbf{X}) & \gamma^2_{2j+1}(\mathbf{X}) & \cdots & \gamma^{2J-2j-2}_{2j+1}(\mathbf{X}) & \end{bmatrix}$$

After second step we collect $\bar{C}_{2j+1}(\mathbf{X}) = \gamma^0_{2j+1}(\mathbf{X})$. We again discard the first row of $\Gamma^{(A)}_{2j-1}(\mathbf{X})$ and add these new elements as the new row to get the matrix:

$$\Gamma^{(A)}_{2j}(\mathbf{X}) = \begin{bmatrix} \gamma^1_{2j}(\mathbf{X}) & \gamma^3_{2j}(\mathbf{X}) & \cdots & \gamma^{2J-2j-1}_{2j}(\mathbf{X}) \\ \gamma^0_{2j+1}(\mathbf{X}) & \gamma^2_{2j+1}(\mathbf{X}) & \cdots & \gamma^{2J-2j-2}_{2j+1}(\mathbf{X}) \end{bmatrix}$$

If we repeat this recursion $J-1$ times we will get $\bar{C}_1(\mathbf{X})$, $\bar{C}_3(\mathbf{X})$, $\cdots$, $\bar{C}_{2J-1}(\mathbf{X})$ at the end. See Appendix B.6 for an algorithmic flow-chart for EP-FQCh type A recursion.

Recursion for Sine terms:

Similarly, for Sine terms, we can also define: $\delta^m_j(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \sin^m(\mathbf{x}) T_j\big(\sin(\mathbf{x})\big)$ for $m, j \in \mathbb{N}$. Observe that, unlike the Cosine terms, we don't directly get $\delta^3_0(\mathbf{X})$, $\cdots$, $\delta^{2J-1}_0(\mathbf{X})$ and $\delta^2_1(\mathbf{X})$, $\delta^4_1(\mathbf{X})$, $\cdots$, $\delta^{2J-2}_1(\mathbf{X})$ from the output of D&R step.

Now recall equation 3.10, if we take a summation over all $\mathbf{x} \in \mathbf{X}$, we get:

$$\delta^{2j+1}_0(\mathbf{X}) = \delta^{2j}_1(\mathbf{X}) = \bar{S}_1(\mathbf{X}) - \sum_{k=1}^{k} \overline{SC}^{2l}(\mathbf{X}). \tag{3.14}$$

Using equation 3.14, we can compute $\delta^0_3(\mathbf{X}), \cdots, \delta^0_{2J-1}(\mathbf{X})$ and $\delta^1_2(\mathbf{X}), \cdots, \delta^1_{2J-2}(\mathbf{X})$ by a series of recursive steps.

First, note that for the Sine terms also, we can have recursive relations like 3.13:

$$\begin{aligned}\delta_{2j+1}^{2m}(\mathbf{X}) &= 2\delta_{2j}^{2m+1}(\mathbf{X}) - \delta_{2j-1}^{2m}(\mathbf{X}) \\ \delta_{2j}^{2m+1}(\mathbf{X}) &= 2\delta_{2j-1}^{2m+2}(\mathbf{X}) - \delta_{2j-2}^{2m+1}(\mathbf{X})\end{aligned} \tag{3.15}$$

So, if we start with the matrix

$$\bar{\mathbb{S}}_0(\mathbf{X}) = \begin{bmatrix} \delta_0^1(\mathbf{X}) & \delta_0^3(\mathbf{X}) & \cdots & \delta_0^{2J-1}(\mathbf{X}) \\ \delta_1^0(\mathbf{X}) & \delta_1^2(\mathbf{X}) & \cdots & \delta_1^{2J-2}(\mathbf{X}) \end{bmatrix}$$

and apply EP-FQCh type A recursion $J-1$ times, at the end, we will get the set of statistics $\delta_1^0(\mathbf{X})$, $\delta_3^0(\mathbf{X})$, $\cdots$, $\delta_{2J-1}^0(\mathbf{X})$.

A general jth recursive step would look like:

First step:

$$\begin{bmatrix} & \delta_{2j-2}^1(\mathbf{X}) & \delta_{2j-2}^3(\mathbf{X}) & \cdots & \delta_{2j-2}^{2J-2j-1}(\mathbf{X}) & \delta_{2j-2}^{2J-2j+1}(\mathbf{X}) \\ \delta_{2j-1}^0(\mathbf{X}) & \delta_{2j-1}^2(\mathbf{X}) & \delta_{2j-1}^4(\mathbf{X}) & \cdots & \delta_{2j-1}^{2J-2j}(\mathbf{X}) & \\ & \downarrow & \downarrow & \cdots & \downarrow & \\ & \delta_{2j}^1(\mathbf{X}) & \delta_{2j}^3(\mathbf{X}) & \cdots & \delta_{2j}^{2J-2j-1}(\mathbf{X}) & \end{bmatrix}$$

Second Step:

$$\begin{bmatrix} \delta_{2j-1}^0(\mathbf{X}) & \delta_{2j-1}^2(\mathbf{X}) & \cdots & \delta_{2j-1}^{2J-2j-2}(\mathbf{X}) & \delta_{2j-1}^{2J-2j}(\mathbf{X}) \\ \delta_{2j}^1(\mathbf{X}) & \delta_{2j}^3(\mathbf{X}) & \cdots & \delta_{2j}^{2J-2j-1}(\mathbf{X}) & \\ \downarrow & \downarrow & \cdots & \downarrow & \\ \delta_{2j+1}^0(\mathbf{X}) & \delta_{2j+1}^2(\mathbf{X}) & \cdots & \delta_{2j+1}^{2J-2j-2}(\mathbf{X}) & \end{bmatrix}$$

at the end of second step of the jth iteration, we collect $\bar{S}_{2j+1}(\mathbf{X}) = (-1)^j \delta_{2j+1}^0(\mathbf{X})$. So, after $J-1$ iterations we will get $\bar{S}_1(\mathbf{X})$, $\bar{S}_3(\mathbf{X})$, $\cdots$, $\bar{S}_{2J-1}(\mathbf{X})$.

After the D&R step, for each quantile p we maximize the optimization criteria in equation 3.7 to get corresponding $\hat{Q}_P^{J_0}(\mathbf{X})$. For an algorithmic flow chart, see Appendix B.3.

Limitations:

EP-FQCh algorithm should not be used to calculate quantiles for large data, because it is extremely inaccurate, and, in this section, we will explain why. A processor is limited in precision, whenever try to compute Cosine of any $\mathbf{x}$ using a programming language, we round it to certain decimal places. So, for most values of $\mathbf{x}$, instead of $\cos(\mathbf{x})$, we actually get a round-off value $\cos(\mathbf{x}) + \varepsilon\big(\cos(\mathbf{x})\big)$. When we compute $\bar{C}^{2j-1}(\mathbf{X})$ by D&R, we make an error(say $\varepsilon\big(\bar{C}^{2j-1}(\mathbf{X})\big)$) to get $\bar{C}^{2j-1}(\mathbf{X}) + \varepsilon\big(\bar{C}^{2j-1}(\mathbf{X})\big)$ as the D&R output.

Chebyshev's polynomials have the explicit expression for odd index $2j-1$

$$T_{2j-1}(\mathbf{x}) = \frac{2j-1}{2}\sum_{j'=0}^{j-1}(-1)^{j'}\frac{(2j-j'-2)!}{(j')!(2j-2j'-1)!}(2\mathbf{x})^{2j-2j'-1} \tag{3.16}$$

If we replace $\mathbf{x}$ with $\cos(\mathbf{x})$, and then take a sum over $\mathbf{x} \in \mathbf{X}$, we get

$$\bar{C}_{2j-1}(\mathbf{X}) = \sum_{j'=0}^{j-1}\Big((-1)^{j'}2^{2j-2j'-2}\frac{(2j-1)(2j-j'-2)!}{(j')!(2j-2j'-1)!}\Big)\bar{C}^{2j-2j'-1}(\mathbf{X}) \tag{3.17}$$

We use EP-FQCh type A iteration instead of directly using 3.17 to get $\bar{C}_{2j-1}(\mathbf{X})$, because transformation is fast, as it requires less operations. But from this expression we realize that, we make error $\varepsilon\big(\bar{C}_{2j-1}(\mathbf{X})\big)$ while computing $\bar{C}_{2j-1}(\mathbf{X})$ (in EP-FQCh we don't compute $\bar{C}_{2j-1}(\mathbf{X})$ directly), here

$$\varepsilon\big(\bar{C}_{2j-1}(\mathbf{X})\big) = \sum_{j'=0}^{j-1}\Big((-1)^{j'}2^{2j-2j'-2}\frac{(2j-1)(2j-j'-2)!}{(j')!(2j-2j'-1)!}\Big)\varepsilon\big(\bar{C}^{2j-2j'-1}(\mathbf{X})\big) \tag{3.18}$$

Even though $\varepsilon\big(\bar{C}^{2j-1}(\mathbf{X})\big)$ is small for each j, the weighted error $\varepsilon\big(\bar{C}_{2j-1}(\mathbf{X})\big)$ is not small, in fact it grows exponentially with j. The error $\varepsilon\big(\bar{S}_{2j-1}(\mathbf{X})\big)$ for Sine term $\bar{S}_{2j-1}(\mathbf{X})$ also increases exponentially.

Here is plot to visualize how the average error increases as we increase the index j in EP-FQCh1 method for the test-data we introduced in section 3.2.

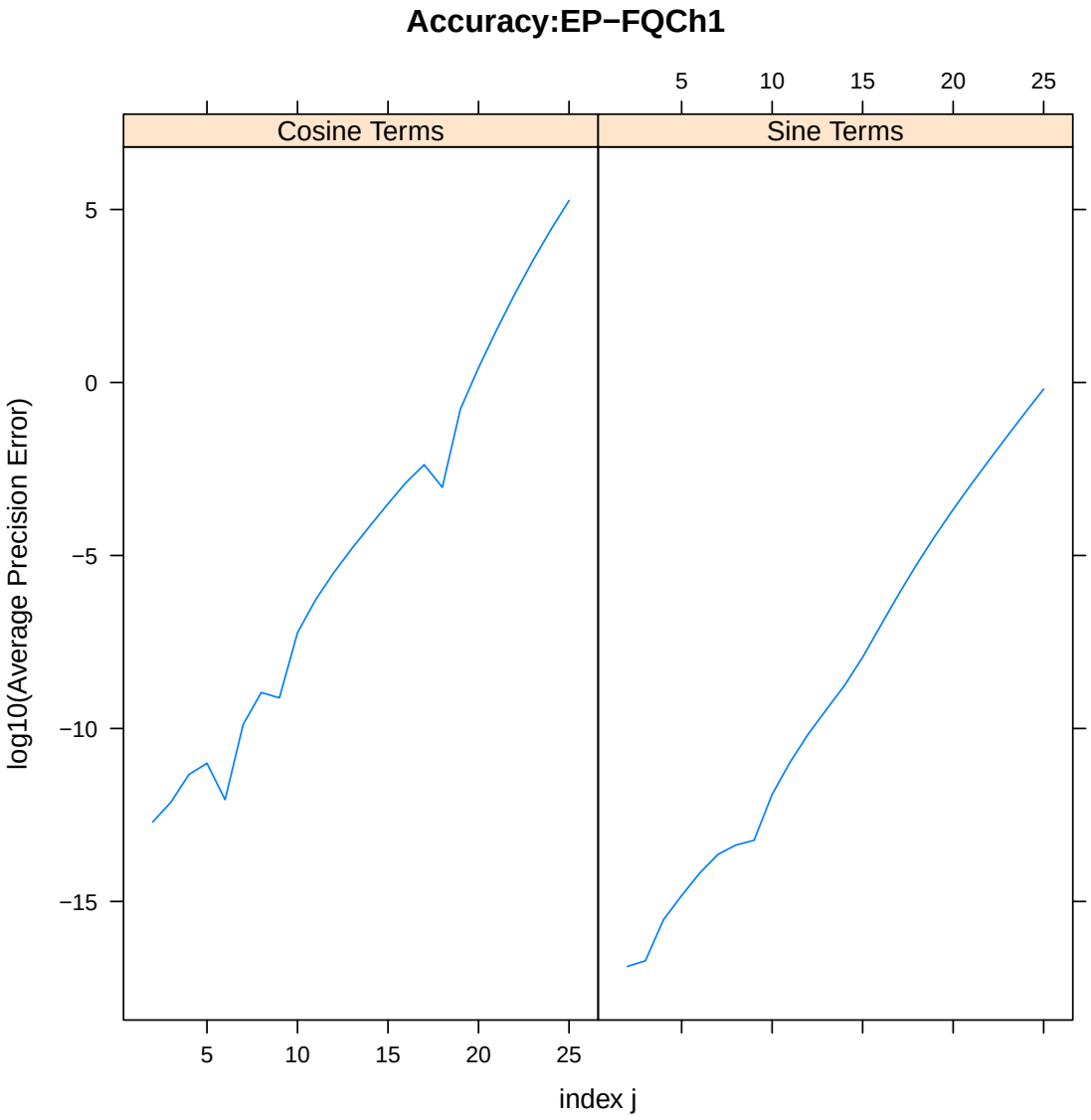

Figure 3.1. Accuracy of Cosine/Sine statistics: EP-FQCh1

The Average Precision Error is $\varepsilon\big(\bar{C}_{2j-1}(\mathbf{X})\big)$ and $\varepsilon\big(\bar{S}_{2j-1}(\mathbf{X})\big)$ for jth Cosine and Sine term respectively. From this plots but we can say that for this $\mathbf{X}$, due to exponential increasing nature of the error terms, after 15 terms error becomes too large and it does not makes sense to perform EP-FQCh1 method for good accuracy. As a rule of thumb we observed that, $\log(\varepsilon\big(\bar{C}_{2j-1}(\mathbf{X})\big)) \propto j$. With only 15 terms

we do not get convergence for Fourier series and the approximate quantiles becomes numerically inaccurate.

3.5 2-way Fourier-Chebyshev method: EP-FQCh2 algorithm

Now, we know that regular EP-FQCh algorithm is not so accurate, suppose instead of just one Cosine operation for each observation $\mathbf{x}$, we allow two Cosine operations for each $\mathbf{x}$. Remember, our goal is to compute $\bar{C}_1(\mathbf{X})$, $\bar{C}_3(\mathbf{X})$, $\cdots$, $\bar{C}_{2J_0-1}(\mathbf{X})$ and $\bar{S}_1(\mathbf{X})$, $\bar{S}_3(\mathbf{X})$, $\cdots$, $\bar{S}_{2J_0-1}(\mathbf{X})$ fast and accurately for a large number J_0. Let us first fix integers J and K, such that $J_0 = J \times K$.

In this section, we will introduce EP-FQCh2 algorithm which is a 2-way modification of EP-FQCh algorithm. We will see, if we allow two Cosine operations for each observation $\mathbf{x}$, then, we can perform fast and accurate D&R computation of all the $2JK$ Fourier coefficients. First, we define 2-way generalizations of the statistics introduced in last section. For $\{j,k\} \in \mathbb{N}^2$, define:

$$\begin{aligned}
\bar{C}^{j,k}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\cos^j(\mathbf{x}).\cos^k(2J\mathbf{x}) \Big) \\
\overline{SC}^{j,k}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(sin(\mathbf{x}).\cos^j(\mathbf{x}).\cos^k(2J\mathbf{x}) \Big) \\
\bar{S}^{j,k}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(sin^j(\mathbf{x}).\cos^k(2J\mathbf{x}) \Big)
\end{aligned} \tag{3.19}$$

D&R step:

In Ep-FQCh2 algorithm, we compute two $J \times K$ matrices of SEP statistics: $\bar{\mathbf{C}}^{J,K}(\mathbf{X}) = \{\{\bar{C}^{2j-1,k-1}(\mathbf{X})\}\}$ and $\overline{\mathbf{SC}}^{J,K}(\mathbf{X}) = \{\{\overline{SC}^{2j-2,k-1}(\mathbf{X})\}\}$ in a single D&R step. We will show in this section, that, we can get $\bar{C}_1(\mathbf{X})$,$\bar{C}_3(\mathbf{X})$,$\cdots$,$\bar{C}_{2J_0-1}(\mathbf{X})$ and $\bar{S}_1(\mathbf{X})$,$\bar{S}_3(\mathbf{X})$,$\cdots$,$\bar{S}_{2J_0-1}(\mathbf{X})$ from $\bar{\mathbf{C}}^{J,K}(\mathbf{X})$ and $\mathbf{SC}^{J,K}(\mathbf{X})$ respectively, by series of linear transformations.

This D&R step is still pretty fast, because for each observation $\mathbf{x}$, we are computing two Cosines: $\cos(\mathbf{x})$ and $\cos(2J\mathbf{x})$. We get $sin(\mathbf{x}) = \sqrt{1-\cos^2(\mathbf{x})}$ for $\mathbf{x} \in (-1, 1)$. Then we multiply both $\cos(\mathbf{x})$ and $sin(\mathbf{x})$ in a for-loop by the multiplier $\cos^2(\mathbf{x})$, $J-1$ times, and then in a second for-loop we multiply all these $2J$ numbers by $\cos(2J\mathbf{x})$, $K-1$ times to get summands for all the SEP statistics. Again, a for-loop multiplication with constant multiplier is extremely fast. See the R function *map.EP-FQ2()* in Appendix C for a in-memory implementation of EP-FQCh2 algorithm.

This in-memory computation is still fast as we can see from the following run-time comparisons:

```
> J = 8
> K = 16
> start.time <- proc.time()
> lsf2 <- map.EP.FQCh2(X)
> end.time <- proc.time()
> end.time-start.time
user  system elapsed
2.488   0.044   2.534
>
```

We will explain later the choice of parameters $J = 8, K = 16$ is actually good choice for accuracy. We can see that in-memory run-time is good and beats binning. For this choice of J and K, we compute $8 \times 16 = 128$ Cosine and Sine terms.

Transformations for Cosine terms:

We can get the statistics $\bar{C}_1(\mathbf{X}), \bar{C}_3(\mathbf{X}), \cdots, \bar{C}_{2J_0-1}(\mathbf{X})$ from the matrix $\bar{\mathbf{C}}^{J,K}(\mathbf{X})$ by three consecutive linear transformations.

We define $\gamma_{j,k}^{m,n}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\cos^m(\mathbf{x}).\, T_j\big(\cos(\mathbf{x})\big).\, \cos^n(2J\mathbf{x}).\, T_l(\cos(2J\mathbf{x}))\Big)$, and, to make everything consistent, let $\mathbf{\Gamma}_{0,0}^{J,K}(\mathbf{X}) = \bar{\mathbf{C}}^{J,K}(\mathbf{X}) = \{\{\bar{C}^{2j-1,k-1}(\mathbf{X})\}\}$. Again remember, by definition, $T_0\big(\cos(\mathbf{x})\big) = 1$ and $T_1\big(\cos(\mathbf{x})\big) = \cos(\mathbf{x})$, and if we keep k fixed, then $\gamma_{0,0}^{2j-1,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \big(\cos^{2j-1}(\mathbf{x}).\, 1.\, \cos^{k-1}(2J\mathbf{x}).\, 1\big) = \bar{C}^{2j-1,k-1}(\mathbf{X})$ and $\gamma_{1,0}^{2j-2,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \big(\cos^{2j-2}(\mathbf{x}).\, \cos(\mathbf{x}).\, \cos^{k-1}(2J\mathbf{x}).\, 1\big) =$

$\bar{C}^{2j-1,k-1}(\mathbf{X})$. So, from the output of the D&R step, for each $k = 1, 2, \cdots, K$, we have the statistics $\gamma_{0,0}^{2j-1,k-1}(\mathbf{X})$ and $\gamma_{1,0}^{2j-2,k-1}(\mathbf{X})$ available to us for $j = 1, \cdots, J$.

Transformation 1: We run an iterative transformation on the D&R output, over the first index j, keeping the second index k fixed , for $k = 1, \cdots, K$. From the D&R output, we collect $\gamma_{1,k-1}^{0,0}(\mathbf{X})$ in transformation output and write all statistics in a 2-row matrix:

$$\Gamma_{0,k}^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_{0,k-1}^{1,0}(\mathbf{X}) & \gamma_{0,k-1}^{3,0}(\mathbf{X}) & \cdots & \gamma_{0,k-1}^{2J-1,0}(\mathbf{X}) \\ \gamma_{1,k-1}^{0,0}(\mathbf{X}) & \gamma_{1,k-1}^{2,0}(\mathbf{X}) & \cdots & \gamma_{1,k-1}^{2J-2,0}(\mathbf{X}) \end{bmatrix}$$

From Chebyshev's recursion $T_{j+1}(t) = 2tT_j(t) - T_{j-1}(t)$, we can derive the following recursive relations involving the pair of real numbers t and s,

$$\begin{aligned} t^{2m}T_{2j+1}(t)s^{k-1} &= 2t^{2m+1}T_{2j}(t)s^{k-1} - t^{2m}T_{2j-1}(t)s^{k-1} \\ t^{2m+1}T_{2j}(t)s^{k-1} &= 2t^{2m+2}T_{2j-1}(t)s^{k-1} - t^{2m+1}T_{2j-2}(t)s^{k-1} \end{aligned} \tag{3.20}$$

In equation 3.18, we replace t with $\cos(\mathbf{x})$ and s with $\cos(2J\mathbf{x})$, then take a sum over all $\mathbf{x} \in \mathbf{X}$, we get:

$$\begin{aligned} \gamma_{2j+1,0}^{2m,k}(\mathbf{X}) &= 2\gamma_{2j,0}^{2m+1,k}(\mathbf{X}) - \gamma_{2j-1,0}^{2m,k}(\mathbf{X}) \\ \gamma_{2j,0}^{2m+1,k}(\mathbf{X}) &= 2\gamma_{2j-1,0}^{2m+2,k}(\mathbf{X}) - \gamma_{2j-2,0}^{2m+1,k}(\mathbf{X}) \end{aligned} \tag{3.21}$$

If we examine the terms we can see that, the statistic $\gamma_{j,0}^{m,k}(\mathbf{X})$ is a weighted version of the statistic $\gamma_j^m(\mathbf{X})$ introduced in section 3.4, as for each term $\mathbf{x} \in X$, the sum $\gamma_j^m(\mathbf{X})$ has a corresponding summand $\cos^m(\mathbf{x})T_j\big(\cos(\mathbf{x})\big)$. This summand gets weighted by a factor $\cos^k(2J\mathbf{x})$ in the sum $\gamma_{j,0}^{m,k}(\mathbf{X})$. Observe this weight does not depend on m or j. Also, for fixed k, equation 3.21 is a weighted version of equation 3.13.

Remember EP-FQCh type A iteration introduced in section 3.4: we apply EP-FQCh type A iteration to the set of statistics $\gamma_0^1(\mathbf{X})$, $\gamma_0^3(\mathbf{X})$,$\cdots$,$\gamma_0^{2J_0-1}(\mathbf{X})$, we get

the set of statistics $\gamma_1^0(\mathbf{X})$,$\gamma_3^0(\mathbf{X})$,$\cdots$,$\gamma_{2J_0-1}^0(\mathbf{X})$ as output. Then for a fixed k, we can expect that, if we apply EP-FQCh type A recursion to the set of statistics $\gamma_{0,0}^{1,k}(\mathbf{X})$, $\gamma_{0,0}^{3,k}(\mathbf{X})$, $\cdots$,$\gamma_{0,0}^{2J_0-1,k}(\mathbf{X})$ (these are weighted versions of the set of statistics $\gamma_0^1(\mathbf{X})$, $\gamma_0^3(\mathbf{X})$, $\cdots$, $\gamma_0^{2J_0-1}(\mathbf{X})$), we will get the set of statistics $\gamma_{1,0}^{0,k}(\mathbf{X})$, $\gamma_{3,0}^{0,k}(\mathbf{X})$, $\cdots$, $\gamma_{2J_0-1,0}^{0,k}(\mathbf{X})$ as output.

For a fixed k, a general jth iteration step consists of 2 steps:

First step:

$$\begin{bmatrix} & \gamma_{2j-2,0}^{1,k}(\mathbf{X}) & \gamma_{2j-2,0}^{3,k}(\mathbf{X}) & \cdots & \gamma_{2j-2,0}^{2J-2j-1,k}(\mathbf{X}) & \gamma_{2j-2,0}^{2J-2j+1,k}(\mathbf{X}) \\ \gamma_{2j-1,0}^{0}(\mathbf{X}) & \gamma_{2j-1,0}^{2,k}(\mathbf{X}) & \gamma_{2j-1,0}^{4,k}(\mathbf{X}) & \cdots & \gamma_{2j-1,0}^{2J-2j,k}(\mathbf{X}) & \\ & \downarrow & \downarrow & \cdots & \downarrow & \\ & \gamma_{2j,0}^{1,k}(\mathbf{X}) & \gamma_{2j,0}^{3,k}(\mathbf{X}) & \cdots & \gamma_{2j,0}^{2J-2j-1,k}(\mathbf{X}) & \end{bmatrix}$$

Second step:

$$\begin{bmatrix} \gamma_{2j-1,0}^{0,k}(\mathbf{X}) & \gamma_{2j-1,0}^{2,k}(\mathbf{X}) & \cdots & \gamma_{2j-1,0}^{2J-2j-2,k}(\mathbf{X}) & \gamma_{2j-1,0}^{2J-2j,k}(\mathbf{X}) \\ \gamma_{2j,0}^{1,k}(\mathbf{X}) & \gamma_{2j,0}^{3,k}(\mathbf{X}) & \cdots & \gamma_{2j,0}^{2J-2j-1,k}(\mathbf{X}) & \\ \downarrow & \downarrow & \cdots & \downarrow & \\ \gamma_{2j+1,0}^{0,k}(\mathbf{X}) & \gamma_{2j+1,0}^{2,k}(\mathbf{X}) & \cdots & \gamma_{2j+1,0}^{2J-2j-2,k}(\mathbf{X}) & \end{bmatrix}$$

After jth iteration, we collect $\gamma_{2j+1,0}^{0,k-1}(\mathbf{X})$ in transformation output and get the matrix

$$\Gamma_{2j,k}^{(A)}(\mathbf{X}) = \begin{bmatrix} \gamma_{2j,0}^{1,k-1}(\mathbf{X}) & \gamma_{2j,0}^{3,k-1}(\mathbf{X}) & \cdots & \gamma_{2j,0}^{2J-2j-1,k-1}(\mathbf{X}) \\ \gamma_{2j+1,0}^{0,k-1}(\mathbf{X}) & \gamma_{2j+1,0}^{2,k-1}(\mathbf{X}) & \cdots & \gamma_{2j+1,0}^{2J-2j-2,k-1}(\mathbf{X}) \end{bmatrix}$$

for next iteration.

After $J-1$ iteration we get $\gamma_{1,0}^{0,k-1}(\mathbf{X}), \gamma_{3,0}^{0,k-1}(\mathbf{X}), \cdots, \gamma_{2J-1,0}^{0,k-1}(\mathbf{X})$ in transformation output. If we apply EP-FQCh type A iteration along all columns of the matrix $\mathbf{\Gamma}_{0,0}^{J,K}(\mathbf{X})$ (i.e. for $k = 1, \cdots, K$), we will get the $J \times K$ matrix $\mathbf{\Gamma}_{J,0}^{0,K}(\mathbf{X}) = \{\{\gamma_{2j-1,0}^{0,k-1}(\mathbf{X})\}\}$ as output.

Transformation 2: Now, we run another iterative transformation on the output of transformation 1, but this time over the second index k, while we keep the first index j fixed , for $j = 1, \cdots, J$. Again note that, $T_0\big(\cos(2J\mathbf{x})\big) = 1$ and $T_1\big(\cos(2J\mathbf{x})\big) = \cos(2J\mathbf{x})$. If we keep j fixed, then for $k > 1$, $\gamma^{0,k-1}_{2j-1,0}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(1.\ T_{2j-1}\big(\cos(\mathbf{x})\big)\ \cos^{k-1}(2J\mathbf{x}).\ 1\Big) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(1.\ T_{2j-1}\big(\cos(\mathbf{x})\big)\ \cos^{k-2}(2J\mathbf{x}).\ \cos(2J\mathbf{x})\Big) = \gamma^{0,k-2}_{2j-1,1}(\mathbf{X})$. So, from the output of transformation 1, we have the sets $\gamma^{0,0}_{2j-1,0}(\mathbf{X}), \cdots, \gamma^{0,K-1}_{2j-1,0}(\mathbf{X})$ and $\gamma^{0,1}_{2j-1,0}(\mathbf{X}), \cdots, \gamma^{0,K-2}_{2j-1,0}(\mathbf{X})$ for $j = 1, \cdots, J$ and for each j, we write these two sets as a 2-row statistics matrix:

$$\Gamma^{(B)}_{j,0}(\mathbf{X}) = \begin{bmatrix} \gamma^{0,0}_{2j-1,0}(\mathbf{X}) & \cdots & \gamma^{0,K-2}_{2j-1,0}(\mathbf{X}) & \gamma^{0,K-1}_{2j-1,0}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,1}(\mathbf{X}) & \cdots & \gamma^{0,K-2}_{2j-1,1}(\mathbf{X}) & \end{bmatrix}$$

For this j, we collect $\gamma^{0,0}_{2j-1,0}(\mathbf{X})$ and $\gamma^{0,0}_{2j-1,1}(\mathbf{X})$ in the transformation output. From Chebyshev's recursive relation, we can derive another iterative relation in t and s if we allow s to be the recursion variable:

$$T_{2j-1}(t)s^n T_k(s) = 2T_{2j-1}(t)s^{n+1}T_{k-1}(s) - T_{2j-1}(t)s^l T_{k-2}(s) \tag{3.22}$$

Now, in equation 3.22, again we replace t with $\cos(\mathbf{x})$ and s with $\cos(2J\mathbf{x})$ and take a sum over all $\mathbf{x} \in \mathbf{X}$ to get:

$$\gamma^{0,n}_{2j-1,k}(\mathbf{X}) = 2\gamma^{0,n+1}_{2j-1,k-1}(\mathbf{X}) - \gamma^{0,n}_{2j-1,k-2}(\mathbf{X}). \tag{3.23}$$

This equation gives us another iterative process that we are going to call EP-FQCh type B iteration. For a fixed j, in the first step of EP-FQCh type B iteration, we compute $\gamma^{0,0}_{2j-1,2}(\mathbf{X})$, $\gamma^{0,1}_{2j-1,2}(\mathbf{X})$, $\cdots$, $\gamma^{0,K-3}_{2j-1,2}(\mathbf{X})$ from $\gamma^{0,1}_{2j-1,1}(\mathbf{X})$, $\gamma^{0,2}_{2j-1,1}(\mathbf{X})$, $\cdots$, $\gamma^{0,K-3}_{2j-1,1}(\mathbf{X})$ and $\gamma^{0,0}_{2j-1,0}(\mathbf{X})$, $\gamma^{0,1}_{2j-1,0}(\mathbf{X})$, $\cdots$,$\gamma^{0,K-2}_{2j-1,0}(\mathbf{X})$, using the relation in 3.23. In the matrix $\Gamma^{(B)}_{j,0}(\mathbf{X})$, we shift each element in the 2nd row to left and then apply the relation to columns $2, \cdots, K-2$ to get a new set of statistics as:

$$\begin{bmatrix} & \gamma^{0,0}_{2j-1,0}(\mathbf{X}) & \gamma^{0,1}_{2j-1,0}(\mathbf{X}) & \cdots & \gamma^{0,K-3}_{2j-1,0}(\mathbf{X}) & \gamma^{0,K-2}_{2j-1,0}(\mathbf{X}) & \gamma^{0,K-1}_{2j-1,0}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,1}(\mathbf{X}) & \gamma^{0,1}_{2j-1,1}(\mathbf{X}) & \gamma^{0,2}_{2j-1,1}(\mathbf{X}) & \cdots & \gamma^{0,K-2}_{2j-1,1}(\mathbf{X}) & & \\ & \downarrow & \downarrow & \cdots & \downarrow & & \\ & \gamma^{0,0}_{2j-1,2}(\mathbf{X}) & \gamma^{0,1}_{2j-1,2}(\mathbf{X}) & \cdots & \gamma^{0,K-3}_{2j-1,2}(\mathbf{X}) & & \end{bmatrix}$$

We collect $\gamma^{0,0}_{2j-1,2}(\mathbf{X})$ in transformation output, then, we discard the first row of $\Gamma^{(B)}_{j,0}(\mathbf{X})$ and add these new statistics in a new row to get the statistics matrix

$$\Gamma^{(B)}_{j,1}(\mathbf{X}) = \begin{bmatrix} \gamma^{0,0}_{2j-1,1}(\mathbf{X}) & \cdots & \gamma^{0,K-3}_{2j-1,1}(\mathbf{X}) & \gamma^{0,K-2}_{2j-1,1}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,2}(\mathbf{X}) & \cdots & \gamma^{0,K-2}_{2j-1,2}(\mathbf{X}) & \end{bmatrix}$$

For this fixed j, we start the kth step of the iteration with 2-row statistics matrix:

$$\Gamma^{(B)}_{j,k-1}(\mathbf{X}) = \begin{bmatrix} \gamma^{0,0}_{2j-1,k-1}(\mathbf{X}) & \cdots & \gamma^{0,K-k-1}_{2j-1,k-1}(\mathbf{X}) & \gamma^{0,K-k}_{2j-1,k-1}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,k}(\mathbf{X}) & \cdots & \gamma^{0,K-k-1}_{2j-1,k}(\mathbf{X}) & \end{bmatrix}$$

In kth step, we compute $\gamma^{0,0}_{2j-1,k+1}(\mathbf{X})$, $\gamma^{0,1}_{2j-1,k+1}(\mathbf{X})$, $\cdots$, $\gamma^{0,K-k-2}_{2j-1,k+1}(\mathbf{X})$ from $\gamma^{0,1}_{2j-1,k}(\mathbf{X})$, $\gamma^{0,2}_{2j-1,k}(\mathbf{X})$, $\cdots$, $\gamma^{0,K-k-1}_{2j-1,k}(\mathbf{X})$ and $\gamma^{0,0}_{2j-1,k-1}(\mathbf{X})$, $\gamma^{0,1}_{2j-1,k-1}(\mathbf{X})$, $\cdots$, $\gamma^{0,K-k-2}_{2j-1,k-1}(\mathbf{X})$, using the relation in 3.23. Again, for the matrix $\Gamma^{(B)}_{j,k-1}(\mathbf{X})$, we shift each element in the 2nd row to left and then apply the relation to columns $2, \cdots, K-k-2$ to get new set of statistics:

$$\begin{bmatrix} & \gamma^{0,0}_{2j-1,k-1}(\mathbf{X}) & \gamma^{0,1}_{2j-1,k-1}(\mathbf{X}) & \cdots & \gamma^{0,K-k-2}_{2j-1,k-1}(\mathbf{X}) & \gamma^{0,K-k-1}_{2j-1,k-1}(\mathbf{X}) & \gamma^{0,K-k}_{2j-1,k-1}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,k}(\mathbf{X}) & \gamma^{0,1}_{2j-1,k}(\mathbf{X}) & \gamma^{0,2}_{2j-1,k}(\mathbf{X}) & \cdots & \gamma^{0,K-k-1}_{2j-1,k}(\mathbf{X}) & & \\ & \downarrow & \downarrow & \cdots & \downarrow & & \\ & \gamma^{0,0}_{2j-1,k+1}(\mathbf{X}) & \gamma^{0,1}_{2j-1,k+1}(\mathbf{X}) & \cdots & \gamma^{0,K-k-2}_{2j-1,k+1}(\mathbf{X}) & & \end{bmatrix}$$

After k iteration, we collect $\gamma^{0,0}_{2j-1,k+1}(\mathbf{X})$ in transformation output, then, again, we discard the first row of $\Gamma^{(B)}_{j,k-1}(\mathbf{X})$ and add these in new row to get the statistics matrix

$$\Gamma^{(B)}_{j,k}(\mathbf{X}) = \begin{bmatrix} \gamma^{0,0}_{2j-1,k}(\mathbf{X}) & \cdots & \gamma^{0,K-k-2}_{2j-1,k}(\mathbf{X}) & \gamma^{0,K-k-1}_{2j-1,k}(\mathbf{X}) \\ \gamma^{0,0}_{2j-1,k+1}(\mathbf{X}) & \cdots & \gamma^{0,K-k-2}_{2j-1,k+1}(\mathbf{X}) & \end{bmatrix}$$

We perform $K-2$ iterations and after these iterations, we will have $\gamma^{0,0}_{2j-1,0}(\mathbf{X})$, $\gamma^{0,0}_{2j-1,1}(\mathbf{X})$, $\cdots$, $\gamma^{0,0}_{2j-1,K-1}(\mathbf{X})$. See Appendix B.7 for an algorithmic flow -chart of EP-FQCh type B iteration. We apply EP-FQCh type B iteration along the rows of the matrix $\mathbf{\Gamma}^{0,K}_{J,0}(\mathbf{X})$ (i.e. for $j=1,\cdots,J$) and we get the $J\times K$ matrix $\mathbf{\Gamma}^{0,0}_{J,K}(\mathbf{X}) = \{\{\gamma^{0,0}_{2j-1,k-1}(\mathbf{X})\}\}$.

Transformation 3: Observe that: $\gamma^{0,0}_{2j-1,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\Big(T_{2j-1}\big(\cos(\mathbf{x})\big)\, T_{k-1}\big(\cos(2J\mathbf{x})\big)\Big) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\Big(\cos\big((2j-1)\mathbf{x}\big)\cos\big(2(k-1)J\mathbf{x}\big)\Big)$. For simplicity, let us introduce the statistics: $\bar{C}_{2j-1,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\Big(\cos\big((2j-1)\mathbf{x}\big)\cos\big(2(k-1)J\mathbf{x}\big)\Big)$. Also, we denote the output of transformation 2 as $J\times K$ matrix $\bar{\mathbf{C}}_{J,K}(\mathbf{X}) = \{\{\bar{C}_{2j-1,k-1}(\mathbf{X})\}\}$ $(=\mathbf{\Gamma}^{0,0}_{J,K}(\mathbf{X}))$.

