Computing and Visualizing the Generalized Singular Value Decomposition in Python

Rui Luo
University of Utah

UUCS-19-003

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

29 April 2019

Abstract

The human genome project has been completed, but there are barriers between researchers who study the genetic sequences and clinicians who treat cancers. First of all, there is low reproducibility in genetic studies, caused by different sequencing techniques and batch effects. Secondly, it is difficult for clinicians who do not have a computational background to interpret existing computational methods. To minimize these disconnections, a computational model should be developed to find the significant genes in a genome that separate batch and experimental effects from biological effects. The proposed solution is to use the generalized singular value decomposition (GSVD) to reveal genetic patterns on the transformation of genes, and to separate the tumor-exclusive genotype from experimental inconsistencies.

Here we developed a computation and visualization toolkit to improve computing and visualizing the GSVD in Python.

COMPUTING AND VISUALIZING THE GENERALIZED SINGULAR VALUE DECOMPOSITION IN PYTHON

by

Rui Luo

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Computer Science

School of Computing

The University of Utah

May 2019

ABSTRACT

The human genome project has been completed, but there are barriers between researchers who study the genetic sequences and clinicians who treat cancers. First of all, there is low reproducibility in genetic studies, caused by different sequencing techniques and batch effects. Secondly, it is difficult for clinicians who do not have a computational background to interpret existing computational methods. To minimize these disconnections, a computational model should be developed to find the significant genes in a genome that separate batch and experimental effects from biological effects. The proposed solution is to use the generalized singular value decomposition (GSVD) to reveal genetic patterns on the transformation of genes, and to separate the tumor-exclusive genotype from experimental inconsistencies.

Here we developed a computation and visualization toolkit to improve computing and visualizing the GSVD in Python.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Sequencing human genomes has become less expensive. However, scientists are having a difficult time obtaining the information they expect to derive with the results from genetic sequencing. For example, scientists need to analyze the genome to find out which parts on the genome are significant. In order to determine the potentially significant areas, scientists need to perform a large number of trials among the whole genome. An example of the difficulty researchers face is the regularly observed patterns of copy-number variations in cancer. Although the sequencing results known as recurring DNA alterations have been observed to play an important role in the study of cancer, they have not been applied to actual clinical use.

Genomic signal processing tools are built to remove the barriers of transferring genetic sequencing results to usable information. Genomic signal processing tools that have been built to uncover the underlying patterns of sequencing results and assist clinicians to better interpret recurring DNA alterations.

Most of the research relevant to genome-wide patterns of DNA copy-number variations does not focus on the subtypes of DNA copy-number variations. However, the subtypes of DNA copy-number variations are the indicators. The subtypes of DNA copy-number variations can increase the accuracy of predictions. By including the tools mentioned above in the research experiments, the results become more persuasive.

Our lab's genomic signal processing has been implemented in Mathematica. The functions that are used in Mathematica can be extended into a toolkit. While Mathematica computes matrix decompositions correctly there is no evidence that it uses highly optimized LAPACK routines. The decompositions contained in the Numpy linear algebra library, however, are based on these LAPACK routines. This indicates that we may be able to obtain performance improvements by implementing the GSVD in Python utilizing

the Numpy library. An additional benefit that Python has is that there are many cloud computing pipelines that interface well with Python, but a paucity that are compatible with Mathematica. Finally, due to Python being available at no cost and the ease of creating and sharing libraries Python offers a platform for our labs research to appeal to a broader audience.

CHAPTER 2

BACKGROUND

2.1 The GSVD as a comparative spectral decomposition

Given two column-matched but row-independent real matrices $D_i \in \mathbb{R}^{M_i \times N}$, each with full column rank $N \leq M_i$, the GSVD is an exact simultaneous factorization [1–4],

$$D_i = U_i \Sigma_i V^T = \sum_{n=1}^{N} \sigma_{i,n} u_{i,n} \otimes v_n^T, \quad i = 1, 2, \tag{2.1}$$

where $U_i \in \mathbb{R}^{M_i \times N}$ are real and column-wise orthonormal and $V^T \in \mathbb{R}^{N \times N}$ is real, invertible, and with normalized rows. The $2N$ positive generalized singular values are arranged in $\Sigma_i = \text{diag}(\sigma_{i,n}) \in \mathbb{R}^{N \times N}$ in a decreasing order of the ratio $\sigma_{1,n}/\sigma_{2,n}$. The GSVD is unique up to phase factors of ± 1 of each triplet of corresponding column and row basis vectors, i.e., $u_{i,n}$ and v_n, except in degenerate subspaces defined by subsets of pairs of generalized singular values of equal ratios, i.e., $\sigma_{1,n}/\sigma_{2,n}$. The GSVD generalizes the SVD from one to two matrices. Like the SVD, the GSVD is a mathematical building block of algorithms, e.g., for solving the problem of constrained least squares in algebra [5], and theories, e.g., for describing oscillations near equilibrium in classical mechanics [6].

