Introduction to Data Science

Daniel Gutierrez, Data Scientist

Los Angeles, Calif.

Course Outcomes

- Introduction to R Programming – Part 4

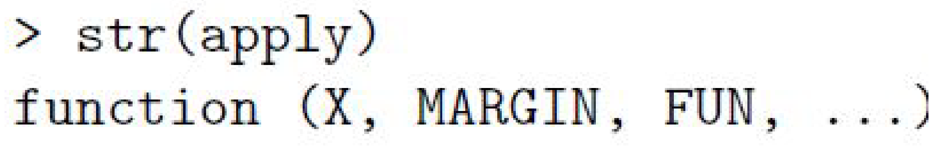
Lesson Objectives

- `apply()` loop function
- `tapply()` loop function
- `split()` function
- `mapply()` loop function
- Generating random numbers – normal, Poisson, binomial
- `sample()` function for random sampling
- Dates and times in R

R Fundamentals – Part 4

- `apply()` is used to evaluate a function (often an anonymous one) over the margins of an array.
  - It is most often used to apply a function to the rows or columns of a matrix
  - It can be used with general arrays, e.g., taking the average of an array of matrices
  - It is not really faster than writing a loop, but it works in one line!

R Fundamentals – Part 4



```
> str(apply)
function (X, MARGIN, FUN, ...)
```

- X is an array
- MARGIN is an integer vector indicating which margins should be “retained”.
- FUN is a function to be applied
- … is for other arguments to be passed to FUN

R Fundamentals – Part 4

- For sums and means of matrix dimensions, we have some shortcuts.
  - `rowSums = apply(x, 1, sum)`
  - `rowMeans = apply(x, 1, mean)`
  - `colSums = apply(x, 2, sum)`
  - `colMeans = apply(x, 2, mean)`
- The shortcut functions are much faster, but you won't notice unless you're using a large matrix.

R Fundamentals – Part 4

- `tapply()` is used to apply a function over subsets of a vector. I don’t know why it’s called tapply.
- `> str(tapply)`

  function (X, INDEX, FUN = NULL …, simplify = TRUE)
  - X is a vector
  - INDEX is a factor or a list of factors (or else they are coerced to factors)
  - FUN is a function to be applied
  - … contains other arguments to be passed FUN
  - Simplify, should we simplify the result?

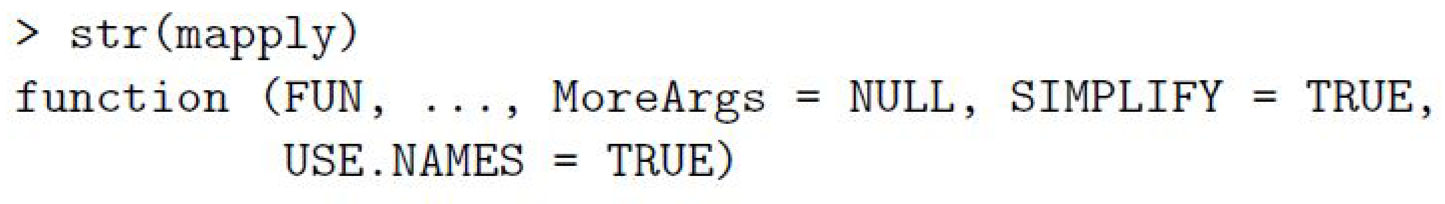
R Fundamentals – Part 4

- `split()` takes a vector or other objects and splits it into groups determined by a factor or list of factors
- `> str(split)`

  function (x, f, drop = FALSE, …)
  - x is a vector (or list) or data frame
  - f is a factor (or coerced to one) or a list of factors
  - drop indicated whether empty factors levels should be dropped

R Fundamentals – Part 4

- `mapply()` is a multivariate apply of sorts which applies a function in parallel over a set of arguments.



```
> str(mapply)
function (FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE,
          USE.NAMES = TRUE)
```

  - FUN is a function to apply
  - … contains arguments to apply over
  - MoreArgs is a list of other arguments to FUN.
  - SIMPLIFY indicates whether the result should be simplified

R Fundamentals – Part 4

- Functions for probability distributions in R
  - `rnorm()`: generate random Normal variates with a given mean and standard deviation
  - `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

R Fundamentals – Part 4

- Probability distribution functions usually have four functions associated with them.
- The functions are prefixed with a:
  - d for density
  - r for random number generation
  - p for cumulative distribution
  - q for quantile function

R Fundamentals – Part 4

- Drawing samples from specific probability distributions can be done with r* functions
- Standard distributions are built in: Normal, Poisson, Binomial, Exponential, Gamma, etc.
- The `sample()` function can be used to draw random samples from arbitrary vectors
- Setting the random number generator seed via `set.seed()` is critical for reproducibility

R Fundamentals – Part 4

- Every programming environment must choose methods for handling date and time values
- In R, you can use base R constructs as we’ll discuss in this class
- Later, you can adopt the features found in the `lubridate` package

R Fundamentals – Part 4

- R has developed a special representation of dates and times for numerical and statistical calculations
- Dates are represented by the `Date` class
- Times are represented by the `POSIXct` or the `POSIXlt` class
- Dates are stored internally as the number of days since 1970-0101
- Times are stored internally as the number of seconds since 1970-01-01
- Character strings can be coerced to Date/Time classes using the `strptime()` function or the `as.Date()`, `as.POSIXlt()`, or `as.POSIXct()` functions.

R Fundamentals – Part 4

- Times are represented using the `POSIXct` or the `POSIXlt` class
  - `POSIXct` is just a very large integer under the hood; it is a useful class when you want to store times in something like a data frame
  - `POSIXlt` is a list underneath and it stores a bunch of other useful information like the day of the week, day of the year, month, day of the month
- There are a number of generic functions that work on dates and times
  - `weekdays()`: gives the day of the week
  - `months()`: gives the month name
  - `quarters()`: give the quarter number (“Q1”, “Q2”, “Q3”, or “Q4”)

Code module

- WEEK 4-1 Code module – `apply()` loop function
- WEEK 4-2 Code module – `tapply()` loop function
- WEEK 4-3 Code module – `split` function
- WEEK 4-4 Code module – `mapply()` loop function
- WEEK 4-5 Code module – generating random numbers
- WEEK 4-6 Code module – `sample()` function
- WEEK 4-7 Code module – dates and times in R part 1
- WEEK 4-8 Code module – dates and times in R part 2

Summary

- In WEEK 4 of Introduction to Data Science we continue to build up our data science toolbox by adding some valuable tools for using R's loop functions, `apply()`, `tapply()` and `mapply()`, as well as the `split()` function.
- We also saw how to generate random numbers in R with distributions like Normal, Poisson and Binomial.
- We also saw how to do random sampling using the `sample()` function
- We saw how to handle date and time values in R