Automating the Marking of R Code

Paul Murrell

The University of Auckland

October 23 2008

Overview

Motivation:	STATS 220
Problem statement:	How to compare R objects?
What doesn't work:	`identical()` and `all.equal()`
A solution:	The **compare** package and the `compare()` function
Apologies for the convenience:	Comparing **lots** of objects for **lots** of students and turning the results into marks
What's it got to do with you?	Other applications

STATS 220 Data Technologies

- HTML (and CSS), XML (and DTDs), SQL (and databases), and **R** (and regular expressions)
- Computer lab each week (worth 0.5%)
- Lab work submitted online
- ~ 100 students

Example Lab Questions

Write R code to create the three **vectors** and the **factor** shown below, with names `id`, `age`, `edu`, and `class`.
You should end up with objects that look like this:

```
> id
[1] 1 2 3 4 5 6

> age
[1] 30 32 28 39 20 25

> edu
[1] 0 0 0 0 0 0

> class
[1] poor   poor   poor   middle
[5] middle middle
Levels: middle poor
```

Example Lab Questions

Combine the objects from the previous question together to make a **data frame** called `IndianMothers`.
You should end up with an object that looks like this:

```
> IndianMothers
  id age edu  class
1  1  30   0   poor
2  2  32   0   poor
3  3  28   0   poor
4  4  39   0 middle
5  5  20   0 middle
6  6  25   0 middle
```

Model Answer

```
# Q1

id <- seq(1, 6)  # id <- 1:6

age <- c(30, 32, 28, 39, 20, 25)

edu <- rep(0, 6)

class <- factor(rep(c("poor", "middle"), each=3))

# Q2

IndianMothers <- data.frame(id, age, edu, class)
```

Example Student Answer 1

```
id <- c(1,2,3,4,5,6)
age <- c(30,32,28,39,20,25)
edu <- c(0,0,0,0,0,0)
classname <- c("poor","middle")
classrep <- rep(classname,c(3,3))
class <- factor(classrep , level=classname)
IndianMother <- data.frame(id,age,edu,class=classrep)
```

Example Student Answer 2

```
#variable definition
id <- seq(1,6)
age <- c(30,32,28,39,20,25)
edu <- rep(0,6)
class<-factor(rep(c("poor","middle"),each=3))


#QUESTION #2 IndianMothers dataframe

IndianMothers <-data.frame(id,age,edu,class)
```

Example Student Answer 3

```
id <-c(1,2,3,4,5,6)
age <-c(30,32,28,39,20,25)
edu<-c(0,0,0,0,0,0)
class<-c("poor","poor","poor","middle","middle","middle")

IndiaMother<-data.frame(id,age,edu,class)
```

Example Student Answer 4

```
id <- c(1, 2, 3, 4, 5, 6)

age <- c(30, 32, 28, 39, 20, 25)

edu <- c(0, 0, 0, 0, 0, 0)

classNames <- c( "poor", "middle")
class <- rep( classNames, c(3, 3))
factor( class, levels= classNames)

indianMothers <- data.frame(id=id,
                            age,
                            edu,
                            class)
```

Example Student Answer 5

```
#Question 1

id = c(1:6)
age = c(30, 32,28,39,20,25)
edu = c(0,0,0,0,0,0)
class1 = c("poor","poor","poor","middle","middle","middle")
class2 = c("middle","poor")

factor(class1, levels=class2)

#Question 2
indianMother = data.frame(id = id,
                          age=age,
                          edu=edu,
                          class=class1)
```

Marking R Code

- Submission is electronic so the code can be **run**
- Easy to automate running code from all students

- *How to compare student answer to model answer?*

Comparing R Objects: identical()

Student 1:

```
> age <- c(30,32,28,39,20,25)
> age
```

```
[1] 30 32 28 39 20 25
```

Model Answer:

```
> c(30, 32, 28, 39, 20, 25)
```

```
[1] 30 32 28 39 20 25
```

