The Data Frame, Type Conversion, Graphing

A. Michelle Lawing
Ecosystem Science and Management
Texas A&M University
College Station, TX 77843
alawing@tamu.edu
Lists

• Used to combine different types of information into a single object

```r
w <- list("yes", 32, TRUE)
w[2]  # just like a vector
```

• List elements can have names

```r
me <- list(name="Michelle", id=40172)
me$name  # "Michelle"
me[1]    # "Michelle"
```
Dataframes

- Rectangular table of information, not necessarily of the same type (numbers, text, etc).

- Usually, this is the form your data will be in when you import it

<table>
<thead>
<tr>
<th></th>
<th>taxon1</th>
<th>taxon2</th>
<th>taxon3</th>
<th>taxon4</th>
</tr>
</thead>
<tbody>
<tr>
<td>site1</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>site2</td>
<td>34</td>
<td>3</td>
<td>44</td>
<td>1</td>
</tr>
<tr>
<td>site3</td>
<td>23</td>
<td>1</td>
<td>112</td>
<td>4</td>
</tr>
<tr>
<td>site4</td>
<td>0</td>
<td>5</td>
<td>67</td>
<td>1</td>
</tr>
</tbody>
</table>
Dataframes

<table>
<thead>
<tr>
<th></th>
<th>taxon1</th>
<th>taxon2</th>
<th>taxon3</th>
<th>taxon4</th>
</tr>
</thead>
<tbody>
<tr>
<td>site1</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>site2</td>
<td>34</td>
<td>3</td>
<td>44</td>
<td>1</td>
</tr>
<tr>
<td>site3</td>
<td>23</td>
<td>1</td>
<td>112</td>
<td>4</td>
</tr>
<tr>
<td>site4</td>
<td>0</td>
<td>5</td>
<td>67</td>
<td>1</td>
</tr>
</tbody>
</table>

- Rows and columns can be accessed like a matrix
  - `abund[1, 3]` 10
  - `abund[2, ]` # all of 2nd row

- Columns can be accessed by name (like a list)
  - `abund$taxon2` # all of 2nd column
Dataframes

\[
\text{abund} = \begin{array}{cccc}
\text{taxon1} & \text{taxon2} & \text{taxon3} & \text{taxon4} \\
\text{site1} & 0 & 20 & 10 & 2 \\
\text{site2} & 34 & 3 & 44 & 1 \\
\text{site3} & 23 & 1 & 112 & 4 \\
\text{site4} & 0 & 5 & 67 & 1 \\
\end{array}
\]

- Use `attach()` to access variable names directly
  ```r
  attach(abund)
  taxon2 # all of 2nd column
  detach(abund) # undoes the attach()
  ```
Datasets in R

• Some datasets are built-in to R for purposes of illustration.

• They can be accessed using the data() function.
Exercise on Dataframes

1. Make the dataset iris available by typing data(iris). Take a look at the data by typing iris. Note that row names are index numbers and the columns are attributes of the flower (a description of what the variables mean can be found by typing ?iris).

2. How many rows and columns are there in this dataset?

3. Use the functions mean() and median() to calculate the mean and median of sepal length and petal width.

4. Calculate the sepal length in millimeters, and save that to a new variable called sl.ml
### Testing Relationships

<table>
<thead>
<tr>
<th>Condition</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than, less than</td>
<td><code>&gt;</code>, <code>&lt;</code></td>
</tr>
<tr>
<td>Greater or equal to, less than or equal to</td>
<td><code>&gt;=</code>, <code>&lt;=</code></td>
</tr>
<tr>
<td>Equal, not equal</td>
<td><code>==</code>, <code>!=</code></td>
</tr>
<tr>
<td>AND, OR</td>
<td><code>&amp;</code>, `</td>
</tr>
</tbody>
</table>

$x \leftarrow 4$

$x > 10$  \hspace{1cm} **FALSE**

$x \leftarrow c(4, 8, 30, 52)$

$x > 10$  \hspace{1cm} **FALSE**  **FALSE**  **TRUE**  **TRUE**

$x > 10 \& x < 50$  \hspace{1cm} **FALSE**  **FALSE**  **TRUE**  **TRUE**  **FALSE**
Subsetting vectors

```
x <- c(4,8,30,52)
```

1. Choose by their indices
```
x[c(3,4)] 30 52
```

2. Using logical (T/F) vectors
```
x[c(FALSE,FALSE,TRUE,TRUE)] 30 52
x > 10  FALSE FALSE TRUE TRUE
x[x > 10] 30 52
```
Exercise: Subsetting

1. Continuing with the iris dataset, compute separately the mean sepal length of each species (e.g., *Iris setosa* (==“setosa”).