2.1.1 Generalized fractions and angular distances

The authors [16] formulated the GSVD as a comparative spectral decomposition that can simultaneously identify the similarity and dissimilarity between two column-matched but row-independent matrices, and, therefore, create a single coherent model from two datasets recording different aspects of interrelated phenomena [7, 8]. This formulation [9–12] is possible because the GSVD is exact, exists, and has uniqueness properties that directly generalize those of the SVD [13, 14] Eq. (2.1). The only assumption is that there exists a one-to-one mapping between the columns of the matrices but not necessarily

between their rows. The authors [16] defined the significance of the row basis vector v_n^T and the corresponding column basis vector $u_{i,n}$ in the corresponding matrix D_i, i.e., the "generalized fraction" $p_{i,n}$, to be proportional to the corresponding generalized singular value $\sigma_{i,n}$, and the "generalized normalized Shannon entropy" of D_i to be proportional to the arithmetic mean of $p_{i,n} \log p_{i,n}$. The authors [16] defined the significance of v_n and $u_{1,n}$ in D_1 relative to that of v_n and $u_{2,n}$ in D_2, i.e., the "GSVD angular distance," to be a function of the ratio $\sigma_{1,n}/\sigma_{2,n}$ that, from the cosine-sine decomposition, is related to an angle,

$$-\pi/4 < \theta_n = \arctan(\sigma_{1,n}/\sigma_{2,n}) - \pi/4 < \pi/4. \tag{2.2}$$

Note that the angular distances θ_n are different from the principal angles corresponding to canonical correlations, just as the GSVD is different from canonical correlations analysis (CCA) [15].

A unique row basis vector v_n^T that is significant in either D_1 or D_2, and with an angular distance of $\theta_n \approx \pm\pi/4$, which corresponds to a ratio of $\sigma_{1,n}/\sigma_{2,n} \gg 1$ or $\ll 1$, respectively, is mathematically approximately exclusive to either D_1 or D_2, and for consistency should be interpreted with the corresponding column basis vector $u_{1,n}$ or $u_{2,n}$ to represent phenomena exclusive to either the first or the second dataset. A unique row basis vector v_n^T that is significant in both D_1 and D_2, and with an angular distance of $\theta_n \approx 0$, which corresponds to $\sigma_{1,n}/\sigma_{2,n} \approx 1$, is mathematically common to D_1 and D_2, and should be interpreted with both $u_{1,n}$ and $u_{2,n}$ to represent phenomena common to both datasets.

The GSVD, which is mathematically invariant under the exchange of the two matrices or the reordering of the pairs of matched columns or the rows, and then here is also blind to the labels of the matrices, the columns, and the rows. These labels are used to interpret only the row and column basis vectors in terms of the phenomena recorded in the datasets.

2.1.2 Computing the GSVD via QR decomposition and the SVD

The GSVD is numerically stably computed by using the QR decomposition of the appended D_1 and D_2, followed by the SVD of the block of the column-wise orthonormal Q that corresponds to D_1, i.e., Q_1,

$$\begin{aligned}\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} = QR &= \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R \\ &= \begin{bmatrix} U_{Q_1}\Sigma_{Q_1}V_{Q_1}^T \\ Q_2 \end{bmatrix} R = \begin{bmatrix} U_{Q_1}\Sigma_{Q_1} \\ U_{Q_2}\Sigma_{Q_2} \end{bmatrix} V_{Q_1}^T R,\end{aligned} \tag{2.3}$$

where R is upper triangular [4, 5]. Since D_1 and D_2 are with full column rank, then Q_1 and Q_2 are also with full column rank, $V_{Q_1}^T$ is orthonormal, and Σ_{Q_1} is positive diagonal. It follows from Eq. (2.3) that the diagonal $\Sigma_{Q_2} = (I - \Sigma_{Q_1}^2)^{\frac{1}{2}}$ is also positive, and that $U_{Q_2} = Q_2 V_{Q_1}(I - \Sigma_{Q_1}^2)^{-\frac{1}{2}}$ is column-wise orthonormal,

$$\begin{aligned} I = Q^T Q &= Q_1^T Q_1 + Q_2^T Q_2 \\ &= V_{Q_1}\Sigma_{Q_1}^2 V_{Q_1}^T + Q_2^T Q_2, \\ \Sigma_{Q_2}^2 &= I - \Sigma_{Q_1}^2 = (Q_2 V_{Q_1})^T (Q_2 V_{Q_1}) > 0, \\ U_{Q_2}^T U_{Q_2} &= I \\ &= [Q_2 V_{Q_1}(I - \Sigma_{Q_1}^2)^{-\frac{1}{2}}]^T [Q_2 V_{Q_1}(I - \Sigma_{Q_1}^2)^{-\frac{1}{2}}]. \end{aligned} \tag{2.4}$$

That is, the SVD of Q_1 also defines an SVD of Q_2, where the singular values are arranged in Σ_{Q_2} in an increasing order, because the singular values of Q_1 are arranged in Σ_{Q_1} in a decreasing order.

It follows from Eq. (2.4) then that the SVD of Q_1 factorizes D_1 and D_2 into the GSVD,

$$\begin{aligned} U_1 &= U_{Q_1}, \\ U_2 &= U_{Q_2}, \\ \Sigma_1 &= \Sigma_{Q_1}\{\mathrm{diag}[(V_{Q_1}^T R)(V_{Q_1}^T R)^T]\}^{\frac{1}{2}}, \\ \Sigma_2 &= (I - \Sigma_{Q_1}^2)^{\frac{1}{2}}\{\mathrm{diag}[(V_{Q_1}^T R)(V_{Q_1}^T R)^T]\}^{\frac{1}{2}}, \\ V^T &= \{\mathrm{diag}[(V_{Q_1}^T R)(V_{Q_1}^T R)^T]\}^{-\frac{1}{2}} V_{Q_1}^T R, \end{aligned} \tag{2.5}$$

where U_1 and U_2 are column-wise orthonormal, Σ_1 and Σ_2 are positive diagonal, and V^T, identical in both factorizations, has normalized rows. The positive generalized singular values are arranged in $\Sigma_1\Sigma_2^{-1} = \Sigma_{Q_1}(I - \Sigma_{Q_1}^2)^{-\frac{1}{2}}$ in a decreasing order.

