Introduction to Data Structures and Functions in R

Yingqi Jing

April 5, 2022

Contents

1 Introduction

1.1 Data types (numeric, character & logical)

- numeric (integer or real)
- character/string
- logical (T/F)

```
pi = 3.14
lang = "Estonian"
exist = TRUE
```

1.2 Data structures (vector, factor, matrix, data.frame, list)

- vector
- factor
- matrix (n-d array)
- data.frame
- list

```
myvec = c(1, 3, 5, NA)
myfact = factor(c("male", "male", "female"), levels = c("male", "female"))
mymat = matrix(1:12, nrow = 3, ncol = 4, byrow = T)
mydf = data.frame(id = 1:4, lang = c("Estonian", "Finnish", "Hungarian", "North Saami"),
                  subfam = factor(c("Finnic", "Finnic", "Ugric", "Saami")))
myls = list(myvec, myfact, mymat, mydf)
```

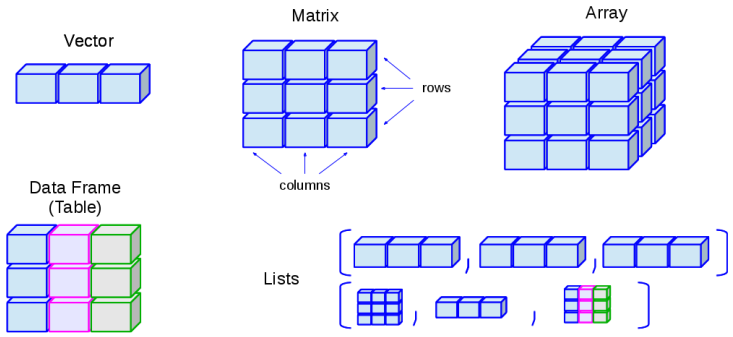

Figure 1: Common data structures in R

It is important to know what type of data you are working with, since all functions and commands in R are defined to specific data types. For example, you can calculate the `sum` of a numeric vector rather than a factor. Sometimes, you need to convert between different data types and structures (see table below).

```
as.character(myvec)
as.integer(myfact)
as.data.frame(mymat)
```

from \ to	vector	matrix	data.frame	list	xtabs
vector		*matrix(d)*	*data.frame(d)*	*list(d)*	
matrix	*as.vector(d)*		*as.data.frame(d)*	*convertRowsToList(d)* *convertColsToList(d)* *in package "BBmisc"*	
data.frame	*as.vector(as.matrix(d))*	*as.matrix(d)*		*split(df, row.names(df))* *convertColsToList(df)* *in package "BBmisc"*	*xtab(y~x+b)* *or* *as.xtabs(df, rowvar="x", colvar="y") in package "mosaic"*
list	*unlist(d)*	*? ldply(d, cbind)* *ldply(d, rbind)* *in package "plyr"* *or* *do.call("rbind", d)*	*?ldply(d, cbind)* *ldply(d, rbind)* *in package "plyr"*		
xtabs			*as.data.frame.matrix(d)*		

Figure 2: Convert between different data structures in R

1.3 Data.frame manipulation

1.3.1 Combine two data.frames (rbind & cbind)

```
x = data.frame(i = c("a", "b", "c"), j = 1:3, stringsAsFactors = FALSE)
y = data.frame(i = c("d", "e", "f"), j = 4:6, stringsAsFactors = FALSE)
rbind(x, y)
```

```
x = data.frame(i = c("a", "b", "c"), j = 1:3, stringsAsFactors = FALSE)
y = data.frame(m = c("d", "e", "f"), n = 4:6, stringsAsFactors = FALSE)
cbind(x, y)
```

