Biometry. Lecture 9

Alexey Shipunov

Minot State University

February 22, 2016

Outline

Outline

Starting...

`> setwd("<working folder>")`

or

“`Change dir`”

in menu!

On Mac, be sure that startup option is working: `getwd()` (`getwd()` checks if R is in working folder, `dir()` checks the folder content)

Comment to the lab: dotchart

```
> a <- read.table("http://ashipunov.info/data/aronia.txt",
+ h=T, sep="\t")
> tv <- table(a$VEIN.P.N)
> dotchart(tv) # warning
> dotchart(c(tv)) # no warning!
> dotchart(c(sort(tv))) # even better
> dotchart(c(sort(tv)), ylab="Number of veins") # does not fit
> oldpar <- par(mar=c(5,4,4,3))
> dotchart(c(sort(tv)), ylab="Number of veins") # now fits well
> par(oldpar)
```

Lifehack 1: `c()` converts its content to vector

Lifehack 2: margins count clockwise from the topmost ("main")

Types of data

Nominal (categorical) data

Factors to numbers

```
> as.numeric(sex.f)
> w <- c(69, 68, 93, 87, 59, 82, 72)
> x <- c(174, 162, 188, 192, 165, 168, 172)
> plot(x, w, pch=as.numeric(sex.f), col=as.numeric(sex.f))
> legend("topleft", pch=1:2, col=1:2, legend=levels(sex.f))
```

Objects `x`, `sex` and `w` could be height, gender and weight of seven people in small office, respectively.

Factors to ranks

```
> m <- c("L", "S", "XL", "XXL", "S", "M", "L") # t-shirts
> m.f <- factor(m)
> levels(m) # Wrong order, alphabetical
> m.o <- ordered(m.f, levels=c("S", "M", "L", "XL", "XXL"))
> levels(m.o)
```

The danger of factors

```
> a <- factor(3:5)
> a
> as.numeric(a) # wrong!!!
> as.numeric(as.character(a)) # correct
```

Some rules about vectors

- For every type of R object, there are functions `is.<something>()` and `as.<something>()` (e.g., `as.vector()` and `as.numeric()` will convert to vector and to numeric vector, respectively).
- Object names must not start with a number
- R is case-sensitive
- Please avoid to use names of popular functions (like `c()`) and reserved keywords: `T` (TRUE), `F` (FALSE), `NA` (missing data), `NaN` (not a number), `Inf` (result of dividing by zero), also constants like `pi`, `letters` and `LETTERS`

If you want *e* constant, use `exp(1)`

Types of data

Secondary data

Types of secondary data

- Fractions (and percents)
- Counts and ranks

Dotcharts for fractions and percents

These kinds of plots are often much better to read than bar plots and pie charts:

```
> height.2 <- cut(height, 3,
+ labels=c("low", "middle", "high"))
> dotchart(table(height.2)) # ignore the warning
```

Ranks and ties

```
> a1 <- c(1,2,3,4,4,5,7,7,7,9,15,17)
> a2 <- c(1,2,3,4,5,7,7,7,9,15,17)
> names(a1) <- rank(a1)
> a1
> names(a2) <- rank(a2)
> a2
```

Ranks may be decimal, same numbers will have same ranks (ties) and R will tell about that: try, for example

```
> wilcox.test(a2)
```

Types of data

Missing data

Missing data is always here

- All real world data have problems
- Numbers may be corrupted, missed, unknown, duplicated etc.
- “Missing data” is often used to cover part of these cases

Entering missing data

We asked seven office co-workers about their average time of sleep. One denied to answer, other answered “I don’t know”, the third was not in the office at the moment:

```
> h <- c(8, 10, NA, NA, 8, NA, 8)
> h
```

How to operate with NA

```
> mean(h) # error!
> mean(h, na.rm=TRUE)
> mean(na.omit(h))
```

Missing data imputation

```
> h.old <- h
> h[is.na(h)] <- mean(h, na.rm=TRUE)
> h
```

Types of data

Outliers

Too big or too small

- Outliers are results of mistypes, misprints and other random events
- It is possible to recognize them using `table()` and `summary()` functions

```
> h.1 <- c(8, 10, NA, NA, 8, NA, 8, 80)
> table(h.1)
> summary(h.1)
> h[!(h.1 > 24)]
```

Types of data

Data conversion and normalization

Logarithmic conversion

- Very often the bell-shape curve or linear relation may me achieved by logarithmic conversion
- In R, it is possible to apply `log()` (natural logarithm) function to any vector. The only problem is that the data should not contain zeroes.
- Other mathematical conversions also exist, e.g., square root conversion.

Scaling data

Function `scale()` will convert all columns of data frame to the same scale:

```
> a <- 1:10
> b <- seq(100, 1000, 100)
> d <- data.frame(a, b)
> d
> scale(d)
```

Columns `a` and `b` became identical!

Inside R: Matrices, lists and data frames

Matrices

What we already know of R internals

- We already know integer, numeric, character and logical *vectors*. All vectors could be *named*.
- In addition, R has *factors* (ordered or unordered)

Matrices are vectors

- In R, numeric tables (*matrices*) are simply vectors with two dimensions.
- It is also possible to create multidimensional *arrays*.

Matrix and vector

```
> m <- 1:4
> ma <- matrix(m, ncol=2, byrow=TRUE)
> str(ma)
> str(m)
> mb <- m
> dim(mb) <- c(2,2)
> mb
```

The structure of objects `m` and `ma` are not significantly different, only screen output is not similar (try it!). Function `dim()` will add dimensions to vector transforming it into matrix or array.

Three-dimensional matrix (array)

```
> m3 <- 1:8
> dim(m3) <- c(2,2,2)
> m3
```

Inside R: Matrices, lists and data frames

Lists

List is a collection of everything

- List may contain any type of objects
- Moreover, list can contain other lists, and so on

List examples

```
> l <- list("R", 1:3, TRUE, NA, list("r", 4))
> l
> str(l)
> fred <- list(name="Fred", wife.name="Mary", no.children=3,
+ child.ages=c(1,5,9))
> fred
```

Finishing...

Save your commands!

(`savehistory(<todaysdate>.r)` or `File -> Save as...` on Mac)

Summary: most important commands

- `NA` is a missing data
- `[`—selects an element, row or column
- `$`—selects by name from list or data frame

For Further Reading

A. Shipunov.
Biometry [Electronic resource].
2012—onwards.
Mode of access:
http://ashipunov.info/shipunov/school/biol_240

A. Shipunov, and many others.
Visual statistics. Use R!
2016—onwards.
Mode of access: http://ashipunov.info/shipunov/school/biol_240/en/visual_statistics.pdf