The QR decomposition is unique and, from Eq. (2.5), the uniqueness properties of the GSVD follow from the uniqueness properties of the SVDs of Q_1 and Q_2.

2.2 Genomic signal processing case study

DNA alterations had been observed in astrocytoma for decades. A copy-number genotype predictive of a survival phenotype was discovered only by using the generalized singular value decomposition (GSVD) formulated as a comparative spectral decomposition.

In this case study, the authors [16] used the GSVD to compare whole-genome sequencing (WGS) profiles of patient-matched astrocytoma and normal DNA. First, the GSVD uncovered a genome-wide pattern of copy-number alterations, which was bounded by patterns recently uncovered by the GSVDs of microarray-profiled patient-matched glioblastoma (GBM) and, separately, lower-grade astrocytoma and normal genomes. Like the microarray patterns, the WGS pattern was correlated with an approximately one-year median survival time. By filling in gaps in the microarray patterns, the WGS pattern revealed that this biologically consistent genotype encoded for transformation via the Notch together with the Ras and Shh pathways. Second, like the GSVDs of the microarray profiles, the GSVD of the WGS profiles separated the tumor-exclusive pattern from normal copy-number variations and experimental inconsistencies, including the WGS technology-specific effects of guanine-cytosine content variations across the genomes that were correlated with experimental batches. Third, by identifying the biologically consistent phenotype among the WGS-profiled tumors, the GBM pattern proved to be a technology-independent predictor of survival and response to chemotherapy and radiation, statistically better than the patient's age and tumor's grade, the best other indicators, and *MGMT* promoter methylation and *IDH1* mutation. The authors [16] concluded that by using the complex structure of the data, comparative spectral decompositions underlay a mathematically universal description of the genotype-phenotype relations in cancer that other methods missed.

Here we demonstrate a computation and visualization toolkit by applying it to the data from this case study.

CHAPTER 3

METHODS

3.1 Computing the GSVD via QR decomposition and the SVD

The QR decomposition and the SVD are important steps in computing the GSVD. The efficient computation of the GSVD is based on the QR and SVD decompositions of the datasets. QR decomposition is performed on the stacked datasets such as $matrix1$ in the code snippet, and the SVD is computed on the rows of Q that correspond to D_1 and D_2 separately.

3.1.1 Computing QR decomposition

The input data can potentially have missing values. We ensured that all Nulls or NAs had been dropped from the input matrices D_1 and D_2. The matrices D_1 and D_2 were then concatenated vertically to form a $(n1 + n2)$ by m matrix for QR decomposition where $n1$ and $n2$ are the rows of D_1 and D_2, respectively, and m are the same columns of either dataset. To perform QR decomposition on D_1 and D_2, we used the linalg.qr command from the Numpy package in Python. Numpy utilizes the LAPACK algorithms, which are highly optimized for linear algebra. Taking advantage of this optimization, we used the Numpy linalg package instead of any others or instead of implementing our own package. The output of the linalg.qr command is the decomposition matrices Q with orthonormal columns and upper triangular matrix R.

```
# Ensure that all Nulls or NA's have been dropped
matrix1 = matrix1.dropna()

# Compute the QR decomposition
q, r = np.linalg.qr(matrix1)
```

3.1.2 Computing the SVD

The matrix Q was split into two matrices, one of size $n1$ by m corresponding to the dimensions of D_1 and the other of size $n2$ by m corresponding to D_2. These matrices are referred to as Q_1 and Q_2, respectively. Using the linalg.svd function from the Numpy package, we decomposed the matrix Q_1 into U_1, Σ_1 and V_1^T and the matrix Q_2 into U_1, Σ_1 and V_2^T.

```
    # Compute the SVD of the block in Q corresponding to matrix1
    q1 = q[:matrix1.shape[0]]
    u1, w1, vt1 = np.linalg.svd(q1, full_matrices=False)
```

3.2 Testing the GSVD raster visualization

To utilize the GSVD and find underlying patterns from the GSVD, we developed the raster visualization for the GSVD. My task was to test if the visualization correctly depicted the datasets. The `raster_display` function shown below can be applied to any decomposition. The positional parameters of `axes`, `matrix`, `row_basis_vectors`, `probe_num` and `step` are unique to each visualization. This `raster_display` can be applied to D_1 and D_2 as well as the results of the GSVD: U_1, U_2, Σ_1, Σ_2 and V^T to visualize all the patterns in the GSVD and original dataset.

```
def raster_display(axes, matrix, row_basis_vectors, probe_num, step,
                   raster_contrast=1, orientation="horizontal", aspect_ratio='auto',
                   color_axes=None):
```

3.3 Computing and visualizing the generalized fractions and angular distances

3.3.1 Computing and visualizing the generalized fractions

The code below demonstrates how generalized fractions were computed. The first step was to multiply each element of the diagonal with itself and store it as the variable `fractions`. Then we summed all the diagonal elements and stored them in variable `total_fractions`. The function returns an array of `fractions` divided by `total_fractions`.

```
    # Compute total fractions
    fractions = (np.diagonal(matrix))**2
    total_fractions = np.sum(fractions)

    return fractions/total_fractions
```

The function used to visualize the generalized fractions is displayed below. The function calls the `generalized_fractions` function described above internally to compute the fractions. The fractions are then joined with an integer list that is the same length as the generalized fractions array to preserve the fractions' original location. This list is displayed in a horizontal bar chart, with the fractions' corresponding integer position as its label.

```
    # create a list of positions, and join it with the generalized fractions
    positions_list = np.arange(1, matrix.shape[0] + 1)
    fractions = generalized_fractions(matrix)
    to_plot = np.vstack((positions_list, fractions)).T
```

