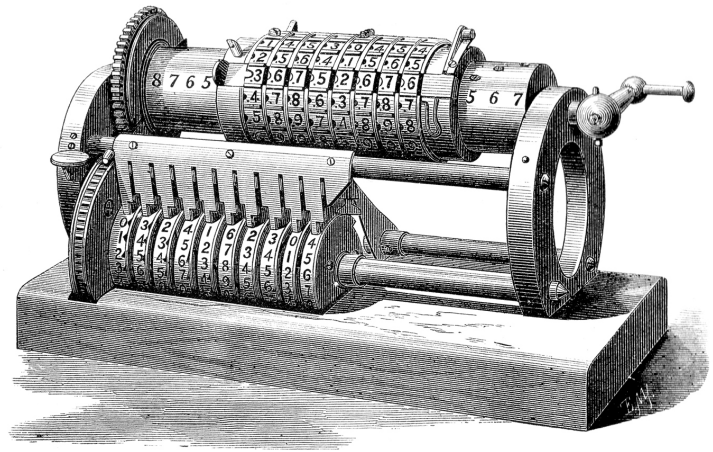
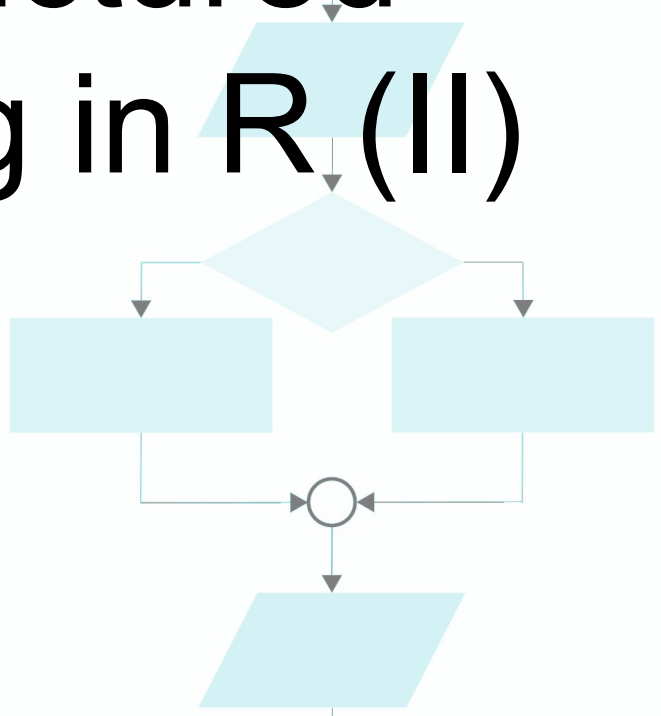
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Programming in R



Unit 2: Structured Programming in R (II)



Structured Programming in R

Remember, this course has multiple goals:

- Learn things about the R language: "R"
- Get to know nice tools to use: "Tools"
- Learn things about software development in general: "Dev"

This unit:

- "R" Track: More Data Types
- "Dev" Track: Programming Style

R Track
More Data Types

Reminder: Data Types

- "atomic" data types: `NULL`, `logical`, `numeric`, `character`
  - (also less importantly: `integer`, `complex`, `raw`)
- "recursive" data types: `list` (and some other more special things)
- many other types are just these with extra attributes!
  - `factor`, `ordered`: `integer` with "`levels`" attribute and class `"factor"` or class `c("ordered", "factor")`
  - `matrix`: `numeric` with "`dim`" attribute
  - `data.frame`: named `list` with "`row.names`" attribute and class `"data.frame"`
  - These have their own ways of handling access with `[ ]`, `[, ]`, `[[ ]]`, and `[[, ]]`

Reminder: Matrix / data.frame access with [,]

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> x[c(1, 3), ]  # select rows 1 and 3
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    3    6    9
> x[, c(1, 3)]  # select columns 1 and 3
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
```

Reminder: Matrix / data.frame access with [,]

```
> x <- matrix(1:9, nrow = 3)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> x[c(1, 3), c(1, 3)]
     [,1] [,2]
[1,]    1    7
[2,]    3    9
```

--> Create a "slice" of columns 1 and 3, and of these slice rows 1 and 3

Reminder: Matrix / data.frame access with [,]

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> y
     [,1] [,2]
[1,]    1    1
[2,]    1    3
[3,]    3    3
> x[y]  # select elements (1,1), (1,3), (3,3) in turn
[1] 1 7 9
```

Reminder: A matrix is just a vector

```
> x <- diag(3)
> x
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> x[4] <- 100
> x[8] <- 200
> x
     [,1] [,2] [,3]
[1,]    1  100    0
[2,]    0    1  200
[3,]    0    0    1
```

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4
```

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4

> attributes(x)$dim <- NULL
> attributes(x)
NULL
> x
[1] 1 2 3 4 5 6 7 8
```

Deleting the "dim" attribute turns a matrix into a vector.

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4

> attributes(x)$dim <- NULL
> attributes(x)
NULL
> x
[1] 1 2 3 4 5 6 7 8
> attributes(x)$dim <- c(4, 2)
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Adding the "dim" attribute turns a vector into a matrix!