Introduce the vector valued statistics: $\bar{\mathbf{C}}_{.,k}(\mathbf{X}) = \big(\bar{C}_{1,k}(\mathbf{X}), \bar{C}_{3,k}(\mathbf{X}), \cdots, \bar{C}_{2J-1,k}(\mathbf{X})\big)^t$. So the columns of $\mathbf{\Gamma}^{0,0}_{J,K}(\mathbf{X})$ are $\bar{\mathbf{C}}_{.,0}(\mathbf{X})$, $\bar{\mathbf{C}}_{.,1}(\mathbf{X})$, $\cdots$, $\bar{\mathbf{C}}_{.,(K-1)}(\mathbf{X})$. Also for any two integers $j_1 < j_2$, introduce the statistics: $\bar{\mathbf{C}}^{j_1}_{j_2}(\mathbf{X}) = \big(\bar{C}_{2j_1-1}(\mathbf{X}), \bar{C}_{2(j_1+1)-1}(\mathbf{X}), \cdots, \bar{C}_{2j_2-1}(\mathbf{X})\big)^t$. First realize, $\bar{\mathbf{C}}_{.,0}(\mathbf{X}) = \bar{\mathbf{C}}^1_J(\mathbf{X})$. Let $Rev(\mathbf{v})$ is the vector whose elements are elements of the vector $\mathbf{v}$ in reverse order.

Now remember sum of Cosine formula, for $k>0$ and $1\le j\le J$, we have:

$$\cos\big((2kJ+2j-1)\mathbf{x}\big)+\cos\big((2kJ-2j+1)\mathbf{x}\big) = 2\cos(2kJ\mathbf{x})\cos\big((2j-1)\mathbf{x}\big) \quad (3.24)$$

If we take sum over all $\mathbf{x}\in\mathbf{X}$, we have:

$$\bar{C}_{2kJ+2j-1}(\mathbf{X}) + \bar{C}_{2kJ-2j+1}(\mathbf{X}) = 2\bar{C}_{2j-1,k}(\mathbf{X}) \quad (3.25)$$

Equivalently, we have:

$$\bar{C}_{2kJ+2j-1}(\mathbf{X}) = 2\bar{C}_{2j-1,k}(\mathbf{X}) - \bar{C}_{2(k-1)J+2(J+1-j)-1}(\mathbf{X}) \quad (3.26)$$

For a given k, this relation holds for $j=1,\cdots,J$, and in terms of vectors:

$$\bar{\mathbf{C}}^{kJ+1}_{(k+1)J}(\mathbf{X}) = 2\bar{\mathbf{C}}_{.,k}(\mathbf{X}) - Rev\big(\bar{\mathbf{C}}^{(k-1)J+1}_{kJ}(\mathbf{X})\big) \quad (3.27)$$

Using the relation in 3.27 we develop an iterative transformation: EP-FQCh type C iteration to get all the Cosine terms. We have the statistics matrix $\mathbf{\Gamma}^{0,0}_{J,K}(\mathbf{X})$, we

start with the first column $\bar{\mathbf{C}}_{.,0}(\mathbf{X}) = \bar{\mathbf{C}}^1_J(\mathbf{X})$ and using 3.27 we iteratively compute $\bar{\mathbf{C}}^{kJ+1}_{(k+1)J}(\mathbf{X})$ for $k = 1, \cdots, K-1$. The final output is the vector valued statistics: $\bar{\mathbf{C}}^1_{JK}(\mathbf{X}) = \big(\bar{\mathbf{C}}^1_J(\mathbf{X}), \bar{\mathbf{C}}^{J+1}_{2J}(\mathbf{X}), \cdots, \bar{\mathbf{C}}^{(K-1)J+1}_{KJ}(\mathbf{X})\big) = \big(\bar{C}_1(\mathbf{X}), \cdots, \bar{C}_{2J_0-1}(\mathbf{X})\big)$, consisting of all Cosine terms. See Appendix B.8 for an algorithmic flow-chart for EP-FQCh type C iteration.

Transformations for Sine terms:

Similarly, we can get the set of statistics $\bar{S}_1(\mathbf{X}), \bar{S}_3(\mathbf{X}), \cdots, \bar{S}_{2J_0-1}(\mathbf{X})$ from the statistics matrix $\overline{\mathbf{SC}}^{J,K}(\mathbf{X})$ by four consecutive similar linear transformations. We perform an initial transformation to get the statistics matrix $\bar{\mathbf{S}}^{J,K}(\mathbf{X})$ from $\overline{\mathbf{SC}}^{J,K}(\mathbf{X})$.

Transformation 1: If we fix k, and multiply both sides of equation 3.10 by $\cos(2J\mathbf{x})^{k-1}$ we get:

$$\sin^{2j-1}(\mathbf{x})\cos(2J\mathbf{x})^{k-1} = \sin(\mathbf{x})\cos(2J\mathbf{x})^{k-1} - \sum_{j'=1}^{j-1}\sin(\mathbf{x})\cos^{2j'}(\mathbf{x})\cos(2J\mathbf{x})^{k-1} \tag{3.28}$$

We take a sum over all $\mathbf{x} \in \mathbf{X}$ to get:

$$\bar{S}^{2j-1,k-1}(\mathbf{X}) = \overline{SC}^{0,k-1}(\mathbf{X}) - \sum_{j'=1}^{j}\overline{SC}^{2j',k-1}(\mathbf{X}) \tag{3.29}$$

We use the relation in 3.29, for each pair $\{j,k\}; j = 1, \cdots, J; k = 1, \cdots, K$ to get the $J \times K$ statistics matrix $\bar{\mathbf{S}}^{J,K}(\mathbf{X}) = \{\{\bar{S}^{2j-1,k-1}(\mathbf{X})\}\}$. We define for $\{m,n,j,k\} \subset \mathbb{N}$, the statistic $\delta^{m,n}_{j,k}(\mathbf{X}) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\Big(\sin^m(\mathbf{x}).\, T_j\big(\sin(\mathbf{x})\big).\cos^n(2J\mathbf{x}).\, T_l\big(\cos(2J\mathbf{x})\big)\Big)$, and again, to keep everything consistent, define: $\mathbf{\Delta}^{J,K}_{0,0}(\mathbf{X}) = \bar{\mathbf{S}}^{J,K}(\mathbf{X}) = \{\{\bar{S}^{2j-1,k-1}(\mathbf{X})\}\}$.

Transformation 2: This step is very similar to *Transformation 1* for Cosine statistics. We apply EP-FQCh type A iteration along the columns of the matrix $\mathbf{\Delta}^{J,K}_{0,0}(\mathbf{X})$, and end up with $J \times K$ statistics matrix $\mathbf{\Delta}^{0,K}_{J,0}(\mathbf{X}) = \{\{\delta^{0,k-1}_{2j-1,0}(\mathbf{X})\}\}$.

Transformation 3: This step is also similar to *Transformation 2* for Cosine statistics. We apply EP-FQCh type B iteration along the rows of the matrix $\mathbf{\Delta}^{0,K}_{J,0}(\mathbf{X})$ to get the $J \times K$ statistics matrix $\mathbf{\Delta}^{0,0}_{J,K}(\mathbf{X}) = \{\{\delta^{0,0}_{2j-1,k-1}(\mathbf{X})\}\}$.

Transformation 4: Now observe that: $\delta^{0,0}_{2j-1,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(T_{2j-1}\big(\sin(\mathbf{x})\big)\, T_{k-1}\big(\cos(2J\mathbf{x})\big)\Big) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} (-1)^{j-1} \Big(\sin\big((2j-1)\mathbf{x}\big)\cos\big(2(k-1)J\mathbf{x}\big)\Big)$. For Sine terms, if we introduce the statistics: $\bar{S}_{2j-1,k-1}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\sin\big((2j-1)\mathbf{x}\big)\cos\big(2(k-1)J\mathbf{x}\big)\Big)$, then, for the statistics matrix $\mathbf{\Delta}^{0,0}_{J,K}(\mathbf{X})$, we have $\mathbf{\Delta}^{0,0}_{J,K}(\mathbf{X}) = \{\{\delta^{0,0}_{2j-1,k-1}(\mathbf{X})\}\} = \{\{(-1)^{j-1}\ \bar{S}_{2j-1,k-1}(\mathbf{X})\}\}$. We multiply, the jth row of $\mathbf{\Delta}^{0,0}_{J,K}(\mathbf{X})$ by $(-1)^{j-1}$ for $j = 1, \cdots, J$, to get the matrix $\bar{\mathbf{S}}_{J,K}(\mathbf{X}) = \{\{\bar{S}_{2j-1,k-1}(\mathbf{X})\}\}$. Also, introduce vector statistics: $\bar{\mathbf{S}}_{.,k}(\mathbf{X}) = \big(\bar{S}_{1,k}(\mathbf{X}), \bar{S}_{3,k}(\mathbf{X}), \cdots, \bar{S}_{2J-1,k}(\mathbf{X})\big)^t$ and $\bar{\mathbf{S}}^{j_1}_{j_2}(\mathbf{X}) = \big(\bar{S}_{2j_1-1}(\mathbf{X}),\ \bar{S}_{2(j_1+1)-1}(\mathbf{X}),\ \cdots,\ \bar{S}_{2j_2-1}(\mathbf{X})\big)^t$ for integers $j_1 < j_2$. Again, $\bar{\mathbf{S}}_{.,0}(\mathbf{X}) = \bar{\mathbf{S}}^1_J(\mathbf{X})$.

From sum of Sines formula, for $k > 0$ and $1 \le j \le J$, we have:

$$\sin\big((2kJ + 2j - 1)\mathbf{x}\big) - \sin\big((2kJ - 2j + 1)\mathbf{x}\big) = 2\sin\big((2j-1)\mathbf{x}\big)\cos(2kJ\mathbf{x}) \quad (3.30)$$

Taking sum over $\mathbf{x} \in \mathbf{X}$, we have:

$$\bar{S}_{2kJ+2j-1}(\mathbf{X}) - \bar{S}_{2kJ-2j+1}(\mathbf{X}) = 2\bar{S}_{2j-1,k}(\mathbf{X}) \quad (3.31)$$

Equivalently, we have:

$$\bar{S}_{2kJ+2j-1}(\mathbf{X}) = 2\bar{S}_{2j-1,k}(\mathbf{X}) + \bar{S}_{2(k-1)J+2(J+1-j)-1}(\mathbf{X}) \quad (3.32)$$

Given k, this relation holds for $j = 1, \cdots, J$, and in terms of vectors:

$$\bar{\mathbf{S}}^{kJ+1}_{(k+1)J}(\mathbf{X}) = 2\bar{\mathbf{S}}_{.,k}(\mathbf{X}) - Rev\big(\bar{\mathbf{S}}^{(k-1)J+1}_{kJ}(\mathbf{X})\big) \quad (3.33)$$

So, using the relation in 3.33 we develop EP-FQCh type D iteration to get all the Sine terms. We have the statistics matrix $\mathbf{\Delta}^{0,0}_{J,K}(\mathbf{X})$, we start with the column $\bar{\mathbf{S}}_{.,0}(\mathbf{X}) = \bar{\mathbf{S}}^1_J(\mathbf{X})$ and using3.33 we iteratively compute $\bar{\mathbf{S}}^{kJ+1}_{(k+1)J}(\mathbf{X})$ for $k = 1, \cdots, K-1$. The final output is the vector valued statistics: $\bar{\mathbf{S}}^1_{JK}(\mathbf{X}) = \big(\bar{\mathbf{S}}^1_J(\mathbf{X}), \bar{\mathbf{S}}^{J+1}_{2J}(\mathbf{X}), \cdots,$

$\bar{\mathbf{S}}_{KJ}^{(K-1)J+1}(\mathbf{X})\big) = \big(\bar{S}_1(\mathbf{X}), \cdots, \bar{S}_{2J_0-1}(\mathbf{X})\big)$, having all Sine statistics as its elements. See Appendix B.9 for an algorithmic flow-chart for EP-FQCh Type D iteration.

After the D&R step, for each quantile p we maximize the optimization criteria in equation 3.7 to get corresponding $\hat{Q}_P^{J_0}(\mathbf{X})$. See Appendix B.4 for an algorithmic flow-chart for EP-FQCh2 algorithm.

Now let us see how accurate the 2-fold Fourier method can be. We get accurate quantiles if the number of terms increase, so if J and K both increases, then we would expect higher accuracy. However, remember from the discussion in last section, that if J and K increases, then even if we compute the statistics $\bar{C}^{j,k}(\mathbf{X})$ accurately by D&R, there will be small errors. which get weighted by exponentially large factors, when we try to compute the statistics $\bar{C}_{j,k}(\mathbf{X})$ by linear transformations.

As a rule of thumb we observed that, $\max_{j,k} \log(\varepsilon\big(\bar{C}_{2kJ2j-1}(\mathbf{X})\big)) \propto 2J + K$. With the constraint that $2J + K$ cannot be large, $J \times K$ takes the maximum value when $2J = K$, and JK should be larger for higher accuracy. So for optimum choice we should have $2J = K$.

Here is the plot to visualize how the average error increases as we increase general the index for 2-fold Fourier method for the same data mentioned as the global variables.

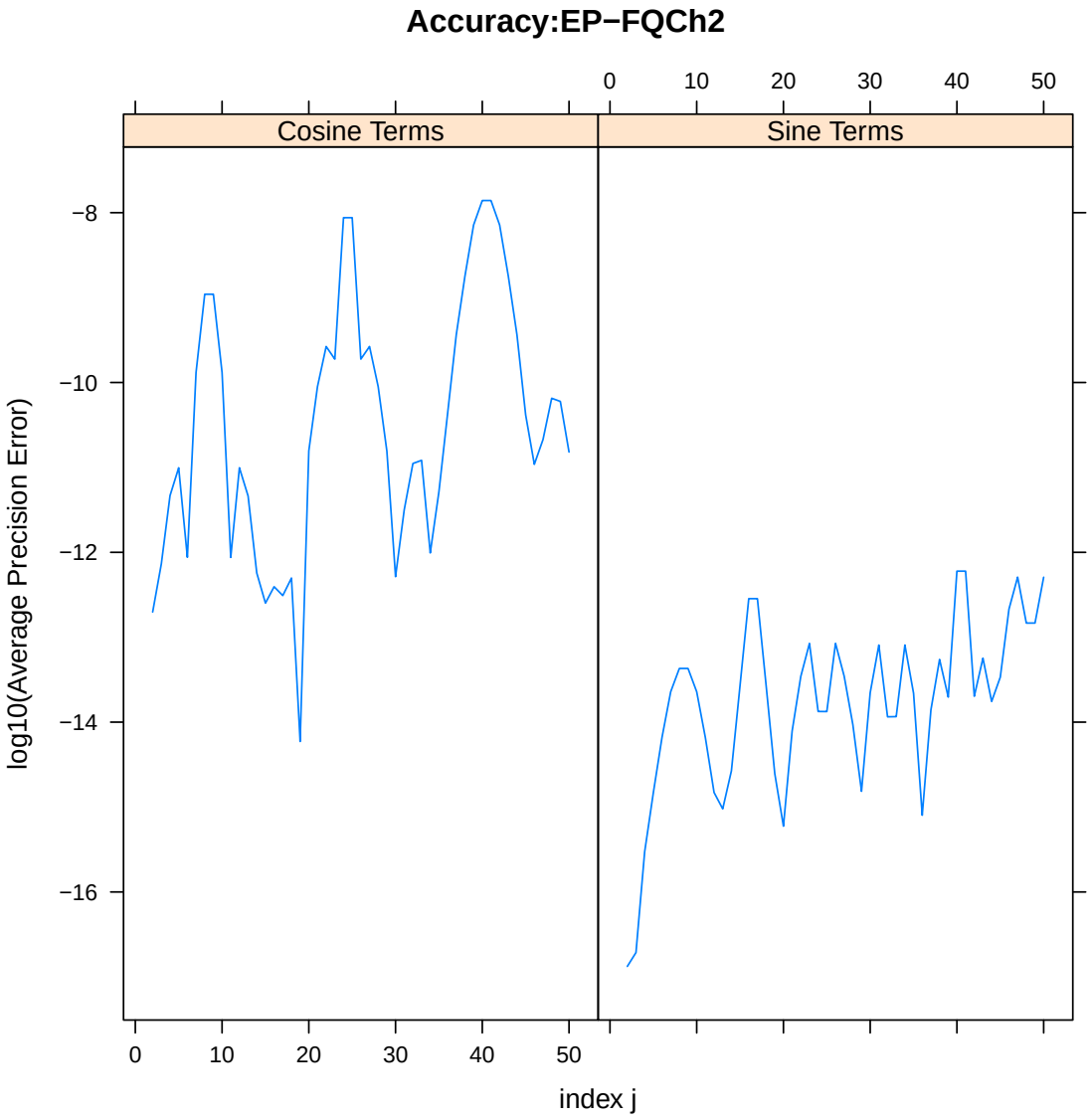

Figure 3.2. Accuracy of Cosine/Sine statistics: EP-FQCh2

We can see some improvement from the output for EP-FQCh1, though it is difficult to quantify the change just by looking at this plot. Remember that EP-FQCh2 beats Binning in in-memory run-time.

3.6 General p-way Fourier-Chebyshev method: EP-FQChp algorithm

2-way Fourier-Chebyshev method doesn't beat Binning in both accuracy and run-time. (we are skipping results, the reader can verify it). However, the idea of 2-way Fourier-Chebyshev method, can be further generalized to general p-way Fourier-Chebyshev method, which beats Binning in both accuracy and run-time. In general p-way Fourier-Chebyshev method we perform p Cosine operation for each $\mathbf{x}$.

We want to compute J_0 Cosine and Sine statistics, first we pick p integers $J_1, \cdots, J_p$ such that $J_0 = J_1 \times \cdots \times J_p$. Let $K_1 = 1, K_2 = 2J_1, K_3 = 2J_1J_2, \cdots, K_p = 2J_1J_2\cdots J_{p-1}$. Also, let $\boldsymbol{j}$ denotes the integer vector $(j_1, \cdots, j_p)' \in \mathbb{N}^p$ and for this $\boldsymbol{j}$, let $\boldsymbol{j_{-1}}$ denotes the integer vector $(j_2, \cdots, j_p)' \in \mathbb{N}^{p-1}$. We now introduce SEP statistics:

$$\begin{aligned}
\bar{C}^{\boldsymbol{j}}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\prod_{t=1}^{p} \cos^{j_t}(K_t\mathbf{x}) \Big) \\
\overline{SC}^{\boldsymbol{j}}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\sin(\mathbf{x}) \prod_{t=1}^{p} \cos^{j_t}(K_t\mathbf{x}) \Big) \\
\bar{S}^{\boldsymbol{j}}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\prod_{t=1}^{p} \sin^{j_t}(K_t\mathbf{x}) \Big)
\end{aligned} \tag{3.34}$$

Let $\boldsymbol{J}$ is the vector $(J_1, \cdots, J_p)$ and the set $\mathbb{J}$ is defined as: $\mathbb{J} = \prod_{t=1}^{p}\{1, \cdots, J_t\}$. Also let $\mathbf{0}$ is the vector $(0, \cdots, 0)$(can be of arbitrary length). In the first step of EP-FQChp algorithm, we compute the p dimensional statistics matrices: $\bar{\mathbf{C}}^{\boldsymbol{J}}(\mathbf{X}) = \{\{\bar{C}^{2j_1-1, j_2-1, \cdots, j_p-1}(\mathbf{X})\}\}$, $\boldsymbol{j} \in \mathbb{J}$ for Cosines, and $\overline{\mathbf{SC}}^{\boldsymbol{J}}(\mathbf{X}) = \{\{\overline{SC}^{2j_1-2, j_2-1, \cdots, j_p-1}(\mathbf{X})\}\}$, $\boldsymbol{j} \in \mathbb{J}$ for Sines in a single D&R step, as all of these $2J_0$ statistics are SEP. This is still a fast D&R step, for each observation $\mathbf{x}$, we compute p cosine terms $\cos(K_1\mathbf{x})(= \cos(\mathbf{x})), \cos(K_2\mathbf{x}), \cdots, \cos(K_p\mathbf{x})$, and again we get $sin(\mathbf{x}) = \sqrt{1-\cos^2(\mathbf{x})}$ for $\mathbf{x} \in (-1, 1)$, then we get summands for all SEP statistics via p for loops, which are fast. Afterwards, we get the Cosine and Sine statistics from $\bar{\mathbf{C}}^{\boldsymbol{J}}(\mathbf{X})$ and $\overline{\mathbf{SC}}^{\boldsymbol{J}}(\mathbf{X})$ respectively through two sequence of transformations.

Transformations for Cosine terms:

Again, we can get the statistics $\bar{C}_1(\mathbf{X})$, $\bar{C}_3(\mathbf{X})$, $\cdots$, $\bar{C}_{2J_0-1}(\mathbf{X})$ from the p dimensional statistics array $\bar{\mathbf{C}}^{J,K}(\mathbf{X})$ by three consecutive linear transformations. Again, we define $\gamma_{\boldsymbol{j}}^{\boldsymbol{m}}(\mathbf{X}) = \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}}\Big(\prod_{t=1}^{p}\cos^{m_t}(K_t\mathbf{x}).T_{j_t}\big(\cos(\mathbf{x})\big)\Big)$ for $\boldsymbol{j}, \boldsymbol{m} \in \mathbb{N}^p$ and for consistency, define: $\mathbf{\Gamma}_{\mathbf{0}}^{\boldsymbol{J}}(\mathbf{X}) = \bar{\mathbf{C}}^{\boldsymbol{J}}(\mathbf{X}) = \{\{\bar{C}^{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})\}\}$.

Transformation 1: Now, like 3.21, for fixed $\boldsymbol{j_{-1}} = j_2, \cdots, j_p$, we can establish the iterative relation:

$$\begin{aligned}\gamma_{2j_1+1,\mathbf{0}}^{2m_1,\boldsymbol{j_{-1}}}(\mathbf{X}) &= 2\gamma_{2j_1,\mathbf{0}}^{2m_1+1,\boldsymbol{j_{-1}}}(\mathbf{X}) - \gamma_{2j_1-1,\mathbf{0}}^{2m_1,\boldsymbol{j_{-1}}}(\mathbf{X})\\ \gamma_{2j_1,\mathbf{0}}^{2m_1+1,\boldsymbol{j_{-1}}}(\mathbf{X}) &= 2\gamma_{2j_1-1,\mathbf{0}}^{2m_1+2,\boldsymbol{j_{-1}}}(\mathbf{X}) - \gamma_{2j_1-2,\mathbf{0}}^{2m_1+1,\boldsymbol{j_{-1}}}(\mathbf{X})\end{aligned} \tag{3.35}$$

So, if we apply EP-FQCh type A iteration along the first co-ordinate of $\mathbf{\Gamma}_{\mathbf{0}}^{\boldsymbol{J}}(\mathbf{X})$, we will get the p dimensional statistics array $\mathbf{\Gamma}_{J_1,0,\cdots,0}^{0,J_2,\cdots,J_p}(\mathbf{X}) = \{\{\gamma_{2j_1-1,0,\cdots,0}^{0,j_2-1,\cdots,j_p-1}(\mathbf{X})\}\}$, for $\boldsymbol{j} \in \mathbb{J}$.

Transformation 2: Now, consider a coordinate t, such that $t > 1$ and let e_t is the unit vector along t. If we keep $j_{t'}$ constant for $t' \neq t$, then similar to equation 3.22, we have another iterative relation:

$$\gamma_{\boldsymbol{j}+e_t}^{\boldsymbol{m}}(\mathbf{X}) = 2\gamma_{\boldsymbol{j}}^{\boldsymbol{m}+e_t}(\mathbf{X}) + \gamma_{\boldsymbol{j}-e_t}^{\boldsymbol{m}}(\mathbf{X}) \tag{3.36}$$

So, if we let $t = 2$, and apply EP-FQCh type B recursion along second co-ordinate of $\mathbf{\Gamma}_{J_1,0,\cdots,0}^{0,J_2,\cdots,J_p}(\mathbf{X})$, we get p dimensional statistics array $\mathbf{\Gamma}_{J_1,J_2,0,\cdots,0}^{0,0,J_3,\cdots,J_p}(\mathbf{X}) = \{\{\gamma_{2j_1-1,j_2-1,0,\cdots,0}^{0,0,j_3-1,\cdots,j_p-1}(\mathbf{X})\}\}$ for $\boldsymbol{j} \in \mathbb{J}$. Afterwards, if we let $t = 3$, and apply EP-FQCh type B recursion along third co-ordinate of $\mathbf{\Gamma}_{J_1,J_2,0,\cdots,0}^{0,0,J_3,\cdots,J_p}(\mathbf{X})$, we get p dimensional statistics array $\mathbf{\Gamma}_{J_1,J_2,J_3,0,\cdots,0}^{0,0,0,J_4,\cdots,J_p}(\mathbf{X}) = \{\{\gamma_{2j_1-1,j_2-1,j_3-1,0,\cdots,0}^{0,0,0,j_4-1,\cdots,j_p-1}(\mathbf{X})\}\}$ for $\boldsymbol{j} \in \mathbb{J}$. If we repeat this procedure another $p-3$ times for $t = 4, \cdots, p$; we get the p dimensional statistics array $\mathbf{\Gamma}_{\boldsymbol{J}}^{\mathbf{0}}(\mathbf{X}) = \{\{\gamma_{2j_1-1,j_2-1,\cdots,j_p-1}^{0,0,\cdots,0}(\mathbf{X})\}\}$.

Transformation 3: Observe that:

$$\begin{aligned}\gamma^{0,0,\cdots,0}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} T_{2j_1-1}\big(cos(\mathbf{x})\big)\prod_{t=2}^{p} T_{j_p-1}\big(cos(2K_p\mathbf{x})\big)\\ &= \frac{1}{N(\mathbf{X})}\sum_{\mathbf{x}\in\mathbf{X}} \cos\big((2j_1-1)\mathbf{x}\big)\prod_{t=2}^{p}\cos\big((j_p-1)K_p\mathbf{x}\big)\end{aligned} \tag{3.37}$$

For convenience, we call this statistics $\bar{C}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})$ and also let $\bar{\mathbf{C}}_{\boldsymbol{J}}(\mathbf{X}) = \{\{\gamma^{0,0,\cdots,0}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})\}\}(= \boldsymbol{\Gamma}^{\mathbf{0}}_{\boldsymbol{J}}(\mathbf{X})$. Now, remember the Sum of Cosines identity in equation 3.24, if we replace J with J_1, j with j_1 and k with j_2, multiply both sides with $\prod_{t=3}^{p}\cos^{j_t}(K_t\mathbf{x})$, then take a sum over $\mathbf{x}\in\mathbf{X}$, we get

$$\bar{C}_{2J_1j_2+2j_1-1,j_3-1,\cdots,j_p-1}(\mathbf{X}) + \bar{C}_{2J_1j_2-2j_1+1,j_3-1,\cdots,j_p-1}(\mathbf{X}) = 2\bar{C}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X}) \tag{3.38}$$

So, we apply EP-FQCh type C recursion along the first two co-ordinates of $\bar{\mathbf{C}}_{\boldsymbol{J}}(X)$ to get $p-1$ dimensional array statistic $\bar{\mathbf{C}}_{J_1J_2,J_3,\cdots,J_p}(\mathbf{X}) = \{\{\bar{C}_{2j_1-1,j_3-1,\cdots,j_p-1}(\mathbf{X})\}\}$, here $(j_1,j_3,\cdots,j_p)\in\{1,\cdots,J_1J_2\}\times\prod_{t=3}^{p}\{1,\cdots,J_t\}$. Again, we apply EP-FQCh type C recursion on the resulting array $\bar{\mathbf{C}}_{J_1J_2,J_3,\cdots,J_p}(\mathbf{X})$ to get the $p-2$ dimensional array statistic $\bar{\mathbf{C}}_{J_1J_2J_3,J_4\cdots,J_p}(\mathbf{X})\{\{\bar{C}_{2j_1-1,j_4-1,\cdots,j_p-1}(\mathbf{X})\}\}$, once again, $(j_1,j_4,\cdots,j_p)\in\{1,\cdots,J_1J_2J_3\}\times\prod_{t=4}^{p}\{1,\cdots,J_t\}$. If we repeat this process $p-3$ times, each time applying EP-FQCh type C recursion on the resulting array, eventually at the end, we will have the statistics vector $\bar{\mathbf{C}}(\mathbf{X}) = \{\bar{C}_{2j-1}(X)\}$, $j=1,\cdots,J_0$, where, $J_0 = J_1J_2\cdots J_p$.

Transformations for Sine terms:

We get statistics $\bar{S}_1(\mathbf{X})$, $\bar{S}_3(\mathbf{X})$, $\cdots$, $\bar{S}_{2J_0-1}(\mathbf{X})$ from the p dimensional statistics array $\overline{\mathbf{SC}}^{J,K}(\mathbf{X})$ by four consecutive linear transformations.

Transformation 1: If we replace $j=j_1$ in 3.10, multiply both sides with $\prod_{t=2}^{p}\cos^{j_t}(K_t\mathbf{x})$, then take a sum over $\mathbf{x}\in\mathbf{X}$, we get

$$\bar{S}^{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X}) = \overline{SC}^{0,j_2-1,\cdots,j_p-1}(\mathbf{X}) - \sum_{j'=1}^{j_1} \overline{SC}^{2j',j_2-1,\cdots,j_p-1}(\mathbf{X}) \tag{3.39}$$

For the Sine terms, in Step 1 we get the statistics $\bar{S}^{2j_1-1,j_2-1,\cdots,j_p-1}(X)$ from D&R output statistic $\overline{SC}^{2j_1-2,j_2-1,\cdots,j_p-1}(X)$ for $\boldsymbol{j} \in \mathbb{J}$ using equation 3.39. We define $\delta^{\boldsymbol{m}}_{\boldsymbol{j}}(\mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} \Big(\sin^{m_1}(\mathbf{x}).T_{j_1}\big(\sin(\mathbf{x})\big) \prod_{t=2}^{p} \cos^{m_t}(K_t\mathbf{x}).T_{j_t}\big(\cos(\mathbf{x})\big)\Big)$ for $\boldsymbol{j}, \boldsymbol{m} \in \mathbb{N}^p$ and again, for consistency, define: $\boldsymbol{\Delta}^{\boldsymbol{J}}_{\mathbf{0}}(\mathbf{X}) = \bar{\mathbf{S}}^{\boldsymbol{J}}(\mathbf{X}) = \{\{\bar{S}^{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})\}\}$.

Transformation 2: Again, for fixed $\boldsymbol{j_{-1}} = j_2, \cdots, j_p$, we have iterative relations:

$$\begin{aligned} \delta^{2m_1,\boldsymbol{j_{-1}}}_{2j_1+1,\mathbf{0}}(\mathbf{X}) &= 2\delta^{2m_1+1,\boldsymbol{j_{-1}}}_{2j_1,\mathbf{0}}(\mathbf{X}) - \delta^{2m_1,\boldsymbol{j_{-1}}}_{2j_1-1,\mathbf{0}}(\mathbf{X}) \\ \delta^{2m_1+1,\boldsymbol{j_{-1}}}_{2j_1,\mathbf{0}}(\mathbf{X}) &= 2\delta^{2m_1+2,\boldsymbol{j_{-1}}}_{2j_1-1,\mathbf{0}}(\mathbf{X}) - \delta^{2m_1+1,\boldsymbol{j_{-1}}}_{2j_1-2,\mathbf{0}}(\mathbf{X}) \end{aligned} \tag{3.40}$$

We apply EP-FQCh type A iteration along the first co-ordinate of $\boldsymbol{\Delta}^{\boldsymbol{J}}_{\mathbf{0}}(\mathbf{X})$ to get the statistics array $\boldsymbol{\Delta}^{0,J_2,\cdots,J_p}_{J_1,0,\cdots,0}(\mathbf{X}) = \{\{\delta^{0,j_2-1,\cdots,j_p-1}_{2j_1-1,0,\cdots,0}(\mathbf{X})\}\}$, for $\boldsymbol{j} \in \mathbb{J}$.

Transformation 3: For $t > 1$ we have iterative relation:

$$\delta^{\boldsymbol{m}}_{\boldsymbol{j}+e_t}(\mathbf{X}) = 2\delta^{\boldsymbol{m}+e_t}_{\boldsymbol{j}}(\mathbf{X}) + \delta^{\boldsymbol{m}}_{\boldsymbol{j}-e_t}(\mathbf{X}) \tag{3.41}$$

We apply EP-FQCh type B recursion along the co-ordinates $2, \cdots, p$ of $\boldsymbol{\Delta}^{0,J_2,\cdots,J_p}_{J_1,0,\cdots,0}(\mathbf{X})$ iteratively to get p dimensional statistics array $\boldsymbol{\Delta}^{\mathbf{0}}_{\boldsymbol{J}}(\mathbf{X}) = \{\{\delta^{0,0,\cdots,0}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})\}\}$.

Transformation 4: Finally, observe:

$$\begin{aligned} \delta^{0,0,\cdots,0}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X}) &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} T_{2j_1-1}\big(sin(\mathbf{x})\big) \prod_{t=2}^{p} T_{j_p-1}\big(cos(2K_p\mathbf{x})\big) \\ &= \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x}\in\mathbf{X}} (-1)^{j_1-1} \sin\big((2j_1-1)\mathbf{x}\big) \prod_{t=2}^{p} \cos\big((j_p-1)K_p\mathbf{x}\big) \end{aligned} \tag{3.42}$$

So, we multiply $\delta^{0,0,\cdots,0}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X})$ by $(-1)^{j_1-1}$ for $\boldsymbol{j} \in \mathbb{J}$ to get the p dimensional array $\bar{\mathbf{S}}_{\boldsymbol{J}}(\mathbf{X}) = \{\{\sin\big((2j_1-1)\mathbf{x}\big)\prod_{t=2}^{p}\cos\big((j_p-1)K_p\mathbf{x}\big)\}\}$. In the Sum of Sines identity in equation 3.24, we replace J with J_1, j with j_1 and k with j_2, multiply both sides with $\prod_{t=3}^{p}\cos^{j_t}(K_t\mathbf{x})$, then take a sum over $\mathbf{x} \in \mathbf{X}$, we get

$$\bar{S}_{2J_1j_2+2j_1-1,j_3-1,\cdots,j_p-1}(\mathbf{X}) - \bar{S}_{2J_1j_2-2j_1+1,j_3-1,\cdots,j_p-1}(\mathbf{X}) = 2\bar{S}_{2j_1-1,j_2-1,\cdots,j_p-1}(\mathbf{X}) \tag{3.43}$$

We apply EP-FQCh type D recursion along the first and second co-ordinates of $\bar{\mathbf{S}}_{\boldsymbol{J}}(X)$, to get $p-1$ dimensional array statistic $\bar{\mathbf{S}}_{J_1J_2,J_3,\cdots,J_p}(\mathbf{X}) = \{\{\bar{S}_{2j_1-1,j_3-1,\cdots,j_p-1}(\mathbf{X})\}\}$, here $(j_1, j_3, \cdots, j_p) \in \{1,\cdots,J_1J_2\} \times \prod_{t=3}^{p}\{1,\cdots,J_t\}$. Repeat this process another $p-2$ times, each time applying EP-FQCh type D recursion on the resulting array to get final statistics vector $\bar{\mathbf{S}}(\mathbf{X}) = \{\bar{S}_{2j-1}(X)\}$, $j = 1,\cdots,J_0$, $J_0 = J_1J_2\cdots J_p$.

Again for optimum result, we have the choice $2J_1 = J_2 = \cdots = J_p$. After the D&R step, for each quantile p we maximize the optimization criteria in equation 3.7 to get corresponding $\hat{Q}_P^{J_0}(\mathbf{X})$. For an algorithmic flow chart, see Appendix B.5.

3.7 Performance Study

3.7.1 Data generation

In this section we are going to demonstrate relative performances of EP-FQChp and Binning method, when computing a set of sample quantiles for a very large data. Realize that it is impossible to know exact sample quantiles for any arbitrarily large simulated data that is bigger than memory and stored in a cluster. We can't compare accuracy if we don't know the exact sample quantiles, both EP-FQChp and Binning gives approximate sample quantiles as output.

To deal with this problem, we don't simulate randomly but generate large data in a way, so that we already know the exact quantile values. We choose an invertible continuous distribution function $F(\mathbf{x})$ and a very large integer N. Our data $\mathbf{X}$ is a permutation of the numbers $F^{-1}(\frac{1}{N+1}), F^{-1}(\frac{2}{N+1}), \cdots, F^{-1}(\frac{N}{N+1})$. These N numbers

are population qunatiles of F, for N equidistant f-values in $(0, 1)$. Then, for a large N, the Pth sample quantile $Q_P(\mathbf{X}) \approx F^{-1}(P)$, so, we know the exact sample quantiles of $\mathbf{X}$. The permutation step is required for fair run-time comparison, as Binning is much faster for sorted data.

We generate data for 10 distributions, eg Normal(0,1),Uniform(0,1),Chi-Squared(4), Chi-Squared(8),Chi-Squared(16),Beta(1,4),Beta(1,8),Beta(1,16),t(4) and t(8), to demonstrate relative performance of EP-FQ and Binning. For each distribution, we simulate $N = 179 \times 173 \times 100,000 (\approx 3$ billion) observations. We divide the data into 179 blocks in HDFS, each block contains 173 subsets, and each subset has $100,000$ observations. This generation process creates blocks of size 127.3 MB, so we are close to the range of it Cloudera recommended block size for optimum Hadoop job. The Hadoop cluster we use has 200 nodes and number of blocks is chosen to be $200 - 1 = 199$, so that each container gets one process to run at a time.

