TestThat Tests Documentation

Ammar Hasan 150454388

20 November 2018

Contents

1 Introduction

This document documents the testing carried on the Data Preparation phase functions using TestHat. The tests consisted of test_that blocks with test_that functions in two files:

- 1Cleaning.R for testing functions involved with cleaning the data.
- 2AggregationMerge.R for testing functions involved with aggregating and merging data

Both files are in the tests directory of the ProjectTemplate project file structure.

2 Tests

2.1 Cleaning Functions Tests

2.1.1 Archetype Cleaning

```r
archetypesDf = archetypesClean(cyber.security.7.archetype.survey.responses)

test_that("Check if arhcetype clean removed appropriate fields", {
  expect_true(is.null(archetypesDf$responded_at))
  expect_true(is.null(archetypesDf$id))
})

test_that("Check if arhcetype clean fixed ID format to
          character", {
  expect_that(class(archetypesDf$learner_id), equals('character'))
})
```

These tests ensure that the archetypeClean() method carries its tasks described in the Data Preparation report, ensuring:

- That fields that are not needed are removed
- The ID format is appropriate and will not cause any issues with the joins

2.1.2 Questions Cleaning

```r
questionsDf = questionsClean(cyber.security.7.question.response)

test_that("Check if questions clean removed appropriate fields", {
  # Check if redudant quiz_question field was removed
  expect_true(is.null(questionsDf$quiz_question))

  # Check if non-unique question type field was removed
  expect_true(is.null(questionsDf$question_type))

  # Check if the cloze response field was removed
  expect_true(is.null(questionsDf$cloze_response))
})

test_that("Check if questions clean fixed ID format to
          character", {
  expect_that(class(questionsDf$learner_id), equals('character'))
})

test_that("Check if questions clean fixed date formats", {
  expect_that(class(questionsDf$submitted_at), equals('Date'))
})

test_that("Check if questions clean fixed 'correct' format to
          logical", {
  expect_that(class(questionsDf$correct), equals('logical'))
})

test_that("Check if questions clean dropped rows with empty ID", {
  expect_that(nrow(questionsDf),
              equals(nrow(drop_na(cyber.security.7.question.response, learner_id))))
})
```

These tests ensure that the questionsClean() method carries its tasks described in the Data Preparation report, ensuring:

- That fields that are not needed are removed
- The ID format is appropriate and will not cause any issues with the joins
- Date formats are fixed to date for future comparisons
- Correct field format is logical for logical operations
- No empty IDs after cleaning

2.1.3 Steps Cleaning

```r
stepsDf = stepsClean(cyber.security.7.step.activity)

test_that("Check if steps clean removed appropriate fields", {
  # Check if redudant step field was removed
  expect_true(is.null(stepsDf$step))
})

test_that("Check if steps clean fixed date formats", {
  expect_that(class(stepsDf$first_visited_at),
              equals('Date'))
  expect_that(class(stepsDf$last_completed_at),
              equals('Date'))
})

test_that("Check if steps clean fixed ID format to
          character", {
  expect_that(class(stepsDf$learner_id), equals('character'))
})
```

These tests ensure that the stepsClean() method carries its tasks described in the Data Preparation report, ensuring:

- That fields that are not needed are removed
- The ID format is appropriate and will not cause any issues with the joins
- Date formats are fixed to date for future comparisons

2.1.4 Enrol Cleaning

```r
countryDf = enrolClean(cyber.security.7.enrolments)
keptFieldsCount = 2

test_that("Check if steps entrol removed appropriate fields", {
  # Check if only learner id and expected country remain
  expect_that(ncol(countryDf), equals(keptFieldsCount))
  expect_true(!is.null(countryDf$learner_id))
  expect_true(!is.null(countryDf$detected_country))
})

test_that("Check if enrol clean removed -- for Not Detected", {
  expect_that(nrow(subset(countryDf, detected_country == '--')), equals(0))
  expect_true(!'--' %in% levels(countryDf$detected_country))
  expect_true('Not Detected' %in% levels(countryDf$detected_country))
})

test_that("Check if enrol clean fixed ID format to
          character", {
  expect_that(class(countryDf$learner_id), equals('character'))
})
```

