

# **Telescope: online analysis pipeline for high-density tiling microarray data**

Zhengdong D. Zhang<sup>1</sup>, Joel Rozowsky<sup>1</sup>, Hugo Y.K. Lam<sup>2</sup>, Jiang Du<sup>3</sup>,  
Michael Snyder<sup>4</sup>, Mark Gerstein<sup>1,3,§</sup>

<sup>1</sup> Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT 06520, USA; <sup>2</sup> Interdepartmental Program in Computational Biology and Bioinformatics, Yale University, New Haven, CT 06520, USA; <sup>3</sup> Department of Computer Science, Yale University, New Haven, CT 06520, USA; <sup>4</sup> Department of Molecular, Cellular and Developmental Biology, Yale University, New Haven, CT 06520, USA;

<sup>§</sup> Corresponding author (E-mail: [zdzmg@bioinfo.mbb.yale.edu](mailto:zdzmg@bioinfo.mbb.yale.edu))

E-mail addresses:

Zhengdong D. Zhang: [zhengdong.zhang@yale.edu](mailto:zhengdong.zhang@yale.edu)  
Joel Rozowsky: [joel.rozowsky@yale.edu](mailto:joel.rozowsky@yale.edu)  
Hugo Y.K. Lam: [hugo.yk.lam@yale.edu](mailto:hugo.yk.lam@yale.edu)  
Jiang Du: [jiang.du@yale.edu](mailto:jiang.du@yale.edu)  
Michael Snyder: [michael.snyder@yale.edu](mailto:michael.snyder@yale.edu)  
Mark Gerstein: [mark.gerstein@yale.edu](mailto:mark.gerstein@yale.edu)

Running title: microarray data analysis pipeline

Key words: high-density tiling microarray,  
high-density oligonucleotide microarray,  
microarray data analysis

For test data sets, sample result web pages, and annotation of a result web page, please visit

<http://telescope.gersteinlab.org/sample.html>

To test Telescope, first please read the direction outlined at

<http://telescope.gersteinlab.org/test.html>

## **Abstract**

Newly emerged tiling array technology can cover entire genomes in a systematic manner with unprecedented resolution. As this new technology is fast gaining popularity, however, a streamlined data processing software system for such high-density tiling array experiments is largely underdeveloped. Existing data processing software for traditional microarrays cannot be used since the considerably larger size and different nature of tiling array data require a new analysis approach. Responding to this need, we developed Telescope, a fully integrated data processing pipeline for analyzing tiling array data (<http://telescope.gersteinlab.org>). In a completely automated fashion, it will normalize signals between channels and across arrays, combine replicate experiments, score each array element, and identify genomic features. At major data processing steps, such as normalization and feature identification, several methods are available to be chosen from. Given the modular architecture of the pipeline, new analysis algorithms can be readily incorporated as they are developed. Telescope is capable of handling very large data sets, such as ones generated by whole genome ChIP-chip experiments, as we developed a new data compression algorithm to efficiently reduce the data size for fast online data transmission, and the modularity of the pipeline also makes parallel computing and multiprocessing possible. Telescope is designed with a graphic user-friendly interface to facilitate a user's data analysis task, and the results, presented in an organized manner on a web page, can be downloaded for further analysis.

## **Rationale**

Microarray technology is now more accessible than ever before. Thanks to its unrivaled capability to carry out a very large number of parallel quantitative measurements, since its emergence in the early 1990s [1, 2], this technology has been widely applied to systematic studies of various biological phenomena, ranging from differential gene expression, to DNA copy number polymorphism, and to transcription factor binding.

Traditional microarrays, constructed by mechanically depositing or printing polymerase chain reaction (PCR) products typically of ~1 Kb in length in a dense matrix on a glass slide, have been successfully used in numerous studies and have become prevalent in the research field. Many computer programs and software tools, including free software packages, such as ExpressYourself [3] or MIDAS [4], are available to process and analyze the data sets generated in such studies. However, limited by its manufacturing methodology, traditional microarrays are not amenable for systematic coverage of large genomes or even some large genomic regions. To fully realize the parallel-measurement potential of microarray technology, the current trend is to present large genomic regions (e.g., ENCODE regions or a complete human chromosome) or even an entire genome on one or several microarrays in an unbiased fashion by using oligonucleotides (a.k.a. tiles) uniformly sampled from presented genomic sequences. Recent technology breakthroughs [5, 6] made it possible for such oligonucleotides, typically of 25-60

bp in length, to be chemically synthesized directly on the microarray slides in a very high density (up to 6.6million elements in less than 2 cm<sup>2</sup>). Such oligonucleotide tiling microarrays, which give unprecedented genomic coverage and resolution, can be used for genomic studies of gene expression [7-10], chromatin immuno-precipitation (ChIP-chip) [11], copy number variation [12], histone modification [13], and chromatin DNaseI sensitivity [14].

Like for any other nascent technologies, ready-to-use data analysis software packages for tiling array experiments are hard to find. Existing data processing software for traditional microarrays cannot be used since the considerably larger size and different nature of tiling array data require a new analysis approach [15]. Recently, a model-based method for tiling array ChIP-chip data analysis has been proposed [16]. Two other methods, based on curve fitting [17] and multi-channel combination [18] respectively, have also been developed for tiling array transcription data analysis. The excellent open-source Bioconductor software project [19] provides many sophisticated statistical methods written in R for microarray data analysis. However, as a software toolbox and a programming environment, it is rather difficult for non-programmers to use.

Here we present Telescope, an automated data processing pipeline for analyzing data sets generated in experiments using high-density tiling microarrays. Suitable microarray data processing methods, either previously published elsewhere or newly developed, were implemented and made available conveniently in a single online software pipeline. It has a user-friendly interface and is freely accessible over the Web. The software performs data normalization, combination of replicate experiments, tile scoring, and feature identification. We demonstrate the modular nature of the pipeline design by showing how different methods can be plugged in—at major data processing steps, such as normalization and feature identification, several methods are available to be chosen from depending on the nature of the data and the user’s data-analysis goal. The program can process gene expression and ChIP-chip tiling microarray data. The results, presented in a clear, well organized manner, can be downloaded for further analysis.

## **System implementation and user interface**

Telescope was entirely developed in Java. Java was chosen as the programming language because of its build-in threading capability and its excellent library support for graphic user interface and networking development. More importantly, it was chosen because of its object-oriented nature: the program code is organized into different coherent classes and thus it naturally modularizes the system, which greatly facilitates parallel system development and subsequent system updating, a *desideratum* for any software engineering project of non-trivial complexity.

