

Advanced R Programming - Lecture 4

Leif Jonsson

Linköping University

leif.jonsson@ericsson.com

leif.r.jonsson@liu.se

September 1, 2016

Today

Linear algebra using R

Dynamic reporting with knitr and R-markdown

ggplot2

Object orientation

Questions since last time?

Big Bang Theory!

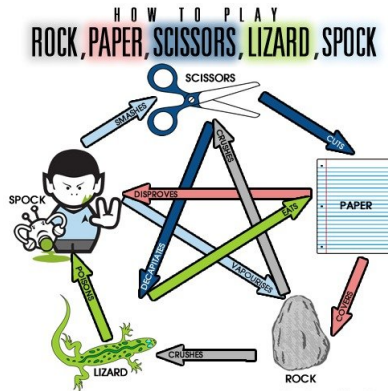


Figure: Rock-paper-scissors according to Sheldon!

sheldon_game

```
sheldon_game <- function(player1, player2){  
  alt <- c("rock", "lizard", "spock", "scissors",  
  stopifnot(player1 %in% alt, player2 %in% alt)  
  alt1 <- which(alt %in% player1)  
  alt2 <- which(alt %in% player2)  
  
  if(any((alt1 + c(1,3)) %% 5 == alt2)) {  
    return("Player_1_wins!")  
  } else {  
    return("Player_2_wins!")  
  }  
  return("Draw!")  
}
```

Why unit testing?

Basics in base

Uses LINPACK or LAPACK

Extra functionality : Matrix package
(extra LAPACK functionality)

Linear algebra

```
# Create matrix
```

```
A <- matrix(1:9, ncol=3)
```

```
# Block matrices
```

```
cbind(A,A)
```

```
rbind(A,A)
```

```
# Transpose
```

```
t(A)
```

```
# Addition and subtraction
```

```
A + A
```

```
A - A
```

```
# Matrix multiplication
```

```
A%*%A
```

Linear algebra

```
# Eigenvalues  
eigen(A)
```

```
# Determinants  
det(A)
```

```
# Matrix factorization  
svd(A)  
qr(A)
```

```
# Cholesky decomposition  
chol(A)
```


Donald E. Knuth, Literate Programming, 1984

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to humans what we want the computer to do.

- Donald E. Knuth, Literate Programming, 1984

Background

Reproducible research

Literate programming

Dynamic (repeated) reports

(Tutorials)

markdown



simple markup language

alternative to HTML (and LaTeX)

developed further by R-studio
(see coursepage)

knitr + md = rmd

Add R to markdown

$\text{knitr} + \text{md} = \text{rmd}$

Add R to markdown



(a)
.rmd

Figure: Flow

$$\text{knitr} + \text{md} = \text{rmd}$$

Add R to markdown

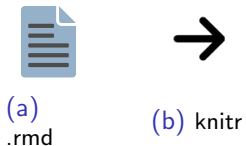


Figure: Flow

$$\text{knitr} + \text{md} = \text{rmd}$$

Add R to markdown

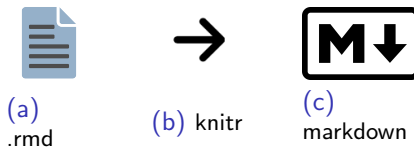


Figure: Flow

knitr + md = rmd

Add R to markdown

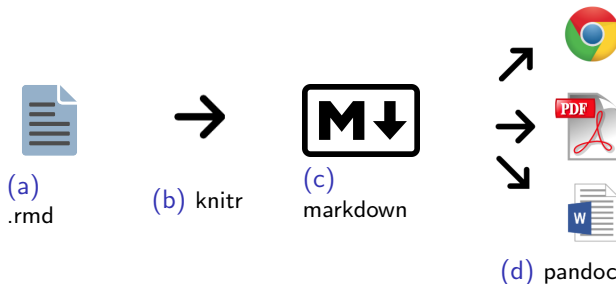


Figure: Flow

Background

popular visualization package

"The grammar of graphics"
- the language of visualization

flexible

ggplot examples

Background

Create a graph layer by layer

Store as object (print to plot)

Three (main) parts:

<code>data</code>	The data to visualize (data.frame)
<code>geom</code>	The geometric representation of data
<code>aes</code>	The mapping of colors/shape to data

geom

`geom_point`

Scatterplots

`geom_line`

Lineplots

`geom_boxplot`

Boxplot

`geom_histogram`

Histograms

`geom_bar`

Bar chart

aes

x
y
size
color
shape

Special aes

geom	Special aes
geom_point	point shape, point size
geom_line	line type, line size
geom_bar	y min, y max, fill color, outline color

