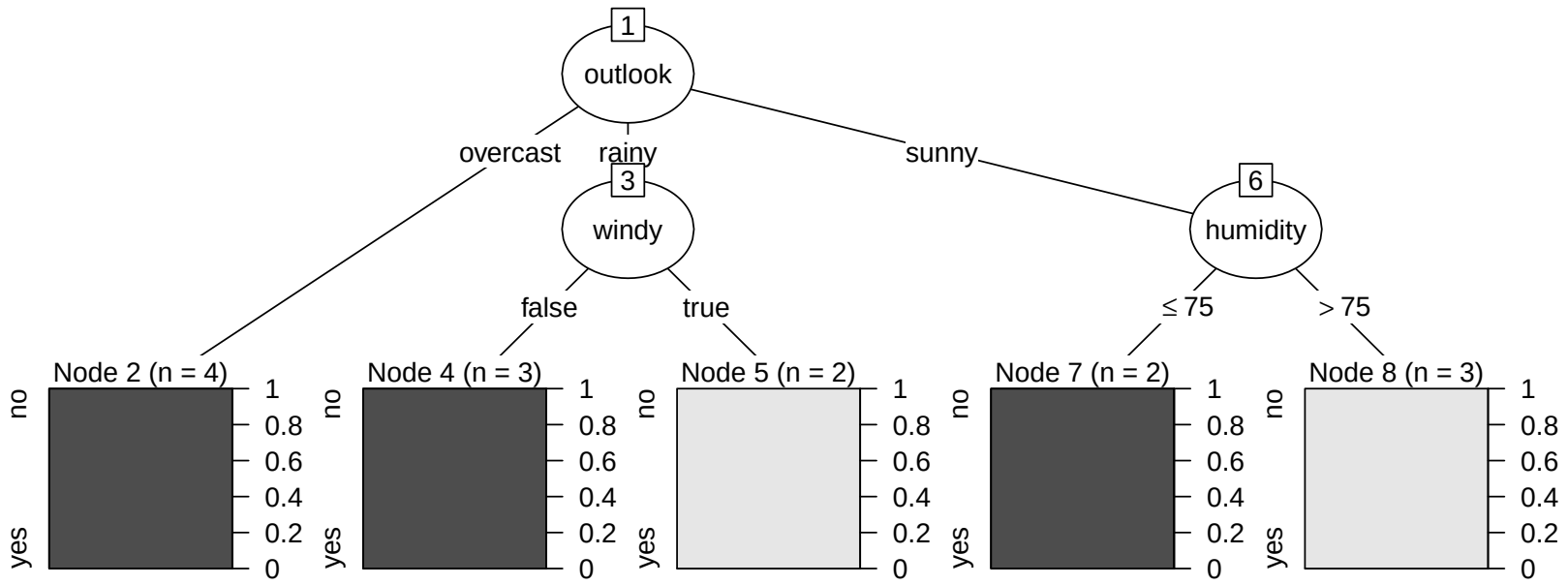
Introduction to R programming language

Karel Vaculík

Tuesday, February 17, 2015

Motivation: R example

```
library(RWeka)
library(partykit)
weather <- read.csv("weather.csv")
model <- J48(play ~ ., data = weather)
plot(model)
```



Introduction

What is R

- free software environment and programming language primarily designed for statistical computing and graphics
- object-oriented, multi-paradigm, interpreted language (console, scripts)
- can be downloaded from `http://www.r-project.org/`
- RStudio IDE: `http://www.rstudio.com/` (recommended!)

R is useful for

- easy manipulation with files
- various operations on vectors, matrices and data frames
- statistical computing / data analysis / machine learning / data mining
- graphical output

Introduction

Some other advantages

- multi-platform
- allows integration of C, C++, ... code for higher performance
- a lot of packages available (CRAN, Bioconductor, ...)
- available in many tools (SAS, SPSS, Matlab, RapidMiner, ...)