As a web-accessible program system, Telescope is composed of three connected components: an applet, a servlet, and a pipeline program. The applet is the graphical interface through which the user interacts with Telescope. It is automatically downloaded and launched inside a Java-enabled web browser whenever the pipeline web page is browsed. Through the Telescope applet, a user can upload array data files to the pipeline server, select appropriate pipeline parameters and methods, run the data processing program, and view or download analysis results. The applet, however, cannot run the pipeline program directly. Instead, it makes data processing requests to the servlet, a server program that acts as the proxy of the pipeline program on the Web and communicates with the applet upon requests. The servlet, the central layer of Telescope, runs two ‘daemon’ threads in the background to handle—i.e., accept and schedule or reject based on the current system load—file upload or data processing requests, prepare the pipeline running environment, and initiate with user-specified parameters the back-end pipeline program, which carries out the heavy lifting—the actual data processing procedure. This modular design—the separation between the request handling and the data processing itself—enables the usage of a computer farm for parallel computing and multiple concurrent processing.

On the web form of the Telescope applet (Figure 2A), a user can either upload a parameter file, if available from a previous use of Telescope, to have all parameters set accordingly in one easy step or set parameters one by one manually, which is more likely to happen if an array data set is to be analyzed for the first time. The main body of the form was organized into two panels, one for setting the tile scoring parameters and the other for selecting the feature identification method, reflecting two main stages of data processing in the pipeline. After the pipeline program is started on the server, the users can monitor its progress through pipeline messages, which are constantly updated by the server throughout each pipeline run.

When data processing is done, a web page with analysis results will be presented to the user in a new browser window (Figure 2B). On the result web page, the parameters and methods that were used to analyze the data are summarized at the top, followed by log-intensity scatter plots for each array and log-intensity histograms for all arrays in the data set before and after normalization. These enlargeable plots enable the user to quickly identify any problematic arrays visually and subsequently exclude them from further consideration. Both tile maps with log-ratio and P-value annotations and the feature list in various text formats can be downloaded for further processing and analysis. The feature list in regular tab-delimited text format gives the user the chromosome (or other genomic sequence ID), the genomic start and end coordinates, the log-ratio, the P-value, and, if the tiled genome is specified, the upstream and downstream genes of each feature. If it is the human genome that is under investigation, Telescope will also provide links to display identified features on custom tracks in the UCSC genome browser. Moreover, if the tiling array was designed from a previous human genome build (e.g. hg16, NCBI 34), Telescope will also provide an additional feature list with the coordinates lifted over to the current human genome build (e.g. hg17, NCBI 35).

## Data processing in Telescope

Telescope processes the data in a sequential fashion using the major steps shown in Figure 1A. These steps can be approximately grouped into three stages: data input, tile scoring, and feature identification. Here, we use the data set from a ChIP-chip experiment of the transcription factor STAT1 to demonstrate how high-density tiling microarray data are processed by Telescope. We compared features of Telescope and several other programs that are explicitly applicable to high-density tiling microarray data, and the result is tallied in Table 1.

### *Data input*

The data input to Telescope reside in tab-delimited text files generated by image analysis software. Currently, Telescope recognizes data files in Affymetrix, Pair [20] and GFF [20, 21] formats. Whenever available, GFF format is always recommended since it is a more standardized format and thus less problematic for processing. Although the aforementioned formats are not fully inter-compatible, they all provide the essential data, namely the chromosomes (or other genomic identifiers, such as Contig IDs and ENCODE region IDs), the genomic coordinate, and the fluorescence intensity (Telescope automatically detects the base-two logarithm of intensity) for each array element.

### *Data format optimization and standardization*

The tab-delimited text files generated by different image analysis software are exceptionally large in size and can have different ASCII formats. This raises several issues. First, large files require large disk storage space, long transmission time, and long file access time. Second, the result of a single microarray experiment could be specified by several files, such as both the design and the data files, which can be problematic to handle and organize. Third, various file formats require different programs to parse the data. Although currently several open specifications, like MIAME and MAGE-ML, are available to standardize the array data format, these standards usually neglect the data size issue and require users to provide information more than necessary and thus are cumbersome for regular data analyses.

Not only did we implement Telescope to support various formats of the input data file—the GFF format, the PAIR format, the NimbleGen format (POS + GFF/PAIR), and the Affymetrix format (BPMAP + CEL), but also developed a new algorithm (Supplementary Table 1-3) that can reduce the physical data file size, handle the data set in an organized manner, and enhance the performance of Telescope. In order to reduce the data size, we store the data values as their primitive types rather than in ASCII format. For example, an integer ‘147,971,601’ will occupy 9 bytes in an 8-bit ASCII format but only 4 bytes in our format.

Another example is the nucleotide sequence. A sequence such as “TCGAGGCCTTAAGCTCTTGAGAGGT” will occupy 25 bytes in ASCII but only 7 bytes in our format since we use two bits to represent each base. After optimizing the data file representation, we compress all the files, including the meta-files storing the information describing those data files, into one single archive file.

Our bit optimization approach not only achieves significant file size reduction but also enables us to efficiently locate by random access a particular record for a certain set of coordinates when necessary. This is because we store every entry in a fixed number of bits and sort all the entries by their coordinates, so that the offset of the record in the byte stream can be easily calculated given that all the data are in contiguous coordinates. Moreover, an even higher compression rate can be obtained if a ‘DEFLATE’ algorithm [22] is applied after the array data is optimized than it is not (Supplementary Table 3). This is especially advantageous for file transfer over the network.

Since our algorithm allows us to select which data columns in the text files to be included, it omits any unnecessary information in the original data set and consequently further reduces the size of the file to be transmitted. This capability, together with the aforementioned improvements, can reduce the size of the original data set by up to 90%. This size reduction greatly reduces the amount of time required to transmit the data from the client computer to our pipeline server over the Internet. To ensure the consistency of the data file processing, we convert the archived data files into the standardized GFF format on the pipeline server. As a result, the data files can always be processed in a consistent manner.

### *Data normalization*

Unlike printed PCR arrays, the array elements (oligonucleotides) of a tiling microarray are directly synthesized on the array slide. Direct *in situ* synthesis creates morphologically uniform array elements, which to a large degree obviates the need for spot filtering, an imperative procedure for PCR microarray data analysis. Moreover, direct oligonucleotide synthesis makes it possible to have a very large number of spots in a small area (thus high-density). Miniature slide design of the tiling microarrays allows more uniform hybridization and thus greatly reduces the spatial heterogeneity in the probing conditions across the slide, a potentially severe problem suffered by PCR microarrays.

