Intro to R for Biologists
Session 2
Data exploration

Irina & Rao

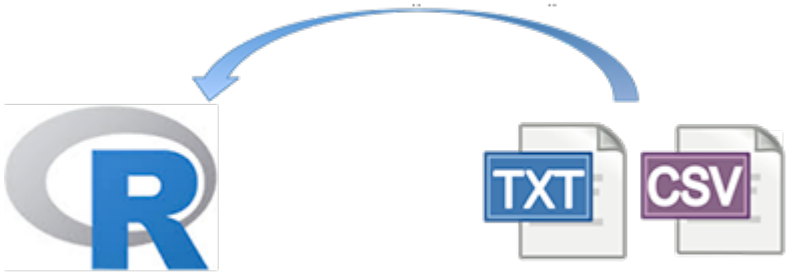
07/07/2021

Summer 2021

INTRO TO R FOR BIOLOGISTS

- **Data exploration**
  - Reading and writing data files
  - Managing working directory
  - Functions to explore data
  - Cleaning data
  - Functions to modify a data.frame
  - Basic plots to visualize data
  - Basic statistics
  - Filtering data

Reading data into R



Base functions:

- **read.table**(file, header = FALSE, sep = "", quote = "\"'", dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"), row.names, col.names, as.is = ! stringsAsFactors, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, …)
- **read.csv**(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, comment.char = "", …)
- **read.delim**(file, header = TRUE, sep = "\t", quote = "\"", dec = ".", fill = TRUE, comment.char = "", …)

Reading data into R

xls, xlsx files

`install.packages("readxl")` - install the package
`library(readxl)` - load the package

- **`read_excel`**`(path, sheet = NULL, range = NULL, col_names = TRUE, col_types = NULL, na = "", trim_ws = TRUE, ...)`
- **`read_xls`**`(path, sheet = NULL, range = NULL, col_names = TRUE, col_types = NULL, na = "", trim_ws = TRUE, ...)`
- **`read_xlsx`**`(path, sheet = NULL, range = NULL, col_names = TRUE, col_types = NULL, na = "", trim_ws = TRUE, ...)`

https://www.rdocumentation.org

More examples (over 21 000 packages)

Managing working directory

The **working directory** is a file path on your computer that sets the default location of any files you read into **R**, or save out of **R**.

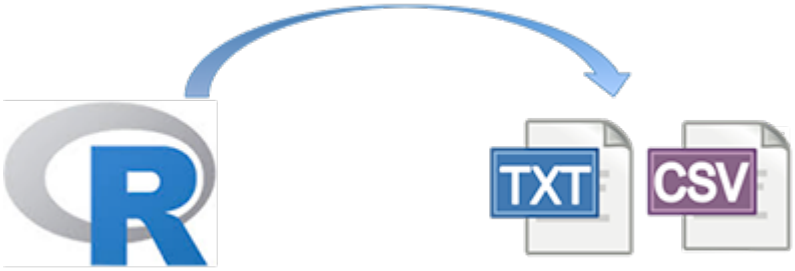
`getwd()` - returns the current working directory

`setwd("path/to/your_new_wd")` - changes the working directory

`list.files()` - displays the list of all the files in wd

Note: `getwd()` function has no arguments!

Write files from R



Base functions:

- `write.table(my_data, file = "my_data.txt", append = FALSE, sep = " ", dec = ".", row.names = TRUE, col.names = TRUE)`

