Statistical Techniques for Neuroscientists

Y. Truong and M. Lewis

Contents

Preface

Why Data Analysis in neuroscience

The brain contains about ten bililion neurons (nerve cells) and they relay signals with each other. Some form a group of neurons each of which receives signals from a large number of other neurons, then sends off its own signals to the other neurons. One of the central problems for the neural scientists is to relate signals (electrical activity) observed from the brain to the underlying physiological processes. A variety of recording methods are being used to observe this activity in the active brain. Intracellular recordings of the membrane potentials of individual neurons, extra-cellular spike recordings from one or more individual neurons, and recordings of signals that measure the activity of ensemble of neurons either locally as the local field potential, or from larger brain regions via electroencephalography (EEG), magnetoencephalography (MEG) or functional magnetic resonance imaging (fMRI). Some recordings are very high in temporal resolution and are only possible for single, or small ensemble of neurons. Others tend to explore the spatial characteristics for the whole brain but lacking the ability to capture the real dynamic of the activity. Thus any particular choice of the recording method will be closely related to the research aims of the investigator about the mechanisms of neuronal processing.

Inspired by Hebbs influential work, and motivated by more recent physiological and anatomical findings, it is now very common to observe the activity of multiple single-neurons simultaneously in order to understand the principles of coordinated neuronal activity and its spatio-temporal scales. While the coordinated dynamics is apparent in time-resolved multiple-channel measurements among neurons and groups of neurons, methods based on EEG, MEG or fMRI for brain activation focus more on the brain regions of interest. Thus, the analysis of data from simultaneously recorded spike trains or larger brain regions (EEG or fMRI) will allow us to relate concerted activity of ensembles of neurons or brain function to behavior and cognition. Different statistical data analysis are thereby relevant to distinguish different or even complementary spatio-temporal scales. Statistical analysis of brain data is the logical follow-up to improve our understanding of the neuronal activity or network underlying information processing in the brain.

Purpose of the Book

The book aims at introducing new and useful methods for data analysis involving simultaneous recording of neuron or large cluster (brain region) of neuron activiity. The statistical estimation and tests of hypotheses are based on the likelihood principle derived from stationary point processes and time series. Algorithms and software development are given in each chapter, the main objective is to reproduce the computer simulatied results described therein.

Intended Audience

This book is intended for neural scientists who are interested in analyzing multi-channel time series data arising from brain functions or activities. It is also for statisticians who are familiar with traditional multivariate time series analysis and spatial modeling. It is specially for readers who wish to employ computing intense methods to extract important features or information directly from the data rather than relying heavily on models built upon leading cases such as linear regression or Gaussian processes. It is also for practioners or scientists who are interested in reproducible research. Basic knowledge of R and MATLAB will be essential for understanding and reproducing the numerical results.

Organization of the Book

The first part of the book deals with the traditional multivariate time series analysis applied to the context of multichannel spike trains and fMRI using respectively the probability structures or likelihood associated with time to fire and discrete Fourier transforms (DFT) of point processes. The second part introduces a relatively new way of statistical spatio-temporal modelling for fMRI and EEG data analysis.

Some useful facts and computing tools are collected in the appendices for quick reviews or tutorials in using the software described in each chapter.

How to Read This Book

For multi-channel spike train or fMRI voxel time series data analysis, the reader should start with Part One of the book. For spatio-temporal modeling, start from Part Two. Instructions or tutorials for setting up the computing environment and our software packages are given in the appendices at the end of the book

Software Download

Software packages can be inquired by contacting the first Editor (YT) or they will be made available online.

Acknowledgements

We wish to thank Professor David R. Brillinger for his vision of applications of time series to neural science, he has been one of the pioneers in the field long before we started to analyze neural spike train and fMRI data. We are very grateful for his teaching and research contributions to statistical computing and neural science. We also wish to acknowledge the immense constributions of Dr. X. Huang for her continuing supports of this project; her insights, energy and the human brain data from her lab have been instrumental in shaping up our statistical approaches to fMRI data analysis. The book was initiated by Dr. Sid Simon, the Editor of this series, his encouragement and patience have led to countless improvements in preparing this research monograph. We wish to thank Barbara Norwitz, Editor of the book series at Taylor and Francis, for her patience and understanding of my family issues during the course of preparing the book. It has been a pleasure to work with her. We also thank Jill Jurgensen for lending her technical support on the LaTeX package and template for this book series. Part of this project has been supported by National Science Foundation NSF DMS-1106962, National Institute of Child Health and Human Development NIH HD003110-45 (PI: Dr. Piven), and UNC Clinical Translation Science Award - Biostatistics Core TR000083-05.