3.7.2 Accuracy comparison

For P^{th} sample quantile $Q_P(\mathbf{X})$ of data $\mathbf{X}$, let the EP-FQ approximate quantile is $\hat{Q}_P^F(\mathbf{X})$ and the Binning approximate quantile is $\hat{Q}_P^B(\mathbf{X})$. Let us define the EP-FQ error for P^{th} quantile: $\varepsilon_P^F(\mathbf{X}) = \log_{10} |\hat{Q}_P^F(\mathbf{X}) - Q_P(\mathbf{X})|$ and the Binning error for P^{th} quantile: $\varepsilon_P^B(\mathbf{X}) = \log_{10} |\hat{Q}_P^B(\mathbf{X}) - Q_P(\mathbf{X})|$. (We consider error in $\log_{10}$-scale because there errors are extremely small and difficult to compare in original scale.)

We need to compare these two variables. To numerically compare these two variables we take their difference $\varepsilon_P^B(\mathbf{X}) - \varepsilon_P^F(\mathbf{X})$ and plot against values of P. If it is positive, then Binning error is bigger and EP-FQ is more accurate. So beside plotting this difference, we also write down the number of times EP-FQ is more accurate. We call this number Fourier Success Rate or FSR.

To judge whether statistically the variable $\varepsilon_P^F(\mathbf{X})$ is bigger than $\varepsilon_P^B(\mathbf{X})$ we look at their pairwise QQ-plot.

Comparison of Accuracy : Normal(0,1)

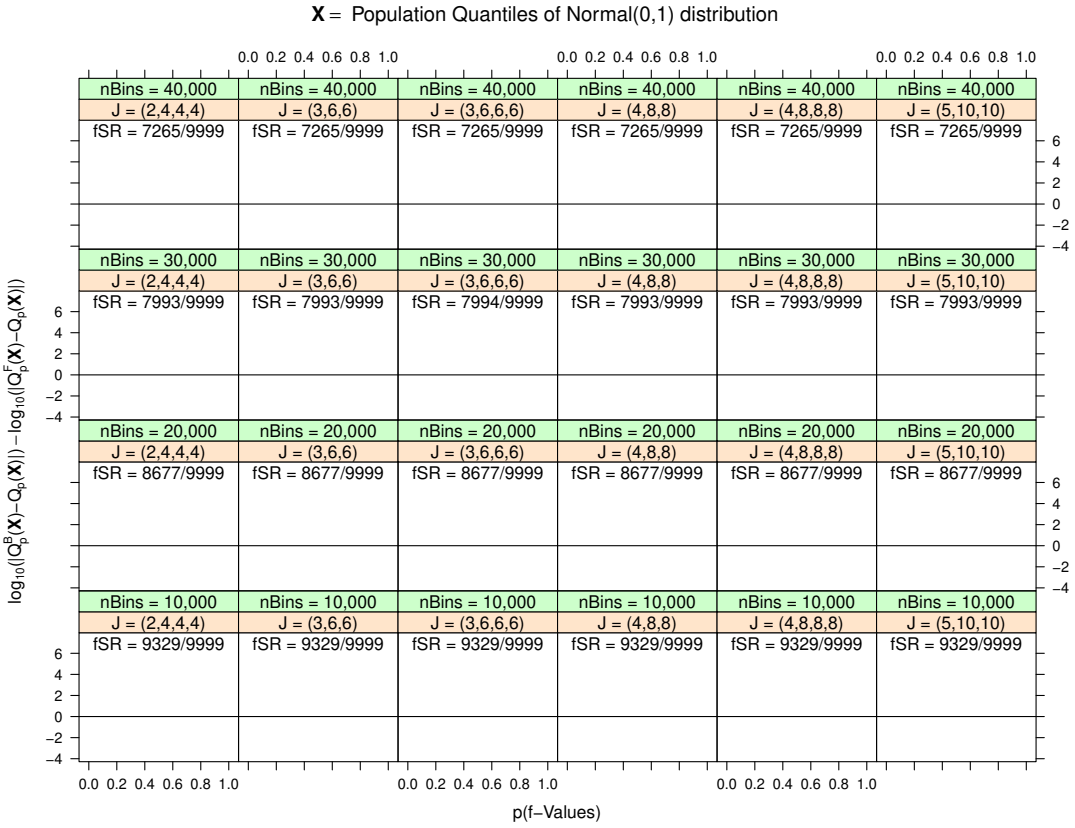

Figure 3.3. Accuracy for Normal(0,1) data: xy-plot of error differences

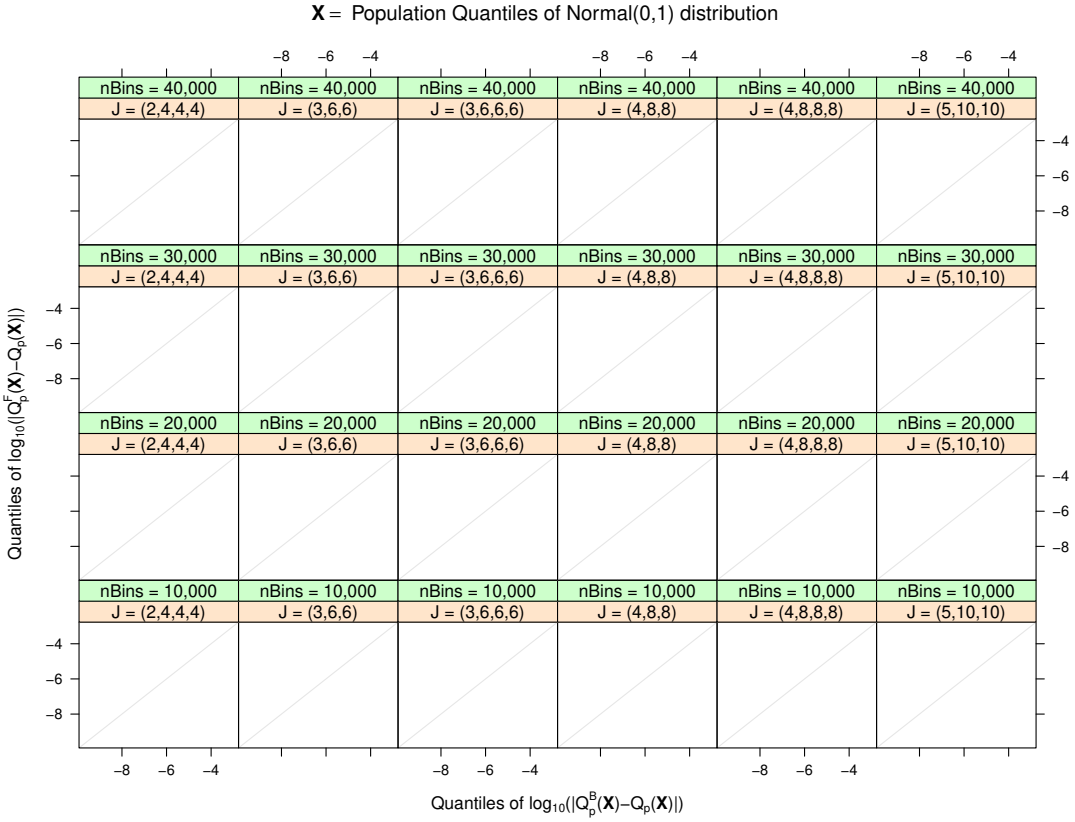

Figure 3.4. Accuracy for Normal(0,1) data: QQ-plot of errors

Comparison of Accuracy : Uniform(0,1)

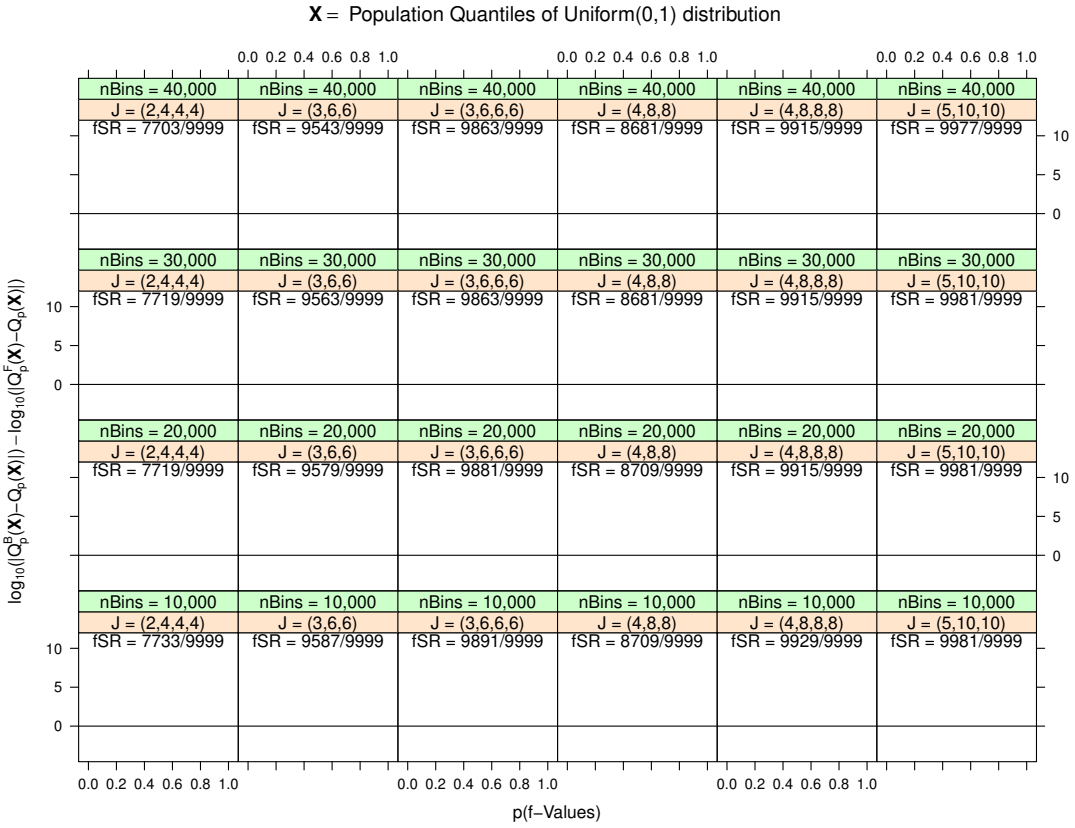

Figure 3.5. Accuracy for Uniform(0,1) data: xy-plot of error differences

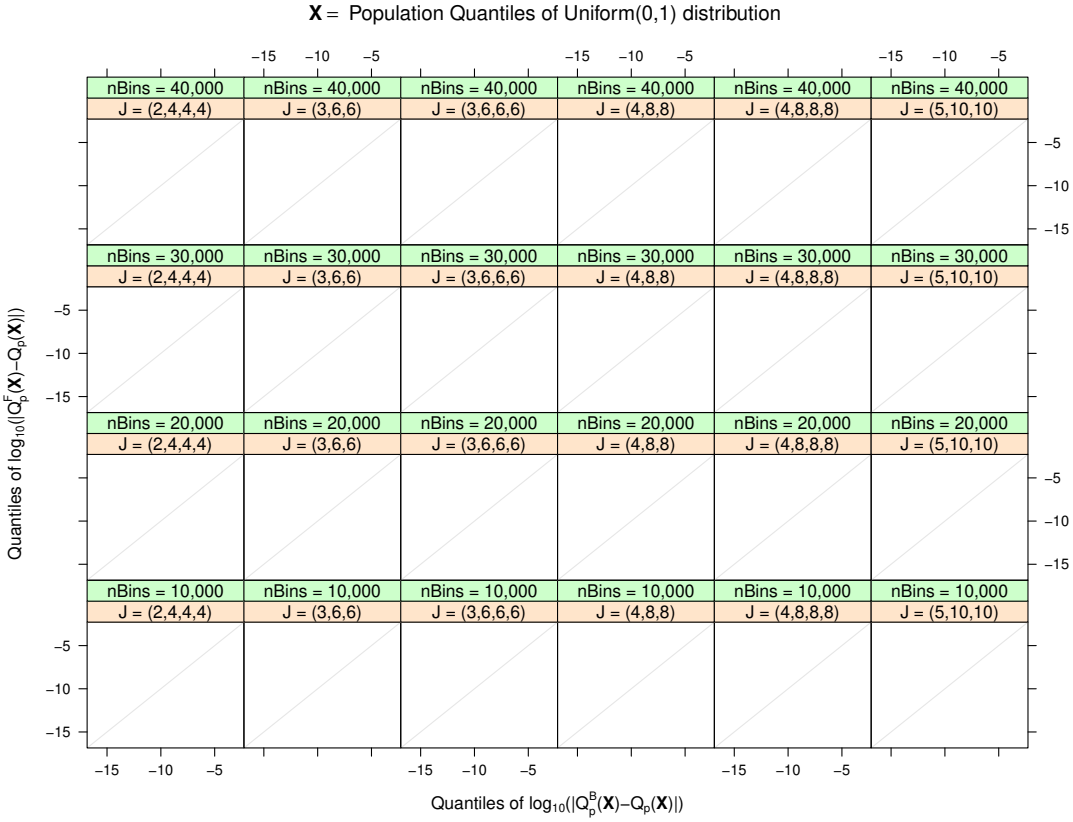

Figure 3.6. Accuracy for Uniform(0,1) data: QQ-plot of errors

Comparison of Accuracy : Chi-Squared(4)

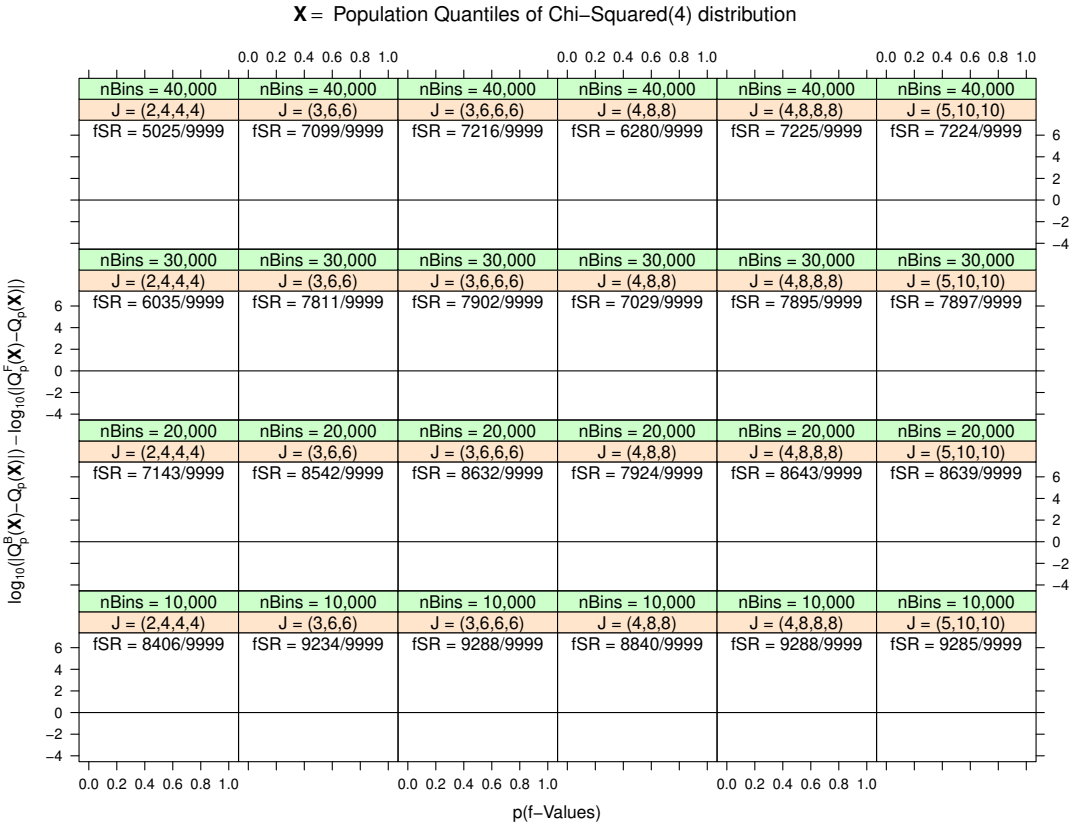

Figure 3.7. Accuracy for Chi-Squared(4) data: xy-plot of error differences

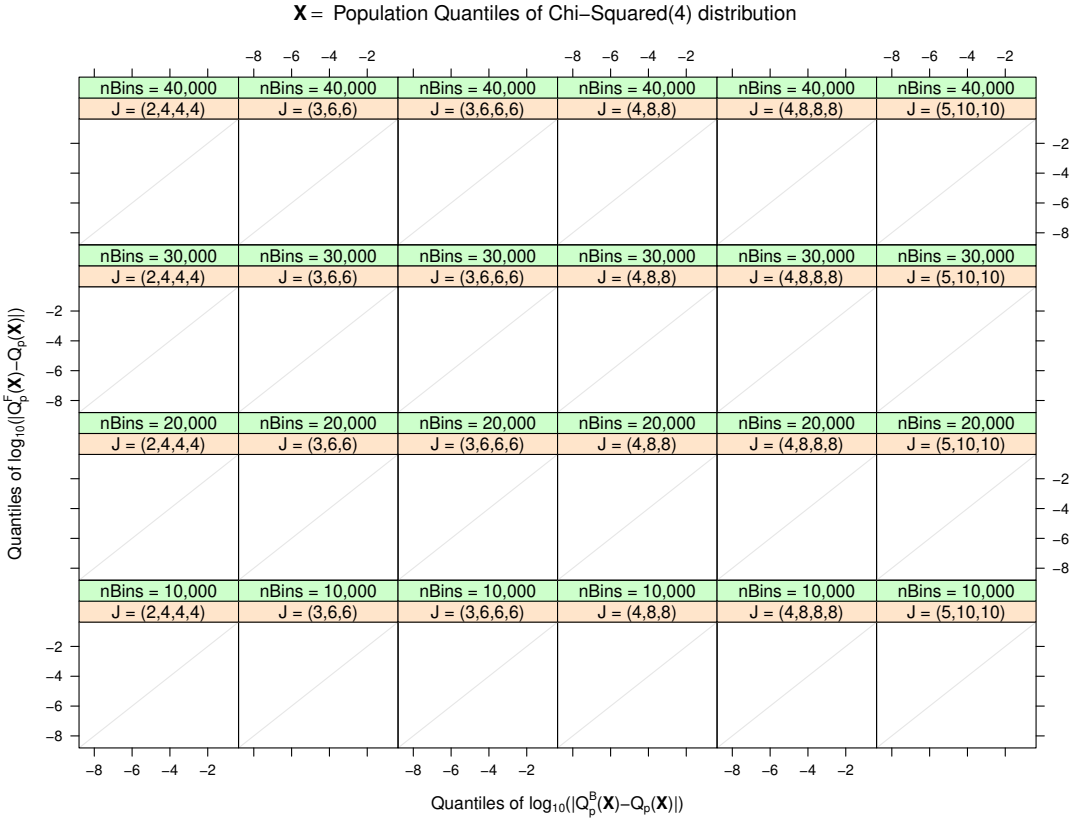

Figure 3.8. Accuracy for Chi-Squared(4) data: QQ-plot of errors

Comparison of Accuracy : Chi-Squared(8)

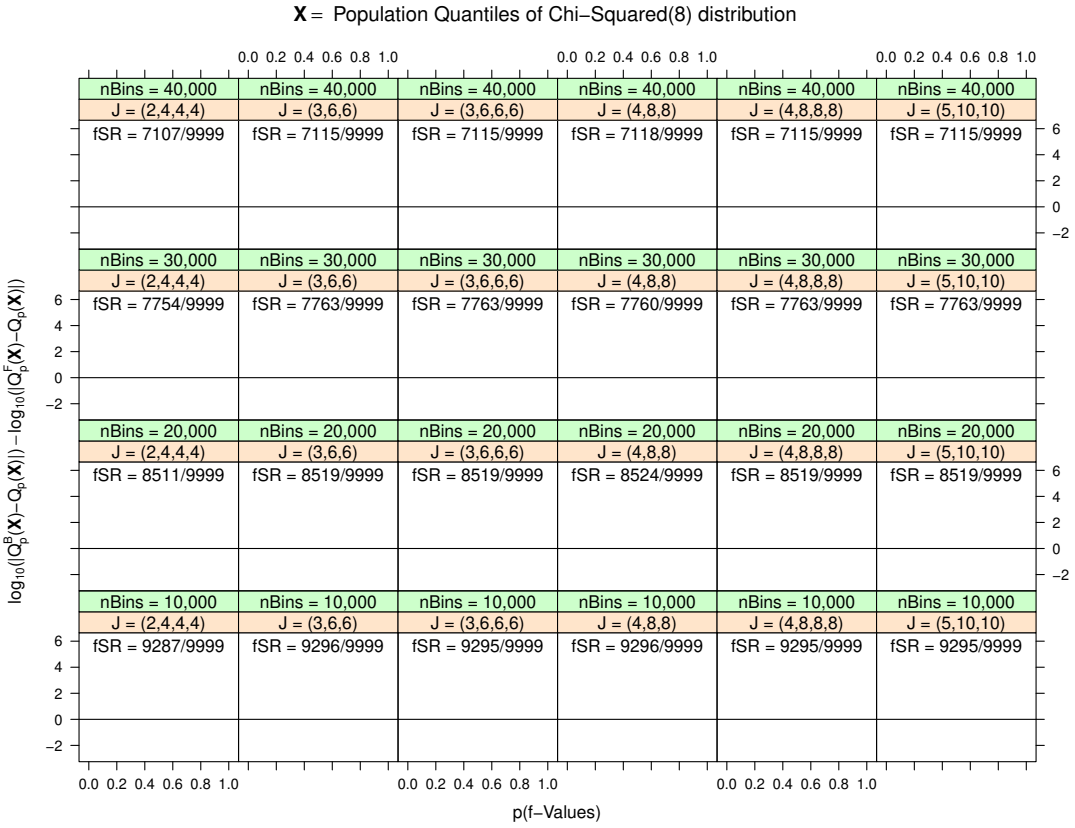

Figure 3.9. Accuracy for Chi-Squared(8) data: xy-plot of error differences

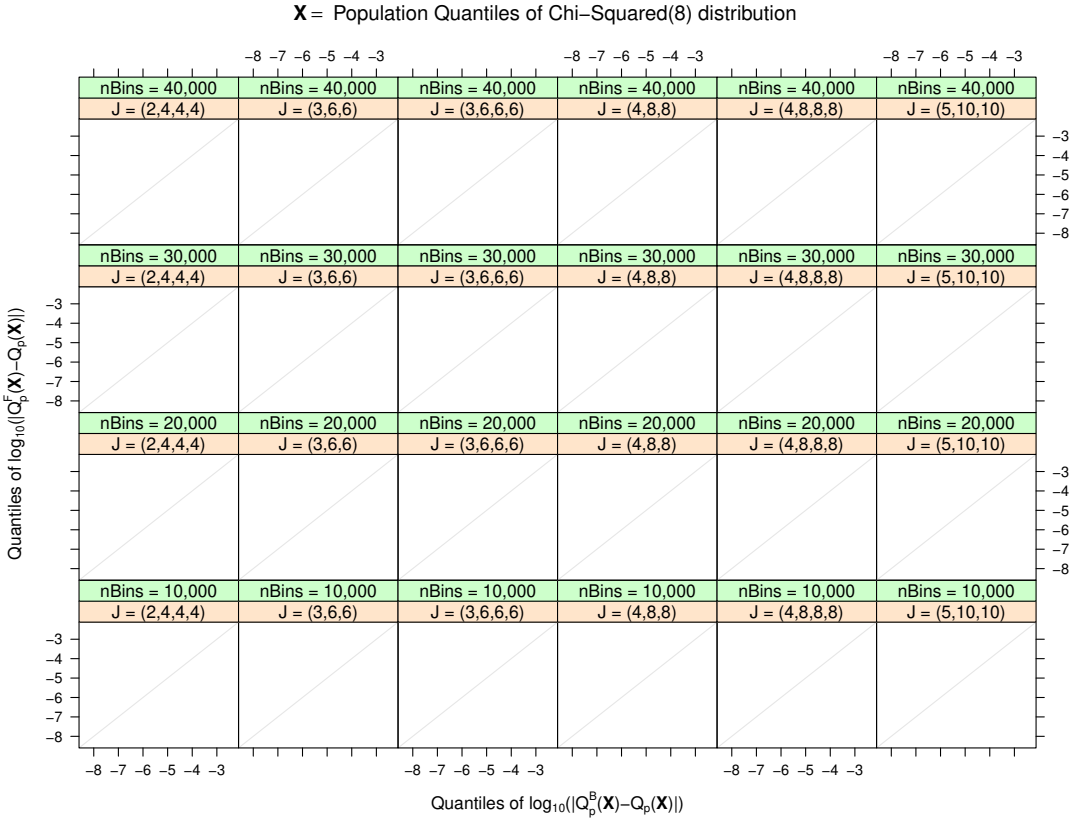

Figure 3.10. Accuracy for Chi-Squared(8) data: QQ-plot of errors

Comparison of Accuracy : Chi-Squared(16)

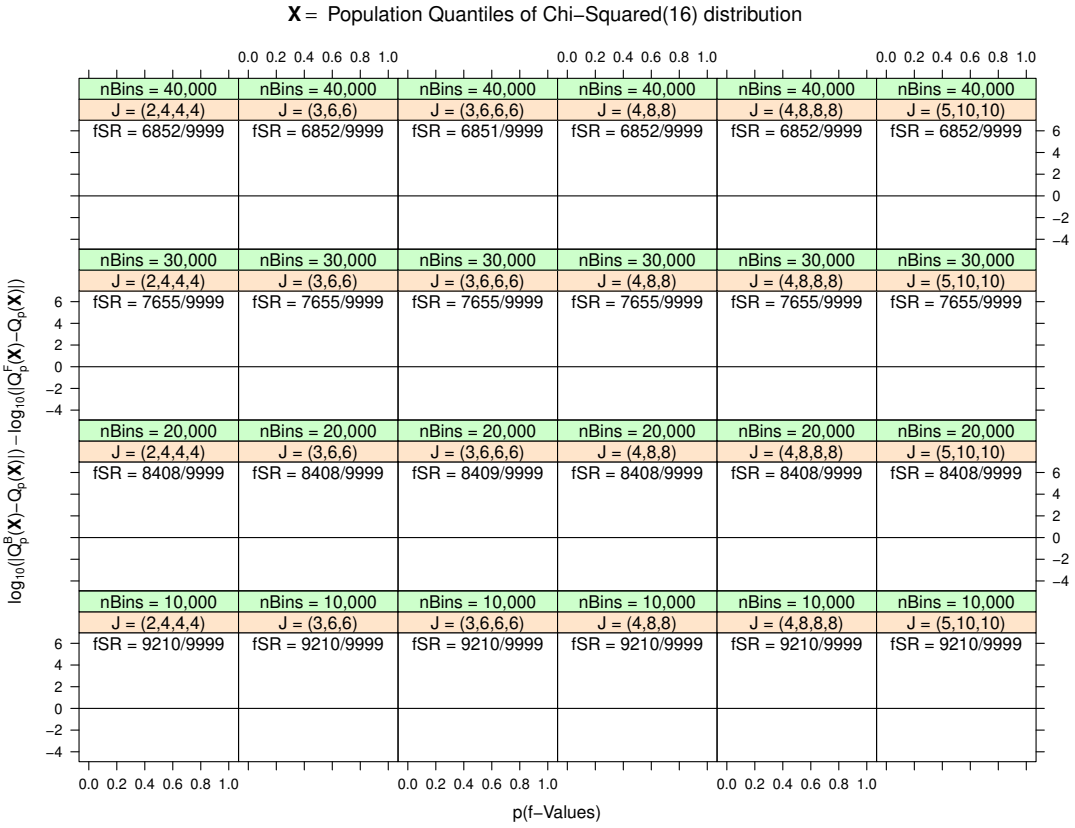

Figure 3.11. Accuracy for Chi-Squared(16) data: xy-plot of error differences

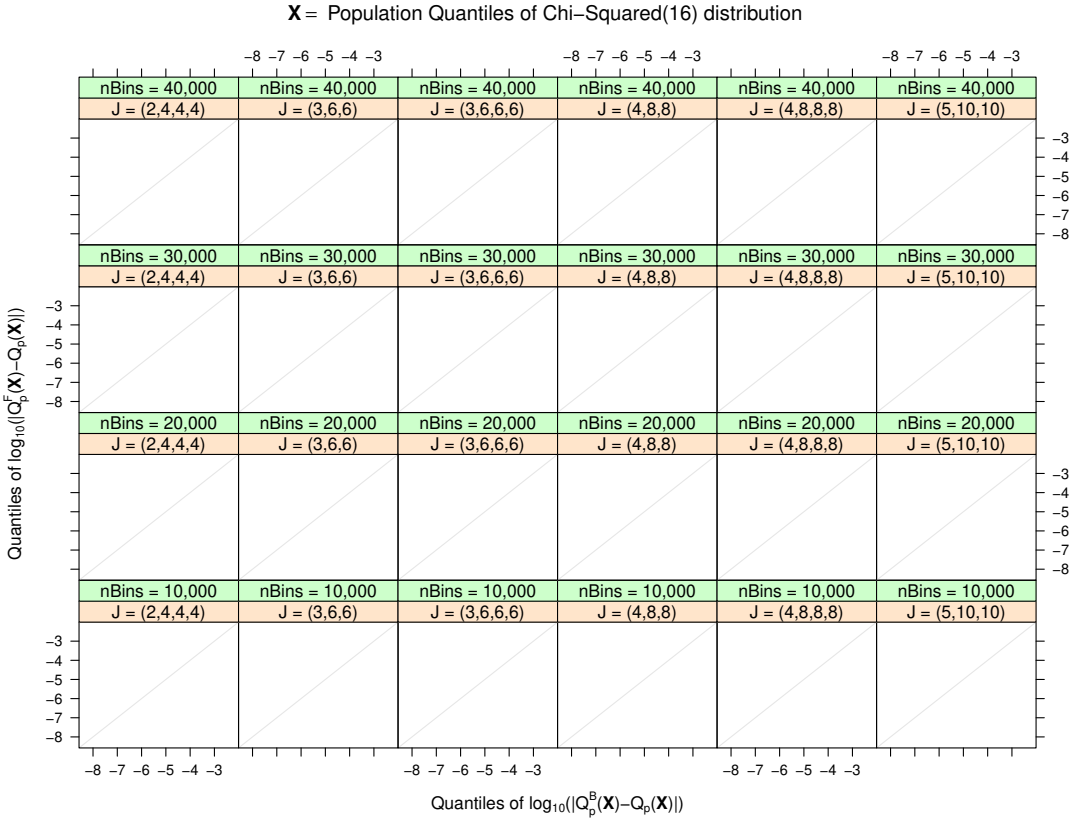

Figure 3.12. Accuracy for Chi-Squared(16) data: QQ-plot of errors

Comparison of Accuracy : Beta(1,4)

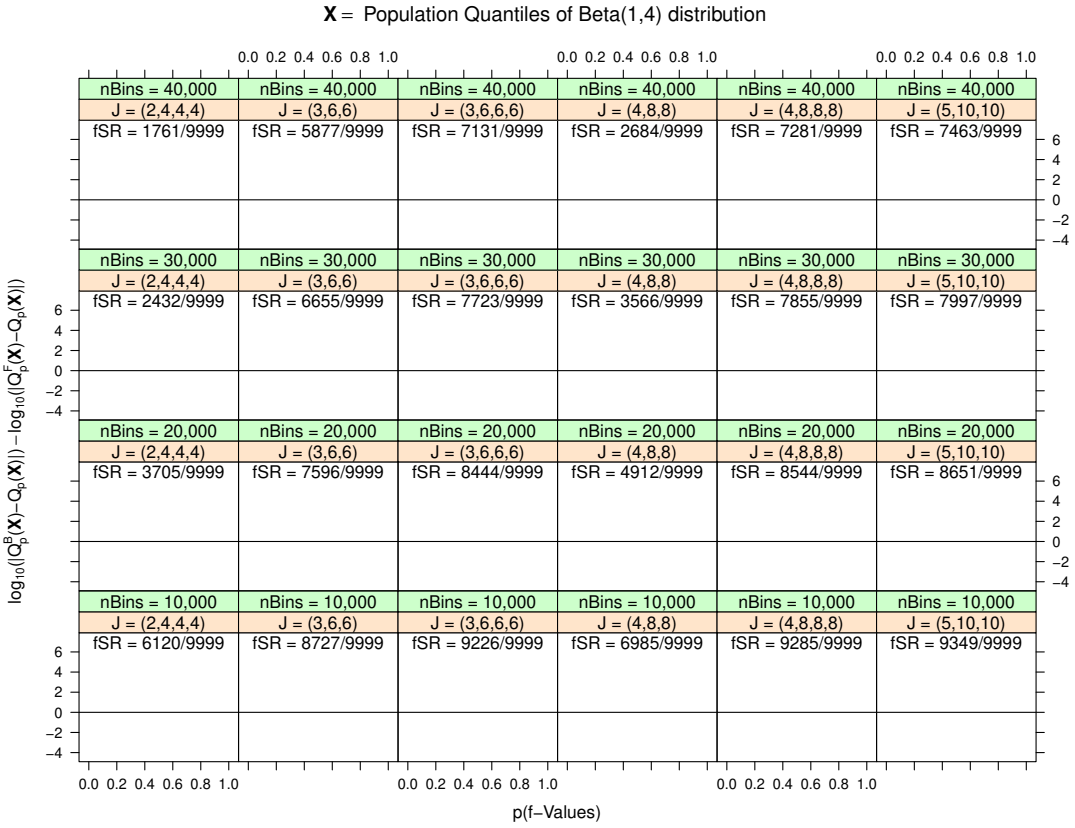

Figure 3.13. Accuracy for Beta(1,4) data: xy-plot of error differences

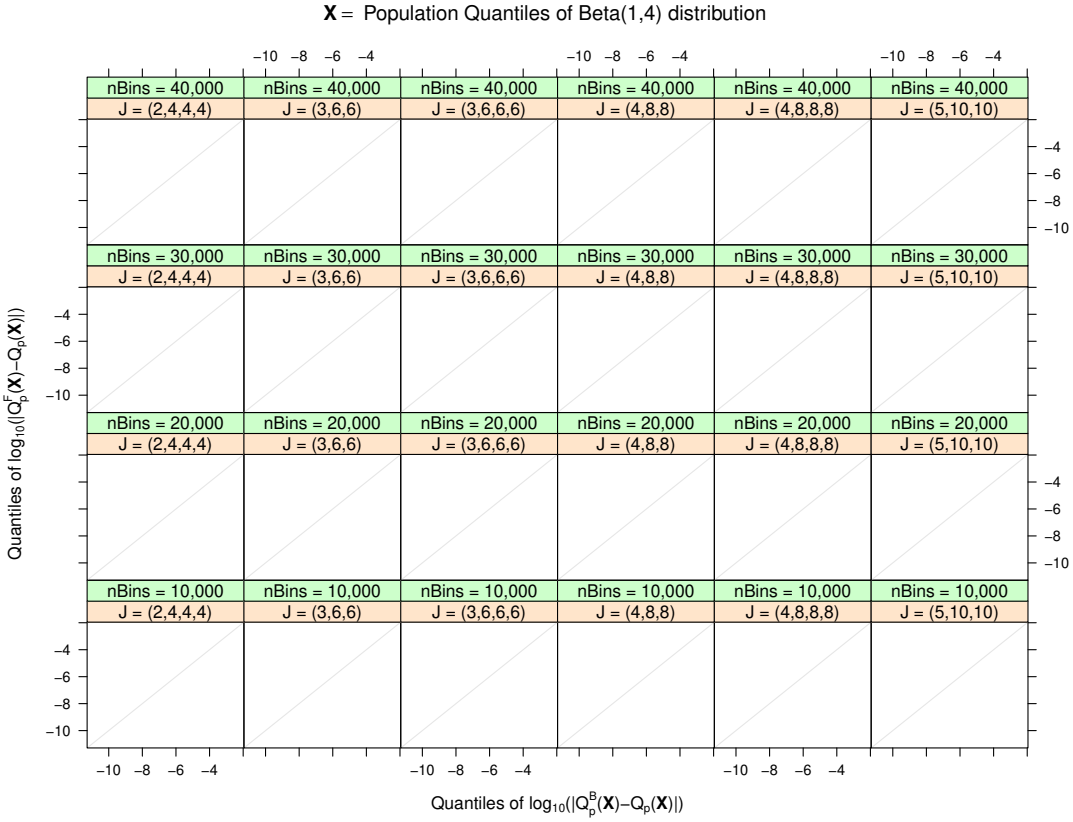

Figure 3.14. Accuracy for Beta(1,4) data: QQ-plot of errors

Comparison of Accuracy : Beta(1,8)

