Introduction to $\mathcal{R}$

Session 2: Data management

Dag Tanneberg[1]

Potsdam Center for Quantitative Research
University of Potsdam, Germany
October 11/12, 2018

[1]Chair of Comparative Politics, UP, dag.tanneberg@uni-potsdam.de

Introduction

Words of Warning

Learning to manage data in $\mathcal{R}$ is hard. *Everybody* struggles at first. Reasons include:

- Where's my spreadsheet?
- Barrage of new concepts
- Explosion of functions and notation
- "Too" many options

Stay with me, and...

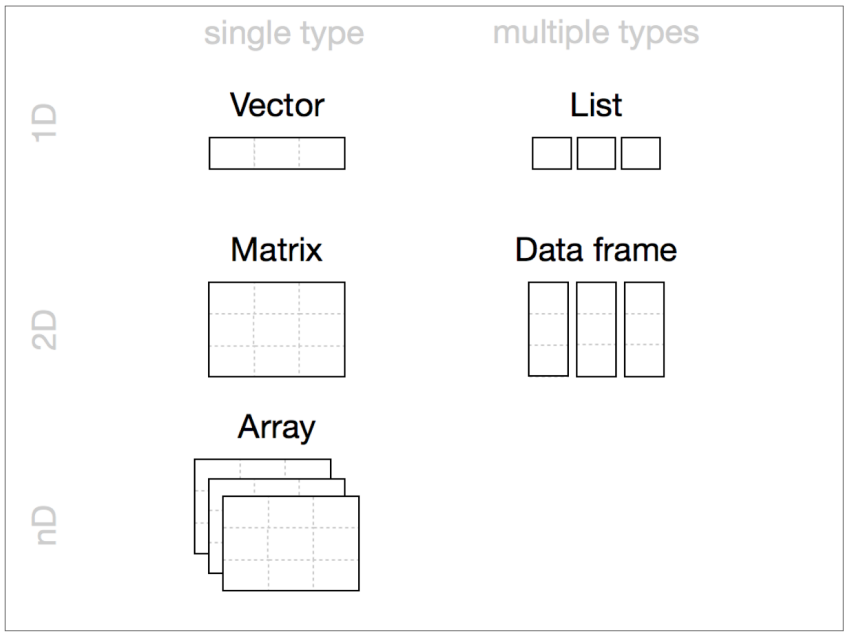
Outline

Data Structures

Organizing Principles[2]

- **Dimensionality**
  How many qualities allowed?
- **Homogeneity**
  All data single-typed?
- **Attributes**
  What metadata available?
- **Today's Focus**
  Vectors, data frames, and attributes

Figure 1: Typical Data Structures



[2] Grolemund, G. 2014. Hands-on Programming with R. Sebastopol: O'Reilly, 62.

Atomic vectors: Creation, Dimensionality, Homogeneity

- **Creation**

```
die <- 1:6 # Remember our first session?
die <- c(1, 2, 3, 4, 5, 6) # Standard assignment: c()
```

- **Dimensionality** 1D set of data points
- **Homogeneity** All data must have the same type.
  - Logical: TRUE or FALSE values
  - Integer: Whole numbers ($\mathcal{Z}$)
  - Double a.k.a. Numeric: Real numbers ($\mathcal{R}$)
  - Character: Text data

```
logic <- c(TRUE, FALSE, FALSE); typeof(logic)
int <- c(-1L, 99L); typeof(int)
dbl <- sqrt(8L); typeof(dbl)
chr <- c(TRUE, "Hello", 2.0, "R"); typeof(chr)
```

Atomic Vectors: Attributes

- Provide information *about* your vector
- Atomic Vectors have attributes type, length, and names.
  - type: What kind of data is it?
  - length: How many elements does your vector contain?
  - names: What do you call an element?

```
die <- 1:6
length(die) # Check the length of an atomic vector
length(c(die, die)) # Will the length differ?
names(die) # Explain the output.
names(die) <- c(
  "one", "two", "three", "four", "five", "six"
)
die; die + 1
names(die) <- NULL # Reset the attribute
```

Data Frames: Creation

- **Creation**

```
students <- data.frame(
  first_name = c("Alex", "Jessy", "Barbara", "Jacob"),
  student_id = c(349857, 796245, 143577, 987456),
  passed = c(TRUE, TRUE, FALSE, FALSE),
  stringsAsFactors = FALSE # Will be explained later
); students
```

```
##   first_name student_id passed
## 1       Alex     349857   TRUE
## 2      Jessy     796245   TRUE
## 3    Barbara     143577  FALSE
## 4      Jacob     987456  FALSE
```