3.3.2 Computing and visualizing angular distances

The steps of computing angular distances follow Eq.(2.2). The variable `distances` contains an array of angular distances to be plotted.

```
    # Compute theta between the diagonal elements
    distances = [(np.arctan(matrix1[i, i]/matrix2[i, i]) - math.pi/4)
                 for i in range(matrix1.shape[0])]

    return distances
```

In order to visualize the angular distances, we took the variables `d_1` and `d_2` from the GSVD as input to the function `angular_distances`. Then we indexed the angular distances array by its length and plotted the array in the form of a horizontal bar chart.

```
    # Plotting horizontal bars according to the distribution of angular distances
    distance = angular_distances(d1, d2)
    index = np.arange(len(distance))
    axes.barh(index + bar_width, distance, bar_width, align='center', color='red')

    return
```

3.4 Computing and visualizing Kaplan-Meier survival analysis

3.4.1 Lifelines versus scikit-survival

Lifelines and scikit-survival are two libraries in Python. Lifelines has more features built into it, including its own plot method, and has been extensively tested against R and SAS. The scikit-survival is dependent on the sklearn module and is not extensively supported. Lifelines was chosen for support and capability reasons.

3.4.2 Extending lifelines visualization

Here we initialized the Kaplan-Meier objects by calling the function `KaplanMeierFitter`, which is a lifelines function. `KaplanMeierFitter` takes time and event data and fits the survival function to the data. Helper functions not shown here generate Boolean arrays that separate time and event data into user-defined subgroups. These Boolean arrays are generated based on event column data and group column data in a dataset provided by the user.

```
    km_objs = [KaplanMeierFitter().fit(time[bool_groups[i]],
                                       event_observed=event[bool_groups[i]],
                                       label=levels[i]) for i in range(len(levels))]

    return km_objs
```

The Kaplan-Meier objects have a built-in plot method. Each Kaplan-Meier object in `km_objs` is called on a subplot given by the `ax` parameter and plotted with our default styles specified by the `colors` parameter and `censor_style` dictionary parameter. If a group has too few patients included, then it can be removed with `remove_list`. The parameters of `annotation_coords` and `patch_color` are used to annotate the plot area and increase the amount of information that can be conveyed.

```
def km_curve_display(ax, plot_label, df, event_columns, group_columns=[], colors=None,
                     annotation_coords=None, dataset_label=None, remove_list=[],
                     Cox_PH=False, logrank=True, time_conversion=(365.25/12),
                     patch_color='#ffffcc', xtick_pos=[0, 40, 80, 120],
                     ytick_pos=[0, 0.25, 0.50, 0.75, 1.0], title_pos='center',
                     censor_style={'marker': '|', 'ms': 10, 'mew': 1.5}):
```

3.5 Creating boxplot displays

3.5.1 Mapping columns (attributes) to similar groups

Before displaying of patterns in the column and row basis vectors in a boxplot format, the data have to be separated into two groups based on annotations. The user supplies annotations that separate the data into two groups based on a column in a pandas dataframe. The pandas factorize method is then used to create dummy variables that are integers in the place of strings. By using these dummy variables, the mapping step can work independently of any user input regardless of the annotations in the input column.

CHAPTER 4

RESULTS

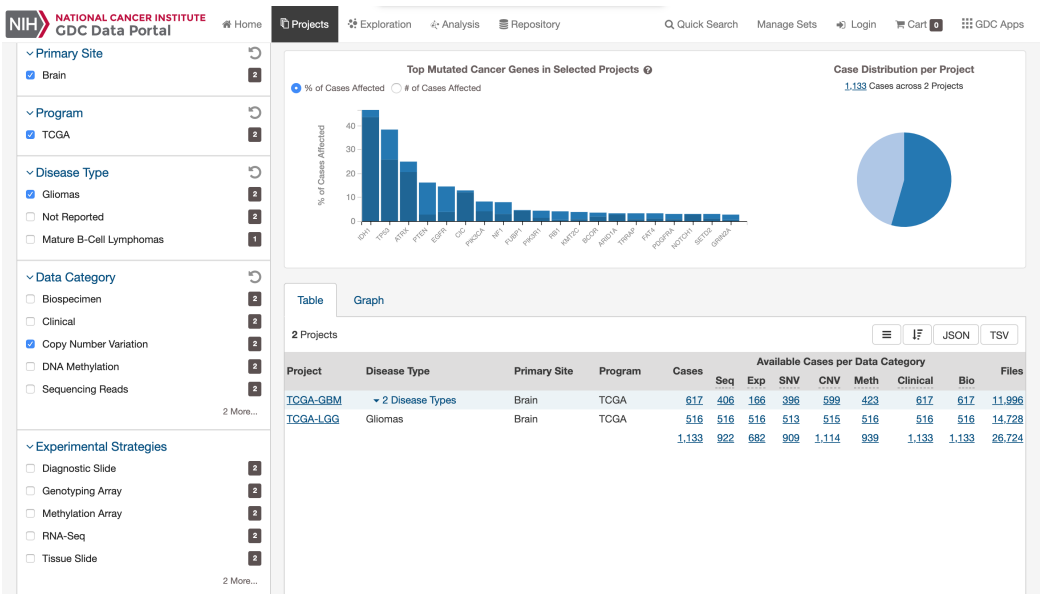
Case study of the TCGA astrocytoma datasets

4.1 TCGA astrocytoma tumor and patient-matched normal DNA copy-number datasets

The Cancer Genome Atlas (TCGA) astrocytoma datasets are obtained from the Genomic Data Commons (GDC).

