20080530 - math 480 - rpy

Using R from Sage via RPy

WARNING If you installed Sage <= 3.0.2 as a binary or built from source and moved the install, then rpy will not work at all. This is fixed in sage-3.0.3. See this patch. (Don't worry -- there will be no homework on Rpy.)

The R project for statistical computing

R is widely considered the "standard" for statistical computing in *scientific research*, and research papers in statistics are often connected with R code. There are also several good very easy-to-use (in some ways) commercial statistics program such as SAS, SPSS, STATA, etc. If you're going to carry out a project that involves a lot of statistics, make sure you are reasonably aware of all your options and don't reject anything (even commercial options) out of hand. R is the premier system for research on statistics, but the other tools are very smooth and high quality, and can be very very useful for "everyday work".

The following is an edited version of *What is R?* from the R website:

> R is a language and environment for statistical computing and graphics.
>
> R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.
>
> R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.
>
> The R environment R is an *integrated suite of software* facilities for data manipulation, calculation and graphical display. It includes
>
> 1. an effective data handling and storage facility,
> 2. a suite of operators for calculations on arrays, in particular

matrices,

3. a large, coherent, integrated collection of intermediate tools for data analysis,
4. graphical facilities for data analysis and display either on-screen or on hardcopy, and
5. a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a *true computer language*, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced *users can write C code* to manipulate R objects directly.

Every copy of Sages comes with R. You can use it directly from the command line by typing `sage -R`:

```
> sage -R

R version 2.6.1 (2007-11-26)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> mean(c(3,5,6))
[1] 4.666667
```

Rpy: http://rpy.sourceforge.net/

If you want to use R from within Sage, currently the most robust option is to use rpy, which is a powerful and mature way to use R from Python in library mode. From the rpy website:

> RPy is a very simple, yet robust, Python interface to the R Programming Language. It can manage all kinds of R objects and can execute arbitrary R functions (including the graphic functions). All errors from the R language are converted to Python exceptions. Any module installed for the R system can be used from within Python.
>
> This code is inspired by RSPython from the Omegahat project. The main goals of RPy are:
>
> 1. to have a very robust interface for using R from Python
> 2. the interface should be as transparent and easy to use as possible
> 3. it should be usable for real scientific and statistical computations

The Rpy Demo in Sage

In order to fully make use of R in Sage, you should learn the basics of R and read the rpy documentation. The following demo should give you some sense of how to actually use rpy in the Sage notebook.

```
# import the rpy functions
from rpy import *

faithful_data = {"eruption_duration":[],
                "waiting_time":[]}
```

```
# The data in the faithful.dat dataset contains
# (1) the duration of the eruptions along with the
# (2) waiting time between eruptions
# for the Old Faithful geyser in Yellowstone National Park.
# So the first two lines mean:
# there was an eruption for 3.6 minutes, then 79 minute wait then
# eruption for 1.8 minutes then 54 minute wait.  (I think.)
```

```
print open(DATA + 'faithful.dat').read()
```

```
WARNING: Output truncated!
full_output.txt



"eruptions" "waiting"
3.6 79
1.8 54
3.333 74
```

```
2.283 62
4.533 85
2.883 55
4.7 88
3.6 85
1.95 51
4.35 85
1.833 54
3.917 84
4.2 78
1.75 47
4.7 83
2.167 52
1.75 62
4.8 84
1.6 52
4.25 79
1.8 51
1.75 47
3.45 78
3.067 69
4.533 74
3.6 83
1.967 55
4.083 76
3.85 78
4.433 79
4.3 73
4.467 77
3.367 66
4.033 80
3.833 74
2.017 52
1.867 48
4.833 80
1.833 59
4.783 90
4.35 80
1.883 58
4.567 84
1.75 58
4.533 73
3.317 83
3.833 64
2.1 53
4.633 82
2 59
4.8 75
4.716 90
1.833 54
4.833 80
1.733 54
4.883 83
3.717 71
```

```
1.667 64

...

3.833 75
3.417 64
4.233 76
2.4 53
4.8 94
2 55
4.15 76
1.867 50
4.267 82
1.75 54
4.483 75
4 78
4.117 79
4.083 78
4.267 78
3.917 70
4.55 79
4.083 70
2.417 54
4.183 86
2.217 50
4.45 90
1.883 54
1.85 54
4.283 77
3.95 79
2.333 64
4.15 75
2.35 47
4.933 86
2.9 63
4.583 85
3.833 82
2.083 57
4.367 82
2.133 67
4.35 74
2.2 54
4.45 83
3.567 73
4.5 73
4.15 88
3.817 80
3.917 71
4.45 83
2 56
4.283 79
4.767 78
4.533 84
1.85 58
```

```
4.25 83
1.983 43
2.25 60
4.75 75
4.117 81
2.15 46
4.417 90
1.817 46
4.467 74
```

full_output.txt

```
# load the sample data file.  I've attached it to this worksheet.
# I downloaded this file from
http://rpy.sourceforge.net/faithful.dat

f = open(DATA + 'faithful.dat','r')
for row in f.readlines()[1:]: # skip the column header line
   splitrow = row[:-1].split(" ")
   faithful_data["eruption_duration"].append(float(splitrow[0]))
   faithful_data["waiting_time"].append(int(splitrow[1]))

f.close()
```

```
ed = faithful_data["eruption_duration"]
edsummary = r.summary(ed)
edsummary
```

```
{'Min.': 1.6000000000000001, '1st Qu.': 2.1629999999999998, '3rd
Qu.': 4.4539999999999997, 'Median': 4.0, 'Max.': 5.0999999999999996
'Mean': 3.488}
```

```
# Make something much nicer:

print "Summary of Old Faithful eruption duration data"
for k in edsummary.keys():
   print "%-10s: %.3f" %(k, edsummary[k])
```

```
Summary of Old Faithful eruption duration data
Min.      : 1.600
1st Qu.   : 2.163
3rd Qu.   : 4.454
Median    : 4.000
Max.      : 5.100
Mean      : 3.488
```

```
r.help('summary')
```

```
R Help on 'summary'summary                 package:base
R Documentation

Object Summaries

Description:

     'summary' is a generic function used to produce result
summaries
     of the results of various model fitting functions.  The
function
     invokes particular 'methods' which depend on the 'class' of th
     first argument.

Usage:

     summary(object, ...)

     ## Default S3 method:
     summary(object, ..., digits = max(3, getOption("digits")-3))
     ## S3 method for class 'data.frame':
     summary(object, maxsum = 7,
            digits = max(3, getOption("digits")-3), ...)

     ## S3 method for class 'factor':
     summary(object, maxsum = 100, ...)

     ## S3 method for class 'matrix':
     summary(object, ...)

Arguments:

  object: an object for which a summary is desired.

  maxsum: integer, indicating how many levels should be shown for
          'factor's.

  digits: integer, used for number formatting with 'signif()' (for
          'summary.default') or 'format()' (for
'summary.data.frame').

     ...: additional arguments affecting the summary produced.

Details:

     For 'factor's, the frequency of the first 'maxsum - 1' most
     frequent levels is shown, where the less frequent levels are
     summarized in '"(Others)"' (resulting in 'maxsum' frequencies)

     The functions 'summary.lm' and 'summary.glm' are examples of
     particular methods which summarize the results produced by 'lm
     and 'glm'.

Value:
```

```
print "Stem-and-leaf plot of Old Faithful eruption duration data"
print r.stem(ed)
```

```
Stem-and-leaf plot of Old Faithful eruption duration data

  The decimal point is 1 digit(s) to the left of the |

  16 | 007700335555555555558888
  18 |
0000000022222233333333333333555577777777777777778888888822223333557
  20 | 0000000022222233337788880000003355777788
  22 | 00000022333355557788002233557788
  24 | 0000222288
  26 | 2233
  28 | 008800
  30 | 77
  32 | 22333377
  34 | 225500007777
  36 | 00000000882233557777
  38 | 22333333333355558822222255557777
  40 |
00000000000033355777778888888888000022223333555555557777777788
  42 | 0033333355555555777788880000223333333333555555557777777788
  44 |
0022222222333355555577778800000000000000002233333333335577777788888
  46 | 00000000223333335577770000000000002233557788
  48 | 0000000000002222333355880000333333
  50 | 00337700

None
```