For each array in an experimental set, the relative contributions of the test and reference signals are compared. Ideally, if nucleic acid probes have equal concentration in the test and reference samples, the signals of the two dyes should be approximately equal (i.e., the ratio of the two signals should be close to one for probes hybridizing to an equal degree in both fluorescence channels). In practice, the signals can be rather different due to different chemical properties of dyes and nonspecific or incomplete hybridization to the array. Normalization is used to

compensate for these effects by—depending on what method is being used—either applying a scale factor to equalize signals from probes with unchanged concentration or imposing the same empirical distribution of signal intensities. We put together and implemented standard statistical methods that were described in various literature sources and made them conveniently available for tiling array data analysis. At present Telescope can normalize tiling array data by mean/median, loess, or quantile normalization (Figure 1B). These methods have also been implemented elsewhere, most notably in Bioconductor R packages. The mean/median and the loess normalization methods are both available in the "marray" package. The 'affy' package contains another implementation of the loess and the quantile normalization methods. In addition, other publicly available software, such as TM4 [4] and TAS/GTRANS, provides some similar functionalities for array data normalization. These methods are summarized below with appropriate references.

Mean/median normalization. Normalization by mean or median [3], the so-called 'constant majority' methods, is based on the assumption that the majority of genes do not change their expression level in response to the experimental perturbation [23]. It is carried out by subtracting the mean or median of the base-two logarithm of the ratio of test to reference signal intensities from the log-ratio value of each tile on a single array. This procedure transforms the log-ratio distribution by centering it at zero. The mean is the maximum likelihood estimator of the long-term trend in the data while the median is more robust to the outliers in the data. In theory, they are different measures of the location of a distribution. In practice, however, because the mean and the median of the log-intensities from the probes on each array are often very close to each other, these two methods usually give very similar results. The advantages of these two methods include the easiness of their implementation and their robustness to the violation of the assumption—they remain applicable even in cases where up to 50% of probes have altered concentrations.

Loess normalization. Loess normalization [3, 24, 25] normalizes array data between channels and removes the intensity-specific artifacts in the log-ratio measurements simultaneously. Like normalization by mean or median, loess normalization is also performed on an array-by-array basis. For each array, Telescope first uniformly samples 50,000 log-ratio values from the original data, and then performs the locally weighted regression on the sampled data. The dependency of the log-ratio on the intensity is removed by subtracting predicted log-ratio based on the loess regression from the actual log-ratio, and the new test and reference log-intensities after normalization are recovered from the residuals. The main disadvantage of Loess normalization is that the locally weighted regression is computationally intensive, and thus the necessity of using sampled data instead of the original, much larger data set. Since loess normalization is carried out for each array one by one, even after data sampling, it remains expensive to use.

Quantile normalization. Unlike the normalization methods discussed above, quantile normalization [25] not only normalizes data between channels and across arrays simultaneously but also removes the dependency of the log-ratio on the intensity in one step. It imposes the same empirical distribution of intensities to each channel of every array. To achieve this,

Tilescope first creates an  $n \times 2p$  (log) intensity matrix  $\mathbf{M}$ , where  $n$  is the number of tiles on an array and  $p$  is the number of arrays in an experimental data set, and then sorts each column of  $\mathbf{M}$  separately to give  $\mathbf{M}_s$ . Afterwards, it takes the mean across rows of  $\mathbf{M}_s$  and creates  $\mathbf{M}_s'$ , a matrix of the same dimension as  $\mathbf{M}$ , but where all values in each row are equal to the row means of  $\mathbf{M}_s$ . Finally, Tilescope produces the quantile-normalized (log) intensity matrix  $\mathbf{M}_n$  by rearranging each column of  $\mathbf{M}_s'$  to have the same ordering as the corresponding column of  $\mathbf{M}$ . Quantile normalization is fast and has been demonstrated to outperform other normalization methods [25]. Thus it is the default normalization method used by Tilescope.

### *Tile scoring*

Some arrays are designed to tile genomic sequences of both strands and most array experiments are conducted in replicate. To facilitate subsequent data processing, Tilescope pools the normalized log-ratios of all tiles on every array into a matrix and sorts them based on the tiles' genomic locations regardless of which strand they come from. At the tile scoring step, the program identifies tiles that exhibit differential hybridization. Depending on the nature of the experiment, these tiles ultimately correspond to genes whose expression levels have changed or the locations of transcription factor binding sites (TFBSs).

Compared with traditional PCR arrays, tiling arrays accommodate a much larger number of array elements, which are *in situ* synthesized oligonucleotides, typically dozens of nucleotides long. However, there is a trade-off for better coverage of the genome: as the average length of the array elements gets smaller, the variance of data increases due to the rise of the relative magnitude of random noise and the possibility of cross-hybridization and sequence artifacts. To deal with this problem, Affymetrix used a different method to score (one-channel) tiling arrays [10, 26] than the one used for PCR arrays: instead of considering each tile across array replicates separately, they used a sliding window around each tile to incorporate the hybridization intensity of its neighboring tiles. In our implementation of this method in Tilescope, we modified it by adding a nonparametric statistical test to assess the significance of the intensity difference between the test and the control samples at each tile. This extension enables us to score each tile using two different criteria. Moreover, we also adapted the original method to NimbleGen two-channel tiling arrays data, which in effect significantly increased the usability of this method.

For each tile, given its neighboring tiles across replicates, Tilescope calculates the pseudo-median log-ratio value as its signal. The pseudo-median (a.k.a. the Hodges-Lehmann estimator) of the log-ratio is a nonparametric estimator of the difference between the logged intensities of the test sample and those of the reference sample. It is calculated for each tile using a sliding window. The tiles from all arrays in a sliding window are first collected into a tile set, and the pseudo-median is calculated for this window as  $S = \text{median}[(\log\text{-ratio}_i + \log\text{-ratio}_j)/2]$  from all



$(i, j)$  pairs of tiles in the tile set. As a nonparametric estimator, pseudo-median is less susceptible to distributional abnormalities (such as skewness, unusual kurtosis, and outliers).