1.3.2 Join two data.frames (inner_join, left_join & right_join)

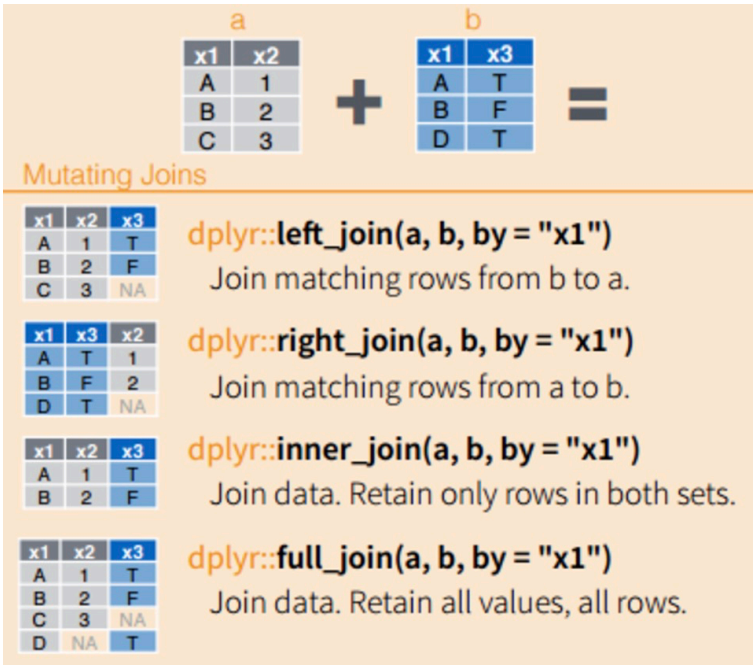

Figure 3: Joint two data frames by certain column

```
lang_df = data.frame(id = 1:4,
                     lang = c("Estonian", "Finnish", "Hungarian", "North Saami"),
                     value = c(0, 1, 0, 1))
subfam_df = data.frame(lang = c("Estonian", "Finnish", "Hungarian", "North Saami"),
                       subfam = c("Finnic", "Finnic", "Ugric", "Saami"))
inner_join(lang_df, subfam_df, by = "lang")
```

```
  id        lang value subfam
1  1    Estonian     0 Finnic
2  2     Finnish     1 Finnic
3  3   Hungarian     0  Ugric
4  4 North Saami     1  Saami
```

1.3.3 Reshape a data.frame (pivot_wider & pivot_longer)

It is quite common to convert data.frame between wide and long formats, so that you can easily aggregate and summarise the results. Here I am going to use `pivot_wider` and `pivot_longer` to reshape the data.

Note: if it does not give you the right format, it is probably due to the duplicated row id you have in your data.

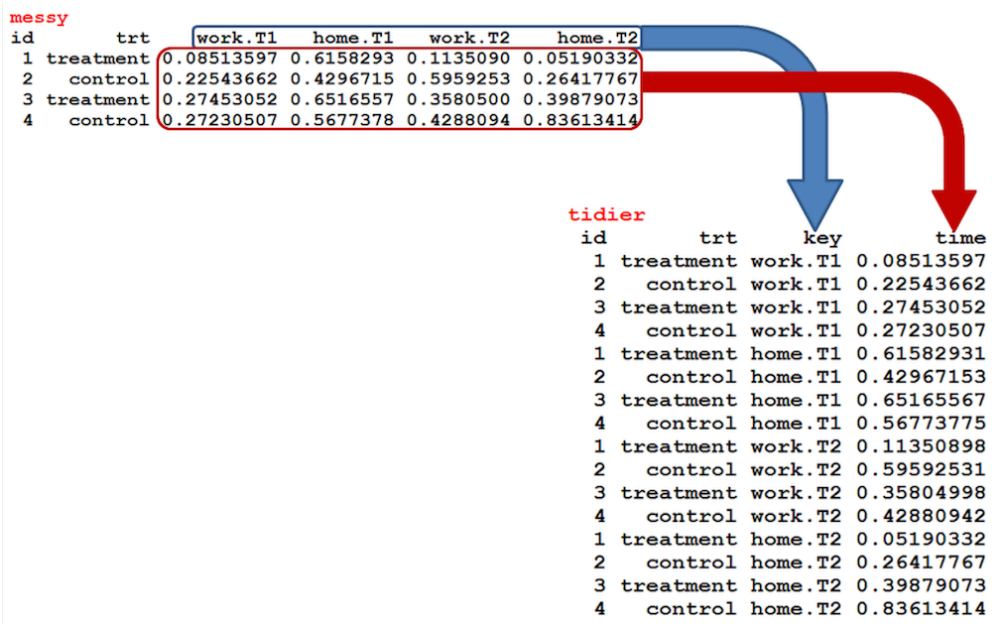

Figure 4: Data transformation from the wide to long format

```
lang_trait = data.frame(lang = c("Estonian", "Finnish", "Hungarian", "North Saami"),
                        trait1 = c(0, 1, 1, 0),
                        trait2 = c(0, 0, 1, 0),
                        trait3 = c(1, 0, 1, 1))
df_long = pivot_longer(lang_trait, -lang,
             names_to = "feature", values_to = "value")
# alternatively: gather(lang_trait, key = feature, value = value, -lang)
```

```
df_wide = pivot_wider(df_long, names_from = "feature", values_from = "value")
# alternatively: spread(df_long, key = feature, value = value)
```

1.4 Exercise

(1) Pls load the UraTyp values and language table, and join the two tables by Language ID. Note: the two data.frames have different names for the language ID.

```
uratyp_df = read.csv("../Data/uratyp-1.1/cldf/values.csv")
uratyp_df = uratyp_df[, c("Language_ID", "Parameter_ID", "Value")]

lang_df = read.csv("../Data/uratyp-1.1/cldf/languages.csv")
lang_df = lang_df[, c("ID", "Name", "Subfamily")]
uratyp_final = inner_join(uratyp_df, lang_df, by = c("Language_ID" = "ID"))
```

2 R programming language

2.1 Conditional statement

```
lang = "Finnish"
if(lang == "Estonian"){
  print("It is true!")
}else{
  print("It is false!")
}
```

```
[1] "It is false!"
```

2.2 For loop

```
langs = c("Estonian", "Finnish", "Hungarian", "North Saami")
for(l in langs){
  print(l)
}
```

```
[1] "Estonian"
[1] "Finnish"
[1] "Hungarian"
[1] "North Saami"
```

2.3 Function

```
mysum = function(numvec){
  total = 0
  for(i in numvec){
    total = total + i
  }
  return(total)
}
```

```
mysum(numvec = 1:10)
```

```
[1] 55
```