`https://portal.gdc.cancer.gov/projects`

`http://www.alterlab.org/astrocytoma_genotype-phenotype/`



The data are composed of two datasets, including one tumor dataset and one normal dataset. For each dataset, rows contain the information of copy number variations along the genomic coordinate, whereas columns contain the identification information of patients. Each column represents DNA copy number variations according to one patient, and

each row represents the DNA copy number variations based on one subposition location on the genome.

During the data preprocessing step, only patients who have both a normal profile and a tumor profile are included in the datasets. Thus, the columns of the two datasets are the same. This is an important criterion for applying the GSVD to these datasets. Both the tumor copy-number variations information and the normal copy-number variations information are needed in order to analyze the tumor exclusive patterns or the normal exclusive patterns or common in both. If the target patients are not the same group of patients in both datasets, the results will no longer be comparative, and they cannot separate patterns from the tumor datasets and normal datasets. There does not have to be an exact match along the rows of the two datasets to be used in the GSVD. As long as the rows have similar meanings, the datasets are valid as input to the GSVD.

4.2 Visualization of the GSVD in this case study

After computing the GSVD as described in sections 3.1–3.2 and after visualization as described in section 3.3, Figure 4.1 below has been generated. This figure matches the previously generated figure in [16]. We directly compared the results of matrices U, Σ and V^T by subtracting Mathematica matrices from the corresponding ones in Python. The absolute value of the difference is less than the default machine precision $1e^{-16}$ in both Mathematica and Python. We also subtracted the angular distances computed by Mathematica from the Python corresponding ones. The absolute difference is less than default machine precision $1e^{-16}$.

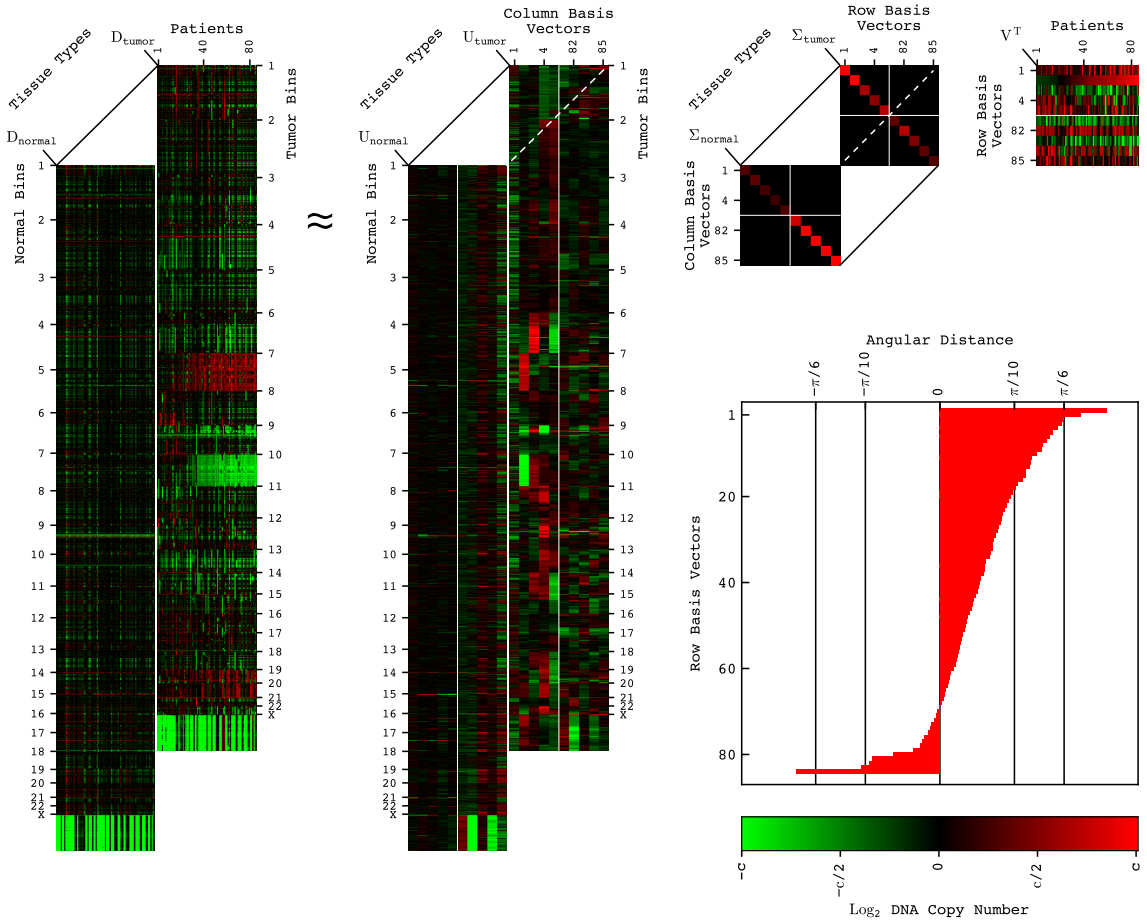


Figure 4.1. **The GSVD of the WGS read-count profiles of patient-matched astrocytoma tumor and normal DNA.** The GSVD is depicted in a raster display with relative WGS read-count, i.e., DNA copy-number amplification (red), no change (black), and deletion (green). This GSVD depiction is denoted as approximate, even though the GSVD of Eq. (2.1) is exact, because only the 1st through the 5th and the 81st through the 85th rows and corresponding tumor and normal column basis vectors and generalized singular values are explicitly shown. The angular distances of Eq. (2.2) are depicted in a bar chart. The red and green contrasts for the datasets D_i, the dataset-specific column basis vectors U_i and generalized singular values Σ_i, and the dataset-shared row basis vectors V^T, are $c = 1,750$ and 0.0005, and 5, respectively.

