R Language Fundamentals

Data Frames

Steven Buechler

Department of Mathematics
276B Hurley Hall; 1-6233

Fall, 2007
Outline

Objects that Hold Data

- More complicated data frame
Another Data Set

with different issues

> alleles1 <- read.csv("../sampleSets/Allele_sizes_kmb_1.csv")
> dim(alleles1)
[1] 468 15
> alleles1[1:3, 1:5]

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>X1F02_</th>
<th>X2N03_</th>
<th>X1C06_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3004</td>
<td>174_183</td>
<td>320_330</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3009</td>
<td>174_183</td>
<td>298_318</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3010</td>
<td>176_185</td>
<td>320_330</td>
</tr>
</tbody>
</table>
Problem

In the alleles1 data frame, replace the Z_Z entries by NA’s. Exclude from the study any rows or columns that have a number NA’s exceeding a user-defined threshold.
### Set Rownames

Use `Name` and a character prefix to set the rownames

```r
> rownames(alleles1) <- paste("S", alleles1$Name, +   sep = "")
> alleles1[1:3, 1:5]
```

<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>X1F02_</th>
<th>X2N03_</th>
<th>X1C06_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3004</td>
<td>1</td>
<td>3004</td>
<td>174_183</td>
<td>320_330</td>
</tr>
<tr>
<td>S3009</td>
<td>2</td>
<td>3009</td>
<td>174_183</td>
<td>298_318</td>
</tr>
<tr>
<td>S3010</td>
<td>3</td>
<td>3010</td>
<td>176_185</td>
<td>320_330</td>
</tr>
</tbody>
</table>
Objects that Hold Data

Inspect Components

> names(alleles1)

[1] "Index" "Name" "X1F02_" "X2N03_"
[5] "X1C06_B" "X1G13_" "X1M18_" "X1F07_"
[9] "X1C08_" "X0C11_" "X1J11_" "X1D09_"
[13] "X1H14_" "X0C03_" "0I01_"

> datNames <- names(alleles1)[-(1:2)]
How do we find Z’s

The function grep searches for entries in a character vector that match a search pattern.

```
hits <- grep( pattern, x )
```

Here, `x` is a character vector. `pattern` is a character vector representing a regular expression. `hits` is the integer vector of indices `i` such that `x[i]` matches the pattern.
A regular expression is a string of characters that describes a string pattern. In $R$ the common use is finding vector entries that match a particular word or “sub-word”. The web site

http://www.regular-expressions.info/reference.html

is a good quick reference. Very little of the power is needed for our purposes.
Regular Expressions

Simple Cases

To find the entries in a vector \( v \) containing a character string \( c \), simply execute

\[
\text{hits} \leftarrow \text{grep}(c, x)
\]

This finds any occurrence of \( c \), regardless of where it is in the entry.
Regular Expressions
Simple Cases

> b1

[1] "The"  "total"  "test"  "string"  "used"
[6] "to"    "to"    "create" "a"    "simple"
[11] "case"  "of"    "grep"

> grep("a", b1)

[1] 2 8 9 11
Regular Expressions

Exact Words

To find an entry that exactly matches a string you create a regular expression with “anchor points” indicating the start and end of a string.

```r
> b1[grep("^to$", b1)]
[1] "to" "to"
```
In a regular expression you can say “match any of the following”, “match white space”, “match anything except”. See the web page. Just know that when including the escape character \ in reg. exp. in R it should be doubled: \\.

\texttt{hits <- grep( "\\d", x )}

matches any digit in an entry in \texttt{x}.
Each data column of the alleles1 data frame is a factor (that can be coerced to a character vector). Search for Z’s in, say column 10, as follows.

```r
> zin10ind <- grep("Z", alleles1[, 10])
> length(zin10ind)

[1] 69

> zin10ind[1:5]

[1] 180 181 182 183 186
```
Replace Z by NAs
Test Case

> col10 <- alleles1[, 10]
> col10[182:187]

11 Levels: 203_213 203_217 203_217? ... Z_Z

> col10[grep("Z", col10)] <- NA
> col10[182:187]

[1] <NA> <NA> 203_213 203_213 <NA>
11 Levels: 203_213 203_217 203_217? ... Z_Z
Objects that Hold Data

Replace Z by NAs
Loop version

```r
> alleles2 <- alleles1
> for (i in 1:length(datNames)) {
+    col <- alleles2[, i]
+    col[grep("Z", col)] <- NA
+    alleles2[, i] <- factor(as.character(col))
+ }
```
Replace Z by NAs

Check the Work

```r
> grep("Z", alleles2[, 11])
integer(0)

> sum(is.na(alleles2[, 11]))
[1] 39
```
First write a function that replaces Z entries by NA in a column and returns a new column.

```r
> fn <- function(x) {
+   col <- alleles1[, x]
+   col[grep("Z", col)] <- NA
+   factor(as.character(col))
+ }
```
Replace Z by NAs
laply version

> newDat <- lapply(datNames, fn)
> names(newDat) <- datNames
> alleles3 <- alleles1
> alleles3[, datNames] <- newDat
Objects that Hold Data

Check New Data Frame

```r
> sum(is.na(alleles3[, 11]))
[1] 39

> grep("Z", alleles3[, 11])
integer(0)
```
The function `na.omit` removes from a data.frame any row (sample) with an NA. Applying this here is inappropriate. It throws away too much.

```r
> all4 <- na.omit(alleles3)
> dim(all4)
[1] 229 15
```
How to Count NAs?

in both rows and columns

We know how to count the number of NAs in a given character vector. We use `lapply` to extend this across all rows and columns. First consider the columns.

```r
> colNAs1 <- lapply(datNames, function(x) {
+   sum(is.na(alleles3[, x]))
+ })
> names(colNAs1) <- datNames
> colNAs <- unlist(colNAs1)
```
Count NAs in Rows

> sampIDs <- rownames(alleles3)
> rowNAs1 <- lapply(sampIDs, function(x) {
+   sum(is.na(alleles3[x, ]))
+ })
> names(rowNAs1) <- sampIDs
> rowNAs <- unlist(rowNAs1)
Objects that Hold Data

Examine the Results

> colNAs

X1F02_  X2N03_  X1C06_B  X1G13_  X1M18_  X1F07_
  31   29   68   34   69   65
X1C08_  X0C11_  X1J11_  X1D09_  X1H14_  X0C03_
  44   69   39   36   45  186
0I01_
  38

> table(rowNAs)

rowNAs
   0  1  2  3  4  5  6  7  8  9 10 11
   229 111 48 15  8 11  6 10  6  2  6  9
     12
      7

Keep only the desired rows and columns based on the biological questions being asked.