Goals:

- Understand when to use R & when not use it.
- Understand basic syntax & write short programs.
- Understand scalability issues & ways to resolve them.
R Programming Language
Lesson Preview

Four parts of this lesson:

- getting started
- data types
- control flow and functions
- scalability and interfaces
## R, Python, and Matlab Similarities

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>R</th>
<th>Python</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run in interactive shell or graphical UI</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Store and manipulate data as arrays</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Many packages</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Slower than C, C++</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Interface with C++</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
R, Python, and Matlab Differences
## R, Python, and Matlab Differences

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>R</th>
<th>Python</th>
<th>Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open source</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Ease of Contribution</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Quality of Contributions</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Suitable for Statistics</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Better Graphics Capabilities</td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
R, Python, and Matlab Differences

**R**
- Statistics
- Bio-statistics
- Social sciences

**Matlab**
- Engineering
- Applied Math

**Python**
- Web Development
- Scripting
Running R

Interactively

Non-Interactively
Running R - Interactively

RStudio
Running R - Interactively

From the terminal:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td><code>$ open -a RStudio .</code></td>
</tr>
<tr>
<td>Linux</td>
<td><code>$ rstudio</code></td>
</tr>
</tbody>
</table>
Running R - Interactively
Running R - Interactively

Data sets in package 'ggplot2':
- diamonds
- economics
- midwest
- movies
- mpg
- models of
- male
- sleep dataset
- presidential
- seals

- Prices of 50,000 round cut diamonds
- US economic time series
- Demographic information of midwest counties
- Movie information and user ratings from IMDB.com
- Fuel economy data from 1999 and 2008 for 38 popular cars
- An updated and expanded version of the mammals
- Terms of 10 presidents from Eisenhower to Bush Jr.
- Vector field of seal movements
Running R - Interactively
Running R - Interactively
Running R - Interactively
4 Ways to Run R

Type R in prompt (type q() to quit)

- R graphic application
- R-Studio
- Within Emacs
Running R - Non-Interactively

- **call script from R**: `source("foo.R")`
- **call script from shell**: `R CMD BATCH foo.R`
- **call script from shell**: `Rscript foo.R`
- **executable script**, prefixed by `#!/usr/bin/Rscript`, followed by `./foo.R < inFile > outFile`
R Language Quiz

Check all statements that are true:

- ☑ white spaces are dropped
- ☐ semicolons are required at the ends of all commands
- ☐ comments are denoted with '%'
- ☐ statically typed
R Language Quiz

Check all statements that are true:

- [ ] white spaces are dropped
- [x] semicolons are required at the ends of all commands

Example of when semicolons are required:

```r
a = "a string"; b = 2
```
R Language Quiz

Check all statements that are true:

- white spaces are dropped ✔️
- semicolons are required at the ends of all commands
- comments are denoted with `%'

Comments are denoted by:
```r
# This is a comment
```
Typing `help()` in the terminal:

```
> help()
```

`help(topic, package = NULL, lib.loc = NULL, verbose =getOption("verbose"), try.all.packages =getOption("help.try.all.packages"), help_type =getOption("help_type"))`

**Arguments:**

- `topic`: usually, a name or character string specifying the topic for which help is sought. A character string (enclosed in explicit single or double quotes) is always taken as naming a topic.

  If the value of `topic` is a length-one character vector the topic is taken to be the value of the only element. Otherwise `topic` must be a name or a reserved word (if syntactically valid) or character string.

  See ‘Details’ for what happens if this is omitted.