A Discrete Fourier Transform

CONTENTS

The definitions of *Discrete Fourier Transform* (DFT) , multivariate normal and its complex extension will be given in this appendex. See also Chapter 3 and books on time series analysis [1, 2].

A.1 DISCRETE FOURIER TRANSFORM (DFT)

The *Discrete Fourier Transform* (DFT) of a vector-valued time series $\mathbf{Y}_t$, $t = 0, 1, \ldots, T-1$, is given by

$$\varphi_Y(k) = \sum_{t=0}^{T-1} \mathbf{Y}_t \exp(-\mathrm{i}2\pi kt/T), \quad k = 0, \pm 1, \pm 2, \ldots, \pm[(T-1)/2]. \tag{A.1}$$

Under reasonable assumptions, it can be shown that $\varphi_Y(k), k = 0, \pm 1, \ldots$ are complex normal [1].

In numerical computational problems involving `R` or `MATLAB`, the `fft` function computes the DFT of a time series using a fast Fourier transform (FFT) algorithm.

A.2 MULTIVARIATE NORMAL DISTRIBUTION

In probability theory and statistics, the multivariate normal distribution or multivariate Gaussian distribution, is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. Given a d-dimensional mean vector

$$\mu = (\mu_1, \mu_2, \ldots, \mu_d)$$

and $d \times d$ (symmetric, positive-definite) covariance matrix

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{pmatrix}, \qquad \sigma_{ij} \in \mathbb{R}, \quad i, j = 1, 2, \ldots, d,$$

a d-dimensional random vector $\mathbf{X} = (x_1, x_2, \ldots, x_d)$ is said to have a multivariate normal distribution , abbreviated by $\mathbf{X} \sim N_d(\mu, \Sigma)$, iff $Y = \mathbf{a}'\mathbf{X} = a_1X_1 + \cdots + a_dX_d$ is normal with mean $\mathbf{a}'\mu$ and variance $\mathbf{a}'\Sigma\mathbf{a}$.

It can be shown that the probability density function of $\mathbf{X}$ is given by

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right), \quad \mathbf{x} \in \mathbb{R}^d.$$

A.3 COMPLEX NORMAL DISTRIBUTION

Definition 2 *The Complex Multivariate Normal Distribution. If $\Sigma = \Sigma_1 + \mathrm{i}\Sigma_2$ is a complex-valued $m \times m$ matrix such that $\Sigma = \Sigma^\tau$ and $\mathbf{a}^\tau\Sigma a \geq 0$ for all $\mathbf{a} \in \mathbb{C}^m$, then we say that $\mathbf{Y} = \mathbf{Y}_1 + \mathrm{i}\mathbf{Y}_2$ is a complex-valued multivariate normal random vector with mean $\mu = \mu_1 + \mathrm{i}\mu_2$ and covariance matrix Σ if*

$$\begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \frac{1}{2}\begin{bmatrix} \Sigma_1 & -\Sigma_2 \\ \Sigma_2 & \Sigma_1 \end{bmatrix}\right). \tag{A.2}$$

We then write $\mathbf{Y} \sim N_m^C(\mu, \Sigma)$.

Bibliography

1. D.R. Brillinger. *Time series*. Holden-Day, 1981.
2. P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods*. Springer, second edition, 1991.

B The R Software Package

CONTENTS

B.1 SOFTWARE INFORMATION

The main tools we introduce in this appendix are the R language [4] and the `knitr` package [7, 8]. These are excellent tools for reproducible research and both packages are available on *Comprehensive R Archive Network* (CRAN) as free and open-source software. You may download them from the CRAN website:

`https://cran.r-project.org/`

or any of its mirrors, such as

`http://cran.rstudio.com`

It is also useful to consider using *RStudio* [5]:

`https://www.rstudio.com/`

B.2 INSTALLATION

Instructions for installation in `Windows`, `Mac OSX` and `Linux` are available in the above websites, and they are very straightforward to install.

