Variables to Data Frames

Lecture 2

Objectives

- To know the various types of variable: numeric, character, logical
- To know the different kinds of data structure: vectors, matrices, data frames

Good coding practices

- Organize files of a project in one directory
- Write a code just like writing a lab notebook (objectives, packages, source files, annotations)
- Use the pound sign **#** to annotate your code
  - To facilitate understanding for yourself and others
  - To explain why, how, comments, observations, results
- Use file names and variable names that are concise and meaningful

R as calculator

```
# Addition
> 1+2
```

```
[1] 3
```

```
# Multiplication
> 2*3
```

```
[1] 6
```

```
# Subtraction
> 3-2
```

```
[1] 1
```

```
# Division
> 6/3
```

```
[1] 2
```

R as calculator

```
# Exponentiation (x^y), e.g. 3^2
> 3**2
> 3^2
```

```
[1] 9
```

```
# Predefined constant: π
> pi
```

```
[1] 3.141593
```

R as calculator

```
# Logarithmic function: log(x), log2(x), log10(x), e.g. log10(1000)
> log10(1e3)
> log(1000, base = 10)
```

[1] 3

```
# Exponential function: e^x, e.g. e^2
> exp(2)
```

[1] 7.389056

Assignment statement

object_name <− value

- press “**Alt**” + “**minus sign**”
- you may use equal sign “=” but may confuse with logical operators, “==”

```
# Assign the variable r the value of e
> r <− exp(1)

# Evaluate twice the value of r
> r * 2
```

```
[1] 5.436564
```

Variable name in R

- give easy to understand variable name, e.g. sex, date_admission, age_year
- can be created using letters, numbers, symbols (dot, underscore, hyphen)
- must not start with a number

```
# Name a variable for age of children in years
> age_year
> age.year
```

```
# Don’t do this
> 1year
```

Error: unexpected symbol in "1year"

Variable type in R

- **Numeric**:
  - double precision real number, floating point value (e.g. **2.05**)
  - integer or whole number (e.g. **2**)
- **Character**: sequence of strings
  - "some words"
- **Boolean**: values correspond to **True** or **False**
  - FALSE

Data structures in R

- **Vector** - a collection of element of the same type (numbers, strings, boolean)
- **Array** - a vector with one or more dimensions
- **Matrix** - a two-dimensional array
- **Data Frame** - an array in which the type of each element can be different
- **List** - collection of objects of any type, e.g. list of vectors, list of data frames

Vectors in R

- An ordered list of homogeneous elements

```
# Create a vector for age of patients and assign values
> age_year <- c(1,5,2,3,4)

# Print the contents of age_year
> print(age_year)
```

```
[1] 1 5 2 3 4
```

```
# Use typeof( ) function to determine vector type
> typeof(age_year)
```

```
[1] "double"
```

Character vectors

- Character strings are delimited in double quotes " "

```
# Use concatenate function c( ) to combine values
> patient_ID <− c("P001", "P002", "P003", "P004", "P005")
> print(patient_ID)

# Name a variable for sex of patient and assign a value
> sex <− c("female", "female", "female", "male", "male")
```

```
# Use typeof( ) function to determine vector type
> typeof(patient_ID)
```

[1] "character"

Matrix

- A collection of elements of the same data type (numeric, character, or boolean) with a fixed number of rows and columns
- A 2-D array since it has rows and columns

```
# Create a 3x3 matrix using the matrix( ) function
> a_matrix <− matrix(c(1:9), byrow=TRUE, nrow=3)
> a_matrix
```

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

Data Frame

- Use to store data table in R
- Can be considered as a matrix in which columns can contain different data types

```
# Create a data frame from pre-existing objects using data.frame( ) function
> patient_info <− data.frame(patient_ID, sex, age_year)
> print(patient_info)
```

	patient_ID	sex	age_year
1	P001	female	1
2	P002	female	5
3	P003	female	2
4	P004	male	3
5	P005	male	4