Some drawbacks

- by default, objects must be stored in physical memory (however, there are some solutions)
- authors of packages are responsible for the state of the packages
- not ideal for all purposes

R basics

- prompt >
- commands separated by a new line or ;
- help as ?<*command*> or help.search("string_to_be_found") or F1 key
- `# comments begin with a hashtag`
- `library()` for loading packages
- paths are relative to current working directory, use functions `getwd` and `setwd` to get it or change it

R basics

Variables

- no declaration
- assignment:

```
x <- 5
```

- names can contain `.` (dot), e.g.:

```
y.1 <- 10
```

Data types

R has five **atomic classes**:

- **character** (strings)
- **numeric** (real numbers)
- **integer**
- **complex**
- **logical** (possible values: `TRUE`, `T`, `FALSE`, `F`)

```
x1 <- "Hello everyone!"
x2 <- 5.4
x3 <- 20L
x4 <- 1 + 1i
x5 <- TRUE
```

Data types

Vectors

- basic data type
- all elements of the same atomic class
- creation by enumeration:

```
x <- 5
y <- c(1,1,2,3,5,8)
p <- c("one", "two", "three")
r <- c(T, F, TRUE, FALSE)
r
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

The first element has index 1!!!
In output: [i] means that the following element has index i

Data types

- creation by range specification

```
z1 <- 1:10
z1 <- seq(1, 10, 1)  # from, to, by
z1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
z2 <- 5:-5
z2 <- seq(5, -5, -1)
z2
```

```
##  [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

- length of a vector:

```
length(z2)
```

```
## [1] 11
```

Data types

- creation of an empty vector:

```
a <- numeric(0)
length(a)
```

```
## [1] 0
```

```
b <- numeric(5)
b
```

```
## [1] 0 0 0 0 0
```

- similarly by functions `character`, `logical`, etc.

Data types

Lists

- can contain elements of different types, example:

```
list1 <- list("a", FALSE, 4)
list2 <- list(1, list1)  # nested list
list2
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [[2]][[1]]
## [1] "a"
##
## [[2]][[2]]
## [1] FALSE
##
## [[2]][[3]]
## [1] 4
```

Data types

Matrices

- all elements of the same atomic class
- function `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`

```
m1 <- matrix(1:8, nrow = 2, ncol = 4)
m1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

```
m2 <- matrix(0, nrow = 2, ncol = 2)
m2
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

Data types

Data frames

- each column contains elements of only one class
- function `data.frame()`

```
first = 1:3
second = c("a", "b", "cd")
d <- data.frame(first, second)
d
```

```
##   first second
## 1     1      a
## 2     2      b
## 3     3     cd
```

Data types

Factors

- for categorical (nominal) data
- internally represented by numbers
- used by many functions by default, e.g. `read.csv()`, `data.frame()`

```
char.vector <- c("male", "female", "female", "male")
f <- factor(char.vector)
f
```

```
## [1] male   female female male
## Levels: female male
```

```
levels(f)
```

```
## [1] "female" "male"
```

Operators

- mostly element-wise (help: `?Arithmetic`):

```
1 + 1
```

```
## [1] 2
```

```
c(1,1,2) - c(3,5,8)
```

```
## [1] -2 -4 -6
```

```
c(1,1,2) * 3
```

```
## [1] 3 3 6
```

- character concatenation:

```
paste(c("You", "are", "welcome!"), collapse = " ")
```

```
## [1] "You are welcome!"
```

Operators

- dot product:

```
c(2,3,5) %*% c(1,2,3)
```

```
##      [,1]
## [1,]   23
```

- matrix multiplication:

```
a <- matrix(1:4, 2, 2); a
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
a %*% a
```

```
##      [,1] [,2]
## [1,]    7   15
## [2,]   10   22
```

Operators

- relational and logical operators:

```
y <- c(1,1,2,3,5,8)
y > 1
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

```
!(y > 1)
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
y > 1 & y < 8    # AND
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE
```

```
y == 2 | y == 5    # OR
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE FALSE
```

Operators

- matching operator:

```
y <- c(1,1,2,3,5,8)
y %in% 2:5
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE FALSE
```

Control Structures

- if-else:

```
x <- 2
if (x > 4) {
  print("YES")
} else {
  print("NO")
}
```

```
## [1] "NO"
```

Control Structures

- `for` loop:

```
for(i in c("a", "b", "c")) {
    print(i)
}
```

```
## [1] "a"
## [1] "b"
## [1] "c"
```

- useful functions: `seq`, `seq_along`, `seq_len`

Control Structures

- `while` loop:

```
x <- 1
while(x < 5) {
    print(x)
    x <- x+1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

Control Structures

- repeat loop:

```
x <- 1
repeat {
    if (x == 3) {
        x <- x+1
        next    # continue
    }
    if (x > 5) {
        break
    }
    print(x)
    x <- x+1
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 5
```

User-defined functions

- example:

```
myFunc <- function(x, y = 3) {
  x + y
}

myFunc(2)
```

```
## [1] 5
```

```
myFunc(2, 5)
```

```
## [1] 7
```

Data indexing

- indexing vectors

```
y <- c(1,1,2,3,5,8)
y[3]
```

```
## [1] 2
```

```
y[c(F, F, T, T, F, F)]
```

```
## [1] 2 3
```

```
y[2:4]
```

```
## [1] 1 2 3
```

```
y[-c(2,3)]
```

```
## [1] 1 3 5 8
```

Data indexing

```
y <- c(1,1,2,3,5,8)
y[y >= 2]   # returns values
```

```
## [1] 2 3 5 8
```

```
which(y >= 2)  # returns indices
```

```
## [1] 3 4 5 6
```

```
which(c(T,F,T))
```

```
## [1] 1 3
```

Data indexing

- indexing data frames

```
d <- data.frame(first = 4:6, second = c("a", "b", "c"))
d
```

```
##   first second
## 1     4      a
## 2     5      b
## 3     6      c
```

```
d[2,2]
```

```
## [1] b
## Levels: a b c
```

```
d[2, 1:2] # also: d[2, ]
```

```
##   first second
## 2     5      b
```

Data indexing

```
d <- data.frame(first = 1:3, second = c("a", "b", "c"))
d$first
```

```
## [1] 1 2 3
```

```
d[[1]]
```

```
## [1] 1 2 3
```

```
d[2, "first"]
```

```
## [1] 2
```

File reading and writing

- `read.csv`, `read.table`, `write.csv`, `write.table`

```
weather <- read.csv("weather.csv")
weather[1:3, ]
```

```
##     outlook temperature humidity windy play
## 1     sunny          85       85 false   no
## 2     sunny          80       90  true   no
## 3 overcast          83       86 false  yes
```

- `readLines` and `writeLines` functions
- `XML` package
- `xlsx` package
- `foreign` package (SAS, SPSS, Weka, Octave, . . .)

Data inspection

- inspecting data frames

```
weather <- read.csv("weather.csv")
head(weather)  # see also tail function
```

```
##    outlook temperature humidity windy play
## 1    sunny          85       85 false   no
## 2    sunny          80       90  true   no
## 3 overcast          83       86 false  yes
## 4    rainy          70       96 false  yes
## 5    rainy          68       80 false  yes
## 6    rainy          65       70  true   no
```

Data inspection

```
summary(weather)
```

```
##       outlook   temperature       humidity       windy    play
##  overcast:4   Min.   :64.00   Min.   :65.00   false:8   no :5
##  rainy   :5   1st Qu.:69.25   1st Qu.:71.25   true :6   yes:9
##  sunny   :5   Median :72.00   Median :82.50
##               Mean   :73.57   Mean   :81.64
##               3rd Qu.:78.75   3rd Qu.:90.00
##               Max.   :85.00   Max.   :96.00
```

```
str(weather)
```

```
## 'data.frame':    14 obs. of  5 variables:
##  $ outlook    : Factor w/ 3 levels "overcast","rainy",..: 3 3
##  $ temperature: int  85 80 83 70 68 65 64 72 69 75 ...
##  $ humidity   : int  85 90 86 96 80 70 65 95 70 80 ...
##  $ windy      : Factor w/ 2 levels "false","true": 1 2 1 1 1
##  $ play       : Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2
```