B.2.1 WINDOWS INSTALLATION

For Windows, follow these steps:

1. Navigate to `https://cran.r-project.org/`
2. Click on `Download R for Windows`
3. Execute the downloaded `.exe` to install R.

B.2.2 LINUX-UBUNTU INSTALLATION

For Linux-Ubuntu, follow these steps:

1. Navigate to `https://cran.r-project.org/`
2. Click on `Download R for Linux` and select `ubuntu`.
3. In the `/etc/apt/sources.list` file, add the CRAN mirror entry.
4. Download and update the package lists from the repositories using the `sudo apt-get update` command.
5. Install R system using the `sudo apt-get install r-base` command.

B.2.3 LINUX-RHEL/CENTOS INSTALLATION

For Linux-RHEL/CentOS, follow these steps:

1. Navigate to `https://cran.r-project.org/`
2. Click on `Download R for Linux` and select `Red Hat OS`.
3. Download the `R-*core-*.rpm` file.
4. Install the `.rpm` package using the `rpm -ivh R-*core-*.rpm` command.
5. Install R system using `sudo yum install R`.

B.2.4 MAC OS X INSTALLATION

For Mac, follow these steps:

1. Navigate to `https://cran.r-project.org/`
2. Click on `Download R for (Mac) OS X` and
3. Download the `R-*.pkg` file.
4. Click on `tools` or go to `https://cran.r-project.org/bin/macosx/tools`, download `gfortran-*.dmg` and `tcltk-*.dmg`.
5. Install the `R-*.pkg` file.
6. Install the `gfortran-*.dmg` and `tcltk-*.dmg` files.

Under Mac OS X, some R packages do not come with binaries, and so `gfortran` would come in handy for compiling from the sources.

After installing the `base R` package, it is advisable to install *RStudio*, which is a powerful and intuitive Integrated Development Environment (IDE) for R.

B.2.5 INSTALLING RSTUDIO

To install *RStudio*, follow the steps below:

1. Navigate to `https://www.rstudio.com/products/rstudio/download/`
2. Download the latest version of *RStudio* for your operating system.
3. Execute the installer file and install *RStudio*.

B.2.6 INSTALLING R PACKAGES, TASK VIEWS

There are several thousands of `R` packages, consisting collections of `R` functions, data, and compiled code. Browse for packages by name, author, title, and published date or find packages organized by task at:

`https://cran.r-project.org/web/packages/`

There are base packages (which come with `R` automatically), and contributed packages (which must be downloaded for installation).

There are instructions for installing these packages. For starters, we recommend using *RStudio*.

It is also very useful to check out `Task Views`:

`https://mran.revolutionanalytics.com/taskview/`.

These are guides on CRAN that group sets of `R` packages and functions by type of analysis, fields, or methodologies. Instructions for installing `Task Views` is available at:

`https://mran.revolutionanalytics.com/rpackages/`.

B.3 DOCUMENTATION

There is a section for `R Manuals`, browse them at:

`https://cran.r-project.org/manuals.html`

These documents are available in pdf, html or epub. We recommend *An Introduction to R*

`https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf`

to get started. Most of the examples described there can be run and reproduced using *RStudio*.

Some useful free books are available at

`https://cran.r-project.org/other-docs.html`.

For starters, R can be approached as a programming language [3], or as a general statistical tool for applications [2]. The latter has a very informative website:

```
http://www.statmethods.net/
```

B.4 TUTORIALS

R by itself has the capabilities necessary to gather data, analyze it, and, with a little help from *knitr*/*rmarkdown* and markup languages, present results in a way that is highly reproducible. *RStudio* will do all of these things, it will simplify many of them and navigate through them more easily. It also is a happy medium between R's text-based interface and a pure GUI.

To run the examples in this book, it is necessarly to treat each chapter as a project as there are R functions written specifically for that chapter, and this is another reason for using *RStudio*as it has an excellent project management feature. Moreover, *RStudio* is a highly advanced text editor for R, it has R's help system, version control, packages management, and many other useful features to expedite the program development process in a single application. *RStudio* does not perform any computation or statistical analysis; it only makes it easier for you to perform such tasks with R. Most importantly, *RStudio* offers many facilities that make working reproducibly a lot easier. A good place to learn more about it is [6].