- `package`: a name or character vector giving the packages to look into for documentation, or ‘NULL’. By default, all packages whose namespaces are loaded are used. To avoid a name being deparsed use e.g. `‘(pkg_ref)’` (see the examples).
Typing `help()` in the terminal:
R Help Documentation

Typing `help()` in the terminal:
Typing `help()` in the terminal:
To get help on a specific command, type:

```
help("specific-command")
```

For example:

```
help("load")
```
R Commands

- `ls()` - list variable names in workspace memory
- `save.image(file="R_workspace")` - Saving variables to a file
- `save(new.var, legal.var.name, file = "R_workspace")` - save specified variables
- `load("R_workspace")` - load variables saved in a file
R Commands

Environment Commands:

- `install.packages("ggplot2")` - install the ggplot2 package
- `library(ggplot2)` - load the ggplot2 package

System Commands:

- `system("ls -al")` - executes a command in the shell, for example `ls -al`
## Scalars

### Major Scalar Types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Result of Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>numeric</td>
<td><code>a = 3.2; b = 3</code></td>
<td><code>a: num 3.2</code> <code>b: num 3</code></td>
</tr>
<tr>
<td>integer</td>
<td><code>c = as.integer(b)</code></td>
<td><code>c: int 3</code></td>
</tr>
<tr>
<td>logical</td>
<td><code>d = TRUE</code></td>
<td><code>d: logi TRUE</code></td>
</tr>
<tr>
<td></td>
<td><code>e = as.numeric</code></td>
<td><code>e: num 1</code></td>
</tr>
<tr>
<td>string</td>
<td><code>f = “This is a string”</code></td>
<td><code>f: chr “This is a string”</code></td>
</tr>
</tbody>
</table>
Factors are variables in R which take on a limited number of different values.
Ordered Factor

current.season = factor("summer", levels = c("summer", "fall", "winter", "spring"), ordered = TRUE)

UnOrdered Factor:

my.eye.color = factor("brown", levels = c("brown", "blue", "green"), ordered = FALSE)
Vectors Quiz 1

Fill in the blanks with the outcome of each ‘R’ command.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>concatenate</td>
<td>$x = c(4,3,3,4,3,1)$</td>
<td>$x = 4\ 3\ 3\ 4\ 3\ 1$</td>
</tr>
<tr>
<td>get length of vector or array</td>
<td><code>length(x)</code></td>
<td><code>length = 6</code></td>
</tr>
<tr>
<td>assign a boolean vector</td>
<td><code>y = vector(mode = &quot;logical&quot;, length = 4)</code></td>
<td><code>y = FALSE\ \ FALSE\ \ FALSE\ \ FALSE\ \ FALSE</code></td>
</tr>
<tr>
<td>assign a numeric vector</td>
<td><code>z = vector(length = 3, mode = &quot;numeric&quot;)</code></td>
<td><code>z = 0\ \ 0\ \ 0</code></td>
</tr>
</tbody>
</table>
Fill in the blanks with the outcome of each 'R' command.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat value multiple times</td>
<td><code>q = rep(3.2, times = 10)</code></td>
<td><code>q = 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2 3.2</code></td>
</tr>
<tr>
<td>load values in increments</td>
<td><code>w = seq(0, 1, by = 0.1)</code></td>
<td><code>w = 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0</code></td>
</tr>
<tr>
<td>load values in equally spaced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>increments</td>
<td><code>w = seq(0, 1, length.out = 11)</code></td>
<td><code>w = 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0</code></td>
</tr>
</tbody>
</table>
## Comparison Commands Quiz

**Fill in the boxes** with the result of each example command.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean vector</td>
<td>w &lt;= 0.5</td>
<td>w = TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE</td>
</tr>
<tr>
<td>Checking for true elements</td>
<td>any(w &lt;= 0.5)</td>
<td>TRUE</td>
</tr>
<tr>
<td>Checking for all true elements</td>
<td>all(w &lt;= 0.5)</td>
<td>FALSE</td>
</tr>
<tr>
<td>Which elements are true</td>
<td>which(w &lt;= 0.5)</td>
<td>1 2 3 4 5 6</td>
</tr>
</tbody>
</table>
## Subset Commands Quiz

