Ashish Mahabal
California Institute of Technology

R - III

Sponsored by the Jet Propulsion Laboratory and by the Keck Institute for Space Studies, California Institute of Technology

Packages

R CMD INSTALL packagename

installs a package.

R CMD check packagename

runs the QA tools on the package.

R CMD build packagename

creates a package file

Packages …

R> library()

- Tells you which packages are installed

R> library(packagename)

- Loads the installed package
- Example: **library(LearnBayes)**
- Bayesian Computation with R by Jim Albert

R> search()

tells you which packages are loaded

Handling datasets

```
> data(achievement)
Warning message:
In data(achievement) : data set ‘achievement’ not found
> help(achievement)
No documentation for ‘achievement’ in specified packages and libraries:
you could try ‘??achievement’
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
> library(LearnBayes)
> search()
 [1] ".GlobalEnv"         "package:LearnBayes" "package:stats"
 [4] "package:graphics"   "package:grDevices"  "package:utils"
 [7] "package:datasets"   "package:methods"    "Autoloads"
[10] "package:base"
>
```

achievement package:LearnBayes R Documentation

School achievement data

Description:

Achievement data for a group of Austrian school children

Usage:

```
achievement
```

Format:

A data frame with 109 observations on the following 7 variables.

Gen gender of child where 0 is male and 1 is female

```
> head(achievement,7)
  Gen Age  IQ math1 math2 read1 read2
1   1 121  99    12    11    27    17
2   0 124  83    13     4    12    15
3   1 103 117     5     8    30    26
4   1 127  83     8     6    30    12
5   0 115 109     7     4    26    27
6   0 108 111     6     3    17    21
7   1 106  92     9     9    25    30
>
```

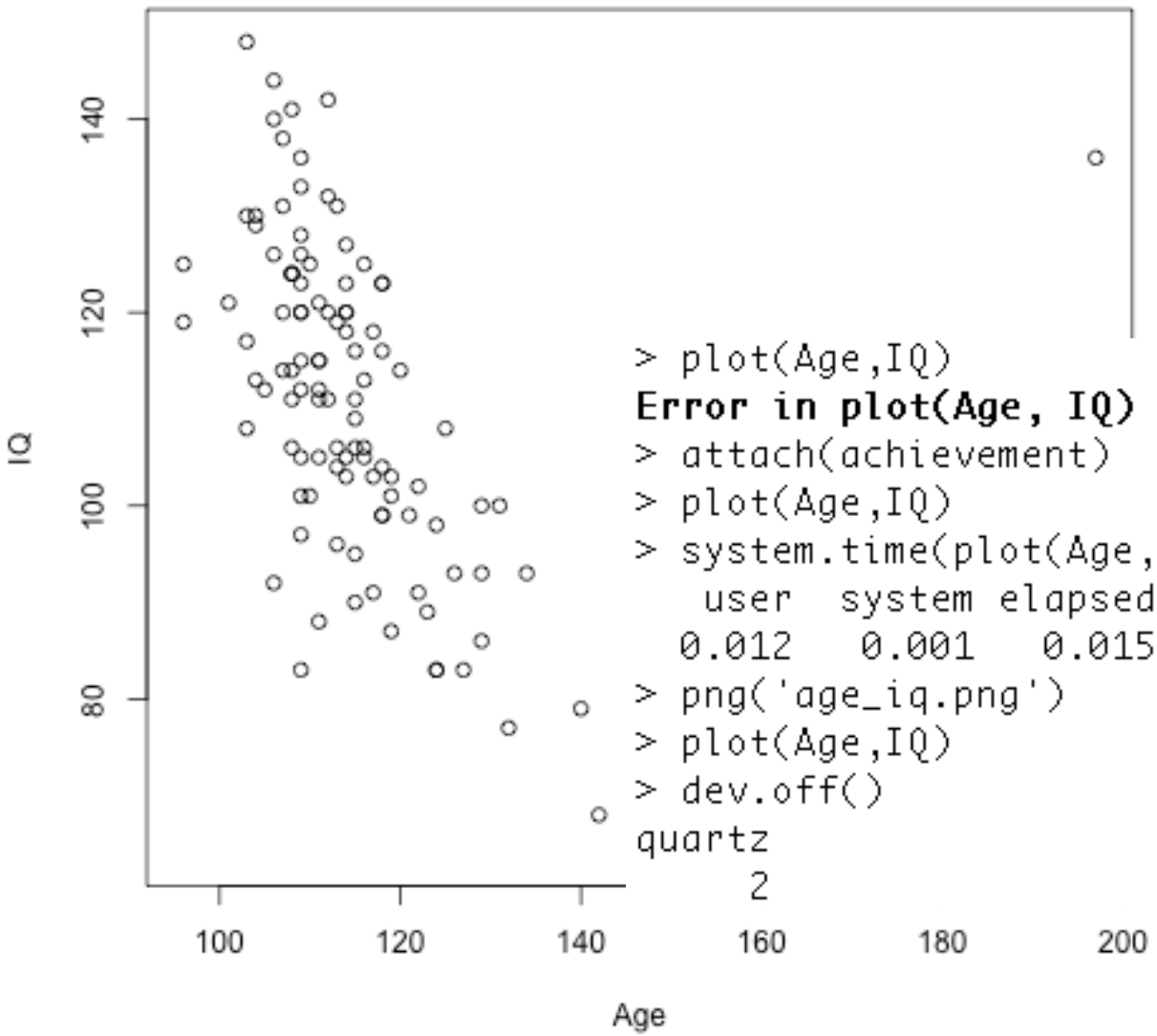
Profiling

R> proc.time() returns the current time. Save it before a task and subtract from the value after a task.

R> system.time() times the evaluation of expression

R> Rprof(filename) turns on the profiler, and **Rprof(NULL)** turns it off. The profiler writes a list of the current functions being run to filename many times per second.

R> summaryRprof(filename) summarizes this to report how much time is spent in each function.




```
> plot(Age,IQ)
Error in plot(Age, IQ) : object 'Age' not found
> attach(achievement)
> plot(Age,IQ)
> system.time(plot(Age,IQ))
   user  system elapsed
  0.012   0.001   0.015
> png('age_iq.png')
> plot(Age,IQ)
> dev.off()
quartz
     2
```

```
> Rprof('plotprof')
> system.time(plot(Age,IQ))
   user  system elapsed
  0.012   0.001   0.016
> Rprof(NULL)
> summaryRprof('plotprof')
$by.self
       self.time self.pct total.time total.pct
"gc"         0.04    66.67       0.04     66.67
"axis"       0.02    33.33       0.02     33.33

$by.total
               total.time total.pct self.time self.pct
"system.time"        0.06    100.00      0.00     0.00
"gc"                 0.04     66.67      0.04    66.67
"axis"               0.02     33.33      0.02    33.33
"Axis.default"       0.02     33.33      0.00     0.00
"Axis"               0.02     33.33      0.00     0.00
"localAxis"          0.02     33.33      0.00     0.00
"plot.default"       0.02     33.33      0.00     0.00
"plot"               0.02     33.33      0.00     0.00

$sample.interval
[1] 0.02

$sampling.time
[1] 0.06

>
```

ashishmahabal% cat ~/progs/R/plotprof
sample.interval=20000
"gc" "system.time"
"gc" "system.time"
"axis" "Axis.default" "Axis" "localAxis" "plot.default" "plot" "system.time"

Debugging

traceback() shows location of last error: what function it was in, where this was called from, and so on back to your top-level command.

options(error=dump.frames) saves the entire state of your program when an error occurs. **debugger()** then lets you start the debugger to inspect any function that was being run.

options(error=recover) starts the debugger as soon as an error occurs.

browser() starts the debugger at this point in your code.

options(warn=2) turns warnings into errors.

debug(fname) starts the debugger when function **fname()** is called.

RStudio also has elaborate debugging framework

Accessing built-in datasets

```
R> data()
R> data(AirPassengers)
R> ?AirPassengers
R> new <- edit(AirPassengers)
R> x <-array(c(AirPassengers[1:144]),dim=c(12,12))
R> pairs(x)
```

```
R> plot(cars)
R> help(cars)
R> cars
R> ?UCBAdmissions
R> plot(UCBAdmissions)
```

Basic plotting

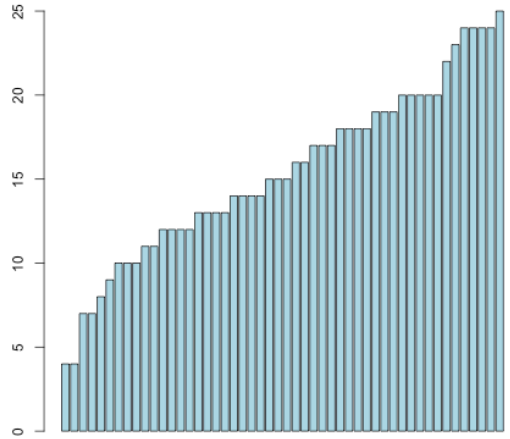
```
R> plot(x, y)    # scatterplot
R> plot(xy)          # scatterplot from 2-col matrix
R> plot(x)       # timeseries or real/img
R> plot(f)       # barplots for factors
R> plot(f, y)        # boxplots for factors
R> pairs(X)
R> hist(), dotchart(), image(), contour(), …
R> points(), text(), math, multiple, interactive
```

Basic statistics

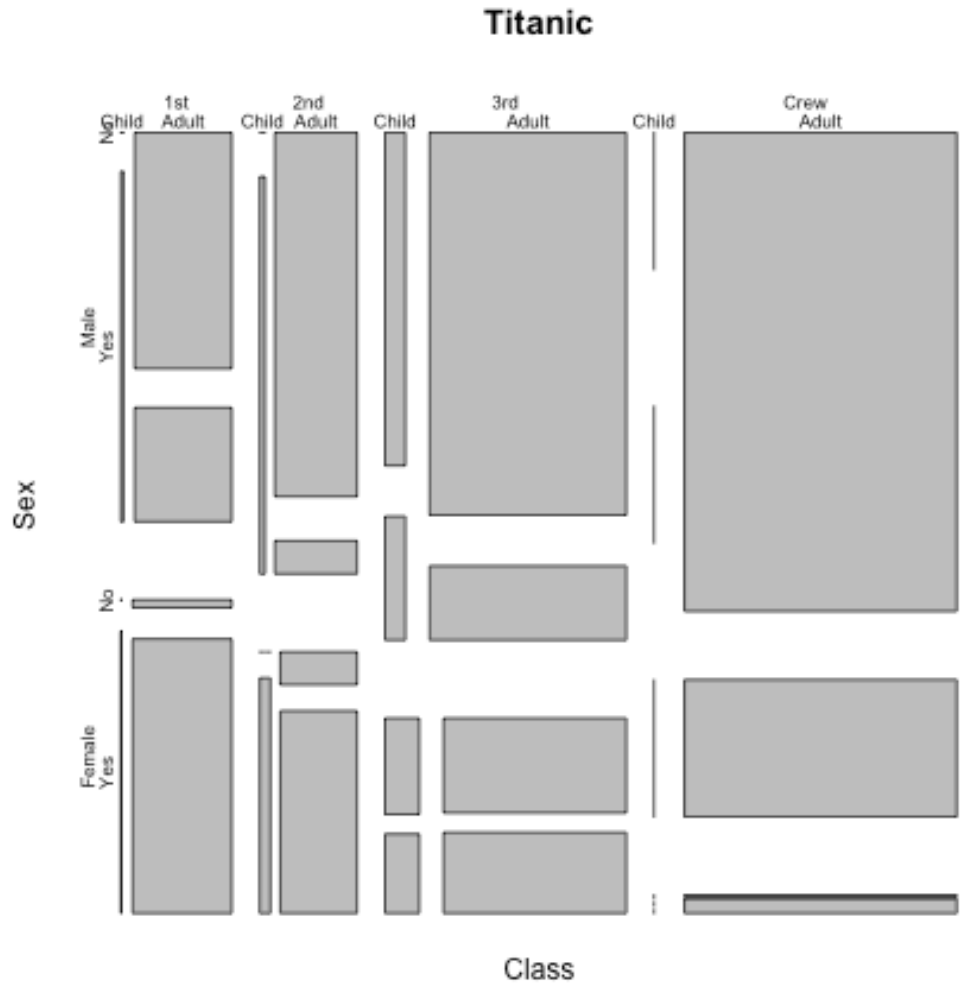
```
R> attach(cars)   # see ?cars for details

R> sum(speed)
R> mean(speed)
R> sd(speed)
R> min(speed)
R> max(speed)
R> summary(cars)
```

A few other plots

```
R> plot(speed)
R> dotchart(speed)
R> barplot(speed,dist,horiz=T, col="lavender")
R> barplot(speed,col = "lightblue")
R> pie(speed)
R> hist(speed)
R> plot(density(speed))
R> detach(cars)
```



plot(Titanic)



Next Time …

astRowRap

swirl