Set separator to `sep = "\t"` to get tab separated txt file

- `write.csv(my_data, file = "my_data.csv")` - comma-separated csv files

Write files from R

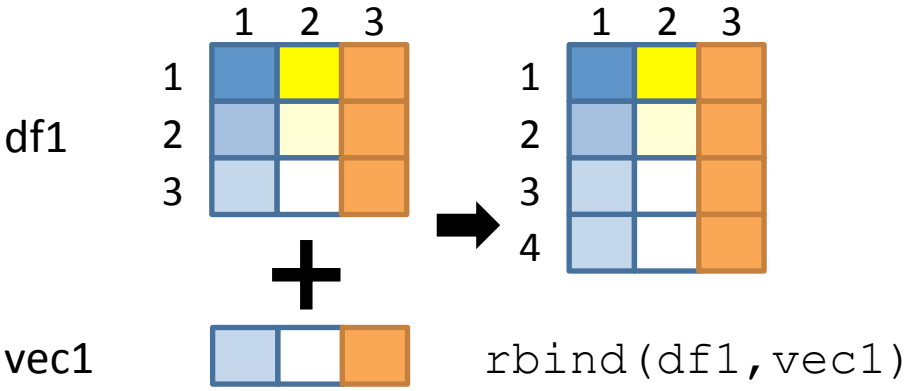
xls, xlsx files

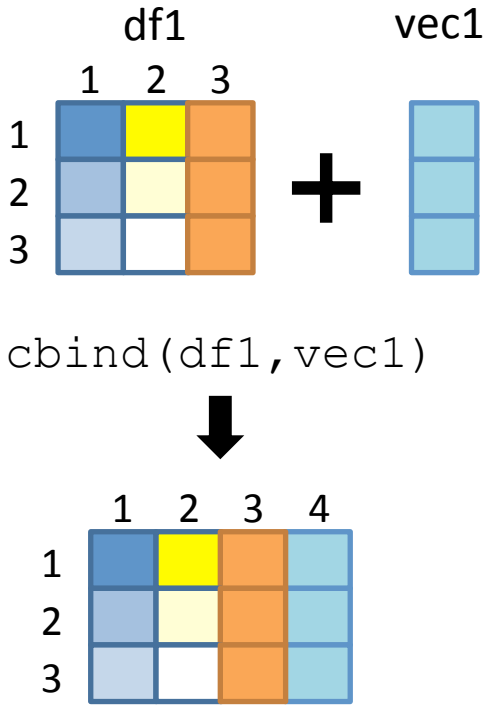
`install.packages("xlsx")` - install the package
`library(xlsx)` - load the package

```
write.xlsx(my_data, file = "result.xlsx", sheetName =
"Sheet1", col.names = TRUE, row.names = TRUE, append =
FALSE)
```

Let’s explore practically

Adding rows or columns to a data.frame





Migration and morphology data

scientific **data**

Explore content | Journal information | Publish with us

nature › scientific data › data descriptors › article

Open Access | Published: 01 March 2017

Systematic high-content genome-wide RNAi screens of endothelial cell migration and morphology

Steven P. Williams, Cathryn M. Gould, Cameron J. Nowell, Tara Karnezis, Marc G. Achen, Kaylene J. Simpson & Steven A. Stacker

Scientific Data **4**, Article number: 170009 (2017) | Cite this article

747 Accesses | **12** Citations | **6** Altmetric | Metrics

Abstract

Many cell types undergo migration during embryogenesis and disease. Endothelial cells line blood vessels and lymphatics, which migrate during development as part of angiogenesis, lymphangiogenesis and other types of vessel remodelling. These processes are also important in wound healing, cancer metastasis and cardiovascular conditions. However, the molecular control of endothelial cell migration is poorly understood. Here, we present a dataset containing siRNA screens that identify known and novel components of signalling pathways regulating migration of lymphatic endothelial cells. These components are compared to signalling in blood vascular endothelial cells. Further, using high-content microscopy, we captured a dataset of images of migrating cells following transfection with a genome-wide siRNA library. These datasets are suitable for the identification and analysis of genes involved