Fill in the boxes with the result of each example command.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracting entries</td>
<td>$w[w &lt;= 0.5]$</td>
<td>0.0 0.1 0.2 0.3 0.4 0.5</td>
</tr>
<tr>
<td>Subset function</td>
<td>subset(w, w &lt;= 0.5)</td>
<td>0.0 0.1 0.2 0.3 0.4 0.5</td>
</tr>
<tr>
<td>Zero out components</td>
<td>$w[w &lt;= 0.5] = 0$</td>
<td>$w = 0.0$ 0.0 0.0 0.0 0.0 0.0 0.6 0.7 0.8 0.9 1.0</td>
</tr>
</tbody>
</table>
Creating Arrays Quiz

\[ z = \text{seq}(1, 20, \text{length.out} = 20) \]
\[ x = \text{array}(\text{data} = z, \text{dim} = c(4, 5)) \]

Fill in the boxes with the values stored in the array.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>[2,]</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>[3,]</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>[4,]</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>
Given the following array, fill in the blanks with the results of each command.

\[ x[2, 3] = 10 \]
\[ x[2, \_] = 2 \ 6 \ 10 \ 14 \ 18 \]

\[ x[-1, \_] = \]

y = \[x[c(1,2), \ c(1,2)]\]
**Manipulating Arrays Quiz**

Given the following array, determine the outcomes of the following commands.

\[
y = \begin{bmatrix}
[,1] & 1 & 5 \\
[,2] & 2 & 6 \\
\end{bmatrix}
\]

1. \(2 \times y + 1\)

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, ]</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>[2, ]</td>
<td>5</td>
<td>13</td>
</tr>
</tbody>
</table>

2. \(y \times y\)

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, ]</td>
<td>11</td>
<td>35</td>
</tr>
<tr>
<td>[2, ]</td>
<td>14</td>
<td>46</td>
</tr>
</tbody>
</table>
Inner Product and Transpose Quiz

Given the array `x`, determine the outcome of the following commands.

```
x[1,] %*% x[1,]
```

```
t(x)
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[,1]</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>[1,]</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>[,2]</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>[,3]</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>[,4]</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

```

```
```
Outer Product Quiz

Given the array 'x', determine the outcome of the following commands.

\[ \text{outer}(x[,1], x[,1]) \]
Determine the outcome of the following commands.

**rbind(x[1,], x[1,])**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
</tr>
</tbody>
</table>

**cbind(x[1,], x[1,])**

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[2,]</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>[3,]</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>[4,]</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>[5,]</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>
Lists Quiz

Given the following list command, fill in the blanks with the result of each command.

```r
L = list(name = 'John', age = 55, no.children = 2, children.ages = c(15, 18))
```

- `names(L)`
  - `name`, `age`, `no.children`, `children.ages`

- `L['name']`
  - `John`

- `L$children.ages[2]`
  - `18`

- `L[[4]][2]`
  - `18`

- `L[[2]]`
  - `55`

- `L$name`
  - `John`
Dataframes Quiz

Assume the following commands have been executed, fill in the blanks with the corresponding outputs:

```r
vecn = c("John Smith","Jane Doe")
veca = c(42, 45)
vecs = c(50000, 55000)
R = data.frame(name = vecn, age = veca, salary = vecs)
```

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>age</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Smith</td>
<td>42</td>
<td>50000</td>
</tr>
<tr>
<td>2</td>
<td>Jane Doe</td>
<td>45</td>
<td>55000</td>
</tr>
</tbody>
</table>
null
# Datasets Quiz 1

Write the 'R' command that will perform the listed task.