We usually carry out the statistical analysis using R by grouping files and data into a so called working directory, and files containing R functions will have an extension R. For example, collect all the `R` functions in Chapter 3 and put them in a folder or directory called `ch03`. The file `BIV.R` contains one or more `R` functions written by us. Now start *RStudio*and select `New Project` under `File`, then select `Existing Directory` in the pop-up panel `Create project from:`. Now navigate to the directory `ch03` and click `Create Project`. After this, *RStudio*will manage the files and data created in the project.

When *RStudio* is first started, there will be three panels: A long one on the left is the Console where R commands can be entered to conduct statistical operations. On the right, the top panel records the `Workspace` and `History`, the bottom panel has several tabs. Click on `File` to display the contents of the working or project directory. The `Plot` is for graphical display of R's many excellent graphical procedures for data analysis. The `Packages` shows what R packages have been installed in your system. Here you can easily install or update packages. The next tab is `Help` where all the R help files are kept, you can search them by entering the command name or some key words.

To call the functions in `BIV.R` into R, issue the R command below into the *RStudio* Console:

```
> source("BIV.R")
```

Instead of typing the commands directly into the `Console` window, you can use *RStudio*to create a file and enter all the commands there. An example is given in the file `prep.R`. Each line or group of lines in this file can be run in `R` by highlighting them, followed with a click on the `Run` icon at the top of that window. The results should be reproduced through `R`.

R comes with base packages installed and they are loaded when R is launched. For example, the default base packages loaded at startup are

```
"datasets"  "utils"    "grDevices" "graphics"  "stats"    "methods"
```

The command to get the base packages is:

```
> getOption("defaultPackages")
```

Projects such as those in Chapters 2, 6, or 7 will require additional `R` packages to be installed. These are called contributed packages. For example, Chapter 2 will call upon the R package `polspline`. This can be easily installed by clicking the `Packages` tab in the lower right panel of *RStudio*, and select `Install`. Enter `polspline` into the the pop-up panel, then wait for *RStudio* to complete the installation. Alternatively, it can be installed by issuing the following command

```
> install.packages("polspline", depends = TRUE)
```

into the `RStudio Console`. After the package has been installed, loaded it into R by checking the box next to the package name: `polspline` listed in the `Packages` panel, or enter the following command into the Console:

```
> library(polspline)
```

To learn more about the package, click the name in the `Packages` panel. Before using any package, it may be a good idea to learn more about it by visiting its website. The package `polspline` is located at:

```
https://cran.r-project.org/web/packages/polspline/index.html
```

Next, repeat the steps described above to install the package `coloredICA`:

```
https://cran.r-project.org/web/packages/coloredICA/index.html
```

These packages will be stored in the memory as long as *RStudio* (or R) is up and running. To free up the memory space, uncheck that package if it is no longer needed in the statistical analysis. If it is needed again in the analysis, just check it. Also, all the loaded packages will be removed from the memory once *RStudio* has ended. After re-starting *RStudio*, the installed packages can be re-loaded into the memory by checking the boxes next to the package names. A list of installed packages is available for viewing in the `Packages` tab in *RStudio*.

After a project is completed, the next step is to copy and paste the results into a document in various formats (LATEX to pdf, Markdown to HTML). The R package *knitr* is ideal for this purpose [8, 7]. Install it via *RStudio* `Packages/Install` or using the `Console` with

```
> install.packages("knitr")
```

Together with a typesetting program such as $\TeX$/$\LaTeX$, R, *RStudio* and *knitr* form a very powerful tool for reproducible research. Many excellent examples are located at:

`http://yihui.name/knitr/`

and

`https://github.com/christophergandrud/Rep-Res-Book`

The latter has provided the source to reproduce the whole book [1].

