Week 5: R Basics, Data Manipulation, and Descriptive Statistics

Instructor: Nazik Dinçtopal Deniz | TA: Onur Keleş

How to use this handout. Copy the code into an R script in RStudio. Lines starting with # are comments, and R ignores them. Place your cursor on a line (or highlight several lines) and press **Cmd + Enter** (Mac) or **Ctrl + Enter** (Windows) to send it to the Console.

0. Warm-up: objects, assignment, and vectors

In R, we store information in *objects* with the assignment arrow `<-`. Shortcut: Option + - (Mac) or Alt + - (Windows). Type an object's name to print its contents. A *vector* is a 1D collection of items of the same type, combined with `c()`.

```
sample_size <- 10
sample_size

vowels <- c("i", "e", "a", "o", "u")              # character vector (text)
f1_sample <- c(280.5, 490.2, 780.0, 510.4, 310.8)  # numeric vector (numbers)
```

1. Working directory

The working directory is the folder where R looks for files to load and saves the files you export. To change it, replace the path with a real folder on your computer, remove the #, and run the line. In RStudio, you can also use *Session → Set Working Directory → Choose Directory*.

```
getwd()
# setwd("~/Documents/LING411/RPractice")
```

2. Creating and inspecting a linguistic dataset

A `data.frame` is R's standard table format, like an Excel spreadsheet. Each **row** is one observation (here, one recorded vowel token), and each **column** is one variable (a vector of equal length). These are illustrative acoustic values, not a representative sample.

```
vowel_data <- data.frame(
  token       = 1:10,   # integer sequence from 1 to 10
  speaker     = c("S1", "S1", "S1", "S1", "S1",
                  "S2", "S2", "S2", "S2", "S2"),
  vowel       = c("i", "e", "a", "o", "u",
                  "i", "e", "a", "o", "u"),
  height      = c("high", "mid", "low", "mid", "high",
                  "high", "mid", "low", "mid", "high"),
  backness    = c("front", "front", "central", "back", "back",
                  "front", "front", "central", "back", "back"),
  f1_hz       = c(280.5, 490.2, 780.0, 510.4, 310.8,
                  300.1, 520.6, 805.2, 535.7, 325.4),
  f2_hz       = c(2250.0, 1840.5, 1220.0, 950.0, 870.5,
                  2180.0, 1795.0, 1285.0, 990.0, 910.2),
  duration_ms = c(95, 110, 145, 120, 90,
                  102, 116, 152, 126, 96),
  rounded     = c(FALSE, FALSE, FALSE, TRUE, TRUE,
                  FALSE, FALSE, FALSE, TRUE, TRUE)
)
```

Inspecting the data frame.

```
print(vowel_data)     # print the entire table
head(vowel_data, 3)   # first 3 rows (useful for huge datasets)
tail(vowel_data, 3)   # last 3 rows

dim(vowel_data)       # number of rows, then number of columns
```

```
nrow(vowel_data)      # number of observations (rows)
ncol(vowel_data)      # number of variables (columns)
names(vowel_data)     # column names

str(vowel_data)       # structure and variable types
```

In the output of `str()`: `int` = integer, `chr` = character (text), `num` = continuous number, `lgl` = logical (TRUE/FALSE).

Converting categorical variables to factors. By default, R sorts text alphabetically (`"high", "low", "mid"`). In linguistics, we often want a meaningful order, for example along the vowel trapezoid. We use `factor()` with `levels`:

```
vowel_data$height   <- factor(vowel_data$height,   levels = c("high", "mid", "low"))
vowel_data$backness <- factor(vowel_data$backness, levels = c("front", "central", "back"))
vowel_data$speaker  <- factor(vowel_data$speaker,  levels = c("S1", "S2"))
vowel_data$vowel    <- factor(vowel_data$vowel,    levels = c("i", "e", "a", "o", "u"))

str(vowel_data)
summary(vowel_data)   # factors show token counts; numbers show summaries
```

3. Exporting and importing data (CSV files)

`row.names = FALSE` stops R from adding an extra unnamed column of row numbers.

```
write.csv(vowel_data, "vowel_measurements.csv", row.names = FALSE)
df_vowel <- read.csv("vowel_measurements.csv")
str(df_vowel)
```

Important. CSV files are plain text, so they do **not** remember factor levels. In `str(df_vowel)`, `height`, `backness` and `speaker` are back to `chr`. When you import a CSV for real analysis, apply the factor levels again:

```
df_vowel$height   <- factor(df_vowel$height,   levels = c("high", "mid", "low"))
df_vowel$backness <- factor(df_vowel$backness, levels = c("front", "central", "back"))
df_vowel$speaker  <- factor(df_vowel$speaker,  levels = c("S1", "S2"))
df_vowel$vowel    <- factor(df_vowel$vowel,    levels = c("i", "e", "a", "o", "u"))
```

4. Indexing, filtering, and subsetting

Method A: the `$` operator extracts a single column as a vector.

```
df_vowel$f1_hz
df_vowel$duration_ms
```

Method B: brackets `data[rows, columns]`. Leaving one side of the comma blank means "keep ALL rows" or "keep ALL columns".

```
df_vowel[1, ]                       # row 1, all columns
df_vowel[1:3, c("vowel", "f1_hz")]  # rows 1 to 3, only 'vowel' and 'f1_hz'
```

Logical operators.

