R Language Fundamentals

Data Frames

Steven Buechler

Department of Mathematics
276B Hurley Hall; 1-6233

Fall, 2007

Tabular Data

Frequently, experimental data are held in tables, an Excel worksheet, for example. Naturally, *R* has very robust structures for holding tabular data, including importing spreadsheets and saving to CSV files.

Outline

Matrix

a square array of numbers

The entries in a matrix `X` are arranged in rows and columns. Think of it as a two-dimensional version of a numeric vector. `X` is $n \times m$ if it has n rows and m columns. Create a 3×4 matrix all of whose entries are 0:

```
> X <- matrix(0, nrow = 3, ncol = 4)
> X
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
> dim(X)
[1] 3 4
```

Dimension, `dim(X)`, is an integer vector giving the number of rows and columns.

Matrix Entries, Rows, ...

A more interesting matrix:

```
> Y <- matrix(1:12, nrow = 3, ncol = 4)
> Y
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> Y[1, 3]
[1] 7
> Y[1, ]
[1]  1  4  7 10
> Y[, 2]
[1] 4 5 6
```

Matrix=Vector with Dimension

```
> x <- 1:15
> dim(x) <- c(3, 5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
> class(x)
[1] "matrix"
```

Submatrices

```
> Z <- X[1:2, 3:4]
> Z
     [,1] [,2]
[1,]    0    0
[2,]    0    0
```

Matrix Subsetting

There are no (few) surprises.

```
> x <- Y[1, ]
> class(x)
```

```
[1] "integer"
```

Matrix Assignment

There are no (few) surprises.

```
> X[1, 3] <- 1
> X[, 1] <- c(-1, -2, -3)
> X
```

```
     [,1] [,2] [,3] [,4]
[1,]   -1    0    1    0
[2,]   -2    0    0    0
[3,]   -3    0    0    0
```

```
> X[, 4] <- 2
> X
```

```
     [,1] [,2] [,3] [,4]
[1,]   -1    0    1    2
[2,]   -2    0    0    2
[3,]   -3    0    0    2
```

Subsetting with Logicals

Review for vectors

Remember, if `x` is a vector and `L` a logical vector of the same length, `x[L]` is a (probably shorter) vector comprised of the entries where `L` is `TRUE`. If `r` is a number, `x[L] <- r` replaces the entry in `x` by `r` when `L` is `TRUE` and leaves the entry alone otherwise.

Subsetting with Logicals

Tricky

For a matrix `X` and logical `L` of matching dimensions, `X[L]` is the numeric vector of entries where `L` is `TRUE`.

```
> X > 0
```

```
      [,1]  [,2]  [,3] [,4]
[1,] FALSE FALSE  TRUE TRUE
[2,] FALSE FALSE FALSE TRUE
[3,] FALSE FALSE FALSE TRUE
```

```
> X[X > 0]
```

```
[1] 1 2 2 2
```

Assignment with Logicals

Sometimes useful

Assignments of the form `X[L] <- r` work as they do for vectors.

```
> X[1, 1] <- NA
> is.na(X)
```

```
      [,1]  [,2]  [,3]  [,4]
[1,]  TRUE FALSE FALSE FALSE
[2,] FALSE FALSE FALSE FALSE
[3,] FALSE FALSE FALSE FALSE
```

```
> X[is.na(X)] <- 0
> X
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    2
[2,]   -2    0    0    2
[3,]   -3    0    0    2
```

Matrix and Number Arithmetic

For `X` a matrix and `r` a number `X+r` and `X*r` are the results of adding, resp. multiplying `r` entrywise to `X`. What if `r` is a vector? Experiment and find out.

If `X` is $n \times m$ and `Y` is $m \times p$, `X %*% Y` is the matrix product.

`t(X)` is the transpose of `X`; i.e., the matrix obtained from `X` by switching rows and columns. `diag(X)` is the vector of elements on the main diagonal.

Row and Column Stats

Frequently we'll want to extract statistics from the rows or columns of a matrix. Let `f` be a function that produces a number given a vector. If `X` is a matrix `apply(X, 1, f)` is the result of applying `f` to each row of `X`; `apply(X, 2, f)` to the columns. The former outputs a vector with one result for each row.

Row and Column Stats

```
> Z <- matrix(rnorm(20), nrow = 4, ncol = 5)
> V <- apply(Z, 1, mean)
> V
[1]  0.1603  0.3566  0.1926 -1.1196
> W <- apply(Z, 2, min)
> W
[1]  0.3452 -0.5668 -1.5553 -1.8376 -2.8867
```

Names

Just as you can name indices in a vector you can (and should!) name columns and rows in a matrix with `colnames(X)` and `rownames(X)`. These can be used in subsetting just like vectors.

Getting the most varying genes

Given a matrix of expression data find the 4000 probes (genes) that are most widely varying across the samples. Restrict the expression matrix to these probes.

```
> load("/Users/steve/Documents/Bio/breast/analysisRecord/Da
> class(expUPPS1pos)

[1] "matrix"

> dim(expUPPS1pos)

[1] 22283   101

> expUPPS1pos[1:3, 1:4]

           GSM110625 GSM110629 GSM110631 GSM110635
1007_s_at     10.310    10.197    10.130    10.169
1053_at        5.071     6.227     5.498     5.684
117_at         4.151     4.171     4.105     3.927
```

Getting the most varying genes

For each probe (row in the matrix) we need a measure of how much it is varying. We use inter-quartile range as the measure of change. For a vector `x`, `IQR(x)` is the third quartile minus the first quartile.

```
> iqrExp <- apply(expUPPS1pos, 1, IQR)
> names(iqrExp) <- rownames(expUPPS1pos)
```

Getting the most varying genes

Sorting the measures

`iqrExp` contains the measures of change for each probe. It is a numeric vector. We sort it to find the largest 4000.

```
> siqrExp <- sort(iqrExp, decreasing = TRUE)
> top4k <- names(siqrExp)[1:4000]
> expUPPS1top <- expUPPS1pos[top4k, ]
> dim(expUPPS1top)
```

```
[1] 4000  101
```

Outline

Fundamental Object for Experimental Data

A `data.frame` object in *R* has similar dimensional properties to a matrix but it may contain categorical data, as well as numeric. The standard is to put data for one sample across a row and covariates as columns.

