Week 3: Simulation and Profiling

Objectives

Call the str function on an arbitrary R object

Describe the difference between the “by.self” and “by.total” output produced by the R profiler

Simulate a random normal variable with an arbitrary mean and standard deviation

Simulate data from a normal linear model

Str

- Compactly display the internal structure of an R object
- Alternative to summary
- str means structure
- nice for lists

Example 1: str

```
str(str)

## function (object, ...)

str(lm)

## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
```

Example 2: str

```
x <- rnorm(100,2,4)
summary(x)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.9470 -0.4328  1.9273  2.0356  4.3963 12.1238

str(x)

##  num [1:100] 4.047 0.294 6.894 3.472 -2.124 ...

f<- gl(40,10)
str(f)
```

```
##  Factor w/ 40 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Example 3: str with data set

```
library(datasets)
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
str(airquality)
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
s <- split(airquality, airquality$Month)
str(s)
```

```
## List of 5
##  $ 5:'data.frame':   31 obs. of  6 variables:
##   ..$ Ozone  : int [1:31] 41 36 12 18 NA 28 23 19 8 NA ...
##   ..$ Solar.R: int [1:31] 190 118 149 313 NA NA 299 99 19 194 ...
##   ..$ Wind   : num [1:31] 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##   ..$ Temp   : int [1:31] 67 72 74 62 56 66 65 59 61 69 ...
##   ..$ Month  : int [1:31] 5 5 5 5 5 5 5 5 5 5 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 6:'data.frame':   30 obs. of  6 variables:
##   ..$ Ozone  : int [1:30] NA NA NA NA NA NA 29 NA 71 39 ...
##   ..$ Solar.R: int [1:30] 286 287 242 186 220 264 127 273 291 323 ...
##   ..$ Wind   : num [1:30] 8.6 9.7 16.1 9.2 8.6 14.3 9.7 6.9 13.8 11.5 ...
##   ..$ Temp   : int [1:30] 78 74 67 84 85 79 82 87 90 87 ...
##   ..$ Month  : int [1:30] 6 6 6 6 6 6 6 6 6 6 ...
##   ..$ Day    : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 7:'data.frame':   31 obs. of  6 variables:
##   ..$ Ozone  : int [1:31] 135 49 32 NA 64 40 77 97 97 85 ...
##   ..$ Solar.R: int [1:31] 269 248 236 101 175 314 276 267 272 175 ...
##   ..$ Wind   : num [1:31] 4.1 9.2 9.2 10.9 4.6 10.9 5.1 6.3 5.7 7.4 ...
##   ..$ Temp   : int [1:31] 84 85 81 84 83 83 88 92 92 89 ...
##   ..$ Month  : int [1:31] 7 7 7 7 7 7 7 7 7 7 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 8:'data.frame':   31 obs. of  6 variables:
```

```
##   ..$ Ozone  : int [1:31] 39 9 16 78 35 66 122 89 110 NA ...
##   ..$ Solar.R: int [1:31] 83 24 77 NA NA NA 255 229 207 222 ...
##   ..$ Wind   : num [1:31] 6.9 13.8 7.4 6.9 7.4 4.6 4 10.3 8 8.6 ...
##   ..$ Temp   : int [1:31] 81 81 82 86 85 87 89 90 90 92 ...
##   ..$ Month  : int [1:31] 8 8 8 8 8 8 8 8 8 8 ...
##   ..$ Day    : int [1:31] 1 2 3 4 5 6 7 8 9 10 ...
##  $ 9:'data.frame':   30 obs. of  6 variables:
##   ..$ Ozone  : int [1:30] 96 78 73 91 47 32 20 23 21 24 ...
##   ..$ Solar.R: int [1:30] 167 197 183 189 95 92 252 220 230 259 ...
##   ..$ Wind   : num [1:30] 6.9 5.1 2.8 4.6 7.4 15.5 10.9 10.3 10.9 9.7 ...
##   ..$ Temp   : int [1:30] 91 92 93 93 87 84 80 78 75 73 ...
##   ..$ Month  : int [1:30] 9 9 9 9 9 9 9 9 9 9 ...
##   ..$ Day    : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
```

Generating Random Numbers

Function for probability distributions - rnorm: generate random Normal variates with a given mean and standard dev - dnorm: evaluate the Normal probability density with a given mean/sd at a point or vector of points - pnorm: evaluate the cumulative distribution function for a Normal distribution - rpois: generate random Poisson variates with a given rate

Functions prefixes - d: density - r: random number generation - p: cumulative distribution - q: quantile function

Example 4: Generating Random Numbers

