Free University of Bozen-Bolzano
Faculty of Computer Science

Data Indexing for the Interactive Visualization of Genome-Wide Association Studies

Master Thesis

Author:
Daniel Taliun

Faculty Supervisor:
Prof. Johann Gamper

Co-Supervisor:
Ph.D. Christian Fuchsberger
University of Michigan
Department of Biostatistics

2010

Abstract

Complex biological data sets are often explored in a visual way. In genetics, for example, Manhattan plots are widely used to summarize the results of a genome-wide association (GWA) study. In a typical GWA study millions of genetic variants are tested for the association with a disease. The visualization of these millions of points in an interactive manner is challenging. First, the massive data sets introduce the overplotting. Second, the extremely fast processing of user queries is required and must not take more than one second in general.

This thesis provides a solution for the time efficient interactive visual exploration of data arising from GWA studies. In particular, it faces the following problems: (i) complete or partial elimination of the overplotting; (ii) the fast and effective access to the data, which is necessary to produce the plot on the fly; (iii) the fast retrieval of additional and more detailed data on request; (iv) the fast retrieval of data when the plot projection is changed and the plot needs to be updated.

We propose a main memory based index structure, termed RDS tree, and provide algorithms for three types of queries: selection query, window query, top K query. Together they completely fulfill the mentioned requirements and therefore allow the interactivity. Empirical evaluation with synthetic data shows the high efficiency of RDS tree with large data sets containing tens of millions of data items.

Acknowledgements

I would like to express my deepest gratitude to supervisors Johann Gamper and Christian Fuchsberger for their invaluable assistance and extremely helpful guidance.

My special thanks is to Cristian Pattaro for his supervision of my work in Institute of Genetic Medicine. I would also like to express my thankfulness for all other members of Institute of Genetic Medicine for the provided opportunity to work in such interesting field and learn from their great experience.

Finally, my sincere appreciation is to Aistė Ivonytė for her great support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces [39]. This relatively new multidisciplinary field focuses on handling massive, heterogeneous, and dynamic volumes of information through integration of human judgment by means of visual representations and interaction techniques in the analysis process. In particular, the main idea of visual analytics is to visually represent information, allowing the human to directly interact with it, to gain insight, to draw conclusions, and to ultimately make better decisions [19, 1, 31]. The focus of the thesis is the interactive visualization of genome-wide association studies. It tackles the problem of the fast data retrieval, which is mandatory requirement of all interactive queries in the visualization.

In this chapter we provide the necessary genetic background and describe the basic idea of genome-wide association studies. The most common data visualizations in the field and their applications are considered. Afterward, we introduce the interactive tasks, that must be supported to fulfill the basic interactivity in visualization of genome-wide association studies. We identify the problems in supporting the interactive tasks and present our contribution.

1.1 Genome-Wide Association Studies

1.1.1 Genetic Background

All the genetic information is carried by deoxyribonucleic acid (DNA) [2, 4, 12, 25]. The DNA is made up of two long helical strands that together form a spiral staircase known as a double helix (see Figure 1.1a) [17]. Each strand consists of a linear sequence of nucleotide bases: adenine (A), thymine (T), guanine (G), and cytosine (C). The two strands are held together in the double helix by weak bonds between their opposite bases. An adenine on one strand always binds to a thymine on the opposite strand, while a guanine always binds to a cytosine. Therefore, the DNA strands are complementary and oriented in opposite directions.

The DNA molecules are organized into chromosomes, where one chromosome corresponds to a single extremely long molecule (see Figure 1.1b). Every chromosome contains many regions, portions of DNA, that are called genes. A single

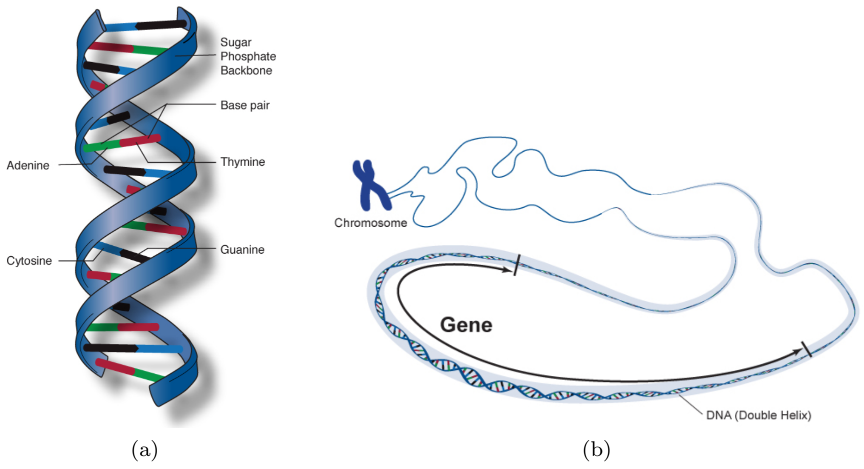


Figure 1.1: The Double Helix.

gene typically encodes the information for one protein. Every normal human cell, with the exception of eggs and sperm, has two versions of each chromosome and, therefore, gene. Hence, it contains 23 pairs of chromosomes where one pair is the sex chromosomes that determine sex. The entire set of 46 chromosomes is called genome.

The human genome contains 3.3 billion base pairs and 99% of it is identical for any of two unrelated individuals. However, there are still millions of potential variations of the DNA sequence. The differences in a gene structure that occur in more than 1% of individuals in a population are called polymorphisms. Several different forms of polymorphisms exist and the most common of them is the single-nucleotide polymorphism (SNP). As the name suggests, this polymorphism corresponds to a change of a nucleotide in a single base pair. It is expected, that the whole human genome contains up to 15 millions of SNPs [40]. Every different form that a polymorphism may obtain is called an allele. For example, a particular SNP can have from two to four alleles, although the major part of known SNPs has only two alleles. The different alleles of non-synonymous SNPs, that are located in genes, may lead to the production of different forms of proteins. However, even if a SNP is not located in a gene, then its alleles may still affect a cell function. For example, they may alter the amount of a protein that a cell builds. Thus, potentially, all types of SNPs can cause a disease or can influence some specific characteristic – a trait.

The focus of a genetic association study is to find a proof that a specific variation in a gene, measured across a study population, is causally associated with an observed disease or a trait [2, 4, 30, 6, 40]. The basic idea is to compare the allele frequencies in individuals with a trait of interest to a group of individuals without the trait [30, 15]. The association study, which is known as a candidate gene association, targets only a small set of specific genes. These genes are defined based on already known biology, stated hypotheses or previous results. Another approach is to screen the entire genome and to investigate hundreds of thousands of SNPs for an association with a trait without considering any of previous hypotheses. This type of study is called a genome-wide association study. The typical stages of the GWA study are: (i) selection of thousands of case subjects and control subjects; (ii) genotyping of SNPs across the entire genome; (iii) applying a quality control and cleaning of the gathered SNPs data; (iv) all the SNPs that pass the quality control and cleaning are tested for an association with a trait; (v) a subset of SNPs is selected, based on a statistical significance alone or a combination of statistical significance and biologic plausibility, for replication in an independent sample set.

1.1.2 Data in GWA Studies

A data in GWA study consists of a set of SNPs together with their attributes. The number of SNPs depends on the genotyping platform that was used in the study and varies from 100000 up to more than 1 million [7, 30, 28]. Since individual GWA studies have only enough power to detect associated SNPs with large effects, often the results from multiple studies are combined for a meta-analysis, increasing the sample size and power [7, 26]. However, the genotyping platform may differ from study to study and, therefore, the set of SNPs may differ too. Thus, to make the set of SNPs uniform across all the combined studies, the imputation is applied and the number of SNPs is increased up to 7 millions. Genotype imputation is a technique to predict unmeasured SNPs based on an external reference panel [23].

The set of SNP attributes may differ between studies. For the ease of exchanging GWA results the minimum set of attributes, that is sufficient to perform a meta-analysis, was proposed [7]: name, chromosome, position, coded allele, noncoded allele, frequency of the coded allele, strand orientation of the alleles, beta coefficient, standard error, p-value, call rate, imputation quality. Table 1.1 shows an example of a typical GWA study data.

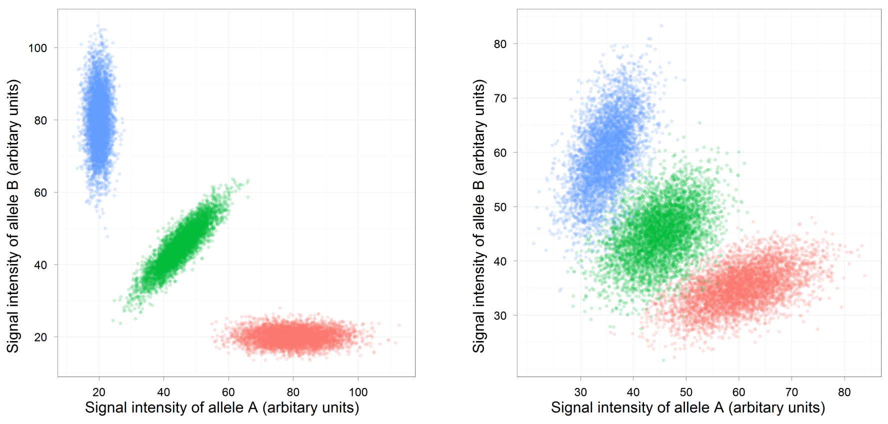
name	chromosome	coded allele	noncoded allele	allele frequency	beta	standard error	imputation quality	strand orientation	position	P-value
rs10	7	C	A	0.947	0.462	2.431	0.4042	+	92221824	0.8492
rs1000000	12	A	G	0.22	-0.244	0.897	0.9454	+	125456933	0.7856
rs10000010	4	T	C	0.485	-0.649	0.720	0.8413	+	21227772	0.368
rs10000012	4	C	G	0.865	0.431	1.101	0.9753	+	1347325	0.6957
rs10000013	4	A	C	0.746	-1.233	0.824	0.9825	+	36901464	0.1344
rs10000017	4	C	T	0.772	0.045	0.890	0.9268	+	84997149	0.9595
rs1000002	3	T	C	0.458	-0.080	0.753	0.9919	+	185118462	0.9154
rs10000023	4	G	T	0.4	-0.625	0.735	0.6723	+	95952929	0.3946

Table 1.1: The GWA Study Data.

1.1.3 Data Visualization in GWA Studies

Several visualization techniques for GWA study data exists. These visualizations help in validating and cleansing the data, localizing the most significant regions for further and more deeper consideration, presentation and sharing of the results. Sometimes, only a quick view on the visualization may determine the next crucial investigational steps in a study. Below we provide a short description of the four data visualization techniques, that are used in the previously identified stages of GWA study (see Section 1.1.1).

Genotype (or signal intensity) cluster plot visualizes the assignment of every SNP to one of the three possible genotypes. [30, 26, 45, 34]. Each of the two axes represents the fluorescence signal intensity of one of the two possible SNP alleles, that is obtained from a genotyping platform. Based on the allele signal intensity a calling algorithm assigns the SNP to one of the genotypes. Therefore, the SNPs of the same genotype are represented with the points of the same color. Figure 1.2 shows the genotype cluster plots of hypothetical genotyping data. The cluster of blue points corresponds to genotype B/B, the cluster of green points corresponds to genotype A/B, and the red points correspond to genotype A/A. On Figure 1.2a we observe three well defined clusters of genotypes, which suggest the correct genotype calling. However, an inaccurate genotype calling may produce the overlapping clusters as on Figure 1.2b. The SNPs with ambiguous genotype assignment are usually excluded from studies, since may lead to false-negative associations.

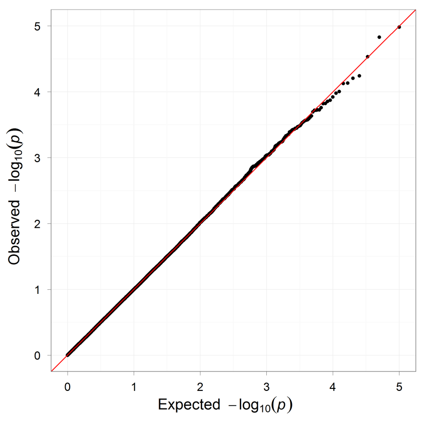


(a) (b)

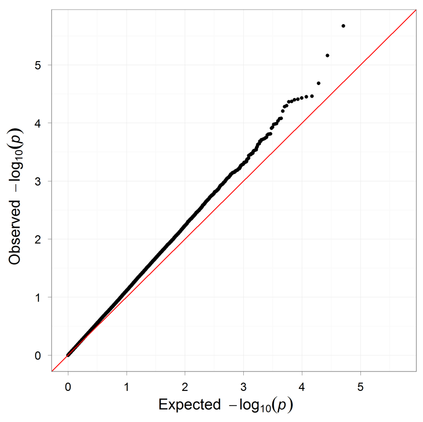
Figure 1.2: The Genotype Cluster Plots.

A quantile-quantile (Q-Q) plot compares the distribution of observed associations between SNPs and trait in a GWA study to the expected distribution if there is no association [26, 30]. The plot is constructed as follows: (i) the negative base 10 logarithmic transformation ($-log_{10}$) is applied for P values of every SNP; (ii) the transformed values are ranked in order from smallest to largest on the y-axis and plotted against the expected values on the x-axis. Figure 1.3 shows several Q-Q plots with a hypothetical data from a GWA study. Each plot displays two lines: the red identity line indicates the expected associations, while the black line shows the observed associations. The almost complete correspondence of the two lines on Figure 1.3a suggests little evidence of associations. However, the strong deviation of the lines on Figure 1.3b indicates a potential population stratification or cryptic relatedness, that may lead to misleading results [3]. The Q-Q plot on Figure 1.3c shows little evidence of the population stratification and suggests a number of large effect associations, which creates a deviation between the top ends of the two lines.

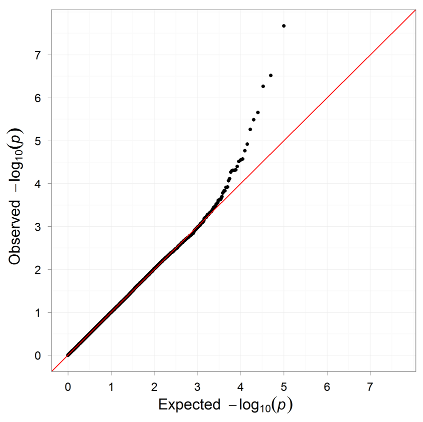
The chromosomal region plot displays the observed associations in a specific



(a)



(b)



(c)

Figure 1.3: The Quantile-Quantile Plots.

portion of a chromosome, that typically has several hundreds of thousands base pairs width [30]. The P value of every SNP after the negative base 10 logarithmic transformation is plotted on the y-axis against the physical position of the SNP on the chromosome represented with x-axis. Therefore, the strongest associations with the lowest P values have bigger y-axis coordinates. Also, the chromosomal region plot may be combined with the visual representation of linkage disequilibrium (LD) information for every pair of SNPs in the region. Thus, a pair of SNPs is associated with a colored rhomb, where the more intensive color infers higher LD. This information allows an estimation of independence of the observed associations and is crucial in establishing the real causal association with a trait, since it is possible that some other SNP in LD with one under study is the true causal variable [2]. Figure 1.4 shows an example of the chromosomal region plot [29].

The Manhattan plot represents the observed associations in the whole genome [26, 30, 15]. As in the chromosomal region plot, the negative logarithmically transformed P values of SNPs are plotted against their physical genomic positions. However, this time we do not focus on a particular genomic region, but instead

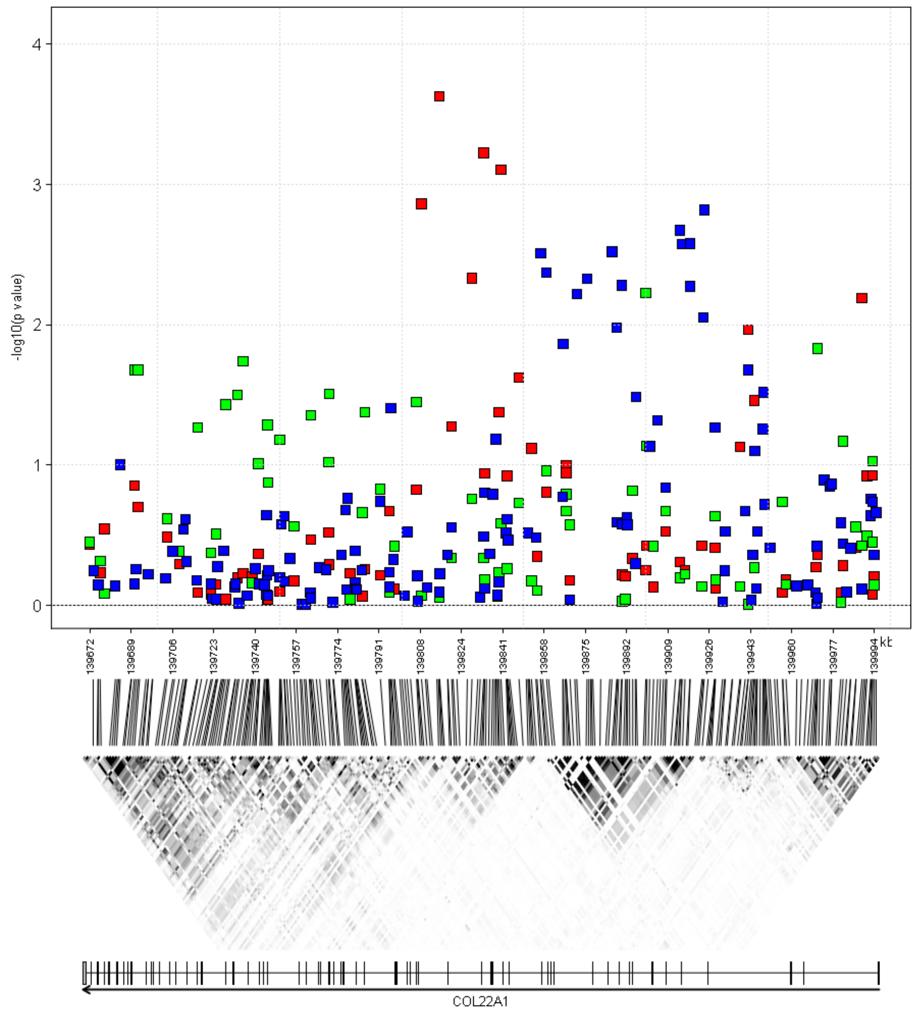


Figure 1.4: The Chromosomal Region Plot.

we visualize GWA data from the whole genome. Figure 1.5 shows the Manhattan plot of a hypothetical data. It highlights associations of a particular interest by displaying them on the top of the others. Also, the Manhattan plot enables to see which associations and in which regions satisfy a particular P value threshold. Only the SNPs satisfying the threshold are considered as genome-wide significant [3]. Since in a GWA study the association tests are repeated for each of more than several hundreds of thousands of SNPs, then it introduces the problem of multiple comparisons. Indeed, if we consider the common P value threshold of 0.05 and test only 1000 SNPs, then there is 99.99% of probability to find a false-positive association. Exist several methods facing the problem of multiple comparisons and one of them is Bonferroni method. According to this method, the P value is divided by the number of tested SNPs.

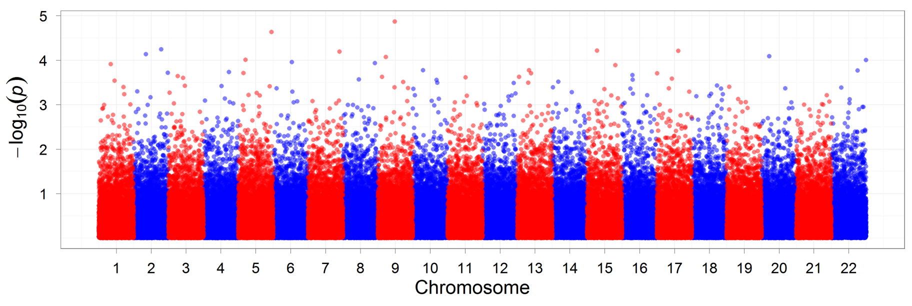


Figure 1.5: The Manhattan Plot.

