CompStat/R - Paper 2

Group 2: Carlo Michaelis, Patrick Molligo, Lukas Ruff

21 June 2016

Part I: Functions

Functions I

Below we define a function `dropNa` which given an atomic vector `x` as argument, returns `x` after removing missing values.

```
dropNa <- function(x) {
  # expects an atomic vector as an argument and returns it without missing
  # values
  #
  # Args:
  #   x: atomic vector
  #
  # Returns:
  #   The atomic vector x without missing values

  # To remove the NAs, we use simple logical subsetting
  y <- x[!is.na(x)]

  # Return y
  y
}
```

Let's test our implementation with the following line of code:

```
all.equal(dropNa(c(1, 2, 3, NA, 1, 2, 3)), c(1, 2, 3, 1, 2, 3))
```

```
## [1] TRUE
```

As we can see from this positive test, our implementation was successful.

Functions II

Part I Below we define a function `meanVarSdSe` which given a numeric vector `x` as argument, returns the mean, the variance, the standard deviation and the standard error of `x`.

```
meanVarSdSe <- function(x) {
  # expects a numeric vector as an argument and returns the mean,
  # the variance, the standard deviation and the standard error
  #
  # Args:
  #   x: numeric vector
  #
  # Returns:
```

```
  #   a numerical vector containing mean, variance, standard deviation
  #   and standard error of x

  # We check if x is numeric vector
  # If not: stop and throw error
  if( !is.numeric(x) ) {
    stop("Argument need to be numeric.")
  }

  # Create vector object
  y <- vector()

  # Calculate mean, variance, standard deviation and standard error
  # and save it in y
  y[1] <- mean(x)
  y[2] <- var(x)
  y[3] <- sd(x)
  y[4] <- y[3]/sqrt(length(x))

  # Set names to vector entries
  names(y) <- c("mean", "var", "sd", "se")

  # Return the numeric vector y
  y
}
```

To test the function, we define a numeric vector, which contains numbers from 1 to 100 and use it as an argument for our function `meanVarSdSe`:

```
x <- 1:100
meanVarSdSe(x)
```

```
##       mean        var         sd         se
##  50.500000 841.666667  29.011492   2.901149
```

Finally we can confirm, that the result is of type `numeric`:

```
class(meanVarSdSe(x))
```

```
## [1] "numeric"
```

Part II Now we will have a look at the case below. We would expect that the function will return a vector with `NA`s:

```
x <- c(NA, 1:100)
meanVarSdSe(x)
```

```
## mean  var   sd   se
##   NA   NA   NA   NA
```

The reason for the result is that the functions `mean()`, `var()` and `sd()` use `na.rm = FALSE` as default, which means that missing values are not removed. If the vector `x` contains a missing value, the `mean()` function (`var()` and `sd()` respectively) will just return `NA` to inform about missing values. In the case of calculating standard error we use the result from our `sd()` function and calculate a `NA` value with some other numeric values, which will results in `NA` again.

To solve the problem, we should can add `na.rm = TRUE` to those three functions. To make this optionally, we will improve the `meanVarSdSe` function from above as follows:

```
meanVarSdSe <- function(x, ...) {
  # expects a numeric vector and flag to handle missing values as an argument
  # and returns the mean, the variance, the standard deviation
  # and the standard error
  #
  # Args:
  #   x: numeric vector, na.rm: boolean
  #
  # Returns:
  #   a numerical vector containing mean, variance, standard deviation
  #   and standard error of x

  # We check if x is numeric vector
  # If not: stop and throw error
  if( !is.numeric(x) ) {
    stop("Argument need to be numeric.")
  }

  # Create vector object
  y <- vector()

  # Calculate mean, variance, standard deviation and standard error
  # and save it in y
  y[1] <- mean(x, ...)
  y[2] <- var(x, ...)
  y[3] <- sd(x, ...)
  y[4] <- y[3]/sqrt(length(x) - sum(is.na(x)))

  # Set names to vector entries
  names(y) <- c("mean", "var", "sd", "se")

  # Return the numeric vector y
  y
}
```

We define the function with an ellipse `...`. Our function can now get multiple arguments after the first `x`. These arguments are used in `mean()`, `var()` and `sd()`. If we want to remove missing values in all these functions (to get a result in case of having missing values), we can pass `na.rm = TRUE` as another argument, like `meanVarSdSe(x, na.rm = TRUE)`. We just have to be aware of `length(x)` in this case. If we want to have the same result as above we have to remove the sum of `NA` values from the length of `x`. Otherwise the function will calculate a different result than in Part I, because then lentgh differs.

Lets confirm the result:

```r
meanVarSdSe(c(x, NA), na.rm = TRUE)
```

```
##       mean        var         sd         se
##  50.500000 841.666667  29.011492   2.901149
```

Part III Now we will use the function `dropNa` from Functions I to deal with missing values in `meanVarSdSe`.

```r
meanVarSdSe <- function(x) {
  # expects a numeric vector as an argument and returns the mean,
  # the variance, the standard deviation and the standard error
  # it also removes missing values if x contains some
  #
  # Args:
  #   x: numeric vector
  #
  # Returns:
  #   a numerical vector containing mean, variance, standard deviation
  #   and standard error of x

  # We check if x is numeric vector
  # If not: stop and throw error
  if( !is.numeric(x) ) {
    stop("Argument need to be numeric.")
  }

  # We check if x contains missing values
  # If so: remove missing values using dropNA
  if( sum(is.na(x)) > 0 ) {
    x <- dropNa(x)
  }

  # Create vector object
  y <- vector()

  # Calculate mean, variance, standard deviation and standard error
  # and save it in y
  y[1] <- mean(x)
  y[2] <- var(x)
  y[3] <- sd(x)
  y[4] <- y[3]/sqrt(length(x))

  # Set names to vector entries
  names(y) <- c("mean", "var", "sd", "se")

  # Return the numeric vector y
  y
}
```

We used the function from Part I and added a condition which checks if we have missing values in `x`, using `is.na`. If the sum of `NA` values is greater than 0 (if there is one or more missing value), we use the function `dropNA` from the beginning to remove all missing values. The remaining code of the function can stay like above in Part I.

We can confirm the result:

```
meanVarSdSe(c(x, NA))
```

```
##       mean        var         sd         se
##  50.500000 841.666667  29.011492   2.901149
```

Functions III

In this section we define an infix function `%or%`. This function should behave like the logical operator `|`.

```
# Define infix function %or%
`%or%` <- function(a, b) {
  # Check if vector a and b is logical
  if( !(is.logical(a) & is.logical(b)) ) {
    stop("a and/or b have to be logical vectors.")
  }

  # Use ifelse to calculate result and return it directly
  # If the sum of entry of vector a and entry of vector b
  # is greater or equal to 1, set result to TRUE, otherweise to FALSE
  ifelse(a + b >= 1,  TRUE, FALSE)
}
```

First we check if we have logical vectors. If `a` and/or `b` are not logical, we leave the function and throw an error. Otherwise we can calculate the `or` operation using `ifelse` function and return the result directly after calculation. Inside of the `ifelse` function, the first argument checks the condition if the sum of the values `a` and `b` are greater or qual to 1, where `TRUE` equals to 1 and `FALSE` quals to 0.

To confirm the function, we test an example:

```
c(TRUE, FALSE, TRUE, FALSE) %or% c(TRUE, TRUE, FALSE, FALSE)
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

Part II: Scoping and related topics

Scoping I

Scoping II

Scoping III

Dynamic lookup