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>List the dimension (column) names</td>
<td><code>names(iris)</code></td>
</tr>
<tr>
<td>Show the first four rows</td>
<td><code>head(iris,4)</code></td>
</tr>
<tr>
<td>Show the first row</td>
<td><code>iris[1]</code></td>
</tr>
<tr>
<td>Sepal length of the first 10 samples</td>
<td><code>iris$Sepal.Length[1:10]</code></td>
</tr>
<tr>
<td>Allow replacing iris$Sepal.Length with shorter Sepal.Length</td>
<td><code>attach(iris, warn.conflicts = FALSE)</code></td>
</tr>
</tbody>
</table>
# Datasets Quiz 2

Write the 'R' command that will perform the listed task:

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average of Sepal.Length across all rows</td>
<td><code>mean(Sepal.Length)</code></td>
</tr>
<tr>
<td>Means of all four numeric columns</td>
<td><code>colMeans(iris[,1:4])</code></td>
</tr>
<tr>
<td>Create a subset of sepal lengths less than 5 in the setosa species</td>
<td><code>subset(iris, Sepal.Length &lt; 5 &amp; Species == &quot;setosa&quot;)</code></td>
</tr>
<tr>
<td>number of rows corresponding to setosa species</td>
<td><code>dim(subset(iris, Species == &quot;setosa&quot;))[1]</code></td>
</tr>
<tr>
<td>summary of the dataset iris</td>
<td><code>summary(iris)</code></td>
</tr>
</tbody>
</table>
a = 10; b = 5; c = 1
if (a < b) {
    d = 1
} else if (a == b) {
    d = 2
} else {
    d = 3
}
print(d)
AND: &&, OR: ||,
equality: ==, inequality: !=
Loops Quiz

Use a 'for; loop to write an 'R' program that adds the numbers (num) 1 to 100 and stores it in a variable called 'sum'

sum=0

# repeat for 100 iteration, with num taking values 1:100

for (num in seq(1, 100, by = 1)) {
  sum = sum + num
}

Repeat Loops Quiz

Using a repeat loop, write an 'R' program that subtracts the numbers (num) 100 to 1 from a variable called sum. If the sum becomes '0' or less, exit the repeat loop. Use a variable called 'num' for the numbers, and 'sum' for the sum.

sum = 5050

```
repeat {
    sum = sum - num
    num = num - 1
    if (sm == 0) break
}
```
While Loops Quiz

Given two variables (a, b) and a sum = 0, write a while loop to perform the following task: While b > a, increment the variables sum and ‘a’, and decrement the variable 'b'.

```
a = 1; b = 10
while (b>a) {
    sm = sm + 1
    a = a + 1
    b = b - 1
}
```
Functions Quiz

The given function is expecting variables to be in the order x,y,z. Fill in the blanks to call the function for each situation.

Assume x=10, y=20, z=30

| Call foo with the variables in x,y,z, order | foo(10,20,30) |
| Call foo with the variables in y,x,z order | foo(y=20, x=10, z=30) |
| Call foo with the variables x and y set to default, z = 30 | foo(z = 30) |
myPower = function (bas = 10, pow = 2) {
    res = bas^pow
    return(res)
}
myPower(2, 3)

myPower(pow = 3, bas = 2)

myPower(bas = 3)
Vectorized Code

a = 1:10000000; res = 0
system.time(for (e in a) res = res + e^2)

## user  system elapsed
## 3.742 0.029 3.800
Vectorized Code

system.time(sum(a^2))

## user  system elapsed
##  0.180  0.032  0.250
Newer packages: Rcpp, RcppArmadillo, RcppEigen

```r
dyn.load("fooC2.so")  # load compiled C code

A = seq(0, 1, length = 10)
B = seq(0, 1, length = 10)
.Call("fooC2", A, B)
```
<table>
<thead>
<tr>
<th></th>
<th>13.34</th>
<th>17.48</th>
<th>21.21</th>
<th>24.71</th>
<th>28.03</th>
<th>31.24</th>
<th>34.34</th>
<th>37.37</th>
<th>40.33</th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td>##</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
<td>[1]</td>
</tr>
</tbody>
</table>

**External/ Native API**

![Graph showing computational time vs array size](image)

- **C**
- **R**

**Language**

- **array size**
- **computational time (sec)**