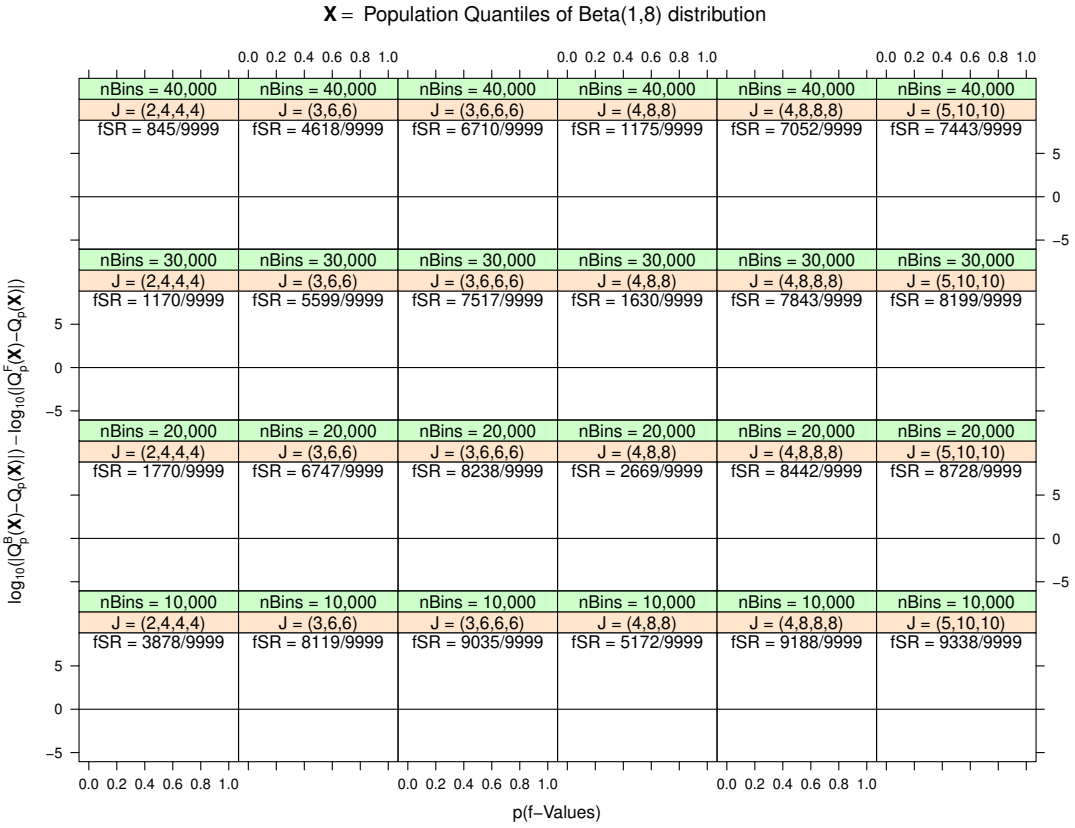

Figure 3.15. Accuracy for Beta(1,8) data: xy-plot of error differences

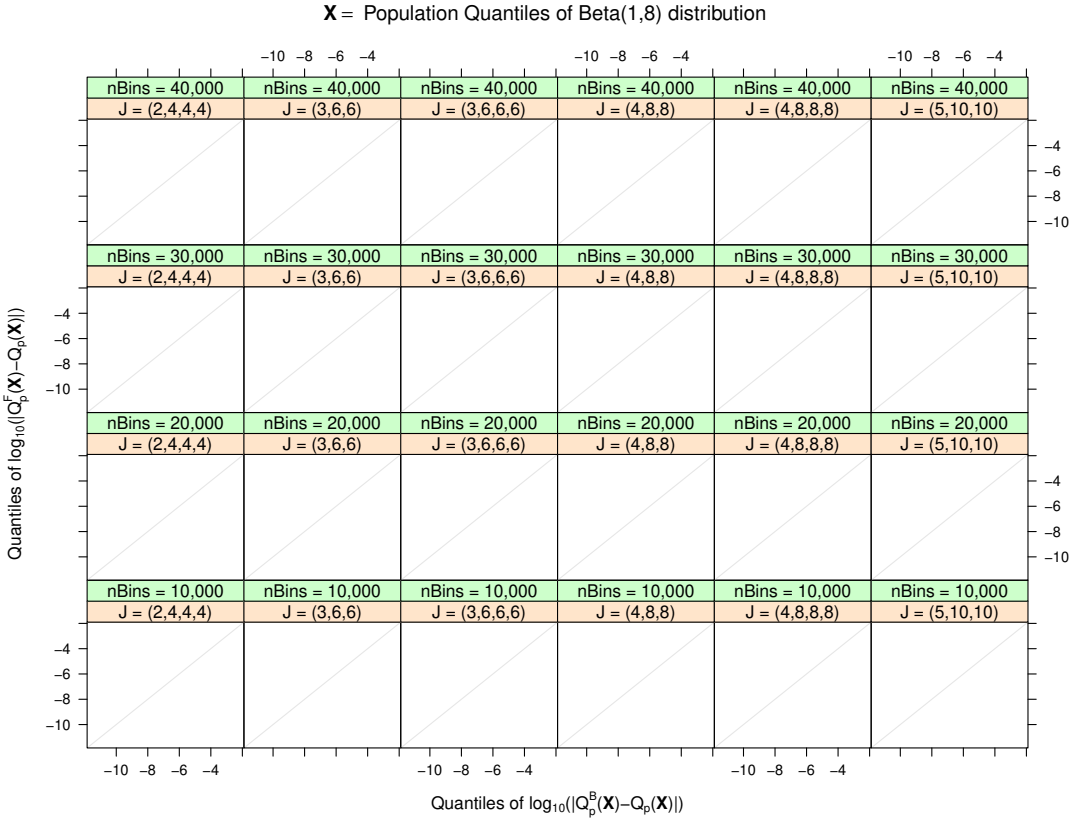

Figure 3.16. Accuracy for Beta(1,8) data: QQ-plot of errors

Comparison of Accuracy : Beta(1,16)

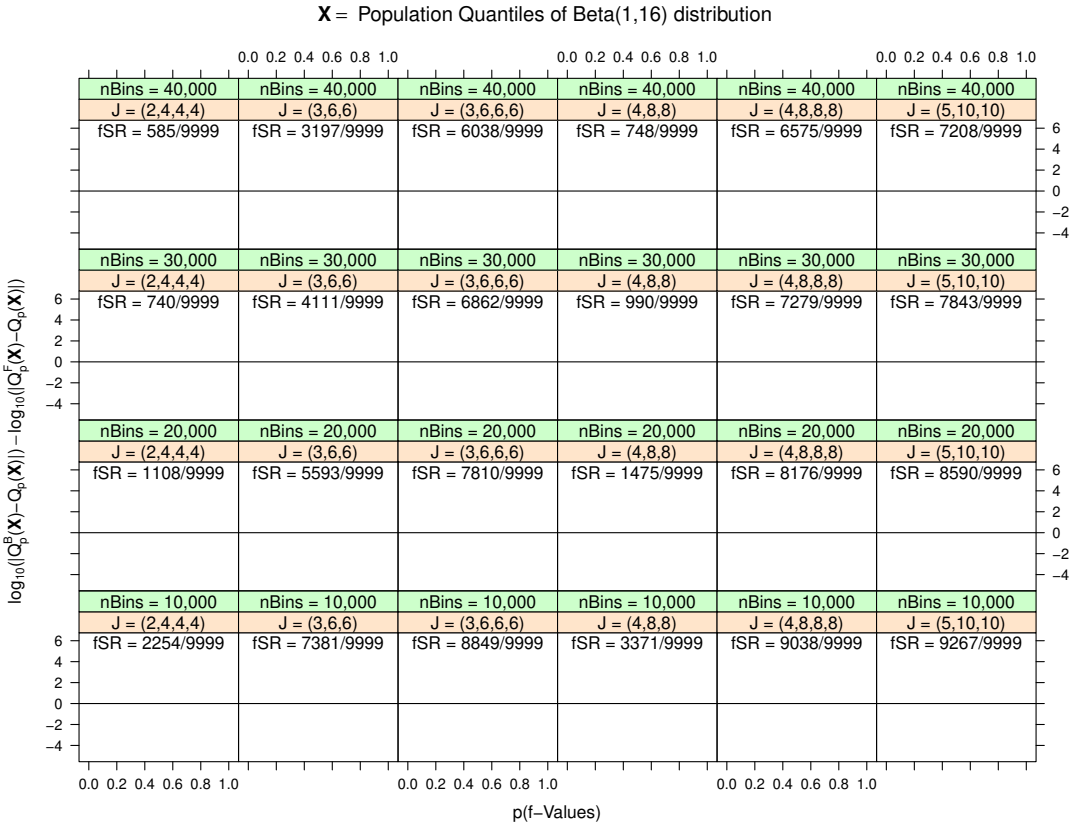

Figure 3.17. Accuracy for Beta(1,16) data: xy-plot of error differences

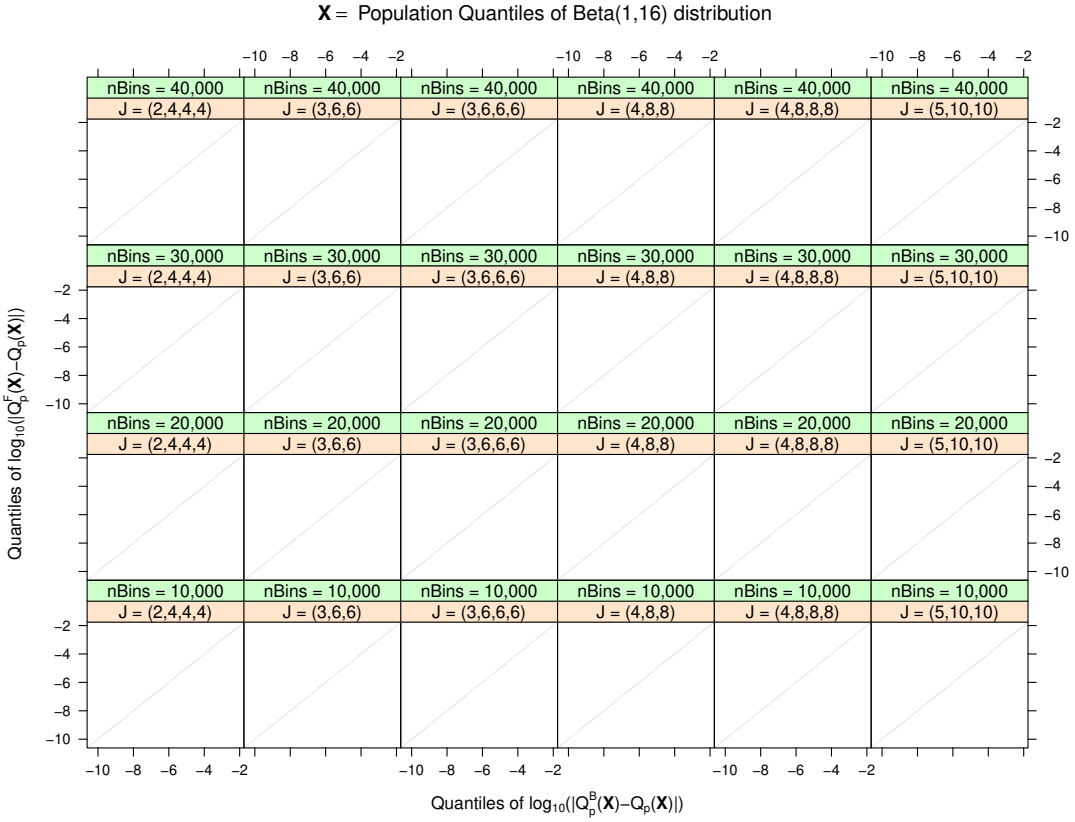

Figure 3.18. Accuracy for Beta(1,16) data: QQ-plot of errors

Comparison of Accuracy : t(4)

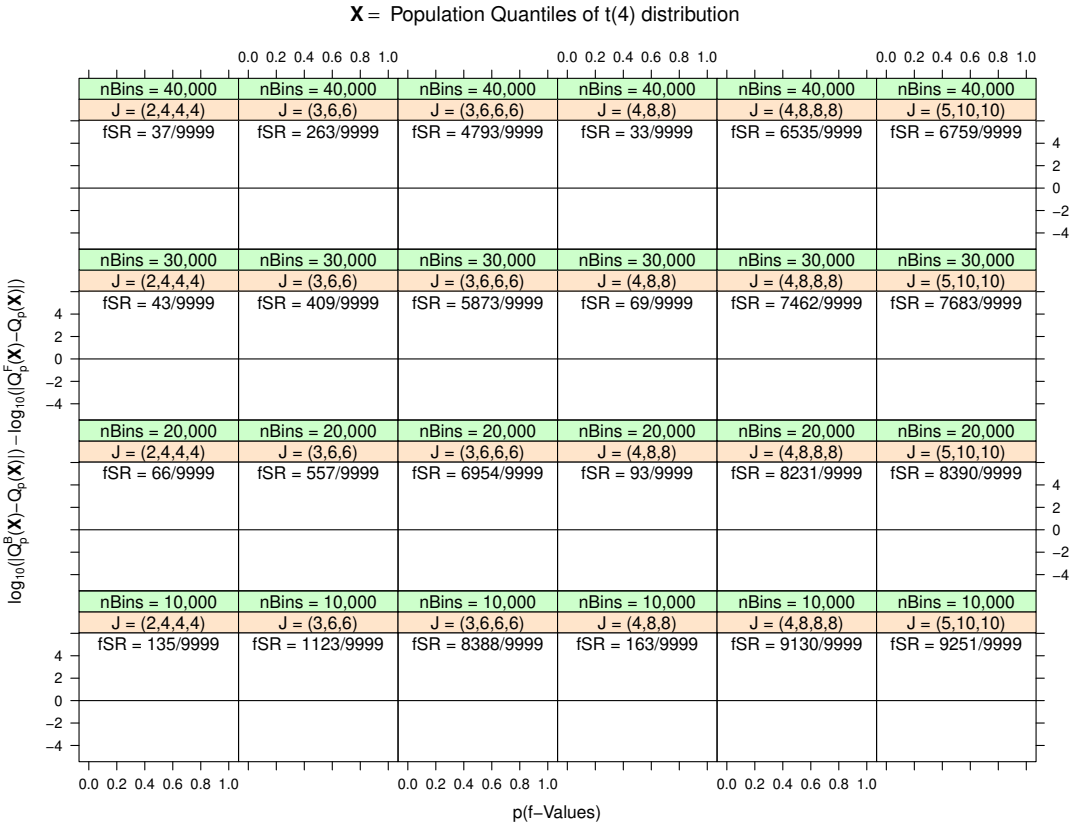

Figure 3.19. Accuracy for t(4) data: xy-plot of error differences

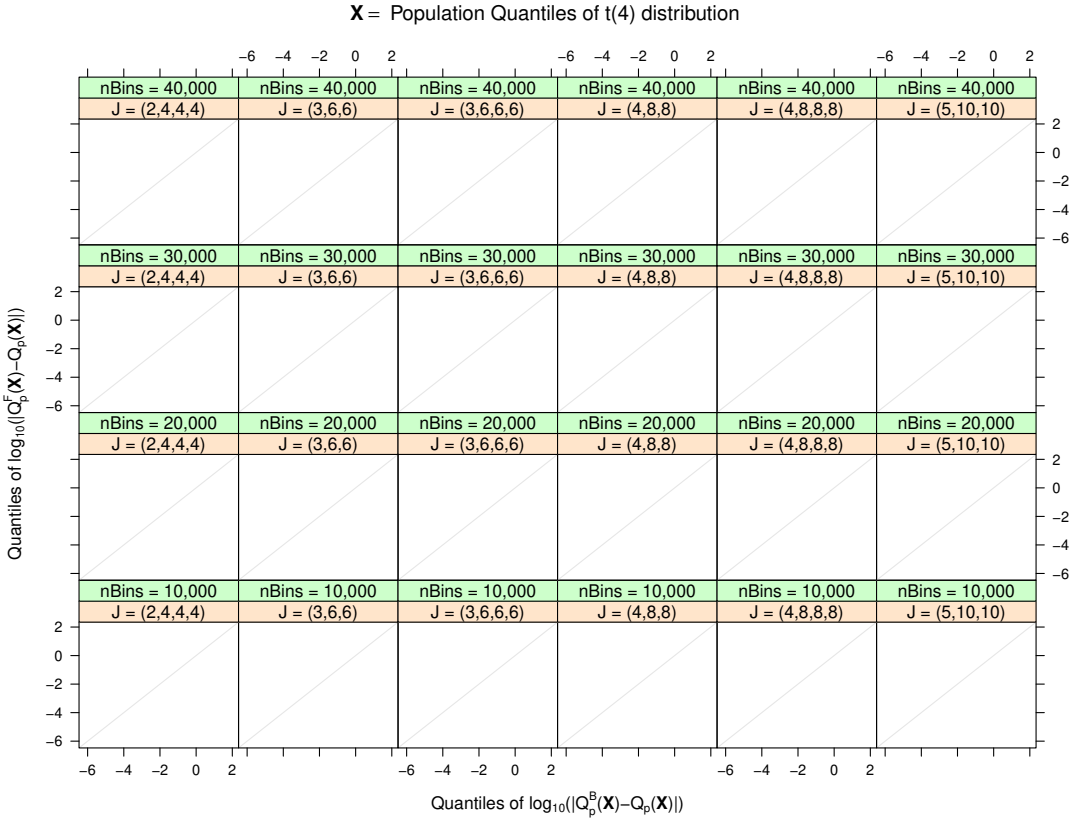

Figure 3.20. Accuracy for t(4) data: QQ-plot of errors

Comparison of Accuracy : t(8)

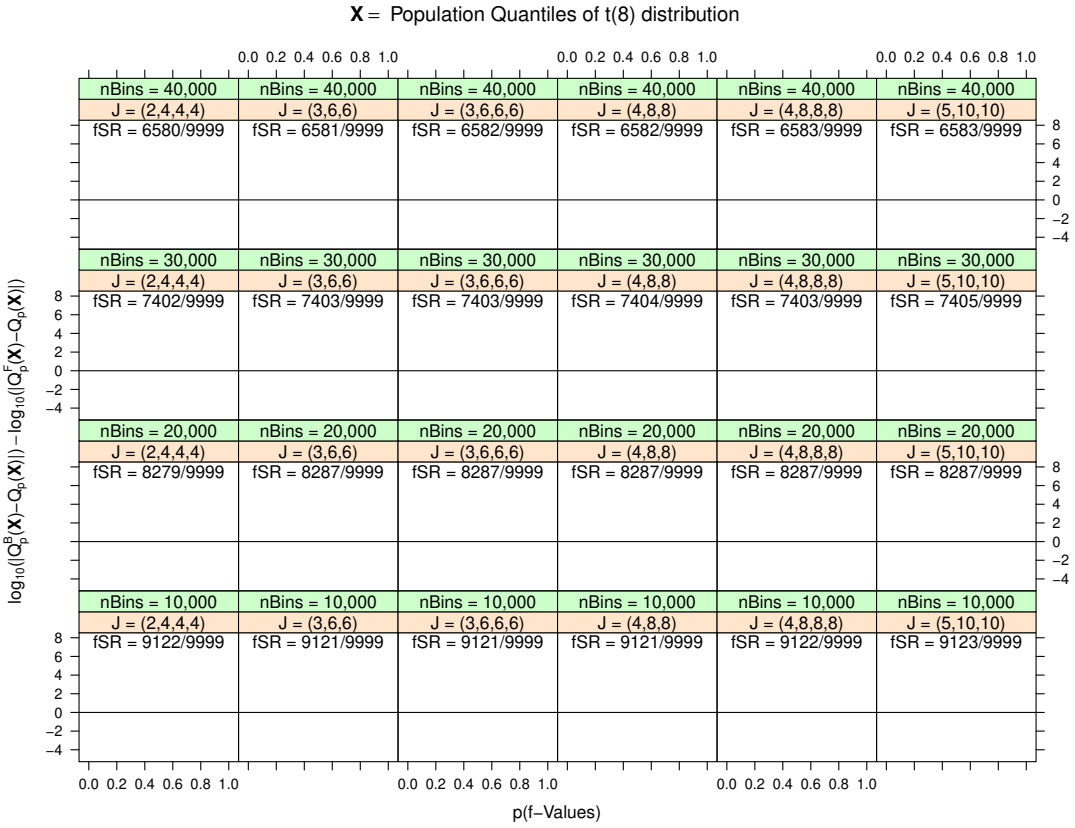

Figure 3.21. Accuracy for t(8) data: xy-plot of error differences

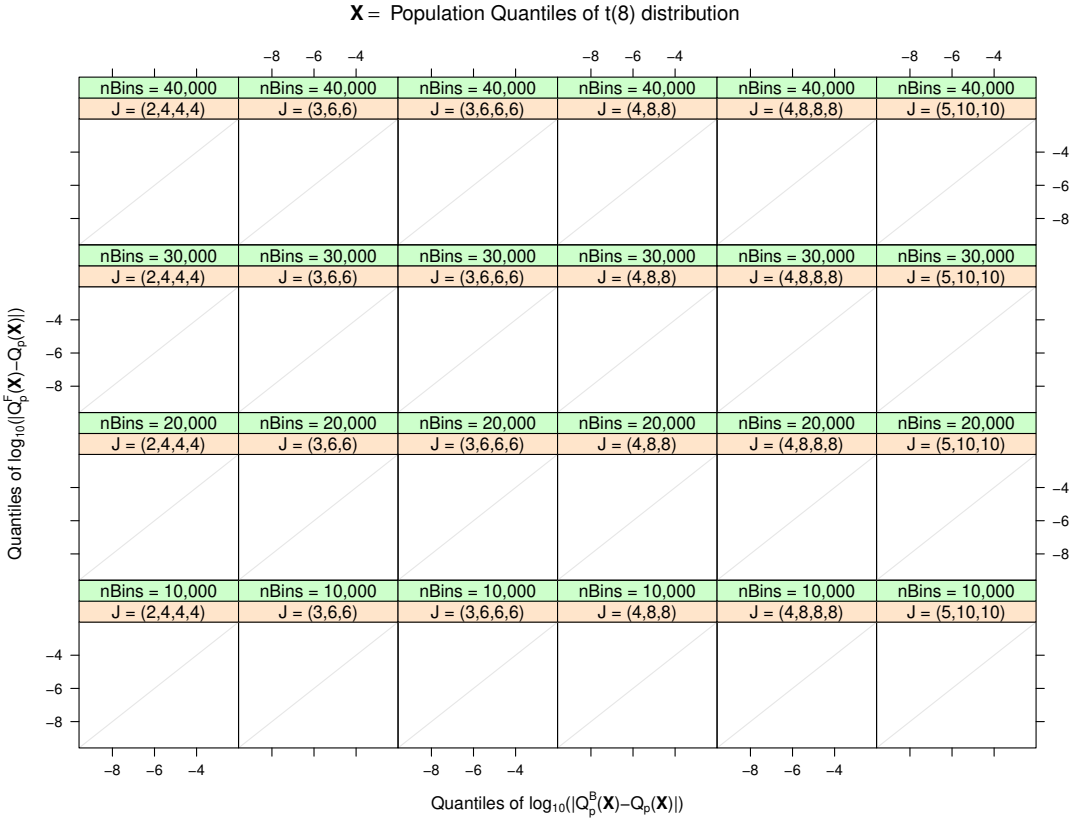

Figure 3.22. Accuracy for t(8) data: QQ-plot of errors

3.7.3 Runtime comparison

Comparison of Run-time:

Here is a plot demonstrating run-time for EP-FQ and Binning methods

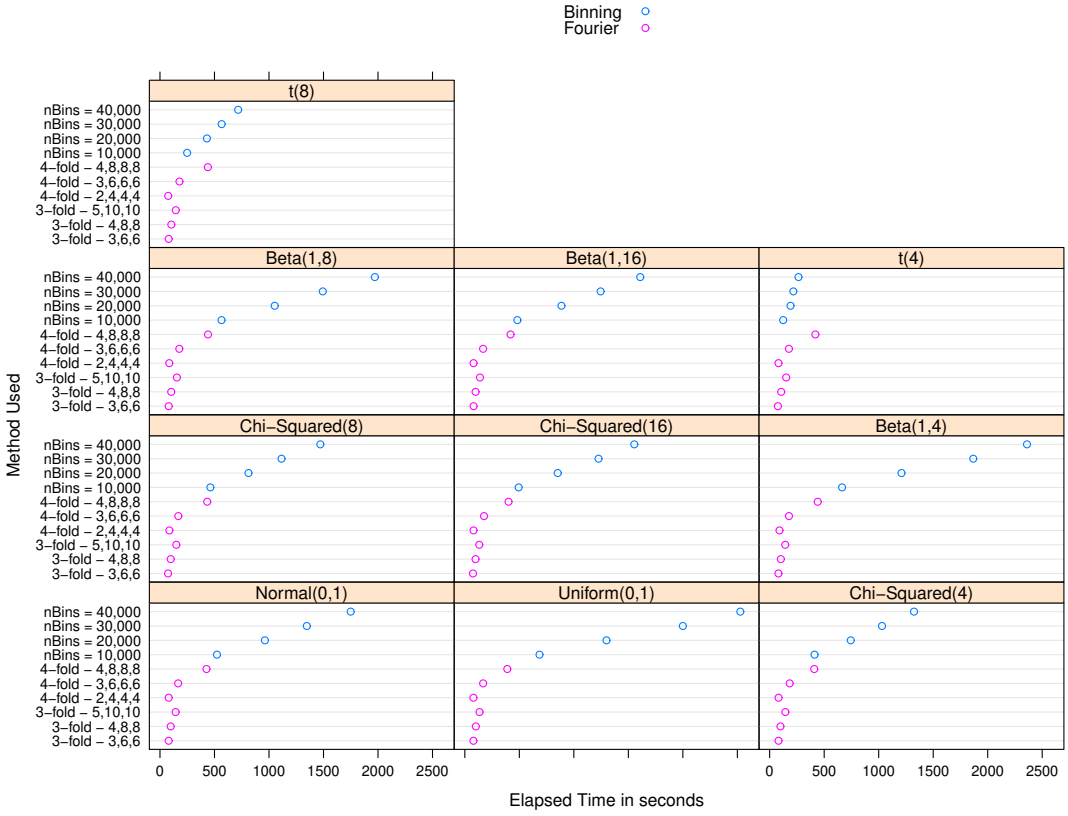

Figure 3.23. Run-time for Quantile computations

3.8 Discussion

A number of methods have been implemented for computing quantiles for Big Data. The GK algorithm [3]is a deterministic ϵ approximation method that computes quantiles with a summary size bounded in $\frac{1}{\epsilon}\log \epsilon N$, realize that the summary grows as N gets bigger, and in EPS-FQ we will see that the summary does not depend on N, which is a strong property. Z.huang et all [4] developed a sampling algorithm for sensor networks, but for sampling algorithms there is some sampling error, big data is always finite sample and local information is important. Another sampling algorithm is Random Mergeable Summaries [5]. The Q-digest algorithm developed by Shrivastava et all [6] is another deterministic quantile computation method for Big Data, however it only works for integers and have a fixed universe. Ted Dunning further developed this algorithm in his t-digest algorithm, which assigns weight on the accuracy, making it more accurate near quantile values close to zero or one. At the median it becomes highly inaccurate, which is not always desired for example we might be interested to construct KD tree for distributed data, and we need accurate median computation. So we are looking for algorithm that can be carried out in parallel with minimal interaction, uses information from all of the data equally, the summary data structure does not grow with N and there is a way to control the accuracy and $EP - FQ$ serves the purpose. It has been proved theoretically by Manku et all [7] that exact quantile computation requires $O(N \log N)$ computation time, it is not possible to have a linear exact algorithm for quantile computation.

4. THE EP-FOURIER-KD-TREE ALGORITHM

In machine learning, a KD-tree or K Dimensional tree [8] is a data structure that partitions a k-dimensional space based on the data. KD-trees have several applications, like range search in a statistical decision tree, clustering, construction of nearest neighbors, local regression etc. The KD-tree is a actually a generalization of binary tree in which every node is a k-dimensional point. Like binary trees, every non-leaf node divides the space into two parts, known as half-spaces. This node acts as a boundary point for the two half-spaces. Points to the left of this boundary along a specific chosen co-ordinate are represented by the left sub-tree of that node and points right of this boundary are represented by the right sub-tree. So, for example, if for a particular node or boundary the x coordinate is chosen, all points in the sub tree with a smaller x value than the node will appear in the left sub-tree and all points with larger x value will be in the right sub-tree.

4.1 Cannonical construction of KD-tree

The canonical method of KD-tree construction involves computation of median in each level of construction. As we move down the levels, we choose data variables periodically to select the splitting planes. For example, in a 3-dimensional tree, the root would have an x-aligned plane, the root's immediate children would have y-aligned planes, the root's grandchildren would all have z-aligned planes and again the root's great-grandchildren would all have x-aligned planes, and so on. We split planes along the median of the selected variable for canonical construction. Note that, we read the entire data into the algorithm up-front as we require the medians. Choosing the medians as splitting planes constructs a balanced KD-tree, each leaf node is approximately the same distance from the root.

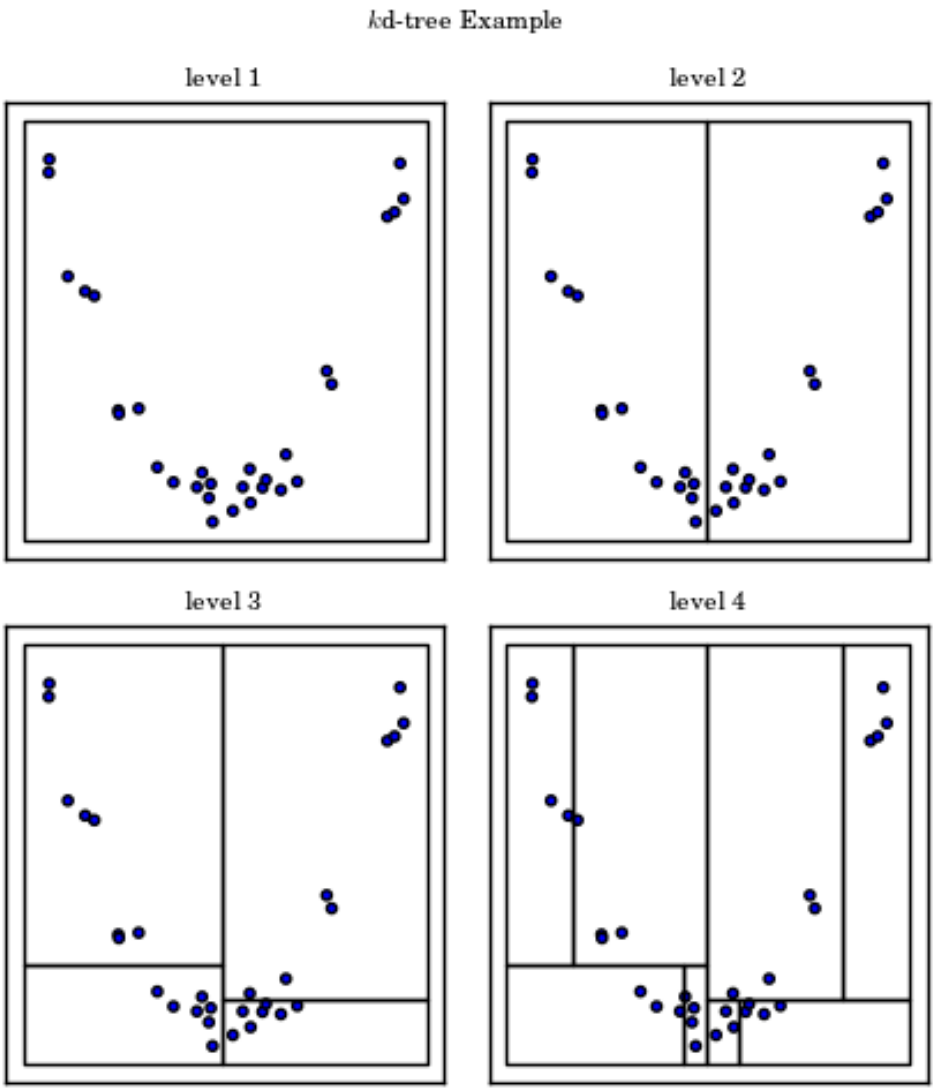

Figure 4.1. Canonical KD-tree construction

This figure illustrates construction of a KD-tree in 2 dimensional data. Initially at level 1, all data-points lie in the big-rectangle . First cut is made at the median of x co-ordinates of the points and the cut produces two new rectangles. Then, at level 2, we look at all points on the left rectangle and the second cut is made at the median of y co-ordinate of the points present in this rectangle. Similarly we divide the right rectangle. At level 3, again we choose coordinate x to make the split. Observe that in

this process, the rectangle or cell-counts are almost equal in each level, which makes the KD-tree balanced.

Building a balanced KD-tree from N points has the worst-case complexity of $O(N \log^2 N)$ if an $O(N \log N)$ sort like Heap-sort/Merge-sort is used to find the median at each level of construction. If an $O(N)$ median of medians algorithm is used then the worst case complexity is $O(N \log N)$. If we choose to presort these N points in each of p dimensions using an $O(N \log N)$ sort, then the worst-case complexity becomes $O(pN \log N)$. However, for distributed big data the problem becomes much more challenging, as it is impractical to implement these sorting algorithms by D&R.

In this chapter we provide a $O(N)$ D&R algorithm to find an approximate balanced KD-tree. The splitting median-points or the cell-boundaries are approximated and the cell-counts still stays almost equal, which serves the purpose of a balanced KD-tree.

4.2 Basic idea

In this section, we try to understand KD-tree construction procedure in terms of boundary points or vertices and neighborhood or cells. Let us consider the usual construction algorithm of a KD-tree. It is a recursive algorithm, in each recursion, we get a new level or set of nodes, from the previous level or set of nodes. These nodes can be also thought in terms vertices of neighborhoods. Suppose we are at a certain level d of the KD-tree, and we want to get the vertices for the next level. We have a number of neighborhoods in this level, and if we get the new vertices corresponding to each of these neighborhoods, we can easily get all the sub-neighborhoods for the next level $d + 1$.

First, let us figure out how to get the vertex along which we divide a neighborhood. Let a general neighborhood at depth d is $(\mathbf{a}, \mathbf{b}) = \big((a_1, \cdots, a_p), (b_1, \cdots, b_p)\big)$. Suppose, at depth d we want divide the data along the variable x_t. Then, we have to get the median $m_t(\mathbf{a}, \mathbf{b})$ along the variable x_t for all the observations $\mathbf{x}$, that lie

inside this interval. To get this median, we have to minimize $\sum |x_t - m|$ w.r.t. m subject to $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ (We are considering coordinate -wise inequality.)

Equivalently we have to minimize $\sum_{\mathbf{x} \in \mathbf{X}} |x_t - m| \prod_{l=1}^{p} I(a_l \leq x_l \leq b_l)$ w.r.t. m. If we define,

$$I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = |x_t - m| \prod_{l=1}^{p} I(a_l \leq x_l \leq b_l) \tag{4.1}$$

then we can get $m_t(\mathbf{a}, \mathbf{b})$ as:

$$m_t(\mathbf{a}, \mathbf{b}, \mathbf{X}) = \underset{m \in (a_t, b_t)}{\arg\max} F_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m) \tag{4.2}$$

Where,

$$F_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m) = \sum_{\mathbf{x} \in \mathbf{X}} I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) \tag{4.3}$$

We consider the average absolute difference $f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$, instead of the total absolute difference $F_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$, It is easy to maximize for really big data. $f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$ always lies inside $[0, 1]$, as opposed to $F_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$ which grows with $N(\mathbf{X})$.

$$f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x} \in \mathbf{X}} I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) \tag{4.4}$$

We will still have,

$$m_t(\mathbf{a}, \mathbf{b}, \mathbf{X}) = \underset{m \in (a_t, b_t)}{\arg\max} f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m) \tag{4.5}$$

With this split of $(\mathbf{a}, \mathbf{b})$ at the median $m_t(\mathbf{a}, \mathbf{b}, \mathbf{X})$, we get two disjoint neighborhoods $(\mathbf{a}^{left}, \mathbf{b}^{left})$ and $(\mathbf{a}^{right}, \mathbf{b}^{right})$ whose union is $(\mathbf{a}, \mathbf{b})$. Here $\mathbf{a}^{left} = \mathbf{a}$; $\mathbf{b}_l^{left} = \mathbf{b}_l$ for $1 \leq l \leq p, l \neq t$ and $\mathbf{b}_t^{left} = m_t(\mathbf{a}, \mathbf{b}, \mathbf{X})$. Similarly, $\mathbf{a}_l^{right} = \mathbf{a}_l$ for $1 \leq l \leq p, l \neq t$; $\mathbf{a}_t^{right} = m_t(\mathbf{a}, \mathbf{b}, \mathbf{X})$ and $\mathbf{b}^{right} = \mathbf{b}$.

We start with a single neighborhood $(-\mathbf{1}, \mathbf{1}) = \big((-1, \cdots, -1), (1, \cdots, 1)\big)$ at level 1, and we split along x_1 at the median value $m_1^0(\mathbf{X}) = m_1(-\mathbf{1}, \mathbf{1}, \mathbf{X})$. In general, we cycle through all the co-ordinates, suppose at level d we pick the co-ordinate x_t, we start with 2^d number of neighborhoods and we split each of these neighborhood $(\mathbf{a}, \mathbf{b})$

in two sub-neighborhoods along x_t at the median value $m_t(\mathbf{a}, \mathbf{b}, \mathbf{X})$. At level d we get 2^d median values $m_1^d(\mathbf{X}), \cdots, m_{2^d}^d(\mathbf{X})$. So, we will have 2^d splits at level d, resulting in 2^{d+1} neighborhoods at level $d+1$. If we continue and construct a KD-tree of depth D, we will compute $2^D - 1$ conditional median statistics $m_1^0(\mathbf{X})$; $m_1^1(\mathbf{X})$, $m_2^1(\mathbf{X})$; $\cdots$; $m_1^{D-1}(\mathbf{X})$, $\cdots, m_{2^{D-1}}^{D-1}(\mathbf{X})$ along the construction process. Observe that these $2^D - 1$ statistics gives us all the information's we need to construct a KD-tree. These are not EP statistics, can't be computed in parallel by D&R.

In this paper, we propose an approximation algorithm EP-KD to construct approximate KD-tree. We propose to approximate $m_k^d(\mathbf{X})$ by Jth term of a sequence of WEP statistics $\{\hat{m}_{k,j}^d(\mathbf{X})\}_0^\infty$, bigger J gives more accuracy. Here $\hat{m}_{k,J}^d(\mathbf{X})$ is minimizer of an objective function which approximates $f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$ by taking Jth partial sum of Fourier series expansion of each product term in $I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$. Observe, for any such general neighborhood $(\mathbf{a}, \mathbf{b})$, we have, $-\mathbf{1} \leq \mathbf{a} \leq \mathbf{1}$ and $-\mathbf{1} \leq \mathbf{b} \leq \mathbf{1}$ coordinate-wise. Also we are only considering $a_t < m < b_t$, when we define $f_{\mathbf{X},t,\mathbf{a},\mathbf{b}}(m)$. So that $x_t - m$ can only take value in $(-2, 2)$, and $|x_t - m| < \pi$. Also by similar argument, $|\mathbf{x} - \mathbf{a}| < \pi$ and $|\mathbf{x} - \mathbf{b}| < \pi$ coordinate-wise.

4.2.1 Fourier expansion:

Now consider Fourier expansion of the function $f(z) = |z|$

$$|z| = \frac{\pi}{2} - \frac{4}{\pi} \sum_{j=1}^{\infty} \frac{\cos\big((2j-1)z\big)}{(2j-1)^2} \tag{4.6}$$

If we substitute $z = x_t - m$, we have

$$\begin{aligned} |x_t - m| &= \frac{\pi}{2} - \sum_{j=1}^{\infty} \frac{4}{\pi(2j-1)^2} \cos\big((2j-1)(x_t - m)\big) \\ &= \frac{\pi}{2}.1 + \sum_{j=1}^{\infty} \Big(- \frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2} \Big). \cos\big((2j-1)x_t\big) \\ &+ \sum_{j=1}^{\infty} \Big(- \frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2} \Big). \sin\big((2j-1)x_t\big) \end{aligned} \tag{4.7}$$

For convenience, let us define: $f_0(m) = \frac{\pi}{2}$ and

$f_{2j-1}(m) = -\frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}$

$f_{2j}(m) = -\frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}$ for $j \in \mathbb{Z}^+$.

Also define: $c_0(x_t) = 1$ and

$c_{2j-1}(x_t) = \cos\big((2j-1)x_t\big)$

$c_{2j}(x_t) = \sin\big((2j-1)x_t\big)$ for $j \in \mathbb{Z}^+$.

Then we have:

$$|x_t - m| = \sum_{j=0}^{\infty} f_j(m)c_j(x_t). \tag{4.8}$$

Now partial sums of Fourier expansion of $f(z) = |z|$ converges uniformly to its limit for any $z \in (-\pi, \pi)$, which implies that for the above expression, partial sums of sequence of functions $\sum_{j=0}^{2J} c_j(x_t)f_j(m)$ converges point-wise and uniformly to its limit $|x_t - m|$ for any $m \in (-1, 1)$ and $x_t \in (-1, 1)$.

For the indicator functions, first observe that if $a_l < b_l$, then $I(a_l \le x_l \le b_l) = I(a_l \le x_l) - I(b_l \le x_l)$. The idea is to approximate these terms by partial sum of Fourier-series expansion, but, if $a_l = -1$, then we already know that for any $x_l \in (-1, 1)$, we have $a_l < x_l$, and in that case $I(a_l \le x_l)$ becomes a constant(=1) which is independent of x_l. Then, it does not makes sense to approximate it with a Fourier-series. Similarly if $b_l = 1$, then $I(b_l \le x_l) = 0$. So, we consider a little modification of above expression as follows,

$$\begin{aligned}
&I(a_l \le x_l \le b_l) = I(a_l = -1).I(b_l = 1).1 + I(a_l > -1).I(b_l = 1).I(a_l < x_l) \\
&+ I(a_l = -1).I(b_l < 1).I(x_l < b_l) + I(a_l > -1).I(b_l < 1).I(a_l < x_l < b_l) \\
&= \big(1 - I(a_l > -1)\big)\big(1 - I(b_l < 1)\big) + I(a_l > -1)\big(1 - I(b_l < 1)\big)I(a_l < x_l) \\
&+ \big(1 - I(a_l > -1)\big)I(b_l < 1)\big(1 - I(b_l < x_l)\big) \\
&+ I(a_l > -1)I(b_l < 1)\big(I(a_l < x_l) - I(b_l < x_l)\big) \\
&= 1 - I(a_l > -1) + I(a_l > -1)I(a_l < x_l) - I(b_l < 1)I(b_l < x_l)
\end{aligned} \tag{4.9}$$

Again consider Fourier expansion of the function $g(z) = I(0 \le z)$

$$I(z>0) = \frac{1}{2} + \frac{2}{\pi}\sum_{j=1}^{\infty} \frac{\sin\big((2j-1)z\big)}{(2j-1)} \tag{4.10}$$

We replace z with $x_l - a_l$ and $x_l - b_l$ and substitute in 4.8, $I(a_l < x_l) = I(0 < x_l - a_l)$ and $I(b_l < x_l) = I(0 < x_l - b_l)$, we have

$$\begin{aligned}
&I(a_l \le x_l \le b_l) = 1 - I(a_l > -1) + I(a_l > -1)I(a_l < x_l) - I(b_l < 1)I(b_l < x_l) \\
&= 1 - I(a_l > -1) + I(a_l > -1)I(0 < x_l - a_l) - I(b_l < 1)I(0 < x_l - b_l) \\
&= 1 - I(a_l > -1) + \frac{I(a_l > -1)}{2} + \sum_{j=1}^{\infty} \frac{2I(a_l > -1)}{\pi(2j-1)} \sin\big((2j-1)(x_l - a_l)\big) \\
&- \frac{I(b_l < 1)}{2} - \sum_{j=1}^{\infty} \frac{2I(b_l < 1)}{\pi(2j-1)} \sin\big((2j-1)(x_l - b_l)\big) \\
&= \big(1 - \frac{I(a_l > -1)}{2} - \frac{I(b_l < 1)}{2}\big).1 \\
&+ \sum_{j=1}^{\infty} \frac{2I(b_l < 1)\sin\big((2j-1)b_l\big) - 2I(a_l > -1)\sin\big((2j-1)a_l\big)}{\pi(2j-1)} . \cos\big((2j-1)x_l\big) \\
&+ \sum_{j=1}^{\infty} \frac{2I(a_l > -1)\cos\big((2j-1)a_l\big) - 2I(b_l < 1)\cos\big((2j-1)b_l\big)}{\pi(2j-1)} . \sin\big((2j-1)x_l\big)
\end{aligned} \tag{4.11}$$

And, again for convenience, we define:

$g_0(m) = \big(1 - \frac{I(a_l>-1)}{2} - \frac{I(b_l<1)}{2}\big)$ and

$g_{2j-1}(a_l, b_l) = \frac{2I(b_l<1)\sin\big((2j-1)b_l\big) - 2I(a_l>-1)\sin\big((2j-1)a_l\big)}{\pi(2j-1)}$

$g_{2j}(a_l, b_l) = \frac{2I(a_l>-1)\cos\big((2j-1)a_l\big) - 2I(b_l<1)\cos\big((2j-1)b_l\big)}{\pi(2j-1)}$ for $j \in \mathbb{Z}^+$.

Then we have:

$$I(a_l \le x_l \le b_l) = \sum_{j=0}^{\infty} c_j(x_l) g_j(a_l, b_l). \tag{4.12}$$

However, for a given neighborhood (a_l, b_l), the above series expansion of $I(a_l \le x_l \le b_l)$ does not converges to its value uniformly for all $x_l \in (-1, 1)$. For the indicator function $g(z) = I(z < 0)$, partial sums of Fourier expansion converges point-wise to its limit except at $z = 0$, and also we do not have uniform convergence of the Fourier series for $I(z > 0)$. In the next section, we will see if we stay away from 0, partial

sums of Fourier expansion of $I(0 < z)$ converges uniformly to its value in the interval $(-\pi, \pi) \setminus (-\delta, \delta)$ for any $\delta > 0$.

For a general set S, let us introduce notations $S^1 = S$ and $S^0 = \Phi$(Null set). For a general interval $(a_l, b_l) \subseteq [-1, 1]$ of x_l and any $\delta > 0$, define sets

$$\begin{aligned} P_{(a_l,b_l)} &= (-1,1) \setminus \big(\{a_l\}^{I(a_l>-1)} \cup \{b_l\}^{I(b_l<1)}\big) \\ U^{\delta}_{(a_l,b_l)} &= (-1,1) \setminus \big((a_l-\delta, a_l+\delta)^{I(a_l>-1)} \cup (b_l-\delta, b_l+\delta)^{I(b_l<1)}\big) \end{aligned} \tag{4.13}$$

Define,$P_{(\mathbf{a},\mathbf{b})} = \times_{l=1}^{p} P_{(a_l,b_l)}$ and $U^{\delta}_{(\mathbf{a},\mathbf{b})} = \times_{l=1}^{p} U^{\delta}_{(a_l,b_l)}$ Then from the equations 4.1,4.7 and 4.11, for $\mathbf{x} \in P_{(\mathbf{a},\mathbf{b})}$, we have the series expansion:

$$I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = \sum_{j=0}^{\infty} \big(c_j(x_t) f_j(m)\big) \prod_{l=1}^{p} \big(\sum_{j=0}^{\infty} c_j(x_l) g_j(a_l, b_l)\big) \tag{4.14}$$

Also, in the next section, we will see that, given any $\delta > 0$, the series expansion in 4.14 converges uniformly to its limit for $\mathbf{x} \in U^{\delta}_{(\mathbf{a},\mathbf{b})}$.

4.2.2 Approximation:

We approximate $I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ with $I^J_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$, we consider Jth partial sum for Fourier expansions of $|x_t - m|$ and $I(a_l < x_l < b_l)$ for $1 \le l \le p$,

$$I^J_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = \big(\sum_{j=0}^{2J} c_j(x_t) f_j(m)\big) \prod_{l=1}^{p} \big(\sum_{j=0}^{2J} c_j(x_l) g_j(a_l, b_l)\big) \tag{4.15}$$

Let $\mathbb{N}_J = \{0, 1, \cdots, 2J\}$, and let $\boldsymbol{j} = \{j_1, \cdots, j_p\}$ denote a general index element of $\mathbb{N}_J^{[p]} = \mathbb{N}_J \times \cdots \times \mathbb{N}_J (p \text{ times})$. Define $\mathbf{c}^{[t]}_{k,\boldsymbol{j}}(\mathbf{x}) = c_k(x_t) \prod_{l=1}^{p} c_{j_l}(x_l)$ and $\mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) = \prod_{l=1}^{p} g_{j_l}(a_l, b_l)$.

Then,

$$\begin{aligned} I^J_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) &= \sum_{j=0}^{2J} \sum_{\boldsymbol{j} \in \mathbb{N}_J^{[p]}} c_k(x_t) f_k(m) \prod_{l=1}^{p} c_{j_l}(x_l) g_{j_l}(a_l, b_l) \\ &= \sum_{\{k,\boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} \mathbf{c}^{[t]}_{k,\boldsymbol{j}}(\mathbf{x}) f_k(m) \mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) \end{aligned} \tag{4.16}$$

Define the error in approximation as $e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) - I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$. The approximation of $F_t(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$ will be $F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$ defined as:

$$F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = \sum_{\mathbf{x} \in \mathbf{X}} \sum_{\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} \mathbf{c}_{k,\boldsymbol{j}}^{[t]}(\mathbf{x}) f_k(m) \mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) \tag{4.17}$$

This is a finite sum and we can exchange summation to get:

$$F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \sum_{\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} f_k(m) \mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{c}_{k,\boldsymbol{j}}^{[t]}(\mathbf{x}) \tag{4.18}$$

Let $C_{k,\boldsymbol{j}}^{[t]}(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} c_{k,\boldsymbol{j}}^{[t]}(\mathbf{x})$ and $\bar{C}_{k,\boldsymbol{j}}^{[t]}(\mathbf{X}) = \frac{C_{k,\boldsymbol{j}}^{[t]}(\mathbf{X})}{N(\mathbf{X})}$, then

$$F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \sum_{\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} f_k(m) \mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) C_{k,\boldsymbol{j}}^{[t]}(\mathbf{X}) \tag{4.19}$$

and

$$f_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \frac{1}{N(\mathbf{X})} F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \sum_{\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} f_k(m) \mathbf{g}_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) \bar{C}_{k,\boldsymbol{j}}^{[t]}(\mathbf{X}) \tag{4.20}$$

Define the total error $e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$ and the average error $\bar{e}_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$ as

$$\begin{aligned} e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) &= F_t(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) - F_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) \\ \bar{e}_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) &= f_t(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) - f_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) \end{aligned} \tag{4.21}$$

4.3 Stochastic Properties of the Error term

In these section, we assume that all the $\mathbf{x}$ observations are i.i.d. realizations of a p variate random variable $\tilde{\mathbf{x}}$ for $\mathbf{x} \in \mathbf{X}$. In this section we will show that, if $\tilde{\mathbf{x}}$ has a density, then the expected value of $e_t^J(m, \mathbf{a}, \mathbf{b}, \tilde{\mathbf{x}})$ approaches 0, if we keep increasing J. We assume that t, m, $\mathbf{a}$ and $\mathbf{b}$ are given and constant in this section.

Also for reference let us write down Jth partial sums the Fourier series expansion of $f(z) = |z|$ and $g(z) = I(z > 0)$ as mentioned in equations 4.8 and 4.12, we define

$$
\begin{aligned}
|z|^{[J]} &= \frac{\pi}{2} - \frac{4}{\pi}\sum_{j=1}^{J}\frac{cos\big((2j-1)z\big)}{(2j-1)^2} \\
I^J(0<z) &= \frac{1}{2} + \frac{2}{\pi}\sum_{j=1}^{J}\frac{sin\big((2j-1)z\big)}{(2j-1)}
\end{aligned}
\tag{4.22}
$$

Also, define

$$
\begin{aligned}
|z-m|^{[J]} &= \sum_{j=0}^{2J} f_j(m)c_j(z) \\
I^{[J]}(a<z<b) &= \sum_{j=0}^{2J} g_j(m)c_j(z)
\end{aligned}
\tag{4.23}
$$

We need a few lemmas for the final result

Lemma 4.3.1. *The sequence of functions $|z|^{[J]}$ converge uniformly to the limit $|z|$ in the interval $[-\pi,\pi]$, and hence are uniformly bounded in the interval $[-\pi,\pi]$.*

Corollary 4.3.2. *Given $m \in (-1,1)$, the sequence of functions $|z-m|^{[J]}$ converges uniformly to its $|z-m|$ limit in $(-1,1)$.*

Lemma 4.3.3. *The sequence of functions $I^{[J]}(0<z)$ converge uniformly to the limit $I(0<z)$ in the interval $[-\pi,-\delta)\bigcup(\delta,\pi]$, for any $0<\delta<\pi$.*

Corollary 4.3.4. *Given $-1<a<b<1$, the sequence of functions $I^{[J]}(a<z<b)$ converges uniformly to its limit $I(a<z<b)$ in $[(-1,1)\setminus\{(a-\delta,a+\delta)\bigcup(b-\delta,b+\delta)\}]$, for any $0<\delta<1$.*

Lemma 4.3.5. *The sequence of functions $I^{[J]}(0<z)$ are uniformly bounded in the interval $[-\pi,\pi]$.*

Remember the functions $I_t(m,\mathbf{a},\mathbf{b},\mathbf{x})$ and $I_t^J(m,\mathbf{a},\mathbf{b},\mathbf{x})$ defined in 4.1 and 4.16. We have:

$$
I_t(m,\mathbf{a},\mathbf{b},\mathbf{x}) = |x_t - m|\prod_{l=1}^{p}\big(I(a_l<x_l<b_l)\big) \tag{4.24}
$$

and

$$I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = |x_t - m|^{[J]} \prod_{l=1}^{p} I[J](a_l < x_l < b_l) \tag{4.25}$$

Consider p dimensional variable $\mathbf{z}$. Let $E \subset \mathbb{R}^p$, and let $F_J(\mathbf{z})$ and $G_J(\mathbf{z})$ are two sequences of real-valued functions converging uniformly to their limiting real-valued functions $F(\mathbf{z})$ and $G(\mathbf{z})$, on E. We need the following lemma from Real Analysis:

Lemma 4.3.6. *If $F_J(\mathbf{z})$ and $G_J(\mathbf{z})$ are uniformly bounded in E, then, their product $F_J(\mathbf{z})G_J(\mathbf{z})$ uniformly converges to the function $F(\mathbf{z})G(\mathbf{z})$ on E.*

For $\delta > 0$, let $N_\delta^{\mathbf{a},\mathbf{b}}$ be defined as the interval: $N_\delta^{\mathbf{a},\mathbf{b}} = \big(\mathbf{a}-\delta, \mathbf{a}+\delta) \cup (\mathbf{b}-\delta, \mathbf{b}+\delta)\big) \in E = (-1, 1)^p$.

Theorem 4.3.7. *For any given m, $\mathbf{a}$ and $\mathbf{b}$; $F_J(\mathbf{x}) = I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ uniformly converges to $F(\mathbf{x}) = I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ on $E - N_\delta^{\mathbf{a},\mathbf{b}}$ for $\delta > 0$.*

Theorem 4.3.8. *Let $e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) = I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) - I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$. Let $P : \mathbb{B}(-1, 1)^p \mapsto [0, 1]$, be a Probability-Measure absolutely continuous with respect to the Lebesgue measure λ on $\mathbb{B}(-1, 1)^p$. Then $E_P(e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})) \to 0$ as $J \to \infty$.*

Suppose $\tilde{\mathbf{x}}$ is a bounded random variable. Then we have the following theorem:

Theorem 4.3.9. *$\bar{e}_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X}) = \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x} \in \mathbf{X}} e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) \to 0$, almost surely in E.*

4.4 KD-Tree construction algorithms

Recall our data $\mathbf{X}$ is a distributed Big-Data, divided in R subsets $\mathbf{X}_1, \cdots, \mathbf{X}_R$ and we are trying to construct a KD tree from this data. The data is multivariate, an observation $\mathbf{x}$ has p coordinates: $\mathbf{x} = (x_1, \cdots, x_p)$.

Remember, the equivalent version of canonical KD-tree construction procedure in section 4.2, we start with the interval $((-1, \cdots, -1), (1, \cdots, 1))$ and at dth step we divide 2^{d-1} intervals obtained from previous step along a coordinate x_t at the median of x_ts for each of these intervals. We get the medians by minimizing equation 4.5 and

as seen in section 4.2if we have the medians $m_1^0(\mathbf{X})$; $m_1^1(\mathbf{X})$, $m_2^1(\mathbf{X})$; $\cdots$; $m_1^{D-1}(\mathbf{X})$, $\cdots$, $m_{2^{D-1}}^{D-1}(\mathbf{X})$, then we can construct the whole KD tree, we need to compute $2^D - 1$ medians to construct KD-tree of depth D.

For large, distributed data, it is not easy to optimize the function in equation 4.5. An iterative D&R procedure will be very time consuming, we have to read the same data over and over for each iteration. As an alternative, we can use the EP-FQ method or Binning method as described in chapter 3.3 to get the medians for distributed data. Then, we have to perform a 2 map-reduce step to add each new level, during the construction of the KD-tree. First D&R step to divide the data in neighborhoods (except 1st level, because every data point lie inside $\big((-1, \cdots, -1), (1, \cdots, 1)\big)$ at the beginning) and second D&R step to calculate medians. Then the KD-tree construction is an iterative procedure consisting $2D-1$ D&R steps ($2D$ steps if the data requires scaling).

4.4.1 EP-FKD0 algorithm

Now, we know that, we can accurately approximate $f_t(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$ by $f_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{X})$. We choose J w.r.t. our desired accuracy. Then we can get an approximate conditional median as:

$$\hat{m}_t^J(\mathbf{a}, \mathbf{b}, \mathbf{X}) = \underset{m \in (a_t, b_t)}{\arg\min}\, f_t^J m, \mathbf{a}, \mathbf{b}, \mathbf{X}(m) = \underset{m \in (a_t, b_t)}{\arg\min} \frac{1}{N(\mathbf{X})} \sum_{\mathbf{x} \in \mathbf{X}} I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) \quad (4.26)$$

From equation 4.20, we have

$$\hat{m}_t^J(\mathbf{a}, \mathbf{b}, \mathbf{X}) = \underset{m \in (a_t, b_t)}{\arg\min} \sum_{\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}} f_i(m) G_{\mathbf{j}}(\mathbf{a}, \mathbf{b}) \bar{C}_{k,\boldsymbol{j}}^t(\mathbf{X}) \quad (4.27)$$

$C_{k,\boldsymbol{j}}^t(\mathbf{X})$ is SOT (and hence SEP) statistic for any $\{t, k, \boldsymbol{j}\}$. Hence $\bar{C}_{k,\boldsymbol{j}}^t(\mathbf{X}) = \frac{C_{k,\boldsymbol{j}}^t(\mathbf{X})}{N(\mathbf{X})}$ is WEP statistic for any $\{t, k, \boldsymbol{j}\}$ and they all can be computed by a single D&R step. In this section we describe D&R implementation of this idea which we are going to call the EP-FKD0 algorithm. First, we fix an integer J. Again, we use Map- Reduce

to implement EP-FKD0 via R package datadr. In D&R step of EP-FKD0 we compute statistics $N(\mathbf{X})$ and $C^t_{k,\boldsymbol{j}}(\mathbf{X})$ for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$. The number of SEP statistics we compute in D&R step of EP-FKD0 is $n_{\{EP-FKD0\}} = (2J+1)^{p+1} + 1$.

The division method[D] divides the data in subsets and stores in HDFS. In EP-FQ we compute EP statistics independent of the division method used, and the the D&R output we get in EP-FQ would be same for any division of the data. Entire data $\mathbf{X}$ is divided in R subsets $\mathbf{X}_1, \cdots, \mathbf{X}_R$. The observations are p-dimensional vectors and the data structure in each subset are numeric matrices. For the Map-Reduce model, initial Map inputs are R number of key-value pairs $\{1, \mathbf{X}_1\}, \cdots, \{R, \mathbf{X}_R\}$.

Afterwards, an analytic method[A] is applied to subset data, this part of D&R is Embarrassingly parallel, computation is done within the subsets and without any interaction between the subsets and carried out in Map stage of Map-Reduce model. In a general rth subset $\mathbf{X}_r$, for each observation $\mathbf{x}$ of $\mathbf{X}_r$, we compute $n_{\{EP-FKD0\}}$ summands: $\mathbf{c}^t_{k,\boldsymbol{j}}(\mathbf{x})$ for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$ and take a sum over all $\mathbf{x}$ in $\mathbf{X}_r$ to get subset statistics $C^t_{k,\boldsymbol{j}}(\mathbf{X}_r)$ for $t = 1, \cdots, p$, $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$. When we are computing these summands, we need to make sure the computation is memory-space efficient and non repetitive. (See the Appendix $B.10$ for an efficient simultaneous computation of these summand terms). We also count $N(\mathbf{X}_r)$, the number of observations in $\mathbf{X}_r$. These are values for Map output key-value pair, we assign the tuple $\{t, k, \boldsymbol{j}\}$ as key. So, for the rth subset, the Map output or the intermediate key-value pairs of Map-Reduce model are $\{\{t, k, \boldsymbol{j}\}, C^t_{k,\boldsymbol{j}}(\mathbf{X}_r)\}$ for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$.

Finally, for the recombination method[R], since $\mathbf{X}$ is a disjoint union of $\mathbf{X}_1, \cdots, \mathbf{X}_R$, for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$, we have, $C^t_{k,\boldsymbol{j}}(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} \mathbf{c}^t_{k,\boldsymbol{j}}(\mathbf{x}) = \sum_{\mathbf{x} \in \cup_{r=1}^R \mathbf{X}_r} \mathbf{c}^t_{k,\boldsymbol{j}}(\mathbf{x}) = \sum_{r=1}^R \left(\sum_{\mathbf{x} \in \mathbf{X}_r} \mathbf{c}^t_{k,\boldsymbol{j}}(\mathbf{x}) \right) = \sum_{r=1}^R C^t_{k,\boldsymbol{j}}(\mathbf{X}_r)$. Similarly, $N(\mathbf{X}) = \sum_{r=1}^R N(\mathbf{X}_r)$. So, in the Reduce stage, for each key $\{t, k, \boldsymbol{j}\}$, we add corresponding values from the set of all intermediate key-value pairs. Final Map-Reduce output is the set of key-value pairs $\{\{t, k, \boldsymbol{j}\}, C^t_{k,\boldsymbol{j}}(\mathbf{X}_r)\}$ for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$.

We read the Map-Reduce output key-value pairs in the front-end machine to get our D&R output $C^t_{k,\boldsymbol{j}}(\mathbf{X}_r)\}$ for $t = 1, \cdots, p$ and $\{k, \boldsymbol{j}\} \in \mathbb{N}_J^{[p+1]}$. Then, $\hat{m}^J_{0,1}(\mathbf{X})$; $\hat{m}^J_{1,1}(\mathbf{X})$, $\hat{m}^J_{1,2}(\mathbf{X})$; $\cdots$; $\hat{m}^J_{D-1,1}(\mathbf{X})$, $\cdots$, $\hat{m}^J_{D-1,2^{D-1}}(\mathbf{X})$ can all be computed in post reduce optimization. For each J, The collection of approximate conditional median values corresponds to a KD-tree, and this sequence of KD-tree is our approximation to the actual KD-tree. We will provide simulation studies to demonstrate how accurate these approximate KD-trees are if we keep increasing J.

See Appendix $B.11$ for an algorithmic flow chart of EP-FKD0.

4.4.2 EP-FKD algorithm

In this section we briefly discuss how we can improve the run-time of EP-FKD, while keeping the accuracy unchanged. For real variable $z \in (-1, 1)$, unknown parameter $m \in (-1, 1)$ and interval $[a, b] \subseteq [-1, 1]$, let us define $I^J(z, m, a, b) = |z - m|^{[J]} I^{[J]}(a < z < b)$. Then we have the following lemma:

Lemma 4.4.1.

$$I^J(z, m, a, b) = \sum_{j=0}^{J} \omega_j(m, a, b) c_j(z) + \sum_{j=0}^{4J-2} \xi_j^J(m, a, b) \zeta_j(z)$$

$$I_t^J(\mathbf{x}, m, \mathbf{a}, \mathbf{b}) = \sum_{\boldsymbol{j} \in \mathbb{N}_J^{[p]}} \omega_{j_t}(m, a_t, b_t) \mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a}, \mathbf{b}) \mathbf{c}_{\boldsymbol{j}}(\mathbf{x}) + \sum_{\boldsymbol{j} \in \mathbb{N}_J^{[p,t]}} \xi_{j_t}^J(m, a_t, b_t) \mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a}, \mathbf{b}) \boldsymbol{\zeta}_{\boldsymbol{j}}^{[J,t]}(\mathbf{x})$$

Where,

$$\zeta_0(z) = 1; \zeta_{2j-1}(z) = \cos(2jz); \zeta_{2j}(z) = \sin(2jz); \omega_0(m, a, b) = \frac{\pi}{4};$$

$$\omega_{2j-1}(m, a, b)$$
$$= \frac{\sin\big((2j-1)b\big) I(b < 1) - \sin\big((2j-1)a\big) I(a > -1)}{(2j-1)} - \frac{2\cos\big((2j-1)m\big)}{\pi(2j-1)^2};$$

$$\omega_{2j}(m, a, b)$$
$$= \frac{\cos\big((2j-1)a\big) I(a > -1) - \cos\big((2j-1)b\big) I(b < 1)}{(2j-1)} - \frac{2\sin\big((2j-1)m\big)}{\pi(2j-1)^2};$$

$$\xi_0^J(m,a,b) = \sum_{k=1}^{J} C_{k,k}(m,a,b)$$

$$\xi_{2j-1}^J(m,a,b);$$

$$= \sum_{k=\max\{1,j-J+1\}}^{j} A_{k,j-k+1}(m,a,b) + I(j<J)\sum_{k=1}^{J-j}\big(C_{k+j,k}(m,a,b) + C_{k,k+j}(m,a,b)\big)$$

$$\xi_{2j}^J(m,a,b);$$

$$= \sum_{k=\max\{1,j-J+1\}}^{j} B_{k,j-k+1}(m,a,b) + I(j<J)\sum_{k=1}^{J-j}\big(D_{k+j,k}(m,a,b) - D_{k,k+j}(m,a,b)\big)$$

here,

$$A_{j,k}(m,a,b)$$

$$=\frac{8\Big(\sin\big((2j-1)m+(2k-1)a\big)I(a>-1) - \sin\big((2j-1)m+(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)};$$

$$B_{j,k}(m,a,b)$$

$$=-\frac{8\Big(\cos\big((2j-1)m+(2k-1)a\big)I(a>-1) - \cos\big((2j-1)m+(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)};$$

$$C_{j,k}(m,a,b)$$

$$=-\frac{8\Big(\sin\big((2j-1)m-(2k-1)a\big)I(a>-1) - \sin\big((2j-1)m-(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)};$$

$$D_{j,k}(m,a,b)$$

$$=\frac{8\Big(\cos\big((2j-1)m-(2k-1)a\big)I(a>-1) - \cos\big((2j-1)m-(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)}$$

From the second identity in lemma 4.4.1 and equation 4.20, we have:

$$\hat{m}_t^J(\mathbf{a},\mathbf{b},\mathbf{X})$$

$$=\underset{m\in(a_t,b_t)}{\arg\min}\sum_{\mathbf{x}\in\mathbf{X}}\left(\sum_{\boldsymbol{j}\in\mathbb{N}_J^{[p]}}\omega_{j_t}(m,a_t,b_t)\mathbf{g}_{\boldsymbol{j}_{\text{-}t}}(\mathbf{a},\mathbf{b})\mathbf{c}_{\boldsymbol{j}}(\mathbf{x}) + \sum_{\boldsymbol{j}\in\mathbb{N}_J^{[p,t]}}\xi_{j_t}^J(m,a_t,b_t)\mathbf{g}_{\boldsymbol{j}_{\text{-}t}}(\mathbf{a},\mathbf{b})\boldsymbol{\zeta}_{\boldsymbol{j}}^{[J,t]}(\mathbf{x})\right) \tag{4.28}$$

Interchanging summations, we get

$$\begin{aligned}&\hat{m}_t^J(\mathbf{a},\mathbf{b},\mathbf{X})\\&=\arg\min_{m\in(a_t,b_t)}\left(\sum_{\boldsymbol{j}\in\mathbb{N}_J^{[p]}}\omega_{j_t}(m,a_t,b_t)\mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a},\mathbf{b})\mathbf{c}_{\boldsymbol{j}}(\mathbf{x})+\sum_{\boldsymbol{j}\in\mathbb{N}_J^{[p,t]}}\xi_{j_t}^J(m,a_t,b_t)\mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a},\mathbf{b})\boldsymbol{\zeta}_{\boldsymbol{j}}^{[J,t]}(\mathbf{x})\right)\end{aligned}\tag{4.29}$$

We optimize the right side of equation 4.29 w.r.t. m, as opposed to the right side of equation 4.29. In EP-FKD, we compute p dimensional statistics matrices $\{\{\mathbf{c}_{\boldsymbol{j}}(\mathbf{X})\}\}$ for $\boldsymbol{j}\in\mathbb{N}_J^{[p]}$ and $\{\{\boldsymbol{\zeta}_{\boldsymbol{j}}^{[J,t]}(\mathbf{X})\}\}$ for $\boldsymbol{j}\in\mathbb{N}_J^{[p,t]}, t=1,\cdots,p$. All these statistics are SOT(hence SEP) statistics and can be computed in one single D&R step.

Realize that the number of SEP statistics we compute in this process is $n_{EP-FKD}=(2J+1)^p+p*(4J-2)*(2J+1)^{p-1}=(2J+1)^{p-1}\big(2J+1+p*(4J-2)\big)$ We have $n_{EP-FKD}<n_{EP-FKD0}$. The details of the D&R procedure of EP-FKD is exactly similar to the D&R procedure of EP-FKD0, and we are skipping it. See Appendix $B.13$ for algorithmic flow-chart of EP-FKD and see Appendix $B.12$ for efficient computation of summand terms in EP-FKD.

4.5 Performance Study

4.5.1 Data generation

In this section, we are going to demonstrate accuracy and run-time of EP-FKD algorithm for constructing KD-tree of simulated data. For data, we simulate observations from multivariate normal distribution with zero means and equi- correlated correlation matrix. So all $\mathbf{x}\in\mathbf{X}$ are i.i.d. $\sim N_p(\mu,\boldsymbol{\Sigma}^{[r]})$. Here, $\mu_{p\times1}=\mathbf{0}_p'$ and $\boldsymbol{\Sigma}_{p\times p}^{[r]}=(1-r)I_p+rJ_p$, here r is the common correlation between each pair of variables. We take $p=2,3$ and $r=0,0.25,0.5,0.75$ for our simulation. We simulate as distributed sequence-files and store the data in HDFS.

For each combination of p and r, we simulate $N=199\times159\times199\times159(\approx1$ billion) observations. We divide the data into 199 blocks in HDFS, each block contains 159

subsets, and each subset has 159×199 random observations. This simulation process creates blocks of size 85.4 MB for $p = 2$ and 125.7 MB for $p = 3$, so we are within the range of it Cloudera recommended block size for optimum Hadoop job. The cluster we use has 200 nodes and number of blocks is chosen to be $200 - 1 = 199$, so that each container gets one process to run at a time.

In the R package drEP we have function drKDtree() which implements EP-FKD to construct KD-tree for distributed data, which we use for demonstration.

4.5.2 Accuracy comparison

For a distributed big data, it is difficult to judge accuracy in terms of cell boundaries. The exact KD-tree construction is impractical and simply not feasible. So, we judge accuracy in terms of cell-counts. Even if the conditional median calculations are not very accurate, the approximate KD-tree serves the purpose of dividing data into equal subsets, if the cell counts are close to each other. We construct KD-tree for each combination of p and r, and make a QQ-plot to see distributions of cell counts at each level from 6 to 13 (KD-tree depth from 5 to 12.)