Data inspection

```
table(weather$outlook)
```

```
##
## overcast    rainy    sunny
##        4        5        5
```

```
table(weather$outlook, weather$play)
```

```
##
##              no yes
##    overcast   0   4
##    rainy      2   3
##    sunny      3   2
```

Data inspection

```
dim(weather)  # dimension
```

```
## [1] 14  5
```

```
nrow(weather)  # number of rows
```

```
## [1] 14
```

```
ncol(weather)  # number of columns
```

```
## [1] 5
```

Missing values

- special value `NA`; for each atomic class

```
a <- c(1,2,3,NA,5,NA)
```

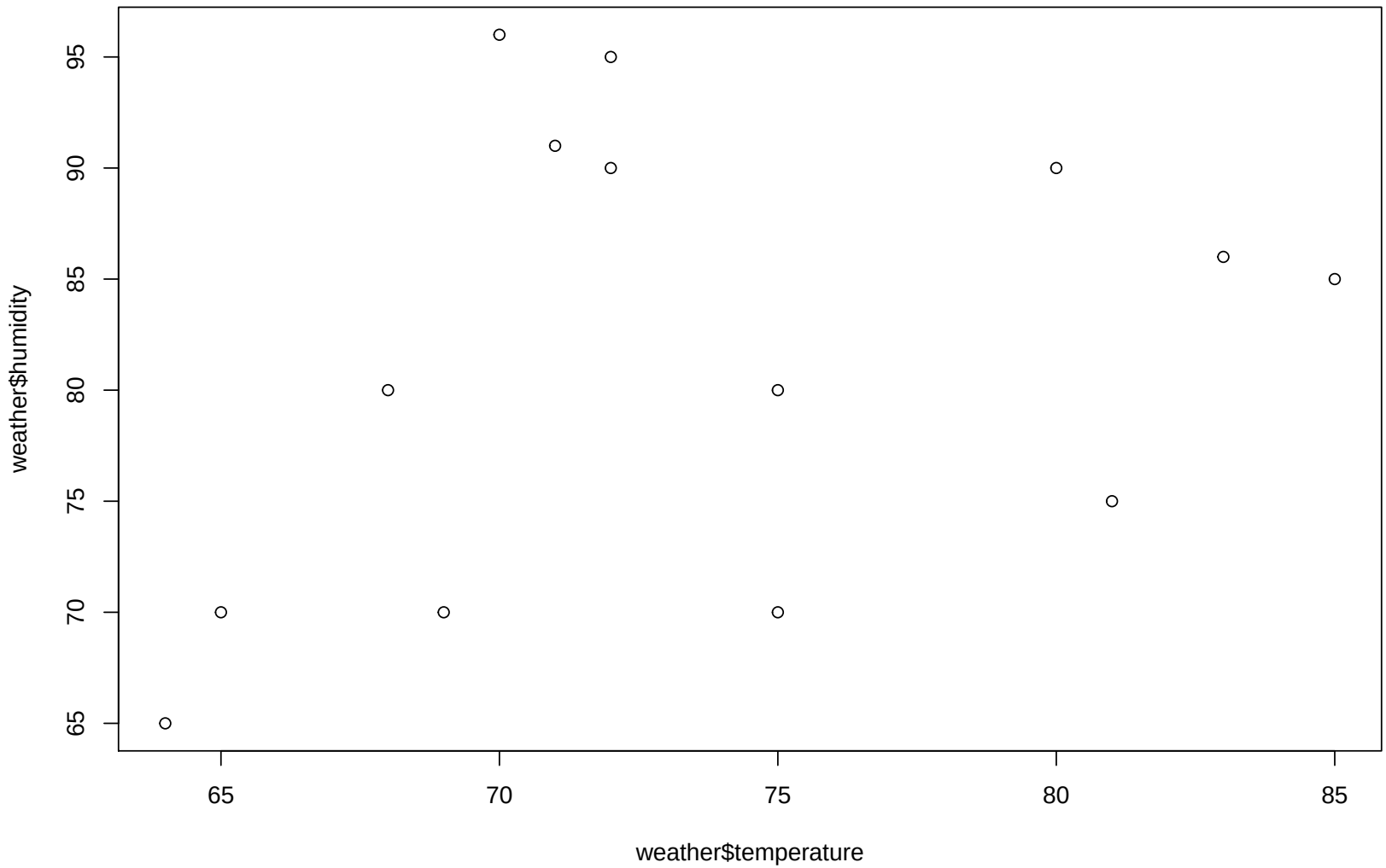
- checking for `NA` values:

```
is.na(a)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE  TRUE
```

