Getting Started—A Cheat Sheet

Benjamin Buchmuller

## Basic Interactions

### Code and Comments

* # is the comment character
* a = b and a <- b or b -> a **assigns** the value of b to a variable named a
* function\_a %>% function\_b **pipes** the result of function\_a to function\_b as input

Object names cannot start with a number, some names are reserved.

### Functions and Packages

The basic idiom to define a function is new\_function <- function(arg1, arg2, ...) {...}.

Packages

* include new functions for a specific task,
* are installed with install.packages("some\_package"),
* are loaded with library(some\_package).

Functions from a package can be specifically pointed with some\_package::this\_function(...).

### Getting Help

If you know the command’s name use ?command.

If you don’t know the name, but a keyword, use ??keyword.

## Data Types

|  |  |  |
| --- | --- | --- |
| data type | stores | example |
| double | floating point numbers | 3.141, 1.8e-3 |
| integer | integers | -5L, 1L, 2L, 3L |
| character | strings | "banana", "apple", "melon" |
| logical | boolean values | TRUE, FALSE |
| list | sets with different data types | list(3, "apple") |

typeof(obj) # data type of obj  
class(obj) # class of obj; will tell functions how treat obj  
attributes(obj) # metadata of obj; a named list

## Vectors

Vectors contain elements of one single data type only!

length(obj) # number of elements in obj  
c(obj\_a, obj\_b) # concatenate obj\_a and obj\_b; may involve coercion   
names(obj) # names of each element in obj; if set, a character vector

### Numerical Vectors

1:10 # vector of type integer  
seq(1, 10, 0.5) # vector of type double

### Logical Vectors

1:4 > 2 # vector of type logical

# negation of logical vectors  
!(1:4 > 2)  
# boolean operations   
c(TRUE, FALSE) & c(TRUE, TRUE) # AND : c(TRUE, FALSE)  
c(TRUE, FALSE) | c(TRUE, TRUE) # OR : c(TRUE, TRUE)  
# boolean queries  
any(...) # at least one element TRUE?  
all(...) # all elements TRUE?

### Indexing Vectors

# index by position  
noble\_gases[c(1, 3, 5)]  
# index by exclusion  
noble\_gases[c(-1, -3)]  
# index by name  
noble\_gases[c("Argon", "Radon")]  
# index by logical  
noble\_gases[noble\_gases < 20]

### Applying Functions to Vectors

Functions that take *one* vector as input are applied fun(obj).

Functions that take *two* vectors as input are applied fun(obj\_a, obj\_b), this includes arithmetic operations obj\_a + obj\_b.

* If obj\_a and obj\_b have the same number of elements, fun is applied by pairs,
* if length(obj\_a) is a multiple of length(obj\_b), obj\_b is recycled along obj\_a to make pairs,
* else, there will be a result with a warning.

To apply a function to vectors inside a list, the basic idiom for is lapply(obj, fun).

One can apply even complicated functions using lapply(obj, function(i) ...).

Using vectorization is preferred over for-loops!

## Working with Strings

# number of characters  
nchar(chr)  
  
# glue together chr\_a and chr\_b by pairs of elements  
paste(chr\_a, chr\_b) # with sep=...  
paste0(chr\_a, chr\_b) # with no separator  
# glue together chr\_a and chr\_b by pairs of elements, then combine in a single string  
paste(chr\_a, chr\_b, collapse = ...) # with sep=... between elements  
paste0(chr\_a, chr\_b, collapse = ...) # with no separator  
# split chr\_a at each occurence of chr\_b  
stringr::str\_split(chr\_a, pattern = chr\_b)  
  
# extract the substring from the fourth to the eighths character  
stringr::str\_sub(chr, start = 4, end = 8)  
  
# find first chr\_b (can be regex) in chr\_a  
stringr::str\_extract(chr\_a, pattern = chr\_b)  
# find all chr\_b (can be regex) in chr\_a  
stringr::str\_extract\_all(chr\_a, pattern = chr\_b)  
  
# replace first chr\_b (can be regex) in chr\_a with chr\_c  
stringr::str\_replace(chr\_a, pattern = chr\_b, replacement = chr\_c)  
# replace all chr\_b (can be regex) in chr\_a with chr\_c  
stringr::str\_replace\_all(chr\_a, pattern = chr\_b, replacement = chr\_c)

### Regular Expressions

#### Character classes.

|  |  |
| --- | --- |
| operator | meaning |
| . | any character (except \n) |
| \s | any whitespace |
| \t | horizontal tabulation signs |
| \n or \r | line feeds or carriage returns |
| \d | any digit (0, 1, 2, …, 9) |
| \w | any alphabetic and decimal numbers |
| [abc] | a, b, or c |
| [a-c] | every character between a and z (case-sensitive) |
| [^abc] | anything except a, b, or c |

#### Quantifiers.

|  |  |
| --- | --- |
| operator | repetitions |
| ? | 0 or 1 |
| + | 1 or more |
| \* | 0 or more |
| {n} | exactly *n* times |
| {n,} | *n* or more times |
| {n,m} | between *n* and *m* |
| ...? | (make not greedy) |

#### Anchors.

|  |  |
| --- | --- |
| operator | position |
| ^ | start of a line |
| $ | end of a line |