These tests ensure that the enrolClean() method carries its tasks described in the Data Preparation report, ensuring:

- That fields that are not needed are removed
- The ID format is appropriate and will not cause any issues with the joins
-   - * factor level is removed in favor of Not Detected

2.2 Aggregation and Merge Functions Tests

2.2.1 Steps Aggregation

```r
week1MaxSteps = 19
week2MaxSteps = 23
week3MaxSteps = 20
stepsAggDf = stepsAgg(stepsDf)

test_that("New fields are created
          after steps aggregation", {
  expect_false(is.null(stepsAggDf$week1_completed_steps))
  expect_false(is.null(stepsAggDf$week2_completed_steps))
  expect_false(is.null(stepsAggDf$week3_completed_steps))
})

test_that("completed steps can't be more than actual
          after steps aggregation", {
  expect_that(max(stepsAggDf$week1_completed_steps),
              equals(week1MaxSteps))
  expect_that(max(stepsAggDf$week2_completed_steps),
              equals(week2MaxSteps))
  expect_that(max(stepsAggDf$week3_completed_steps),
              equals(week3MaxSteps))
})

test_that("no NAs after steps aggregation", {
    expect_that(length(stepsAggDf[is.na(stepsAggDf)]), equals(0))
})

test_that("IDs are unique after steps aggregation", {
  expect_that(length(unique(stepsAggDf$learner_id)),
              equals(length(stepsAggDf$learner_id)))
})
```

These tests ensure that the stepsAgg() method carries its tasks described in the Data Preparation report, ensuring:

- The new aggregated fields (week completed steps) are created
- Completed steps cannot somehow exceed actual steps in week
- No NAs are left after aggregation
- IDs are still unique after aggregation

2.2.2 Questions Aggregation

```r
questionsAggDf = questionsAgg(questionsDf)

test_that("New fields are created
          after questions aggregation", {
  expect_false(is.null(questionsAggDf$week1_total_marks))
  expect_false(is.null(questionsAggDf$week2_total_marks))
  expect_false(is.null(questionsAggDf$week3_total_marks))
  expect_false(is.null(questionsAggDf$week1_total_attempts))
  expect_false(is.null(questionsAggDf$week2_total_attempts))
  expect_false(is.null(questionsAggDf$week3_total_attempts))
})

test_that("No NAs after questions aggregation", {
  expect_that(
    length(questionsAggDf[is.na(questionsAggDf)]), equals(0))
})

test_that("IDs are unique after questions aggregation", {
  expect_that(length(unique(questionsAggDf$learner_id)),
              equals(length(questionsAggDf$learner_id)))
})
```

These tests ensure that the questionsAgg() method carries its tasks described in the Data Preparation report, ensuring:

- The new aggregated fields (week total marks and attempts) are created
- No NAs are left after aggregation
- IDs are still unique after aggregation

2.2.3 Merge

```r
progressByArchetypeDf = mergeDfs(archetypesDf, questionsAggDf, stepsAggDf)
countryProgressByArchetypeDf = countryMergeDfs(progressByArchetypeDf, countryDf)

test_that("No NAs after merge", {
  expect_that(
    length(progressByArchetypeDf[is.na(progressByArchetypeDf)]), equals(0))
})

test_that("IDs are unique after merge", {
  expect_that(length(unique(progressByArchetypeDf$learner_id)),
              equals(length(progressByArchetypeDf$learner_id)))
})

test_that("No NAs after country merge", {
  expect_that(
    length(countryProgressByArchetypeDf[is.na(countryProgressByArchetypeDf)]), equals(0))
})

test_that("IDs are unique country after merge", {
  expect_that(length(unique(countryProgressByArchetypeDf$learner_id)),
              equals(length(countryProgressByArchetypeDf$learner_id)))
})
```

These tests ensure that the mergeDfs() and countryMergeDfs methods carry their tasks described in the Data Preparation report, ensuring:

- No NAs are left after merge
- IDs are still unique after merge

3 Running the Tests

To run all the tests in the suite use the test.project() function after loading the Project Template Library (library(ProjectTemplate)) and loading the project (project.load()). Make sure the working directory is set to the project root and that Project Template is installed (install.packages(ProjectTemplate)).