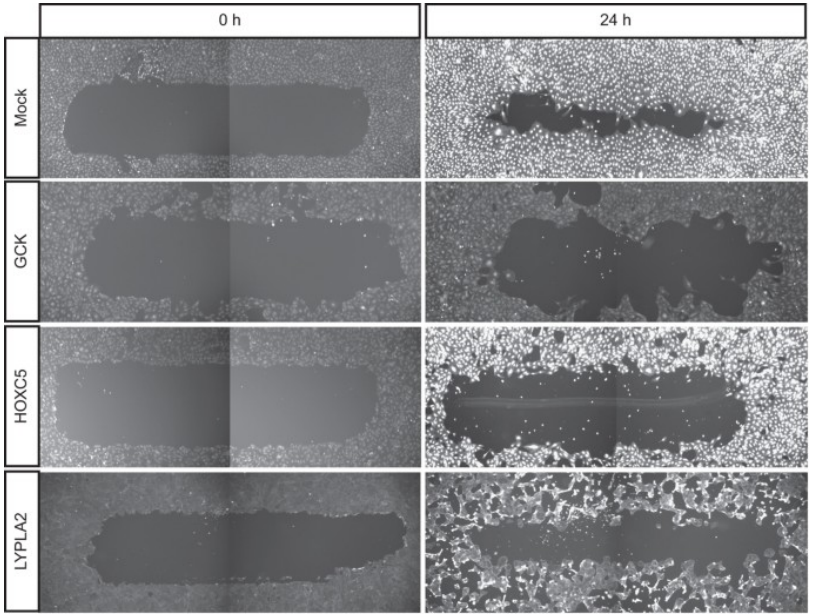
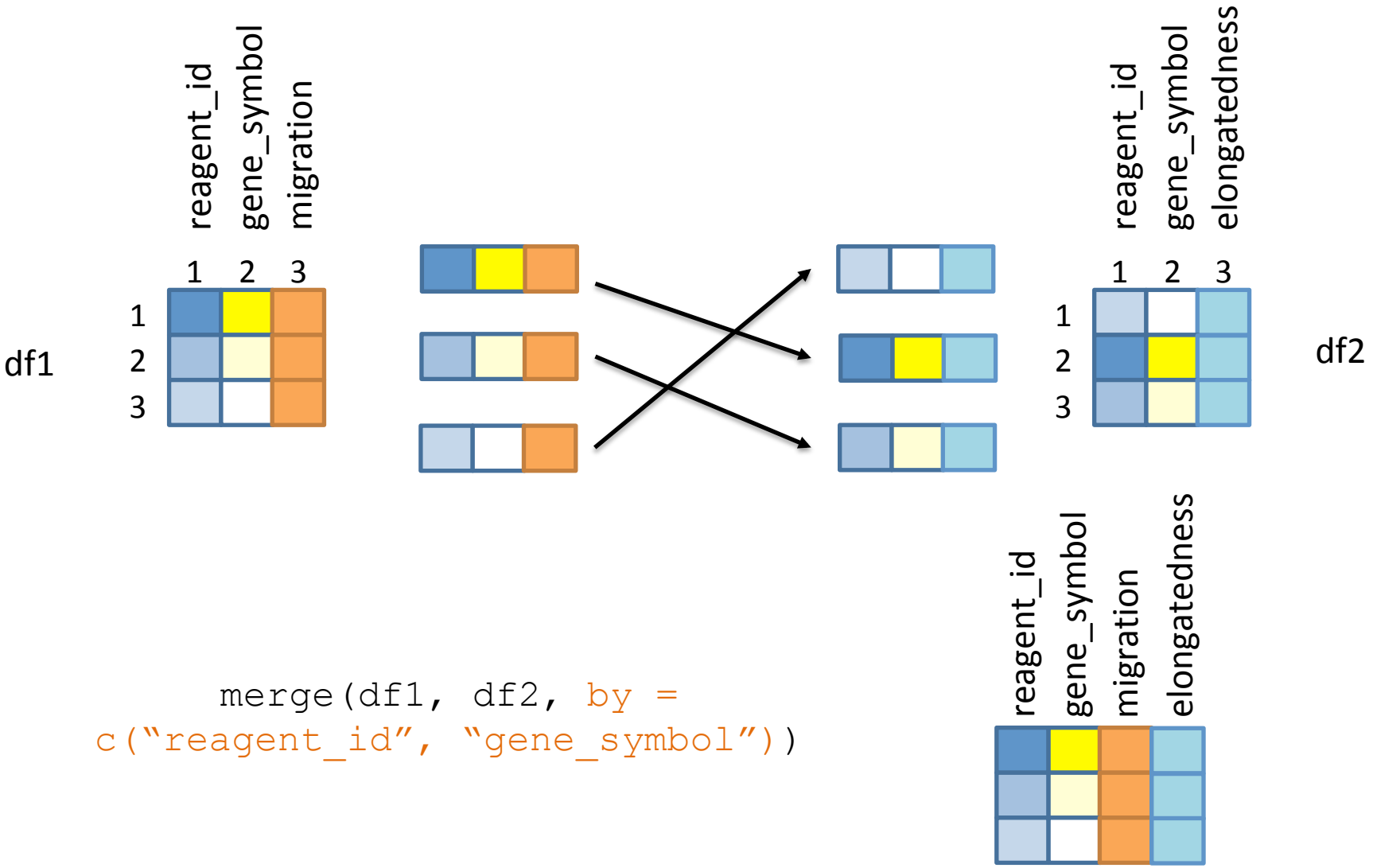


Table 1: Migration data
- col1: reagent_id
- col2: gene_symbol
- col3: migration

Table 2: Morphology data
- col1: reagent_id
- col2: gene_symbol
- col3: elongatedness