Equi-correlated Bi-variate Standard Normal

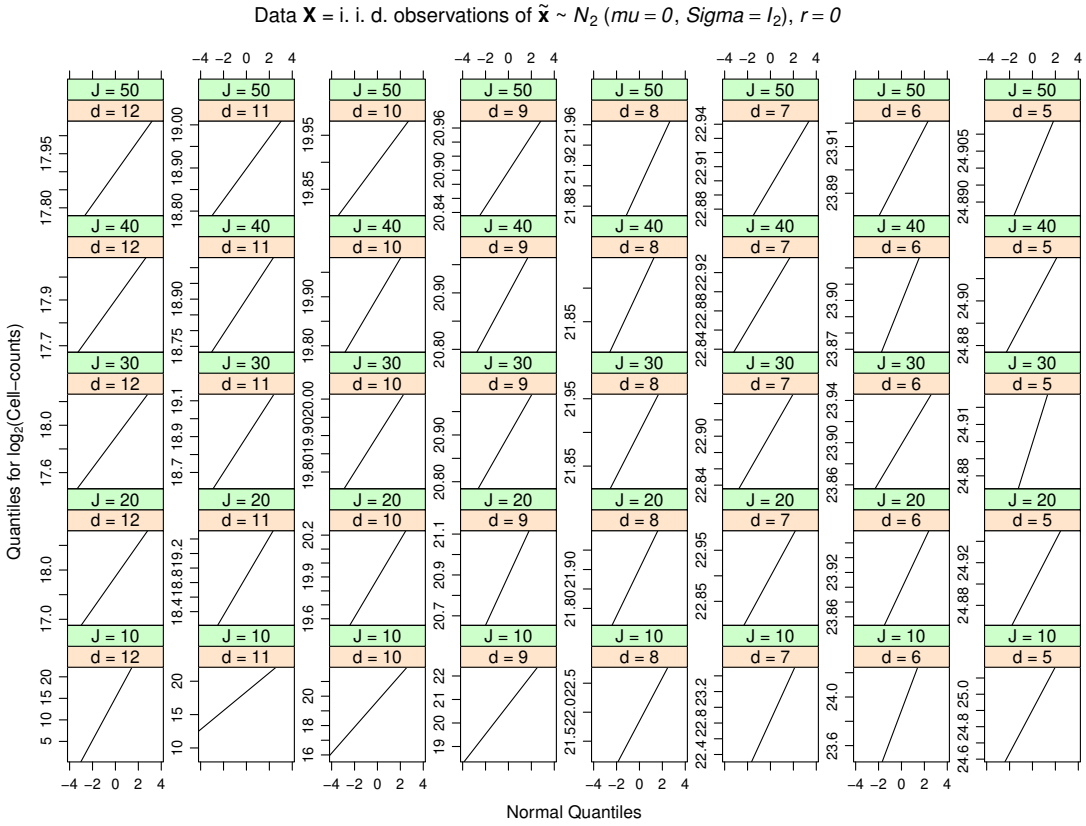

Figure 4.2. Accuracy in KD-tree construction by EP-FKD: $\rho = 0$, $p = 2$

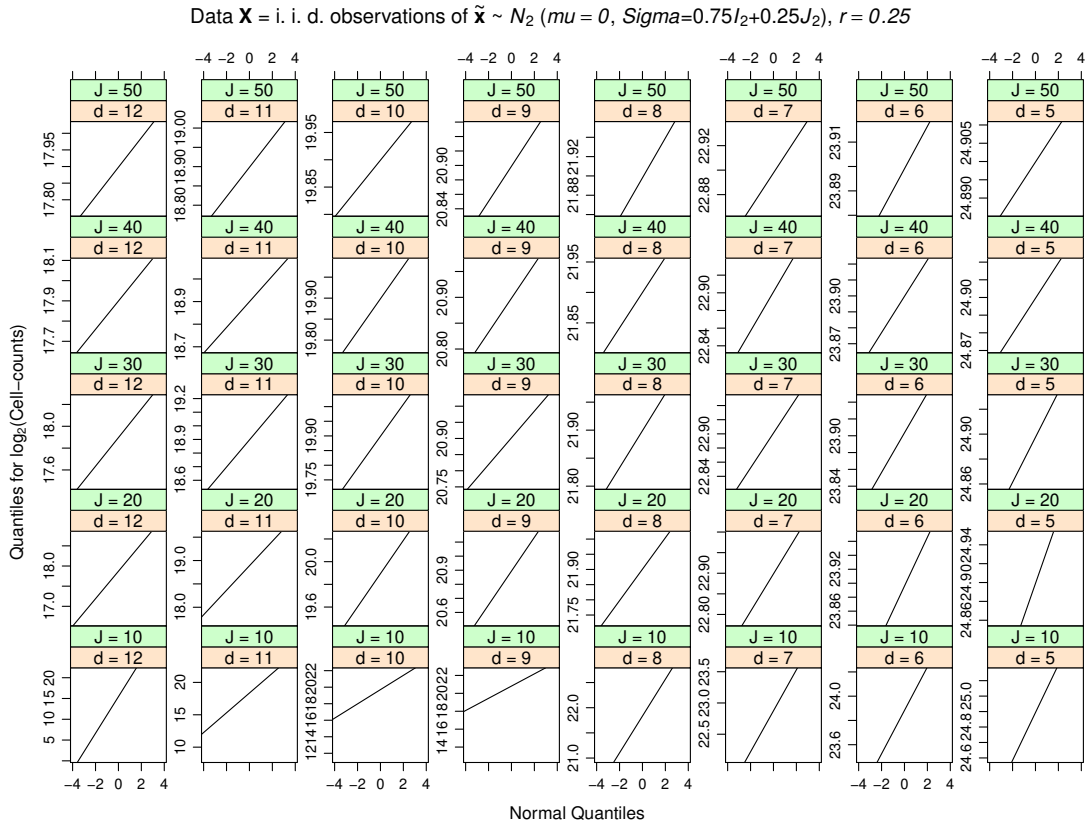

Figure 4.3. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.25$, $p = 2$

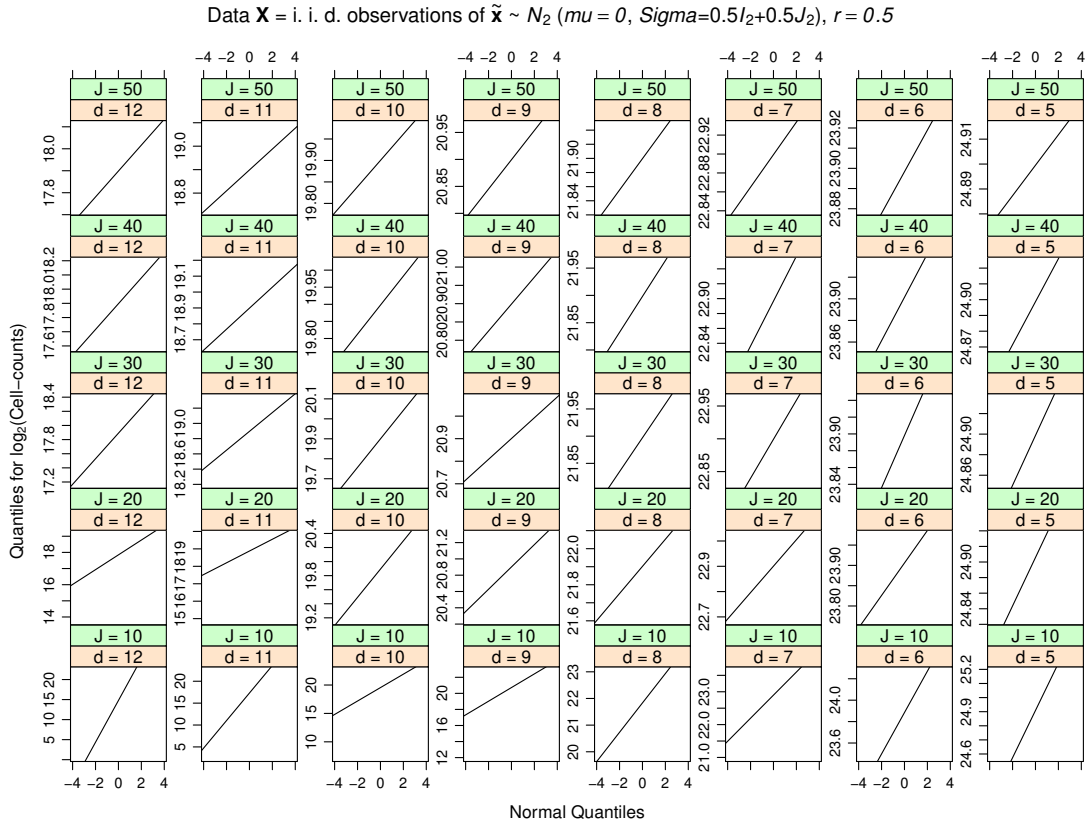

Figure 4.4. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.5$, $p = 2$

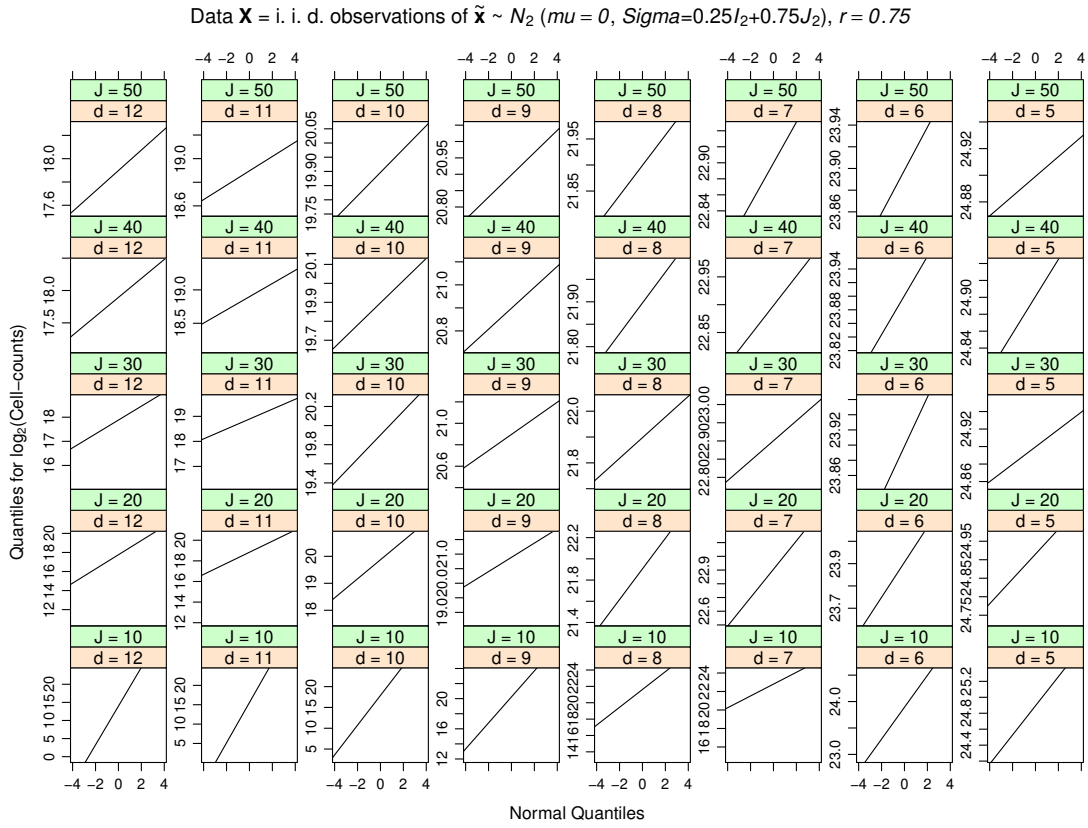

Figure 4.5. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.75$, $p = 2$

Equi-correlated Tri-variate Normal

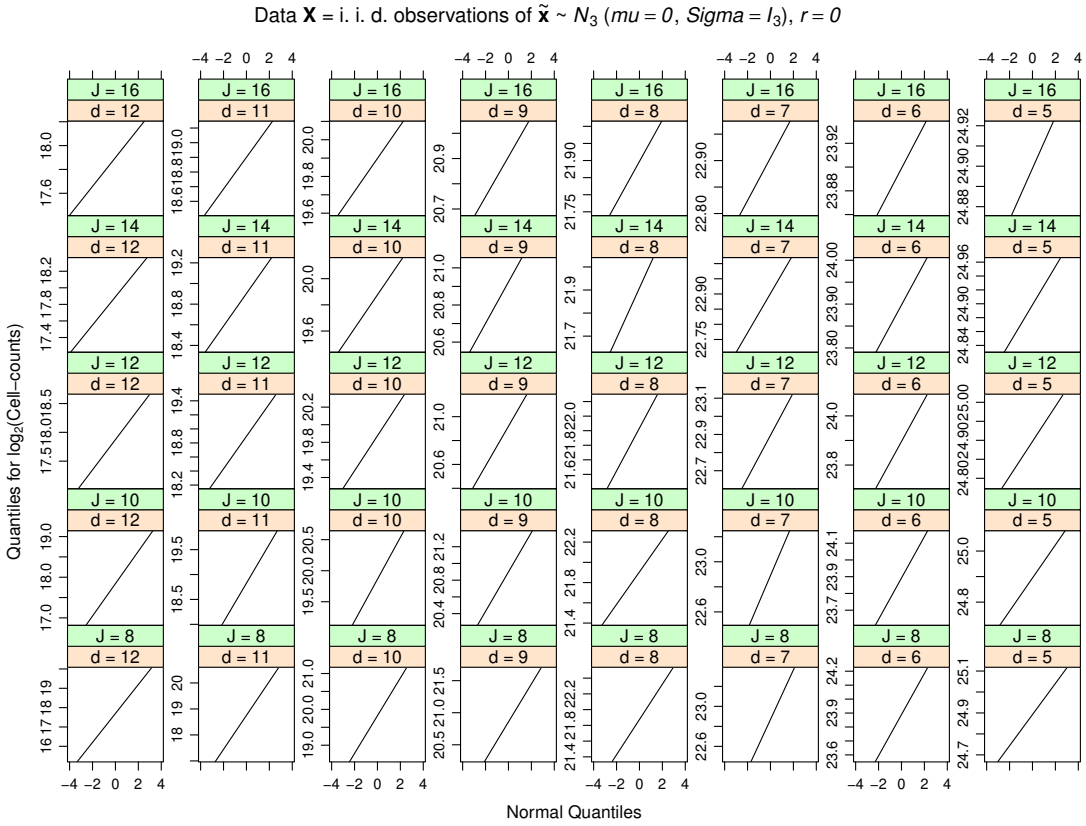

Figure 4.6. Accuracy in KD-tree construction by EP-FKD: $\rho = 0$, $p = 3$

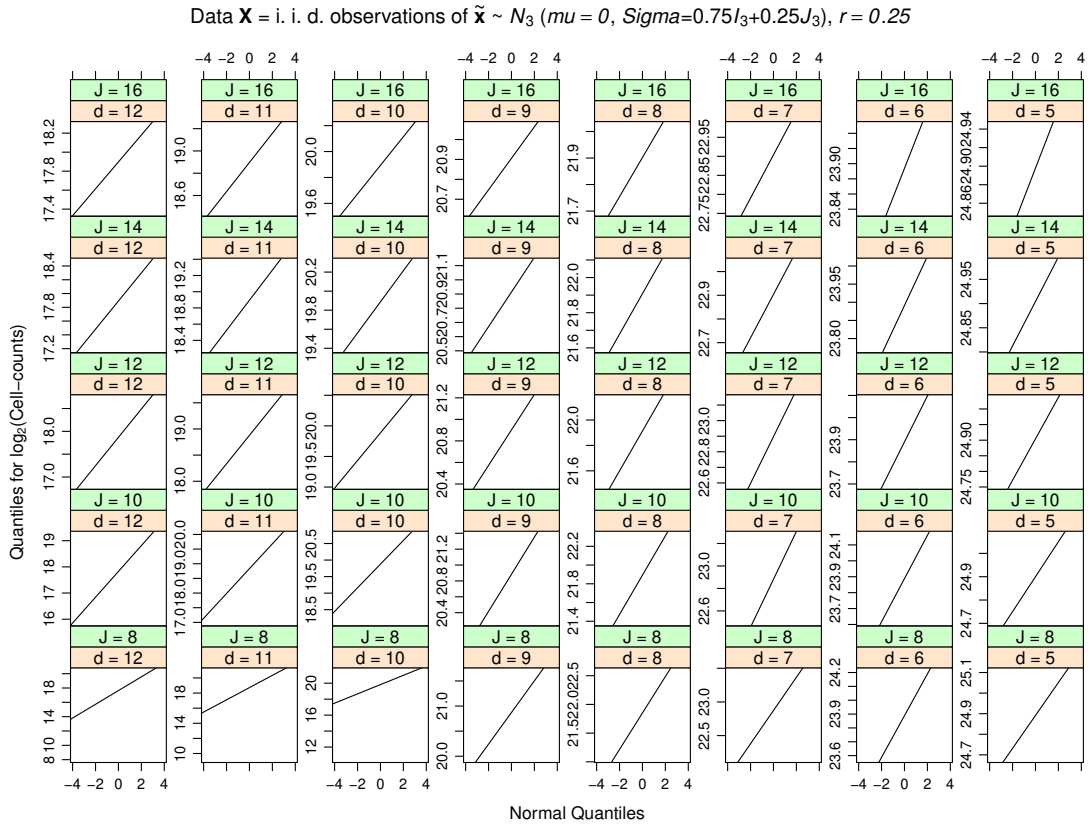

Figure 4.7. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.25$, $p = 3$

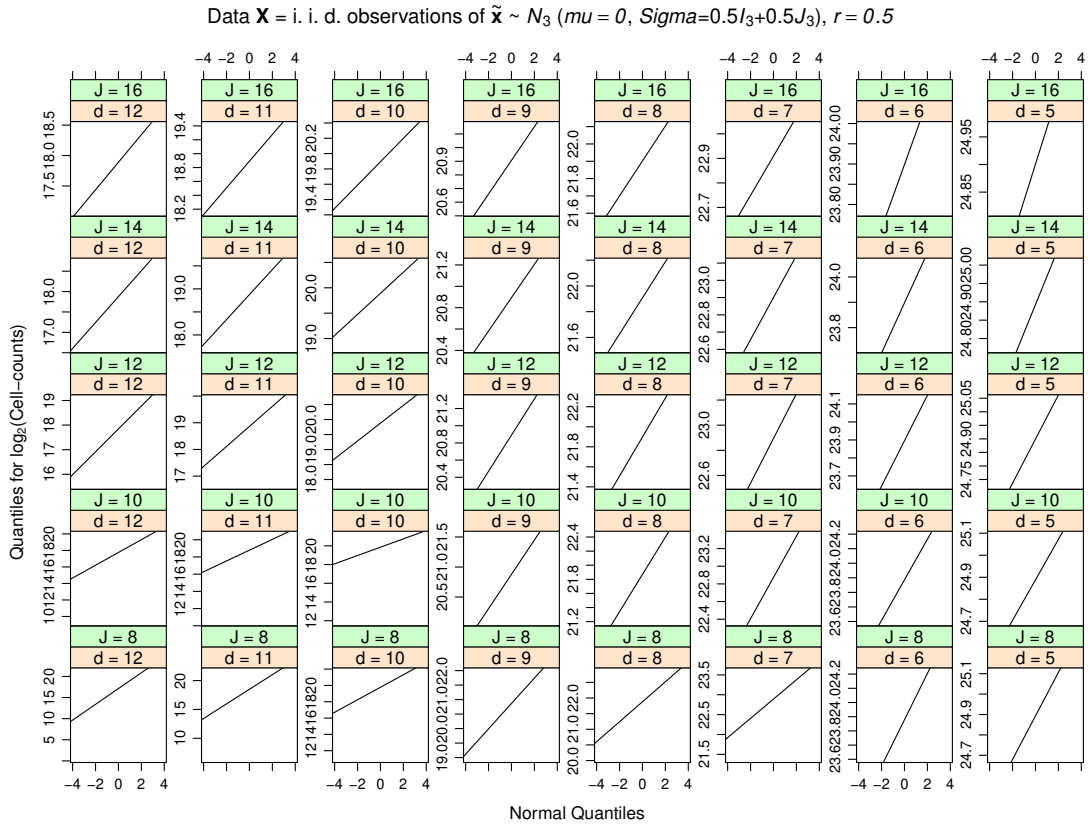

Figure 4.8. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.5$, $p = 3$

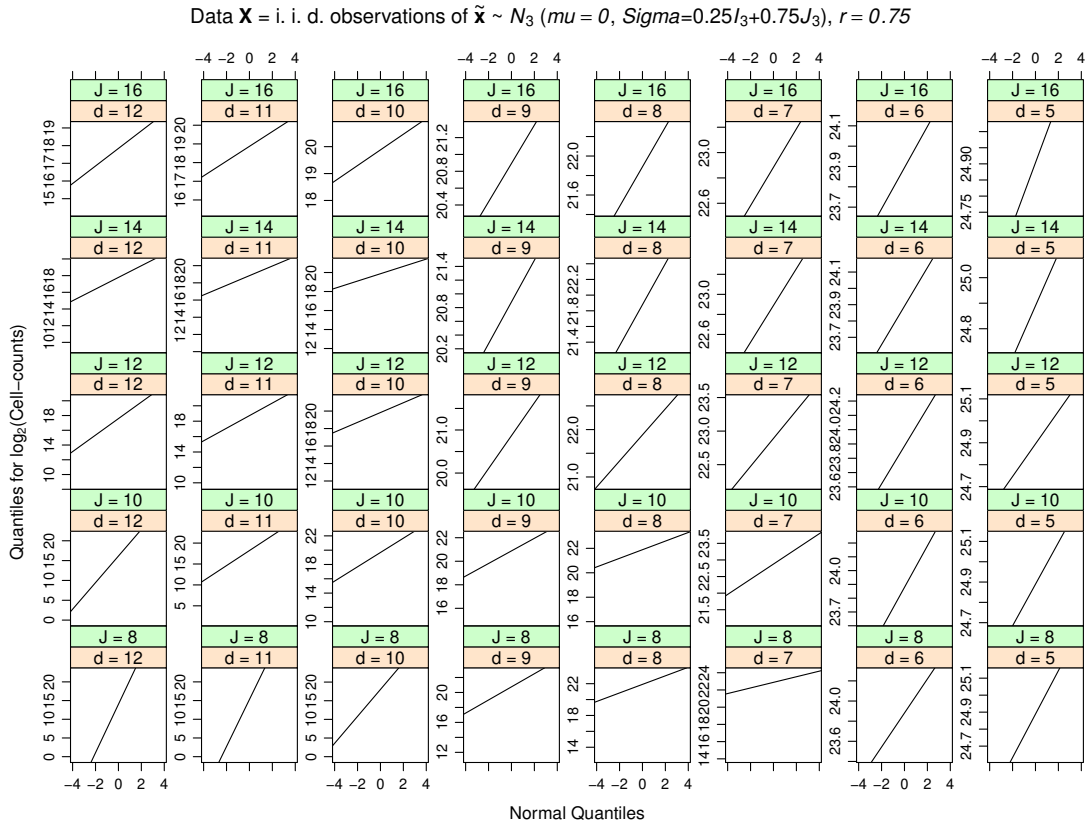

Figure 4.9. Accuracy in KD-tree construction by EP-FKD: $\rho = 0.75$, $p = 3$

4.5.3 Runtime comparison

Here are two plots demonstrating run-time for KD-tree construction for EP-FKD method for dimensions $p = 2$ and $p = 3$, for equi-correlated multivariate Normal data.

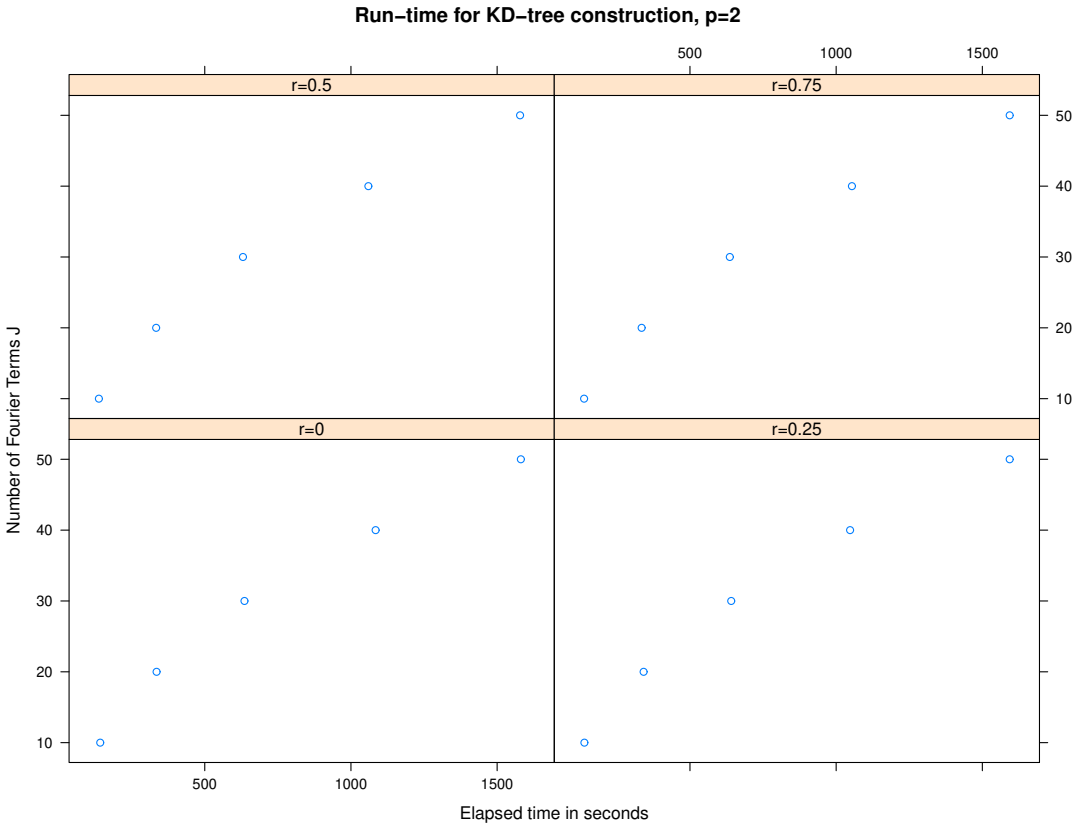

Figure 4.10. Run-time for KD-tree construction by EP-FKD: $p = 2$

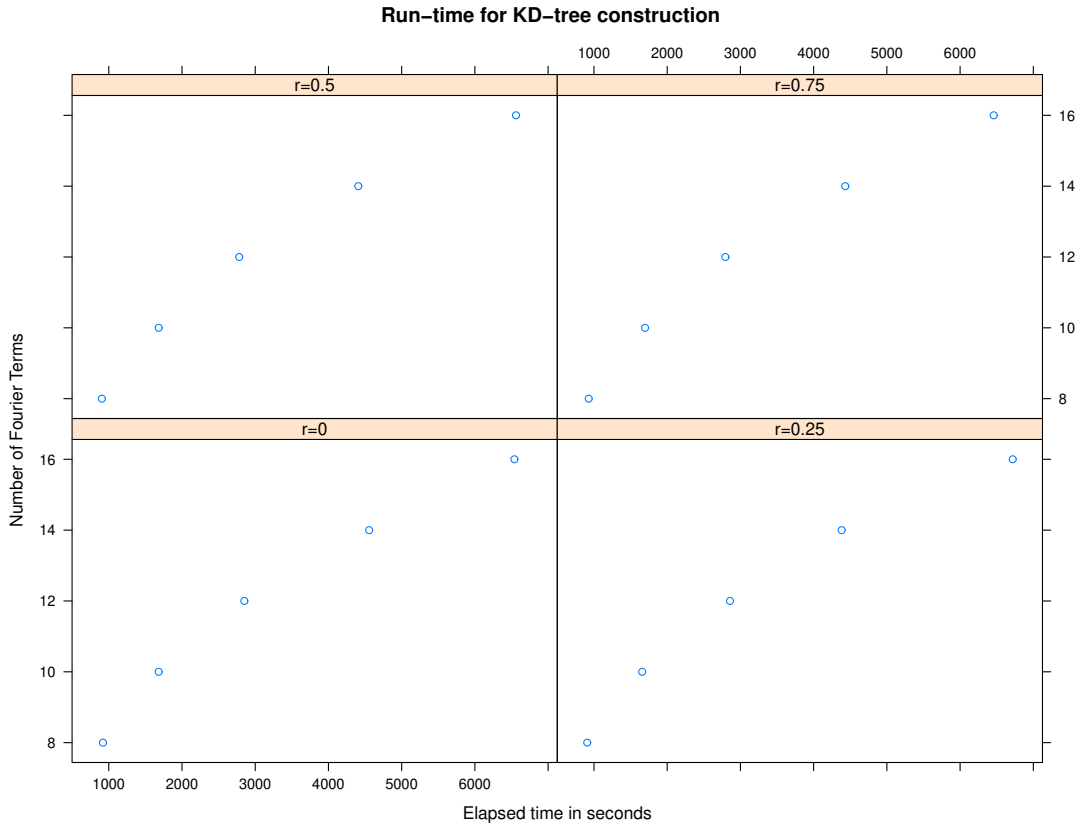

Figure 4.11. Run-time for KD-tree construction by EP-FKD: $p = 3$

5. RECOMMENDATIONS AND SCOPE:

We now present a list of some recommended areas to explore in future and potential applications of Embarrassingly Parallel Statistics:

1.Application of Quantiles and KD-tree construction have a lot of immediate applications viz local regression, near-exact replicate division of a distributed big-data, K-nearest neighbor search for big data. Any rank based statistics for big-data can be approximated by D&R

2. MLE estimation for complex likelihoods: If a complex log-likelihood function $l(\mathbf{x}, \beta)$ is L^2 integrable then, from 2.3.8, we know it can be approximated by FAS functions. So D&R MLE estimation can be carried out.

3. SVM: The hinge loss function is L^2 integrable with some constraints, can be approximated by FAS and can led to D&R SVM algorithm.

4. Quantile Regression: Again one may try to approximate the loss function by FAS and D&R implementation. etc.

REFERENCES

REFERENCES

[1] Saptarshi Guha, Ryan Hafen, Jeremiah Rounds, Jin Xia, Jianfu Li, Bowei Xi, and William S. Cleveland. Large complex data: divide and recombine (d&r) with rhipe. *Stat*, 1(1):53–67, 2012.

[2] Xingxing Lv and Shimeng Shen. On chebyshev polynomials and their applications. *Advances in Difference Equations*, 2017, 12 2017.

[3] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 30, 11 2001.

[4] Zengfeng Huang, Lu Wang, Ke Yi, and Yunhao Liu. Sampling based algorithms for quantile computation in sensor networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 745–756, New York, NY, USA, 2011. ACM.

[5] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 23–34, New York, NY, USA, 2012. ACM.

[6] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: New aggregation techniques for sensor networks. *CoRR*, cs.DC/0408039, 2004.

[7] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 426–435, New York, NY, USA, 1998. ACM.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[9] Jaromr Simsa. The best l2-approximation by finite sums of functions with separable variables. *Aequationes mathematicae*, 43(2-3):248–263, 1992.

APPENDICES

A. PROOFS AND VERIFICATIONS:

A.1. Proof of Theorem 2.2.1

Proof. Let $T_m(.)$ be the Minimal Sufficient Statistic for parameter β of the parametric model $\tilde{\mathbf{x}} \sim p(\mathbf{x};\beta)$. Then $T_m(\mathbf{X}_r)$ Minimal Sufficient Statistic of rth subset $\mathbf{X}_r$ for β and $T_m(\mathbf{X})$ Minimal Sufficient Statistic of the entire data $\mathbf{X}$ for β . Consider the vector valued statistic $T_1(\mathbf{X}) = \{T_m(\mathbf{X}_1), \cdots, T_m(\mathbf{X}_R)\}$. As all $\mathbf{x} \in \mathbf{X}$ are i.i.d., by factorization theorem $T_1(\mathbf{X})$ is a Sufficient Statistic of $\mathbf{X}$ for β. Hence, from the definition of Minimal Sufficiency, $T_m(\mathbf{X})$ should be a function of $T_1(\mathbf{X})$. So there exists a function $f(.)$ such that $T_m(\mathbf{X}) = f(T_m(\mathbf{X}_1), \cdots, T_m(\mathbf{X}_R))$, i.e. $T_m(.)$ is SEPS. □

A.2. Proof of Theorem 2.2.2

Proof. Suppose $T(.)$ is SOT and is induced by $\tau(.)$. Since $\mathbf{X}$ is disjoint union of $\mathbf{X}_1, \cdots, \mathbf{X}_r$, we have, $T(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \tau(\mathbf{x}) = \sum_{\mathbf{x}\in\cup_{r=1}^{R}\mathbf{X}_r} \tau(\mathbf{x}) = \sum_{r=1}^{R} \sum_{\mathbf{x}\in\mathbf{X_r}} \tau(\mathbf{x}) = \sum_{r=1}^{R} T(\mathbf{X_r}) = f(T(\mathbf{X_1}), \cdots, T(\mathbf{X_r}))$, here $f(.)$ is the function that sums its arguments and $T(.)$ is SEPS. □

A.3. Proof of Theorem 2.3.1

Proof. Suppose $F(.,.)$ is STF and is induced by $H(.,.)$. Again, as $\mathbf{X}$ is a disjoint union of $\mathbf{X}_1, \cdots, \mathbf{X}_r$, we have, $F(\mathbf{X},\beta) = \sum_{\mathbf{x}\in\mathbf{X}} H(\mathbf{x},\beta) = \sum_{r=1}^{R} \sum_{\mathbf{x}\in\mathbf{X_r}} H(\mathbf{x},\beta) = \sum_{r=1}^{R} F(\mathbf{X_r},\beta) = f_\beta(F(\mathbf{X_1},\beta), F(\mathbf{X_2},\beta), \cdots, F(\mathbf{X_r},\beta))$, for any $\beta \in \mathbb{Y}$, $f_\beta(.)$ is the function that sums its arguments and hence $F(.,.)$ is EPF. □

A.4. Proof of Theorem 2.3.7

Proof. Let $F(\mathbf{X},\beta)$ is induced by $H(\mathbf{x},\beta)$ and $H(\mathbf{x},\beta) = \sum_{l=1}^{L} \eta_l f_l(\mathbf{x}) g_l(\beta)$. Then, we have $F(\mathbf{X},\beta) = \sum_{\mathbf{x}\in\mathbf{X}} H(\mathbf{x},\beta) = \sum_{\mathbf{x}\in\mathbf{X}} \sum_{l=0}^{L} \eta_l f_l(\mathbf{x}) g_l(\beta) = \sum_{l=0}^{L} \eta_l F_l(\mathbf{X}) g_l(\beta)$, here $F_l(.)$ is SOT induced by $f_l(.)$ for $l = 1, \cdots, L$. Given a β, $F(\mathbf{X},\beta)$ is dependent on $\mathbf{X}$ through the values of the statistics $F_1(\mathbf{X}), \cdots, F_L(\mathbf{X})$. So the set S_F is characterised by the set of SEP statistics $F_1(\mathbf{X}), \cdots, F_L(\mathbf{X})$ and any arbitrary funtional operator on S_F would be a function of them, hence WEP. □

Now, we state a result from 2.3.8 is an application of this result. Let $(\mathbb{X}, \mu)$ and $\mathbb{Y}, \lambda$ are assumed to be two fixed measure spaces with σ-additive and σ-finite measures μ and λ respectively. $L^2(\mathbb{X})$ denotes the space of all real- or complex-valued measurable functions $f : \mathbb{X} \mapsto \mathbb{K}$ ($\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{C}$) satisfying $\int_{\mathbb{X}} |f|^2 d\mu < \infty$, which is a Hilbert space with the inner product $\langle f_1, f_2 \rangle = \int_{\mathbb{X}} f_1 \bar{f}_2 d\mu$. Since we will consider the spaces $L^2(\mathbb{X})$, $L^2(\mathbb{Y})$ and $L^2(\mathbb{X} \times \mathbb{Y})$ simultaneously, we use a subscript to denote norm in a space H as $\| \ \|_H$.

Note that $fg \in L^2(\mathbb{X} \times \mathbb{Y})$ whenever $f \in L^2(\mathbb{X})$ and $g \in L^2(\mathbb{Y})$. This is the reason the classes of functions $\zeta_0 \subseteq \zeta_1 \subseteq \zeta_2 \subseteq \cdots$ defined by:

$$\zeta_J = \Big\{ \sum_{j=1}^{J} f_j g_j \ \Big| \ f_j \in L^2(\mathbb{X}) \text{ and } g_j \in L^2(\mathbb{Y}),\ 0 \leq j \leq J \Big\}$$

form a family of subsets in $L^2(\mathbb{X} \times \mathbb{Y})$. For convenience, we also consider the one element set ζ_0 containing the zero function of $L^2(\mathbb{X} \times \mathbb{Y})$. The following theorem shows the existence of a L^2 approximation:

A.5. Theorem: Suppose that $h \in L^2(\mathbb{X} \times \mathbb{Y})$ is a non-zero function (i.e. $h \notin \zeta_0$) and put

$$\omega = \begin{cases} J & if\ h\ \in\ \zeta_J/\zeta_{J-1} \\ \infty & if\ h\ \notin\ \zeta_J\ for\ any\ J \in \mathbb{N} \end{cases}$$

Then there exist two orthonormal systems $\{u_j\}_{j=1}^{\omega} \subset L^2(\mathbb{X}), \{v_j\}_{j=1}^{\omega} \subset L^2(\mathbb{Y})$ and a non-increasing sequence $\{\eta_j\}_{j=1}^{\omega}$ of positive reals such that

$$H(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{\omega} \eta_j \bar{u}_j(\mathbf{x}) v_j(\mathbf{y}) \quad for\ almost\ all\ (\mathbf{x}, \mathbf{y}) \in (\mathbb{X} \times \mathbb{Y}).$$

The subtotals of the extension form above are the best L_2-approximation of the function H in the sense that

$$\|H - \sum_{j=1}^{J} \eta_j \bar{u}_j v_j\| = \rho_J(H) = \sqrt{\|H\|^2 - \sum_{j=0}^{J} \eta_j^2} \quad for\ any\ n < \omega$$

Finally, the numbers η_j satisfy

$$\sum_{j=1}^{\omega} \eta_j^2 = \|H\|^2$$

Theorem 2.3.8 follows from [9], if we take $\mathbb{X}$ to be the data space with $\mathbf{x}$ as a general observation and $\mathbb{Y} = \Theta$, the parameter space with $\mathbf{y}$ = the parameter β. Here μ and λ are regular Lebesgue measures in $\mathbb{X}$ and Θ respectively.

A.6. lemma: The set of functions $U_j(t) = \frac{e^{i(2j-1)t}}{\sqrt{\pi}}$ for $j \in \mathbb{N}$ is an orthonormal system in $(-\frac{\pi}{2}, \frac{\pi}{2})$.