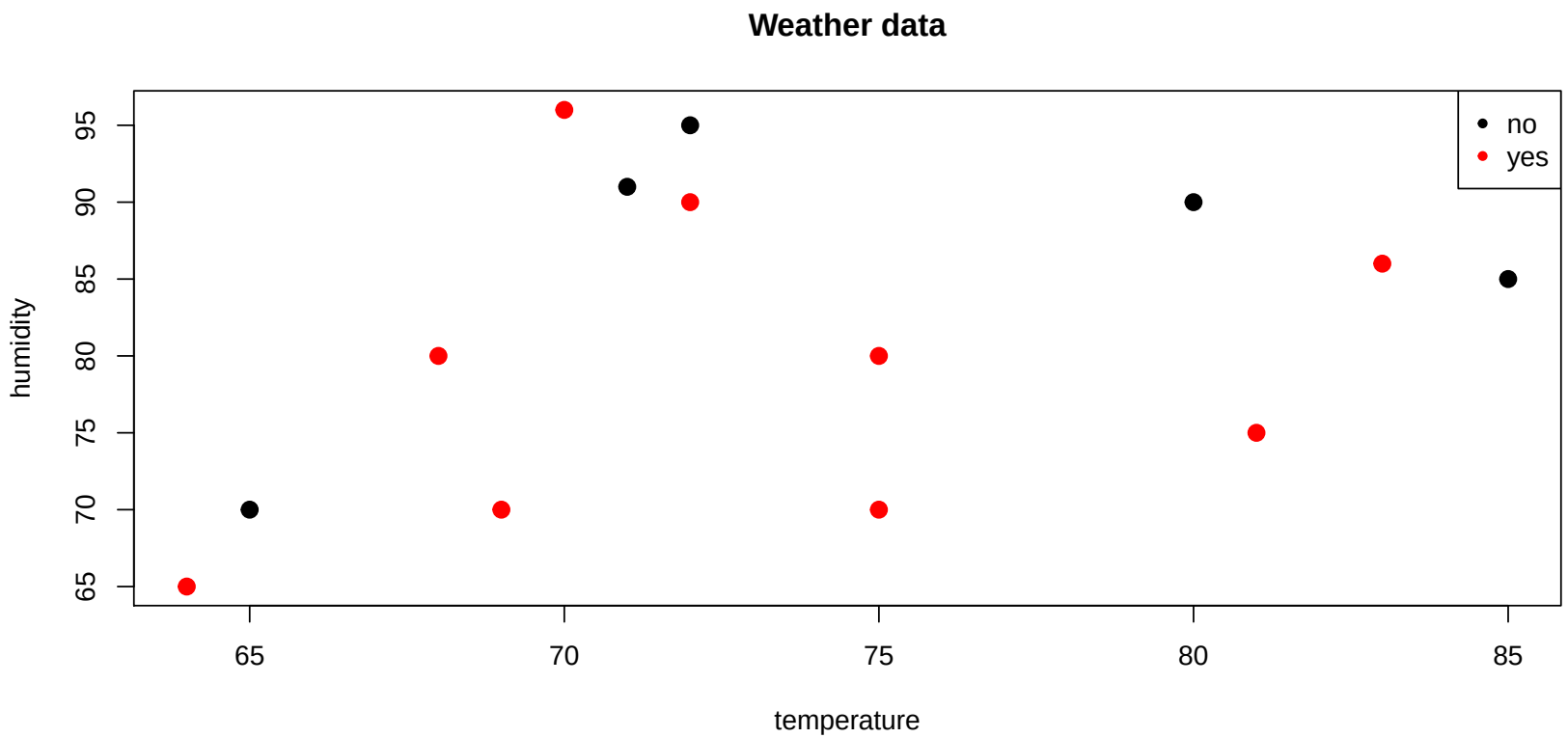
Plots

```
plot(x = weather$temperature, y = weather$humidity)
```



