The R Programs to accompany Fundamental Probability: A Computational Approach

General Remarks

As discussed in the Preface to the book, the popularity, functionality, and ease-of-use of Matlab made it a natural choice for the coding of the programming examples. The R language has all of these features, but it is also free, with an ever-growing user base in academia and industry. As such, we decided to provide the reader the option of which to use. The book's web page provides the R translations of virtually all of the Matlab programs and functions used in the book. For example, the function on page 58, Listing 2.3, called "threegirls" is the Matlab file `threegirls.m`, and the R file `threegirls.r`.

Also included are the handful of short code segments throughout the book. They are in the files called `pageXX.m` and `pageXX.r`, where XX denotes the page in the book it appears on.

While the two languages, Matlab and R, have important similarities, there are enough differences in syntax, built-in functions, and data structures that make a translation exercise of all the functions rather time-consuming and occasionally (pleasantly) challenging. I wish to thank Sergey Goriatchev, who has competently and singlehandedly completed this sizable task and written the remainder of this document.

R Intallation and Documentation

R is a computer language and an environment developed primarily for statistical computing. The R language is similar to the S language, developed at Bell Laboratories. R's similarity to S allows one to migrate to the commercially supported S-Plus software if desired. Like Matlab, R is a very flexible system for data analysis that can be extended as needed with the help of numerous additional packages. These packages are being constantly developed and maintained by the diverse, wide-spread, and very helpful R–user community. Many, but not all, of these packages reside on the R homepage, and can be easily installed (and updated) from the main GUI. There are several hundred packages available (a number of them are automatically included in the base distribution of R), offering state-of-the-art abilities for free.

On the R homepage, CRAN, (`http://cran.r-project.org/`) there are instructions for downloading a base distribution of R for your operating system (Windows, Mac, Linux, or Unix), a description of the different add-on packages, and various manuals for the R language.

There is quite an abundance of documentation on R available at CRAN. The student is advised to download the official manual **An Introduction to R**. Under *Documentation* on the CRAN homepage, the user may click on *Contributed* to browse through R-user supplied documentation and tutorials. There is a very wide choice of manuals in different languages. Out of personal experience, the tutorials **Using R for Data Analysis and Graphics - Introduction, Examples and Commentary** by John Maindonald and **Simple R** by John Versani are very comprehensive and purposeful introductions to R. **Practical Regression and Anova using R** by Julian Faraway is a fabulous tutorial on fitting and diagnosing linear models in R. In fact, all three of these tutorials have been turned into textbooks. For econometrics, there is a concise but very good tutorial **Econometrics in R** by Grant Farnsworth, and the forthcoming book by Christian Kleiber and Achim Zeileis, **Applied Econometrics with R**. There are also several reference cards available for downloading, in particular **R reference card** by Tom Short.

R is not only a professional statistical system, it also provides very powerful, highly flexible and fully programmable graphics utilities. A recent addition to the growing literature on R is the excellent book **R Graphics** written by one of the main developers of R graphics systems, Paul Murrell.

Once the base distribution of R has been installed, the R program files (scripts) we provide can be downloaded and placed into the R working directory. To find out what the working directory is, type `getwd()` at the R prompt. To test any function, first open a script in R GUI, then load the function code by selecting all the code in the script with Ctrl+A and running it at the R prompt with Ctrl+R.

Remarks regarding coding in Matlab and R

Below are a few remarks about language differences that came to light during the recoding of functions written in Matlab into R. Also, specific use of code is often explained in the program scripts themselves. The explanations are presented within the code as comments that begin with # sign, or at the end of each particular script as Notes. To read these explanations, open a script inside R GUI.

Because of the specifics of each programming language, R and Matlab code for each particular program are sometimes not “one-to-one”, in the sense that the only difference between the programs is the cosmetic syntax of the languages. For example, in program `rowofbirthdays.r`, the R and Matlab code are not one-to-one, because R allows “backward” unincremented looping, while Matlab does not(explained in more detail in one of the remarks below).

Another issue which has influenced the R code is that we tried to make the output of the R program resemble, as close as reasonably possible, that of the corresponding

Matlab program. This was done mainly for "education purposes", though in practice, an R-user would just use the default style of output in R.

The new user should keep in mind that it takes time to learn the intricacies and specifics of a programming language. When going through the programs, keep in mind that our code was not always designed to be optimally efficient or elegant, and the user is recommended to construct alternative programs which improve upon ours by using more advanced features of R.

- A very useful function in R is `RSiteSearch(‘‘...’’)`, where instead of the dots you type your key word of interest, or a phrase. For this function to work you have to be connected to the internet, because it searches online mail lists and displays documents, R-help files, and R function descriptions containing the specified key words.
- Because the R language has a reserved, built-in function "c", the translation of the Matlab program `c.m` for computing the binomial coefficient ("n choose k") is called `mychoose.r`.
- The Matlab user-defined function `permvec(N)`, which is used in, among others, `permvecsim.m` is replaced in R with built-in function `sample(N,N)`.
- Matlab function `rand('seed', 999)` is similar to R function `set.seed(999)` (see `egg.r`).
- In Matlab, if, say, $m = 5$, code `(0:m+2)` produces a sequence from 0 to 7, while in R it produces a sequence from 2 to 7, unless the R code is written `(0:(m+2))` (see `couplesaroundtable.r`).
- Matlab-type of indexing, for example `M[m:m+1,]`, will not work in R. You have to write something like `M[c(m,m+1),]`.
- In Matlab, for x a positive number, `log(-x)` returns a complex number, while in R, `log(-x)` produces NaN.
- Histograms are plotted differently in Matlab and R. The reason for this is the coding of bins. In Matlab, histograms are plotted with the following code:

  ```
  [histcount, histgrd] = hist(Y,X)
  ```