Proof. Let $j, k \in \mathbb{N}$, then $\langle U_j, U_k \rangle = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{\bar{U}_j(t)}{\sqrt{\pi}} . \frac{U_k(t)}{\sqrt{\pi}} dt = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{-2i(j-k)t} dt$. If $j = k$, then $\langle U_j, U_j \rangle = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^0 dt = 1$ for $j \in \mathbb{N}$. If $j \neq k$, then $\langle U_j, U_k \rangle = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos\big(2(j-k)t\big) dt - \frac{i}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin\big(2(j-k)t\big) dt = \frac{1}{2\pi} \int_{-\pi}^{\pi} \cos\big((j-k)t\big) dt$ (as $\sin\big(2(j-k)t\big)$ is odd function) $= \left[\frac{\sin\big((j-k)t\big)}{(j-k)} \right]_{-\pi}^{\pi} = 0$. So, for $j \in \mathbb{N}$, $U_j(t) = \frac{e^{i(2j-1)t}}{\sqrt{\pi}}$ is an orthonormal system in $(-\frac{\pi}{2}, \frac{\pi}{2})$. □

A.7. lemma: The following identities hold for $j \in \mathbb{N}$:

$$\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \cos\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta = 0$$
$$\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \sin\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta = 0$$

Proof. For the first identity, we have,

$$\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \cos\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta$$
$$= \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \Big(\cos\big((2j-1)\mathbf{x}\big) \cos\big((2j-1)\beta\big) + \sin\big((2j-1)\mathbf{x}\big) \sin\big((2j-1)\beta\big) \Big) d\mathbf{x}\, d\beta$$
$$= \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \cos\big((2j-1)\mathbf{x}\big) \cos\big((2j-1)\beta\big) d\mathbf{x}\, d\beta$$
$$+ \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x} - \beta| \sin\big((2j-1)\mathbf{x}\big) \sin\big((2j-1)\beta\big) d\mathbf{x}\, d\beta$$

Now,

$$
\begin{aligned}
&\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x}-\beta|\cos\big((2j-1)\mathbf{x}\big)\cos\big((2j-1)\beta\big)d\mathbf{x}\;d\beta\\
=&\int_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta|\cos\big((2j-1)\mathbf{x}\big)\cos\big((2j-1)\beta\big)d\mathbf{x}\;d\beta\\
+&\int_{\frac{-\pi}{2}\le\mathbf{x}<\beta\le\frac{\pi}{2}} |\mathbf{x}-\beta|\cos\big((2j-1)\mathbf{x}\big)\cos\big((2j-1)\beta\big)d\mathbf{x}\;d\beta\\
=&2\int_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta|\cos\big((2j-1)\mathbf{x}\big)\cos\big((2j-1)\beta\big)d\mathbf{x}\;d\beta \text{ (by symmetry)}\\
=&2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos\big((2j-1)\beta\big)\bigg(\int_{\beta}^{\frac{\pi}{2}}\mathbf{x}\cos\big((2j-1)\mathbf{x}\big)d\mathbf{x}\bigg)d\beta\\
&-2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\cos\big((2j-1)\beta\big)\bigg(\int_{\beta}^{\frac{\pi}{2}}\cos\big((2j-1)\mathbf{x}\big)d\mathbf{x}\bigg)d\beta\\
=&2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos\big((2j-1)\beta\big)\bigg(-\frac{(-1)^j\frac{\pi}{2}}{(2j-1)}-\frac{\beta\sin\big((2j-1)\beta\big)}{(2j-1)}-\frac{\cos\big((2j-1)\beta\big)}{(2j-1)^2}\bigg)d\beta\\
&-2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\cos\big((2j-1)\beta\big)\bigg(-\frac{(-1)^j}{(2j-1)}-\frac{\sin\big((2j-1)\beta\big)}{(2j-1)}\bigg)d\beta\\
=&-\frac{\pi(-1)^j}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos\big((2j-1)\beta\big)d\beta-\frac{2}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\cos\big((2j-1)\beta\big)d\beta\\
&-\frac{2}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos^2\big((2j-1)\beta\big)d\beta+\frac{2(-1)^j}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\cos\big((2j-1)\beta\big)d\beta\\
&+\frac{2}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\cos\big((2j-1)\beta\big)d\beta\\
=&-\frac{\pi(-1)^j}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos\big((2j-1)\beta\big)d\beta-\frac{2}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\cos^2\big((2j-1)\beta\big)d\beta\\
=&\frac{2\pi}{(2j-1)^2}-\frac{\pi}{(2j-1)^2}=\frac{\pi}{(2j-1)^2}.
\end{aligned}
$$

Similarly, we also have,

$$
\begin{aligned}
&\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x}-\beta|\sin\big((2j-1)\mathbf{x}\big)\sin\big((2j-1)\beta\big)d\mathbf{x}\;d\beta\\
=&2\int_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta|\sin\big((2j-1)\mathbf{x}\big)\sin\big((2j-1)\beta\big)d\mathbf{x}\;d\beta\\
=&2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin\big((2j-1)\beta\big)\bigg(\int_{\beta}^{\frac{\pi}{2}}\mathbf{x}\sin\big((2j-1)\mathbf{x}\big)d\mathbf{x}\bigg)d\beta\\
&-2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\bigg(\int_{\beta}^{\frac{\pi}{2}}\sin\big((2j-1)\mathbf{x}\big)d\mathbf{x}\bigg)d\beta\\
=&2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin\big((2j-1)\beta\big)\bigg(-\frac{(-1)^j}{(2j-1)^2}+\frac{\beta\cos\big((2j-1)\beta\big)}{(2j-1)}-\frac{\sin\big((2j-1)\beta\big)}{(2j-1)^2}\bigg)d\beta\\
&-2\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\bigg(\frac{\cos\big((2j-1)\beta\big)}{(2j-1)}\bigg)d\beta\\
=&-\frac{(-1)^j}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin\big((2j-1)\beta\big)d\beta-\frac{2}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\cos\big((2j-1)\beta\big)d\beta\\
&-\frac{2}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin^2\big((2j-1)\beta\big)d\beta+\frac{2}{(2j-1)}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\sin\big((2j-1)\beta\big)\cos\big((2j-1)\beta\big)d\beta\\
=&-\frac{(-1)^j}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin\big((2j-1)\beta\big)d\beta-\frac{2}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\sin^2\big((2j-1)\beta\big)d\beta\\
=&0-\frac{\pi}{(2j-1)^2}=-\frac{\pi}{(2j-1)^2}.
\end{aligned}
$$

Now, substitute the terms to get:

$$\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x}-\beta|\cos\big((2j-1)(\mathbf{x}-\beta)\big)d\mathbf{x}\;d\beta=\frac{\pi}{(2j-1)^2}-\frac{\pi}{(2j-1)^2}=0.$$

Now, for the second identity,

$$
\begin{aligned}
&\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta \\
=&\int_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta \\
+&\int_{\frac{-\pi}{2}\le\mathbf{x}<\beta\le\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta \\
=&\int_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)(\mathbf{x}-\beta)\big) d\mathbf{x}\, d\beta \\
+&\int_{\frac{-\pi}{2}\le-\beta<-\mathbf{x}\le\frac{\pi}{2}} |-\mathbf{x}+\beta| \sin\big((2j-1)(-\mathbf{x}+\beta)\big) d(-\mathbf{x})\, d(-\beta) \\
=&\iint_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)\mathbf{x}\big) \cos\big((2j-1)\beta\big) d\mathbf{x}\, d\beta \\
-&\iint_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}} |\mathbf{x}-\beta| \sin\big((2j-1)\mathbf{x}\big) \cos\big((2j-1)\beta\big) d\mathbf{x}\, d\beta = 0
\end{aligned}
$$

□

A.8. Verification of the two conditions in theorem 2.3.8 for example 2.3.9

Proof. In example 2.3.9 we had $H(\mathbf{x},\beta) = \frac{\pi}{2} - |\mathbf{x}-\beta|$, $u_{2j-1}(\mathbf{x}) = \frac{e^{-i(2j-1)\mathbf{x}}}{\sqrt{\pi}}$, $u_{2j}(\mathbf{x}) = \frac{e^{i(2j-1)\mathbf{x}}}{\sqrt{\pi}}$, $v_{2j-1}(\beta) = \frac{e^{-i(2j-1)\beta}}{\sqrt{\pi}}$, $v_{2j}(\beta) = \frac{e^{i(2j-1)\beta}}{\sqrt{\pi}}$, $\eta_{2j-1} = \frac{2}{(2j-1)^2}$ and $\eta_{2j} = \frac{2}{(2j-1)^2}$ for $j \in \mathbb{Z}^+$. So, we have:

$$\langle H, \eta_{2j-1}\bar{u}_{2j-1}v_{2j-1}\rangle = \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} H(\mathbf{x},\beta)\overline{\eta_{2j-1}\bar{u}_{2j-1}(\mathbf{x})v_{2j-1}(\beta)}d\mathbf{x}\ d\beta$$

$$=\frac{2}{\pi(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}(\frac{\pi}{2}-|\mathbf{x}-\beta|)e^{-i(2j-1)(\mathbf{x}-\beta)}d\mathbf{x}\ d\beta$$

$$=\frac{1}{(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} e^{-i(2j-1)(\mathbf{x}-\beta)}d\mathbf{x}\ d\beta - \frac{2}{\pi(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}|\mathbf{x}-\beta|e^{-i(2j-1)(\mathbf{x}-\beta)}d\mathbf{x}\ d\beta$$

$$=\frac{1}{(2j-1)^2}\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} e^{-i(2j-1)\mathbf{x}}d\mathbf{x}\Big)\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}} e^{i(2j-1)\beta}d\beta\Big)$$

$$-\frac{2}{\pi(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}|\mathbf{x}-\beta|\cos\big((2j-1)(\mathbf{x}-\beta)\big)d\mathbf{x}\ d\beta$$

$$+\frac{2}{\pi(2j-1)^2}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}|\mathbf{x}-\beta|\sin\big((2j-1)(\mathbf{x}-\beta)\big)d\mathbf{x}\ d\beta$$

$$=\frac{1}{(2j-1)^2}.\frac{2(-1)^{j-1}}{2j-1}.\frac{2(-1)^{j-1}}{2j-1}+0+0\ (\text{ from previous lemma A.5})$$

$$=\frac{4}{(2j-1)^4}=\eta_{2j-1}^2$$

We can similarly show $\langle H, \eta_{2j}\bar{u}_{2j}v_{2j}\rangle = \eta_{2j}^2$. So, $\langle H, \eta_j\bar{u}_jv_j\rangle = \eta_j^2$ for $j \in \mathbb{Z}^+$. We also have $\langle \eta_j\bar{u}_jv_j, H\rangle = \overline{\langle H, \eta_j\bar{u}_jv_j\rangle} = \eta_j^2$ for $j \in \mathbb{Z}^+$. Also, for $j,k \in \mathbb{Z}^+$ we have:

$$\langle \eta_j\bar{u}_jv_j, \eta_k\bar{u}_kv_k\rangle = \int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\eta_j\bar{u}_j(\mathbf{x})v_j(\beta)\overline{\eta_k\bar{u}_k(\mathbf{x})v_k(\beta)}d\mathbf{x}\ d\beta$$

$$=\eta_j\eta_k\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\bar{u}_j(\mathbf{x})u_k(\mathbf{x})d\mathbf{x}\Big)\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}v_j(\beta)\bar{v}_k(\beta)d\beta\Big)=\eta_j\eta_k\delta_{jk}$$

Now, let us verify the first condition 2.4 in theorem 2.3.8. We have:

$$\|H-\sum_{j=1}^{J}\eta_j\bar{u}_jv_j\|^2=\langle H-\sum_{j=1}^{J}\eta_j\bar{u}_jv_j,H-\sum_{j=1}^{J}\eta_j\bar{u}_jv_j\rangle$$

$$=\langle H,H\rangle-\langle H,\sum_{j=1}^{J}\eta_j\bar{u}_jv_j\rangle-\langle\sum_{j=1}^{J}\eta_j\bar{u}_jv_j,H\rangle+\langle\sum_{j=1}^{J}\eta_j\bar{u}_jv_j,\sum_{j=1}^{J}\eta_j\bar{u}_jv_j\rangle$$

$$=\|H\|^2-\sum_{j=1}^{J}\langle H,\eta_j\bar{u}_jv_j\rangle-\sum_{j=1}^{J}\langle\eta_j\bar{u}_jv_j,H\rangle+\sum_{j=1}^{J}\sum_{k=1}^{J}\langle\eta_j\bar{u}_jv_j,\eta_k\bar{u}_kv_k\rangle$$

$$=\|H\|^2-\sum_{j=1}^{J}\eta_j^2-\sum_{j=1}^{J}\eta_j^2+\sum_{j=1}^{J}\langle\eta_j\bar{u}_jv_j,\eta_j\bar{u}_jv_j\rangle$$

$$=\|H\|^2-\sum_{j=1}^{J}\eta_j^2-\sum_{j=1}^{J}\eta_j^2+\sum_{j=1}^{J}\eta_j^2=\|H\|^2-\sum_{j=1}^{J}\eta_j^2$$

So, LHS $=\|H-\sum_{j=1}^{J}\eta_j\bar{u}_jv_j\|=\sqrt{\|H\|^2-\sum_{j=1}^{J}\eta_j^2}=$ RHS, and the first condition is verified. For the second condition in 2.5, observe that:

$$\text{LHS}=\|H\|^2=\langle H,H\rangle=\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}H(\mathbf{x},\beta)^2d\mathbf{x}\;d\beta$$

$$=\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\big(\frac{\pi}{2}-|\mathbf{x}-\beta|\big)^2d\mathbf{x}\;d\beta$$

$$=\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\big(\frac{\pi^2}{4}-\pi|\mathbf{x}-\beta|+(\mathbf{x}-\beta)^2\big)d\mathbf{x}\;d\beta$$

$$=\frac{\pi^4}{4}+\pi\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\mathbf{x}^2d\mathbf{x}+\pi\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta^2d\beta-2\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\mathbf{x}d\mathbf{x}\Big)\Big(\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta d\beta\Big)-2\pi\iint\limits_{\frac{-\pi}{2}\le\beta<\mathbf{x}\le\frac{\pi}{2}}|\mathbf{x}-\beta|d\mathbf{x}\;d\beta$$

$$=\frac{\pi^4}{4}+\pi.\frac{\pi^3}{12}+\pi.\frac{\pi^3}{12}-2.0.0-2\pi\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\Big(\int_{\beta}^{\frac{\pi}{2}}\mathbf{x}d\mathbf{x}\Big)d\beta+2\pi\int_{\frac{-\pi}{2}}^{\frac{\pi}{2}}\beta\Big(\int_{\beta}^{\frac{\pi}{2}}d\mathbf{x}\Big)d\beta$$

$$=\frac{\pi^4}{4}+\frac{\pi^4}{6}+\frac{\pi^4}{6}-\frac{\pi^4}{3}-\frac{\pi^4}{6}=\frac{\pi^4}{12}$$

$$\text{RHS}=\sum_{j=1}^{\infty}\eta_j^2=\sum_{j=1}^{\infty}(\eta_{2j-1}^2+\eta_{2j}^2)=\sum_{j=1}^{\infty}\big((\frac{2}{(2j-1)^2})^2+(\frac{2}{(2j-1)^2})^2\big)$$

$$=8\sum_{j=1}^{\infty}\frac{1}{(2j-1)^4}=8\Big(\sum_{j=1}^{\infty}\frac{1}{(2j-1)^4}+\sum_{j=1}^{\infty}\frac{1}{(2j)^4}\Big)-8\sum_{j=1}^{\infty}\frac{1}{(2j)^4}$$

$$=8\sum_{j=1}^{\infty}\frac{1}{j^4}-\frac{8}{2^4}\sum_{j=1}^{\infty}\frac{1}{j^4}=8\big(1-\frac{1}{2^4}\big)\sum_{j=1}^{\infty}\frac{1}{j^4}=8\big(\frac{15}{16}\big)\big(\frac{\pi^4}{90}\big)=\frac{\pi^4}{12}$$

So LHS = RHS, and the second condition is also verified. □

A.9. Proof of Corollary 2.3.10

Proof. We have almost surely

$$|H - \sum_{j=1}^{J} \eta_j \bar{u}_j v_j| = |\sum_{j=J+1}^{\infty} \eta_j \bar{u}_j v_j| < \sup_{j\in\mathbb{Z}^+, \mathbf{x}\in\mathbb{X}} |u_j(\mathbf{x})| \sup |v_j(\beta)| \sum_{j=J+1}^{\infty} \eta_j$$

□

A.10. Proof of lemma 4.3.1

Proof. We already have point-wise convergence of $|z|^{[J]}$ to the limit $|z|$ for each $z \in [-\pi, \pi]$, now consider the tail sum $e_J(z) = |z| - |z|^{[J]}(z)$, we have:

$$\begin{aligned} |e^J(z)| = ||z| - |z|^{[J]}(z)| &= |\frac{4}{\pi} \sum_{j=J+1}^{\infty} \frac{\cos\big((2j-1)z\big)}{(2j-1)^2}| \\ &\leq \frac{4}{\pi} \sum_{j=J+1}^{\infty} \frac{|\cos\big((2j-1)z]|}{(2j-1)^2} = \frac{4}{\pi} \sum_{j=J+1}^{\infty} \frac{1}{(2j-1)^2} \end{aligned}$$

$\sum_{j=J+1}^{\infty} \frac{1}{(2j-1)^2}$ is the tail sum of a convergent series, given $\epsilon > 0$ we can choose J large enough to ensure $\frac{4}{\pi} \sum_{j=J+1}^{\infty} \frac{1}{(2j-1)^2} < \epsilon$. This implies $|e^J(z)| < \epsilon$ for all $z \in [-\pi, \pi]$ for large J. So, $|z|^{[J]}$ uniformly converges to $f(z)$.

Also, we note the functions $|z|^{[J]}$ are all bounded functions, as they converges uniformly, they are uniformly bounded. □

A.11. Proof of Corollary 4.3.2

Proof. Since $m \in (-1, 1)$ and $z \in (-1, 1)$, we have $z - m \in (-2, 2) \subset [-\pi, \pi]$. So the result follows from previous lemma. □

A.12. Proof of lemma 4.3.3

Proof. We already have point-wise convergence of $I^{[J]}(0 < z)$ to the limit $g(z)$ for each $z \in [-\pi, \pi]$, now consider the sum $E^J(z) = \sum_{j=1}^{J} \sin\big((2j-1)z\big) = \frac{\sin(2Jz)}{2\sin(z)}$.

We have $|E^J(z)| = \frac{|\sin(2Jz)|}{2|\sin(z)|} \leq \frac{1}{2|\sin(\delta)|}$, for all $z \in [-\pi, -\delta)\bigcup(\delta, \pi]$, for any J. Now consider the Cauchy tail sum $e^{J,K}(z) = I^{[K]}(0 < z) - I^{[J]}(0 < z)$ for $J < K$, we have:

$$\begin{aligned} e^{J,K}(z) &= \frac{2}{\pi}\sum_{j=J+1}^{L}\frac{\sin\big((2j-1)z\big)}{(2j-1)} = \frac{2}{\pi}\sum_{j=J+1}^{K}\frac{E^j(z)-E^{j-1}(z)}{(2j-1)} \\ &= \frac{2}{\pi}[\sum_{j=J+1}^{K}\frac{E^j(z)}{(2j-1)} - \sum_{j=J+1}^{K}\frac{E^J(z)}{(2j+1)} - \frac{E^J(z)}{(2J+1)} + \frac{E^K(z)}{(2K+1)}] \\ &= \frac{2}{\pi}[\sum_{j=J+1}^{K}E^j(z)\{\frac{1}{(2j-1)} - \frac{1}{(2j+1)}\} - \frac{E^J(z)}{(2J+1)} + \frac{E^K(z)}{(2K+1)}] \end{aligned}$$

This means:

$$\begin{aligned} |e_g^{K,L}(z)| &\leq \frac{1}{\pi|\sin(\delta)|}[\sum_{j=J+1}^{K}\{\frac{1}{(2j-1)} - \frac{1}{(2j+1)}\} + \frac{1}{(2J+1)} + \frac{1}{(2K+1)}] \\ &= \frac{1}{\pi|\sin(\delta)|}[\frac{1}{(2J+1)} - \frac{1}{(2K+1)} + \frac{1}{(2J+1)} + \frac{1}{(2K+1)}] = \frac{2}{(2J+1)\pi|\sin(\delta)|} \end{aligned}$$

Given $\epsilon > 0$ we can choose J large enough to ensure $\frac{2}{(2J+1)\pi|\sin(\delta)|} < \epsilon$. This implies $|e_g^{J,K}(z)| < \epsilon$ for all $z \in [-\pi, -\delta)\bigcup(\delta, \pi]$, for large J and $K > J$. So, by Cauchy criterion $I^{[J]}(0 < z)$ uniformly converges to $I(0 < z)$. □

A.13. Proof of Corollary 4.3.4

Proof. Since $-1 < a < b < 1$ and $z \in [(-1, 1) - \{(a - \delta, a + \delta)\bigcup(b - \delta, b + \delta)\}]$, we have $z - a \in (-2, -\delta)\bigcup(\delta, 2) \subset [-\pi, -\delta)\bigcup(\delta, \pi]$, similarly $z - b \in [-\pi, -\delta)\bigcup(\delta, \pi]$. Also observe that $g_{a,b}(z) = g(z - a) - g(z - b)$.So the result follows from previous lemma. □

A.14. Proof of lemma 4.3.5

Proof. Because $I^{[J]}(0 < z)$ is an odd function we have,$|I^{[J]}(0 < -z)| = |I^{[J]}(0 < z)|$, so it suffice to show that $I^{[J]}(0 < z)$ is uniformly bounded in the interval $[0, \pi]$. Now, for $z \in [0, \pi]$, we have the inequality $\frac{2}{\pi} \leq \frac{\sin(z)}{z} \leq 1$. The left hand inequality comes from the fact that the function $c(z) = \frac{\sin(z)}{z}$ has its maxima at $z = \frac{\pi}{2}$ where its value is $\frac{2}{\pi}$ and the right hand inequality is a standard trigonometric fact that holds for any

z. Let j_0 is biggest integer such that $2j_0 - 1 < \frac{1}{z}$. Then $z < \frac{1}{2j_0-1}$ and $z \geq \frac{1}{2j_0+1}$. Observe if $j < j_0$, then $(2j-1)z < (2j_0-1)z < 1 < \pi$, so that $\frac{\sin\big((2j-1)z\big)}{(2j-1)z} \leq 1$ or $\frac{\sin\big((2j-1)z\big)}{(2j-1)} \leq \frac{1}{z}$. On the other way, if $j > j_0$ then $\frac{1}{2j-1} \leq \frac{1}{2j_0+1}$.Given that, we have:

$$\begin{aligned} I^{[J]}(0 < z) &= \frac{1}{2} + \frac{2}{\pi}\sum_{j=1}^{J} \frac{\sin\big((2j-1)z\big)}{(2j-1)} \\ &= \frac{1}{2} + \frac{2}{\pi}[\sum_{j=1}^{j_0} \frac{\sin\big((2j-1)z\big)}{(2j-1)} + \sum_{k=j_0+1}^{J} \frac{\sin\big((2j-1)z\big)}{(2j-1)}] \end{aligned}$$

This means:

$$\begin{aligned} |I^{[J]}(0 < z)| &\leq \frac{1}{2} + \frac{2}{\pi}[|z|\sum_{j=1}^{j_0} |\frac{\sin\big((2j-1)z\big)}{\big((2j-1)z\big)}| + |\sum_{j=j_0+1}^{J} \frac{\sin\big((2j-1)z\big)}{(2j-1)}|] \\ &< \frac{1}{2} + \frac{2}{\pi}[z\sum_{j=1}^{j_0} 1 + \frac{1}{(2j_0+1)}|\sum_{j=j_0+1}^{J} \sin\big((2j-1)z\big]|] \\ &= \frac{1}{2} + \frac{2j_0 z}{\pi} + \frac{2}{\pi(2j_0+1)}|\sum_{j=j_0+1}^{J} \sin\big((2k-1)z\big]| \\ &< \frac{1}{2} + \frac{2j_0}{\pi(2j_0-1)} + \frac{2z}{\pi}|E^J(z) - E^{j_0}(z)| \\ &< \frac{1}{2} + \frac{2}{\pi} + \frac{2z}{\pi}.\frac{2}{\sin(\frac{z}{2})} = \frac{1}{2} + \frac{2}{\pi} + \frac{4}{\pi}.\frac{2}{\frac{\sin(\frac{z}{2})}{\frac{z}{2}}} \\ &\leq \frac{1}{2} + \frac{2}{\pi} + \frac{4}{\pi}.\frac{2}{\frac{2}{\pi}} = \frac{9}{2} + \frac{2}{\pi}. \end{aligned}$$

So the sequence of functions $I^{[J]}(0 < z)$ is uniformly bounded by the number $\frac{9}{2} + \frac{2}{\pi}$. □

A.15. Proof of Theorem 4.3.7

Proof. From Corollary 4.3.2 we know $F_J^0(\mathbf{x}) = |x_t - m|^{[J]}$ converges uniformly to $F^0(\mathbf{x}) = |x_t - m|$ on E and is uniformly bounded. From Corollary 4.3.4 we know $F_J^l(\mathbf{x}) = I^{[J]}(0 < x_l - a_l) - I^{[J]}(0 < x_l - b_l)$ converges uniformly to $F^l(\mathbf{x}) = I(0 < x_l - a_l) - I(0 < x_l - b_l)$ on $E - N_\delta^{\mathbf{a},\mathbf{b}}$ and from lemma 4.3.5 is uniformly bounded on

E. Note that $F(\mathbf{x}) = \prod_{l=0}^{p} F^l(\mathbf{x})$ and $F_J(\mathbf{x}) = \prod_{l=0}^{p} F_J^l(\mathbf{x})$. So by using lemma 4.3.6 p times, we get $F_J(\mathbf{x}) = I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ uniformly converges to $F(\mathbf{x}) = I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ on $(E - N_\delta^{\mathbf{a},\mathbf{b}}) \bigcap E = E - N_\delta^{\mathbf{a},\mathbf{b}}$. □

A.16. Proof of Theorem 4.3.8

Proof. Let $\epsilon > 0$ given. From previous lemma 4.3.1 and 4.3.3, we know that $I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ is uniformly bounded on $(-1, 1)^p$, also $I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ is bounded for being an indicator function. So that their difference $\mathbf{z}^J = e_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})$ is also uniformly bounded in $(-1, 1)^p$. Let this bound is M, so, $|\mathbf{z}^J| < M$, for any J and any $\mathbf{x} \in (-1, 1)^p$. Now since P is absolutely continuous with respect to λ, there exist a $\eta > 0$, so that if $\lambda(A) < \eta$, then, $P(A) < \frac{\epsilon}{2M}$ for any $A \in \mathbb{B}(-1, 1)^p$. Now pick a δ such that $\lambda(N_\delta^{\mathbf{a},\mathbf{b}}) = (2\delta)^p < \eta$. Set $A = N_\delta^{\mathbf{a},\mathbf{b}}$, so that we have, $P(A) < \frac{\epsilon}{2M}$. Next, because of uniform convergence on $E - A$, by Theorem 4.3.7, we can choose a J large enough to have $|\mathbf{z}^J| = |I_t(m, \mathbf{a}, \mathbf{b}, \mathbf{x}) - I_t^J(m, \mathbf{a}, \mathbf{b}, \mathbf{x})| < \frac{\epsilon}{2}$, for any $\mathbf{x} \in E - A$. Then we have:

$$
\begin{aligned}
|E_P(\mathbf{z}^J)| = |\int_E z^J \mathrm{d}P| &= |\int_{E-A} z^J \mathrm{d}P + \int_A z^J \mathrm{d}P| \leq |\int_{E-A} z^J \mathrm{d}P| + |\int_A z^J \mathrm{d}P| \\
&\leq \int_{E-A} |z^J| \mathrm{d}P + \int_A |z^J| \mathrm{d}P < \int_{E-A} \frac{\epsilon}{2} \mathrm{d}P + \int_A M \mathrm{d}P \\
&= \frac{\epsilon}{2}.P(E - A) + M.P(A) \leq \frac{\epsilon}{2}.1 + M.P(A) < \frac{\epsilon}{2} + M.\frac{\epsilon}{2M} < \epsilon
\end{aligned}
$$

and so, $E_P(\mathbf{z}^J) \to 0$ as $J \to \infty$. □

A.17. Proof of Theorem 4.3.9

Proof. Since, $E_P(|\mathbf{z}^J)|) < M$, an application of Kolmogorov's SLLN yields the result.

□

A.18. Proof of lemma 4.4.1

Proof. We have $I^J(x,m,a,b) = |x-m|^{[J]} I^{[J]}(a<x<b)$, here

$$|x-m|^{[J]} = \sum_{j=0}^{2J} f_j(m) c_j(x)$$

$$I^{[J]}(a<x<b) = \sum_{j=0}^{2J} g_j(a,b) c_j(x)$$

Then,

$$\begin{aligned}
&I^J(x,m,a,b)\\
=&f_0(m)g_0(a,b) + \sum_{j=1}^{2J} \big(f_0(m)g_j(a,b) + f_j(m)g_0(a,b)\big)c_j(x)\\
&+R^J(x,m,a,b)
\end{aligned}$$

Here:

$$\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{k=1}^{J} \Big(f_{2j-1}(m)\cos\big((2j-1)x\big) + f_{2j}(m)\sin\big((2j-1)x\big)\Big)\\
&\times\Big(g_{2j-1}(a,b)\cos\big((2j-1)x\big) + g_{2j}(a,b)\sin\big((2j-1)x\big)\Big)
\end{aligned}$$

So,

$$\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J} \Big(f_{2j-1}(m)g_{2k-1}(a,b)\times 2\cos\big((2j-1)x\big)\cos\big((2j-1)x\big)\Big)\\
&+\sum_{j=1}^{J}\sum_{k=1}^{J} \Big(f_{2j-1}(m)g_{2k}(a,b)\times 2\cos\big((2j-1)x\big)\sin\big((2k-1)x\big)\Big)\\
&+\sum_{j=1}^{J}\sum_{k=1}^{J} \Big(f_{2j}(m)g_{2k-1}(a,b)\times 2\sin\big((2j-1)x\big)\cos\big((2k-1)x\big)\Big)\\
&+\sum_{j=1}^{J}\sum_{k=1}^{J} \Big(f_{2j}(m)g_{2k}(a,b)\times 2\sin\big((2j-1)x\big)\sin\big((2k-1)x\big)\Big)
\end{aligned}$$

or,

$$
\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J} f_{2j-1}(m)g_{2k-1}(a,b)\Big(\cos\big(2(j+k-1)x\big)+\cos\big(2(j-k)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J} f_{2j-1}(m)g_{2k}(a,b)\Big(\sin\big(2(j+k-1)x\big)-\sin\big(2(j-k)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J} f_{2j}(m)g_{2k-1}(a,b)\Big(\sin\big(2(j+k-1)x\big)+\sin\big(2(j-k)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J} f_{2j}(m)g_{2k}(a,b)\Big(-\cos\big(2(j+k-1)x\big)+\cos\big(2(j-k)x\big)\Big)
\end{aligned}
$$

Interchanging the terms,

$$
\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J}\Big(f_{2j-1}(m)g_{2k-1}(a,b)-f_{2j}(m)g_{2k}(a,b)\big)\cos\big(2(j+k-1)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}\Big(f_{2j-1}(m)g_{2k}(a,b)+f_{2j}(m)g_{2k-1}(a,b)\big)\sin\big(2(j+k-1)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}\Big(f_{2j-1}(m)g_{2k-1}(a,b)+f_{2j}(m)g_{2k}(a,b)\big)\cos\big(2(j-k)x\big)\Big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}\Big(f_{2j}(m)g_{2k-1}(a,b)-f_{2j-1}(m)g_{2k}(a,b)\big)\sin\big(2(j-k)x\big)\Big)
\end{aligned}
$$

or,

$$
\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J}A_{j,k}(m,a,b)\cos\big(2(j+k-1)x\big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}B_{j,k}(m,a,b)\sin\big(2(j+k-1)x\big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}C_{j,k}(m,a,b)\cos\big(2(j-k)x\big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)
\end{aligned}
$$

Here,

$$
\begin{aligned}
A_{j,k}(m,a,b) &= f_{2j-1}(m)g_{2k-1}(a,b) - f_{2j}(m)g_{2k}(a,b)\\
B_{j,k}(m,a,b) &= f_{2j-1}(m)g_{2k}(a,b) + f_{2j}(m)g_{2k-1}(a,b)\\
C_{j,k}(m,a,b) &= f_{2j-1}(m)g_{2k-1}(a,b) + f_{2j}(m)g_{2k}(a,b)\\
D_{j,k}(m,a,b) &= f_{2j}(m)g_{2k-1}(a,b) - f_{2j-1}(m)g_{2k}(a,b)
\end{aligned}
$$

Lets simplify these expressions we have:

$$
\begin{aligned}
&A_{j,k}(m,a,b) = f_{2j-1}(m)g_{2k-1}(a,b) - f_{2j}(m)g_{2k}(a,b)\\
=&-\frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(-\sin\big((2k-1)a\big)I(a>-1)+\sin\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
+&\frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(\cos\big((2k-1)a\big)I(a>-1)-\cos\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
=&\frac{8\Big(\sin\big((2j-1)m+(2k-1)a\big)I(a>-1)-\sin\big((2j-1)m+(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)}
\end{aligned}
$$

$$
\begin{aligned}
&B_{j,k}(m,a,b) = f_{2j-1}(m)g_{2k}(a,b) + f_{2j}(m)g_{2k-1}(a,b)\\
=&-\frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(\cos\big((2k-1)a\big)I(a>-1)-\cos\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
+&\frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(-\sin\big((2k-1)a\big)I(a>-1)+\sin\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
=&-\frac{8\Big(\cos\big((2j-1)m+(2k-1)a\big)I(a>-1)-\cos\big((2j-1)m+(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)}
\end{aligned}
$$

$$
\begin{aligned}
&C_{j,k}(m,a,b) = f_{2j-1}(m)g_{2k-1}(a,b) + f_{2j}(m)g_{2k}(a,b)\\
=&-\frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(-\sin\big((2k-1)a\big)I(a>-1)+\sin\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
-&\frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(\cos\big((2k-1)a\big)I(a>-1)-\cos\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
=&-\frac{8\Big(\sin\big((2j-1)m-(2k-1)a\big)I(a>-1)-\sin\big((2j-1)m-(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)}
\end{aligned}
$$

$$
\begin{aligned}
&D_{j,k}(m,a,b) = f_{2j}(m)g_{2k-1}(a,b) - f_{2j-1}(m)g_{2k}(a,b)\\
=&-\frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(-\sin\big((2k-1)a\big)I(a>-1)+\sin\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
+&\frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{2\Big(\cos\big((2k-1)a\big)I(a>-1)-\cos\big((2k-1)b\big)I(b<1)\Big)}{\pi(2k-1)}\\
=&\frac{8\Big(\cos\big((2j-1)m-(2k-1)a\big)I(a>-1)-\cos\big((2j-1)m-(2k-1)b\big)I(b<1)\Big)}{\pi^2(2j-1)^2(2k-1)}
\end{aligned}
$$

Then,

$$
\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J}\big(A_{j,k}(m,a,b)\cos\big(2(j+k-1)x\big)+B_{j,k}(m,a,b)\sin\big(2(j+k-1)x\big)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{J}\big(C_{j,k}(m,a,b)\cos\big(2(j-k)x\big)+D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)
\end{aligned}
$$

or

$$
\begin{aligned}
&R^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J} A_{j,k}(m,a,b)\cos\big(2(j+k-1)x\big)+B_{j,k}(m,a,b)\sin\big(2(j+k-1)x\big)\\
+&\sum_{j=1}^{J}\sum_{k=j+1}^{J} C_{j,k}(m,a,b)\cos\big(2(j-k)x\big)+D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)\\
+&\sum_{k=1}^{J} C_{k,k}(m,a,b)\\
+&\sum_{j=1}^{J}\sum_{k=1}^{j-1} C_{j,k}(m,a,b)\cos\big(2(j-k)x\big)+D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)
\end{aligned}
$$

Let $R^J(x,m,a,b) = R_1^J(x,m,a,b) + R_2^J(x,m,a,b) + R_3^J(m,a,b) + R_4^J(x,m,a,b)$, where

$$
\begin{aligned}
&R_1^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{J} A_{j,k}(m,a,b)\cos\big(2(j+k-1)x\big)+B_{j,k}(m,a,b)\sin\big(2(j+k-1)x\big)\\
=&\sum_{j=1}^{J-1}\Big(\sum_{k=1}^{j} A_{k,j-k+1}(m,a,b)\cos(2jx)+B_{k,j-k+1}(m,a,b)\sin(2jx)\Big)\\
+&\sum_{j=J}^{2J-1}\Big(\sum_{k=j-J+1}^{j} A_{k,j-k+1}(m,a,b)\cos(2jx)+B_{k,j-k+1}(m,a,b)\sin(2jx)\Big)
\end{aligned}
$$

$$
\begin{aligned}
&R_2^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=j+1}^{J} C_{j,k}(m,a,b)\cos\big(2(j-k)x\big)+D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)\\
=&\sum_{j=1}^{J-1}\Big(\sum_{k=1}^{J-j} C_{k,k+j}(m,a,b)\cos(2jx)-D_{k,k+j}(m,a,b)\sin(2jx)\Big)
\end{aligned}
$$

$$R_3^J(m,a,b) = \sum_{k=1}^{J} C_{k,k}(m,a,b)$$

$$
\begin{aligned}
&R_4^J(x,m,a,b)\\
=&\sum_{j=1}^{J}\sum_{k=1}^{j-1} C_{j,k}(m,a,b)\cos\big(2(j-k)x\big) + D_{j,k}(m,a,b)\sin\big(2(j-k)x\big)\\
=&\sum_{j=1}^{J-1}\Big(\sum_{k=1}^{J-j} C_{k+j,k}(m,a,b)\cos(2jx) + D_{k+j,k}(m,a,b)\sin(2jx)\Big)
\end{aligned}
$$

Let $\zeta_{2j-1}(x) = \cos(2jx)$ and $\zeta_{2j}(x) = \sin(2jx)$ for $j \in \mathbb{Z}^+$. Also let $\xi_0(m,a,b)$, $\xi_{2j-1}(m,a,b)$ and $\xi_{2j}(m,a,b)$ denote the constant term and the coefficients of $\zeta_{2j-1}(x)$ and $\zeta_{2j}(x)$ in the series expression of $R^J(x,m,a,b)$. Then we have:

$$\xi_0(m,a,b) = \sum_{k=1}^{J} C_{k,k}(m,a,b)$$

For $1 \le j \le J-1$

$$
\begin{aligned}
&\xi_{2j-1}(m,a,b)\\
=&\sum_{k=1}^{j} A_{k,j-k+1}(m,a,b) + \sum_{k=1}^{J-j} C_{k,k+j}(m,a,b) + \sum_{k=1}^{J-j} C_{k+j,k}(m,a,b)\\
&\xi_{2j}(m,a,b)\\
=&\sum_{k=1}^{j} B_{k,j-k+1}(m,a,b) - \sum_{k=1}^{J-j} D_{k,k+j}(m,a,b) + \sum_{k=1}^{K-j} D_{k+j,k}(m,a,b)
\end{aligned}
$$

For $J \le j \le 2J-1$

$$
\begin{aligned}
\xi_{2j-1}(m,a,b) &= \sum_{j-J+1}^{j} A_{k,j-k+1}(m,a,b)\\
\xi_{2j}(m,a,b) &= \sum_{j-J+1}^{j} B_{k,j-k+1}(m,a,b)
\end{aligned}
$$

So we have,

$$I^J(x,m,a,b) = \omega_0^J(m,a,b) + \sum_{j=1}^{J}\omega_j(m,a,b)c_j(x) + \sum_{j=1}^{4J-2}\xi_j(m,a,b)\zeta_j^J(x)$$

Here

$$\omega_0^J(m,a,b) = f_0(m)g_0(a,b) + \sum_{k=1}^{J} C_{k,k}(m,a,b)$$

$$= \frac{\pi}{4} - \sum_{k=1}^{J} \frac{8\Big(\sin\big((2k-1)(m-a)\big)I(a>-1) - \sin\big((2k-1)(m-b)\big)I(b<1)\Big)}{\pi^2(2k-1)^3}$$

$$\omega_{2j-1}(m,a,b) = f_0(m)g_{2j-1}(a,b) + f_{2j-1}(m)g_0(a,b)$$

$$= \frac{\pi}{2}\cdot\frac{2\Big(\sin\big((2j-1)b\big)I(b<1) - \sin\big((2j-1)a\big)I(a>-1)\Big)}{\pi(2j-1)} - \frac{4\cos\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{1}{2}$$

$$= \frac{\sin\big((2j-1)b\big)I(b<1) - \sin\big((2j-1)a\big)I(a>-1)}{(2j-1)} - \frac{2\cos\big((2j-1)m\big)}{\pi(2j-1)^2}$$

$$\omega_{2j}(m,a,b) = f_0(m)g_{2j}(a,b) + f_{2j}(m)g_0(a,b)$$

$$= \frac{\pi}{2}\cdot\frac{2\Big(\cos\big((2j-1)a\big)I(a>-1) - \cos\big((2j-1)b\big)I(b<1)\Big)}{\pi(2j-1)} - \frac{4\sin\big((2j-1)m\big)}{\pi(2j-1)^2}\cdot\frac{1}{2}$$

$$= \frac{\cos\big((2j-1)a\big)I(a>-1) - \cos\big((2j-1)b\big)I(b<1)}{(2j-1)} - \frac{2\sin\big((2j-1)m\big)}{\pi(2j-1)^2}$$

$$\xi_{2j-1}^J(m,a,b)$$

$$= \sum_{k=\max\{1,j-J+1\}}^{j} A_{k,j-k+1}(m,a,b) + I(j<J)\sum_{k=1}^{J-j}\big(C_{k+j,k}(m,a,b) + C_{k,k+j}(m,a,b)\big)$$

$$\xi_{2j}^J(m,a,b)$$

$$= \sum_{k=\max\{1,j-J+1\}}^{j} B_{k,j-k+1}(m,a,b) + I(j<J)\sum_{k=1}^{J-j}\big(D_{k+j,k}(m,a,b) - D_{k,k+j}(m,a,b)\big)$$

□

B. ALGORITHMS:

B.1 Algorithm: EP-FQ0

Algorithm B.1 Map-Reduce algorithm for EP-FQ0 method of computing approximate quantiles

Input: A distributed Data-Frame(DDF) that has $\mathbf{x}$ as a component.
Output: A list of Quantiles.

procedure DR-EP.FQ-0(conn,J_0,p)
(conn = HDFS Connection to the DDF, J_0 = Number of terms, Pr = A vector of f-values)
Get $M = Max(\mathbf{X}) + \delta$, $m = Min(\mathbf{X}) - \delta$, $N(\mathbf{X}) = Sample\ Size$.
Here δ is a very small number. (Need a pre-map-reduce step if any of them are not given).
Get the scaling parameters $A = -\frac{M+m}{M-m}$, $B = \frac{2}{M-m}$.
Map()
Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
Initialize $2J_0$ length vector $c = 0$.
Start with the 0 length null vector V.
for Each Map-key in the Mapper **do**
Trnasform $\mathbf{x}$ column of the corresponding Map-value(usually a data frame) as $\hat{\mathbf{x}} = A\mathbf{x} + B$
Concatenate with $\hat{\mathbf{x}}$ with V.
end for
for Each element of v of V **do**
for $j = 1, 2, \cdots, J_0$ **do**
c(2j-1) = c(2j-1)+cos$\big($2j-1)v$\big)$.
c(2j) = c(2j)+sin$\big($2j-1)v$\big)$.
end for
end for
for $j = 1, 2, \cdots, 2J_0$ **do**
Collect(key = j, value = $c(j)$)
end for
end Map()
Reduce()
Input : key = Reduce-key(An integer j), values = Reduce-Values(real numbers)
Initialize real number $s = 0$.
for Each Reduce-Values v **do**
Set $s = s + v$
end for
collect(key = Reduce-key(j), value = $\frac{s}{N(\mathbf{X})}$)
Collect(key = 0, value = subset-size N)
end Reduce()
Execute a Map-Reduce job and read all the key-value pairs.
Get all terms corresponding to odd indices as keys, divide by N and store in a J_0 length vector $\bar{C}$.
Get all terms corresponding to even indices as keys and store in a J_0 length vector $\bar{S}$.
Let $f_P(\beta) = \frac{\pi}{2}\sum_{j=1}^{J_0} \frac{\bar{C}(j)cos\big((2j-1)\beta\big)+\bar{S}(j)sin\big((2j-1)\beta\big)}{(2j-1)^2} + p\beta$
for Each P in Pr **do**
Get $\hat{Q}_P$ by maximizing the function $f_P(\beta)$ w.r.t. β.
end for
Return($(\hat{Q}_P - B)/A$ for all P)
end procedure

B.2 Algorithm: Binning

Algorithm B.2 Map-Reduce algorithm for Binning method of computing approximate quantiles

Input: A distributed Data-Frame that has X as a component(like a DDF object in R package *Datadr*).
Output: A list of Quantiles.

procedure drQuantileBinning(conn = HDFS Connection to the DDF, nBins = Number of Bins, p = A vector of f-values)
  Get M = maximum(X), m = minimum(X).(Perform a pre map-reduce step if any of these are not given).
  Set $\delta = \frac{M-m}{nBins}$.
  Get the vector Cut, it is a sequence of monotonically increasing cut points staring from $m - \frac{\delta}{2}$ and ending at $M + \frac{\delta}{2}$.
  There are $nBins + 1$ number of cut points and $nBins$ bins of width δ. For simplicity, denote the bins by bin-index b.
  Map()
  Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
  Initialize $nBins$ length vector $f = 0$.
  Start with the 0 length null vector V.
  for Each Map-key in the Mapper **do**
    Concatenate x column of the corresponding Map-value(usually a data frame) with V.
  end for
  Initialize $size(V)$ length vector $B = 0$.
  for Each element of v of V **do** Use a binary search procedure over the bins to get the bin $B(v)$ that v corresponds to.
  end for
  From the vector B, count and tabulate frequency count $f(b)$ for each bin $b; 1 \leq b \leq nBins$.
  for $b = 1, 2, \cdots, nBins$ **do**
    Collect(key = b, value = $f(b)$)
  end for
  end Map()
  Reduce()
  Input : key = Reduce-key(A bin b), values = Reduce-Values(Integers)
  Initialize real number $s = 0$.
  for Each Reduce-Values v **do**
    Set $s = s + v$
  end for
  collect(key = Reduce-key(k), value = s)
  end Reduce()
  Execute a Map-Reduce job and read all the key-value pairs.
  Get total frequency count $T(b)$ for each bin $b; 1 \leq b \leq nBins$.
  for Each p in P **do**
    Get Q_p from F, by calculating cumulative frequency and interpolation.
  end for
  Return(Q_p for all p)
end procedure

B.3 Algorithm: EP-FQCh1

Algorithm B.3 Map-Reduce algorithm for EP-FQCh1 method of computing approximate quantiles

Input: A distributed Data-Frame(DDF) that has $\mathbf{x}$ as a component.
Output: A list of Quantiles.

procedure DR-EP.FQ-CH1(conn,J_0,p)
  (conn = HDFS Connection to the DDF, J_0 = Number of terms, p = A vector of f-values)
  Get $M = Max(\mathbf{X}) + \delta$, $m = Min(\mathbf{X}) - \delta$.
  Here δ is a very small number. (Need a pre-map-reduce step if any of them are not given).
  Get the scaling parameters $A = -\frac{M+m}{M-m}$, $B = \frac{2}{M-m}$.
  Map()
  Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
  Initialize $2J_0$ length vector $C = 0$ and J_0 length vectors $c = 0$ and $s = 0$.
  Start with the 0 length null vector V.
  for Each Map-key in the Mapper **do**
    Trnasform $\mathbf{x}$ column of the corresponding Map-value(usually a data frame) as $\hat{\mathbf{x}} = A\mathbf{x} + B$
    Concatenate with $\hat{\mathbf{x}}$ with V.
  end for
  for Each element of v of V **do**
    $c_0 = cos(v)$.
    $s_0 = \sqrt{1 - c_0^2}$.
    for $j = 1, 2, \cdots, J_0$ **do**
      $c(j) = c(j) + c_0^{2j-1}$.
      $s(j) = s(j) + s_0 \times c_0^{2j-2}$.
    end for
  end for
  Combine and store c and s in the big vector C.
  for $j = 1, 2, \cdots, 2J_0$ **do**
    Collect(key = j, value = $C(j)$)
    Collect(key = 0, value = subset-size N)
  end for
  end Map()
  Reduce()
  Input : key = Reduce-key(An integer j), values = Reduce-Values(real numbers)
  Initialize real number $s = 0$.
  for Each Reduce-Values v **do**
    Set $s = s + v$
  end for
  collect(key = Reduce-key(j), value = v)
  end Reduce()
  Execute a Map-Reduce job and read all the key-value pairs.
  Collect value N for key = 0.
  Initialize J_0 length vectors $\bar{C} = 0, \overline{SC} = 0$ and $\bar{S} = 0$.
  Collect values with keys = $1, \cdots, J_0$ as keys, divide by N and store in J_0 length vector $\bar{C}$.
  Apply Chebyshev's type A recursion (algorithm B.6) on $\bar{C}$ to get new $\bar{C}$.
  Collect values with keys = $J_0 + 1, \cdots, 2J_0$ as keys, divide by N and store in J_0 length vector $\overline{SC}$.
  Set $\bar{S}(1) = \overline{SC}(1)$.
  for $j = 2, \cdots, J_0$ **do**
    $\bar{S}(j) = \bar{S}(1) - \sum_{j'=2}^{j} \overline{SC}(j)$
  end for
  Apply Chebyshev's type A recursion (algorithm B.6) on $\bar{S}$ to get new $\bar{S}$.
  Multiply odd terms of $\bar{S}$ with -1 to get new $\bar{S}$.
  Let $f_P(\beta) = \frac{\pi}{2} \sum_{j=1}^{J_0} \frac{\bar{C}(j)cos\left((2j-1)\beta\right) + \bar{S}(j)sin\left((2j-1)\beta\right)}{(2j-1)^2} + p\beta$
  for Each P in Pr **do**
    Get $\hat{Q}_P$ by maximizing the function $f_P(\beta)$ w.r.t. β.
  end for
  Return($(\hat{Q}_P - B)/A$ for all P)
end procedure

B.4 Algorithm: EP-FQCh2

Algorithm B.4 Map-Reduce algorithm for EP-FQCh2 method of approximating quantiles

Input: A distributed Data-Frame(DDF) that has $\mathbf{x}$ as a component.
Output: A list of Quantiles.

procedure DR-EP.FQ-CH2(conn,J,K,p)
(conn = HDFS Connection to the DDF, J, K = Parameters of EP.FQCh2, p = A vector of f-values)
Get $M = Max(\mathbf{X}) + \delta$, $m = Min(\mathbf{X}) - \delta$.
Here δ is a very small number. (Need a pre-map-reduce step if any of them are not given).
Get the scaling parameters $A = -\frac{M+m}{M-m}$, $B = \frac{2}{M-m}$.
Map()
Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
Initialize $2JK$ length vector $C = 0$ and $J \times K$ dimentional matrices $c = 0$ and $s = 0$.
Start with the 0 length null vector V.
for Each Map-key in the Mapper **do**
Transform $\mathbf{x}$ column of the corresponding Map-value(usually a data frame) as $\hat{\mathbf{x}} = A\mathbf{x} + B$
Concatenate with $\hat{\mathbf{x}}$ with V.
end for
for Each element of v of V **do**
$c_1 = cos(v)$.
$c_2 = cos(2Jv)$.
$s_1 = \sqrt{1 - c_0^2}$.
for $j = 1, \cdots, J$ **do**
for $k = 1, \cdots, K$ **do**
$c(j,k) = c(j,k) + c_1^{2j-1} c_2^{k-1}$.
$s(j,k) = s(j) + s_1 c_1^{2j-2} c_2^{k-1}$.
end for
end for
end for
Combine and store c and s in the big vector C.
for $j = 1, 2, \cdots, 2JK$ **do**
Collect(key = j, value = $C(j)$)
Collect(key = 0, value = subset-size N)
end for
end Map()
Reduce()
Input : key = Reduce-key(An integer j), values = Reduce-Values(real numbers)
Initialize real number $s = 0$.
for Each Reduce-Values v **do**
Set $s = s + v$
end for
collect(key = Reduce-key(j), value = s)
end Reduce()
Execute a Map-Reduce job and read all the key-value pairs.
Collect value N for key = 0.
Initialize $J \times K$ dimentional matrices $\bar{\Gamma} = 0$, $\overline{SC} = 0$ and $\bar{\Delta} = 0$.
Collect values with keys $= 1, \cdots, JK$ as keys, divide by N and store in $J \times K$ dimentional matrix $\bar{\Gamma}$.
Apply Chebyshev's type A recursion (algorithm B.6) along the rows of $\bar{\Gamma}$ to get new $\bar{\Gamma}$.
Apply Chebyshev's type B recursion (algorithm B.7) along the columns of $\bar{\Gamma}$ to get new $\bar{\Gamma}$.
Apply Chebyshev's type C recursion (algorithm B.8) on the matrix $\bar{\Gamma}$ to get vector $\bar{C}$.
Collect values with keys $= JK + 1, \cdots, 2JK$ as keys, divide by N and store in $J \times K$ dimentional matrix $\overline{SC}$.
for $k = 1, \cdots, K$ **do**
Set $\bar{\Delta}(1,k) = \overline{SC}(1,k)$.
for $j = 2, \cdots, J$ **do**
$\bar{\Delta}(j,k) = \bar{\Delta}(1,k) - \sum_{j'=2}^{j} \overline{SC}(j',k)$
end for
end for
Apply Chebyshev's type A recursion (algorithm B.6) along the rows of $\bar{\Delta}$ to get new $\bar{\Delta}$.
Apply Chebyshev's type B recursion (algorithm B.7) along the columns of $\bar{\Delta}$ to get new $\bar{\Delta}$.
Multiply odd rows of $\bar{\Delta}$ with -1 to get new $\bar{\Delta}$.
Apply Chebyshev's type D recursion (algorithm B.9) on the matrix $\bar{\Delta}$ to get vector $\bar{S}$.
Let $f_P(\beta) = \frac{\pi}{2} \sum_{j=1}^{JK} \frac{\bar{C}(j)cos\left((2j-1)\beta\right) + \bar{S}(j)sin\left((2j-1)\beta\right)}{(2j-1)^2} + p\beta$
for Each P in Pr **do**
Get $\hat{Q}_P$ by maximizing the function $f_P(\beta)$ w.r.t. β.
end for

B.5 Algorithm: EP-FQChp

Algorithm B.4 Map-Reduce algorithm for EP-FQCh2 method of approximating quantiles

Input: A distributed Data-Frame(DDF) that has **x** as a component.
Output: A list of Quantiles.

procedure DRQUANTILEFOURIER(conn = HDFS Connection to the DDF, $J_1, J_2, \cdots, J_p$ = Parameters of p-fold fourier method, pr = A vector of f-values)
(conn = HDFS Connection to the DDF, $J_1, J_2, \cdots, J_p$ = Parameters of EP.FQChp, $Probs$ = A vector of f-values)
Get $M = Max(\mathbf{X}) + \delta$, $m = Min(\mathbf{X}) - \delta$.
Here δ is a very small number. (Need a pre-map-reduce step if any of them are not given).
Get the scaling parameters $A = -\frac{M+m}{M-m}$, $B = \frac{2}{M-m}$.
Let $\mathbb{N}_{\boldsymbol{J}} = \{1, \cdots, J_1\} \times \{1, \cdots, J_2\} \times \cdots \times \{1, \cdots, J_p\}$.
Let $J_0 = J_1.J_2.\cdots.J_p$.
Map()
Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
Initialize $2J_1J_2\cdots J_p$ length vector $C = 0$ and p dimentional arrays $c = 0$ and $s = 0$, with dimension $J_1, J_2, \cdots, J_p$.
Start with the 0 length null vector V.
for Each Map-key in the Mapper **do**
Concatenate x column of the corresponding Map-value(usually a data frame) with V.
end for
for Each element of v of V **do**
$c_1 = \cos(v), c_2 = \cos(2J_1v), c_3 = \cos(2J_1J_2v), \cdots, c_p = \cos(2J_1J_2\cdots J_{p-1}v)$.
$s_1 = \sqrt{1 - c_1^2}$
for Every index $\boldsymbol{j} \in \mathbb{N}_{\boldsymbol{J}}$ **do**
$c(\boldsymbol{j}) = c(\boldsymbol{j}) + c_1^{2j_1-1}c_2^{j_2-1}\cdots c_p^{j_p-1}$.
$s(\boldsymbol{j}) = s(\boldsymbol{j}) + s_1c_1^{2j_1-2}c_2^{j_2-1}\cdots c_p^{j_p-1}$.
end for
end for
Combine and store c and s in the big vector C.
for $j = 1, 2, \cdots, 2J_0$ **do**
Collect(key = $\boldsymbol{j}$, value = $C(\boldsymbol{j})$)
Collect(key = 0, value = subset-size N)
end for
end Map()
Reduce()
Input : key = Reduce-key(An integer j), values = Reduce-Values(real numbers)
Initialize real number $s = 0$.
for Each Reduce-Values v **do**
Set $s = s + v$
end for
collect(key = Reduce-key(j), value = s)
end Reduce()
Execute a Map-Reduce job and read all the key-value pairs.
Get all terms corresponding odd indices as keys and store in a $J_1 \times J_2 \times \cdots \times J_p$ dimentional array $\hat{C}$.
Apply algorithm A (algorithm B.6) along 1^{st} co-ordinate of the array $\hat{C}$.
Apply algorithm B (algorithm B.7) along $2^{nd}, 3^{rd}, \cdots, p^{th}$ co-ordinate of the output matrix,
Apply algorithm C (algorithm B.8) on the output matrix recursively to get J_0 length vector $\bar{C}$.
Get all terms corresponding to even indices as keys, divide by N and store in a $J_1 \times J_2 \times \cdots \times J_p$ dimentional array $\hat{S}$.
Apply algorithm A (algorithm B.6) along 1^{st} co-ordinate of the array $\hat{C}$.
Apply algorithm B (algorithm B.7) along $2^{nd}, 3^{rd}, \cdots, p^{th}$ co-ordinate of the output matrix,
Multiply the 1^{st} co-ordinate even entries of the output matrix by -1
Apply algorithm D (algorithm B.9) on the output matrix recursively to get J_0 length vector $\bar{S}$.
Let $f_P(\beta) = \frac{\pi}{2}\sum_{j=1}^{J_0} \frac{\bar{C}(j)cos\big((2j-1)\beta\big) + \bar{S}(j)sin\big((2j-1)\beta\big)}{(2j-1)^2} + p\beta$
for Each P in Pr **do**
Get $\hat{Q}_P$ by maximizing the function $f_P(\beta)$ w.r.t. β.
end for
Return$((\hat{Q}_P - B)/A$ for all $P)$
end procedure

B.6 Algorithm: EP-FQCh type A recursion

Algorithm B.6 EP-FQCh type A recursion

Input: A vector C_{in} of length $J0$.
Output: A vector C_{out} of length $J0$.

procedure EP-FQCH.TYPE-A(Numeric vector C_{in})
Introduce numeric vector $Even$,Odd,C_{out}.
$Even = C(1, 2, \cdots, J0 - 1)$; $Odd = C(1, 2, \cdots, J0)$.
for $j = 1, 2, \cdots, J0$ **do**
$\bar{C}(j) = Odd(1)$.
$Even = 2 * Odd(2, 3, \cdots, J0) - Even(1, 2, \cdots, J0 - 1)$.
$Odd = 2 * Even(1, 2, \cdots, J0 - 1) - Odd(1, 2, \cdots, J0 - 1)$.
end for
Return(vector C_{out})
end procedure

B.7 Algorithm: EP-FQCh type B recursion

Algorithm B.7 EP-FQCh type B recursion

Input: A vector C_{in} of length $J0$.
Output: A vector C_{out} of length $J0$.

procedure EP-FQCh.type-B(Numeric vector C_{in})
  Introduce numeric vector Old,New,$Temp$,C_{out}.
  Set $Old = C_{in}[1, 2, \cdots, J0]; New = C_{in}[2, 3, \cdots, J0]; C_{out}[1] = C_{in}[1]; C_{out}[2] = C_{in}[2]$.
  for $j = 3, 4, \cdots, J0$ **do**
    $Temp = New$
    $New = 2 * New[2, 3, \cdots, J0] - Old[1, 2, \cdots, J0 - 1]$.
    $Old = Temp$.
    $C_{out}[j] = New[1]$.
  end for
  Return(vector C_{out})
end procedure

B.8 Algorithm: EP-FQCh type C recursion

Algorithm B.8 EP-FQCh type C recursion

Input: A matrix A_{in} of dimension J and K.
Output: A vector C_{out} of length JK.

procedure EP-FQCh.type-C(Numeric Matrix A_{in})
  Introduce numeric vector C_{out},Add.
  Set $C_{out} = A_{in}[., 1]$;
  Set $Add = A_{in}[., 1]$;
  for $k = 2, 3, \cdots, K$ **do**
    $Add = 2 * A_{in}[., k] - reverse(Add)$.
    $C_{out} = append(C_{out}, Add)$.
  end for
  Return(vector C_{out})
end procedure

B.9 Algorithm: EP-FQCh type D recursion

Algorithm B.9 EP-FQCh type D recursion

Input: A matrix A_{in} of dimension J and K.
Output: A vector S_{out} of length JK.

procedure EP-FQCh.type-D(Numeric Matrix A_{in})
  Introduce numeric vector S_{out},Add.
  Set $S_{out} = A_{in}[., 1]$;
  Set $Add = A_{in}[., 1]$;
  for $k = 2, 3, \cdots, K$ **do**
    $Add = 2 * A_{in}[., k] + reverse(Add)$.
    $S_{out} = append(S_{out}, Add)$.
  end for
  Return(vector S_{out})
end procedure

B.10 Algorithm: Computing summands for EP-FKD0

Algorithm Algorithm to get all summands for EP-FKD0

Input: A p dimensional vector $\mathbf{x} = (x_1, x_2, \cdots, x_p)$ and an integer J.
Output: A $p+2$ dimensional array $C(\mathbf{x}) = \{\{C_{t,k,\boldsymbol{j}}(\mathbf{x})\}\}$, dimensions of C are $\{p, 2J+1, \cdots, 2J+1(p+1 \text{ times})\}$.
First compute the $p \times (2J+1)$ matrix $L = \{\{l(t,j) = c_j(x_t)\}\}$.
Set $C = L$.
for Each row $l(t,.)$ of L do **do**
Set $C = Outer\big(C, l(t,.)\big)$.($Outer(.,.)$ is outer product of two arrays)
end for
Return(C)

B.11 Algorithm: EP-FKD0

Algorithm Map-Reduce algorithm: EP-FKD0

Input: A distributed Data-Frame(like a DDF object in R package "Datadr").
Output: A list of neighborhoods.
procedure EP-FKD0(conn = HDFS Connection to the DDF, J = Number of terms in Fourier Series, D = depth of KDTree)
Get p = Number of Variables
Scale all the variables in the range (-1,1).(Have to do a pre-Map-Reduce to get the range of the data if range is not given)
Map()
Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
initialize $p+2$ dimensional array $V = 0$, dimensions of V are $\{p, 2J+1, \cdots, 2J+1(p+1 \text{ times})\}$.
for Each Map-key in the Mapper **do**
for Each row $\mathbf{x}$ of the data-frame Map-Value of that Map-key **do**
Calculate the matrix M, such that, $M_{t,k,\boldsymbol{j}} = \{\{C_{t,k,\boldsymbol{j}}(\mathbf{x})\}\}$ using algorithm $B.10$.
Set $V = V + M$
end for
end for
for Each $p+2$ tuple $\{t, k, \boldsymbol{j}\}$ **do**
Collect$\Big($key $= \{t, k, \boldsymbol{j}\}$, value $= V\big(t, k, \boldsymbol{j}\big)\Big)$
end for
Collect(key = 0, value = Number of rows of the data-frame Map-Value)
end Map()
Reduce()
Input : key = Reduce-key(either 0 or a $p+2$ dimensional index), values = Reduce-Values(real numbers)
Initialize real number $s = 0$.
for Each Reduce-Values v **do**
Set $s = s + v$
end for
collect(key = Reduce-key(either 0 or a $p+1$ dimensional index), value = s)
end Reduce()
Execute a Map-Reduce job and read all the key-value pairs.
Initialize $p+2$ dimensional array $\bar{C} = 0$, dimensions of $\bar{C}$ are $\{p, 2J+1, \cdots, 2J+1(p+1 \text{ times})\}$.
Get the value N corresponding to the key 0.
for Each $p+2$ tuple $\{t, k, \boldsymbol{j}\}$ **do**
Get the value v corresponding to the key $\{t, k, \boldsymbol{j}\}$.
Set $\bar{C}[t, k, \boldsymbol{j}] = \frac{v}{N}$
end for
Start with the list of 1 neighborhood $L = \big(\mathbf{a} = (-1, -1, \cdots, -1), \mathbf{b} = (1, 1, \cdots, 1)\big)$
for $d = 1, 2, \cdots, D$ **do**
for each neighborhood $\big(\mathbf{a}, \mathbf{b}\big)$ of L **do**
Set $t = (d-1) \mod p+1$
Set $m_d^t = \underset{m \in (a_t, b_t)}{\arg\max}\ F(m)$ where $F(m) = \sum_{\{t,k,\boldsymbol{j}\}} f_k(m) G_{\boldsymbol{j}}(\mathbf{a}, \mathbf{b}) \bar{C}\big(t, k, \mathbf{j}\big)$
Set $\mathbf{a}_{left} = \mathbf{a}, \mathbf{b}_{left} = \mathbf{b}, \mathbf{a}_{right} = \mathbf{a}, \mathbf{b}_{right} = \mathbf{b}, \mathbf{a}_{right}(t) = m_d^t, \mathbf{b}_{left}(t) = m_d^t$
Replace $\big(\mathbf{a}, \mathbf{b}\big)$ with the collection $\Big(\big(\mathbf{a}_{left}, \mathbf{b}_{left}\big); \big(\mathbf{a}_{right}, \mathbf{b}_{right}\big)\Big)$ in L
end for
end for
Return(L)
end procedure

B.12 Algorithm: Computing summands for EP-FKD

Algorithm Algorithm to get all summands for EP-FKD

Input: A p dimensional vector $\mathbf{x} = (x_1, x_2, \cdots, x_p)$ and an integer J.
Output: (1) A p dimensional array $C(\mathbf{x}) = \{\{C_j(\mathbf{x})\}\}$, dimensions of C are $\{2J+1, \cdots, 2J+1(p \text{ times})\}$;
(2) p number of p dimensional arrays $\Xi^t(\mathbf{x}) = \{\{\Xi^t_j(\mathbf{x})\}\}$, dimensions of Ξ^t are $\{2J+1, \cdots, 2J+1, 4J-2(t^{th} \text{ co-ordinate}), 2J+1, \cdots, 2J+1\}$

Compute the $p \times 2J+1$ matrix $L = \{\{l(t,j) = c_j(x_t)\}\}$.
Compute the $p \times 4J-1$ matrix $M = \{\{m(t,j) = \xi_j(x_t)\}\}$.
Set $C = l(1,.)$(First row of L).
for $t = 2, \cdots, L$ do **do**
  Set $C = Outer\big(C, l(t,.)\big)$.($Outer(.,.)$ is outer product of two arrays)
end for
for Each row $m(t,.)$ of M do **do**
  Set $\Xi^t = Outer\Big(C\big((1:2J+1), \cdots, (1:2J+1), 0(t^{th} \text{ co-ordinate}), (1:2J+1), \cdots, (1:2J+1)\big), m(t,.)\Big)$.
end for
Return(C and Ξ^t for each t)

B.13 Algorithm: EP-FKD

Algorithm Map-Reduce algorithm: EP-FKD

Input: A distributed Data-Frame(like a DDF object in R package "Datadr").
Output: A list of neighborhoods.

procedure EP-FKD(conn = HDFS Connection to the DDF, J = Number of terms in Fourier Series, D = depth of KDTree)
Get p = Number of Variables
Scale all the variables in the range (-1,1).(Have to do a pre-Map-Reduce to get the range of the data if range is not given)
Map()
Input : keys = Map-keys(index), values = Map-Values(Data-Frame)
initialize p dimensional array $U = 0$, dimensions of U are $\{2J+1, \cdots, 2J+1(p \text{ times})\}$.
initialize p number of p dimensional array $V^t = 0$, dimensions of V^t are $\{2J+1, \cdots, 2J+1\big((p-1) \text{ times}\big), 4J-1\}$.
for Each Map-key in the Mapper **do**
for Each row $\mathbf{x}$ of the data-frame Map-Value of that Map-key **do**
Calculate the matrices $M, N^t, t = 1, \cdots, p$, such that, $M_{\boldsymbol{j}} = \{\{C_{\boldsymbol{j}}(\mathbf{x})\}\}$, $N^t_{\boldsymbol{j}} = \{\{\Xi^t_{\boldsymbol{j}}(\mathbf{x})\}\}$, using algorithm $B.12$.
Set $U = U + M$
for $t = 1, \cdots, p$ do **do**
Set $V^t = V^t + N^t$
end for
end for
end for
for Each p tuple $\boldsymbol{j} \in \mathbb{N}_J^{[p]}$ **do**
Collect$\Big(\text{key} = \boldsymbol{j}, \text{ value} = U(\boldsymbol{j})\Big)$
end for
for $t = 1, \cdots, p$ do **do**
for Each p tuple $\boldsymbol{j} \in \mathbb{N}_J^{[p,p]}$ **do**
Collect$\Big(\text{key} = \big(t, \boldsymbol{j}\big), \text{ value} = V^t(\boldsymbol{j})\Big)$
end for
end for
Collect(key = 0, value = Number of rows of the data-frame Map-Value)
end Map()
Reduce()
Input : key = Reduce-key(either 0 or an index vector), values = Reduce-Values(real numbers)
Initialize real number $s = 0$.
for Each Reduce-Values v **do**
Set $s = s + v$
end for
collect(key = Reduce-key(either 0 or an index vector), value = s)
end Reduce()
Execute a Map-Reduce job and read all the key-value pairs.
Get the value N corresponding to the key 0.
Initialize p dimensional array $\bar{C} = 0$, dimensions of $\bar{C}$ are $\{2J+1, \cdots, 2J+1(p \text{ times})\}$.
for Each p tuple $\boldsymbol{j} \in \mathbb{N}_J^{[p]}$ **do**
Get the value v corresponding to the key $\boldsymbol{j}$.
Set $\bar{C}\big(\boldsymbol{j}\big) = \frac{v}{N}$
end for
for $t = 1, \cdots, p$ **do**
Initialize p dimensional array $\bar{\Xi}^t = 0$, dimensions of $\bar{\Xi}^t$ are $\{2J+1, \cdots, 2J+1\big((p-1) \text{ times}\big)\}$.
for Each p tuple $\boldsymbol{j} \in \mathbb{N}_J^{[p,p]}$ **do**
Get the value v corresponding to the key $\boldsymbol{j}$.
Set $\bar{\Xi}\big(\boldsymbol{j}\big) = \frac{v}{N}$
end for
end for
Start with the list of 1 neighborhood $L = \big(\mathbf{a} = (-1, -1, \cdots, -1), \mathbf{b} = (1, 1, \cdots, 1)\big)$
for $d = 1, 2, \cdots, D$ **do**
for each neighborhood $\big(\mathbf{a}, \mathbf{b}\big)$ of L **do**
Set $t = (d-1) \mod p + 1$
Set $m_d^t = \underset{m \in (a_t, b_t)}{\arg\max}\ F(m)$ where
$F(m) = \sum_{\boldsymbol{j} \in \mathbb{N}_J^{[p]}} \omega_{j_t}(m, a_t, b_t)\mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a}, \mathbf{b})\bar{C}(\boldsymbol{j}) + \sum_{\boldsymbol{j} \in \mathbb{N}_J^{[p,p]}} \xi^J_{j_t}(m, a_t, b_t)\mathbf{g}_{\boldsymbol{j}_{-t}}(\mathbf{a}, \mathbf{b})\bar{\Xi}^t(\boldsymbol{j})$
Set $\mathbf{a}_{left} = \mathbf{a}, \mathbf{b}_{left} = \mathbf{b}, \mathbf{a}_{right} = \mathbf{a}, \mathbf{b}_{right} = \mathbf{b}, \mathbf{a}_{right}(t) = m_d^t, \mathbf{b}_{left}(t) = m_d^t$
Replace $\big(\mathbf{a}, \mathbf{b}\big)$ with the collection $\Big(\big(\mathbf{a}_{left}, \mathbf{b}_{left}\big); \big(\mathbf{a}_{right}, \mathbf{b}_{right}\big)\Big)$ in L
end for
end for
Return(L)
end procedure

C. R FUNCTIONS USED IN ILLUSTRATIONS:

```
map.bin <- function(x){
  x <- x[!is.na(x)]
  if(length(x) > 0) {
    cutTab <- as.data.frame(table(cut(x, cuts, labels = FALSE)), responseName = "Freq", stringsAsFactors = FALSE)
    cutTab$Var1 <- as.integer(cutTab$Var1)
    ls <- list()
    if(nrow(cutTab) == 0) {
      warning("data outside specified range")
    } else {
      for(i in 1:nrow(cutTab)) {
        ls[[i]]<-list(list(i, cutTab$Var1[i]), cutTab$Freq[i])
      }
    }
  }
  return(ls)
}
```

function *map.bin()*

```
map.EP.FQ0 <- function(x){
  x <- x[!is.na(x)]
  if(length(x) > 0) {
    x <- A*x+B
    c <- ftQ(x,J0)
    ls <- list()
    for(i in 1:2*J0) {
        ls[[i]]<-list(list(i), c[i])
    }
  }
  return(ls)
}
```

function *map.EP.FQ0()* The **R** function *map.EP.FQ0()* makes a call to the **C++** function *ftQ()*, see Appendix D

```
map.EP.FQCh1 <- function(x){
  x <- x[!is.na(x)]
  if(length(x) > 0) {
    x <- A*x+B
    c <- ftQ1(x,J0)
    ls <- list()
    for(i in 1:2*J0) {
        ls[[i]]<-list(list(i), c[i])
    }
  }
  return(ls)
}
```

function *map.EP.FQCh1()* The **R** function *map.EP.FQCh1()* makes a call to the **C++** function *ftQ1()*, see Appendix D

```
map.EP.FQCh2 <- function(x){
  x <- x[!is.na(x)]
  if(length(x) > 0) {
    x <- A*x+B
    c <- ftQ2(x,J,K)
    ls <- list()
    for(i in 1:2*J*K) {
        ls[[i]]<-list(list(i), c[i])
    }
  }
  return(ls)
}
```

function *map.EP.FQCh2()* The **R** function *map.EP.FQCh2()* makes a call to the **C++** function *ftQ2()*, see Appendix D

D. C++ FUNCTIONS USED IN R SCRIPTS:

```
NumericVector ftQ(NumericVector x, int J0) {

  int i,j,I;
  NumericVector C(2*J0);
  double xx, cc;

  for (i = 0; i < x.size(); i++){
    xx = x[i];
    cc = 2*xx;
    I = 0;
    for (j = 0; j < J0-1; j++){
      C[I] += cos(xx);
      I +=1;
      C[I] += sin(xx);
      I +=1;
      xx += cc;
    }
    C[I] += cos(xx);
    I +=1;
    C[I] += sin(xx);
  }
  return C;
}
```

function *ftQ()*

```
NumericVector ftQ1(NumericVector x, int J0) {

  int i,j,I;
  NumericVector C(2*J0);
  double c, s, c1, s1, t;

  for (i = 0; i < x.size(); i++){
    c1 = cos(x[i]);
    t = c1*c1;
    c = c1;
    s1 = sqrt(1-t)*((x[i] < 0) ? -1 : (x[i] > 0));
    s = s1;
    I = 0;
    for (j = 0; j < J0-1; j++){
      C[I] += c;
      c *= t;
      I +=1;
      C[I] += s;
      s *= t;
      I +=1;
    }
    C[I] += c;
    I +=1;
    C[I] += s;
  }
  return C;
}
```

function *ftQ1()*

```
NumericVector ftQ2(NumericVector x, int J, int K) {

  int i,j,k,I;
  NumericVector C(2*J*K);
  double c, s, c1, s1, c2, t, u;

  for (i = 0; i < x.size(); i++){
    c1 = cos(x[i]);
    t = c1*c1;
    c = c1;
    s1 = sqrt(1-t)*((x[i] < 0) ? -1 : (x[i] > 0));
    s = s1;
    c2 = cos(2*J*x[i]);
    u = 1;
    I = 0;
    for (k = 0; k < K-1; k++){
      for (j = 0; j < J; j++){
        C[I] += c;
        c *= t;
        I +=1;
        C[I] += s;
        s *= t;
        I +=1;
      }
      u *= c2;
      c = c1*u;
      s = s1*u;
    }
    for (j = 0; j < J-1; j++){
      C[I] += c;
      c *= t;
      I +=1;
      C[I] += s;
      s *= t;
      I +=1;
    }
    C[I] += c;
    I +=1;
    C[I] += s;
  }
  return C;
}
```

function *ftQ2()*