4.3 Visualization of the bar charts in this case study

The bar charts in Figure 4.2 were generated using the functions described in section 3.3. The charts match the previously generated figures in [16]. We compared the results of generalized fractions by subtracting Mathematica generalized fractions from the corresponding ones in Python. The absolute value of the difference is less than the default machine precision $1e^{-16}$.

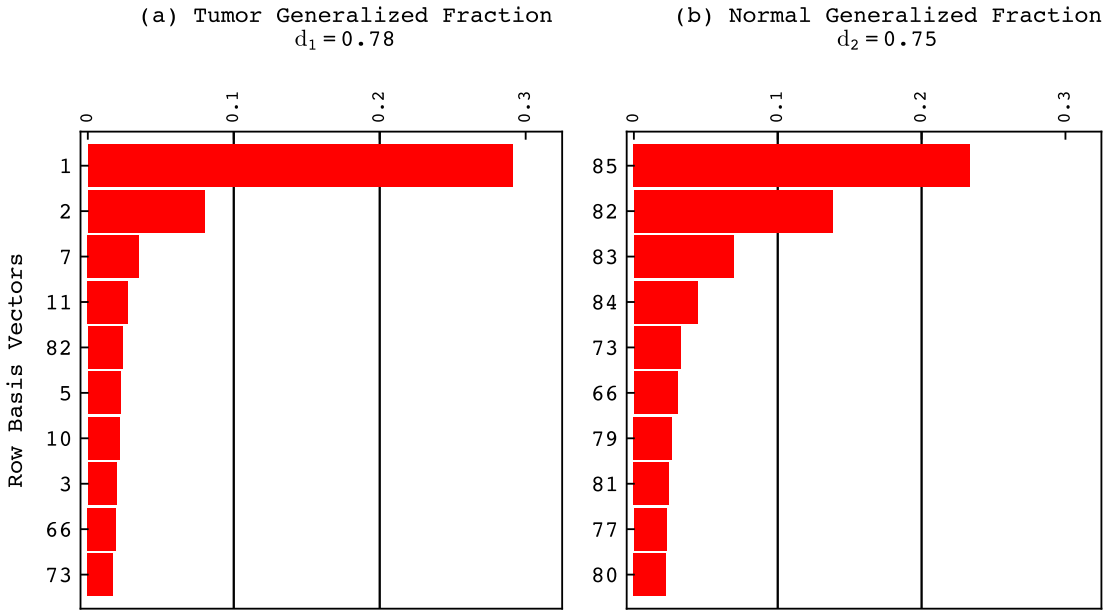


Figure 4.2. The most significant row basis vectors uncovered by the GSVD of the WGS astrocytoma tumor and normal datasets. (*a*) The 10 largest generalized fractions in the WGS astrocytoma tumor dataset are depicted in a bar chart, showing that the two most tumor-exclusive row basis vectors, i.e., the first and second, are also the first and second most significant in the tumor dataset and capture ≈29% and 8% of the information, respectively. The corresponding generalized normalized Shannon entropy is 0.78. (*b*) The 10 largest generalized fractions in the normal dataset are depicted in a bar chart, showing that the most normal-exclusive row basis vector, i.e., the 85th, is also the most significant in the normal dataset and captures ≈23% of the information. The 82nd row basis vector, which is approximately common to both datasets, is the second and fifth most significant and captures ≈14% and 2% of the information in the normal and tumor datasets, respectively.

4.4 Visualization of the Kaplan-Meier survival analyses in this case study

The extension of lifelines Kaplan-Meier survival analyses can be seen in Figure 4.3. These extensions, which are described in section 3.4 and allow more information to be displayed. The figure here matches the previously generated figures in [16]. We compared the results of p-values and hazard ratios by subtracting Mathematica p-values and hazard ratios from the corresponding ones in Python. The absolute values of the differences are less than the default machine precision $1e^{-16}$. Additionally, the number of patients and the number of events observed in each group match as well as the median survival time and survivor function between Mathematica and Python are the same for each survival analysis in Figure 4.3.