GGPlot2: Example

```
library(ggplot2)

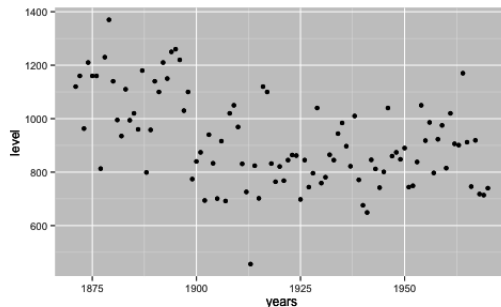
# Preprocessing
data(Nile)
Nile <- as.data.frame(Nile)
colnames(Nile) <- "level"
Nile$years <- 1871:1970
Nile$period <- "-_1900"
Nile$period[Nile$years >= 1900] <- "1900_-_1945"
Nile$period[Nile$years > 1945] <- "1945_+_ "
Nile$period <- as.factor(Nile$period)
```

GGPlot2: geom_point

```
pl <-
```

```
  ggplot(data=Nile) +  
    aes(x=years, y=level) +  
    geom_point()
```

```
pl
```

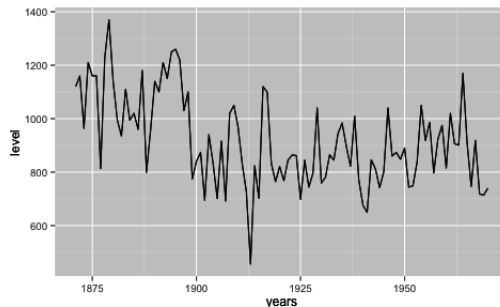


GGPlot2: geom_line

```
pl <-
```

```
  ggplot(data=Nile) +  
    aes(x=years, y=level) +  
    geom_line()
```

```
pl
```

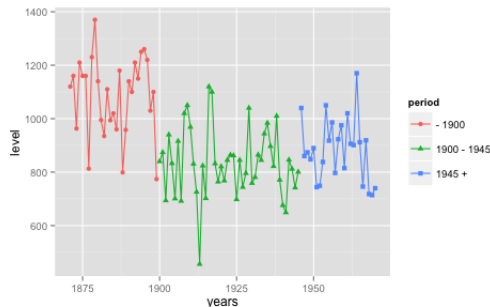


GGPlot2: geom_line + geom_line + colors!

```
pl <-
```

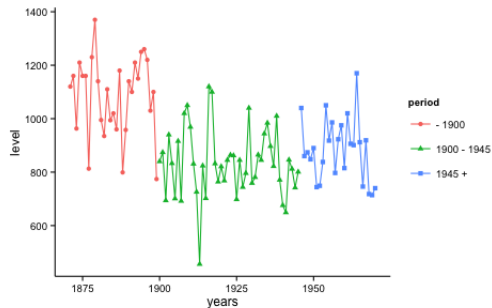
```
  ggplot(data=Nile) +  
    aes(x=years, y=level, color=period) +  
    geom_line(aes(type=period)) +  
    geom_point(aes(shape=period))
```

```
pl
```



GGPlot2: use BW theme

```
pl + theme_bw()
```



Object orientation

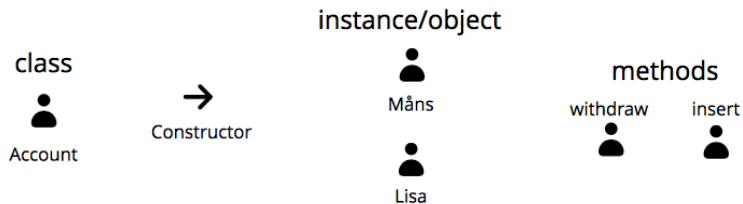
Programming paradigm

Mutable states

Key abstraction is "an object"

R is *not* purely object oriented

Object orientation



Object orientation

Fields

currency (12/24) : class variable

current_amount : object variable

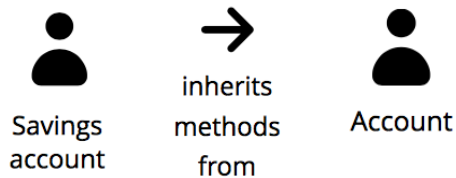
no_withdraws : object variable

Methods

insert()

withdraw()

Inheritance



Object orientation in R

S3

Simple

Methods belongs
to functions

Object orientation in R

S3	S4
Simple	More formal
Methods belongs to functions	Methods belongs to functions
	@Fields
	Parents

Object orientation in R

S3	S4	RC
Simple	More formal	Latest (R 2.12)
Methods belongs to functions	Methods belongs to functions	no copy-on-modify
	@Fields	Methods belongs to objects
	Parents	Objects have Fields and methods \$

S3

```
# Create object  
x <- 1:100  
class(x) <- "my_numeric"
```

S3

```
# Create object  
x <- 1:100  
class(x) <- "my_numeric"  
  
# Create generic function  
f <- function(x) UseMethod("f")
```

S3

```
# Create object
x <- 1:100
class(x) <- "my_numeric"

# Create generic function
f <- function(x) UseMethod("f")

# Create method
print.my_numeric <- function(x, ...){
  cat("This is my_numeric vector.")
}
```

RC

```
# Create object with fields and methods
Account <- setRefClass("Account",
  fields = list(balance = "numeric"),
  methods = list(
    withdraw = function(x) {
      balance <<- balance - x
    },
    deposit = function(x) {
      balance <<- balance + x
    }
  )
)

object$copy()
```

The End... for today.
Questions?
See you next time!