Plots

```
plot(x = weather$temperature, y = weather$humidity,
     col = weather$play, pch = 20, cex = 2,
     xlab = "temperature", ylab = "humidity",
     main = "Weather data")
legend("topright", levels(weather$play),
       col = 1:2, pch = 20)
```

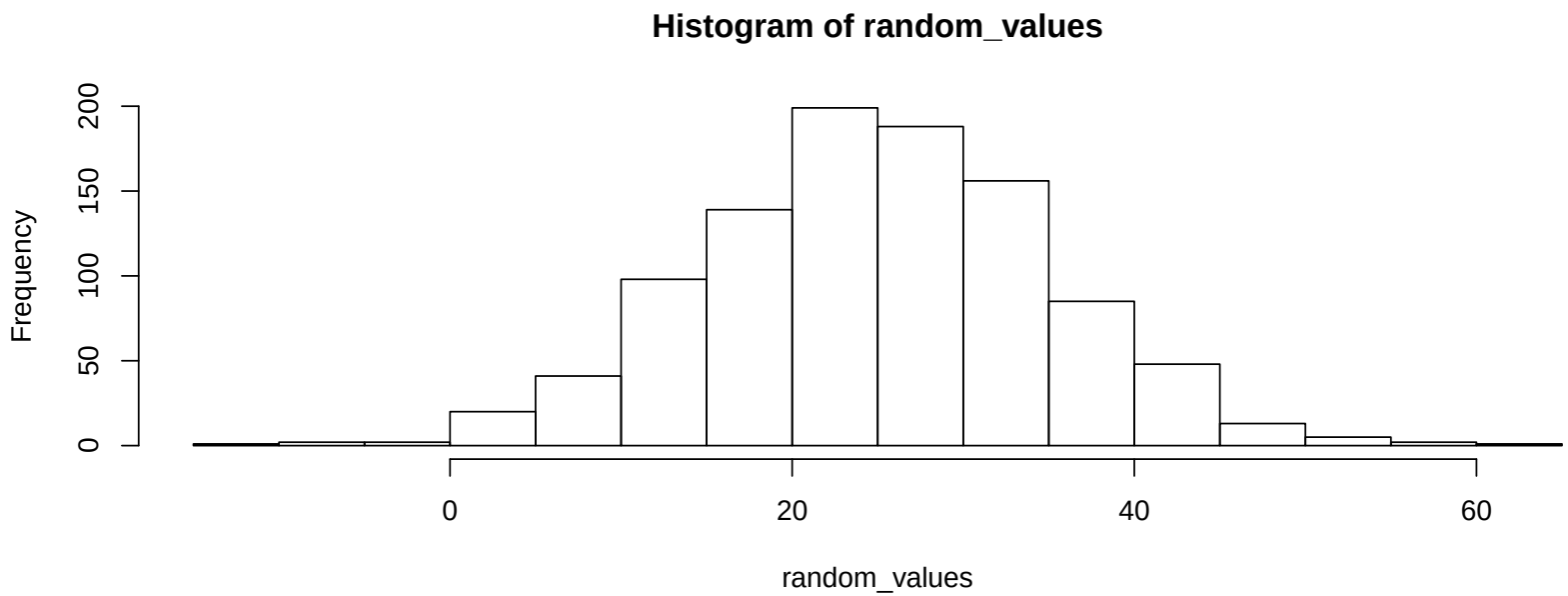


Plots

```
random_values <- rnorm(1000, mean = 25, sd = 10)
head(random_values)
```