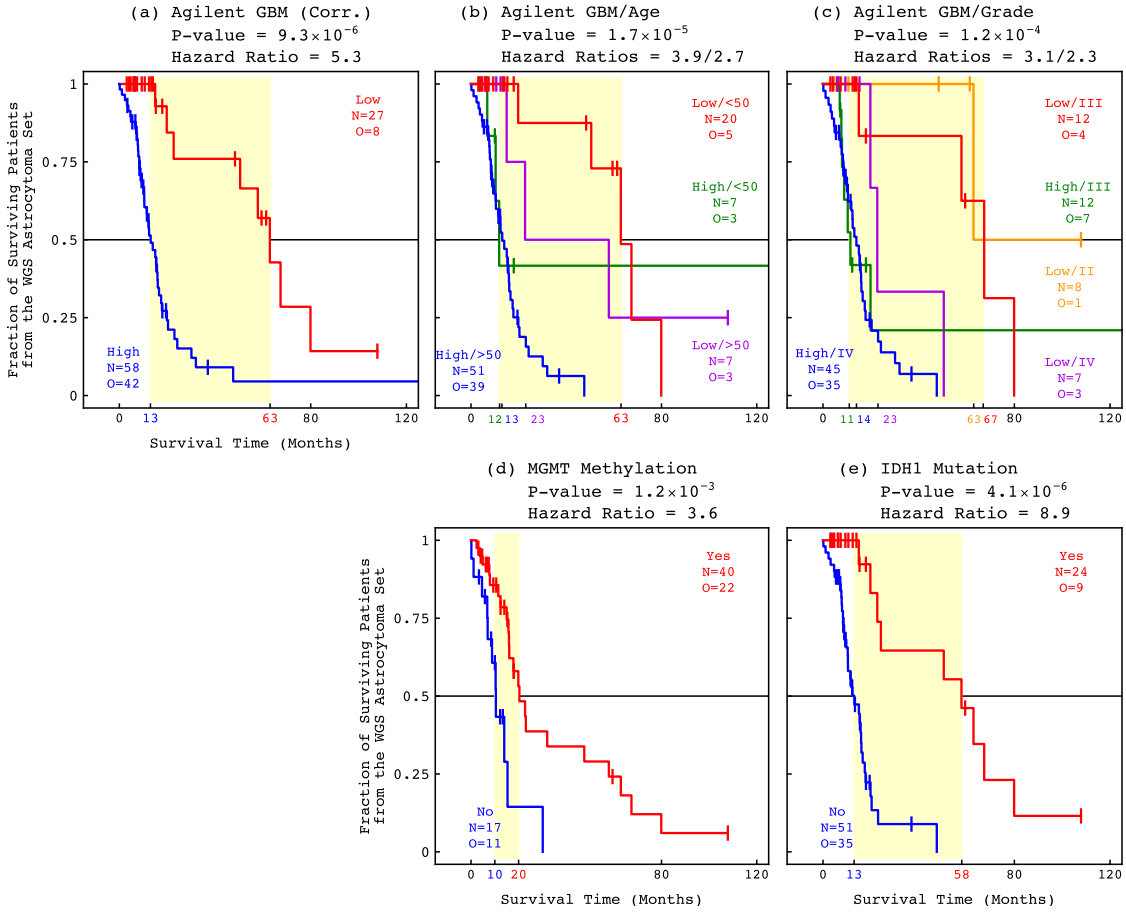


Figure 4.3. **Survival analyses of the WGS astrocytoma patients.** The classifications of the 85 patients based upon (*a*) the Agilent GBM pattern and, in addition, (*b*) age or (*c*) grade, or (*d*) *MGMT* promoter methylation or (*e*) *IDH1* mutation, are depicted in KM curves highlighting median survival time differences (yellow) with the corresponding log-rank *P*-values and Cox hazard ratios.

4.5 Visualization of the boxplots in this case study

The column and corresponding row basis vectors can be interpreted by boxplot visualization. Figures 4.4 and 4.5 were generated with the methods described in section 3.5. These images match the previously generated ones in [16]. We compared the results of p-values by subtracting Mathematica p-values from the corresponding ones in Python. The absolute values of the differences are less than the default machine precision $1e^{-16}$. The values of the median, first and third quartiles and the whiskers match between Python and Mathematica.

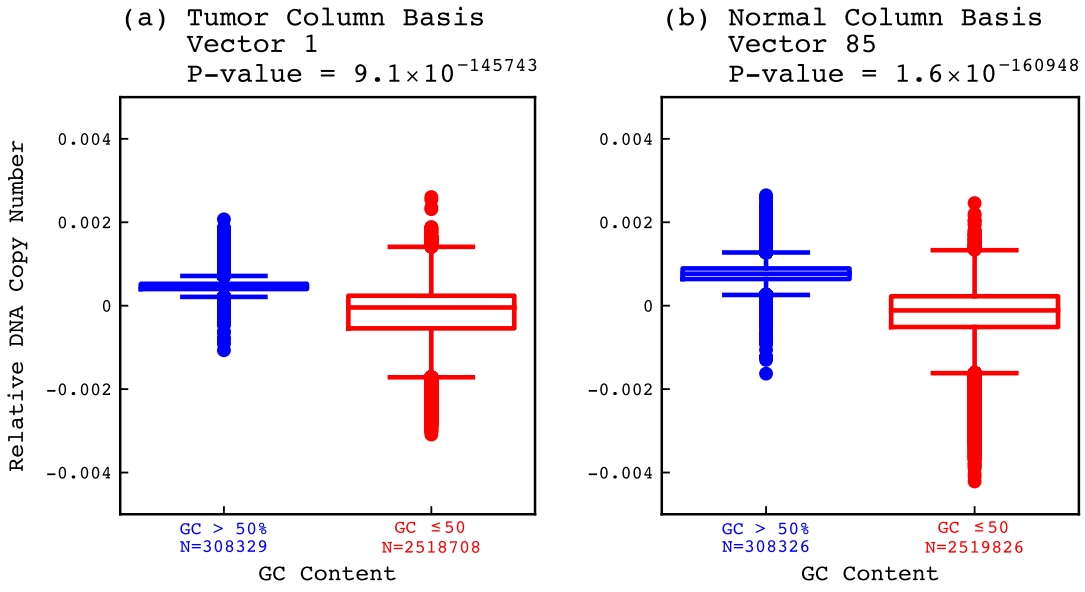


Figure 4.4. The first tumor and 85th normal column basis vectors are correlated with the fractional guanine-cytosine (GC) content across the tumor and normal genomes. The distributions of the copy numbers listed in the (*a*) first tumor and (*b*) 85th normal column basis vectors between tumor and normal bins, respectively, of >50% and ≤50% GC content are depicted in boxplots with the corresponding Mann-Whitney-Wilcoxon (MWW) *P*-values.

