Testing COPE

Alexandros Rekkas

2021-02-24

Contents

1 Testing supporting functions **1**

2 Testing server functions **2**

2.1 Interactivity 3

1 Testing supporting functions

This set of tests evaluates the functions used to support the server function. These functions are used for calculating relevant quantities, e.g. 28-day mortality risk.

Function `createModelMatrix` creates the model matrix based on a set of covariates and a list of transformations.

```
testthat::test_that(
    "Creation of model matrix works",
    {
        testthat::expect_equal(
            createModelMatrix(
                covariates      = rep(1, 6),
                transformations = transformationsMortality
            ),
            c(1, rep(0, 5))
        )
    }
)
```

```
## Test passed
```

Function `createLinearPredictor` calculates the linear predictor of a prediction based on a provided model matrix, a vector of β coefficients and an intercept

```
testthat::test_that(
    "Creation of linear predictor works",
    {
        testthat::expect_equal(
            createLinearPredictor(
                modelMatrix = rep(1, 6),
                beta        = rep(1, 6),
                intercept   = 1
            ),
            matrix(7)
        )
```

```
    }
)
```

```
## Test passed
```

Function `logisticProbability` calculates the logistic probability (%) based on a provided linear predictor value.

```
testthat::test_that(
    "Logistic probability calculation from linear predictor works",
    {
        testthat::expect_equal(
            logisticProbability(0),
            50
        )
    }
)
```

```
## Test passed
```

Finally, function `extractQuantiles` extracts the required quantiles for the calibration plot, based on the stored dataframe of `calibrationQuantiles`, the outcome of interest and the hospital under consideration.

```
testthat::test_that(
    "Extraction of calibration quantiles works",
    {
        testthat::expect_equal(
            extractQuantiles(
                outcome              = 1,
                center               = 1,
                calibrationQuantiles = data.frame(
                    center  = 1,
                    outcome = 1,
                    quant20 = 20,
                    quant40 = 40,
                    quant60 = 60,
                    quant80 = 80
                )
            ),
            c(
                quant20 = 20, quant40 = 40,
                quant60 = 60, quant80 = 80
            )
        )
    }
)
```

```
## Test passed
```

2 Testing server functions

Here we perform a set of unit tests to ensure that server-side operations work the way they should. We start by looking at calculations tasks run within the server function.

2.1 Interactivity

Interactive values, are server-side variables that depend on the input and, therefore, need to be updated whenever the user alters their selection.

First, the `currentInputData` reactive dataframe should contain all the provided inputs and should update accordingly, when these are changed.

```
shiny::testServer(
    expr = {
        session$setInputs(
            age                       = 70,
            respiratoryRate           = 19,
            ldh                       = 244,
            crp                       = 48,
            albumin                   = 39,
            urea                      = 6.5,
            calculatePredictionButton = "click"
        )

        testthat::test_that(
            "The reactive input dataframe is correct",
            {
                testthat::expect_equal(
                    currentInputData(),
                    data.frame(
                        age             = 70,
                        respiratoryRate = 19,
                        crp             = 48,
                        ldh             = 244,
                        albumin         = 39,
                        urea            = 6.5
                    )
                )
            }
        )

        session$setInputs(
            age                       = 70,
            respiratoryRate           = 19,
            ldh                       = 2440,
            crp                       = 48,
            albumin                   = 39,
            urea                      = 6.5,
            calculatePredictionButton = "click"
        )

        testthat::test_that(
            "The reactive input dataframe is updated",
            {
                testthat::expect_equal(
                    currentInputData(),
                    data.frame(
                        age             = 70,
                        respiratoryRate = 19,
```

```
                        crp             = 48,
                        ldh             = 2440,
                        albumin         = 39,
                        urea            = 6.5
                    )
                )
            }
        )

    }
)
```

```
## Test passed
## Test passed
```

Next, we do the same for the variables that keep track of the current prediction and its placement, relevant to the overall predicted risk fifths, for both outcomes (mortality and ICU admission).

```
shiny::testServer(
    expr = {
        session$setInputs(
            age                       = 70,
            respiratoryRate           = 19,
            ldh                       = 244,
            crp                       = 48,
            albumin                   = 39,
            urea                      = 6.5,
            calculatePredictionButton = "click"
        )

        # Is the initial input admissible?
        testthat::expect_equal(
            admissibleInput(),
            TRUE
        )

        # Is the prediction for the starting values correct?
        testthat::expect_equal(
            currentPrediction(),
            list(
                mortality = 4.8,
                icu       = 13.3
            )
        )

        # Is the predicted mortality risk assigned to the correct stratum of risk?
        testthat::expect_equal(
            riskFifthMortality(),
            4
        )

        # Is the predicted ICU risk assigned to the correct stratum of risk?
        testthat::expect_equal(
            riskFifthIcu(),
            4
```

```
        )


        session$setInputs(
            age                       = 70,
            respiratoryRate           = 19,
            ldh                       = 2440,
            crp                       = 48,
            albumin                   = 39,
            urea                      = 6.5,
            calculatePredictionButton = "click"
        )

        testthat::test_that(
            "The reactive input dataframe is updated",
            {
                testthat::expect_equal(
                    currentInputData(),
                    data.frame(
                        age             = 70,
                        respiratoryRate = 19,
                        crp             = 48,
                        ldh             = 2440,
                        albumin         = 39,
                        urea            = 6.5
                    )
                )
            }
        )

    }
)
```

```
## Test passed
```