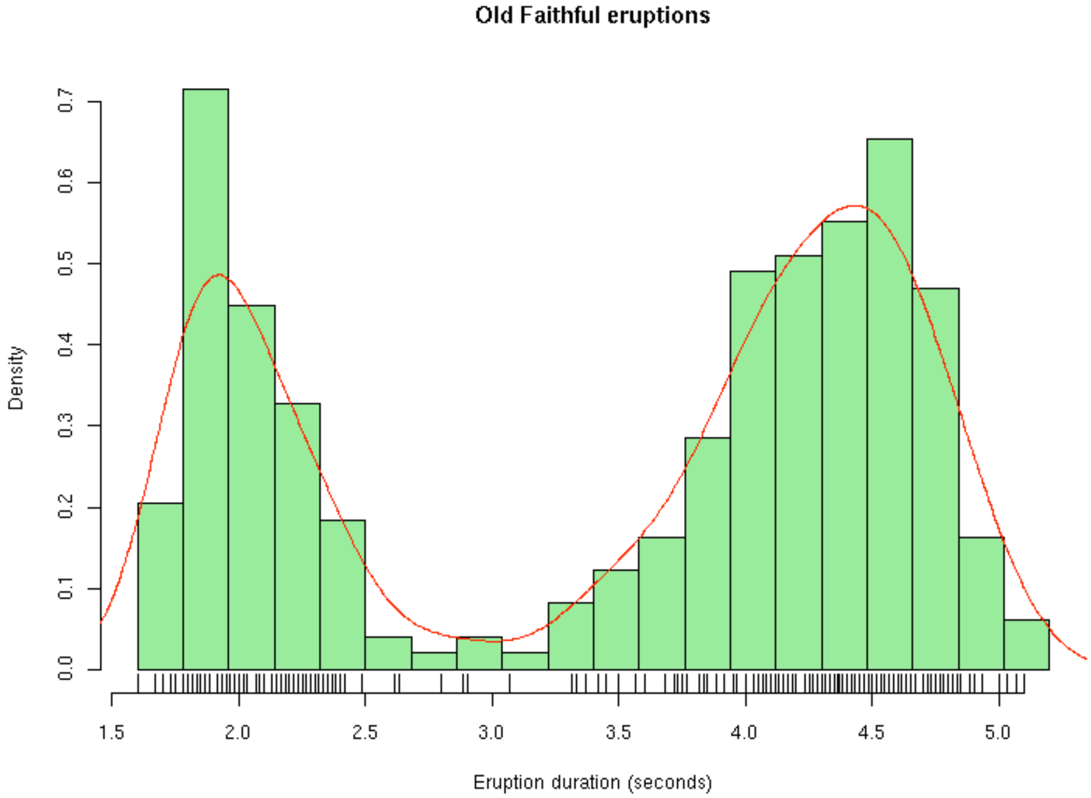
```
r.help('stem')
```

```
R Help on 'stem'stem                 package:graphics
R Documentation

Stem-and-Leaf Plots

Description:

     'stem' produces a stem-and-leaf plot of the values in 'x'. The
     parameter 'scale' can be used to expand the scale of the plot.
A
     value of 'scale=2' will cause the plot to be roughly twice as
long
     as the default.

Usage:

     stem(x, scale = 1, width = 80, atom = 1e-08)

Arguments:

       x: a numeric vector.

   scale: This controls the plot length.

   width: The desired width of plot.

    atom: a tolerance.

References:

     Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The Ne
S
     Language_. Wadsworth & Brooks/Cole.

Examples:

     stem(islands)
     stem(log10(islands))
```

```
@interact
def _(binsize=(0.2,(0.01,0.18))):
   print "binsize = ", binsize
   try:
       r.png('faithful_histogram.png',width=733,height=550)
       r.hist(ed,                        # ed = erruption duration
              r.seq(1.6, 5.2, binsize), # start, stop, binsize
              prob = 1,
              col  = "lightgreen",
              main = "Old Faithful eruptions",
              xlab = "Eruption duration (seconds)")

       # Draw a kernel density with given bandwidth.
       # This is a continuous analogue of a histogram.
       r.lines(r.density(ed, bw=binsize), col="red")
       r.rug(ed)
       _ = r.dev_off()
   except:
       print "Please try a different binsize."
```

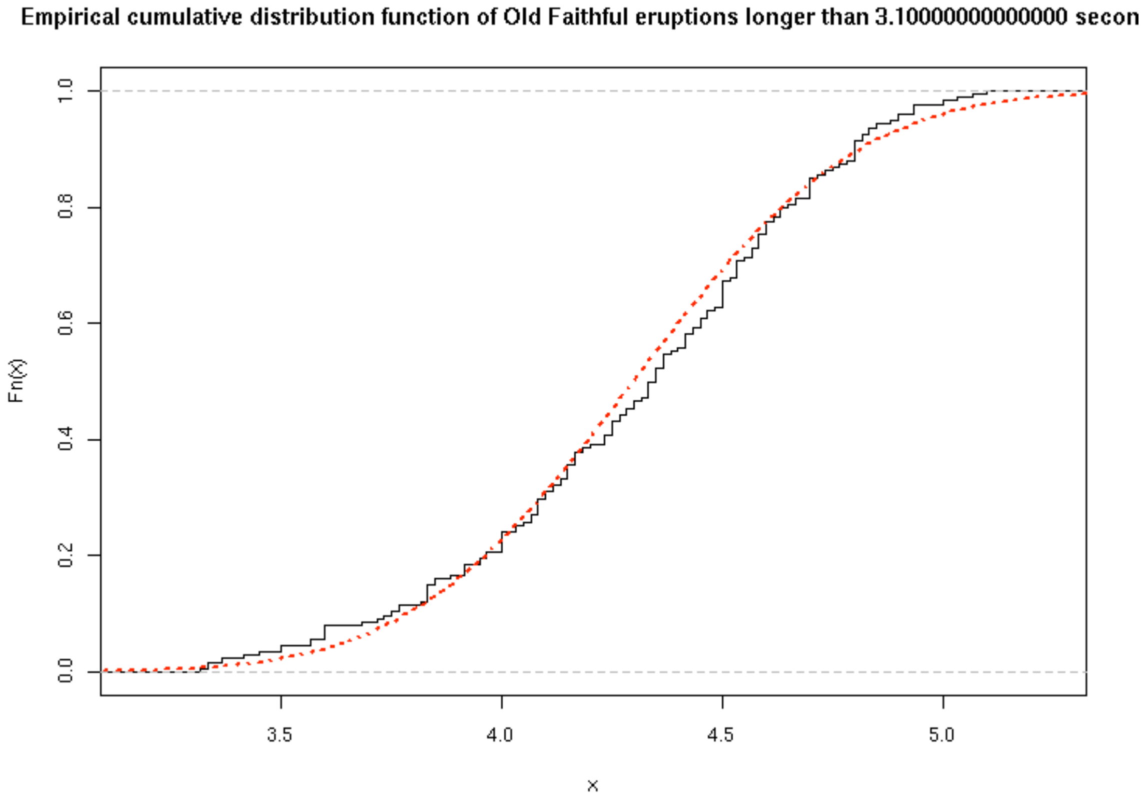
binsize

```
binsize =   0.180000000000000
```



```
@interact
def _(mintime=(3,(0.1..5))):
   long_ed = filter(lambda x: x > mintime, ed)
   r.png('faithful_ecdf.png',width=733,height=550)
   r.library('stats')
   r.plot(r.ecdf(long_ed), do_points=0, verticals=1,
          main="Empirical cumulative distribution function of " +\
               "Old Faithful eruptions longer than %s
seconds"%mintime)
   x = r.seq(mintime,5.4,0.01)
   r.lines(r.seq(mintime,5.4,0.01),
           r.pnorm(r.seq(mintime,5.4,0.01), mean=r.mean(long_ed),
                   sd=r.sqrt(r.var(long_ed))),
           lty=mintime, lwd=2, col="red")
   _=r.dev_off()
```

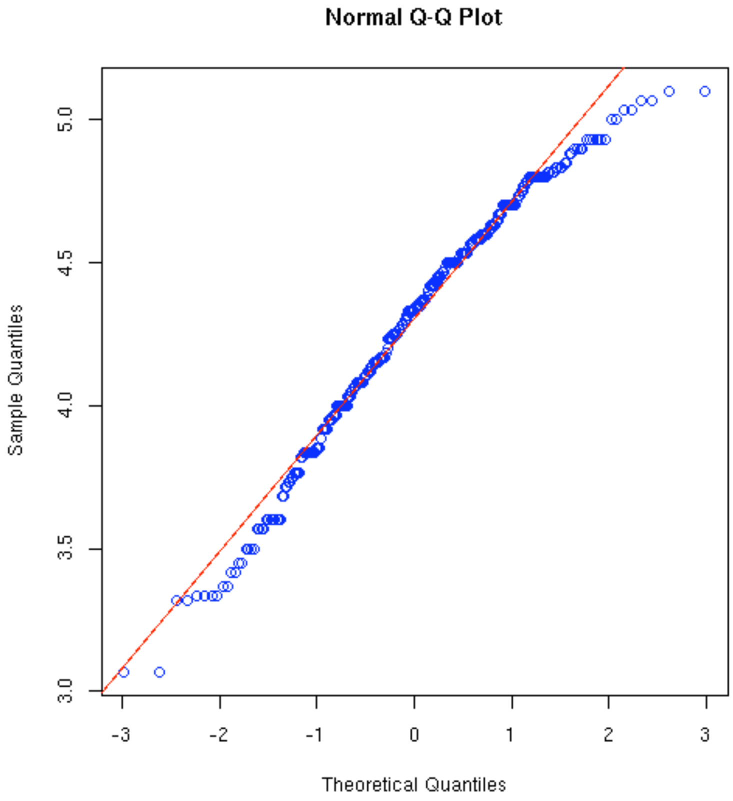
mintime



```
@interact
def _(mintime=(3,(0.1,5.0))):
   print "mintime = ", mintime
   long_ed = filter(lambda x: x > mintime, ed)
   r.png('faithful_qq.png',width=733,height=550)
   r.par(pty="s")
   r.qqnorm(long_ed,col="blue")
   r.qqline(long_ed,col="red")
   _=r.dev_off()
```

mintime

```
mintime =   3.02020202020202
```



```
r.help('qqnorm')
```

```
R Help on 'qqnorm'qqnorm                   package:stats
R Documentation

Quantile-Quantile Plots

Description:

     'qqnorm' is a generic function the default method of which
     produces a normal QQ plot of the values in 'y'. 'qqline' adds
     line to a normal quantile-quantile plot which passes through
the
     first and third quartiles.

     'qqplot' produces a QQ plot of two datasets.

     Graphical parameters may be given as arguments to 'qqnorm',
     'qqplot' and 'qqline'.

Usage:

     qqnorm(y, ...)
     ## Default S3 method:
     qqnorm(y, ylim, main = "Normal Q-Q Plot",
            xlab = "Theoretical Quantiles", ylab = "Sample
Quantiles",
            plot.it = TRUE, datax = FALSE, ...)

     qqline(y, datax = FALSE, ...)

     qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
            ylab = deparse(substitute(y)), ...)

Arguments:

       x: The first sample for 'qqplot'.

       y: The second or only data sample.

xlab, ylab, main: plot labels.  The 'xlab' and 'ylab' refer to the
          and x axes respectively if 'datax = TRUE'.

 plot.it: logical. Should the result be plotted?

   datax: logical. Should data values be on the x-axis?

ylim, ...: graphical parameters.

Value:

     For 'qqnorm' and 'qqplot', a list with components

       x: The x coordinates of the points that were/would be plotte

       y: The original 'y' vector, i.e., the corresponding y
```

```
r.library('ctest')
print("Shapiro-Wilks normality test of Old Faithful eruptions
longer than 3 seconds")
sw = r.shapiro_test(long_ed)
print "W = %.4f" % sw['statistic']['W']
print "p-value = %.5f" % sw['p.value']
```

```
Warning message:
package 'ctest' has been merged into 'stats'
Shapiro-Wilks normality test of Old Faithful eruptions longer than
seconds
W = 0.9789
p-value = 0.00005
```

Another Example With Generated Data

Thanks to Harald Schilly for suggesting this example.

```
r.options(warn=-1)

@interact
def _(nvals=(30..1000), mean1=(0,5), mean2=(1,(0,5)),
          sd1=(0.25,(0.25,3.0)), sd2=(1.3,(0.25,3.0)),
          generate=['Regenerate']):
   print 'nvals = ', nvals
   print 'input  sd1 = %s, mean1 = %s,\n       sd2 = %s, mean2 =
%s'%(sd1, mean1, sd2, mean2)
   group1 = r.rnorm(nvals, mean=mean1, sd=sd1)
   group2 = r.rnorm(nvals, mean=mean2, sd=sd2)

   r.png('sage1.png')
   r.plot_new()
   r.hist(group1, nvals//10+10, col='red', main = "Group 1",
xlab="Group 1")
   r.rug(group1)
   r.dev_off()

   r.png('sage2.png')
   r.plot_new()
   r.hist(group2, nvals//10+10, main = "Group 2", xlab="Group 1",
col='blue')
   r.rug(group2)
   r.dev_off()
   print "sample sd1 = %s, mean1 = %s"%(r.sd(group1),
r.mean(group1))
   print "       sd2 = %s, mean2 = %s"%(r.sd(group2),
r.mean(group2))
   r.png('sage3.png')
   r.plot_new()
   r.lines(r.density(group1), col='red')
   r.lines(r.density(group2), col='blue')
   r.rug(group1, col='red')
   r.rug(group2)
   r.dev_off()
```

nvals
mean1
mean2
sd1
sd2
generate Regenerate