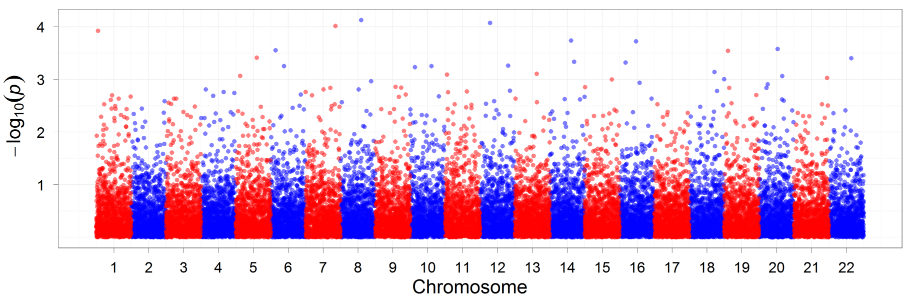
1.2 Problem Description

1.2.1 Interactive Visualization in GWA Studies

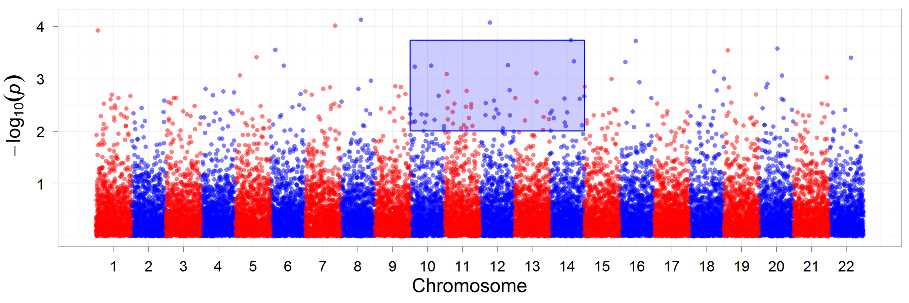
The previous section has introduced the four most widely known visualizations of data in GWA studies and described their applicability [26, 15]. It confirms the importance of the graphical representation of large data in the analysis, both in the context of GWA studies and in general. Although the classic appearance of the presented plots doesn't imply any dynamic feedback to an analyst, the support of interactivity may enable better and faster understanding of the provided information.

Shneiderman [37, 38] emphasizes seven general tasks of information visualization, that are followed by many applications: (i) provide an overview of the entire data; (ii) zoom in on data of interest; (iii) filter out uninteresting items; (iv) provide details on demand; (v) show relationships among data items; (vi) remember user actions to support undo, replay, and progressive refinement; (vii) allow extraction of data subsets according to the query parameters. Certainly, more tasks are being introduced and the existing are being refined, when implementing visualization for a specific domain. This work considers the support of basic interactivity in the visualizations of GWA studies. In particular, as a running example we use the Manhattan plot. It doesn't prioritize this visualization within the three others, since all of them serve for different purposes and are applied in different stages of analysis. Moreover, the discussed problems and proposed solutions are relevant to all of these visualizations.

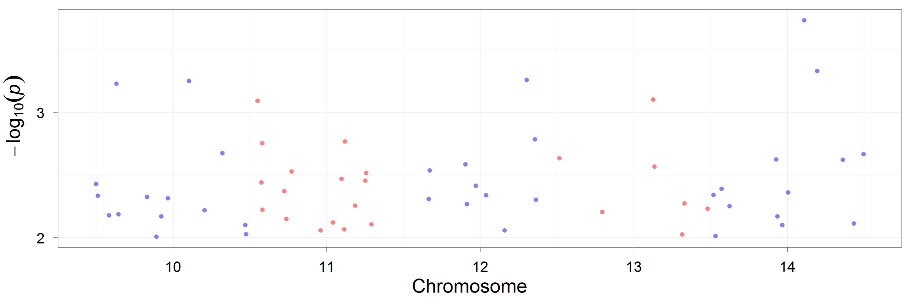
There is a number of interaction techniques, which solve the first two crucial visualization tasks of providing an overview of all data with the capability to zoom in on data subsets of interest. These techniques are oriented towards the usually large amounts of data, that can't be displayed on the screen completely in a straightforward way. One example is the classic distortion-oriented FishEye technique from the Focus+Context visualizations [8, 21]. It geometrically distorts the view under the area of focus so that more space is available and more detailed data is represented. The main benefit of this and other similar distortion-oriented solutions is that the zoomed in area is seen in the context of the whole data. Another well known method of preserving the overall context while drilling down into the more details, is to represent two different views in the same or in two separate windows – one for the overview of all data and another for the zoomed in area of interest. For the simplicity of further discussion and implementation, it is assumed simpler approach of providing these capabilities for the Manhattan plot. Initially, the classic view of the Manhattan plot is provided – this is the detailed overview of full data. Then, using a rectangular lasso selection tool, one is able to differentiate an area of visualization and apply zoom in operation on the selected area (see Figure 1.6) [41]. The drawback of this simplified technique is that when drilling down into the details the context is not preserved. However, it can be overcome by providing additional view of overall data.



(a) The Overview



(b) The Rectangular Lasso Selection Tool

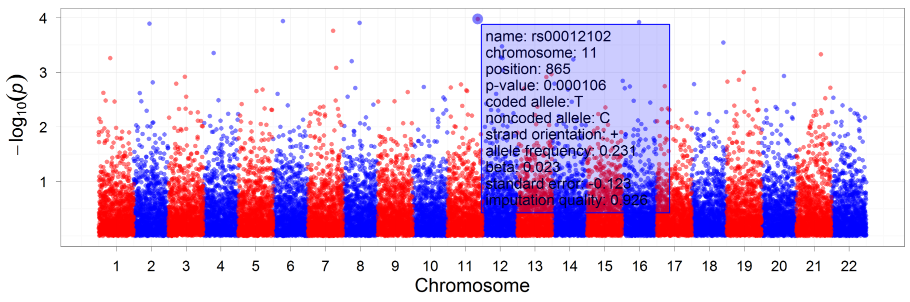


(c) The Zoomed In Area

Figure 1.6: The Interactive Zoom In Task in Manhattan Plot.

The filtering task implies the possibility to control the contents of the visualization by eliminating irrelevant data items. This operation is common and sometimes is even necessary in the context of Manhattan plot. An analyst may be interested in eliminating a SNP from visualization and from further analysis if this SNP doesn't satisfy one or more specified thresholds. One example is the exclusion of low quality SNPs, that exceed thresholds for imputation quality, allele frequency or lack some other essential attributes.

The details-on-demand task involves the retrieval of detailed information about the selected data item or a subset of data items. User selects a single data item by clicking on the representative point on display, or selects a subset of data items using lasso or brush selection tool [41]. Then, the required details about the selected data items are retrieved and displayed. The usual way of representing the retrieved details is using a pop-up window [37]. The example of details-on-demand task in Manhattan plot is the retrieval of all attributes for the selected SNP (see Figure 1.7). The p-value, exact genomic position and name is essential information about SNP, however it can't be displayed for every SNP in Manhattan plot simultaneously due to the large amount of data. Therefore, the support for details-on-demand task is crucial at least regarding these three attributes.



name: rs00012102
chromosome: 11
position: 865
p-value: 0.000106
coded allele: T
noncoded allele: C
strand orientation: +
allele frequency: 0.231
beta: 0.023
standard error: -0.123
imputation quality: 0.926

Figure 1.7: The Details-On-Demand Task in Manhattan Plot.

The most trivial relation between SNPs that may be a question of visualization in Manhattan plot is their association with the same gene. This relationship may be displayed by highlighting a subset of associated SNPs from the rest with different color. Figure 1.8 shows an example of visualized relation between SNPs in Manhattan plot, where SNPs highlighted with green color belong to the same gene. Linkage disequilibrium between SNPs is the another relation that may be visually expressed in the same way in Manhattan plot.

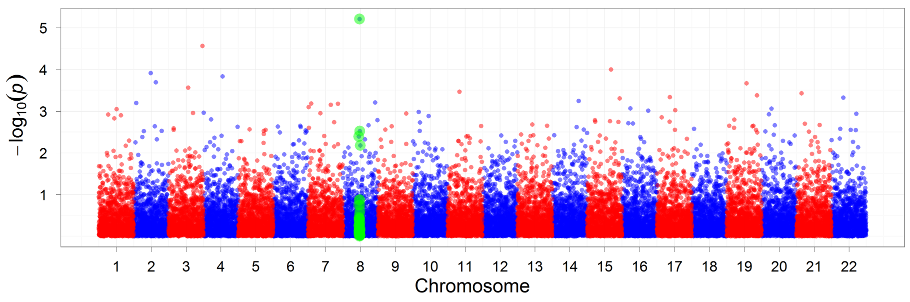


Figure 1.8: The Relation Between SNPs in Manhattan Plot.

The last two visualization task of keeping the history of user actions and allowing the extraction of data subsets into the external files are important, however aren't in the focus of this work. This is mainly a matter of a concrete implementation of software providing the interactive visualization.

1.2.2 Challenges

In the perfect scenario, all the interactive visualization tasks should produce and display the desired query results instantaneously. Obviously, it is not achievable in the reality due to technical limits. Although, the human perception abilities allow delays in the displaying of the results. It is assumed, that the 10 Hz display refresh rate is sufficient for instantaneous graphical changes [8, 10, 38]. Therefore, for any instantaneous interactive query, the results should be produced and displayed in no more than 100 msec. For a graphical transition, in the case of zoom in task, the operation should be completed within 1 sec. Certainly, an analyst gains actual benefit from the interactive tasks only if they are implemented in efficient way and satisfy the mentioned execution time requirements. Large datasets containing millions of items makes the fulfillment of these requirements a not trivial task.

The large datasets often lead to high degree of overplotting and may occlude a significant portion of the shown data items. The primary factor in this problem is the limited amount of pixels of a display, which makes it impossible to fit tens of millions data items without collisions [8, 10, 38, 19]. Indeed, even in the extreme case, when every single data item is mapped to one pixel, then it is possible to represent only around 2 million items on 1600 $\times$ 1200 display. One route to solve the problem is to increase screen resolution by introducing very large displays [43]. However, the size of display is limited by human perception abilities, which vary around 10 million pixels and depend on the distance from display. Another approach is to apply filtering, aggregation and compression techniques to reduce the amount of data without losing significant information. There are several recent works towards this direction, which focus on scatterplots, parallel coordinates and visualization of large time-series data [13, 14, 18, 42].

Summarizing everything, it is obvious that the introduction of interactivity to the visualization of GWA studies with up to 15 million visualized SNPs for a single study brings out the following problems:

1. The overview task requires a fast retrieval of coordinates for all SNPs in the database.
2. The details-on-demand task requires a fast identification of subset of SNPs in the database and then needs a fast retrieval of all associated attributes.
3. The data overview and further visualization of large genomic regions suffer from a high degree of overplotting and, therefore, the majority of the visualized SNPs is occluded.

1.2.3 Contributions

In this thesis we provide a solution for the time efficient interactive visual exploration of data arising from GWA studies. More specifically, the technical contributions are:

- We propose a data reduction based on the defined screen resolution such, that the visualization of the reduced data doesn't suffer from overplotting and is identical to the visualization of the full data.
- We present a novel index structure, termed RDS tree, which indexes data with respect to the hierarchy of reduced data constructed for the predefined

list of screen resolutions. The RDS tree is a space partitioning tree, which is constructed only once and then can be utilized during the visualizations on displays with different resolutions.

- We introduce and provide algorithms for three queries over the RDS tree, which support the basic interactive tasks: (i) selection query – ensures the fast data retrieval for the overview task and reduces the overplotting; (ii) window query – ensures the fast data retrieval for the visualization of the data in the particular region and reduces the overplotting; (iii) top K query – implements the details-on-demand task, that efficiently retrieves top K data items occluded in the visualization.

- The evaluation with synthetic GWA study data confirms the high efficiency of the RDS tree with tens of millions of SNPs, which ensures short response times.

Throughout the discussion we use the example of RDS tree for the interactive Manhattan plot. Nevertheless, our solution can be applied for other visualizations in GWA studies like quantile-quantile plots or genotype cluster plots.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2 we formally define the display model and the data visualization in it. Chapter 3 presents the data reduction with respect to the display resolution and introduces the RDS tree. The selection query, the window query and the top K query over the RDS tree are defined and explained in Chapter 4. Chapter 5 describes the evaluation of RDS tree. In Chapter 6 we provide the overview of the related work in the field. Chapter 7 presents the conclusions and the future work.

Chapter 2

Display Model

In this chapter we introduce a display model and describe the data visualization in it. The further discussion completely relies on this model and its assumptions.

2.1 Model Description

The display model consists of a two dimensional grid of regularly spaced elements – pixels. A pixel is the smallest addressable element and is referenced with a coordinate $(i,\, j)$, where $i \in \mathbb{N}^*$ and $j \in \mathbb{N}^*$ are the horizontal and the vertical positions respectively. Additionally, it is assumed, that all pixels are squares of the same size. A display resolution is defined as the maximum number of pixels, which the display is capable to represent. It is denoted as $N \times M$, where $N \in \mathbb{N}^*$ and $M \in \mathbb{N}^*$ are accordingly the number of pixels horizontally and vertically. For instance, if the resolution is 1600×1200, then the display handles about 2 million pixels aligned in 1600 columns and 1200 rows. Continuing with the example, the coordinate of the bottom left pixel is $(1, 1)$, while the coordinate of the top right pixel is $(1600, 1200)$. Figure 2.1 depicts the display model of resolution $N \times M$, and Definition 1 formally defines a set Δ of all available display pixels.

Definition 1 (Set of Display Pixels). *Let $N \times M$ be the display resolution. Then, the finite set of all available display pixels is defined as $\Delta = \{\, (i, j) \mid 1 \leq i \leq N \wedge 1 \leq j \leq M \,\}$.* ♣

2.2 Data Visualization

When data is visualized in $\mathbb{R}^2$ Euclidean space, then every data item is assigned to at least one display pixel. The assignment is done in two steps:

1. Association
2. Mapping

Next sections provide the detailed description of these steps.

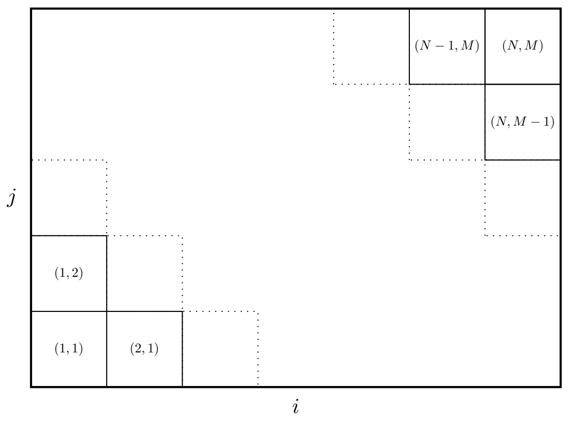


Figure 2.1: The Display Model of Resolution $N \times M$.

2.2.1 Association

In this step a data item is associated with a single point in $\mathbb{R}^2$ Euclidean space. We refer to this point as the *associated point*. Consider a finite data set D and a single data item $d \in D$. Every data item has the same finite set of attributes $A = \{ A_1, A_2, \dots, A_n \}$, where $n \in \mathbb{N}^*$. A particular attribute $A_i \, (1 \leq i \leq n)$ of data item d is denoted as $d.A_i$. Then, Definition 2 defines the association of data item d with a point in $\mathbb{R}^2$ Euclidean space, and Definition 3 defines a set of all associated points in $\mathbb{R}^2$ Euclidean space.

Definition 2 (Association). *Let $f : D \to \mathbb{R}$ and $g : D \to \mathbb{R}$ be two functions relating data item $d \in D$ with a real number according to its attributes in set A. Then, the data item $d \in D$ is associated with point $\big(f(d), \, g(d)\big)$ in $\mathbb{R}^2$ Euclidean space, which is called the associated point.* ♣

Definition 3 (Set of Associated Points). *Let $f : D \to \mathbb{R}$ and $g : D \to \mathbb{R}$ be two functions relating data item $d \in D$ with a real number according to its attributes in set A, and point $\big(f(d), \, g(d)\big)$ be the associated point in $\mathbb{R}^2$ Euclidean space. Then, the set of all points in $\mathbb{R}^2$ Euclidean space associated with data set D is defined as $P = \{ (x, y) \mid (x, y) \in \mathbb{R}^2 \land d \in D \land (x, y) = \big(f(d), \, g(d)\big) \}$.* ♣

Example 1 To illustrate the association, assume the data set D in Table 2.1. It contains ten data items, where each data item represents a SNP with the attributes *name*, *chr*, *pos* and *pval*. Attribute *name* is the unique identifier of SNP, while *chr* and *pos* correspond to the chromosome and the chromosomal position of SNP respectively. Attribute *pval* denotes a hypothetical P value of SNP in a GWA study. Then, to produce the Manhattan plot, the functions f and g are defined as follows:

$$f(d) = \sum_{i=1}^{d.chr-1} \Big(e(i) - s(i) + 1 \Big) + \big(d.pos - s(d.chr) \big) \tag{2.1}$$

and

$$g(d) = -\log_{10} d.pval \,, \tag{2.2}$$

where $s(z) = \min\{\, d.pos \mid d \in D \wedge d.chr = z \,\}$ and $e(z) = \max\{\, d.pos \mid d \in D \wedge d.chr = z \,\}$. Functions $s(z)$ and $e(z)$ extract the smallest and the largest chromosomal positions within all SNPs from the same chromosome z, respectively. For example, $s(2)$ extracts the smallest position within 60718, 62106 and 63495 positions of all SNPs from the chromosome 2, which is equal to 60718. Function f transforms *chr* and *pos* attributes of SNP to the x coordinate of the associated point, while function g transforms *pval* attribute of SNP to the y coordinate of the associated point. The functions are applied to every of ten SNPs in data set D. As a result, Table 2.1 lists set P of all associated points in $\mathbb{R}^2$ Euclidean space. Figure 2.2 depicts the points in Cartesian coordinate system and, therefore, depicts the Manhattan plot too. The shape of a point corresponds to the chromosome of SNP, which was associated with this point. In particular, the rectangles refer to SNPs from the 1st chromosome, the diamonds refer to SNPs from the 2nd chromosome, and the circles denote SNPs from the 3rd chromosome. $\square$

D

	name	chr	pos	pval
d_1	rs987669	1	60160	0.5000
d_2	rs10399597	1	62157	0.1500
d_3	rs55858216	2	60718	0.1100
d_4	rs73140430	2	62106	0.0050
d_5	rs730038	2	63495	0.5500
d_6	rs28729284	3	60202	0.0100
d_7	rs13067307	3	60596	0.0110
d_8	rs9755941	3	61044	0.0018
d_9	rs9756992	3	61113	0.0015
d_{10}	rs13081384	3	61466	0.0010

$\xrightarrow{\text{Step 1}}$

P

	x	y
d_1	0	0.301
d_2	1997	0.824
d_3	1998	0.959
d_4	3386	2.301
d_5	4775	0.260
d_6	4776	2.000
d_7	5170	1.959
d_8	5618	2.745
d_9	5687	2.824
d_{10}	6040	3.000

$\xrightarrow{\text{Step 2}}$

S

	i	j
d_1	1	1
d_2	3	2
d_3	3	2
d_4	5	5
d_5	7	1
d_6	7	4
d_7	7	4
d_8	8	6
d_9	8	6
d_{10}	8	6

Table 2.1: SNPs Assignment to Pixels on a Display of Resolution 8×6.