Bibliography

1. Christopher Gandrud. *Reproducible Research with R and RStudio*. CRC Press, 2013.
2. Robert Kabacoff. *R in Action*. Manning Publications Co., 2014.
3. Norman Matloff. *The art of R programming: A tour of statistical software design*. No Starch Press, 2011.
4. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.
5. RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2015. URL `http://www.rstudio.com/`.
6. Mark PJ Van der Loo. *Learning RStudio for R statistical computing*. Packt Publishing Ltd, 2012.
7. Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL `http://yihui.name/knitr/`. ISBN 978-1498716963.
8. Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2015. URL `http://yihui.name/knitr/`. R package.

C MATLAB

CONTENTS

C.1 GETTING STARTED

`MATLAB` (MATrix LABoratory) is a powerful tool for neurocomputing. The best way to get started is to launch the software and familiarize it by browsing the tutorials, which also contain a few short videos. After launching `MATLAB`, it is waiting for your instructions which can be given next to `>>` with a blinking bar `|` in the (center) panel called the `Command Window`. Start your tutorials by clicking `Getting Started` in:

```
New to MATLAB? Watch this Video, see Examples, or read Getting Started
```

Alternatively, after launching `MATLAB`, select `HOME/Help/Examples/Getting Started`. Next, learn more about the `MATLAB Desktop` by watching the video:

```
Working in The Development Environment.
```

After these videos, get a more comprehensive introduction to `MATLAB` by tabbing `Help/Documentation` and browse the topics there.

By now there are thousands of book on `MATLAB`. We specifically mention three: A short *Learning MATLAB* [1], a mathematically oriented text *MATLAB Guide* [2] and an introductory text in numerical methods, Matlab, and technical computing [3], which is also available at

```
https://www.mathworks.com/moler/chapters.html
```

`Experiments with MATLAB` is a collection of mathematical and computational projects:

```
http://www.mathworks.com/moler/exm
```

Finally, worth mentioning is a book on `MATLAB`for neuroscientists [4], which can be served as an extension of [3].

A few important things about `MATLAB`:

1. Use the up- or down-arrow key to cycle through commands that have been issued.
2. Type `edit` in the `Command Window` to launch the `MATLAB` Editor for writing code.
3. Run your code step by step to debug.
4. Use the `Window` in the `Tool Bar` to locate windows related to the current project. This is also useful for showing the window of interest.
5. Save graphs from the `Figure Window` to the format of your choice.
6. Download and install `MATLAB` project manager from

   `http://www.mathworks.com/matlabcentral/fileexchange/41585-matlab-project-manager`

C.2 TUTORIALS

Download `Experiments with MATLAB` (`http://www.mathworks.com/moler/exm`). Create a directory called `exm` and unpack the downloaded file into it.

`http://www.mathworks.com/moler/exm`

Launch `MATLAB`, enter the command `pathtool`

```
>> help pathtool
>> pathtool
```

and navigate to the directory `exm` created above.

Assume that the `projects` was properly installed and is in the `MATLAB` search path. Check it out by entering the following command:

```
>> help projects
... ...
Examples:
   projects list
   projects save myProject
   projects close
   projects load default
   projects rename myProject myLibrary

All projects are stored in the %userpath%/projects.mat. This file with
empty "default" project is created at the first run of the script. If
%userpath% is empty, the script will execute userpath('reset').

First project always has name "default"
```

If it is not installed correctly or if `projects` is not in `MATLAB` path, an error message will be printed. Suppose the above message is on the screen (ie. `MATLAB` Command Window), enter the following

```
>> projects
List of available projects:
-> 1: default
```

Since there are no named projects other than the current one, it is called the default. Now enter

```
>> projects save exm
Project "exm" was saved

>> projects
List of available projects:
   1: default
-> 2: exm
```

You just added a new project to MATLAB. To create another one, repeat the above steps after downloading

`https://www.mathworks.com/moler/ncm.zip`

Now MATLAB should have your two projects listed:

```
>> projects
List of available projects:
   1: default
-> 2: exm
   3: ncm
```

Using this function to manage your projects is essential though it may seem unnecessary at the beginning. Soon you will appreciate the project manager after checking through the MATLAB functions listed at the end of each chapter in this book. By creating a project for each chapter and examining the steps in each function, perhaps two or three files were opened in the editor in the project chapter2. If MATLAB is needed for something else outside the book chapters, close the project:

```
>> project close
```

so that the other task can be accomplished. After this, you can return to the previous project via

```
>> project load chapter2
```

The manager will remember where the exact settings of project A and the files opened in the MATLAB editor.