On one level, as the notation will reflect, a data frame is a list. Each component corresponds to a variable; i.e., the vector of values of a given variable for each sample. A data frame is like a list with components as columns of a table.

Data Frame Restrictions

When can a list be made into a data.frame?

- Components must be vectors (numeric, character, logical) or factors.
- All vectors and factors must have the same lengths.

Matrices and even other data frames can be combined with vectors to form a data frame if the dimensions match up.

Creating Data Frames

Explicitly like a list

```
> measrs <- data.frame(gender = c("M", "M",
+     "F"), ht = c(172, 186.5, 165), wt = c(91,
+     99, 74))
> measrs
```

```
  gender    ht wt
1      M 172.0 91
2      M 186.5 99
3      F 165.0 74
```

Entries in a data.frame are indexed like a matrix:

```
> measrs[1, 2]
```

```
[1] 172
```

Data Frame Attributes

Both List and Matrix

```
> names(measrs)
[1] "gender" "ht"     "wt"
> rownames(measrs) <- c("S1", "S2", "S3")
> measrs$ht
[1] 172.0 186.5 165.0
```

Compenents as Vectors

The components of a data frame can be extracted as a vector as in a list:

```
> height <- measrs$ht
> height
```

```
[1] 172.0 186.5 165.0
```

```
> names(height) <- rownames(measrs)
```

Warning: Character vectors in a data frame are always stored as a factor. It's assumed that's what you should do.

```
> class(measrs$gend)
```

```
[1] "factor"
```

Extracting ALL Compenents

All components in a data frame can be extracted as vectors with the corresponding name:

```
> attach(measrs)
> wt
[1] 91 99 74
> detach(measrs)
```

Expanding Data Frames

Components can be added to a data frame in the natural way.

```
> measrs$age <- c(28, 55, 43)
> measrs
   gender    ht wt age
S1      M 172.0 91  28
S2      M 186.5 99  55
S3      F 165.0 74  43
```

Expanding Data Frames

Row Bind, Column Bind

If you expand the experiment to add data, use row binding to expand.

```
> m2 <- data.frame(gender = c("M", "F"),
+      ht = c(170, 166), wt = c(68, 72),
+      age = c(38, 22))
> rownames(m2) <- c("S4", "S5")
> measrs2 <- rbind(measrs, m2)
```

If other data are kept on the same samples in another data frame it can be combined with the original using `cbind`.

Outline

Reading Tables

The Working Directory

R can read in files on your machine and create data files and graphics. Paths to these files are computed relative to the working directory. Paths are specified in the format appropriate for the machine.

Typically, separate analyses are stored in different directories.

Can Move Between R and Excel

Reading .CSV

Excel allows you to save files as "comma separated values". All formatting is lost but the information content is there. There is a function in *R* that reads the .CSV file and produces a table of data.

```
> setwd("lect2WorkDir")
> clinUpps1 <- read.csv(file = "Clinical_Upps.csv")
> setwd("../")
> class(clinUpps1)
```

```
[1] "data.frame"
```

Can Move Between R and Excel

Writing .CSV

```
> write.csv(measrs2, file = "measrs2.csv")
```

These are special cases of the more general functions `read.table, write.table`. There are numerous options such as "Is the first line header information?" See the help entries.

Outline

Select Rows Based Variable Values

Commonly, we'll want to select those rows in a data.frame in which one of the variables has specific values. The entries in `measrs2` with height ≥ 170 are found as follows.

```
> talls <- measrs2[measrs2$ht >= 170, ]
> talls
```

```
   gender    ht wt age
S1      M 172.0 91  28
S2      M 186.5 99  55
S4      M 170.0 68  38
```

Select Rows Based Variable Values

Combining variables

Attaching the components of the data frame as variables allows for simpler formulas.

```
> attach(measrs2)
> tallNthin <- measrs2[ht >= 170 & wt <=
+     75, ]
> tallNthin
```

```
   gender  ht wt age
S4      M 170 68  38
```

Sort a Data Frame by Selected Column

Often data are better viewed when sorted. The function `order` sorts a column and gives output that can sort the rows of a data.frame. The following sorts `measrs2` by age.

```
> measrsByAge <- measrs2[order(measrs2[,
+     "age"]), ]
> measrsByAge
```

```
   gender    ht wt age
S5      F 166.0 72  22
S1      M 172.0 91  28
S4      M 170.0 68  38
S3      F 165.0 74  43
S2      M 186.5 99  55
```

Sort a Data Frame by Selected Column

Reverse order

`rev` reverses the order from increasing to decreasing.

```
> measrsByHt <- measrs2[rev(order(measrs2[,
+     "ht"])), ]
> measrsByHt
```

```
   gender    ht wt age
S2      M 186.5 99  55
S1      M 172.0 91  28
S4      M 170.0 68  38
S5      F 166.0 72  22
S3      F 165.0 74  43
```