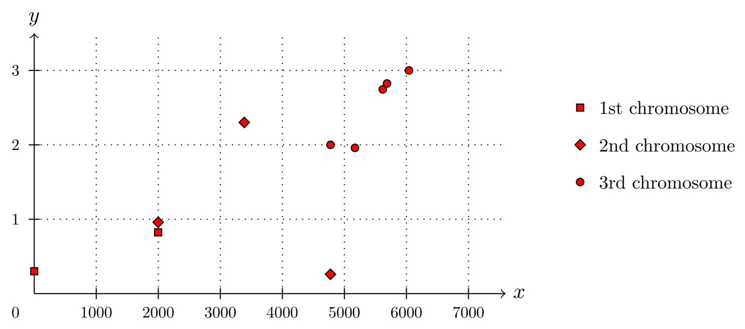


Figure 2.2: Points Associated with SNPs in Cartesian Coordinate System.

2.2.2 Mapping

After the 1st step, when a data item was associated with a point in $\mathbb{R}^2$ Euclidean space, the obtained point is mapped to a single display pixel. We refer to this pixel as the *occupied pixel.* In this step we don't consider any geometrical transformations, but only the orthogonal projection to the two dimensional display surface. The mapping concludes the transitive assignment of data item to pixel and, therefore, the visualization in overall. Definition 4 formally defines the mapping function.

Definition 4 (Mapping Function). *Let Ω be the bounded subset of $\mathbb{R}^2$ Euclidean space such that $|\Omega| > 1$, and Δ be the set of all available display pixels. Let $\Omega_x = \{\, x \mid (x,y) \in \Omega \,\}$ and $\Omega_y = \{\, y \mid (x,y) \in \Omega \,\}$ denote two sets corresponding to all distinct x and all distinct y coordinates in set Ω, respectively. Let $\Delta_i = \{\, i \mid (i,j) \in \Delta \,\}$ and $\Delta_j = \{\, j \mid (i,j) \in \Delta \,\}$ denote two sets corresponding to all distinct i and all distinct j coordinates in set Δ, respectively. Then, the mapping function $\mu : \Omega \to \Delta$ maps every point in Ω to exactly one pixel in Δ, which is called the occupied pixel:*

$$\mu\big((x,y)\big) = (i,j), \tag{2.3}$$

where

$$i = \begin{cases} \left\lceil |\Delta_i| \times \left| \frac{x - \inf \Omega_x}{\sup \Omega_x - \inf \Omega_x} \right| \right\rceil & , \inf \Omega_x < x \leq \sup \Omega_x \\ 1 & , x = \inf \Omega_x \end{cases}$$

and

$$j = \begin{cases} \left\lceil |\Delta_j| \times \left| \frac{y - \inf \Omega_y}{\sup \Omega_y - \inf \Omega_y} \right| \right\rceil & , \inf \Omega_y < y \leq \sup \Omega_y \\ 1 & , y = \inf \Omega_y \end{cases}$$

♣

Taken together, the mapping function μ creates a tessellation of a bounded rectangular area on $\mathbb{R}^2$ Euclidean plane (i.e., splits it into disjoint subsets) with every tile (i.e., subset) assigned to a pixel. Thereby, all points belonging to the same tile are mapped to the same pixel. Consequently, all data items associated with points from the same tile are assigned to the same pixel. Definition 5 defines a set of all occupied display pixels.

Definition 5 (Set of Occupied Display Pixels). *Let P be the set of points in $\mathbb{R}^2$ Euclidean space associated with data set D, and Δ be the set of all available display pixels. Then, we define the set $S = \{\, (i,j) \mid (i,j) \in \Delta \wedge (x,y) \in P \wedge (i,j) = \mu\big((x,y)\big) \,\}$ to be the set of all occupied display pixels to which points in set P are mapped.* ♣

Example 2 For instance, consider SNPs in data set D and set P of all associated points in $\mathbb{R}^2$ Euclidean space listed in Table 2.1. The associated points in P constitute the bounded subset Ω of $\mathbb{R}^2$ Euclidean space. Therefore, $\inf \Omega_x = 0$ and $\sup \Omega_x = 6040$, while $\inf \Omega_y = 0.260$ and $\sup \Omega_y = 3.000$. If we assume, that the display resolution is 8×6, then $|\Delta_i| = 8$ and $|\Delta_j| = 6$. Consequently, the mapping function μ is the following:

$$\mu\big((x,y)\big) = (i,j), \tag{2.4}$$

where

$$i = \begin{cases} \left\lceil 8 \times \frac{x}{6040} \right\rceil & , 0 < x \leq 6040 \\ 1 & , x = 0 \end{cases}$$

and

$$j = \begin{cases} \left\lceil 6 \times \frac{y-0.260}{3.000-0.260} \right\rceil & , 0.260 < y \leq 3.000 \\ 1 & , y = 0.260 \end{cases}$$

When μ function is specified, then it is applied to every of ten points in set P of all points in $\mathbb{R}^2$ Euclidean space associated with SNPs. As a result, Table 2.1 lists set S of occupied display pixels, to which points in set P are mapped and, therefore, to which SNPs are assigned. Figure 2.3 depicts the mapping. On the left are represented points in $\mathbb{R}^2$ Euclidean space and the tessellation of space produced by the μ function. As before, the rectangles refer to SNPs from the 1st chromosome, the diamonds refer to SNPs from the 2nd chromosome, and the circles denote SNPs from the 3rd chromosome. On the right is represented the display of resolution 8×6 with highlighted occupied pixels. As reported in the example, the mapping of points in $\mathbb{R}^2$ Euclidean space and, therefore, the assignment of SNPs to pixels is not unique. For instance, pixel $(8, 6)$ represents three SNPs *rs9755941*, *rs9756992* and *rs13081384* from the 3rd chromosome at the same time. □

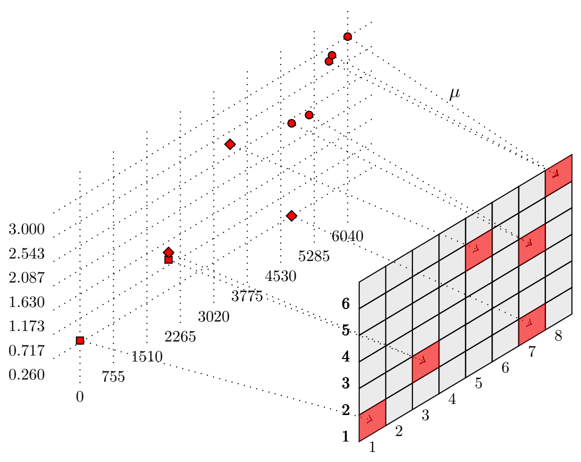


Figure 2.3: Mapping of Points in $\mathbb{R}^2$ Euclidian Space Associated with SNPs to Pixels on a Display of Resolution 8×6.

Chapter 3

The RDS Tree

Naturally, the visualization requires to access the whole data. If data is stored in a secondary storage and includes millions of items, then the retrieval and processing of it consumes significant amount of time. This is a major bottleneck in interactive visualization, where the response time to user actions is crucial. In this chapter we introduce an index structure for data visualization on multiresolution displays. It indexes data so, that only the data items, which contribute to the perception, are accessed and processed during the visualization.

3.1 Data Reduction

As discussed earlier, the μ function may define mapping which results in the assignment of more than one data item to the same display pixel. In this case we say that data items *overplot* one another. Intuitively, the overplotting is most probable when the display resolution is significantly smaller than the size of a data set, or the extent of data set in $\mathbb{R}^2$ Euclidean space is significantly wider than the extent of resolution. Obviously, the exclusion of all oveplotting items doesn't produce any difference in visualization, unless transparency or colour intensity is used. Therefore, it is not necessary required to read the whole data set to produce the visualization, instead it is possible to read only a part of the data set and have the same visual result. In this section we introduce a data set reduction with respect to $N \times M$ display resolution such, that there is no difference between the visual representation of the reduced data set and the full data set.

Definition 6 defines a selection function. This function selects a point in a subset of all points in $\mathbb{R}^2$ Euclidean space associated with data set such, that the selected point is most suitable to be visualized if all other points in the subset are excluded from the visualization. We refer to this point as the *representative point.* The selection of the representative point can directly depend on the attributes of the data item associated with this point, or on some other criteria. For instance, a point can be selected as representative in the subset of associated points if it is a medoid of this subset with respect to the x and y coordinates.

Definition 6 (Selection Function). *Let P be the set of points in $\mathbb{R}^2$ Euclidean space associated with data set D. Then, function $\sigma : \mathcal{P}(P) \setminus \emptyset \to P$ is called a selection function and selects a representative point from a subset of set P of all associated points.* ♣

Example 3 To illustrate the selection of the representative point, consider SNPs in data set D and set P of all associated points in $\mathbb{R}^2$ Euclidean space listed in Table 2.1. Assume a point is defined as representative of a set of points if it is associated with a SNP, which has the lowest P value between all SNPs associated with the other points in the set. Thereby, set P is a totally ordered finite set such that for any pair of points (x, y) and (x', y') in P, $(x, y) < (x', y')$ if and only if $y < y'$, or $y = y$ and $x < x'$. Thus, the selection function $\sigma : \mathcal{P}(P) \setminus \emptyset \to P$ is defined as the maximum function:

$$\sigma(P') = \max(P'), \quad \text{where } P' \subseteq P \tag{3.1}$$

For instance, consider set $P' = \{\,(1998, 0.959), (3386, 2.301), (4775, 0.260)\,\}$ containing points associated with SNPs $rs55858216$, $rs73140430$ and $rs730038$ respectively. Then $\sigma(P')$ selects the point (3386, 2.301) associated with SNP $rs73140430$, which has the lowest P value of 0.0050 comparing to the other two SNPs. Thus, the point (3386, 2.301) is the representative point of all points in set P'. □

Definition 7 defines a reduction function, which reduces set P of all points in $\mathbb{R}^2$ Euclidean space associated with data set with respect to the specified $N \times M$ display resolution and the mapping function μ.

Definition 7 (Reduction Function). *Let P be the set of all points in $\mathbb{R}^2$ Euclidean space associated with data set D, and $\sigma(P')$ be the selection function with P' being a subset of set P. Let Ω be the bounded subset of $\mathbb{R}^2$ Euclidean space such that $P \subseteq \Omega$, Δ be the set of all available display pixels, and $\mu : \Omega \to \Delta$ be the mapping function. Then, function $\rho : P \to P_\rho$ is a reduction function and set $P_\rho \subseteq P$ is a reduced set of associated points:*

$$\rho\big((x, y)\big) = \sigma\big(\{\,(x', y') \mid (x', y') \in P \wedge \mu\big((x', y')\big) = \mu\big((x, y)\big)\,\}\big) \qquad ♣$$

In particular, the mapping function creates a tessellation of set P of all points in $\mathbb{R}^2$ Euclidean space associated with data set. The number of created tiles equals to the size of set Δ of all available pixels on a display of resolution $N \times M$. Then, for every tile the reduction function selects the representative point of all points in the tile. The selection is done with the specified selection function σ. As a result, the reduced set of points in $\mathbb{R}^2$ Euclidean space associated with data set consists of the union of all selected representative points. Definition 8 defines a reduced data set consisting only of data items, which are associated with the reduced set of points in $\mathbb{R}^2$ Euclidean space.

Definition 8 (Reduced Data Set). *Let $f : D \to \mathbb{R}$ and $g : D \to \mathbb{R}$ be two functions relating data item $d \in D$ with a real number according to its attributes in set A, and point $\big(f(d),\, g(d)\big)$ be the associated point in $\mathbb{R}^2$ Euclidean space. Let P_ρ be the reduced set of points in $\mathbb{R}^2$ Euclidean space associated with data set D. Then, the reduced data set is defined as $D_\rho = \{\, d \mid d \in D \wedge \big(f(d),\, g(d)\big) \in P_\rho \,\}$.*♣

Example 4 For example, again consider SNPs in data set D and set P of associated points in $\mathbb{R}^2$ Euclidean space listed in Table 2.1. Equation 3.1 defines the selection function σ. As previously, suppose the display resolution is 8×6 and points in set P constitute the bounded subset Ω of $\mathbb{R}^2$ Euclidean space. Then, the mapping function μ is defined in Equation 2.4. Consequently, the reduction function ρ is specified as follows:

$$\rho\big((x,y)\big) = \max\big(\{\,(x',y') \mid (x',y') \in P \wedge \mu\big((x',y')\big) = \mu\big((x,y)\big)\,\}\big) \tag{3.3}$$

The mapping function μ creates the tessellation of set P consisting of 24 tiles, where only 6 of them actually include the points associated with SNPs. Table 3.1 lists the full set P, where rows corresponding to the points from the same tile have the same color. The reduction function ρ is applied to every point in set P. First, it picks all points which belong to the same tile as the input point. Afterwards, the selection function σ selects the representative point of all the picked points. Consider the input point (5687, 2.824), which belongs to the tile $\{\,(x,y) \mid 5285 < x \leq 6040 \wedge 2.543 < y \leq 3.000\,\}$. Then, $\rho\big((5687, 2.824)\big)$ picks three points $(5618, 2.745)$, $(5687, 2.824)$ and $(6040, 3.000)$ from the same tile including the input point. Since point $(6040, 3.000)$ is the maximum point between all three, then it is selected by the σ function as the representative point and, therefore, $\rho\big((5687, 2.824)\big) = (6040, 3.000)$. Table 3.1 lists the reduced set P_ρ of the associated points. After the reduction, points in the reduced set P_ρ are mapped to display pixels using the same initial μ function. Table 3.1 lists set S of the occupied display pixels and Figure 3.1 depicts the mapping together with the reduction of the associated points. As before, on the left are represented initial points in $\mathbb{R}^2$ Euclidean space and the tessellation of space produced by the μ function. The different shapes of points indicate the different chromosomes of the associated SNPs. In the middle are represented points from the reduced set and the same tessellation of space. On the right is depicted the display of resolution 8×6 with highlighted occupied pixels. The reduced data

P

	x	y
d_1	0	0.301
d_2	1997	0.824
d_3	1998	0.959
d_4	3386	2.301
d_5	4775	0.260
d_6	4776	2.000
d_7	5170	1.959
d_8	5618	2.745
d_9	5687	2.824
d_{10}	6040	3.000

$\xrightarrow{\rho}$

P_ρ

	x	y
d_1	0	0.301
d_3	1998	0.959
d_4	3386	2.301
d_5	4775	0.260
d_6	4776	2.000
d_{10}	6040	3.000

$\xrightarrow{\text{Step 2}}$

S

	i	j
d_1	1	1
d_3	3	2
d_4	5	5
d_5	7	1
d_6	7	4
d_{10}	8	6

Table 3.1: Reduction of Points in $\mathbb{R}^2$ Euclidean Space and Mapping to Pixels on a Display of Resolution 8×6.

set D_ρ is defined with respect to the reduced set P_ρ of points in $\mathbb{R}^2$ Euclidean space associated with data set D. Thus, a SNP is included in D_ρ if and only if there exists the corresponding associated point in the reduced set P_ρ. On the left Table 3.2 lists the full data set D of SNPs, where rows have the same color if they correspond to the SNPs associated with points in the same space tile. On the right Table 3.2 lists the reduced data set D_ρ of SNPs associated with points in the reduced set P_ρ. $\square$

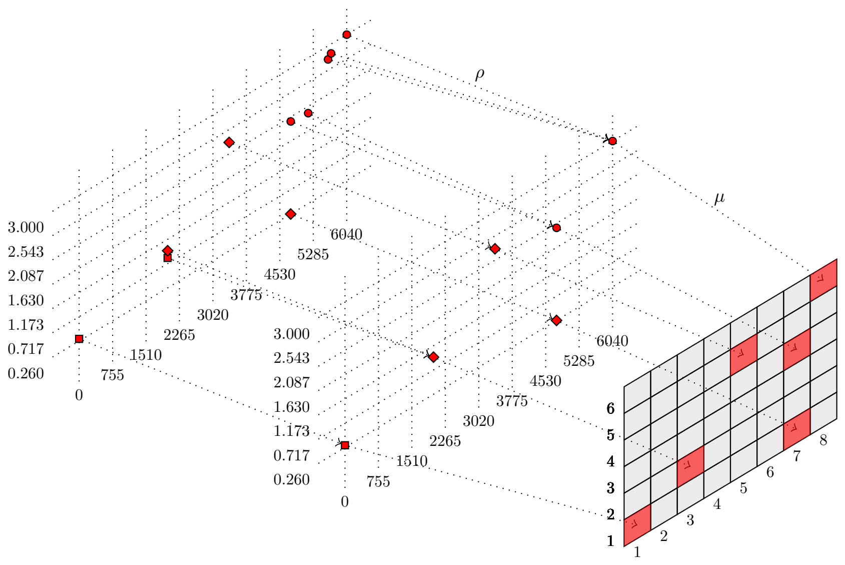


Figure 3.1: Reduction of Points in $\mathbb{R}^2$ Euclidean Space Associated with SNPs and Mapping to Pixels on a Display of Resolution 8×6.

D

	name	chr	pos	pval
d_1	rs987669	1	60160	0.5000
d_2	rs10399597	1	62157	0.1500
d_3	rs55858216	2	60718	0.1100
d_4	rs73140430	2	62106	0.0050
d_5	rs730038	2	63495	0.5500
d_6	rs28729284	3	60202	0.0100
d_7	rs13067307	3	60596	0.0110
d_8	rs9755941	3	61044	0.0018
d_9	rs9756992	3	61113	0.0015
d_{10}	rs13081384	3	61466	0.0010

$$\xrightarrow[\text{Data Reduction}]{P_\rho}$$

D_ρ

	name	chr	pos	pval
d_1	rs987669	1	60160	0.5000
d_3	rs55858216	2	60718	0.1100
d_4	rs73140430	2	62106	0.0050
d_5	rs730038	2	63495	0.5500
d_6	rs28729284	3	60202	0.0100
d_{10}	rs13081384	3	61466	0.0010

Table 3.2: Reduction of SNPs According to Reduced Set of Associated Points in $\mathbb{R}^2$ Euclidean Space.

To conclude, comparing Figure 2.3 and Table 2.1 with Figure 3.1 and Table 3.1 we can see, that the set S of occupied display pixels is the same using both the reduced and the full sets of points associated with data items. Therefore, the visualizations of the reduced data set D_ρ and the full data set D using the same initial μ function, without modifying the display resolution and the bounded subset Ω in $\mathbb{R}^2$ Euclidean space, are identical. Although, the reduced data set may contain less data items than the full initial data set.

Lemma 1 (Equivalency of Visualizations): *Let P be the set of all points in $\mathbb{R}^2$ Euclidean space associated with data set D, and $P_\rho \subseteq P$ be the reduced set of points in $\mathbb{R}^2$ Euclidean space associated with the reduced data set $D_\rho \subseteq D$. Let Ω be the bounded subset of $\mathbb{R}^2$ Euclidean space such that $P \subseteq \Omega$, Δ be the set of all available display pixels, and $\mu : \Omega \rightarrow \Delta$ be the mapping function. If S and S_ρ are two sets of occupied display pixels obtained with the same mapping function μ and sets P and P_ρ respectively, then $S = S_\rho$.* $\diamondsuit$

PROOF Consider the two following cases:

a.) Pixel $s = (i, j)$ belongs to S_ρ and doesn't belong to S.

If pixel s belongs to S_ρ, then exists point (x, y) in the reduced set P_ρ of points in $\mathbb{R}^2$ Euclidean space associated with the reduced data set D_ρ such that $\mu\big((x, y)\big) = s$. Therefore, in the reduced data set D_ρ exists such data item d, that $f(d) = x$ and $g(d) = y$. Since the reduced data set D_ρ is the subset of the full data set D according to Definition 8 and data item d belongs to D_ρ, then data item d belongs to D too. When functions f, g and μ are the same, then point (x, y) belongs to set P of points in $\mathbb{R}^2$ Euclidean space associated with the full data set D and pixel s belongs to set S of occupied display pixels. This is a contradiction and, therefore, the assumption is wrong.

b.) Pixel $s = (i, j)$ belongs to S and doesn't belong to S_ρ.