`==`	is exactly equal to (two equals signs!)	`!=`	is not equal to
`>`, `>=`	greater than (or equal to)	`<`, `<=`	less than (or equal to)
`&`	AND (both conditions are true)	`|`	OR (at least one is true)
`%in%`	belongs to a set of values		

```
# 1. Only low vowels (keep the trailing comma to keep all columns):
low_vowels <- df_vowel[df_vowel$height == "low", ]

# 2. Tokens with F1 greater than 500 Hz:
high_f1 <- df_vowel[df_vowel$f1_hz > 500, ]

# 3. AND: rounded vowels longer than 100 ms:
long_rounded <- df_vowel[df_vowel$rounded == TRUE & df_vowel$duration_ms > 100, ]
```

```
# 4. %in%: high OR low vowels (excluding mid):
extreme_height <- df_vowel[df_vowel$height %in% c("high", "low"), ]
```

Method C: `subset()` filters rows and selects columns without repeating `df_vowel$`.

```
front_vowels <- subset(df_vowel, backness == "front")
front_formants <- subset(df_vowel, backness == "front",
                         select = c(speaker, vowel, f1_hz, f2_hz))
```

5. Descriptive statistics (base R)

5A. Categorical variables: frequency and proportion tables.

```
table(df_vowel$height)                        # tokens per vowel height
table(df_vowel$height, df_vowel$backness)     # cross-tabulation
prop.table(table(df_vowel$height)) * 100      # percentages
```

5B. Continuous variables: central tendency and dispersion. Note that F1 is inversely related to vowel height: low vowels have high F1.

```
mean(df_vowel$f1_hz)       # arithmetic average
median(df_vowel$f1_hz)     # middle value (50th percentile; robust to outliers)

sd(df_vowel$f1_hz)         # standard deviation (in Hz)
var(df_vowel$f1_hz)        # variance (sd squared; in Hz^2)
range(df_vowel$f1_hz)      # minimum and maximum
min(df_vowel$f1_hz)
max(df_vowel$f1_hz)
quantile(df_vowel$f1_hz)   # 0%, 25%, 50%, 75%, 100%
IQR(df_vowel$f1_hz)        # 75th minus 25th percentile (spread of middle 50%)

summary(df_vowel$f1_hz)    # Min, Q1, Median, Mean, Q3, Max
summary(df_vowel)
```

5C. Grouped descriptive statistics. Averaging F1 over /i, e, a, o, u/ together is not phonetically informative. Instead, we want the mean or SD of F1 *by* vowel height or vowel category.

```
# Option 1: tapply(continuous_variable, grouping_variable, function)
tapply(df_vowel$f1_hz, df_vowel$height, mean)
tapply(df_vowel$f1_hz, df_vowel$height, sd)
tapply(df_vowel$f2_hz, df_vowel$backness, mean)   # front vowels: much higher F2

# Option 2: aggregate(outcome ~ group, data = ..., FUN = ...)
aggregate(f1_hz ~ height, data = df_vowel, FUN = mean)
aggregate(duration_ms ~ height + rounded, data = df_vowel, FUN = mean)   # two factors
```

5D. Missing data (NA). In real phonetic or psycholinguistic data, some tokens fail to track or are excluded. R marks missing values as `NA` (Not Available). By default, `mean()` returns `NA`. Add `na.rm = TRUE` to compute the statistic on the remaining tokens.

```
f1_with_missing <- c(280.5, 490.2, NA, 510.4, 310.8)
mean(f1_with_missing)                 # NA
mean(f1_with_missing, na.rm = TRUE)
sd(f1_with_missing, na.rm = TRUE)
```

6. Practice exercises

Exercise 1. Filter `df_vowel` to create a new data frame called `back_vowels` that contains only vowels where `backness` is `"back"`.

Exercise 2. Using your new `back_vowels` data frame, what is the mean duration (`duration_ms`) of back vowels vs. front vowels?

Exercise 3. Calculate the median and IQR of F2 (`f2_hz`) across the whole dataset.

Exercise 4. Filter `df_vowel` to find all tokens produced by Speaker 2 (`"S2"`) that have an F2 higher than 1500 Hz. Which vowels are they?