Bibliography

1. Tobin A Driscoll. *Learning MAtlAB*. SIAM, 2009.
2. Nicholas J Higham. *The matrix computation toolbox for MATLAB (version 1.0)*. Manchester Centre for Computational Mathematics, 2002.
3. Cleve B Moler. *Numerical Computing with MATLAB: Revised Reprint*. SIAM, 2008.
4. Pascal Wallisch, Michael E Lusignan, Marc D Benayoun, Tanya I Baker, Adam Seth Dickey, and Nicholas G Hatsopoulos. *MATLAB for neuroscientists: an introduction to scientific computing in MATLAB*. Academic Press, 2014.

D Singular Value Decomposition

CONTENTS

In matrix computation, *singular value decomposition* (SVD) is a factorization of a matrix into a product of special matrices. There are many important applicatons in numerical analysis, signal processing and statistics [1]. It is an essential tool for dimension reduction in the methodology for ICA.

D.1 SINGULAR VALUE DECOMPOSITION (SVD)

Suppose $\mathbf{Y} \in \mathbb{R}^{m\times n}$ is a $m \times n$ matrix whose entries are real numbers. Then there exist orthogonal matrices

$$\mathbf{U} \in \mathbb{R}^{m\times m}, \qquad \mathbf{V} \in \mathbb{R}^{n\times n}$$

such that

$$\mathbf{U}'\mathbf{Y}\mathbf{V} = \mathrm{diag}(\sigma_1, \ldots, \sigma_p), \qquad p = \min\{m, n\}, \tag{D.1}$$

where $\mathrm{diag}(\sigma_1, \ldots, \sigma_p)$ is a diagonal matrix of non-negative *singular values* and

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0.$$

Rewrite (D.1) so that

$$\mathbf{Y} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_p)\,\mathbf{V}'.$$

This is a well-known factorization and is referred to as a *singular value decomposition* (SVD) of $\mathbf{Y}$ [1]. The SVD also holds for matrices with complex entries. In this case, $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices.

D.2 SVD IN MATLAB

In `MATLAB`, SVD of of a matrix Y is achieved by issuing the command:

```
>> [U,S,V] = svd(Y);
```

Below is an example of the SVD of a small, square matrix is provided by one of the test matrices from the `MATLAB` gallery.

```
>> Y = gallery(3)

Y =

  -149   -50  -154
   537   180   546
   -27    -9   -25

>> [U,S,V] = svd(Y);
>> U

U =

   -0.2691   -0.6798    0.6822
    0.9620   -0.1557    0.2243
   -0.0463    0.7167    0.6959

>> S

S =

  817.7597         0         0
         0    2.4750         0
         0         0    0.0030

>> V

V =

    0.6823   -0.6671    0.2990
    0.2287   -0.1937   -0.9540
    0.6944    0.7193    0.0204
```

The expression `U*S*V’` generates the original matrix to within roundoff error.

```
>> U*S*V’

ans =

 -149.0000  -50.0000 -154.0000
  537.0000  180.0000  546.0000
  -27.0000   -9.0000  -25.0000
```

D.3 SVD IN R

In R, SVD of a matrix Y is given by the command:

```
> svd(Y)
```

Below is a R example of the SVD using the above square matrix Y.

```
> Y <- matrix(c(-149, 537, -27, -50, 180, -9, -154, 546, -25), ncol=3)
> svd(Y)
$d
[1] 8.177597e+02 2.474974e+00 2.964523e-03

$u
            [,1]       [,2]      [,3]
[1,] -0.26906707 -0.6798212 0.6822361
[2,]  0.96200923 -0.1556695 0.2242883
[3,] -0.04627257  0.7166660 0.6958798

$v
          [,1]       [,2]        [,3]
[1,] 0.6822779 -0.6671414  0.29903068
[2,] 0.2287120 -0.1937185 -0.95402513
[3,] 0.6943974  0.7193021  0.02041339
```

Here `d` is the diagonal matrix D in the SVD of Y. It agrees with the `MATLAB` version.

Bibliography

1. Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.