If pixel s belongs to S, then exists point (x, y) in set P of points in $\mathbb{R}^2$ Euclidean space associated with the full data set D such that $\mu\big((x, y)\big) = s$. Therefore, in the full data set D exists such data item d, that $f(d) = x$ and $g(d) = y$. If data item d also belongs to the reduced data set D_ρ and functions f, g and μ are the same, then point (x, y) belongs to the reduced set P_ρ of points in $\mathbb{R}^2$ Euclidean space associated with the reduced data set D_ρ and pixel s belongs to set S_ρ of occupied display pixels. This is a contradiction and, therefore, the assumption is wrong if the data item d belongs to D and D_ρ at the same time. If data item d doesn't belong to the reduced data set D_ρ and functions f, g and μ are the same, then point (x, y) doesn't belong to the reduced set P_ρ of points in $\mathbb{R}^2$ Euclidean space associated with the reduced data set D_ρ. Since point (x, y) doesn't belong to P_ρ and pixel s doesn't belong to S_ρ, then following from Definition 7 there is no point (x', y') in P such that $\mu\big((x', y')\big) = \mu\big((x, y)\big)$ and, therefore, pixel s doesn't belong to set S of occupied display pixels. This is a contradiction and, therefore, the assumption is wrong if the data item d belongs to D and doesn't belong to D_ρ also. ■

3.2 Tree Definition

A single reduced data set is not sufficient for visualization since it is constructed for the constant display resolution, which usually is a variable value. Our goal is to have an index data structure supporting visualizations in different resolutions. Thus, further we consider a multiple reduced data sets constructed for a finite set of predefined resolutions. They can be stored and queried in a hierarchical tree structure. We refer to this structure as the tree of reduced data sets or the *RDS tree*.

Definition 9 (RDS Tree). *Let D be the data set and P be the set of all points in $\mathbb{R}^2$ Euclidean space associated with data items in D. Let $\mathfrak{R} = \{\, N_1 \times M_1, \ldots, N_k \times M_k \,\}$ be the finite set of display resolutions, where $k \in \mathbb{N}^*$. Then, the RDS tree $\mathfrak{T}$ of reduced data sets is the space-partitioning hierarchical data structure for storing reduced data sets $D_{\rho_1}, \ldots, D_{\rho_k}$, where D_{ρ_i} corresponds to the resolution $N_i \times M_i$ with $1 \le i \le k$.*

The inner node of RDS tree $\mathfrak{T}$ contains the set of elements $(area_i, item_i, child_i)$, where $1 \leq i \leq |D|$. The $area_i$ is the rectangular area on $\mathbb{R}^2$ Euclidean plane. The $item_i$ is the data item in data set D associated with a point $p \in P$ such, that p is the representative point of all points in $area_i$. The $child_i$ is the pointer to a child node.

The leaf node of RDS tree $\mathfrak{T}$ contains the set of elements $(area_i, item_i, data_i)$, where $1 \leq i \leq |D|$. The $area_i$ is the rectangular area on $\mathbb{R}^2$ Euclidean plane. The $item_i$ is the data item in data set D associated with a point $p \in P$ such, that p is the representative point of all points in $area_i$. The $data_i$ is the pointer to all data items in D associated with points in P, that belong to the $area_i$. ♣

Definition 9 defines the RDS tree $\mathfrak{T}$ together with the inner and the leaf nodes of the tree. A particular element entry $area_i$, $item_i$, or $child_i$ of an inner node v is accordingly denoted as $v.area_i$, $v.item_i$, or $v.child_i$. Any two inner nodes can have different number of elements and, therefore, different number of children. The same notation is applied for element entries of a leaf node. Also, as inner nodes, any two leaf nodes can have different number of elements and, therefore, can reference to different number of data subsets. Although, the number of elements in a single inner or a single leaf node can't exceed the number of data items in data set. Taken together, the set of inner and leaf nodes forms such hierarchy, that all rectangular areas on $\mathbb{R}^2$ Euclidean plane in the child node $child_i$ are enclosed by the rectangular area $area_i$ in the parent node. Figure 3.2 depicts the RDS tree consisting of one inner node v_{00} and three

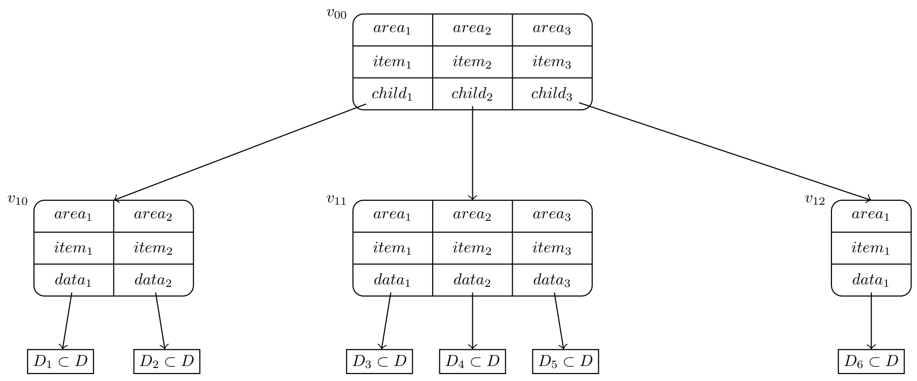


Figure 3.2: RDS Tree

leaf nodes v_{10}, v_{11} and v_{12}. It splits a data set D into six subsets $D_1, \ldots, D_6$ such, that $D = D_1 \cup \cdots \cup D_6$ and $D_1 \cap \cdots \cap D_6 = \emptyset$. If we consider the leaf node v_{10}, then the area $v_{10}.area_1$ encloses all points associated with data items in D_1, while $v_{10}.area_2$ encloses all points associated with data items in D_2. The $v_{10}.item_1 \in D_1$ is associated with the representative point of all points in $v_{10}.area_1$, which are associated with data items in D_1. The $v_{10}.item_2 \in D_2$ is associated with the representative point of all points in $v_{10}.area_2$, which are associated with data items in D_2. Furthermore, $v_{10}.area_1 \cup v_{10}.area_1 \subseteq v_{00}.area_1$ and $v_{00}.item_1 \in D_1 \cup D_2$. Thus, $v_{00}.item_1$ is associated with the representative point of all points in $v_{10}.area_1 \cup v_{10}.area_2$, which are associated with data items in $D_1 \cup D_2$.

3.3 Tree Construction

Consider the finite set $\mathfrak{R} = \{ N_1 \times M_1, \ldots, N_k \times M_k \}$ of display resolutions, where $k \in \mathbb{N}^*$. Assume, we need to construct the RDS tree $\mathfrak{T}$ to support the display resolutions in $\mathfrak{R}$. In particular, the tree must store the reduced data sets $D_{\rho_1}, \ldots, D_{\rho_k}$, where D_{ρ_i} corresponds to the resolution $N_i \times M_i$ with $1 \leq i \leq k$.

Since the RDS tree $\mathfrak{T}$ represents the hierarchy of nested rectangular areas on $\mathbb{R}^2$ Euclidean plane, then we introduce the following restriction on set $\mathfrak{R}$ of display resolutions:

$$\frac{N_{i+1}}{N_i} + \frac{M_{i+1}}{M_i} - \Big(\Big\lfloor \frac{N_{i+1}}{N_i} \Big\rfloor + \Big\lfloor \frac{M_{i+1}}{M_i} \Big\rfloor\Big) = 0, \quad \text{where } 1 \leq i < k. \tag{3.4}$$

The restriction ensures, that the areas in all inner and all leaf nodes don't intersect and, therefore, the data subsets in all leaf nodes don't intersect too. For instance, set $\mathfrak{R} = \{ 2 \times 2, 4 \times 2 \}$ is an example of the valid set of display resolutions, while set $\mathfrak{R} = \{ 2 \times 2, 3 \times 2 \}$ of display resolutions is not allowed.

Algorithm 1: RDS Tree Construction

```
function BuildTree($D$, $\mathfrak{R}$)
    $P \leftarrow \{ (x, y) \mid d \in D \wedge f(d) = x \wedge g(d) = y \}$
    $\Omega \leftarrow P$
    $root \leftarrow TreeNode(D, P, \Omega, \mathfrak{R}, 1)$
    return $root$

function BuildTreeNode($D$, $P$, $\Omega$, $\mathfrak{R}$, $l$)
    $node \leftarrow \emptyset$
    if $l > 1$ then
        $N \times M \leftarrow \frac{N_l}{N_{l-1}} \times \frac{M_l}{M_{l-1}}$
    else
        $N \times M \leftarrow N_l \times M_l$
    $\Delta \leftarrow \{ (i, j) \mid 1 \leq i \leq N \wedge 1 \leq j \leq M \}$
    foreach $(i, j) \in \Delta$ do
        $\Omega' \leftarrow \{ (x, y) \mid (x, y) \in \Omega \wedge \mu\big((x, y)\big) = (i, j) \}$
        $P' \leftarrow \{ (x, y) \mid (x, y) \in P \wedge (x, y) \in \Omega' \}$
        if $P' \neq \emptyset$ then
            $D' \leftarrow \{ d \mid d \in D \wedge \big(f(d), g(d)\big) \in P' \}$
            $d' \leftarrow d : d \in D' \wedge (f(d), g(d)) = \sigma(P')$
            if $l < k$ then
                $child \leftarrow BuildTreeNode(D', P', \Omega', \mathfrak{R}, l + 1)$
                if $node = \emptyset$ then $node \leftarrow$ new $InnerNode$
                $node \leftarrow node \cup (\Omega', d', child)$
            else
                if $node = \emptyset$ then $node \leftarrow$ new $LeafNode$
                $node \leftarrow node \cup (\Omega', d', D')$
    return $node$
```

Algorithm 1 lists a pseudocode of the function $BuildTree(D, \mathfrak{R})$, which constructs the RDS tree containing the reduced data sets for data set D and display resolutions in set $\mathfrak{R}$. It relies on the invocation of the recursive function $BuildTreeNode(D,\ P,\ \Omega,\ \mathfrak{R},\ l)$, where l indicates the ordinal position of resolution in set $\mathfrak{R}$ and Ω is the rectangular area on $\mathbb{R}^2$ Euclidean plane enclosing all points in set P, which are associated with data items in data set D. First, the function estimates the resolution $N \times M$, which is used to produce the required $N_l \times M_l$ resolution from the previous $N_{l-1} \times M_{l-1}$ resolution. Second, it defines the set Δ of available display pixels and, consequently, defines the mapping function μ for the estimated resolution $N \times M$ and rectangular area Ω. Then, it loops through all pixels in set Δ. For every pixel (i, j) it estimates two sets Ω' and P'. The former is the rectangular space tile on $\mathbb{R}^2$ Euclidean plane, which corresponds to the pixel (i, j) and is the result of the tessellation of area Ω produced with the mapping function μ. The latter is the set of all points falling into the area Ω' and associated with data items in set D. If set P' is empty, then no data items are assigned to the pixel (i, j) and, therefore, the corresponding tree node is not created. Otherwise, the function estimates the data subset D' of all data items, which have associated points in set P' only. Then, it selects such data item d' in set D', that it is associated with the representative point in set P'. Afterwards, if not all resolutions in set $\mathfrak{R}$ were covered, then the function is recursively invoked with the input arguments D', P', Ω' and incremented l value. As a result of the recursive invocation, the *child* pointer is obtained. Finally, with respect to the current l value, a new element $(\Omega',\ d',\ child)$ or $(\Omega',\ d',\ D')$ is appended to the inner or to the leaf node respectively.

Example 5 For instance, consider SNPs in data set D listed in Table 2.1. The functions f and g are defined in Equation 2.1 and Equation 2.2 respectively. Equation 3.1 defines the selection function σ. Assume, we need to construct the RDS tree containing the reduced sets of SNPs for display resolutions in $\mathfrak{R} = \{\, 1 \times 1,\ 2 \times 2,\ 4 \times 2,\ 8 \times 6 \,\}$. Then, Figure 3.3 depicts the constructed tree. $\square$

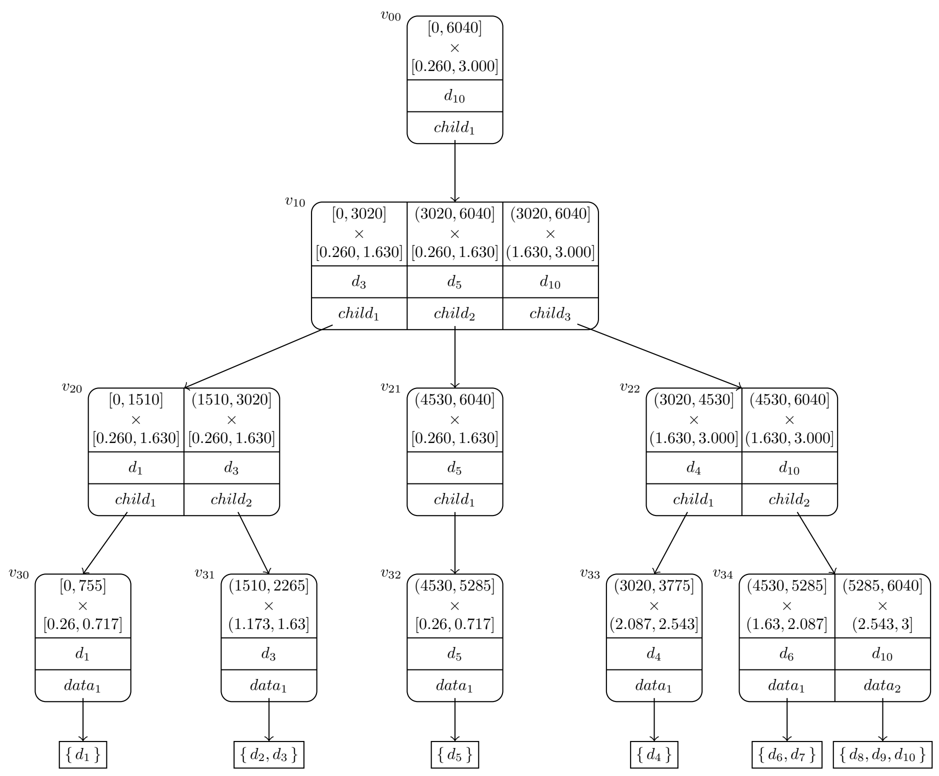

v_{00}
$[0, 6040] \times [0.260, 3.000]$
d_{10}
$child_1$
v_{10}
$[0, 3020] \times [0.260, 1.630]$
$(3020, 6040] \times [0.260, 1.630]$
$(3020, 6040] \times (1.630, 3.000]$
d_3
d_5
d_{10}
$child_1$
$child_2$
$child_3$
v_{20}
$[0, 1510] \times [0.260, 1.630]$
$(1510, 3020] \times [0.260, 1.630]$
d_1
d_3
$child_1$
$child_2$
v_{21}
$(4530, 6040] \times [0.260, 1.630]$
d_5
$child_1$
v_{22}
$(3020, 4530] \times (1.630, 3.000]$
$(4530, 6040] \times (1.630, 3.000]$
d_4
d_{10}
$child_1$
$child_2$
v_{30}
$[0, 755] \times [0.26, 0.717]$
d_1
$data_1$
v_{31}
$(1510, 2265] \times (1.173, 1.63]$
d_3
$data_1$
v_{32}
$(4530, 5285] \times [0.26, 0.717]$
d_5
$data_1$
v_{33}
$(3020, 3775] \times (2.087, 2.543]$
d_4
$data_1$
v_{34}
$(4530, 5285] \times (1.63, 2.087]$
$(5285, 6040] \times (2.543, 3]$
d_6
d_{10}
$data_1$
$data_2$
$\{ d_1 \}$
$\{ d_2, d_3 \}$
$\{ d_5 \}$
$\{ d_4 \}$
$\{ d_6, d_7 \}$
$\{ d_8, d_9, d_{10} \}$

Figure 3.3: RDS Tree of SNPs for Display Resolutions 1×1, 2×2, 4×2 and 8×6.

Chapter 4

Querying the RDS Tree

In this chapter the three essential queries over the RDS tree are introduced. The selection query allows the fast visualization of the whole data set. The window query supports the zoom in interactive operation. While the top K query allows the fast retrieval of additional information on demand.

4.1 Selection Query

The most essential query over the RDS tree is a selection of the reduced data set, which corresponds to the display resolution of interest. In particular, if we want to visualize a data set D on displays of multiple resolutions, then we construct the RDS tree $\mathfrak{T}$ containing the reduced data sets for the finite set $\mathfrak{R}$ of the predefined display resolutions. During the visualization, when the exact display resolution $N \times M$ is defined, we query the RDS tree $\mathfrak{T}$ for the reduced data set D_ρ corresponding to the defined resolution $N \times M$. However, since the RDS tree $\mathfrak{T}$ is constructed for the finite number of predefined display resolutions, then it may not contain the reduced data set D_ρ, which exactly corresponds to the display resolution of interest. If so, such reduced data set ${D_\rho}'$ is extracted from the RDS tree $\mathfrak{T}$, that D_ρ is the subset of ${D_\rho}'$ and ${D_\rho}'$ is the subset of D.

The selection query is based on the depth-first preorder traversal of the RDS tree $\mathfrak{T}$. The traversal termination condition depends on the extent of the rectangular area on $\mathbb{R}^2$ Euclidean plane stored in a tree node element, which encloses all points associated with data items in a child node. In particular, the traversal terminates when the rectangular area falls into a single tile in the space tessellation produced by the mapping function μ to satisfy the current display resolution $N \times M$. Indeed, if the rectangular area falls into a single tile, then all other subareas in a child node fall into the same tile as well and, therefore, all points in these areas are mapped to the same display pixel. Hence, alternatively we can state, that the traversal terminates if the rectangular area is mapped to a single pixel. However, this termination condition is very conservative. Since the major part of visualizations use more than one display pixel to represent a point in $\mathbb{R}^2$ Euclidean space, then this allows the rectangular area to fall into more than one consecutive tiles both horizontally and vertically without the significant influence on the final visualization. For instance, assume a display of resolution $N \times M$ and a point in $\mathbb{R}^2$ Euclidean space mapped to the pixel (i', j'). Then, Figure 4.1 depicts the representation of this point with 21 pixels, which approximate a circle. Next, assume a second point in $\mathbb{R}^2$ Euclidean space

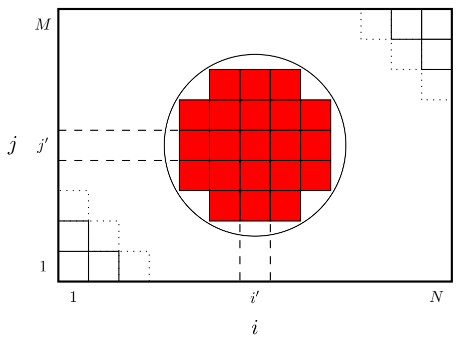


Figure 4.1: Representation of a Point in $\mathbb{R}^2$ Euclidean Space with 21 Pixels on a Display of Resolution $N \times M$ Approximating a Circle.

mapped to the neighbour pixel $(i' + 1, j')$ on the same display of resolution $N \times M$. Then, Figure 4.2 depicts the simultaneous representation of both points on the display. The former point is represented with pixels coloured in red, while the latter point is represented with pixels coloured in semitransparent blue. Although the points are mapped to two different consecutive display pixels, their representations still uses 16 common pixels. As a result, the visualized data items associated with these two points partially overplot one another and are undistinguishable on displays of relatively wide resolution. For that reason, in

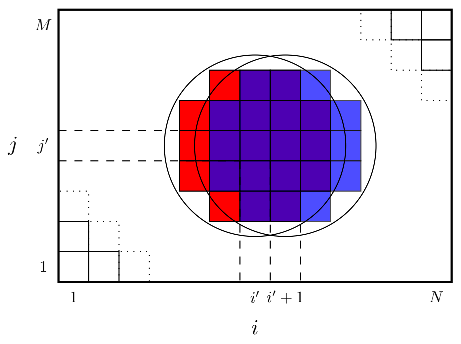


Figure 4.2: Representation of Two Points in $\mathbb{R}^2$ Euclidean Space with 26 Pixels on a Display of Resolution $N \times M$ Approximating Circles.

the tree traversal termination condition we introduce a threshold for the number of tiles to which the rectangular area falls and, therefore, for the number of pixels to which the area is mapped. We refer to this threshold as the *minimum size* of a point on a display. For example, if a point in $\mathbb{R}^2$ Euclidean space is represented with only one display pixel (i.e., only with pixel to which is directly mapped), then the minimum size threshold is the most conservative and is equal to 1. If the *minimum size* threshold is 3, then the point is represented with 3 pixels horizontally and 3 pixels vertically, which in total is 9 pixels.

