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

  # First we check if we have a numerical 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, na.rm = FALSE) {
  # 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 we have a numerical 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, na.rm = na.rm)
  y[2] <- var(x, na.rm = na.rm)
  y[3] <- sd(x, na.rm = na.rm)
  if( na.rm == FALSE ) {
    y[4] <- y[3]/sqrt(length(x))
  } else {
    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 a second argument `na.rm` which has the default value `FALSE`. If we just give one argument, like `meanVarSdSe(x)` the function will take `na.rm = FALSE` for us and respectivly sets the same value as argument 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 use the second parameter by position, like `meanVarSdSe(x, TRUE)`, or by name, 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:

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

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

Part III

Functions III

Part II: Scoping and related topics

Scoping I

Scoping II

Scoping III

Dynamic lookup