Due to the small sample size in each sliding window, whether the intensity distribution is normal or not in a given window cannot be reliably assessed. Without making the normality assumption about the intensity distribution, Telescope uses the nonparametric Wilcoxon signed-rank test [27] to compare the test with the reference signal intensities and quantifies the degree of significance by which the former consistently deviates from the latter across each of the sliding windows. It tests the null hypothesis that the median of the probability distribution of the differences between the logarithm of the intensities from the test sample and those from the reference sample is zero.

At the scoring step, Telescope generates two tile maps, the signal map and the P-value map (Figure 1C). Two values are calculated for each tile position: the pseudo-median of log-ratios, as a measure of the fold enrichment of the hybridization signal in the test sample over the reference at this genomic location and the probability, the P-value, that the null hypothesis—the local intensities of the test and the reference samples are the same—is true. In a recent study of transcript mapping with high-density tiling arrays, Huber et al. [17] used a different approach to score tiles. Their method does not assess intensity difference at individual tiles. Instead, it tries to find a step function that best fits the log-ratio intensities along genomic coordinates.

### *Feature identification*

Given the tile map annotated with pseudo-medians and P-values, Telescope filters away tiles that are below user-specified thresholds. Retained tiles are used to identify either differentially expressed genes or TFBSs. Currently Telescope users can choose one of three methods to identify such features (Figure 1D). The first method, ‘max-gap and min-run’, is a well-used method, initially used by Cawley et al. [26] to analyze their ChIP-chip tiling array data. The second method, ‘iterative peak identification’, is a new method that we developed to find genomic features iteratively. The third method, whose theoretical development is described in full details elsewhere [28], effectuates file segmentation by using a ‘hidden Markov model’ (HMM) explicitly built on validate prior knowledge.

Max-gap and min-run. Based on the observation that a tile is usually too short to constitute a feature alone, the first method, modified from the scoring scheme used in Cawley et al. [26] and Emanuelsson et al. [29], groups together qualified tiles that are close to each other along the genomic sequence into ‘proto-features’ and then discards any proto-features that are too short. To use this method, a user needs to specify the maximum genomic distance (‘max-gap’) below which two adjacent qualified tiles can be joined and the minimum length (‘min-run’) of a proto-feature for it to be qualified as a feature.

Iterative peak identification. The second method, which we have recently developed and implemented as part of the pipeline, does not group tiles above thresholds into features. Instead, it identifies local signal ‘peaks’ in an iterative fashion. This method was developed to generate lists of non-overlapping features of a uniform genomic size.

Taking the signal map that has been generated in the tile scoring step using window-smoothing to integrate the data from multiple replicate arrays, this method first identifies the tile (‘point source’) that corresponds to the peak in the signal map with the global maximum signal that also meets a predefined P-value threshold. A feature is then created centered at the genomic position of the peak with a predefined genomic size. We choose a feature size that is comparable with the average size of the fragmented ChIP DNA (typically  $\sim 1\text{Kb}$ ). The feature is assigned a signal measurement from the associated peak.

All tiles within a predefined distance from the located ‘peak’ are then removed from the signal map data. Typically the distance is the same size as the selected features, though it can be larger. This is to ensure that apparent ‘secondary peaks’ in the signal maps that are really part of the same feature are not separately identified. The procedure is then iterated to find the next maximum ‘peak’ in the remaining signal map data. The iteration generates a list of features ranked by ‘peak’ signals and terminates when the identified ‘peak’ signal is below a specified signal enrichment threshold.

Hidden Markov model. The third method uses a supervised scoring framework based on hidden Markov models to predict and score features in the genome tiled on the microarray [28]. Our method differs from previous HMM-based studies [30-32] by specifically taking *validated biological knowledge* (e.g. experimental validation, gene annotation, etc.) into consideration and systematically incorporating it to score different types of array assays within the same framework.

For identification of transcriptionally active regions (TARs)/transcribed fragments (transfrags) in transcriptional tiling array data, a four-state (TAR, non-TAR, and two other intermediate transition states) HMM is constructed using the knowledge of gene annotation (information of the probes that fall into annotated gene regions). For ChIP-chip data, a two-state (TFBS and non-TFBS) HMM is constructed by using the knowledge of inner regions in genes to estimate the signal emission distribution  $g(t)$  of the non-TFBS state, and by using the subtraction of  $g(t)$  from the overall emission distribution  $h(t)$  to estimate the emission distribution  $f(t)$  of the TFBS state. In a more general ideal scenario, our framework first selects a medium-sized set of sub-regions by using some appropriate analysis methods (e.g. the *MaxEntropy* sampling scheme discussed in [28]), and then utilizes the knowledge in these sub-regions as the training set to build the model for accurate analysis.

Further scoring on the initial analysis results can be also done by computing the posterior probabilities of each probe being active. The scores indicate the confidence in every single probe-level prediction and can be used to refine the previous analysis results by HMM. For instance, the identified active probes can be ranked according to the overall confidence levels in their regions and a threshold confidence level may either be set manually or be learned automatically to refine the original results.

Method comparison. We compared the performance of these three feature identification methods using a well-studied STAT1 ChIP-chip data set. Composed of three technical replicates, this data set was used to identify a list of STAT1 binding sites in the ENCODE regions [33]. These sites were later experimentally tested. We analyzed this data set using Telescope and generated three STAT1 binding site lists, each by a different feature identification method. Since identical tile scoring and thresholding parameters were used, the difference among these three lists reflects the underlying difference among the three feature identification methods. By using the list of the experimentally tested STAT1 binding sites, we were able to assess the sensitivity and specificity of each method. The receiver operating characteristic (ROC) curves in Figure 3 indicate that while in general the three feature identification methods implemented in Telescope have similar performance, the ‘iterative peak identification’ method is appreciably more sensitive at high ( $> 95\%$ ) specificity.

## Conclusions

### *Summary*

Telescope is an online software pipeline for processing high-density tiling microarray data. In a completely automated fashion, it will normalize signals between channels and across arrays, combine replicate experiments, score each array element, and identify genomic features. The program can process data from most gene expression, ChIP-chip, and CGH experiments. Telescope is designed with a graphical user-friendly interface to facilitate a user’s data analysis task, and the results, presented in a clear, well organized manner on a web page, can be downloaded for further analysis.

### *Future improvements*

Telescope is under active development: it is continually updated as better data processing methods become available.

### **Availability**

Tilescope is freely accessible for use at <http://tilescope.gersteinlab.org>. The source code of the pipeline is available at <http://tilescope.gersteinlab.org/download.html>.