Algorithm 2: Selection Query

```
function SelectReducedDataSet($\mathfrak{T}$, $N \times M$, $\delta$)
    $D_\rho \leftarrow \emptyset$
    $node \leftarrow root(\mathfrak{T})$
    for $i \leftarrow 1$ to $size(node)$ do
        $\Omega \leftarrow \Omega \cup node.area_i$
    $\Delta \leftarrow \{\, (i,j) \mid 1 \le i \le N \wedge 1 \le j \le M \,\}$
    $Select(node, \Omega, \Delta, \delta, D_\rho)$
    return $D_\rho$

function Select($node$, $\Omega$, $\Delta$, $\delta$, $D_\rho$)
    for $i \leftarrow 1$ to $size(node)$ do
        $area_x \leftarrow \{\, x \mid (x,y) \in node.area_i \,\}$
        $area_y \leftarrow \{\, y \mid (x,y) \in node.area_i \,\}$
        $(i_{bottom}, j_{bottom}) \leftarrow \mu\big((\inf area_x, \inf area_y)\big)$
        if $\inf area_x \neq \inf \Omega_x$ then $i_{bottom} \leftarrow i_{bottom} + 1$
        if $\inf area_y \neq \inf \Omega_y$ then $j_{bottom} \leftarrow j_{bottom} + 1$
        $(i_{top}, j_{top}) \leftarrow \mu\big((\sup area_x, \sup area_y)\big)$
        if $i_{top} - i_{bottom} + 1 \le \delta \wedge j_{top} - j_{bottom} + 1 \le \delta$ then
            $D_\rho \leftarrow D_\rho \cup node.item_i$
        else if $node$ is $LeafNode$ then
            $D_\rho \leftarrow D_\rho \cup node.data_i$
        else if $node$ is $InnerNode$ then
            $Select(node.child_i, \Omega, \Delta, \delta, D_\rho)$
```

Algorithm 2 lists a pseudocode of the function *SelectReducedDataSet*($\mathfrak{T}$, $N \times M$, δ). It selects the reduced data set D_ρ in the RDS tree $\mathfrak{T}$ according to the display resolution $N \times M$ and the threshold δ, which denotes the minimum size of a point on the display. The function starts with the estimation of the rectangular area Ω on $\mathbb{R}^2$ Euclidean plane, which encloses all points associated with data items in data set D. All rectangular areas stored in the root node of the RDS tree $\mathfrak{T}$ constitute the whole area Ω. After the Ω is known, the set Δ of available display pixels is estimated according to the resolution $N \times M$ and, consequently, the mapping function μ is defined and ready to use in all further steps. Then, the recursive function *Select*(*node*, Ω, Δ, δ, D_ρ) is invoked. It implements the depth-first preorder traversal of the RDS tree $\mathfrak{T}$, which always starts with the root node and, therefore, the *node* input argument is specified accordingly. The D_ρ input argument is the reduced data set of interest, which initially is empty.

For every element in the *node* function *Select*(*node*, Ω, Δ, δ, D_ρ) tests the traversal termination condition. In particular, first it estimates $area_x$ and $area_y$ of the rectangular area $node.area_i$ on the $\mathbb{R}^2$ Euclidean plane, where the former corresponds the horizontal extent and the latter corresponds to the vertical extent of the area. Second, by the means of the mapping function μ it estimates the number of tiles occupied with $node.area_i$. Every tile corresponds to the pixel (i, j) in set Δ of available display pixels. Therefore, the left bottom point ($\inf area_x$, $\inf area_y$) of the extent of $node.area_i$ is mapped to the pixel (i_{bottom}, j_{bottom}), while the right top point ($\sup area_x$, $\sup area_y$) of the extent of $node.area_i$ is mapped to the pixel (i_{top}, j_{top}). Figure 4.3 depicts an example of a hypothetic $node.area_i$ with occupied pixels on a display of resolution $N \times M$. The coordinates of two pixels (i_{bottom}, j_{bottom}) and (i_{top}, j_{top}) describe the total

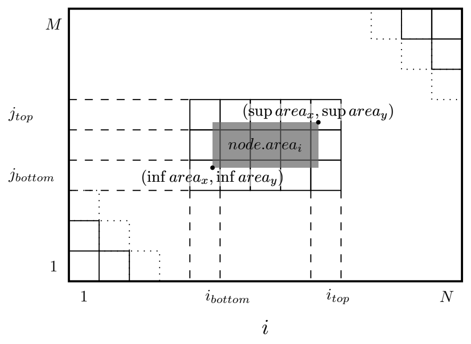


Figure 4.3: A Hypothetic Rectangular Area on $\mathbb{R}^2$ Euclidean Plane with Occupied Pixels on a Display of Resolution $N \times M$.

number of display pixels and, therefore, the total number of tiles occupied with $node.area_i$. The number of occupied tiles horizontally is equal to $i_{top} - i_{bottom} + 1$, while the number of occupied tiles vertically is equal to $j_{top} - j_{bottom} + 1$. If both numbers are smaller than the threshold δ, then the corresponding data item $node.item_i$ is appended to the reduced data set D_ρ and the further traversal to the $node.child_i$ is omitted. Otherwise, if $node$ is a leaf node, then the corresponding set of data items $node.data_i$ is appended to the reduced data set D_ρ. Or, if $node$ is an inner node, then the function is invoked on the corresponding child node $node.child_i$ and, therefore, the tree traversal continues by going one level in depth.

Example 6 To illustrate the selection query, consider data set D of ten SNPs listed in Table 2.1 and the corresponding RDS tree $\mathfrak{T}$ containing the reduced data sets for the display resolutions in $\mathfrak{R} = \{ 1 \times 1,\ 2 \times 2,\ 4 \times 2,\ 8 \times 6 \}$ and depicted in Figure 3.3. Assume, the current display resolution is 4×2 and a point in $\mathbb{R}^2$ Euclidean space is represented with the exactly one display pixel. Consequently, we need to query the RDS tree $\mathfrak{T}$ for the reduced data set $D_{\rho_{4\times2}}$ of SNPs, which corresponds to the display resolution 4×2 and satisfies the threshold $\delta = 1$ for the minimum size of a point on the display.

In the first step, the rectangular area Ω on $\mathbb{R}^2$ Euclidean plane is estimated and is equal to $[0, 6040] \times [0.260, 3.000]$. Then, the set Δ of available display pixels is defined according to the resolution 4×2 and is equal to $\{ (1, 1), \ldots, (4, 1), (1, 2), \ldots, (4, 2) \}$. Therefore, the mapping function μ is defined so, that it maps all points in $\Omega = [0, 6040] \times [0.260, 3.000]$ to the pixels in set Δ. After the mapping function is defined, the recursive tree traversal function is invoked with the arguments $Select(v_{00},\ \Omega,\ \Delta,\ 1,\ D_{\rho_{4\times2}})$.

First, the root node v_{00} is considered. The left bottom point $(0, 0.260)$ of the extent of $v_{00}.area_1$ is mapped to the pixel $(1, 1)$, while the right top point $(6040, 3.000)$ of the extent of $v_{00}.area_1$ is mapped to the pixel $(4, 2)$. Thus, $v_{00}.area_1$ occupies 4 display pixels horizontally and 2 display pixels vertically, which satisfy the minimum size threshold $\delta = 1$. Figure 4.4 depicts the $v_{00}.area_1$ and the occupied display pixels. As a result, the function goes one level in depth to the node v_{10} and considers three areas $v_{10}.area_1$, $v_{10}.area_2$ and $v_{10}.area_3$.

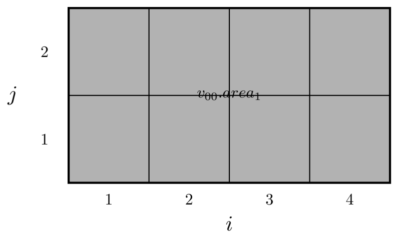


Figure 4.4: Display of Resolution 4×2 and Pixels Occupied with Areas in RDS Tree Node v_{00}.

Again, the left bottom point $(0, 0.260)$ of the extent of $v_{10}.area_1$ is mapped to the pixel $(1, 1)$, while the right top point $(3020, 1.630)$ of this extent is mapped to the pixel $(2, 1)$. Therefore, $v_{10}.area_1$ satisfies the minimum size threshold $\delta = 1$, since it occupies 2 display pixels horizontally. The according extreme points $(3020, 0.260)$ and $(6040, 1.630)$ of the extent of $v_{10}.area_2$ are mapped to the pixels $(3, 1)$ and $(4, 1)$, respectively. Hence, $v_{10}.area_2$ occupies 2 pixels horizontally and, therefore, satisfies the minimum size threshold $\delta = 1$. Finally, the left bottom point $(3020, 1.630)$ and the right top point $(6040, 3.000)$ of the extent of $v_{10}.area_3$ are mapped to the according pixels $(3, 2)$ and $(4, 2)$. Thus, $v_{10}.area_3$ occupies 2 pixels horizontally and satisfies the threshold $\delta = 1$ as well as the previous two areas. Figure 4.5 depicts the areas $v_{10}.area_1$, $v_{10}.area_2$ and $v_{10}.area_3$ with the corresponding occupied display pixels. Since all three considered areas satisfy the necessary conditions, then the tree traversal continues with the corresponding children nodes v_{20}, v_{21} and v_{22}.

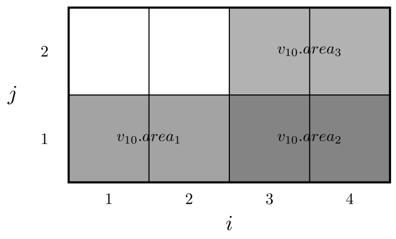


Figure 4.5: Display of Resolution 4×2 and Pixels Occupied with Areas in RDS Tree Node v_{10}.

Let start with the node v_{20} containing two areas $v_{20}.area_1$ and $v_{20}.area_2$. The left bottom point $(0, 0.260)$ and the right top point $(1510, 1.630)$ of the extent of $v_{20}.area_1$ are mapped to the same display pixel $(1, 1)$. Obviously, $v_{20}.area_1$ doesn't satisfy the minimum size threshold $\delta = 1$, since it occupies exactly one pixel. Therefore, corresponding data item d_1 is appended to the reduced data set $D_{\rho_{4\times 2}}$ of SNPs and the child node $v_{20}.child_1$ is not considered in the further traversal. The analogous situation is with the $v_{20}.area_2$, since the according extreme points $(1510, 0.260)$ and $(3020, 1.630)$ of the extent of this area are mapped to the same display pixel $(2, 1)$. Thus, the data item d_3 is appended to the reduced data set $D_{\rho_{4\times 2}}$ of SNPs and $v_{20}.child_2$ is omitted from the further tree traversal. The two other nodes v_{21} and v_{22} of the tree $\mathfrak{T}$ are processed in the same way. Figure 4.6 depicts the areas in nodes v_{20}, v_{21} and v_{22} with the corresponding occupied display pixels. As a result, data items d_5, d_4 and d_{10} are appended to the reduced data set $D_{\rho_{4\times 2}}$ of SNPs and corresponding child nodes $v_{21}.child_1$, $v_{22}.child_1$ and $v_{21}.child_2$ are not considered.

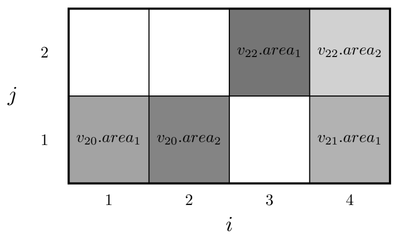


Figure 4.6: Display of Resolution 4 × 2 and Pixels Occupied with Areas in RDS Tree Nodes v_{20}, v_{21} and v_{22}.

	name	chr	pos	pval
d_{10}	rs13081384	3	61466	0.0010

$D_{\rho_{1\times 1}}$

$\xrightarrow{\text{Step 1}}$

	x	y
d_{10}	6040	3.000

$P_{\rho_{1\times 1}}$

$\xrightarrow{\text{Step 2}}$

	i	j
d_{10}	1	1

$S_{1\times 1}$

(a) Display Resolution 1 × 1

	name	chr	pos	pval
d_3	rs55858216	2	60718	0.1100
d_5	rs730038	2	63495	0.5500
d_{10}	rs13081384	3	61466	0.0010

$D_{\rho_{2\times 2}}$

$\xrightarrow{\text{Step 1}}$

	x	y
d_3	1998	0.959
d_5	4775	0.260
d_{10}	6040	3.000

$P_{\rho_{2\times 2}}$

$\xrightarrow{\text{Step 2}}$

	i	j
d_3	1	1
d_5	2	1
d_{10}	2	2

$S_{2\times 2}$

(b) Display Resolution 2 × 2

	name	chr	pos	pval
d_1	rs987669	1	60160	0.5000
d_3	rs55858216	2	60718	0.1100
d_4	rs73140430	2	62106	0.0050
d_5	rs730038	2	63495	0.5500
d_{10}	rs13081384	3	61466	0.0010

$D_{\rho_{4\times 2}}$

$\xrightarrow{\text{Step 1}}$

	x	y
d_1	0	0.301
d_3	1998	0.959
d_4	3386	2.301
d_5	4775	0.260
d_{10}	6040	3.000

$P_{\rho_{4\times 2}}$

$\xrightarrow{\text{Step 2}}$

	i	j
d_1	1	1
d_3	2	1
d_4	3	2
d_5	4	1
d_{10}	4	2

$S_{4\times 2}$

(c) Display Resolution 4 × 2

	name	chr	pos	pval
d_1	rs987669	1	60160	0.5000
d_3	rs55858216	2	60718	0.1100
d_4	rs73140430	2	62106	0.0050
d_5	rs730038	2	63495	0.5500
d_6	rs28729284	3	60202	0.0100
d_{10}	rs13081384	3	61466	0.0010

$D_{\rho_{8\times 6}}$

$\xrightarrow{\text{Step 1}}$

	x	y
d_1	0	0.301
d_3	1998	0.959
d_4	3386	2.301
d_5	4775	0.260
d_6	4776	2.000
d_{10}	6040	3.000

$P_{\rho_{8\times 6}}$

$\xrightarrow{\text{Step 2}}$

	i	j
d_1	1	1
d_3	3	2
d_4	5	5
d_5	7	1
d_6	7	4
d_{10}	8	6

$S_{8\times 6}$

(d) Display Resolution 8 × 6

Table 4.1: Assignment of Reduced Sets of SNPs to Pixels on Displays of Resolutions 1 × 1, 2 × 2, 4 × 2 and 8 × 6.

Table 4.1c lists the reduced data set $D_{\rho_{4\times 2}}$ of SNPs for the display resolution

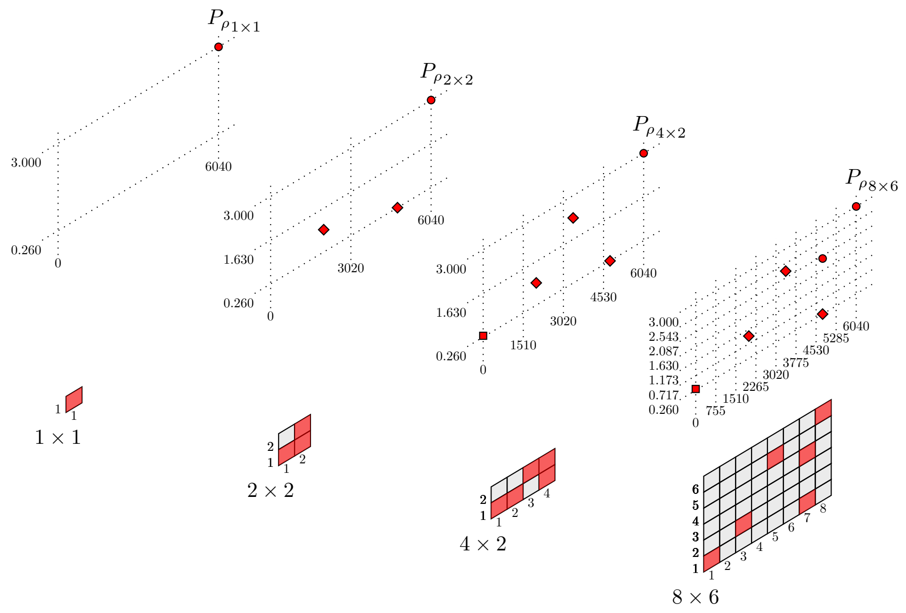


Figure 4.7: Mapping of Reduced Sets of Points in $\mathbb{R}^2$ Euclidean Space Associated with Reduced Sets of SNPs to Pixels on Displays of Resolutions 1×1, 2×2, 4×2 and 8×6.

4×2. Along with the reduced data set of SNPs, the table lists the reduced set $P_{\rho_{4\times2}}$ of points in $\mathbb{R}^2$ Euclidean space associated with SNPs, and set $S_{4\times2}$ of all occupied display pixels. Tables 4.1a, 4.1b and 4.1d list corresponding sets for display resolutions 1×1, 2×2 and 8×6 respectively. Finally, Figure 4.7 depicts the reduced sets of associated points in $\mathbb{R}^2$ Euclidean space and the mapping of these points to pixels. The rectangular points refer to SNPs from the 1st chromosome, the circular points refer to SNPs from the 2nd chromosome, and the points of the shape of a diamond denote SNPs from the 3rd chromosome.$\square$

4.2 Window Query

A window query allows to define the rectangular area on $\mathbb{R}^2$ Euclidean plane and retrieve such data items, that they are associated with points in the defined area only. The query is made after the whole data set is visualized in $\mathbb{R}^2$ Euclidean space, since only then the smaller subregions of interest can be observed and identified. After the subregion of interest is defined, then all available display pixels are dedicated to the visualization of this region particularly. As a result, more pixels are available to visualize the same or the smaller number of data items and, therefore, more fine-grained visualization is produced. Thereby, we refer to the window query as the zoom in operation, or as the drill down operation.

Obviously, the data subset retrieved with window query may still contain millions of data items and a part of them may be assigned to the same display pixels. To reduce the number of retrieved points we define the window query over the RDS tree $\mathfrak{T}$. In this case, the query retrieves data items considering the region of interest, the display resolution and the minimum size of a point on the display. It attempts to retrieve only non-overplotting data items, which produce the actual difference in visualization.

Assume the window query is specified with the rectangular area $W = \{\, (x, y) \mid x \in \mathbb{R} \wedge y \in \mathbb{R} \,\}$ on $\mathbb{R}^2$ Euclidean plane. Further, we refer to this area as the *window area*. Then, all pixels on a display of resolution $N \times M$ are dedicated to visualize only the data items associated with the points belonging to the specified window area. Therefore, the mapping function μ is defined only over the window area and produces the tessellation of it with respect to the display resolution $N \times M$. Consequently, such reduced data set D_ρ is selected in the RDS tree $\mathfrak{T}$, that it contains only data items associated with points in W and corresponds to the resolution $N \times M$. Also, the minimum size of a point on the display is taken into account when constructing the reduced data set D_ρ.