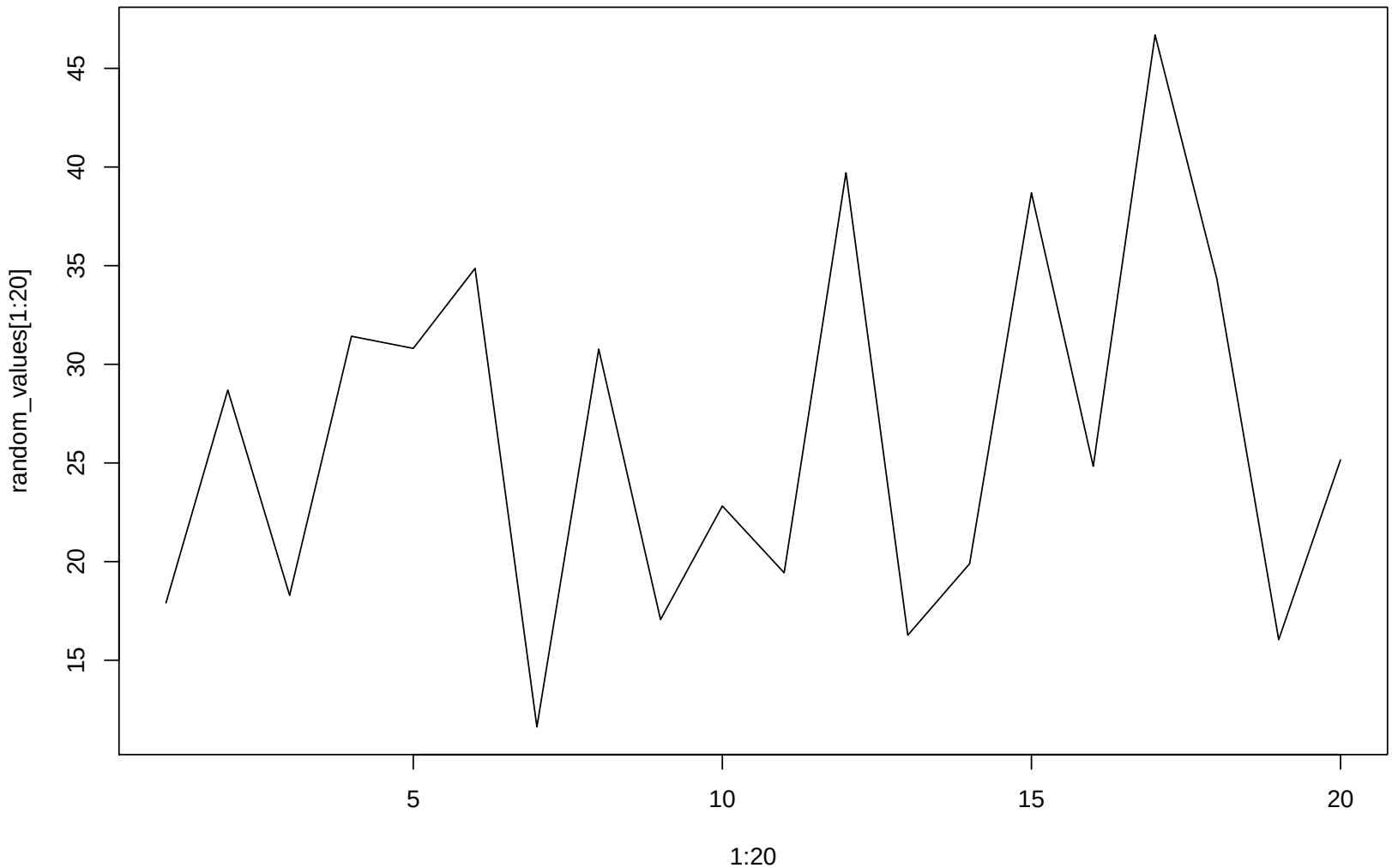
```
## [1] 17.90690 28.69438 18.28820 31.42117 30.80640 34.86196
```

```
hist(random_values)
```