Comparison:

```
> identical(age, c(30, 32, 28, 39, 20, 25))
```

```
[1] TRUE
```

Comparing R Objects: identical()

Student 1:

```
> classname <- c("poor","middle")
> classrep <- rep(classname,c(3,3))
> class <- factor(classrep , level=classname)
> class
```

```
[1] poor   poor   poor   middle middle middle
Levels: poor middle
```

Model Answer:

```
> factor(rep(c("poor", "middle"), each=3))
```

```
[1] poor   poor   poor   middle middle middle
Levels: middle poor
```

Comparison:

```
> identical(class,
            factor(rep(c("poor", "middle"), each=3)))
```

```
[1] FALSE
```

Comparing R Objects: identical()

Student 1:

```
> id <- c(1,2,3,4,5,6)
> id
```

```
[1] 1 2 3 4 5 6
```

Model Answer:

```
> seq(1, 6)
```

```
[1] 1 2 3 4 5 6
```

Comparison:

```
> identical(id, seq(1, 6))
```

```
[1] FALSE
```

Comparing R Objects: `all.equal()`

Student 1:

```
> id <- c(1,2,3,4,5,6)
> id
```

```
[1] 1 2 3 4 5 6
```

Model Answer:

```
> seq(1, 6)
```

```
[1] 1 2 3 4 5 6
```

Comparison:

```
> all.equal(id, seq(1, 6))
```

```
[1] TRUE
```

Comparing R Objects: all.equal()

Student 1:

```
> classname <- c("poor","middle")
> classrep <- rep(classname,c(3,3))
> class <- factor(classrep , level=classname)
> class
```

```
[1] poor   poor   poor   middle middle middle
Levels: poor middle
```

Model Answer:

```
> factor(rep(c("poor", "middle"), each=3))
```

```
[1] poor   poor   poor   middle middle middle
Levels: middle poor
```

Comparison:

```
> all.equal(class,
            factor(rep(c("poor", "middle"), each=3)))
```

```
[1] "Attributes: < Component 2: 2 string mismatches >"
```

Comparing R Objects: all.equal()

Student 3:

```
> class<-c("poor","poor","poor",
           "middle","middle","middle")
> class
[1] "poor"   "poor"   "poor"   "middle" "middle" "middle"
```

Model Answer:

```
> factor(rep(c("poor", "middle"), each=3))
[1] poor   poor   poor   middle middle middle
Levels: middle poor
```

Comparison:

```
> all.equal(class,
            factor(rep(c("poor", "middle"), each=3)))
[1] "Modes: character, numeric"
[2] "Attributes: < target is NULL, current is list >"
[3] "target is character, current is factor"
```

The **compare** package

```
> library(compare)
```

- Compare two objects for identity.
  If that fails ...
- Compare two objects for equality.
  If that fails ...
- Transform the objects and compare them again.
- Record all transformations along the way.

The **compare** package: compare()

Student 1:

```
> age <- c(30,32,28,39,20,25)
> age
```

```
[1] 30 32 28 39 20 25
```

Model Answer:

```
> c(30, 32, 28, 39, 20, 25)
```

```
[1] 30 32 28 39 20 25
```

Comparison:

```
> compare(age, c(30, 32, 28, 39, 20, 25))
```

```
TRUE
```

The **compare** package: compare()

Student 1:

```
> id <- c(1,2,3,4,5,6)
> id
```

```
[1] 1 2 3 4 5 6
```

Model Answer:

```
> seq(1, 6)
```

```
[1] 1 2 3 4 5 6
```

Comparison:

```
> compare(id, seq(1, 6))
```

```
TRUE
```

The **compare** package: compare()

Student 1:

```
> classname <- c("poor","middle")
> classrep <- rep(classname,c(3,3))
> class <- factor(classrep , level=classname)
> class
```

```
[1] poor   poor   poor   middle middle middle
Levels: poor middle
```

Model Answer:

```
> factor(rep(c("poor", "middle"), each=3))
```

```
[1] poor   poor   poor   middle middle middle
Levels: middle poor
```

Comparison:

```
> compare(class,
          factor(rep(c("poor", "middle"), each=3)),
          ignoreLevelOrder=TRUE)
```

```
TRUE
  reordered levels
```

The **compare** package: compare()

Student 3:

```
> class<-c("poor","poor","poor",
           "middle","middle","middle")
> class
```

```
[1] "poor"   "poor"   "poor"   "middle" "middle" "middle"
```

Model Answer:

```
> factor(rep(c("poor", "middle"), each=3))
```

```
[1] poor   poor   poor   middle middle middle
Levels: middle poor
```

Comparison:

```
> compare(class,
          factor(rep(c("poor", "middle"), each=3)),
          coerce=TRUE)
```

```
TRUE
  coerced from <factor> to <character>
```

The **compare** package: compare()

```
compare(model, comparison,
        equal = TRUE,
        coerce,
        shorten,
        ignoreOrder,
        ignoreNameCase,
        ignoreNames,
        ignoreAttrs,
        round,
        ignoreCase,
        trim,
        dropLevels,
        ignoreLevelOrder,
        ignoreDimOrder,
        ignoreColOrder,
        ignoreComponentOrder)
```

The **compare** package: `compare()`

```
compare(model, comparison, allowAll = TRUE)
```

A Pathological Example

```
> model <-
      data.frame(x=1:26,
                 y=letters,
                 z=factor(letters),
                 row.names=letters,
                 stringsAsFactors=FALSE)
> comparison <-
      data.frame(W=26:1,
                 Z=letters,
                 Y=factor(LETTERS),
                 X=1:26,
                 row.names=LETTERS,
                 stringsAsFactors=FALSE)
```

A Pathological Example

```
> head(model)
```

```
  x y z
a 1 a a
b 2 b b
c 3 c c
d 4 d d
e 5 e e
f 6 f f
```

```
> head(comparison)
```

```
   W Z Y X
A 26 a A 1
B 25 b B 2
C 24 c C 3
D 23 d D 4
E 22 e E 5
F 21 f F 6
```

A Pathological Example

```
> compare(model, comparison, allowAll=TRUE)

TRUE
  renamed
  reordered columns
  [Y] coerced from <factor> to <character>
  [Z] coerced from <character> to <factor>
  shortened comparison
  [Y] ignored case
  renamed rows
```

The **compare** package: `compareFile()`

Run an entire **file** of R code and perform comparisons on a specified set of R objects.

```
id <- c(1,2,3,4,5,6)
age <- c(30,32,28,39,20,25)
edu <- c(0,0,0,0,0,0)
classname <- c("poor","middle")
classrep <- rep(classname,c(3,3))
class <- factor(classrep , level=classname)
IndianMother <- data.frame(id,age,edu,class=classrep)
```

student1.R

```
> modelNames <- c("id", "age",
                  "edu", "class",
                  "IndianMothers")
```

The **compare** package: compareFile()

```
> compareFile(file.path("Examples", "student1.R"),
              modelNames,
              file.path("Examples", "model.R"),
              allowAll=TRUE)

$id
TRUE

$age
TRUE

$edu
TRUE

$class
TRUE
  reordered levels

$IndianMothers
FALSE
  object not found
```

The **compare** package: compareFiles()

Run **several files** of R code and perform comparisons on a specified set of R objects.

```
> files <- list.files("Examples",
                      pattern="^student[0-9]+[.]R$",
                      full.names=TRUE)
> results <-
      compareFiles(files,
                   modelNames,
                   file.path("Examples", "model.R"),
                   allowAll=TRUE,
                   resultNames=gsub("Examples.|[.]R",
                                    "", files))

> options(width=200)
> results
```

	id	age	edu	class	IndianMothers
student1	TRUE	TRUE	TRUE	TRUE reordered levels	FALSE object not found
student2	TRUE	TRUE	TRUE	TRUE	TRUE
student3	TRUE	TRUE	TRUE	TRUE coerced from <character> to <factor>	FALSE object not found
student4	TRUE	TRUE	TRUE	TRUE coerced from <character> to <factor>	TRUE renamed object
student5	TRUE	TRUE	TRUE	FALSE object not found	FALSE object not found

The **compare** package: questionMarks()

```
> q1 <-
      questionMarks(
          c("id", "age", "edu", "class"),
          maxMark=2,
          rule("id", 1),
          rule("age", 1),
          rule("edu", 1),
          rule("class", 1,
               transformRule("coerced", 1)))

> q2 <-
      questionMarks("IndianMothers",
                    maxMark=1,
                    rule("IndianMothers", 1))
```

The **compare** package: markQuestions()

```
> markQuestions(results, q1, q2)

         id-age-edu-class IndianMothers
student1                2             0
student2                2             1
student3                1             0
student4                1             1
student5                1             0
```

The **compare** package: questionComments() and commentQuestions

```
> q1comments <-
      questionComments(
          c("id", "age", "edu", "class"),
          comments(
              "class",
              transformComment(
                  "coerced",
                  "'class' is a factor!")))
> commentQuestions(results, q1comments)
```

```
         id-age-edu-class
student1 ""
student2 ""
student3 "'class' is a factor!"
student4 "'class' is a factor!"
student5 ""
```

A Data Analysis Example

```
> head(CO2)
```

```
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
```

```
> lm1CO2 <- lm(uptake ~ Treatment + conc, data=CO2)
> lm2CO2 <- lm(uptake ~ conc + Treatment, data=CO2)
```

A Data Analysis Example

```
> compare(lm1CO2, lm2CO2, allowAll=TRUE)

FALSE [TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE, FAL
  model treated as list
  [qr] model treated as list
  [call] model treated as character
  [terms] model treated as character
  [model] reordered columns
  [coefficients] sorted
  [coefficients] renamed
  [coefficients] dropped names
  [effects] sorted
  [effects] renamed
  [effects] dropped names
  [qr] [qr] sorted
  [qr] [qraux] sorted
  [call] ignored case
  [terms] ignored case
  [model] dropped attributes
```

A Data Analysis Example

```
> result <- compare(lm1CO2, lm2CO2, allowAll=TRUE)
> result$detailedResult
```

coefficients	residuals	effects	rank	fitted.values
TRUE	TRUE	FALSE	TRUE	TRUE
assign	qr	df.residual	contrasts	xlevels
TRUE	FALSE	TRUE	TRUE	TRUE
call	terms	model		
FALSE	FALSE	TRUE		

```
> lm1CO2$coefficients
```

(Intercept)	Treatmentchilled	conc
22.93005171	-6.85952381	0.01773059

```
> lm2CO2$coefficients
```

(Intercept)	conc	Treatmentchilled
22.93005171	0.01773059	-6.85952381

A Data Analysis Example

```
> result$detailedResult
```

```
 coefficients     residuals       effects          rank fitted.values
         TRUE          TRUE         FALSE          TRUE          TRUE
       assign            qr  df.residual     contrasts       xlevels
         TRUE         FALSE          TRUE          TRUE          TRUE
         call         terms         model
        FALSE         FALSE          TRUE
```

```
> qr.R(lm1CO2$qr)
```

```
  (Intercept) Treatmentchilled          conc
1   -9.165151        -4.582576 -3.986841e+03
2    0.000000         4.582576  1.813827e-14
3    0.000000         0.000000  2.695997e+03
```

```
> qr.R(lm2CO2$qr)
```

```
  (Intercept)      conc Treatmentchilled
1   -9.165151 -3986.841    -4.582576e+00
2    0.000000  2695.997     1.024571e-17
3    0.000000     0.000     4.582576e+00
```

Summary

- The `compare()` function compares two R objects for "equality", transforming the objects if necessary.
- The `compareFiles()` function runs files of R code and compares the results from each file to a set of model answers.
- The `markQuestions()` function converts the results of comparisons into a set of marks.
- The `commentQuestions()` function converts the results of comparisons into a set of comments.