Introduction to R

Li Jiang
jiang@cbs.dtu.dk
What is R?

• A software environment within which statistical techniques are implemented.

• Effective facilities for data manipulation, calculation, graphical display and data storage.

• A programming language and highly extensible via packages.
Running R on your own computer

http://mirrors.dotsrc.org/cran/

http://www.rstudio.com
Running R on a remote Unix server

• You can connect to CBS server
  1. Open a terminal, and type
     ssh –X your_student_account@padawan.cbs.dtu.dk
  2. Once log in, under your working directory, and type
     /tools/bin/R-3.0

• To display R graphics, you need to install
  I. XQuartz (Mac), or
  II. MobaXterm (Windows)
Data manipulation and calculation
Data types

character
- "a"
- "this_is_a_string"
- "text"

logical
- TRUE
- FALSE

numeric
- 0.7
- 3.14159265359
- 365
- 1e6
- 10^3
- 3.14159265359
1D
vector

2D
data frame
or
matrix

3D
list
### Data Frame

You can mix data types.

<table>
<thead>
<tr>
<th>NW</th>
<th>E</th>
<th>SE</th>
<th>W</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>up</td>
<td>down</td>
<td>up</td>
<td>down</td>
<td>down</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

### Matrix

Only one data type.

<table>
<thead>
<tr>
<th>columns →</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>columns →</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
</tr>
<tr>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
</tr>
</tbody>
</table>

### Example

- John: red, dog, y
- Paul: blue, cat, y
- Jane: red, none, n
- Frank: green, dog, y
- Donna: blue, fish, n
Assigning values to variables

\[ \text{variable\_name} \leftarrow \text{value\_of\_variable} \]

Examples:

\[ \text{x} \leftarrow 10 \]

\[ \text{My\_name} \leftarrow "Andrea" \]

\[ \text{hours\_in\_a\_year} \leftarrow 24 \times 365 \]
# Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>add</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>subtract</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>multiply</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>divide</td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>to the power of</td>
<td>$10^3 = 1000$</td>
</tr>
<tr>
<td>e</td>
<td>x10 to the power of</td>
<td>$1e3 = 1 \times 10^3$</td>
</tr>
<tr>
<td>%%</td>
<td>modulus</td>
<td>$10 %% 3 = 1$</td>
</tr>
</tbody>
</table>
Functions

FunctionName(argument)

Argument:

• can be a value
• or the name of a variable referring to a value
• there can be many arguments, separated by commas

Examples:

\[ \text{sqrt}(400) \]
\[ \text{log}(x) \]
\[ \text{log}(x, \text{base} = 10) \]

Getting help:

?qsqrt
help(log)
1D

vector

2D

data frame

or

matrix

3D

list
<table>
<thead>
<tr>
<th>1D</th>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector</td>
<td>data frame</td>
<td>list</td>
</tr>
</tbody>
</table>

- **Vector of numbers**
  - my_vector <- c(1, 2, 3)
  - my_vector <- c(1:3)

- **Append more numbers to the end of the vector**
  - my_vector <- c(my_vector, 4, 5, 6)

- **Mix different data types in a vector**
  - my_mixed_vector <- c(1, "two", sqrt(9))
Dataframe from 2 vectors

```
my_dataframe <- data.frame(my_vector, my_vector^2)
```

Matrix of 2 rows and 3 columns

```
my_matrix <- matrix(my_vector, nrow = 2, ncol = 3)
```

Turn a dataframe in to a matrix

```
ew_dataframe <- as.data.frame(my_matrix)
```
Make a list of mixed data types

my_list <- list(my_vector, my_dataframe, my_matrix)

Put a list inside a list

new_list <- list(my_vector, my_list)
Let’s try

Make a vector of numbers:

\[ x \leftarrow c(1:10) \]

Make a “power of” table:

\[ \text{my_table} \leftarrow \text{data.frame}(x, x^2, x^3) \]

Look at it:

\[ \text{my_table} \]

You can give it better names:

\[ \text{colnames(my_table)} \leftarrow c(“x”, “squared”, “cubed”) \]
Let’s try

Let's say the numbers in `my_table` are the diameters of a lot of circles.

Let's then calculate the area of each of those circles, using the well-known formula:

\[
\text{Area} = \pi r^2
\]

\[
\text{my}_\text{radius} \leftarrow \text{my}_\text{table}/2
\]

\[
\text{my}_\text{r2} \leftarrow \text{my}_\text{radius}^2
\]

\[
\text{my}_\text{areas} \leftarrow \pi*\text{my}_\text{r2}
\]

Now, keep the data frames of matching diameter and area values together in a list:

\[
\text{my}_\text{circles} \leftarrow \text{list(}\text{my}_\text{table}, \text{my}_\text{areas})
\]

\[
\text{names(}\text{my}_\text{circles}) \leftarrow \text{c("diameter", "area")}
\]
List

• Our list consists of 2 things.
• I like to think of lists as items on a rope…
• You can keep adding things on the rope.
• You can even hang a new rope on the existing rope.