2. Extract out a vector of sepal width values that are from *I. setosa* and *I. versicolor*. 
Graphing

- Powerful graphing capabilities
- Can be saved as vector graphics (PDF, postscript)
- Can add to a plot, but can’t edit what’s already there (not clickable)

0th rule of data analysis
Plot your data! (J. Sepkoski)
Making Graphs

Generate some fake data

\[ x \leftarrow \text{rnorm}\left(n=20,\text{mean}=0,\text{sd}=1\right) \]  # random normal numbers
\[ y \leftarrow \text{rnorm}\left(n=20,\text{mean}=100,\text{sd}=10\right) \]

Try Some Plots…

```r
plot(x)
plot(x, type = "l")
plot(x, type = "b")
plot(x, y)
plot(x, y, pch = 2)
plot(x, y, pch = 3, col = "red")
```
Other common graphing functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>hist()</td>
</tr>
<tr>
<td>barplot()</td>
</tr>
<tr>
<td>boxplot()</td>
</tr>
<tr>
<td>pie()</td>
</tr>
<tr>
<td>contour()</td>
</tr>
</tbody>
</table>

Common arguments to graphing functions

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td>Color</td>
<td>col=&quot;red&quot;, col=2</td>
</tr>
<tr>
<td>cex</td>
<td>Character size</td>
<td>cex=2 # twice as big</td>
</tr>
<tr>
<td>pch</td>
<td>Plotting symbol</td>
<td>pch=5 # diamonds</td>
</tr>
<tr>
<td>lty</td>
<td>Line type</td>
<td>lty=2 # dashed</td>
</tr>
<tr>
<td>log</td>
<td>Log-scale axes</td>
<td>log=&quot;x&quot;, log=&quot;xy&quot;</td>
</tr>
</tbody>
</table>

Things you can add to existing plots

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>title()</td>
</tr>
<tr>
<td>abline()</td>
</tr>
<tr>
<td>points()</td>
</tr>
<tr>
<td>text()</td>
</tr>
<tr>
<td>polygon()</td>
</tr>
<tr>
<td>legend()</td>
</tr>
<tr>
<td>arrows()</td>
</tr>
<tr>
<td>segments()</td>
</tr>
<tr>
<td>rect()</td>
</tr>
<tr>
<td>symbols()</td>
</tr>
</tbody>
</table>
Interacting with plots

<table>
<thead>
<tr>
<th>locator()</th>
<th>Gives x,y coordinates of clicked location</th>
</tr>
</thead>
<tbody>
<tr>
<td>identify()</td>
<td>Identifies clicked data point</td>
</tr>
<tr>
<td></td>
<td>identify(x,y) # returns index of clicked points</td>
</tr>
<tr>
<td></td>
<td># when done, right-click (Win) or</td>
</tr>
<tr>
<td></td>
<td># ESC (Mac)</td>
</tr>
</tbody>
</table>

Varying parameters for different points

Many graphing arguments can be a vector of the same length as the data.

```r
plot(c(1,2,10,4), pch=c(1,1,2,2))
```

This feature can be used to designate different plotting symbols, colors, etc. for different groups.
Median Value

Description
Compute the sample median.

Usage
median(x, na.rm = FALSE)

Arguments
x an object for which a method has been defined, or a numeric vector containing the values whose median is to be computed.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

Details
This is a generic function for which methods can be written. However, the default method makes use of is.na, sort and mean from package base all of which are generic, and so the default method will work for most classes (e.g. "Date") for which a median is a reasonable concept.

Value
The default method returns a length-one object of the same type as x, except when x is integer of even length, when the result will be double.
If there are no values or if na.rm = FALSE and there are NA values the result is NA of the same type as x (or more generally the result of x[na.rm][na]).

References

See Also
quantile for general quantiles.

Examples
median(1:5)  # 2.5 [even number]
median(c(1:3, 100, 1000))  # 3 [odd, robust]