### **Acknowledgements**

Z.D.Z was funded by an NIH grant (T15 LM07056) from the National Library of Medicine. This work was also supported by other grants from NIH to M.S. and M.G.

## References

1. Schena M, Shalon D, Davis RW, Brown PO: **Quantitative monitoring of gene expression patterns with a complementary DNA microarray.** *Science* 1995, **270**(5235):467-470.
2. Fodor SP, Read JL, Pirrung MC, Stryer L, Lu AT, Solas D: **Light-directed, spatially addressable parallel chemical synthesis.** *Science* 1991, **251**(4995):767-773.
3. Luscombe NM, Royce TE, Bertone P, Echols N, Horak CE, Chang JT, Snyder M, Gerstein M: **ExpressYourself: A modular platform for processing and visualizing microarray data.** *Nucleic Acids Res* 2003, **31**(13):3477-3482.
4. Saeed AI, Sharov V, White J, Li J, Liang W, Bhagabati N, Braisted J, Klapa M, Currier T, Thiagarajan M *et al*: **TM4: a free, open-source system for microarray data management and analysis.** *Biotechniques* 2003, **34**(2):374-378.
5. Lipshutz RJ, Fodor SP, Gingeras TR, Lockhart DJ: **High density synthetic oligonucleotide arrays.** *Nat Genet* 1999, **21**(1 Suppl):20-24.
6. Nuwaysir EF, Huang W, Albert TJ, Singh J, Nuwaysir K, Pitas A, Richmond T, Gorski T, Berg JP, Ballin J *et al*: **Gene expression analysis using oligonucleotide arrays produced by maskless photolithography.** *Genome Res* 2002, **12**(11):1749-1755.
7. Bertone P, Stolc V, Royce TE, Rozowsky JS, Urban AE, Zhu X, Rinn JL, Tongprasit W, Samanta M, Weissman S *et al*: **Global identification of human transcribed sequences with genome tiling arrays.** *Science* 2004, **306**(5705):2242-2246.
8. Stolc V, Gauhar Z, Mason C, Halasz G, van Batenburg MF, Rifkin SA, Hua S, Herreman T, Tongprasit W, Barbano PE *et al*: **A gene expression map for the euchromatic genome of *Drosophila melanogaster*.** *Science* 2004, **306**(5696):655-660.
9. Stolc V, Samanta MP, Tongprasit W, Sethi H, Liang S, Nelson DC, Hegeman A, Nelson C, Rancour D, Bednarek S *et al*: **Identification of transcribed sequences in *Arabidopsis thaliana* by using high-resolution genome tiling arrays.** *Proc Natl Acad Sci USA* 2005, **102**(12):4453-4458.
10. Kampa D, Cheng J, Kapranov P, Yamanaka M, Brubaker S, Cawley S, Drenkow J, Piccolboni A, Bekiranov S, Helt G *et al*: **Novel RNAs identified from an in-depth analysis of the transcriptome of human chromosomes 21 and 22.** *Genome Res* 2004, **14**(3):331-342.
11. Christine EH, Snyder M: **ChIP-chip: a genomic approach for identifying transcription factor binding sites.** 2002.
12. Urban AE, Korbel JO, Selzer R, Richmond T, Hacker A, Popescu GV, Cubells JF, Green R, Emanuel BS, Gerstein MB *et al*: **High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays.** *Proc Natl Acad Sci USA* 2006, **103**(12):4534-4539.
13. Pokholok DK, Harbison CT, Levine S, Cole M, Hannett NM, Lee TI, Bell GW, Walker K, Rolfe PA, Herbolsheimer E *et al*: **Genome-wide map of nucleosome acetylation and methylation in yeast.** *Cell* 2005, **122**(4):517-527.

14. Yuan GC, Liu YJ, Dion MF, Slack MD, Wu LF, Altschuler SJ, Rando OJ: **Genome-scale identification of nucleosome positions in *S. cerevisiae*.** *Science* 2005, **309**(5734):626-630.
15. Royce TE, Rozowsky JS, Bertone P, Samanta M, Stolc V, Weissman S, Snyder M, Gerstein M: **Issues in the analysis of oligonucleotide tiling microarrays for transcript mapping.** *Trends Genet* 2005, **21**(8):466-475.
16. Johnson WE, Li W, Meyer CA, Gottardo R, Carroll JS, Brown M, Liu XS: **Model-based analysis of tiling-arrays for ChIP-chip.** *Proc Natl Acad Sci U S A* 2006, **103**(33):12457-12462.
17. Huber W, Toedling J, Steinmetz LM: **Transcript mapping with high-density oligonucleotide tiling arrays.** *Bioinformatics* 2006, **22**(16):1963-1970.
18. Halasz G, van Batenburg MF, Perusse J, Hua S, Lu XJ, White KP, Bussemaker HJ: **Detecting transcriptionally active regions using genomic tiling arrays.** *Genome Biol* 2006, **7**(7):R59.
19. Gentleman RC, et al.: **Bioconductor: Open software development for computational biology and bioinformatics.** <http://www.bepress.com/bioconductor/paper1> 2004.
20. **NimbleGene Data Formats** [[http://tilescope.gersteinlab.org/Gff\\_Pair\\_Formats.pdf](http://tilescope.gersteinlab.org/Gff_Pair_Formats.pdf)]
21. **GFF: an Exchange Format for Feature Description** [<http://www.sanger.ac.uk/Software/formats/GFF>]
22. **RFC 1951** [<http://www.rfc-editor.org/rfc/rfc1951.txt>]
23. Goryachev AB, Macgregor PF, Edwards AM: **Unfolding of microarray data.** *J Comput Biol* 2001, **8**(4):443-461.
24. Dudoit S, Yang YH, Callow MJ, Speed TP: **Statistical methods for identifying genes with differential expression in replicated cDNA microarray experiments.** *Stat Sin* 2002, **12**(1):111-139.
25. Bolstad BM, Irizarry RA, Astrand M, Speed TP: **A comparison of normalization methods for high density oligonucleotide array data based on variance and bias.** *Bioinformatics* 2003, **19**(2):185-193.
26. Cawley S, Bekiranov S, Ng HH, Kapranov P, Sekinger EA, Kampa D, Piccolboni A, Sementchenko V, Cheng J, Williams AJ *et al*: **Unbiased mapping of transcription factor binding sites along human chromosomes 21 and 22 points to widespread regulation of noncoding RNAs.** *Cell* 2004, **116**(4):499-509.
27. Troyanskaya OG, Garber ME, Brown PO, Botstein D, Altman RB: **Nonparametric methods for identifying differentially expressed genes in microarray data.** *Bioinformatics* 2002, **18**(11):1454-1461.
28. Du J, Rozowsky JS, Korb J, Zhang ZD, Royce TE, Schultz MH, Snyder M, Gerstein M: **A supervised hidden Markov model framework for efficiently segmenting tiling array data in transcriptional and ChIP-chip experiments: systematically incorporating validated biological knowledge.** *Bioinformatics* 2006:10.1093/bioinformatics/btl1515 (In press).

