**Str() 🡺 Compactly Display the Structure of an Arbitrary R Object**

**Loops**

* **Lapply**
  + - Loop over a list and evaluate a function on each element
* **Sapply**
  + - Same as lapply but try to simplify the result
* **Apply**
  + - Apply a function over the margins of an array
* **Tapply**
  + - Apply a function over subsets of a vector
* **Mapply**
  + - Multivariable of apply
* **Split**

**Loop functions – lapply**

Function(x, FUN)

{

FUN <- match.fun(FUN)

If (lis.vector(X) || is. Object(x))

X <- as.list(X)

.internal(lapply(X, FUN))

}

* Lapply always return a list, regardless of the class of the input

X <- list(a = 1:5, b = rnorm(10))

Lapply(x, mean)

Or x <- 1:4

Lapply(x, runif, min = 0, max = 10)

* Lapply and other make heavy use of anonymous functions

Lapply(x, function(elt) elt[, 1])

**Sapply – the SAME as lapply**

* This will try to simplify the result of lapply if possible
  + - If the result is a list of elements of length [1], then a list is returned
    - If the result is a list of vectors of the same length, a matrix is returned
    - If neither, a list is returned

**Loop functions – apply**

* Apply a function over the margins of an array
* Is used to apply a function to the rows or columns of a matrix
* Used with general arrays, ex: taking the avg of an array of matrices

Str(apply)

Function(x, MARGIN, FUN,…)

X <- matrix()

(x, 2, mean)

* **Col/row sums and means**
  + - rowSums = apply(x, 1, sum)
    - rowMeans = apply(x, 1, mean)
    - colSums = apply(x, 2, sum)
    - colMeans = apply(x, 2, mean)

apply(x, 1, quantile, probs = c(0.25, 0.75)

* **Average matrix in an array**

a <- array(rnorm(2 \* 2 \* 10), C(2, 2, 10))

apply(a, c(1, 2), mean)

**or**

rowMeans(a, dims = 2)

**Loop functions – mapply**

* A multivariate apply which applies a function in parallel over a set of arguments

List(rep(1, 4), rep(2, 3), rep(3, 2), rep(4,1)

**Instead we can do**

Mapply(rep, 1:4, 4:1)

**Loop functions – tapply**

* Apply a function over subsets of a vector

**Loop functions – split**

* Split takes a vector or other objects and splits it into groups determined by a factor of a list of factors

X <- c(rnorm(10), runif(10), rnorm(10, 1))

F <- gl(3, 10)

Split(x, f)

* **Example**

Lapply(split(x, f), mean)

* **Splitting a data frame**

Library(“datasets”)

Head(airquality)

S <- split(airquality, airquality$month)

Lapply(s, function(x) colMeans(x[, c(“ozone”, “solar.r”, “wind”)], na.rm = TRUE))

* **Splitting on more than one level**

X <- rnorm(10)

F1 <- gl(2, 5)

F2 <- gl(5, 2)

Interaction(f1, f2)

Split(x, list(f1, f2), drop = TRUE)

**Debugging Tools – Diagnosing the Problem**

* Indications that something is not right
  + - Message
    - Warning
    - Error
    - Condition

**Debugging Tools – Basic Tools**

* Primary tools
  + - Traceback
    - Debug
    - Browser
    - Trace
    - Recover

**Debugging tools – using the tools**

* **Traceback**

Mean(x)

Traceback()

* **Debug**

Debug(lm)