Eric Pitman Summer Workshop in Computational Science

2. Data Structures: Vectors and Data Frames

Data Objects in R

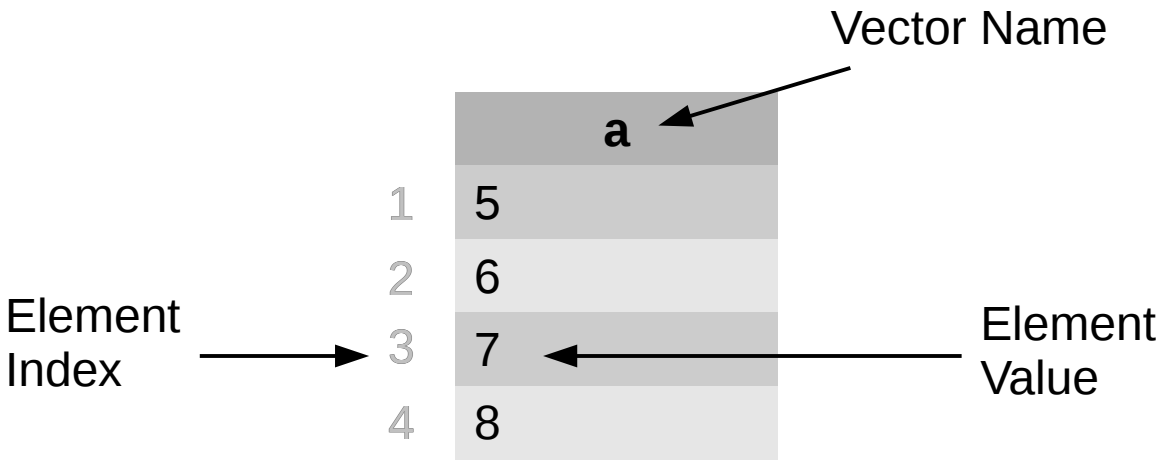
These objects, composed of multiple atomic data elements, are the bread and butter of R:

- (Atomic) **Vectors**
- **Data Frames**

Also: lists; matrices; arrays.

Vector Data Object

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



Lists differ from *vectors* in that a list's contents are not constrained to be the same type!

Construct a Vector Data Object

	a
1	5
2	6
3	7
4	8

Use the c() function:

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

Accessing Vector Data

	d
1	“red”
2	“orange”
3	“green”

	a
1	5
2	6
3	7
4	8

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
1	3
2	5
3	7
4	9

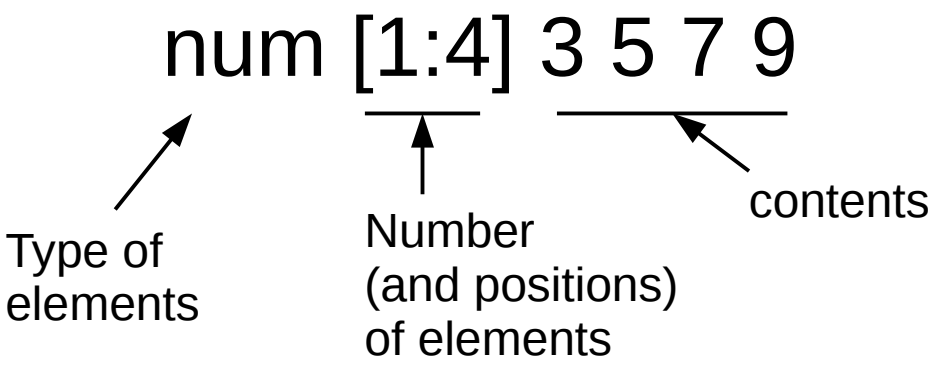
```
> y <- c(3,5,7,9)  # vector with 4 numeric values

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

Information about a Vector

	y
1	3
2	5
3	7
4	9

```
> str(y)      #  structure of the vector: number of
              #  elements, type, and 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	z
1	3	1
2	5	2
3	7	3
4	9	4

```
> 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!

```
# 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

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

```
> 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:

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

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 Frame Example

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

Student population data:

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

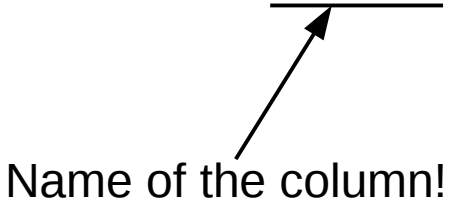
Constructing a Data Frame

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)
```



Data Frame

Organized in rows and columns:

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Rows

Columns (formed from vectors)

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

First index is row, second index is column:

```
> data[2,3]  # retrieves subject B's Age
```

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Return an entire row:

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

Comma is a placeholder in the [row, column] notation

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Return an entire column:

```
> data[ ,4]  # retrieves all Handedness data
```

Comma is a placeholder in the [row, column] notation

Accessing by Name

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

To fetch Height column:

> data[,“Height”]

Or:

> data$Height

Conditional Access: which()

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

```
> which(data$Height > 65)
1 2

# which() returns the indices for which the conditional is true!
```

Conditional Access: subset()

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

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

```
# subset() arguments are:
#                   dataset to subset,
#                   subsetting condition to apply,
#                   columns to return
```

Conditional Access

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Heights over 65 inches:

```
> data[which(data$Height > 65), “Height”]
> subset(data, Height>65, select=“Height”)
# safe if you have NAs in the dataframe.
```

Can’t we just...

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

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

```
> 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

```
str(data)       # structure
dim(data)       # dimensions
is.data.frame(data)     # returns a logical value

View(data)        # open View window of data
head(data)        # beginning of the data frame
tail(data)        # end of the data frame

names(data)         # names of the columns
rownames(data)    # names of the rows
colnames(data)     # names of the columns
```

```
> class(data)
[1] "data.frame"
```

Student Dataset Example

Let's create our own dataset and put it in an R data frame:

- `FirstInitial`
- `LastInitial`
- `School`
- `Height`
- `HtUnit`
- `Age`
- `Handed`
- `Gender`

Student Dataset Example

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

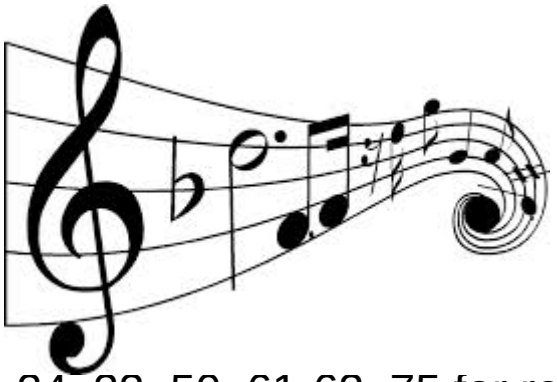
Complete vector/data frame exercises.

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:

http://dzchilds.github.io/aps-data-analysis-L1/data-frames.html