29. Emanuelsson O, Nagalakshmi U, Zheng D, Rozowsky JS, Urban AE, Du J, Lian Z, Stolc V, Weissman S, Snyder M *et al*: **Assessing the performance of different high-density tiling microarray strategies for mapping transcribed regions of the human genome.** *Genome Res* 2006:(In press).
30. Ji H, Wong WH: **TileMap: create chromosomal map of tiling array hybridizations.** *Bioinformatics* 2005, **21**(18):3629-3636.
31. Li W, Meyer CA, Liu XS: **A hidden Markov model for analyzing ChIP-chip experiments on genome tiling arrays and its application to p53 binding sequences.** *Bioinformatics* 2005, **21 Suppl 1**:i274-i282.
32. Munch K, Gardner PP, Arctander P, Krogh A: **A hidden Markov model approach for determining expression from genomic tiling micro arrays.** *BMC Bioinformatics* 2006, **7**:239.
33. Euskirchen G, Rozowsky J, Wei C-L, Lee WH, Zhang ZD, Hartman S, Emanuelsson O, Stolc V, Weissman S, Gerstein M *et al*: **Optimal Mapping of Transcription Factor Binding Sites in Mammalian Cells.** *Genome Res* 2006:(In press).

## Tables

**Table 1** Feature comparison between tiling microarray data analysis software <sup>1</sup>

	TileScope	Bioconductor <sup>2</sup>	TAS <sup>3</sup>	MAT <sup>4</sup>	TileMap
Implementation	Web	R packages	Standalone	Standalone	Standalone
Graphic user interface	√ <sup>5</sup>	×	√	×	×
Intended usage:					
– Transcription data	√	√	√	×	√
– ChIP-chip data	√	√	×	√	√
Applicable array platform:					
– Affymetrix	√	√	√	√	√
– NimbleGen	√	×	×	×	×
Data normalization:					
– Mean/median	√	~	√	/	×
– Loess	√	~	×	/	×
– Quantile	√	~	×	/	√
Feature identification:					
– Max gap and min run	√	~	√	/	√
– Iterative peak identification	√ (new)	×	×	/	×
– Hidden Markov model	√	~	×	/	√

1. Only programs explicitly applicable to high-density tiling microarray data were considered. The websites of the compared programs are listed as follows:

TileScope: <http://tilescope.gersteinlab.org/>

Bioconductor : <http://www.bioconductor.org/>

TAS: <http://www.affymetrix.com/support/developer/downloads/TilingArrayTools/index.affx>

MAT : <http://chip.dfci.harvard.edu/~wli/MAT/>

TileMap : <http://biogibbs.stanford.edu/~jihk/TileMap/index.htm>

2. Strictly speaking, Bioconductor is not a ready-to-run program. It is a collection of software packages/libraries written in R. As a tool box, the analysis methods that it provides need to be written in an R program to run.

3. TAS is previously known as GTRANS.

4. MAT standardizes the probe value through the probe model, which obviates the need for sample normalization.

5. Comparison symbols used in the table:

√, available;

×, not available;

~, available but need to be programmed;

/, not applicable.



## Figure Legends

**Figure 1.** Tiling array data processing by Telescope. (A) Flow chart of major data processing steps. Yellow icons represent data in user-accessible files, and blue ones data in the pipeline program memory. See main text for details. (B) Log-intensity scatter plots of a tiling array from the STAT1 experiment set before and after normalization by four different methods. The first panel is the  $\log_2 T$  versus  $\log_2 R$  plot before normalization, where  $T$  and  $R$  are test intensity and reference intensity respectively. The gray line represents where these two log-intensities are equal. The second panel is  $\log_2(T/R)$  versus  $\log_2(T \times R)^{1/2}$  plot (the MA plot) before normalization. The dependency of the log-ratio on the intensity, evinced by a fitted loess curve, is prominent in the data. The rest panels are the MA plots of array data after mean, media, loess, or quantile normalization. They clearly show that the distribution of log-ratios are centered at zero by all normalization methods, but the intensity-specific artifacts in the log-ratio measurements are removed by only loess or quantile normalization but not by the mean- or median-based method. (C) Signal and P-value maps of all tiles in the ENCODE ENm002 region. In this region, the tiles near the transcription start site of IRF1, a transcription factor known to be regulated by STAT1, give the strongest signals. (D) Telescope-identified STAT1 binding sites at the 5'-end of IRF1 are shown on the custom track in the UCSC genome browser.

**Figure 2.** Screenshots of Telescope. (A) The applet of Telescope, the graphic user interface of the pipeline. (B) An example of data analysis result web page.

**Figure 3.** The ROC curves of the three feature identification methods implemented in Telescope. The comparison of the performance of these methods was based on a well-studied STAT1 ChIP-chip data set and a list of experimentally tested STAT1 binding sites.

Figures

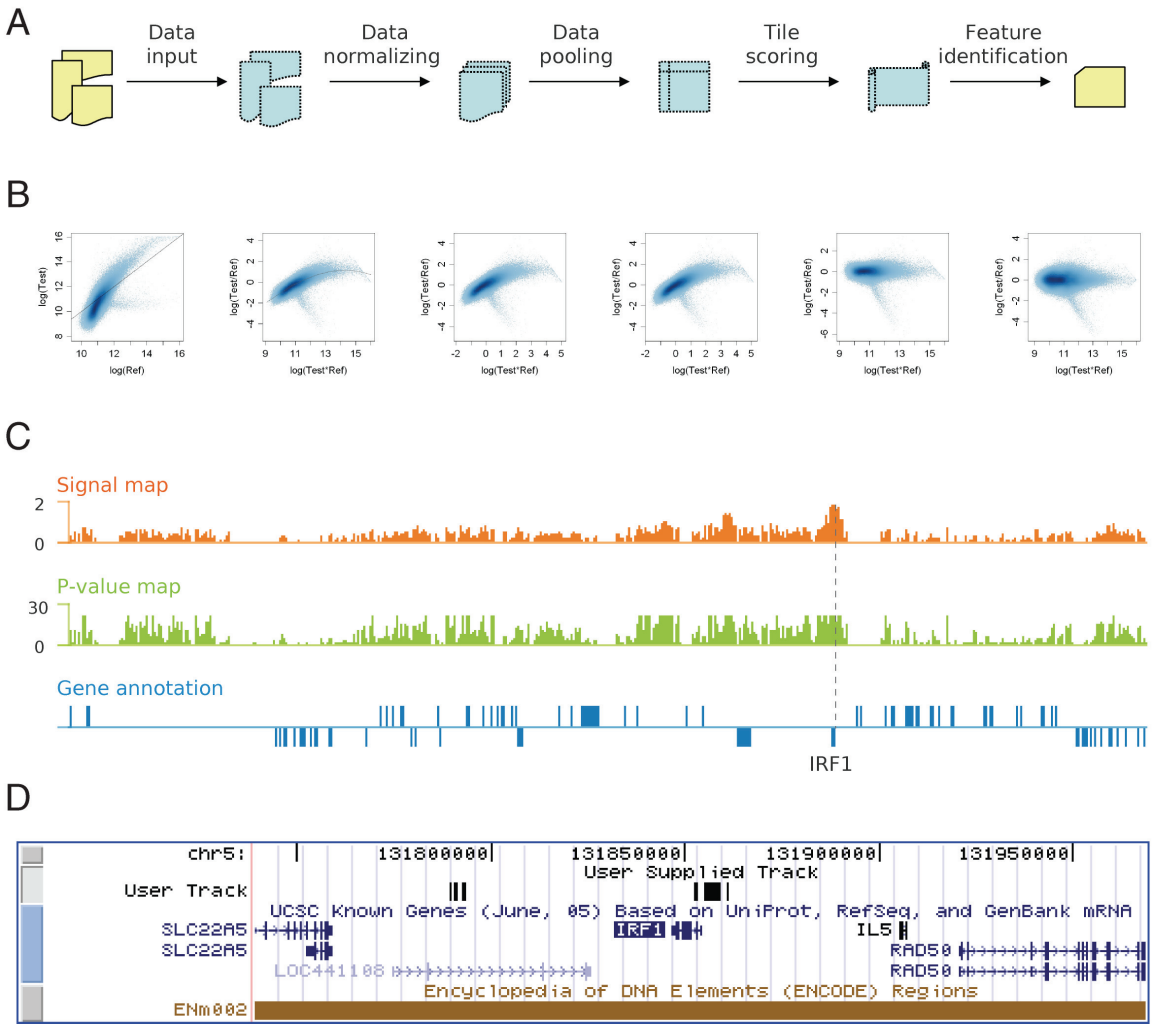
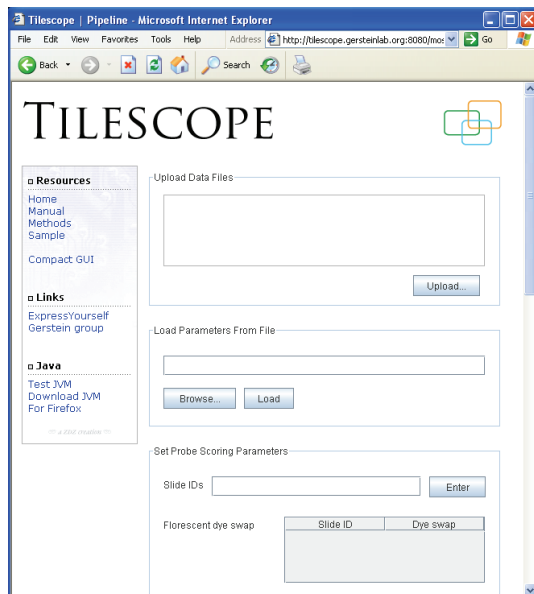


Figure 1.

A



B

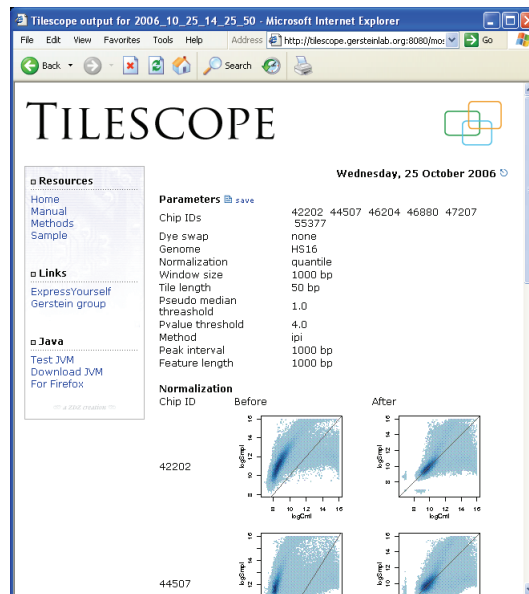
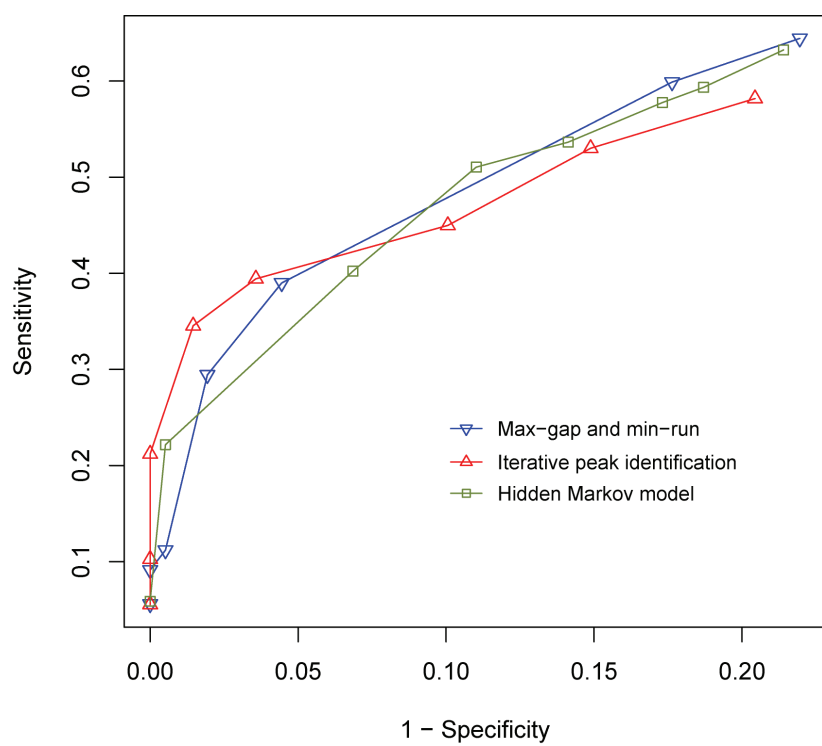


Figure 2.



