Eric Pitman Summer Workshop in Computational Science

2. Data Structures: Vectors and Data Frames
Data Objects in R

These objects, composed of multiple atomic data elements, are the bread and butter of R:

• (Atomic) **Vectors**
• **Data Frames**

Also: lists; matrices; arrays.
Vector Data Object

An (atomic) vector is a collection of elements having the same type.

<table>
<thead>
<tr>
<th>Element Index</th>
<th>Element Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Vector Name

Element Index

Element Value

Lists differ from vectors in that a list’s contents are not constrained to be the same type!
Construct a Vector Data Object

Use the `c()` function:

```r
> a <- c(5,6,7,8)  # vector with 4 numeric values
> d <- c("red", "orange", "green")  # character vector
```
Accessing Vector Data

Access by index or range:

> d[1]  # retrieves “red”
> a[3]  # retrieves 7
> d[1:2]  # retrieves “red”, “orange”

Element numbering starts at 1 in R
Information about a Vector

> y <- c(3,5,7,9)  # vector with 4 numeric values

> length(y)        # how many elements?
> class(y)         # class of a vector object is the class of its elements
Information about a Vector

\[
\begin{array}{|c|}
\hline
\text{y} \\
1 & 3 \\
2 & 5 \\
3 & 7 \\
4 & 9 \\
\hline
\end{array}
\]

\[
> \text{str(y)} \quad \# \text{ structure of the vector: number of elements, type, and contents}
\]

\[
\text{num [1:4] 3 5 7 9}
\]

Type of elements

Number (and positions) of elements

contents
Everything’s a vector!

> 2 + 3 * 5

[1] 17

Q: What's that [1] about?
A: It’s the index of a vector of length 1.

# Try this in the command line:
> 1:500
Matrix: a vector with dimensions

> y <- c(3,5,7,9)  # vectors with 4 numeric values
> z <- 1:4
> m <- cbind(y,z)  # create a matrix

> dim(m)  # how many elements? 4 rows, 2 cols
> class(m)  # class is matrix
> typeof(m)  # numeric
Some Operations on Vectors

- `sum()`  # Sum of all element values
- `length()`  # Number of elements
- `unique()`  # Generate vector of distinct values
- `diff()`  # Generate vector of first differences
- `sort()`  # Sort elements, omitting NAs
- `order()`  # Sort indices, with NAs last
- `rev()`  # Reverse the element order
- `summary()`  # Information about object contents
Repercussions of NA

Any arithmetic operation on a structure containing an NA generates NA!

```r
# NA means “no value known”
> y = c(1, NA, 3, 2, NA)
> sum(y)
[1] NA
```

We must *remove NAs* to make calculations. How?
Finding NAs in a Data Structure

```r
> y = c(1, NA, 3, 2, NA)
```

```r
> summary(y)
```

```
  Min.  1st Qu.   Median     Mean  3rd Qu.    Max.  NA's
  1.0    1.5     2.0      2.0     2.5     3.0     2
```
Handling Missing Data

Remove NAs prior to calculation:

```r
> y = c(1, NA, 3, 2, NA)  # [1, ?, 3, 2, ?]
sum(y, na.rm=TRUE)       # removes NAs, then sums
[1] 6
```

rm = “remove”
Data Frames

- A data frame is a structure consisting of columns of various modes (numeric, character, etc).
- Its rows and columns can be named.
- Data frames are handy containers for experimental data.

Data frames are actually lists!
Data frames are handy containers for data that describe experimental subjects.

