Lecture 4: solutions

Statistical Computing with R

Exercise 2

1

```
library(brolgar)
```

```
my_heights <- data.frame(heights)
head(my_heights)
```

```
##       country continent year height_cm
## 1 Afghanistan      Asia 1870    168.40
## 2 Afghanistan      Asia 1880    165.69
## 3 Afghanistan      Asia 1930    166.80
## 4 Afghanistan      Asia 1990    167.10
## 5 Afghanistan      Asia 2000    161.40
## 6     Albania    Europe 1880    170.10
```

2

The following function gets the country name as input, and proceeds to retrieve the rows in the data frame that match the supplied country name. If there is no match the length of `my_rows` is 0. In this case we display a warning and return `NA`.

```
my_h_function_1 <- function(country){
  my_rows <- which(my_heights$country == country)
  if (length(my_rows) == 0){
    warning("No data available")
    return(NA)
  }
  return(my_heights[my_rows, ])
}
```

For a real input like "Spain", it will return a data frame:

```
my_h_function_1("Spain")
```

```
##      country continent year height_cm
## 1228   Spain    Europe 1740     163.3
## 1229   Spain    Europe 1750     163.6
```

```
## 1230   Spain    Europe 1760     163.2
## 1231   Spain    Europe 1770     164.3
## 1232   Spain    Europe 1780     163.3
## 1233   Spain    Europe 1830     161.0
## 1234   Spain    Europe 1840     163.7
## 1235   Spain    Europe 1850     162.5
## 1236   Spain    Europe 1860     162.7
## 1237   Spain    Europe 1870     162.6
## 1238   Spain    Europe 1880     163.9
## 1239   Spain    Europe 1890     164.0
## 1240   Spain    Europe 1900     164.6
## 1241   Spain    Europe 1910     165.1
## 1242   Spain    Europe 1920     165.6
## 1243   Spain    Europe 1930     165.2
## 1244   Spain    Europe 1940     166.3
## 1245   Spain    Europe 1950     170.8
## 1246   Spain    Europe 1960     174.2
## 1247   Spain    Europe 1970     175.2
## 1248   Spain    Europe 1980     175.6
```

But for a non existing country it returns `NA`:

```
my_h_function_1("Gondor")
```

```
## Warning in my_h_function_1("Gondor"): No data available
```

```
## [1] NA
```

3

Now we have two arguments so we need two different warnings. We need to specify which of the two arguments is the problem.

```
my_h_function_2 <- function(country, min_year){
  my_rows <- which(my_heights$country == country &
                     my_heights$year >= min_year)
  if (length(my_rows) == 0){
    if(sum(my_heights$country == country) == 0){
      warning("No data available for this country")
      return(NA)
    }else{
      warning("No data available beyond this year")}
      return(NA)
  }
  return(my_heights[my_rows, ])
}
```

Notice that since we use the `if` statement when the length of `my_rows` is 0 we don't need to check for both conditions. It must be one or the other.

Again, for available data it will return a data frame:

```r
my_h_function_2("Italy", 1950)
```

```
##     country continent year height_cm
## 690   Italy    Europe 1950    171.30
## 691   Italy    Europe 1960    173.00
## 692   Italy    Europe 1970    174.10
## 693   Italy    Europe 1980    174.48
```

But for non available years or countries it will return `NA`.

```r
# no available data beyond the given year
my_h_function_2("Italy", 1990)
```

```
## Warning in my_h_function_2("Italy", 1990): No data available beyond this year
```

```
## [1] NA
```

```r
# no data available on the given country
my_h_function_2("Gondor", 1990)
```

```
## Warning in my_h_function_2("Gondor", 1990): No data available for this country
```

```
## [1] NA
```

Exercise 3

```r
f <- function(x){
  if(x <= -3){
    return(-5)
  }else if(x > -3 & x < 1){
    return(log(x + 5))
  }else if(x == 1){
    return(2)
  }else if(x > 1 & x <= 14){
    return(sqrt(x + 3))
  }else{
    return(log(x))
  }
}
for (x in seq(-6, 10, 2)){
  print(f(x))
}
```

```
## [1] -5
## [1] -5
## [1] 1.098612
## [1] 1.609438
## [1] 2.236068
## [1] 2.645751
## [1] 3
## [1] 3.316625
## [1] 3.605551
```

Exercise 4

```
g <- function(M){
  if(nrow(M) != ncol(M)){
    return("This matrix is not square")
  }else if(det(M) == 0){
    return("This matrix is square and has determinant 0")
  }else if(det(M) != 0){
    return(solve(M))
  }
}

# checking the function
A <- matrix(1:12, 4)
g(A)
```

```
## [1] "This matrix is not square"
```

```
B <- matrix(1:9, 3)
g(B)
```

```
## [1] "This matrix is square and has determinant 0"
```

```
C <- matrix(1:4, 2)
g(C)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

Exercise 5

```
h <- function(df){
  if(nrow(df) < 5){
    if(ncol(df) < 5){
      as.matrix(df)
    }else if(ncol(df) < 10){
      as.matrix(df[,(ncol(df)-4):ncol(df)])
    }else if(ncol(df) >= 10){
      as.matrix(df[,1:5])
    }
  }else if(nrow(df) >= 5){
    if(ncol(df) < 5){
      as.matrix(df[(nrow(df)-4):nrow(df),])
    }else if(ncol(df) < 10){
      as.matrix(df[(nrow(df)-4):nrow(df),(ncol(df)-4):ncol(df)])
    }else if(ncol(df) >= 10){
      as.matrix(df[(nrow(df)-4):nrow(df),1:5])
```

```
    }
  }
}

# checking the function
A <- as.data.frame(matrix(1:(20*20), 20))

h(A[1:4, 1:4])
```

```
##   V1 V2 V3 V4
## 1  1 21 41 61
## 2  2 22 42 62
## 3  3 23 43 63
## 4  4 24 44 64
```

```
h(A[1:4, 1:7])
```

```
##   V3 V4 V5  V6  V7
## 1 41 61 81 101 121
## 2 42 62 82 102 122
## 3 43 63 83 103 123
## 4 44 64 84 104 124
```

```
h(A[1:4, 1:11])
```

```
##   V1 V2 V3 V4 V5
## 1  1 21 41 61 81
## 2  2 22 42 62 82
## 3  3 23 43 63 83
## 4  4 24 44 64 84
```

```
h(A[1:7, 1:4])
```

```
##   V1 V2 V3 V4
## 3  3 23 43 63
## 4  4 24 44 64
## 5  5 25 45 65
## 6  6 26 46 66
## 7  7 27 47 67
```

```
h(A[1:7, 1:7])
```

```
##   V3 V4 V5  V6  V7
## 3 43 63 83 103 123
## 4 44 64 84 104 124
## 5 45 65 85 105 125
## 6 46 66 86 106 126
## 7 47 67 87 107 127
```

```r
h(A[1:7, 1:11])
```

```
##   V1 V2 V3 V4 V5
## 3  3 23 43 63 83
## 4  4 24 44 64 84
## 5  5 25 45 65 85
## 6  6 26 46 66 86
## 7  7 27 47 67 87
```