The two cases are considered during the traversal of the RDS tree $\mathfrak{T}$: (i) the rectangular area stored in the tree node is completely enclosed with the window area W; (ii) the rectangular area stored in the tree node only partially intersects with the window area W and, therefore, is on the border of the region of interest. On the left side of Figure 4.8 is depicted a hypothetical space tessellation stored in the tree node and the two cases. The rectangular areas colored in light blue correspond to the first case, while the areas colored in red correspond to the second case. In the first case, the depth-first tree traversal terminates if the rectangular area in a tree node doesn't satisfy the minimum size threshold. Specifically, it terminates when the number of pixels occupied with the area is smaller than the minimum size of a point on the display, and the data item associated with the representative point of all points in the area is appended to the reduced data set D_ρ. In the second case, the tree traversal continues until the leaf node of the RDS tree $\mathfrak{T}$ is reached. When the leaf node is reached, then all data items referenced by the leaf node and associated with points in window area W are appended to the reduced data set D_ρ. As a result, more fine-grained tessellation of space is considered near the borders of the window area W and, therefore, more data items may be selected in this areas. On the right side of Figure 4.8 is depicted a hypothetical example of the second case with more fine-grained space tessellation near the borders of W.

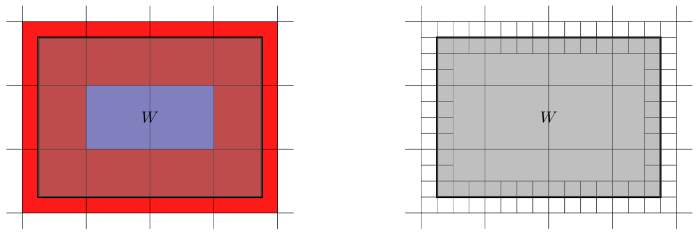


Figure 4.8: Two Cases in the Traversal of the RDS Tree for Window Query.

Algorithm 3 lists a pseudocode of the function $SelectWindowRDS(W, \mathfrak{T}, N \times M, \delta)$, where W is the rectangular area on $\mathbb{R}^2$ Euclidean plane denoting the window query, $\mathfrak{T}$ is the RDS tree, $N \times M$ is the current display resolution, and δ is the minimum size of point on the display. The function selects the reduced data set D_ρ in the RDS tree $\mathfrak{T}$ satisfying the

Algorithm 3: Window Query

```
function SelectWindowRDS($W$, $\mathfrak{T}$, $N \times M$, $\delta$)
    $D_\rho \leftarrow \emptyset$
    $node \leftarrow root(\mathfrak{T})$
    $\Omega \leftarrow W$
    $\Delta \leftarrow \{\, (i,j) \mid 1 \leq i \leq N \wedge 1 \leq j \leq M \,\}$
    $SelectWindow(node,\ \Omega,\ \Delta,\ \delta,\ D_\rho)$
    return $D_\rho$

function SelectWindow($node$, $\Omega$, $\Delta$, $\delta$, $D_\rho$)
    for $i \leftarrow 1$ to $size(node)$ do
        $overlap \leftarrow \Omega \cap node.area_i$
        if $overlap = node.area_i$ then
            $area_x \leftarrow \{\, x \mid (x,y) \in node.area_i \,\}$
            $area_y \leftarrow \{\, y \mid (x,y) \in node.area_i \,\}$
            $(i_{bottom},\ j_{bottom}) \leftarrow \mu\big((\inf area_x,\ \inf area_y)\big)$
            if $\inf area_x \neq \inf \Omega_x$ then $i_{bottom} \leftarrow i_{bottom} + 1$
            if $\inf area_y \neq \inf \Omega_y$ then $j_{bottom} \leftarrow j_{bottom} + 1$
            $(i_{top},\ j_{top}) \leftarrow \mu\big((\sup area_x,\ \sup area_y)\big)$
            if $i_{top} - i_{bottom} + 1 \leq \delta \wedge j_{top} - j_{bottom} + 1 \leq \delta$ then
                $D_\rho \leftarrow D_\rho \cup node.item_i$
            else if $node$ is $LeafNode$ then
                $D_\rho \leftarrow D_\rho \cup node.data_i$
            else if $node$ is $InnerNode$ then
                $SelectWindow(node.child_i,\ \Omega,\ \Delta,\ \delta,\ D_\rho)$
        else if $overlap \neq \emptyset$ then
            if $node$ is $LeafNode$ then
                foreach $item \in node.data_i$ do
                    if $\big(f(item), g(item)\big) \in overlap$ then
                        $D_\rho \leftarrow D_\rho \cup item$
            else if $node$ is $InnerNode$ then
                $SelectWindow(node.child_i,\ \Omega,\ \Delta,\ \delta,\ D_\rho)$
```

current display resolution, the minimum size threshold, and containing only the data items associated with points in window area W. It starts with the initialization of set D_ρ to the empty set and specifying the root node of the RDS tree $\mathfrak{T}$ as the start node for the tree traversal. The bounded rectangular subset Ω of $\mathbb{R}^2$ Euclidean space is initialized with the window area W. The set Δ of available display pixels is defined according to the display resolution $N \times M$. Finally, the recursive function $SelectWindow(node,\ \Omega,\ \Delta,\ \delta,\ D_\rho)$ is invoked, which implements the depth-first preorder traversal of the RDS tree. As a result, the reduced data set D_ρ satisfying the window query, the display resolution $N \times M$ and the minimum size threshold is returned.

Function $SelectWindow(node,\ \Omega,\ \Delta,\ \delta,\ D_\rho)$ checks every element in the *node*. In particular, it estimates intersection between window area W and the rectangular area $node.area_i$. If intersection uquals to the whole $node.area_i$ and, therefore, W completely contains $node.area_i$, then the function calculates the number of display pixels occupied with $node.area_i$. If it doesn't satisfy the minimum size threshold δ, then the corresponding data item $node.item_i$ is appended to the reduced data set D_ρ. Otherwise, if node is a leaf node, then the corresponding set of data items $node.data_i$ is appended to the reduced data set D_ρ. Or, if node is an inner node, then the function is invoked on the corresponding child node $node.child_i$ and, therefore, the tree traversal continues by going one level in depth. If intersection between W and $node.area_i$ is not equal to the whole $node.area_i$ and is not the empty set, then the function traverse the RDS tree $\mathfrak{T}$ until the leaf node is reached. Then, every data item in $node.data_i$ associated with points in intersection between W and $node.area_i$ is appended to the reduced data set D_ρ.

Example 7 Consider data set D of ten SNPs and set P of points in $\mathbb{R}^2$ Euclidean space associated with SNPs listed in Table 2.1. Figure 3.3 depicts the RDS tree $\mathfrak{T}$ containing the reduced data sets for the display resolutions in $\mathfrak{R} = \{1 \times 1,\ 2 \times 2,\ 4 \times 2,\ 8 \times 6\}$. Assume, the whole data set is initially visualized on the display of resolution 8×6 and every point in $\mathbb{R}^2$ Euclidean space is represented with single display pixel. Also, the initial visualization utilized the corresponding reduced data set in the RDS tree $\mathfrak{T}$. Suppose the window query $W = [5000, 2.000] \times [6040, 3.000]$ is executed. Then, the bounded rectangular subset Ω of $\mathbb{R}^2$ Euclidean space is equal to the window area W and set Δ of available display pixels is specified according to the resolution 8×6. Therefore, the corresponding mapping function μ is defined as follows:

$$\mu\big((x, y)\big) = (i, j), \tag{4.1}$$

where

$$i = \begin{cases} \left\lceil 8 \times \frac{x-5000}{6040-5000} \right\rceil & , 5000 < x \le 6040 \\ 1 & , x = 5000 \end{cases}$$

and

$$j = \begin{cases} \left\lceil 6 \times \frac{y-2.000}{3.000-2.000} \right\rceil & , 2.000 < y \le 3.000 \\ 1 & , y = 2.000 \end{cases}$$

The depth-first preorder traversal of the RDS tree $\mathfrak{T}$ starts at the root node v_{00} with the initially empty reduced data set D_ρ and with the minimum size threshold $\delta = 1$. Since the $v_{00}.area_1$ partially intersects with the window area W, then the traversal continues with the inner node v_{10} containing three areas. The first two areas $v_{10}.area_1$ and $v_{10}.area_2$ don't intersect with W and, therefore, the corresponding child nodes $v_{10}.child_1$ and $v_{10}.child_2$ are not considered further. Although, the last area $v_{10}.area_3$ with the extent $[3020, 6040] \times [1.630, 3.000]$ partially intersects with W. Since the intersection is only partial, then the minimum size threshold is not considered and the traversal goes one level in depth to the node v_{22}. Again, in the inner node v_{22} only one area, $v_{22}.area_2$, partially intersects with the window query W and, therefore, directs the further traversal to the child node v_{34}. Since v_{34} is the leaf node and $v_{34}.area_1$ partially intersects with W, then we consider all data items in $v_{34}.data_1$. The data item d_6 is associated with point $(4776, 2.000)$, while the data item d_7 is associated with point $(5170, 1.959)$. Both points are outside of the window area and the corresponding data items don't contribute to the reduced data set D_ρ. The next area $v_{34}.area_1$ completely intersects with the window area W. The left bottom point $(5285, 2.543)$ of the extent of $v_{34}.area_1$ is mapped to the display pixel $(3, 4)$, while the right top point $(6040, 3.000)$ of this extent is mapped to the display pixel $(8, 6)$. Therefore, the area satisfies the minimum size threshold $\delta = 1$ both horizontally and vertically. As a result, all data items in $v_{34}.data_2$ (i.e., d_8, d_9 and d_{10}) are appended to the reduced data set D_ρ.

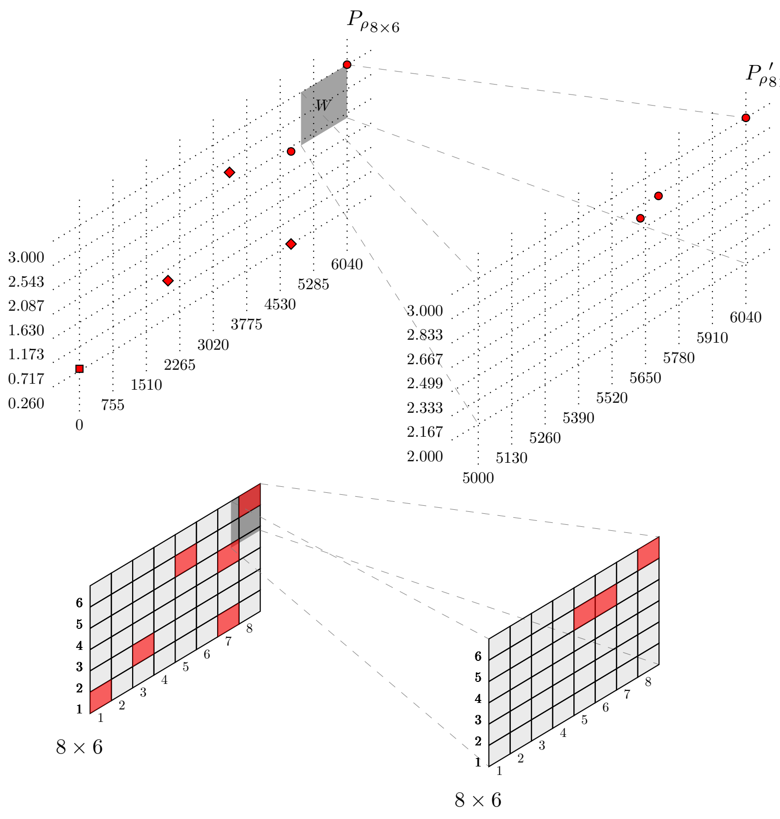


Figure 4.9: Mapping of the Reduced Set of Points in $\mathbb{R}^2$ Euclidean Space Satisfying the Window Query and Associated with the Reduced Set of SNPs to Pixels on a Display of Resolution 8×6.

Figure 4.9 depicts the reduced sets of associated points in $\mathbb{R}^2$ Euclidean space and the mapping of these points to pixels on a display of resolution 8×6 with respect to the window area W. On the left side is depicted the complete reduced set $P_{\rho_{8\times6}}$ of all associated points for the resolution 8×6 and the mapping of these points to the display pixels. The rectangular area $W = [5000, 2.000]\times[6040, 3.000]$ corresponding to the window query is highlighted with gray color. At the same time, on the right side of Figure 4.9 is depicted the reduced set $P'_{\rho_{8\times6}}$ of all associated points satisfying the window area W and the mapping of these points to the display pixels. In the latter mapping, the three data items d_8, d_9 and d_{10} are clearly distinguishable, since all display pixels are dedicated for the visualization of the relatively small area W.

4.3 Top K Query

As was discussed before, the visualization of the reduced data set produces the same perceptional information as the visualization of the full data set. However, to have a complete and a comprehensive view of the whole data, the information about the data items which were eliminated and, therefore, are not directly represented in the visualization is necessary. In this section is introduced an algorithm extracting K data items from an arbitrary node in the RDS tree $\mathfrak{T}$. The extracted data items are not visualized in $\mathbb{R}^2$ Euclidean space, although they constitute highly important auxiliary information about the data set. For instance, if an analyst knows what data items are assigned to the same display pixel, then he or she can decide to zoom in to the particular area and, therefore, have more fine-grained visualization of the area of interest.

Since only K data items are selected in the RDS tree $\mathfrak{T}$, then we need to prioritize all data items according to their significance in the data set and select only top K of them. The exact selection procedure depends on the way the RDS tree $\mathfrak{T}$ is constructed. In particular, it is important which associated point is selected as the representative point of all points in the rectangular area on $\mathbb{R}^2$ Euclidean plane stored in the tree node. The two cases are considered:

Case 1:

The selection of the representative point of all points in a rectangular area on $\mathbb{R}^2$ Euclidean plane stored in the tree node is done without any consideration to the associated data items. In this case, no prior information is known about the data item stored in the tree node. With the same probability, it can be the most significant data item between all other items referenced in the whole subtree, or can be the most unimportant item. Therefore, the traversal of the RDS tree $\mathfrak{T}$ starts with an arbitrary node of interest and the termination condition is not known in advance. As a result, the traversal continues until the leaf nodes are reached. Then, for every data item in every data subset referenced with the leaf node three cases are considered: (i) the result data set already contains K data items and the data item in subset is less significant than any of them; (ii) the result data set already contains K data items and the data item in subset is more significant than some of them; (iii) the result data set contains less than K data items. In the first case, the data item in subset is not considered, since it doesn't have influence on the final result data set. In the second case, the data item in subset is appended to the result data set, while the least significant data item in the result data set is removed to keep only K items. In the last case, the data item in subset is appended to the result data set without any additional checks.

Algorithm 4 lists a pseudocode of the function $SelectTopK(node, K)$, which selects top K data items in the RDS tree $\mathfrak{T}$ starting with the specified $node$. The function initializes a data set D_{top} of top K data items to the empty set. Then, it invokes the recursive function $NodeK(node, K, D_{top})$ implementing the depth-first recursive traversal of the RDS tree $\mathfrak{T}$. After the recursion terminates, the set D_{top} contains top K data items of interest ordered in descending order according to their significance.

Algorithm 4: Top K Query

```
function SelectTopK(node, K)
    $D_{top} \leftarrow \emptyset$
    $NodeK(node, K, D_{top})$
    return $D_{top}$

function NodeK(node, K, D_top)
    for $i \leftarrow 1$ to $size(node)$ do
        if $node$ is $InnerNode$ then
            $NodeK(node.child_i, K, D_{top})$
        else if $node$ is $LeafNode$ then
            foreach $item \in node.data_i$ do
                if $size(D_{top}) \geq K$ then
                    if $item < D_{top}[K]$ then
                        $D_{top} \leftarrow D_{top} \setminus D_{top}[K]$
                        $D_{top} \leftarrow D_{top} \cup item$
                        $D_{top} \leftarrow order(D_{top})$
                else
                    $D_{top} \leftarrow D_{top} \cup item$
                    $D_{top} \leftarrow order(D_{top})$
```

The recursive function $NodeK(node, K, D_{top})$ considers every element in the current *node* of the RDS tree $\mathfrak{T}$. If *node* is the inner node, then the function goes one level in depth and further considers the child node $node.child_i$. Otherwise, when *node* is the leaf node, it loops through every data item in the data subset $node.data_i$. According to the first case, if the result data set D_{top} already contains K data items and the *item* in $node.data_i$ is less significant than the least significant item $D_{top}[K]$ in the result data set D_{top}, then *item* is not considered further and the procedure continues with the next data item in $node.data_i$. In the second case, if set D_{top} already contains K data items and *item* is more significant than $D_{top}[K]$, then $D_{top}[K]$ is removed from the result data set D_{top} and *item* is appended to D_{top}. After the new data item is appended, then all data items in the result data set D_{top} are ordered in descending order according to their significance. Following the third case, when the size of set D_{top} is less than K, then *item* in $node.data_i$ is appended to D_{top} without any additional checks. Again, after the new data item is appended, then D_{top} is ordered in descending order according to the significance of the data items.

Example 8 For instance, consider the data set D of ten SNPs listed in Table 2.1 and the RDS tree $\mathfrak{T}$ containing the reduced data sets for the display resolutions in $\mathfrak{R} = \{ 1 \times 1, 2 \times 2, 4 \times 2, 8 \times 6 \}$ and depicted in Figure 3.3. Assume we need to extract such two SNPs from the node v_{22}, that their names are first in the alphabetical order. First, the leaf node v_{33} is considered. The D_{top} is empty and, therefore, the third case is satisfied. As a result, the data item d_4 in $v_{33}.data_1$ is appended to D_{top}. Second, the leaf node v_{34} is considered and $D_{top} = \{ d_4 \}$. The first data item d_6 in $v_{34}.data_1$ is appended to D_{top}, since the third case was satisfied again. Then, d_4 and d_6 in D_{top} are ordered descendingly according to their *name* attributes $rs73140430$ and $rs28729284$, respectively. Since $rs28729284$ is before $rs73140430$ in descending alphabetical order, then $d_6 < d_4$ and $D_{subset} = \{ d_6, d_4 \}$. Consequently, when considering next data item d_7 in $v_{34}.data_1$, then size of D_{top} is equal to 2 and $D_{top}[2] = d_4$. Evidently, d_7 is more significant than d_4, since $rs13067307$ is before $rs73140430$ in descending alphabetical order. Then, according to the second case, d_4 is removed from D_{top} and d_7 is appended to D_{top}. After the ordering the result data set D_{top} is equal to $\{ d_7, d_6 \}$. The next data subset to consider is $v_{34}.data_2 = \{ d_8, d_9, d_{10} \}$. Since D_{top} already contains two data items and $d_8 > d_6$ with $d_9 > d_6$, then d_8 and d_9 are not considered with respect to the first case. However, d_{10} is more significant than d_6 and, therefore, d_6 is removed from D_{top} while d_{10} is appended according to the second case. When D_{top} is ordered, then it is equal to $\{ d_7, d_{10} \}$. □

Case 2:

The representative point of all points in a rectangular area on $\mathbb{R}^2$ Euclidean space is selected such, that it is associated with the most significant data item between all data items associated with points in the same area. In this case, the traversal of the RDS tree $\mathfrak{T}$ can be terminated according to the data item stored in the element of tree node. Therefore, the number of visited leaf nodes is decreased significantly. Indeed, assume D_{top} already contains K data items and *node.item*$_i$ is the most significant data item in the whole subtree referenced with *node.child*$_i$. Then, if *node.item*$_i$ is less significant than the least significant data item $D_{top}[K]$ in D_{top}, then the whole subtree starting with *node.child*$_i$ is omitted from the traversal.

Algorithm 5: Top K Query

```
function SelectTopK(node, K)
    $D_{top} \leftarrow \emptyset$
    if $node$ is $InnerNode$ then
        $InnerNodeK(node, K, D_{top})$
    else
        $LeafNodeK(node, K, D_{top})$
    return $D_{top}$
```

Algorithm 5 lists a pseudocode of the function *SelectTopK*(*node*, *K*), which selects top K data items in the RDS tree $\mathfrak{T}$ starting with the *node* and assuming *node.item*$_i$ is the most significant data item in the whole subtree referenced with *node.child*$_i$. It initializes a data set D_{top} of top K data items to the empty set and invokes function $InnerNodeK(node, K, D_{top})$ or $LeafNodeK(node, K, D_{top})$ if *node* is an inner or a leaf node, respectively. The former function implements the depth-first recursive traversal of the RDS tree $\mathfrak{T}$, while the latter function considers data subsets referenced in the leaf node. As a result, the set D_{top} contains top K data items of interest ordered in descending order according to their significance.