Student population data:

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>120</td>
<td>16</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>160</td>
<td>17</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>118</td>
<td>16</td>
</tr>
</tbody>
</table>
Constructing a Data Frame

1. Construct the vectors that hold column data:
   
   height = c(68, 75, 60)  # inches
   age = c(16, 17, 16)  # years
   handed = c("L", "R", "R")  # dominant hand: R=right, L=left

2. Construct the data frame by associating the columns:
   
   data = data.frame(Height=height, Age=age, Hand=handed)

Name of the column!
## Data Frame

Organized in rows and columns:

<table>
<thead>
<tr>
<th>Rows</th>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>120</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>160</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>118</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

Columns (formed from vectors)
### Accessing by Index

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 68</td>
<td>120</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B 75</td>
<td>160</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C 60</td>
<td>118</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

First index is row, second index is column:

> data[2,3] # retrieves subject B's Age
## Accessing by Index

<table>
<thead>
<tr>
<th></th>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>120</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>160</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>118</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

Return an entire row:

> data[2, ]  # retrieves subject B's data

Comma is a placeholder in the [row, column] notation
### Accessing by Index

<table>
<thead>
<tr>
<th></th>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>120</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>160</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>118</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

Return an entire column:

> data[,4]  # retrieves all Handedness data
### Accessing by Name

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

# To fetch Height column:

> `data[ , "Height"]`

# Or:

> `data$Height`
Conditional Access: which()

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>120</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>160</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>118</td>
<td>R</td>
</tr>
</tbody>
</table>

> which(data$Height > 65)

1 2

# which() returns the *indices* for which the conditional is true!
Conditional Access: `subset()`

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

```
> subset(data, Height>65, select="Height")
```

# `subset()` arguments are:
# 1. dataset to subset,
# 2. subsetting condition to apply,
# 3. columns to return
# Conditional Access

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

**Heights** over 65 inches:

```r
> data[which(data$Height > 65), "Height"]
> subset(data, Height>65, select="Height")
```

# safe if you have NAs in the dataframe.
Can’t we just...

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight</th>
<th>Age</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>68</td>
<td>16</td>
<td>L</td>
</tr>
<tr>
<td>B</td>
<td>75</td>
<td>17</td>
<td>R</td>
</tr>
<tr>
<td>C</td>
<td>60</td>
<td>16</td>
<td>R</td>
</tr>
</tbody>
</table>

Heights over 65 inches:

> data$Height[data$Height > 65]  # subset of a column
   # of the data frame

Caution: Not NA safe!
Try it:
Accessing by Index

> source("data-frame-simple-example.R")
> data[2,3]  # retrieves subject B's Age
> data[2,]   # retrieves all subject B data
> data[ ,3]  # retrieves all Age data
Try it:
Accessing by Name

> source("data-frame-simple-example.R")
> data["B","Age"] # retrieves B's Age
> data["B", ] # retrieves all B data
> data$Age # retrieves all Age data
Try it: Conditional Access

```r
> source("data-frame-simple-example.R")
# subset of the data frame having age<17 years:
> data[which(data$Age < 17), ]

# subset of a column of data frame, age<17 years:
> data$Age[which(data$Age < 17)]
```
Data Frame Information

\begin{itemize}
  \item \texttt{str(data)} \# structure
  \item \texttt{dim(data)} \# dimensions
  \item \texttt{is.data.frame(data)} \# returns a logical value
  \item \texttt{View(data)} \# open View window of data
  \item \texttt{head(data)} \# beginning of the data frame
  \item \texttt{tail(data)} \# end of the data frame
  \item \texttt{names(data)} \# names of the columns
  \item \texttt{rownames(data)} \# names of the rows
  \item \texttt{colnames(data)} \# names of the columns
\end{itemize}

\begin{verbatim}
> class(data)
[1] "data.frame"
\end{verbatim}
Let's create our own dataset and put it in an R data frame:

- FirstInitial
- LastInitial
- School
- Height
- HtUnit
- Age
- Handed
- Gender
Now we can write some R to select subsets of our data. Examples:

- How many students younger than 17?
- List heights of students at Williamsville North
- Genders of left-handers?
Interlude

Complete vector/data frame exercises.

Open in the RStudio source editor:

<workshop>/exercises/2-exercises-vectors-matrices-dataframes.R
Interlude++

Once you have completed the exercises, read about R:

An R tutorial (Check out slides 23-24, 33, 50, 61-63, 75 for relevant material):

- http://jaredknowles.com/s/Tutorial1_Intro.html

Data Wrangling with R: