Arrays and Useful R functions

Introduction to R for Public Health Researchers

Data Frames versus Matrices

You will likely use data.frame class for a lot of data cleaning and analysis. However, some operations that rely on matrix multiplication (like performing many linear regressions) are (much) faster with matrices. Also, as we will touch on later, some functions for iterating over data will return the matrix class, or will be placed in empty matrices that can then be converted to data.frames

Data Frames versus Matrices

There is also additional summarization functions for matrices (and not data.frames) in the `matrixStats` package, like `rowMins()`, `colMaxs()`, etc.

```
library(matrixStats,quietly = TRUE)
avgs = select(circ, ends_with("Average"))
rowMins(as.matrix(avgs),na.rm=TRUE)[500:510]
```

```
##  [1] 3538.5 3402.5 3862.5 3347.5 2837.5 2704.0 3138.5 32
## [11] 3046.0
```

Data Classes

Extensions of “normal” data classes:

- N-dimensional classes:
- Arrays: any extension of matrices with more than 2 dimensions, e.g. 3x3x3 cube
- Lists: more flexible container for R objects.

Arrays

These are just more flexible matrices - you should just be made aware of them as some functions return objects of this class, for example, cross tabulating over more than 2 variables and the `tapply` function.

Arrays

Selecting from arrays is similar to matrices, just with additional commas for the additional slots.

```
ar = array(1:27, c(3,3,3))
ar[,,1]
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
ar[,1,]
```

```
##      [,1] [,2] [,3]
## [1,]    1   10   19
## [2,]    2   11   20
## [3,]    3   12   21
```

Splitting Data Frames

The `split()` function is useful for splitting `data.frames`

"`split` divides the data in the vector `x` into the groups defined by `f`. The replacement forms replace values corresponding to such a division. `unsplit` reverses the effect of split."

```
> dayList = split(circ,circ$day)
```

Splitting Data Frames

Here is a good chance to introduce `lapply`, which performs a function within each list element:

```
r   > # head(dayList)   > lapply(dayList, head, n=2)
```

“‘ $Friday day date orangeBoardings orangeAlightings orangeAverage 5 Friday 01/15/2010 1645 1643 1644.0 12 Friday 01/22/2010 1401 1388 1394.5 purpleBoardings purpleAlightings purpleAverage greenBoardings 5 NA NA NA NA 12 NA NA NA NA greenAlightings greenAverage bannerBoardings bannerAlightings 5 NA NA NA NA 12 NA NA NA NA bannerAverage daily 5 NA 1644.0 12 NA 1394.5

$Monday day date orangeBoardings orangeAlightings orangeAverage 1 Monday 01/11/2010 877 1027 952.0 8 Monday 01/18/2010 999 1000 999.5 purpleBoardings purpleAlightings purpleAverage greenBoardings 1 NA NA NA NA 8 NA NA NA NA greenAlightings greenAverage bannerBoardings bannerAlightings 1 NA NA NA NA 8 NA NA NA NA bannerAverage daily 1 NA 952.0 8 NA 999.5

$Saturday day date orangeBoardings orangeAlightings orangeAverage 6 Saturday 01/16/2010 1457 1524 1490.5 13

```
r   > # head(dayList)   > lapply(dayList, dim)
```

“‘ $Friday [1] 164 15

$Monday [1] 164 15

$Saturday [1] 163 15

$Sunday [1] 163 15

$Thursday [1] 164 15

$Tuesday [1] 164 15

$Wednesday [1] 164 15 “‘