Algorithm 6 lists a pseudocode of the function $InnerNodeK(node, K, D_{top})$. First, it orders the data items in *node* in descending order according to their

Algorithm 6: Top K Query Over Inner Node

```
function $InnerNodeK(node,\ K,\ D_{top})$
    $D_{items} \leftarrow order(node.items)$
    foreach $node.item_i \in D_{items}$ do
        if $size(D_{top}) \geq K \land node.item_i \geq D_{top}[K]$ then
            return
        else
            if $node.child_i$ is $InnerNode$ then
                $InnerNodeK(node.child_i,\ K,\ D_{top})$
            else
                $LeafNodeK(node.child_i,\ K,\ D_{top})$
```

significance. Then, the function loops through the ordered set of data items and considers every *data.item*$_i$. If set D_{top} already contains K data items and *data.item*$_i$ is less significant than the least significant item $D_{top}[K]$ in D_{top}, then the traversal of the RDS tree $\mathfrak{T}$ terminates with D_{top} containing the final result. Otherwise, the traversal continues with the child node *node.child*$_i$. If *node.child*$_i$ is an inner node, then $InnerNodeK(node.child_i,\ K,\ D_{top})$ is invoked. If *node.child*$_i$ is a leaf node then $LeafNodeK(node.child_i,\ K,\ D_{top})$ is invoked.

Algorithm 7 lists a pseudocode of the function $LeafNodeK(node,\ K,\ D_{top})$, which selects top K significant data items from the data subsets referenced with the leaf *node*. It considers the same three cases as the function $NodeK(node,\ K,\ D_{top})$ listed in Algorithm 4. However, since the most significant data item is stored explicitily as *node.item*$_i$ for every data subset *node.data*$_i$, then the data items in subset *node.data*$_i$ are read only when they can contribute to the result data set D_{top}.

Example 9 For instance, consider the data set D of ten SNPs listed in Table 2.1 and the RDS tree $\mathfrak{T}$ containing the reduced data sets for the display resolutions in $\mathfrak{R} = \{\, 1 \times 1,\ 2 \times 2,\ 4 \times 2,\ 8 \times 6 \,\}$ and depicted in Figure 3.3. Every element *node.item*$_i$ in the tree node corresponds to the SNP with the lowest *pval* attribute between all SNPs associated with points in *node.area*$_i$. Assume we need to extract two SNPs from the node v_{22}, which has the lowest *pval* attribute. First, the node v_{22} is considered. The data items d_4 and d_{10} are ordered ascendingly according to their *pval* attributes. Since $0.0010 < 0.0050$, then $d_{10} < d_4$ and $D_{items} = \{d_{10},\ d_4\}$. Consequently, the node v_{34} is considered next. Again, items d_6 and d_{10} are ordered in ascending order with respect to the *pval* attribute and, therefore, $D_{items} = \{\, d_{10},\ d_6 \,\}$. As a result, two data items d_{10} and d_9 with the lowest *pval* attributes in the data subset $v_{34}.data_2 = \{\, d_8,\ d_9,\ d_{10} \,\}$ are appended to the result data set D_{top}. Further, since d_6 has greater *pval* attribute than d_9, then the data subset $v_{34}.data_1 = \{\, d_6,\ d_7 \,\}$ is not considered. Analogously, the leaf node v_{33} is omitted from the traversal, since the *pval* attribute of d_4 is greater than the *pval* attribute of d_9. Therefore, the tree traversal terminates with the final data set $D_{top} = \{\, d_{10},\ d_9 \,\}$. □

Algorithm 7: Top K Query Over Leaf Node

```
function LeafNodeK(node, K, $D_{top}$)
    $D_{items} \leftarrow order(node.items)$
    foreach $node.item_i \in D_{items}$ do
        if $size(D_{top}) \geq K$ then
            if $node.item_i \geq D_{top}[K]$ then
                return
            else
                foreach $item \in node.data_i$ do
                    if $item < D_{top}[K]$ then
                        $D_{top} \leftarrow D_{top} \setminus D_{top}[K]$
                        $D_{top} \leftarrow D_{top} \cup item$
                        $D_{top} \leftarrow order(D_{top})$
        else
            foreach $item \in node.data_i$ do
                if $size(D_{top} \geq K)$ then
                    if $item < D_{top}[K]$ then
                        $D_{top} \leftarrow D_{top} \setminus D_{top}[K]$
                        $D_{top} \leftarrow D_{top} \cup item$
                        $D_{top} \leftarrow order(D_{top})$
                else
                    $D_{top} \leftarrow D_{top} \cup item$
                    $D_{top} \leftarrow order(D_{top})$
```

Chapter 5

Experiments

This chapter provides the evaluation of the RDS tree applied on the generated data sets of SNPs. In particular, the efficiency of the tree construction, the selection query and the top K query are considered.

5.1 Setup and Data

For the experiments six data sets of SNPs were generated. The smallest data set contains 5 million of SNPs, while the largest data set contains 30 million of SNPs. For the simplicity, the unique chromosomal position of a SNP was represented with a single unique integer number. The hypothetical P value of a SNP was generated from the continuous uniform distribution $U(0,1)$, which corresponds to the null hypothesis in a GWA study.

The experiments were run on the machine with the following configuration:

- Dual core CPU 2.26 GHz
- 4 GB RAM
- Intel(R) Integrated Graphics Media Accelerator 4500MHD
- 1292 MB of maximum amount of graphics memory
- 32-bit Operating System

5.2 Tree Construction

In this section the implementation of a top-down incremental RDS tree construction is evaluated. The top-down incremental tree construction is performed in two steps:

1. In the first full data set scan, the extent of a data set in $\mathbb{R}^2$ Euclidean space is estimated.
2. In the second full data set scan, every data item is added to the RDS tree individually.

After the first step, the estimated extent of the data set and the predefined set $\mathfrak{R}$ of display resolutions completely describe the tessellation of $\mathbb{R}^2$ Euclidean space in every node of every level in the RDS tree. Therefore, in the second step, the leaf node and the corresponding data subset for storing the data item are located by recursively traversing the RDS tree from the root node and examining all possible space tiles in every node on the traversal path. If the space tile containing the point associated with the data item doesn't have the corresponding element in the tree node, then the new node element is created with respect to the node type. Otherwise, the existing node element is considered. If the element belongs to the inner node, then the data item is passed to the referenced child node and the tree traversal continues. Otherwise, the data item is appended to the referenced data subset.

The predefined set $\mathfrak{R}$ of display resolutions determines the height of the RDS tree and the maximum number of children in every tree node. The height of the RDS tree is equal to the number of resolutions in set $\mathfrak{R}$, while the maximum number of children in every tree node depends on the particular consecutive pairs of resolutions in set $\mathfrak{R}$. Thus, the cost of the data item insertion to the RDS tree doesn't depend on the data set size. Although, it completely depends on the predefined set $\mathfrak{R}$ of display resolutions. If set $\mathfrak{R}$ is such that $\mathbb{R}^2$ Euclidean space is always partitioned in two areas, then the RDS tree is analogous to the BSP tree [33] and the data item insertion operation takes $O(2|\mathfrak{R}|)$ time. Similarly, if set $\mathfrak{R}$ is such that $\mathbb{R}^2$ Euclidean space is always partitioned into four areas, then the RDS tree is analogous to the MX Quadtree [16, 35, 32, 22, 9, 33, 27, 36, 24] and the data item insertion operation takes $O(4|\mathfrak{R}|)$ time. In general, if $\mathfrak{R} = \{\, N_1 \times M_1, \ldots, N_k \times M_k\}$ is the set of display resolutions and $k \in \mathbb{N}^*$, then the data item insertion to the RDS tree takes $O(N_1M_1 + \sum_{i=1}^{k-1} \frac{N_{i+1}M_{i+1}}{N_iM_i})$ time. The maximum size of the RDS tree also depends on the predefined set $\mathfrak{R}$ of display resolutions and is equal to $O(1 + \sum_{i=1}^{k-1} N_iM_i)$.

We examined three nested sets of display resolutions $\mathfrak{R}_1 = \{\, 64 \times 48,\ 128 \times 96,\ 256 \times 192,\ 512 \times 384,\ 1024 \times 768 \,\}$, $\mathfrak{R}_2 = \{\, 16 \times 12,\ 32 \times 24 \,\} \cup \mathfrak{R}_1$ and $\mathfrak{R}_3 = \{\, 4 \times 3,\ 8 \times 6 \,\} \cup \mathfrak{R}_2$. For each of them the RDS tree was constructed using the implementation of the top-down incremental tree construction procedure. Figure 5.1 depicts the RDS tree construction time efficiency and Figure 5.2 depicts the size of the RDS tree for sets $\mathfrak{R}_1$, $\mathfrak{R}_2$ and $\mathfrak{R}_3$. The RDS tree construction time decreases significantly when the sequence of display resolutions in set $\mathfrak{R}$ begins with the relatively small resolution. At the same time, the difference in the number of nodes is not significant. Figure 5.3 separately depicts the time efficiency of the initial full data set scan and the time efficiency of the RDS tree construction for the set $\mathfrak{R}_3$ of display resolutions, which showed the best performance. Similarly, Figure 5.3 separately depicts the number of leaf nodes and the number of inner nodes in the RDS tree constructed for the set $\mathfrak{R}_3$ of display resolutions.

5.3 Selection Query

In the evaluation of the selection query the two outcomes were considered. The first is the number of retrieved data items for the current display resolution and the given minimum size δ of a point on the display. The second is the combined

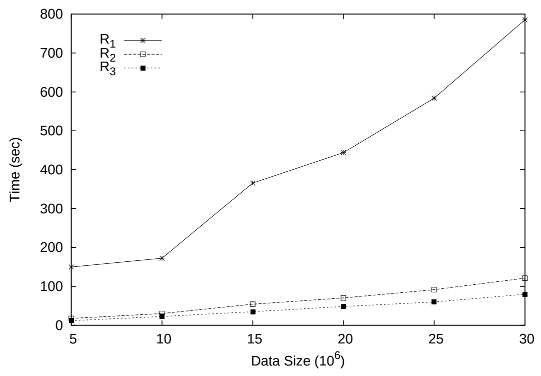


Figure 5.1: Time Efficiency of the RDS Tree Construction.

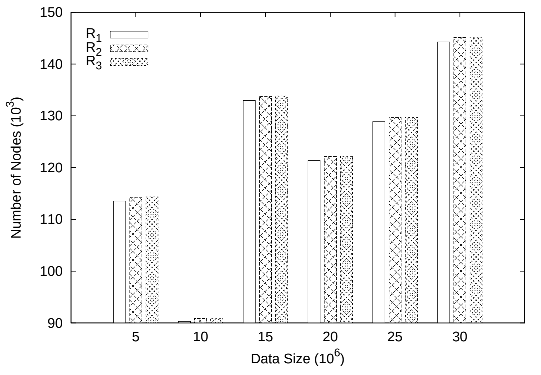


Figure 5.2: Size of the RDS Tree.

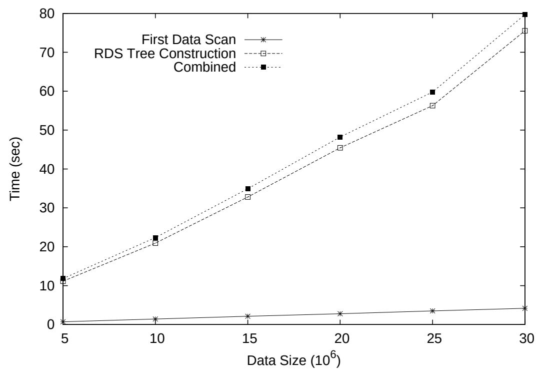


Figure 5.3: Time Efficiency of the RDS Tree Construction in Different Steps.

time efficiency of the data retrieval from the RDS tree and visualization.

The RDS tree was constructed for the set $\mathfrak{R} = \{\, 4 \times 3,\ 8 \times 6,\ 16 \times 12,\ 32 \times 24,\ 64 \times 48,\ 128 \times 96,\ 256 \times 192,\ 512 \times 384,\ 1024 \times 768 \,\}$ of display resolutions. The selection query was executed for the seven display resolutions. The four

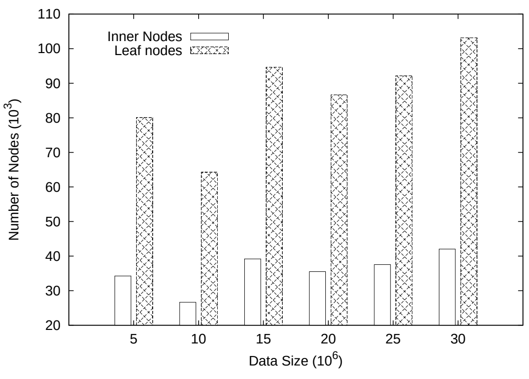


Figure 5.4: Number of Leaf and Inner Nodes in the RDS Tree.

display resolutions 128×96, 256×192, 512×384 and 1024×768 belong to set $\mathfrak{R}$ and have exactly corresponding nodes in the RDS tree. Another three display resolutions 224×144, 384×288 and 768×576 don't belong to set $\mathfrak{R}$ and don't have exactly corresponding nodes in the RDS tree. Figures 5.5, 5.6 and 5.7 depict the number of retrieved data items for these display resolutions, where the minimum size δ of a point on the display equals to 1, 3 and 5 respectively. The number of retrieved data items for the intermediate display resolutions 224×144, 384×288 and 768×576 exceeds the expected value. The most significant data reduction is achieved with the minimum size threshold $\delta = 5$. Generally, very small δ values shouldn't be considered in the interactive visualization, since they reduce the applicability of the interactive operations on single data items.

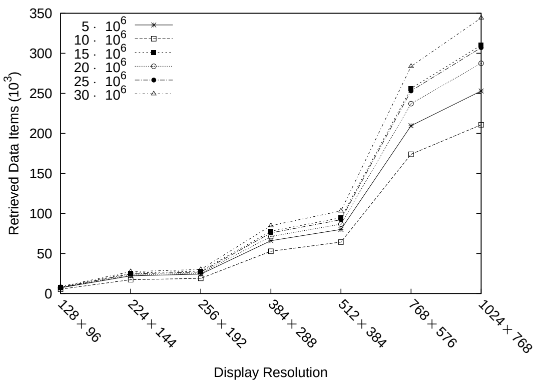


Figure 5.5: Number of Retrieved Data Items for the Minimum Size Threshold $\delta = 1$.

The time efficiency of the selection query was evaluated on the display resolution 1024×768 with the minimum size δ of a point on the display equal to 1. It is the most granular resolution in the constructed RDS tree, and with respect to the smallest minimum size threshold it requires to traverse all the leaf nodes. Additionally, we compare the time efficiency of the selection query against the naive standard visualization of the full data set, where all points

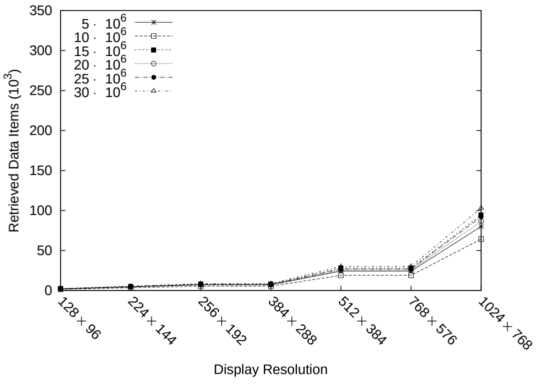


Figure 5.6: Number of Retrieved Data Items for the Minimum Size Threshold $\delta = 3$.

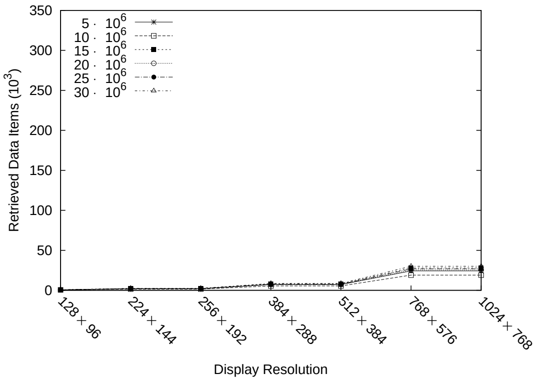


Figure 5.7: Number of Retrieved Data Items for the Minimum Size Threshold $\delta = 5$.

in $\mathbb{R}^2$ Euclidean space corresponding to all data items are visualized. Figure 5.8 depicts the results of the experiment. The execution time of the selection query together with the succeeding visualization of the retrieved data items doesn't vary significantly with the data set size. It remains almost constant and approximately equals to one fourth of a second. At the same time, the execution time of the naive standard visualization grows linearly with the data set size.

Figures 5.9 - 5.13 show the Manhattan plots visualized on the displays of resolutions 256×192, 384×288, 512×384, 768×576 and 1024×768. For each display resolution the size of a point on the display is 5. The visualizations on the left side of the figures use the full data set, while the visualizations on the right side of the figures use the reduced data sets. The reduced data sets were retrieved by the appropriate selection query with $\delta = 5$ from the RDS tree, which was constructed with respect to the set $\mathfrak{R}$ of display resolutions over the data set containing 5 million SNPs. As a result, we observe no significant difference in the visualizations of the full and reduced sets of SNPs.

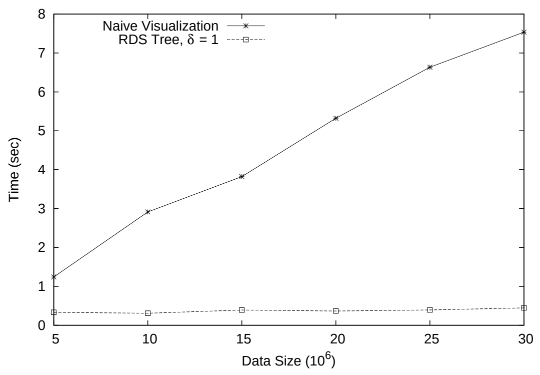
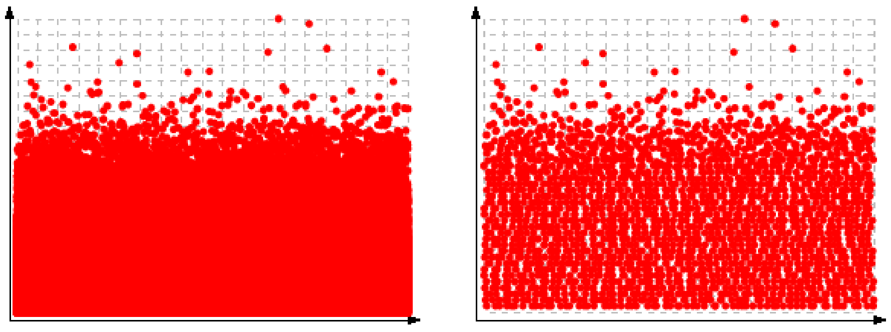
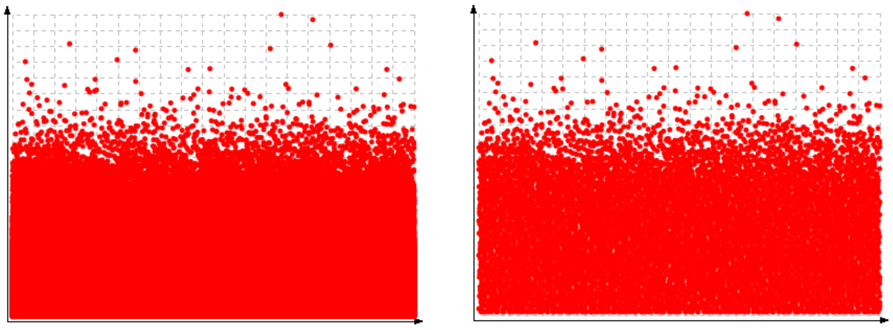


Figure 5.8: Time Efficiency of the Selection Query for the Display Resolution 1024 × 768.



