Solutions to Selected Exercises for Braun and Murdoch's A First Course in Statistical Programming with R

Kristy Alexander, Yiwen Diao, Qiang Fu, and Yu Han
W. John Braun and Duncan J. Murdoch

July 17, 2010

Chapter 2

Introduction to the R Language

2.2 Basic Features of R

1.
```
> 170166719 %% 31079
[1] 9194
```

3. (a)
```
> r <- 1.08
> n <- c(10, 20, 30, 40)
> sum1 <- c()
> for(i in n){
+   x <- 1:i
+   sum1 <- c(sum1, sum(r^x))
+ }
> sum1
[1]  15.64549  49.42292 122.34587 279.78104
```
This gives the calculated sums for n = 10, 20, 30, 40.
```
> sum2 <- (1 - r^(n + 1)) / (1 - r)
> sum2
[1]  16.64549  50.42292 123.34587 280.78104
> sum2 - sum1
[1] 1 1 1 1
```
The formula is the actual sum plus one.

5.
```
> Sum <- function(n){
+        x <- 1:n
+        ans <- sum(x)
+        ans
+ }
```

(a)
```
> n <- 100
> Sum(n)
[1] 5050
> formula <- (n * (n + 1)) / 2
> Sum(n) - formula
[1] 0
```

(c)
```
> n <- 400
> formula <- (n * (n + 1)) / 2
> Sum(n) - formula
```

```
[1] 0
```

9. ```
   > Sums <- function(N){
   +      x <- 1:N
   +      y <- (1/x)
   +      sums <- sum(y)
   +      return(sums)
   + }
   > formula <- function(N){
   +      log(N) + 0.6
   + }
   ```

   (a) ```
   > N <- 500
   > Sums(N) - formula(N)
   [1] -0.02178467
   ```

   (c) ```
   > N <- 2000
   > Sums(N) - formula(N)
   [1] -0.02253436
   ```

   (e) ```
   > N <- 8000
   > Sums(N) - formula(N)
   [1] -0.02272184
   ```

10. R can only store up to 15 or 16 digits. Therefore, for `x[0.9999999999999999]`, R can detect that the index is less than 1 and truncates it down to 0 when using it for indexing. The result is the same as `x[0]` and hence the output is `numeric(0)`. However, for `x[0.99999999999999999]` there are 17 9's so R sees it as equal to 1. It therefore outputs `x[1]`.

11. (a) ```
    > rep(0:4, rep(5,5))
     [1] 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
    ```

12. ```
    > rep( seq(1,5), times = 5) + rep( 0:4, each = 5)
    [1] 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8 5 6 7 8 9
    ```

2.3 Built-in Functions and Online Help

1. (a) ```
   > solar.radiation <- c(11.1,10.6,6.3,8.8,10.7,11.2,8.9,12.2)
   ```

   (b) ```
   > mean(solar.radiation)
   [1] 9.975
   > median(solar.radiation)
   [1] 10.65
   > var(solar.radiation)
   [1] 3.525
   ```

   (c) i. ```
   > sr10 <- solar.radiation + 10
   ```

   ii. ```
   > mean(sr10)
   [1] 19.975
   > median(sr10)
   [1] 20.65
   > var(sr10)
   [1] 3.525
   ```

   iii. The mean and the median are increased by 10. The variance remains unchanged.

(f) Option 1: We can use `?var` or `help(var)` to read details of the formula used by the `var()` function. We discover that the denominator $(n-1)$ is used.

Option 2: We can work the variance out manually for `solar.radiation` and compare with the `var()` results:

```
> var1 <- (1 / length(solar.radiation)) * sum((solar.radiation - mean(solar.radiation))^2)
> var2 <- (1 / (length(solar.radiation) - 1))*sum((solar.radiation - mean(solar.radiation))^2)
> var1
[1] 3.084375
> var2
[1] 3.525
> var(solar.radiation)
[1] 3.525
```

This implies that the denominator $(n-1)$ is used in the `var()` function in R.

2.4 Logical Vectors and Relational Operators

2.4.1 Boolean algebra

1. Truth Table for "xor"

A	B	xor
True	True	False
True	False	True
False	True	True
False	False	False

```
xor = (A or B) and (not (A and B))
```

3. (a) `!(A & B) == (!A)|(!B)`

The event that A and B are not both true is the same as the event that either A is not true or B is not true.

The statement "There is no sun shower" is equivalent to the statement "Either the sky is not clear or it is not raining".

`Truth Table to confirm that !(A & B) == (!A)|(!B)`

A	B	not (A and B)	(not A) or (not B)
True	True	False	False
True	False	True	True
False	True	True	True
False	False	True	True

This implies that they are equivalent.

2.4.2 Logical Operations in R

1. `B` needs to be evaluated in the expression `A||B` when `A` is `FALSE`. If `A` is `TRUE` we already know that `A||B` is `TRUE`. However, if `A` is `FALSE`, the value of `A||B` would depend on the value of `B`.

3. `> b * a`

```
[1] 13  0  0  2
```

This multiplies `b` with the (coerced) numeric values of `a`, element by element.

2.5 Data input and output

2.5.2 dump() and source()

3.
```
> numbers <- c(3,5,8,10,12)
> dump( "numbers", file = "numbers.R")
> rm(numbers)
> ls()  # numbers should not appear in the resulting listing
```

```
 [1] "a"               "ans"             "b"               "formula"
 [5] "i"               "intRate"         "more.colours"    "n"
 [9] "N"               "payment"         "principal"       "r"
[13] "randomdata"      "solar.radiation" "sr10"            "srm2"
[17] "Sum"             "sum1"            "sum2"            "Sums"
[21] "Sumsqrd"         "values"          "var1"            "var2"
[25] "vp"              "vp1"             "x"
```

```
> source("numbers.R")
> ls()  # numbers should now appear in the resulting listing
```

```
 [1] "a"               "ans"             "b"               "formula"
 [5] "i"               "intRate"         "more.colours"    "n"
 [9] "N"               "numbers"         "payment"         "principal"
[13] "r"               "randomdata"      "solar.radiation" "sr10"
[17] "srm2"            "Sum"             "sum1"            "sum2"
[21] "Sums"            "Sumsqrd"         "values"          "var1"
[25] "var2"            "vp"              "vp1"             "x"
```

2.5.6 Lists

1. Creating `pretend.df`:

```
> x <- c(61,175,111,124)
> y <- c(13,21,24,23)
> z <- c(4,18,14,18)
> pretend.df <- cbind(x,y,z)
> pretend.df <- data.frame(pretend.df)
> pretend.df
```

```
    x  y  z
1  61 13  4
2 175 21 18
3 111 24 14
4 124 23 18
```

Displaying the item:

```
> pretend.df[ 1, 3]
```

```
[1] 4
```

Chapter Exercises

1. Download *rnf6080.dat* from the given website and save on your computer as "rnf6080.dat".

```
> rain.df <-read.table("rnf6080.dat", header = FALSE,
+                      na.strings = "-999")
```

(a) `> rain.df[ 2, 4]`

```
[1] 0
```

(b)
```
> names(rain.df)
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12"
[13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24"
[25] "V25" "V26" "V27"
```

(c)
```
> rain.df[ 2, ]
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
  V22 V23 V24 V25 V26 V27
2   0   0   0   0   0   0
```

(e)
```
> rain.df$daily <- rowSums(rain.df[ , 4:27], na.rm=TRUE)
```