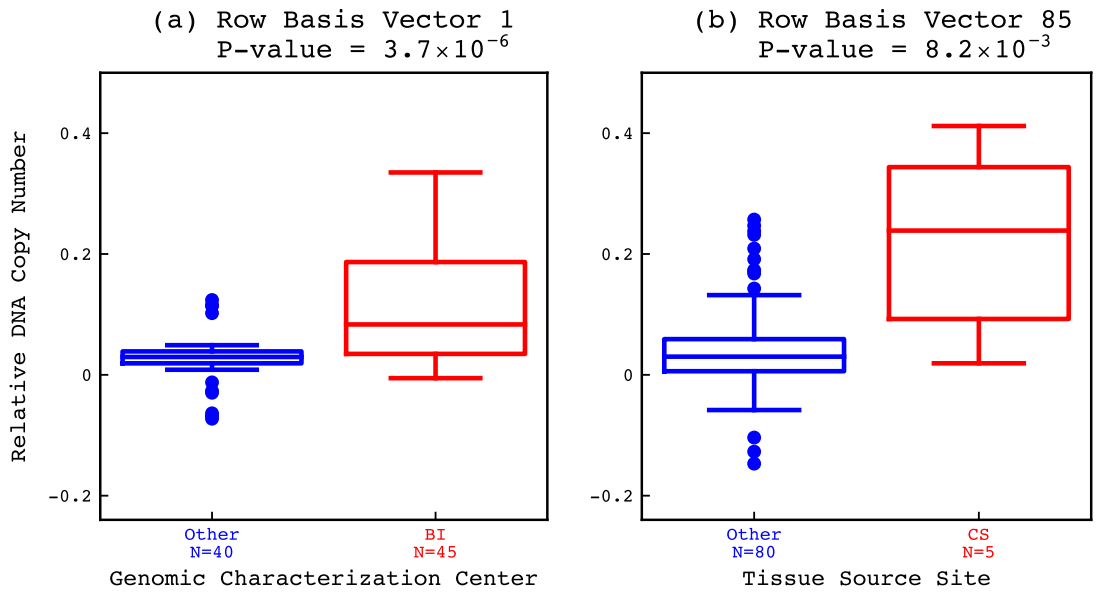


Figure 4.5. **The first and 85th row basis vectors are correlated with experimental batches.** The distributions of the copy numbers listed in the (*a*) first and (*b*) 85th row basis vectors between genomic characterization centers and tissue source sites, respectively, are depicted in boxplots with the corresponding MWW *P*-values.

4.6 Improvement of computational time of the GSVD in Python relative to Mathematica

With over four billion entries in the astrocytoma tumor dataset and astrocytoma normal dataset, Mathematica takes about 10 minutes to finish computing the GSVD, whereas Python takes only about two minutes to finish computing the GSVD. The times here include reading in the datasets and performing the GSVD. This improvement leads us to believe the toolkit developed in Python can be used more efficiently compared to previous approaches.

CHAPTER 5

CONCLUSIONS

As the case study indicated, the genomic signal processing toolkit developed in this research facilitates speedy analysis on large-scale data with no loss of accuracy. Today's datasets are enormous. Instead of waiting for a long time for the results to come out, this toolkit helps speed up the process and while maintaining accuracy. Additionally, the toolkit improves visualization over existing Python libraries. Taking advantage of existing Python libraries and extending their features, it becomes easier to interpret the data visually. Also, this toolkit is reusable and can be applied to multiple datasets. The applications of the toolkit are not limited to genomic data, but all other types of data as well. The computation and visualization of the GSVD can also be applied to many other types of data. Python was chosen to implement this toolkit, because it makes many cloud computing tasks and large datasets more efficient. It is anticipated that this toolkit can be used to analyze extremely large datasets hosted in cloud storage.

REFERENCES

[1] C. F. Van Loan, *SIAM J. Numer. Anal.* **13**, 76 (1976).

[2] C. C. Paige and M. A. Saunders, *SIAM J. Numer. Anal.* **18**, 398 (1981).

[3] S. Friedland, *SIAM J. Matrix Anal. Appl.* **27**, 434 (2005).

[4] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge, UK: Cambridge University Press (2012).

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD: Johns Hopkins University Press (2012).

[6] H. Goldstein, *Classical Mechanics*, 2nd ed. Reading, MA: Addison-Wesley (1980).

[7] O. Alter, P. O. Brown, and D. Botstein, *Proc. Natl. Acad. Sci. USA* **100**, 3351 (2003).

[8] O. Alter, G. H. Golub, P. O. Brown, and D. Botstein, in *Miami Nature Biotechnology Winter Symposium on Cell Cycle, Chromosomes and Cancer* (January 31–February 4, 2004).

[9] S. P. Ponnapalli, G. H. Golub, and O. Alter, in *Stanford University and Yahoo! Research Workshop on Algorithms for Modern Massive Datasets* (June 21–24, 2006).

[10] S. P. Ponnapalli, M. A. Saunders, C. F. Van Loan, and O. Alter, *PLoS One* **6**, e28072 (2011).

[11] P. Sankaranarayanan, T. E. Schomay, K. A. Aiello, and O. Alter, *PLoS One* **10**, e0121396 (2015).

[12] K. A. Aiello, C. A. Maughan, T. E. Schomay, S. P. Ponnapalli, H. A. Hanson, and O. Alter, in *2018 AACR Annual Meeting* (April 14–18, 2018), doi: 10.1158/1538-7445.AM2018-4267.

[13] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*. Philadelphia, PA: SIAM (1997).

[14] A. Edelman, T. A. Arias, and S. T. Smith, *SIAM J. Matrix Anal. Appl.* **20**, 303 (1998).

[15] L. M. Ewerbring and F. T. Luk, *J. Comput. Appl. Math.* **27**, 37 (1989).

[16] K. A. Aiello, S. P. Ponnapalli, and O. Alter, *APL Bioeng.* **2**, 031909 (2018).