Functional Programming HW

Kesicia Dickinson, Elizabeth Brannon, Shane Weary

2/21/2018

```
# install.packages('pryr')
```

Exercises 1

1. Given a function, like "mean", match.fun() lets you find a function. Given a function, can you find its name? Why doesn’t that make sense in R?

- Within R, functions are objects which are not automatically bound to a name. Since there is no syntax specifically for creating a named function you have to use a regular assignment operator to give the function a name when you create it. If you choose not to give a name then you will have an anonymous functions.

2. Use lapply() and an anonymous function to find the coefficient of variation (the standard deviation divided by the mean) for all columns in the mtcars dataset.

```
mtcars

##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
```

```
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
lapply( mtcars, FUN=function(x) sd(x) / mean(x) )
```

```
## $mpg
## [1] 0.2999881
##
## $cyl
## [1] 0.2886338
##
## $disp
## [1] 0.5371779
##
## $hp
## [1] 0.4674077
##
## $drat
## [1] 0.1486638
##
## $wt
## [1] 0.3041285
##
## $qsec
## [1] 0.1001159
##
## $vs
## [1] 1.152037
##
## $am
## [1] 1.228285
##
## $gear
## [1] 0.2000825
##
## $carb
## [1] 0.5742933
```

3. Use integrate() and an anonymous function to find the area under the curve for the following functions. Use Wolfram Alpha to check your answers.

y = x ^ 2 - x, x in [0, 10]

```
integrate( f=function(x) x^2-x, lower=0, upper=10 )

## 283.3333 with absolute error < 3.1e-12
```

y = sin(x) + cos(x), x in [-π, π]

```
integrate( f=function(x) sin(x) + cos(x), -pi, pi )

## 5.231803e-16 with absolute error < 6.3e-14
```

y = exp(x) / x, x in [10, 20]

```
integrate( f=function(x) exp(x)/x, lower=10, upper=20 )

## 25613160 with absolute error < 2.8e-07
```

4. A good rule of thumb is that an anonymous function should fit on one line and shouldn't need to use {}. Review your code. Where could you have used an anonymous function instead of a named function? Where should you have used a named function instead of an anonymous function?

Exercises 2

1. Why are functions created by other functions called closures?

- Functions created by other functions are called closures because they enclose the environment of the parent environment and they can access all of the parent's variables.

2. What does the following statistical function do? What would be a better name for it? (The existing name is a bit of a hint.)

```
bc <- function(lambda) {
  if (lambda == 0) {
    function(x) log(x)
  } else {
```

```
    function(x) (x ^ lambda - 1) / lambda
  }
}
```

- The proceeding function creates two functions depending on the value of lambda. More clearly, if lambda is 0 the it creates a function which calculates the log of x (log x). If lambda is not 0, it creates a function that calculates x raised to the power of lambda minus 1 divided by lambda (x^lambda-1/lambda).

3. What does approxfun() do? What does it return?

- approxfun() is an interpolation function which creates a function that interpolates or interject between data points given in the factor. It returns a function.

4. What does ecdf() do? What does it return?

- ecdf() calculates the value of empirical cumulative distribution function of the data points given to the factory. It returns the percentiles for x.

5. Create a function that creates functions that compute the ith central moment of a numeric vector. You can test it by running the following code:

```
moment <- function(n) {
    function(x) mean((x - mean(x))^n)
}

m1 <- moment(1)
m2 <- moment(2)

x <- runif(100)
stopifnot(all.equal(m1(x), 0))
stopifnot(all.equal(m2(x), var(x) * 99 / 100))
```

6. Create a function pick() that takes an index, i, as an argument and returns a function with an argument x that subsets x with i.

```
pick <- function(i){ # here, the function is taking i
  function(x){  # setting up the function to be returned
    x[[i]]
  }
}

lapply(mtcars, pick(5))
```

```
## $mpg
## [1] 18.7
##
## $cyl
## [1] 8
##
## $disp
## [1] 360
##
## $hp
## [1] 175
##
## $drat
## [1] 3.15
##
## $wt
## [1] 3.44
##
## $qsec
## [1] 17.02
##
## $vs
## [1] 0
##
## $am
## [1] 0
##
## $gear
## [1] 3
##
## $carb
## [1] 2
```

```
# should do the same as this
lapply(mtcars, function(x) x[[5]])
```

```
## $mpg
## [1] 18.7
##
## $cyl
## [1] 8
##
## $disp
## [1] 360
##
## $hp
## [1] 175
##
## $drat
## [1] 3.15
```

```
##
## $wt
## [1] 3.44
##
## $qsec
## [1] 17.02
##
## $vs
## [1] 0
##
## $am
## [1] 0
##
## $gear
## [1] 3
##
## $carb
## [1] 2
```

Exercises 3

1. Implement a summary function that works like base::summary(), but uses a list of functions. Modify the function so it returns a closure, making it possible to use it as a function factory.

```
summary(x)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01292 0.20195 0.47630 0.46571 0.70111 0.99726

# Here, I make the list of functions
summary1 <- list(
  mymin = function(x) min(x),
  my1stqu = function(x) quantile(x,0.25),
  mymedian = function(x) median(x),
  mymean = function(x) mean(x),
  my3rdqu = function(x) quantile(x,0.75),
  mymax = function(x) max(x)
  )

lapply(summary1,function(f) f(x))

## $mymin
## [1] 0.01292257
##
## $my1stqu
##       25%
## 0.2019483
##
```

```
## $mymedian
## [1] 0.4762978
##
## $mymean
## [1] 0.4657063
##
## $my3rdqu
##       75%
## 0.7011053
##
## $mymax
## [1] 0.997257
```

```r
# using the list of functions above for the factory function
summary2 <- function(){
  list(
    mymin = function(x) min(x),
  my1stqu = function(x) quantile(x,0.25),
  mymedian = function(x) median(x),
  mymean = function(x) mean(x),
  my3rdqu = function(x) quantile(x,0.75),
  mymax = function(x) max(x)
  )
}

# here's the closure
m <- summary2()
m[[2]](1:10)
```

```
##  25%
## 3.25
```

2. Which of the following commands is equivalent to with(x, f(z))?

a. x$f\,(x$z).

b. f(x$z). # c. x$f(z).

d. f(z).

e. It depends.

(E) It depends. Assuming z is in x, it's (b): f(x$z). If f is also part of x, then it's (a): x$f(x$z). But if neither is in x, then it's (d): f(z).

Exercises 4

1. Instead of creating individual functions (e.g., midpoint(), trapezoid(), simpson(), etc.), we could store them in a list. If we did that, how would that change the code? Can you create the list of functions from a list of coefficients for the Newton-Cotes formulae?

2. The trade-off between integration rules is that more complex rules are slower to compute, but need fewer pieces. For sin() in the range [0, π], determine the number of pieces needed so that each rule will be equally accurate. Illustrate your results with a graph. How do they change for different functions? sin(1 / x^2) is particularly challenging.