```
x <- rnorm(10)
x
```

```
##  [1] -1.427177142  0.093360425 -1.185010094 -0.444091347 -0.860422137
##  [6]  0.292802136  0.317396246  1.043369276 -0.038322999 -0.002519582
```

```
x <- rnorm(10,20,2) #(n, mean, sd)
x
```

```
##  [1] 23.36012 19.74820 21.26168 21.54490 19.75605 21.43010 19.10583
##  [8] 15.60961 19.20224 19.94453
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.61   19.34   19.85   20.10   21.39   23.36
```

Need to set.seed for ensure reproducibility

Simulation of a linear model

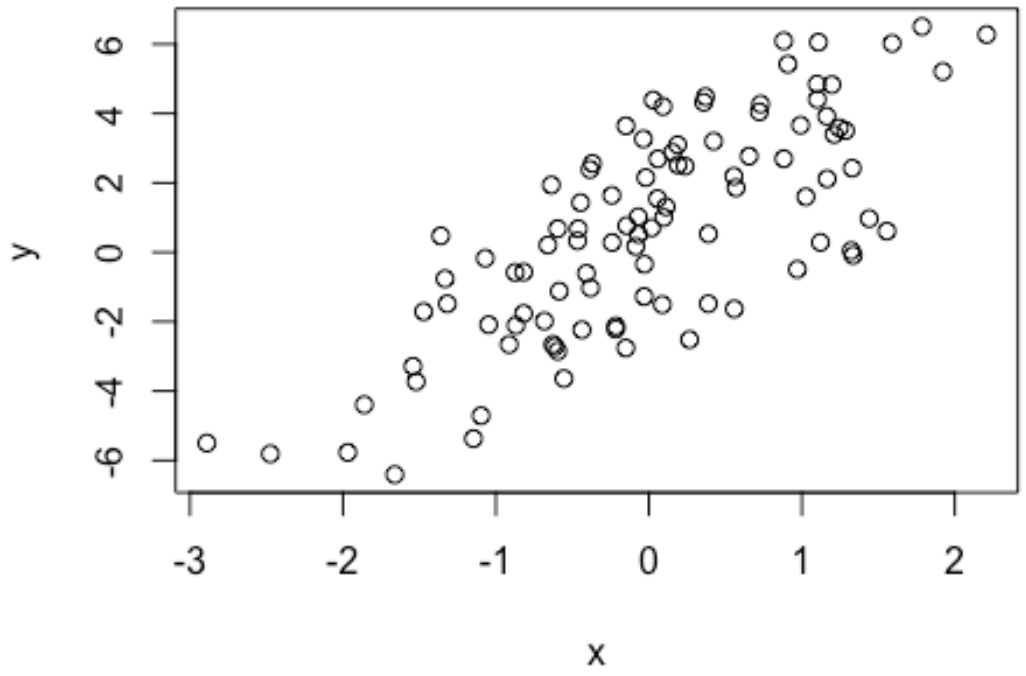
$y = \beta_o + \beta_1 x + \epsilon$

Example 5: Linear Model

```
set.seed(20)
x <- rnorm(100)
e <- rnorm(100,0, 2)
y <- 0.5 + 2 *x + e
summary(y)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.4084 -1.5402  0.6789  0.6893  2.9303  6.5052

plot(x,y)
```



Simulation - Random Sampling

Example 6: Random Sampling

```
set.seed(1)
sample(1:10,4)

## [1] 3 4 5 7

sample(letters,5)
```

```
## [1] "f" "w" "y" "p" "n"
```

```
sample(1:10) #permutation
```

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

```
sample(1:10 , replace = TRUE)
```

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

R Profile

-Systematic way to examine how much time is spent in different parts of a program. -Useful for optimization

Helpful codes: -system.time() -Rprof <- don't use with system.time() -summaryRprof() - by.total: divides time spent in each function by the total run time - by.self: does the same but first subtracts out time spent in functions above in the call stack