**Figure 3.**

## Supplementary material

### *Technical Details of the Optimization Algorithm*

Nearly all the data files and design files of microarray are tab delimited. Generally speaking, each line in the files represents a data record and each tab delimited field represents a particular type of data (see Supplementary Table 1). For example, the first column of the data in a GFF file is a sequence name. Since different columns represent different data types, the data value in a column could be a string, number, decimal, nucleotide sequence, or etc. For example, the sixth column, which represents the X coordinate of a microarray, in a PAIR file is in number format. Due to the fact that these data types are already known in each format, we could take this assumption and optimize the data storage according to their types. As mentioned before, storing the data in ASCII format makes the file larger than it should be if compared to storing the data in their primitive types or any other optimal types. For instance, a 32 bit integer requires 10 bytes in ASCII but only 4 bytes in its primitive type. A more extreme case would be the nucleotide sequence. Since a nucleotide of DNA can only be A, T, C, or G, we actually only need to use 2 bits to store each nucleotide so that one byte can already store 4 nucleotides.

As a result, our optimization algorithm mainly focuses on converting these kinds of data into a more optimal format. To achieve this, we created a configuration file containing the information of the data column for different microarray file formats (see Supplementary Table 2). We also created a set of Java API which parses these files and optimizes the data according to the configuration file. We found that the size of the optimized data is reduced significantly if compared to the original size although the compression ratio is not as good as Zip. However, we also found that a higher compression ratio can be achieved if we zip the optimized data instead of the raw data (see Supplementary Table 3). To this end, we chose to first optimize our input data using our optimization algorithm and then bundle the optimized data with its configuration file using zip. This helps reducing the overall file size substantially and so the transfer time. Therefore, our optimization algorithm is not replacing any compression algorithm but enhancing the compression ratio. An added advantage of our algorithm is that it enables us to locate a data record more easily. This is because we store every data entry in a fixed number of bits and sort all the entries by their coordinates, so that the offset of a particular record in the byte stream can be easily calculated by their coordinates given that all the data are in contiguous coordinates. Although our format is capable of locating a record faster, we instead chose to convert this “middle” format into the standard GFF format for processing independent of the original format of the data. This ensures our processing consistency as we only need to deal with a single format and the most important reason is that we will be able to give the users back their data in a standard format later if necessary.

**Supplementary Table 1** The meaning of columns of various tab-delimited input data file formats.

1	2	3	4	5	6	7	8	9	10
<i>GFF format</i>									
Seq. Name	Source	Feature	Start Pos.	End Pos.	Score	Strand	Frame	Group	—
<i>PAIR format</i>									
Image ID	Gene Exp. Opt.	Seq. ID	Probe ID	Pos.	X	Y	Match Index	Perfect Match	Mis-match
<i>POS format</i>									
Seq. ID	Seq. Name	Position	Probe ID	Count	Length	—	—	—	—
<i>BPMAP format</i>									
X	Y	Seq. Name	Pos.	Probe Seq.	—	—	—	—	—
<i>CEL format</i>									
X	Y	Mean	SD	No. of Pixels	—	—	—	—	—

Note: We assume all the data files, including the design files if any, are in tab delimited format. The data type of each column of data of each format is defined in an XML configuration file. In the configuration file, it also stores the file type, namely ‘design’ and ‘data’, of each supported format.

**Supplementary Table 2** Attributes of the element tag defined in the configuration file §.

Type	Minimum value	Maximum value	Decimal	Size (example)
Number	The minimum possible integer	The maximum possible integer	—	4 bytes (147,971,601)
Decimal	The minimum possible integer	The maximum possible integer	Number of decimal places	3 bytes (65535.99)
Text	Unused (0)	The maximum number of characters	—	6 bytes (N10023)
Nucleotide Sequence	Unused (0)	The number of bases in the sequence	—	7 bytes †
Highly Repetitive String	Unused (0)	The maximum possible number of unique strings	—	1 byte ‡

§ ‘Type’, ‘Min’, ‘Max’, and ‘Decimal’ are the attributes that define the data type of a column in a data file.

†. A string of 25 bases (e.g., AAACGAATTGCCATTAGGCCATTAG)

‡. 256 unique strings.

Note: Following is an example of the definition of the Affymetrix design file format in the configuration file:

```
<dataset type="design" format="txt">
  <elements>
    <element name="x" min="0" max="65535" type="number" disabled="false" />
    <element name="y" min="0" max="65535" type="number" disabled="false" />
    <element name="seq" min="0" max="255" type="hrString" disabled="false" />
    <element name="pos" min="0" max="2147483647" type="number" disabled="false" />
    <element name="probe" min="0" max="25" type="sequence" disabled="false" />
  </elements>
</dataset>
```

The ‘name’ attribute is the column identifier used in the API to retrieve back the data corresponding to that column. The ‘disabled’ attribute indicates if the data of that column should be converted.

**Supplementary Table 3** File size reduction by Zip and our optimization algorithm. <sup>§, †</sup>

	Design File	Data File	All Files	
<i>Affymetrix:</i>				
Original	36	37.5	73.5	
Zip	11.9	7.7	19.6	
Optimization	11.5	10.9	22.5	
Optimization+Zip	—	—	12.1	83.5% <sup>‡</sup>
<i>NimbleGen:</i>				
Original	16.4	32.4	48.8	
Zip	2.3	6.1	8.4	
Optimization	10.6	5.8	16.4	
Optimization+Zip	—	—	6.8	86.1% <sup>‡</sup>

§. This table compares the size of two different sets of sample input files, an Affymetrix data set and a NimbleGen data set, before and after optimization and zip. “Original”, “Zip”, and “Optimization” represent the original, zipped, and optimized size of the data respectively. “Optimization+Zip” represents the size of the data zipped after optimization.

†. File size is in MB.

‡. Percentage of total file size reduction.