Merging data.frames



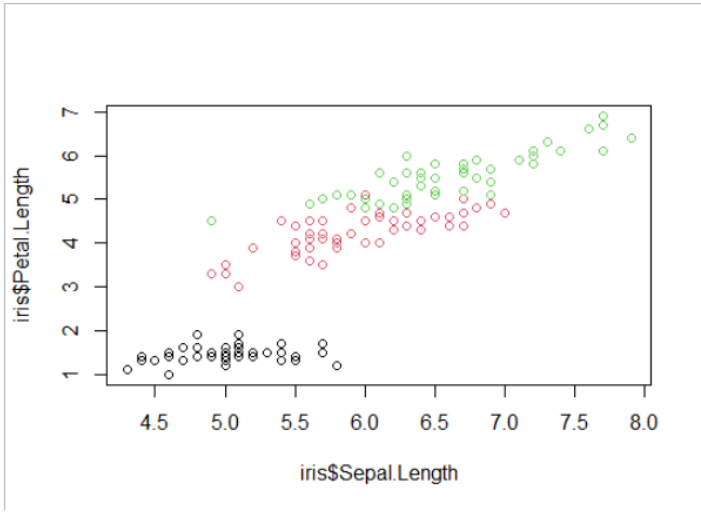

```
merge(df1, df2, by =
c("reagent_id", "gene_symbol"))
```

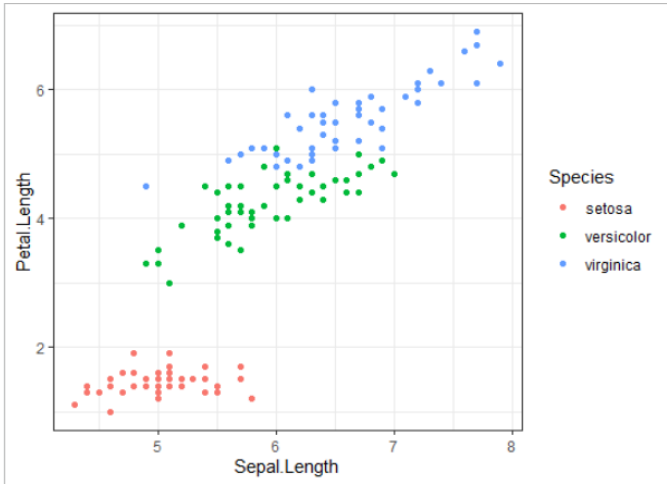
Data exploration and cleaning

- Look at the data
  - `head(), tail(), class(), str(), View()`
- Are the data types correct? If not, convert to appropriate type
  - `as.numeric(), as.character(), as.logical()`
- Is there any missing data? NA or NaN are missing data
  - `na.omit(), complete.cases()`
- Is there unnecessary data? Rows or columns you don't need?
  - Subset the data
  - Filter the data
- Visualise the data with graphs

Visualising data

- Base R plotting
  - Great for initial exploration
  - Quick
  - Simple syntax – many things can be plotted with just `plot()`
  - Not easy to modify colours, aesthetics, etc.
- ggplot2 package
  - Modular – graphs can be built element-by-element
  - Need to remember a few different functions
  - Possible to produce publication quality figures with relative ease





Let’s explore practically

Filtering data

- Keep/remove data that satisfies one or more conditions

```
> my_vec = c(1, 2, 3, 4, 5, 6)
> my_vec < 4
TRUE TRUE TRUE FALSE FALSE FALSE
> my_vec[my_vec < 4]
1 2 3
```

- For more than one condition, we need to use logical operators
  - & (AND)
  - | (OR)

Logical operators

- TRUE & TRUE # evaluates to TRUE
- TRUE & FALSE # evaluates to FALSE
- FALSE & FALSE # evaluates to FALSE
- TRUE | FALSE # evaluates to TRUE

```
> my_vec = c(1, 2, 3, 4, 5, 6)
> my_vec < 4 & my_vec > 2
FALSE FALSE TRUE FALSE FALSE FALSE
> my_vec[my_vec < 4 & my_vec > 2]
3
```

Let’s explore practically

Loop functions

- `lapply()` – perform an action on each element of a vector: returns list
- `sapply()` – as above, returns a simplified object (variable)
- `apply()` – loop over rows or columns of a matrix or df
- `tapply()` – loop over a vector, split based on a factor
- `mapply()` – loop over more than one vector

```
> my_list = list(a = c(1, 2, 3), b = c(4, 5, 6), c = c(7, 8, 9))
> lapply(my_list, mean)
$a
[1] 2

$b
[1] 5

$c
[1] 8
```

Useful reference

- R Programming for Data Science (Roger Peng) – Ch. 3, 4, 5, 9
- Swirl – Interactive learning