Data Frame

```
# Get attributes of data frame
> attributes(patient_info)
```

```
$names
[1] "patient_ID"    "sex"    "age_year"

$class
[1] "data.frame"

$row.names
[1] 1 2 3 4 5
```

```
# Get column names
> colnames(patient_info)
```

```
# Get row names
> rownames(patient_info)
```

List

- Use to store a variety objects under one name

```
# Create a list containing a vector, a matrix, and a data frame
# using list( ) function
> a_list <− list(age_year, a_matrix, patient_info)
> print(a_list)
```

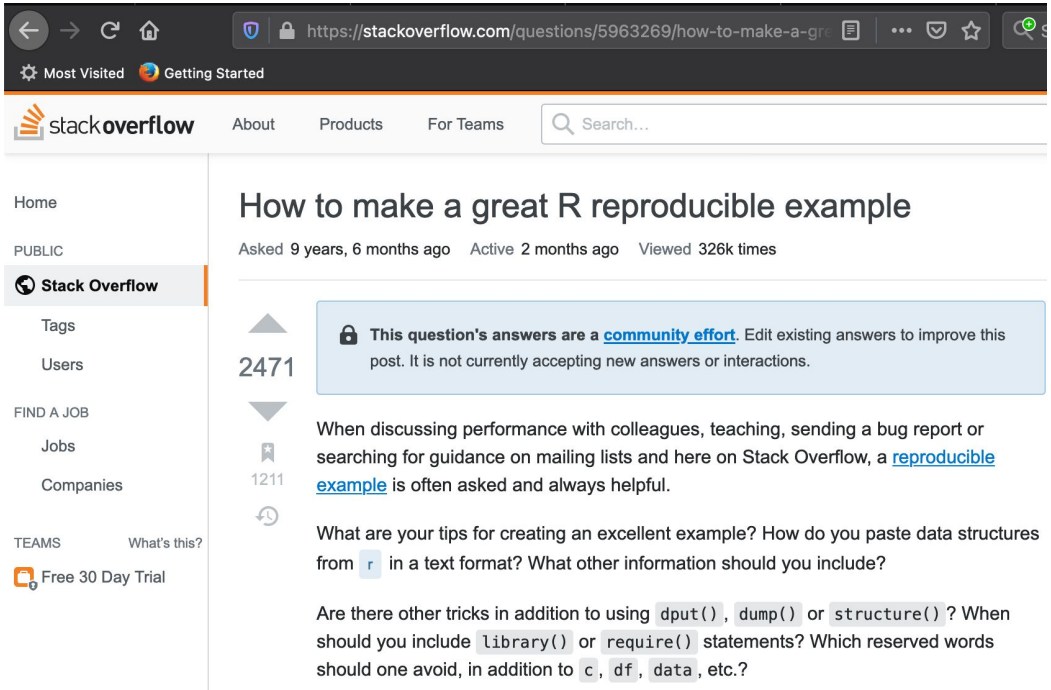
```
[[1]]
[1] 1 5 2 3 4

[[2]]
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

[[3]]
  patient_ID   sex age_year
1       P001 female        1
2       P002 female        5
```

Common problems during writing codes

- Wrong spelling
- Caps and small letters
- Missing pair of " ", ( ), { }, [ ]
- Incomplete expression (return a ‘+’ sign on console)
- Wrong syntax

How to solve?

- ?function_name to read documentation and examples
  - ?sqrt()
  - help(sqrt)
- Google is your friend!
  - copy the error message
  - search in Stack Overflow



Saving workspace

- At the end of each R session, you are prompted to save your workspace
- If you click “Yes”, all objects are saved to the **.RData** file
- When R is re-started, all save objects will be reloaded from .RData file
- Command history stored in .**Rhistory** is also reloaded

Take away message

- R has different types of variables: numeric, character, logical
- R can handle various types of data structures: vectors, matrices, data frames, lists