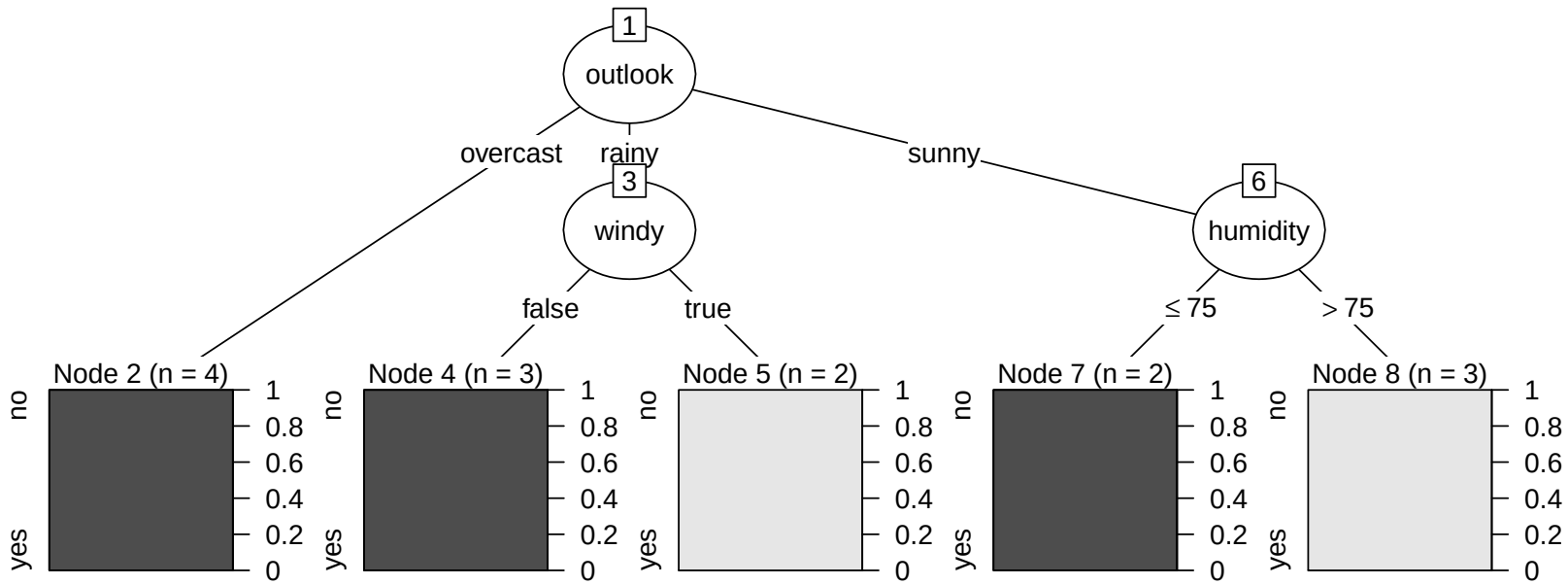
Plots

```
plot(1:20, random_values[1:20], type = "l")
```



R example - once again

```
library(RWeka)
library(partykit)
weather <- read.csv("weather.csv")
model <- J48(play ~ ., data = weather)
plot(model)
```



Other resources

R in general

- Quick-R tutorials: `http://www.statmethods.net/`
- RDocumentation: `http://www.rdocumentation.org/`

Machine learning / data mining

- Wikibook Data Mining Algorithms In R: `https://en.wikibooks.org/wiki/Category:Data_Mining_Algorithms_In_R`
- RDataMining: `http://www.rdatamining.com/`