  where X is a vector. It returns the distribution of Y among bins with centers specified by X. In R, one writes

```
g <- hist(Y,breaks=...,)
```

and this returns an object g of class “histogram”, and the argument `breaks=...` in the function definition could be, among other things, a number of breaks or a vector giving the breakpoints between histogram cells. To get the same number of elements in a bin in Matlab and in R, the breaks in the R histogram function must be adjusted to match those of Matlab. How this is done is shown in function `egg.r`. The Matlab code

```
[histcount, histgrd] = hist(needed, range(needed)+1);
h1 = bar(histgrd, histcount);
```

is coded in R as

```
g<-hist(needed,breaks=seq(min(needed),max(needed),
      by=((max(needed)-min(needed))/((max(needed)-min(needed))+1))),
      plot=FALSE)
names(g$counts)<-round(g$mids, digits=0)
a<-barplot(g$counts, ...)
```

The object g returned by the R function `hist` is a list with a number of components that can be accessed using dollar sign notation. That way we extract `g$mids` which corresponds to the above `histgrd` in the Matlab code, and `g$counts`, which corresponds to `histcount` in the Matlab code.

- Tabulating functions differ in Matlab and R. Matlab function `tabulate(x)` returns a matrix, TABLE. The first column of TABLE contains the unique values of x. The second is the number of instances of each value. The last column contains the percentage of each value. If the elements of x are non-negative integers, then the output includes 0 counts for any integers that are between 1 and $\max(x)$ but do not appear in x. The R function `table(x)` produces an object of class “table”, which is, in principle, a named vector of instances of unique(!) values of x. The names are the unique values of x. When percentages are needed, (especially when these are going to be plotted and the plots in Matlab and R should be similar), they should be produced by hand, and the code for this can be seen, for instance, in `banach.r` or `surrounded.r`. When plotting is involved, to replicate the output of the Matlab tabulating function, we basically create a new, expanded named vector which also includes 0 counts. Then we create a new vector containing percentage of each value by dividing the newly created expanded named vector by the sum of instances of all values.

Further, in the Matlab program `rowofbirthdays.m`, the code snippet

```
tt = tabulate(bday);
days = find($tt(:,2)>0);
```

returns days, while the direct recoding in R would return counts. To get days in R we need to take the name attribute of the table and convert it to integers, like

```
days <- as.integer(names(table(days)))
```

- Interquartile range functions in Matlab and R produce different results because they are based on different ways of calculating quantiles. R contains nine different quantile algorithms (type `?quantile` at R prompt to read about them). The default for R `IQR()` function is algorithm type 7, while in Matlab, the R-equivalent of type 5 is used. In R, to get results similar to those of Matlab, one has to code the `IQR()` function using the `quantile()` function. This is done in program `kerngau.r` as

  ```
  quantile(x, 0.75, type=5) - quantile(x, 0.25, type=5)
  ```

- The Matlab function `unidrnd(X, A, B)` produces an array $A \times B$ of random numbers chosen uniformly from set (*1,2,3...*X). This is achieved in R with function `sample(X, Z, replace=TRUE)` which produces an i.i.d. sample (`...replace=TRUE`) of size Z (=sum($A \times B$)) taken from set (*1,2,3...*X). Similarly, Matlab function `binornd(1,p,A,1)` is replicated in R with `rbinom(kids, 1,p)`.

- Matlab code of type

  ```
  if nargin < 3, kids = 7; end
  ```

  is not necessary in R, as function argument defaults can be given in the function specification (for instance, see program `threegirls.r`).

- In R, the output of the function is the last command evaluated. In Matlab a function can have several outputs. If several outputs are required in R, one has to create a list of outputs and put it as the last command in function code. For a clear example of how this can be done, check and run the code for program `holst.r` and `negbin3.r`.

- In R, numeric values are coerced to strings automatically, where needed. For an example, compare code in functions `leuk.m` and `leuk.r`. In Matlab you write

  ```
  text(pos1, pos2, int2str(s), 'fontsize', 18, 'Color', [0 1 0])
  ```

  while in R, this is written as

  ```
  text(pos1,pos2,labels=s, col="royalblue")
  ```

- R implements negative indexing, which means that command `x[-1]` returns all but the first value of vector x. How negative indexing may simplify Matlab code can be observed in program `hypergeometricsim.r`.

- Backward looping works in R but not in Matlab (though, of course, something like `for j=5:-1:1...` would work). For example, compare programs `gani.m` and `gani.r`. In Matlab, the for-loop `for j=2:m...` does not increment if $m < 2$. The execution jumps to

  ```
  b(i) = any(q==m)
  ```

  bypassing the above–mentioned for–loop. In R, loop `for(j in 2:m)...` works even when $m < 2$, but then matrix dimensions are exceeded, an NA value is returned, and logical comparison fails. We rewrite Matlab code in R by including the if–else loop

  ```
  if(m<2)
          if(sum(q==m)>0) {b(i)=1}
          else {"for-loop and b(i) evaluation"}
  ```