Data Frames: Dimensionality, Homogeneity, Attributes

- **Dimensionality**
  - 2D table of grouped vectors
  - Vectors must have equal length (or will be recycled)
- **Homogeneity**: Data can be of any type.
- **Attributes**: (at least) names and dimensions
  - Names: What are your rows and columns called?
  - Dimensions: No. of rows and columns

```
# Brief description
str(students)
# Dimensions
dim(students); nrow(students); ncol(students)
# Names
rownames(students); names(students) # also: colnames()
```

Data I/O

- **Input**
  - **Easy way**: RStudio's import wizard
  - Structured way: read.table() and variants or package:foreign
- **Output**
  - **Easy way**: Just kidding. There is no easy way.
  - Structured way: write.table(), save(), or package:foreign

```
write.csv(
  x = student_data,
  file = "/PATH/TO/DATA/student_data.csv"
)
```

- **Learn more**
  - The 'R Data Import/Export' manual (see Help in RStudio)
  - https://www.statmethods.net/input/index.html

Break time

Let's catch our breath and take a 5 minute break.

R Notation

So far, we have seen...

- what data structures exist,
- what their elementary properties are,
- what data structures are most important.

The Big Question

How do we access individual values inside our data?

- **Answer**: Indexing
- $\mathcal{R}$ provides numerous ways to index data.
- Most boil down to square brackets: data[ ].
- Dimensionality defines the number of indexes required
  - Vectors are 1D. $\rightarrow$ data$[i]$
  - Data frames are 2D. $\rightarrow$ data$[i, j]$

Integers

- *Positive Integers*:
  - Example: data$[i, j]$, e.g. data$[1, 2]$
  - returns the data indexed by i and j

```
student_data[ , ] # Select the 1st row & col
student_data[ , ] # Select rows 1-5 & cols 1-3
student_data[0, 0] # Explain the result
```

- *Negative Integers*:
  - Example: data$[-i, -j]$, e.g. data$[-1, -2]$
  - returns all data **but** i and j

```
student_data[ , ] # Deselect the 1st row & col
student_data[ , ] # Select rows 1-5, drop cols 2 & 5
student_data[-3:4, 1:2] # Explain the error.
```

Blanks

- Use a blank space to extract every value in a dimension
- Example: data$[i,]$

```
# Exercise: Fill in the blanks
student_data[ , ] # Select the 1st row & all cols
student_data[ , ] # Select all rows for cols 2 & 5
student_data[ , ] # Return the entire data frame
```

Logical Values

- Supply a vector of TRUE & FALSE as your index
- Example: data[c(TRUE, TRUE, FALSE),]

```
student_data[FALSE , ] # Explain the result.
student_data[ , ] # Return only cols 1, 2 & 6
student_data[student_data[, 2] == 1, ] # Explain.
```

Names

- Supply character vectors to select on names
- Requires attribute names
- Example: data[, "*variable*"]

```
student_data[ , ] # Return cols student & student_id
student_data[ , "Student"] # Explain the error.
student_data[ , -"student_id"] # Explain the error.
```

Dollar Signs & Double Brackets

- $ and [[ ]] define a second indexing system
- Applies only to lists and data frames
- Return atomic vectors
- $ requires names, but [[ ]] always works
- Examples: data$variable; data[[variable]]; data[[1]]

```
student_data$ # Return column arts
student_data$ # Return column student_id, rows 1:5
student_data[[ ]] # Extract column gender
student_data[[c("student", "student_id")]] # Explain.
```

A Practical Data Management Challenge

The Challenge

A group of students has taken exams in Science, Arts, and Literature. We want to combine their scores, grade them, and prepare a list for display on our office door. Along the way we have to meet several obstacles.

Part A

1. Some grades are missing. Angela scored 603 in Science, and Cheryl 28 in Literature. Find and fill the gaps.
2. Joel withdrew from class. Remove him from the dataset.
3. Gender was coded accidentily. Drop the variable.
4. The grades are on widely different scales. Scale them to comparable units.

Part B

1. For each student calculate the mean of science, arts, and literature.
2. Find its .2, .4, .6, and .8 quantiles. Assign grades appropriately.
3. Greg was caught cheating. He should receive an F.
4. Create a new data frame grade_data. This data frame should:
   - include only student_id and the final grade;
   - be ordered by student_id.

Congratulations

You just learned a hell lot of $\mathcal{R}$. Enjoy lunch!