Item #1 is a data frame of diameter values
Item #2 contains the areas we calculated
Data

You’ve learnt about

• data types
• and how to make them

Now we will learn

• how to retrieve specific parts of data
• and then we will start data import/export
Indexing

• You can access specific parts of your data using “indexing”

• An “index” explains where the data is located. It is like coordinates on a map.
Indexing

\[ x[8] \]

\[ x[4:6] \]

\[ x[c(2, 4, 6, 8)] \]

\[ my\_table[2, 5] \]

\[ my\_table[2:7, 3:5] \]

\[ my\_table[, 1:5] \]
Indexing

- Can be used both to read and to change data

\[
\begin{align*}
x & \leftarrow c(1:10) \\
\text{What does } x \text{ look like?} \\
y & \leftarrow x[6:10] \\
\text{What does } y \text{ look like?} \\
x[6:10] & \leftarrow 0 \\
\text{What does } x \text{ look like now?}
\end{align*}
\]
Filtering your data

Which are higher than 4?

\[ x > 4 \]

Use that to select the values from \( x \)

\[ x[x > 4] \]

Get the parts of \( y \) that equal 4

\[ y[y == 4] \]
Filtering your data

Let's get the heights of those people that are older than 30.

Do the indexing by the “age” vector:

\[ \text{age} > 30 \]

Use that to filter the “height” vector:

\[ \text{height}\{\text{age} > 30\} \]
Filtering your 2D data

Because the data is 2D, we need 2 indexes: [rows, columns]

data[2, 3]

But this time, we need some more space…

data[ , ]
If you are working with a data frame (and not a matrix!) you can refer to a specific column using $ and/or its name.

In the data frame on the previous slide, these 3 examples all return the same vector:

```r
data[,1]

data$age

data[,"age"]
```
Filtering your 2D data

You only want the rows where “sex” equals “female”

\[
data[\text{data$sex == "F"}, \text{c("age")}]\]

Again, you only want the females, but now you only want their ages

\[
data[\text{data$sex == "F"}, \text{c("age")}]\]

This time you want age and height for all females

\[
data[\text{data$sex == "F"}, \text{c("age", "height")}]\]
## Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
<th>Result of example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>is equal to</td>
<td>3 == 2, 3 == 3, 3 == 4</td>
<td>FALSE, TRUE, FALSE</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>and</td>
<td>TRUE &amp; TRUE, TRUE &amp; FALSE, FALSE &amp; FALSE</td>
<td>TRUE, FALSE, FALSE</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>or</td>
<td>TRUE</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
<td>3 &gt; 2, 3 &gt; 3, 3 &gt; 4</td>
<td>TRUE, FALSE, FALSE</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>smaller than</td>
<td>3 &lt; 2, 3 &lt; 3, 3 &lt; 4</td>
<td>FALSE, FALSE, TRUE</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
<td>3 &gt;= 2, 3 &gt;= 3, 3 &gt;= 4</td>
<td>TRUE, TRUE, FALSE</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>smaller than or equal to</td>
<td>3 &lt;= 2, 3 &lt;= 3, 3 &lt;= 4</td>
<td>FALSE, TRUE, TRUE</td>
</tr>
<tr>
<td><code>%in%</code></td>
<td>is in</td>
<td>4 %in% c(1,2,3,4,5), 8 %in% 1:5, c(1,8) %in% 1:5</td>
<td>TRUE, FALSE, FALSE</td>
</tr>
</tbody>
</table>
Filtering your 2D data

You want the rows where “sex” equals “female” AND whose age is less than 30

\[
\text{data[} \text{data$sex == "F" & data$age > 30, ]}
\]

You want all patients who weigh more than 80, OR are at least 180 cm tall

\[
\text{data[} \text{data$weight > 80 | data$height >= 180, ]}
\]
R Data Import/Export

- For any R object
  - `load()`
  - `save()`

- For spreadsheet-like data (data frame, matrix)
  - `read.table()`
  - `write.table()`
Be aware of

- There are arguments for these functions, such as
  - `file = ""` - name of the file
  - `sep = "","` - comma separated columns
  - `header = T` - first line contains the names for the columns

- For arguments or default settings of a function, e.g. `read.table()`
  - `?read.table`

- Default settings can be overwritten with specific values, such as
  - `sep = "\t"` - tab separated columns
  - `header = F` - first line does not contain the names for the columns
For a R object

# For example, a vector

```r
x

[1] 1 2 3 4 5 6 7 8 9 10
```

# Export to an external file

```r
save(x, file="tmp.RData")
```

# Import into R

```r
load("tmp.RData")
```
Several R objects

# Export to an external file
save(x, my_table, my_areas, file="my_cycle.RData")

# Import into R
load("my_cycle.RData")
For a data frame or a matrix

# Export to an external file
write.table(my_table, file="my_table.txt", sep="\t", col.names=T, row.names=F, quote=F)

# Import into R

table_name <- read.table("my_table.txt", sep="\t", header=T)