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4

> attributes(x)$dim <- NULL
> attributes(x)
NULL
> x
[1] 1 2 3 4 5 6 7 8
> attributes(x)$dim <- c(4, 2)
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Adding the "dim" attribute turns a vector into a matrix!

could also use dim(x) here btw.

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

Most things that you encounter in R are just **atomic vectors**, **lists**, or **functions**, possibly disguised as something else by special attributes.

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4

> attributes(x)$dim <- NULL
> attributes(x)
NULL
> x
[1] 1 2 3 4 5 6 7 8
> attributes(x)$dim <- c(4, 2)
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Behind the Scenes: Attributes

- Almost everything in R can have have "attributes": attached objects that carry additional information
- Some "special" attributes tell R how to handle an object, e.g.:
  - "class": S3-class (we will see this later in the course)
  - "names": names of a named list / vector
  - "dim": dimension of a matrix

```
> x <- matrix(1:8, nrow = 2)
> attributes(x)
$dim
[1] 2 4

> attributes(x)$dim <- NULL
> attributes(x)
NULL
> x
[1] 1 2 3 4 5 6 7 8
> attributes(x)$dim <- c(4, 2)
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

Most things that you encounter in R are just **atomic vectors**, **lists**, or **functions**, possibly disguised as something else by special attributes.

Things that are something else include: NULL, environments, 'language' objects (e.g. formula)

Reminder: A data.frame is a list of vectors

```
> df <- data.frame(x = 1:3, y = letters[1:3], stringsAsFactors = FALSE)
> df
  x y
1 1 a
2 2 b
3 3 c
> df[[2]][1] <- "Z"
> df[[1]][2] <- 9
> df
  x y
1 1 Z
2 9 b
3 3 c
```

Reminder: A data.frame is a list of vectors

```
> df <- data.frame(x = 1:3, y = letters[1:3], stringsAsFactors = FALSE)
> df
  x y
1 1 a
2 2 b
3 3 c
> df[[2]][1] <- "Z"
> df[[1]][2] <- 9
> df
  x y
1 1 Z
2 9 b
3 3 c
```

```
> attributes(df)
$names
[1] "x" "y"

$row.names
[1] 1 2 3

$class
[1] "data.frame"
```

Reminder: A data.frame is a list of vectors

```
> df <- data.frame(x = 1:3, y = letters[1:3], stringsAsFactors = FALSE)
> df
  x y
1 1 a
2 2 b
3 3 c
> df[[2]][1] <- "Z"
> df[[1]][2] <- 9
> df
  x y
1 1 Z
2 9 b
3 3 c
```

```
> attributes(df)
$names
[1] "x" "y"

$row.names
[1] 1 2 3

$class
[1] "data.frame"
```

```
> class(df) <- NULL
> df
$x
[1] 1 9 3

$y
[1] "Z" "b" "c"

attr(,"row.names")
[1] 1 2 3
> class(df) <- "data.frame"
> df
  x y
1 1 Z
2 9 b
3 3 c
```

Without its 'class', it is just a list of vectors!

Reminder: Subset Assignment

- `[ ]` and `[, ]`, make it possible to assign multiple places in a vector, **matrix, data.frame** at once

```
> x <- diag(3)
> x
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> x[c(1, 3), c(1, 3)] <- c(100, 200, 300, 400)
> x
     [,1] [,2] [,3]
[1,]  100    0  300
[2,]    0    1    0
[3,]  200    0  400
```

Assign to the "slice" of rows 1, 3 and columns 1, 3

Reminder: Subset Assignment

- `[ ]` and `[, ]`, make it possible to assign multiple places in a vector at once:

```
> x
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> y
     [,1] [,2]
[1,]    1    1
[2,]    1    3
[3,]    3    3
> x[y] <- c(100, 200, 300)
> x
     [,1] [,2] [,3]
[1,]  100    0  200
[2,]    0    1    0
[3,]    0    0  300
```

Indexing with this matrix: index refers to (row 1, col 1), (row 1, col 3), and (row 3, col 3).

Reminder: which()

- Use which() to get positions of TRUE in logical vector:

```
> which(c(FALSE, TRUE, FALSE, TRUE))
[1] 2 4
```

- You can get the positions of TRUE from expressions that generate logicals:

```
> x <- c(1, 3, 2, 4)
> which(x %% 2 == 0)
[1] 3 4
```

- which() on a matrix treats it like a vector:

```
> mx <- matrix(x, nrow = 2)
> mx
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> which(mx %% 2 == 0)
[1] 3 4
```

- ... unless you use 'arr.ind = TRUE':

```
> which(mx %% 2 == 0, arr.ind = TRUE)
     row col
[1,]   1   2
[2,]   2   2
```

What We Expect You to Know

Structured Programming II

- Matrices and data.frames are internally vectors and lists with special attributes
- Know different ways of accessing and setting elements in matrices, and data.frames
- Matrix creation and manipulation: `rbind, cbind, diag, expand.grid`
- Matrix / Dim info: `dim, colnames, rownames, dimnames, nrow, ncol, NROW, NCOL, length, names`