(a) Full Data Set (b) Reduced Data Set

Figure 5.9: Manhattan plot on Display of Resolution 256 × 192.

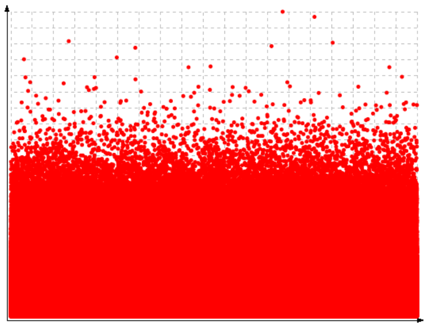


(a) Full Data Set (b) Reduced Data Set

Figure 5.10: Manhattan plot on Display of Resolution 384 × 288.

5.4 Top K Query

The top K query can be executed starting with any node in the RDS tree. Usually, the starting node corresponds to the data item, which was visualized and selected by an analyst in some step of data investigation. In our experiment we execute


(a) Full Data Set

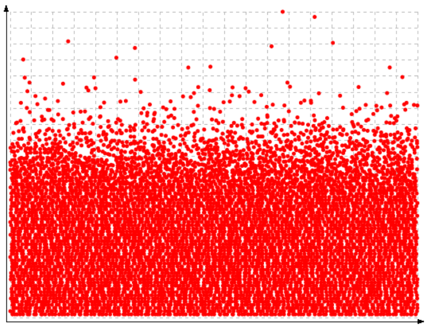
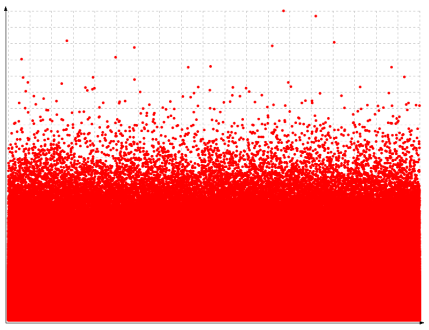

(b) Reduced Data Set

Figure 5.11: Manhattan plot on Display of Resolution 512 × 384.


(a) Full Data Set

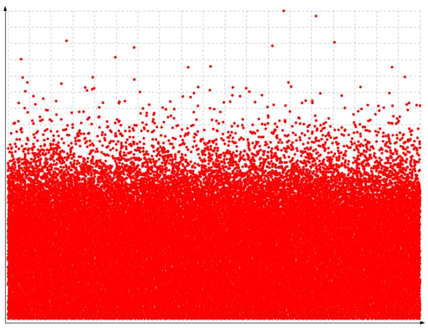
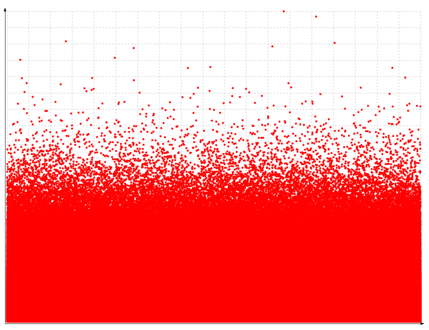

(b) Reduced Data Set

Figure 5.12: Manhattan plot on Display of Resolution 768 × 576.


(a) Full Data Set

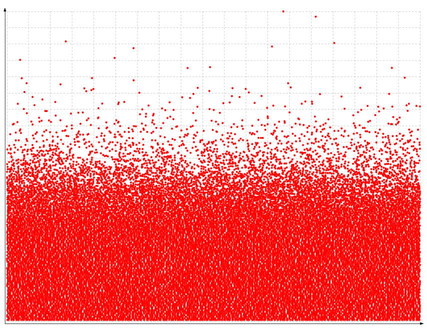

(b) Reduced Data Set

Figure 5.13: Manhattan plot on Display of Resolution 1024 × 768.

the top K query starting with the root node. The root node references the whole data set and, therefore, in the worst scenario the query will visit every leaf node. Since, the top K query corresponds to the details-on-demand operation, then we consider K varying from 10 to 1000. Similarly to the selection query, the top K query was evaluated on the RDS tree constructed for the set $\mathfrak{R} = \{\, 4\times 3,\ 8\times 6,\ 16\times 12,\ 32\times 24,\ 64\times 48,\ 128\times 96,\ 256\times 192,\ 512\times 384,\ 1024\times 768\,\}$ of display resolutions. Figure 5.14 depicts the time efficiency of the top K query.

The experiment on the data set containing 10 million of SNPs emphasizes, that the data set influences the execution time of the query. However, it still remains under the half of a second and for $K \leq 100$ it is less than 0.05 of a second.

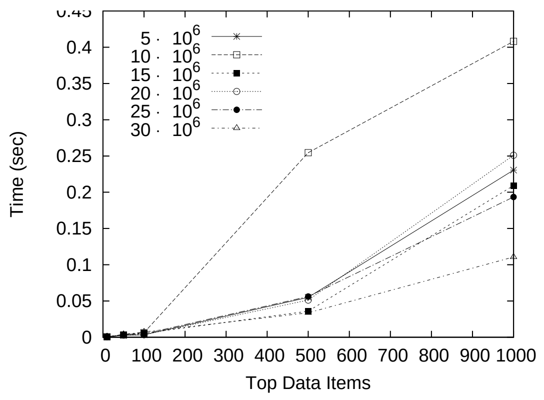


Figure 5.14: Time Efficiency of the Top K Query.

5.5 Summary of Experimental Results

As previously discussed (see Section 1.2.2), for any instantaneous interactive query, the results should be produced and displayed in no more than 100 msec. While the visualization of the whole data set should be completed within 1 sec. This is the gold standard that any interactive visualization should follow. Our experimental results with synthetic GWA study data confirm that the RDS tree provides the sufficient efficiency to support the interactivity. Indeed, the selection query for the visualization of the whole data executes in approximately one fourth of a second. Furthermore, the execution time remains constant with the increase of the data set size. The execution time of top K instantaneous interactive query is less than 100 msec with $K < 200$. This is completely sufficient, since in the common scenario K is equal to 10 or 20 data items, and rarely if ever can exceed 100 data items.

Chapter 6

Related Work

The high degree of overplotting and the required support of rapid display updates are universally recognized as the two main challenges in the visualization of massive data sets [8, 10, 21, 42, 5, 20, 19, 13, 38, 18, 14]. Fekete et al. [10] evaluated the recent hardware-based techniques for the efficient interactive visualization of large data sets and showed sustainable performance for approximately one million of data items. Considering the improving technologies, nowadays we can expect to have sustainable performance with several million of data items. Although, the focus of this thesis is the data sets containing tens of millions of data items. Eick et al. [8] and Keim et al. [19] listed filtering, aggregation, compression, principle component analysis and other data reduction techniques as the possible techniques for data reduction and, therefore, for the solution of the overplotting problem. Shneiderman [38] emphasized the aggregate visualization as the most promising for keeping display complexity low. In the aggregate visualization, the individual markers representing individual data records are organized into the aggregate markers using various clustering strategies. Then, only the aggregate markers, which can represent thousands of records, are visualized. In this chapter we provide an overview of the recent aggregate visualization techniques.

6.1 Data Set Partitioning

The exact aggregation technique depends on the type of data and on the visualization technique. The most common practice is to partition the data set into the predefined number of bins and then to assign aggregated values to every bin. For example, Hao et al. [14] propose the variable binned scatter plots for the visualization of large amount of data without overplotting. The bin size is variable and is computed from the data density. The similar solution is proposed in [13] for the large time-series data, where the display space is allocated in proportion to the degree of interest of data subintervals. However, both solutions change the conventional metaphors of the visualizations and are not scalable to the millions of data items.

6.2 Interactive Hierarchical Displays

Yang et al. [42, 11] introduced a general framework for visualization and exploration of large multivariate data sets, which is called *interactive hierarchical displays* (IHDs). The main goal of the framework is to overcome the overplotting problem. It can be applied to a wide range of existing visualization techniques like parallel coordinates, star glyphs, scatterplot matrices and dimensional stacking. The IHDs use the hierarchical cluster tree constructed upon a data set with a clustering algorithm, such as BIRCH [44]. Then, the subsets of clusters in the hierarchical cluster tree are visualized instead of the whole data set. The user controls the level of details by specifying the minimum cluster size threshold. Only the clusters with the size greater than the specified threshold are visualized. Each cluster is visualized using a technique called *the meanpoint band*, which conveys several essential features of the cluster, such as the mean and extent. The hierarchical relationships are also depicted using color and, therefore, sibling and parent relations are readily observed. As a result, every node in the hierarchical cluster tree represents a cluster and stores the information about the size, population, mean, minimum and maximum bounds of the cluster. Additionally, the IHDs support the zoom in operation through the combination of the brushing and the drill down operations. The meanpoint band technique partially allows the IHDs to preserve the conventional metaphors of the visualizations. However, the IHDs completely depends on the data set and user explicitly needs to specify the level of details to reduce the overplotting. Also, the authors don't provide any interactive query equivalent to the top K query in the RDS tree.

6.3 The MRO tree

Keim et al. [20] propose the MRO tree of multi-resolution objects. The tree represents the hierarchy of clusters constructed with respect to the relevance of each data item defined with the predefined relevance function. The most relevant data items are stored in the top of the MRO tree, while the least relevant data items are stored in the bottom of the MRO tree. To visualize the data set, data items from the tree are selected so that the number and relevance of the selected items is maximized, depending on the given display space. Therefore, in contrast to the IHDs and similar to the RDS tree, the selection of the data items in the MRO tree depends on the availabe display area. However, in contrast to the RDS tree, the construction and characteristics of the MRO tree depend only on the data. Also, authors don't provide the zoom in query or any other instantaneous query for the MRO tree.

Chapter 7

Conclusions And Future Work

In this thesis we presented the space partitioning RDS tree, which indexes the data to support the interactivity and to decrease the overplotting in the data visualization. It is constructed only once and then can be used in visualizations on the displays of different resolutions. The construction of the RDS tree requires two full scans of data set and is relatively efficient. The queries over the RDS tree consider the current display resolution and retrieve such data items, that the overplotting is decreased. Moreover, the execution time of the queries doesn't exceed the required thresholds for the interactive data visualization. The RDS tree was applied to support the interactivity in the Manhattan plot, that is one of the most widely used tools for visual representation of the observed associations of SNPs with a trait in GWA studies. The evaluation of the RDS tree with synthetic GWA study data confirmed its efficiency in the framework of the Manhattan plot.

The following summarizes the contributions of this thesis:

- We propose the data reduction based on the defined screen resolution such, that the visualization of the reduced data doesn't suffer from overplotting and is identical to the visualization of the full data.
- We present the novel index structure, termed RDS tree, which indexes data with respect to the hierarchy of reduced data constructed for the predefined list of screen resolutions.
- We introduce and provide algorithms for three queries over the RDS tree, which support the basic interactive tasks: (i) selection query – ensures the fast data retrieval for the overview task and reduces the overplotting; (ii) window query – ensures the fast data retrieval for the visualization of the data in the particular region and reduces the overplotting; (iii) top K query – implements the details-on-demand task, that efficiently retrieves top K data items occluded in the visualization.
- The evaluation with synthetic GWA study data confirms the high efficiency of the RDS tree with tens of millions of SNPs, which ensures short response times.

- The evaluation with synthetic GWA study data confirms the high efficiency of the RDS tree with tens of millions of SNPs, which ensures short response times.

For the future work we consider the following directions. First, implementation of the hard disk based version of the RDS tree. Second, performing the user based evaluation of the interactive visualizations supported by the RDS tree. Third, application of the RDS tree for other plots like quantile-quantile plots or genotype cluster plots. Finally, development of strategies for the RDS tree construction and selection of the predefined display resolutions.

Bibliography

[1] Wolfgang Aigner, Alessio Bertone, and Silvia Miksch. Tutorial: Introduction to Visual Analytics. In Andreas Holzinger, editor, *HCI and Usability for Medicine and Health Care*, volume 4799 of *Lecture Notes in Computer Science*, pages 453–456. Springer Berlin / Heidelberg, 2007.

[2] John Attia, John P. A. Ioannidis, Ammarin Thakkinstian, Mark McEvoy, Rodney J. Scott, Cosetta Minelli, John Thompson, Claire Infante-Rivard, and Gordon Guyatt. How to use an article about genetic association: A: Background concepts. *JAMA*, 301(1):74–81, 2009.

[3] John Attia, John P. A. Ioannidis, Ammarin Thakkinstian, Mark McEvoy, Rodney J. Scott, Cosetta Minelli, John Thompson, Claire Infante-Rivard, and Gordon Guyatt. How to use an article about genetic association: B: Are the results of the study valid? *JAMA*, 301(2):191–197, 2009.

[4] Paul R. Burton, Martin D. Tobin, and John L. Hopper. Key concepts in genetic epidemiology. *The Lancet*, 366(9489):941–951, 2005.

[5] Chaomei Chen. Top 10 Unsolved Information Visualization Problems. *IEEE Computer Graphics and Applications*, 25(4):12–16, 2005.

[6] Psychiatric GWAS Consortium Coordinating Committee. Genomewide Association Studies: History, Rationale, and Prospects for Psychiatric Disorders. *The American journal of psychiatry*, 166(5):540–556, 2009.

[7] Paul I.W. de Bakker, Manuel A.R. Ferreira, Xiaoming Jia, Benjamin M. Neale, Soumya Raychaudhuri, and Benjamin F. Voight. Practical aspects of imputation-driven meta-analysis of genome-wide association studies. *Human Molecular Genetics*, 17(R2):R122–R128, 2008.

[8] Stephen G. Eick and Alan F. Karr. Visual Scalability. *Journal of Computational & Graphical Statistics*, 11:22–43, 2000.

[9] David Eppstein, Michael T. Goodrich, and Jonathan Z. Sun. The Skip Quadtree: A Simple Dynamic Data Structure for Multidimensional Data. *In Proc. 21st ACM Symposium on Computational Geometry*, pages 296–305, 2005.

[10] Jean-Daniel Fekete and Catherine Plaisant. Interactive Information Visualization of a Million Items. *Proceedings of the 2002 IEEE Symposium on Information Visualization*, page 117, 2002.

[11] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Navigating Hierarchies with Structure-Based Brushes. *1999 IEEE Symposium on Information Visualization*, page 58, 1999.

[12] Anthony J. F. Griffiths, Susan R. Wessler, Richard C. Lewontin, William M. Gelbart, David T. Suzuki, and Jeffrey H. Miller. *An Introduction to Genetic Analysis*. W. H. Freeman, 8th edition, 2004.

[13] Ming C. Hao, Umeshwar Dayal, Daniel A. Keim, and Tobias Schreck. Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data. *Eurographics/IEEE VGTC Symposium on Visualization*, pages 27–34, 2007.

[14] Ming C. Hao, Umeshwar Dayal, Ratnesh K. Sharma, Daniel A. Keim, and Halldr Janetzko. Visual Analytics of Large Multi-Dimensional Data Using Variable Binned Scatter Plots. *Proceedings of the SPIE (The International Society for Optical Engineering)*, 7530, 2010.

[15] John Hardy and Andrew Singleton. Genomewide Association Studies and Human Disease. *The New England Journal of Medicine*, 360(17):1759–1768, 2009.

[16] Gregory M. Hunter and Kenneth Steiglitz. Operations on Images Using Quad Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):145–153, April 1979.

[17] National Human Genome Research Institute. Talking Glossary of Genetic Terms. http://www.genome.gov/glossary/, September 2010.

[18] Daniel A. Keim, Ming C. Hao, Umeshwar Dayal, Halldor Janetzko, and Peter Bak. Generalized scatter plots. *Information Visualization*, 2009.

[19] Daniel A. Keim, Florian Mansmann, Jrn Schneidewind, and Hartmut Ziegler. Challenges in Visual Data Analysis. *In Proceedings of the Tenth International Conference on Information Visualization*, pages 9–16, 2006.

[20] Daniel A. Keim and Jörn Schneidewind. Scalable Visual Data Exploration of Large Data Sets via MultiResolution. *Journal of Universal Computer Science*, 11(11):1766–1779, 2005.

[21] Robert Kosara, Helwig Hauser, and Donna L Gresh. An Interaction View on Information Visualization. *Proceedings of EUROGRAPHICS 2003*, 2003.

[22] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and R-tree Indexes in Oracle Spatial: A Comparison using GIS Data. *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557, 2002.

[23] Yun Li, Cristen Willer, Serena Sanna, and Gonalo Abecasis. Genotype Imputation. *Annual Review of Genomics and Human Genetics*, 10:387–406, 2009.

[24] Ling Liu and M. Tamer Özsu. *Encyclopedia of Database Systems.* Springer, 2009.

[25] Harvey Lodish, Arnold Berk, Paul Matsudaira, Chris A. Kaiser, Monty Krieger, Matthew P. Scott, Lawrence Zipursky, and James Darnell. *Molecular Cell Biology.* W. H. Freeman, 5th edition, 2003.

[26] Mark I. McCarthy, Gonalo R. Abecasis, Lon R. Cardon, David B. Goldstein, Julian Little, John P. A. Ioannidis, and Joel N. Hirschhorn. Genome-wide association studies for complex traits: consensus, uncertainty and challenges. *Nature Reviews Genetics*, 9:356–369, 2008.

[27] Hossein Mirzaee and Farhad Besharati. Tree Based Decomposition of Sunspot Images. *World Academy of Science, Engineering and Technology*, 27:139–143, March 2007.

[28] Jason H. Moore, Folkert W. Asselbergs, and Scott M. Williams. Bioinformatics challenges for genome-wide association studies. *Bioinformatics*, 26(4):445–455, 2010.

[29] Cristian Pattaro, Alessandro De Grandi, Veronique Vitart, and et al. A meta-analysis of genome-wide data from five European isolates reveals an association of COL22A1, SYT1, and GABRR2 with serum creatinine level. *BMC Medical Genetics*, 11:41, 2010.

[30] Thomas A. Pearson and Teri A. Manolio. How to Interpret a Genome-wide Association Study. *JAMA*, 299(11):1335–1344, 2009.

[31] Kai Puolamäki and Alessio Bertone. Introduction to the Special Issue on Visual Analytics and Knowledge Discovery. *SIGKDD Explorations Newsletter*, 11(2):3–4, 2009.

[32] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[33] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, 2006.

[34] Arne Schillert, Daniel F. Schwarz, Maren Vens, Silke Szymczak, Inke R. Knig, and Andreas Ziegler. ACPA: automated cluster plot analysis of genotype data. *BMC Proceedings*, 3(Suppl 7):S58, 2009.

[35] Clifford A. Shaffer and Hauan Samet. Optimal quadtree construction algorithms. *Computer Vision, Graphics, and Image Processing*, 37(3):402–419, March 1987.

[36] Shashi Shekhar and Hui Xiong. *Encyclopedia of GIS.* Springer, 2007.

[37] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[38] Ben Shneiderman. Extreme Visualization: Squeezing a Billion Records into a Million Pixels. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 3–12, 2008.

[39] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* National Visualization and Analytics Ctr, 2005.

[40] William Y. S. Wang, Bryan J. Barratt, David G. Clayton, and John A. Todd. Genome-wide association studies: theoretical and practical concerns. *Nature Reviews Genetics*, 6:109–118, 2005.

[41] Graham J. Wills. Selection: 524,288 Ways to Say "This is Interesting". *Proceedings of the 1996 IEEE Symposium on Information Visualization*, 1996.

[42] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. Interactive Hierarchical Displays: A General Framework for Visualization and Exploration of Large Multivariate Data Sets. *Computers & Graphics*, 27(2):265–283, 2003.

[43] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond Visual Acuity: The Perceptual Scalability of Information Visualizations for Large Displays. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 101–110, 2007.

[44] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 25(2):103–114, 1996.

[45] Andreas Ziegler, Inke R. Knig, and John R. Thompson. Biostatistical aspects of genome-wide association studies. *Biometrical Journal*, 50(1):8–28, 2008.