Data Carpentry: Proposal Tests

Michelle Kelly

23 July, 2018

```
source("dataLoad.R")
```

```
# verify that an error is returned if data file does not exist in directory

# specify non-existant file path
filepath <- "./input_data/FileThatDoesn'tExist.csv"

# display output of script to user, should return detailed error message
output_dataLoad <- dataLoad(filepath)
```

```
## Error: Data file can't be found. Check that filepath is specified correctly.
```

```
# test is passed if script returns error
if (tryCatch(dataLoad(filepath), error = function(c) "Error") == "Error"){
  print("Passed")
} else {
  print("Failed")
}
```

```
## [1] "Passed"
```

```
# verify that column names of raw data are as expected

# create dummy data with an incorrect column name
dummyData <- data.frame(incorrect_colname = 10, dateTime = "1/1/2017 0:00", tz_cd = "PST",
                        Discharge_ft3s = 10, WaterTemp_C = 10, DO_mgL = 10, DOsat_pct = 100)
write.csv(dummyData, file = "./input_data/Tests_DummyData.csv", row.names = FALSE)

# specify correct file path
filepath <- "./input_data/Tests_DummyData.csv"

# display output of script to user, should return detailed error message
output_dataLoad <- dataLoad(filepath)
```

```
## Error: Check that column names are correct. Data file must contain columns labeled:
##  'site_no', 'dateTime', 'tz_cd', 'Discharge_ft3s', 'WaterTemp_C'
##  'DO_mgL', 'DOsat_pct'
```

```
# test is passed if script returns error
if (tryCatch(dataLoad(filepath), error = function(c) "Error") == "Error"){
  print("Passed")
} else {
  print("Failed")
}
```

```
## [1] "Passed"
```

```
# load dummyData (with correct column names)
dummyData <- data.frame(site_no = 10, dateTime = "1/1/2017 0:00", tz_cd = "PST",
                        Discharge_ft3s = 10, WaterTemp_C = 10, DO_mgL = 10, DOsat_pct = 100)
write.csv(dummyData, file = "./input_data/Tests_DummyData.csv", row.names = FALSE)
```

```
# process dummyData through dataLoad
output_dataLoad <- dataLoad(filepath, lat = 50, long = 100, tz = "UTC")

source("dataPrep.R")
```

```
# check that class of dateTime is POSIXct

# change class of dateTime entry from POSIXct to character
output_dataLoad$dateTime <- as.character(output_dataLoad$dateTime)

# display output of script to user, should return detailed error message
output_dataPrep <- dataPrep(output_dataLoad)
```

```
## Error: 'dateTime' must be in POSIXct format.
```

```
# test is passed if script returns error
if (tryCatch(dataPrep(output_dataLoad), error = function(c) "Error") == "Error"){
  print("Passed")
} else {
  print("Failed")
}
```

```
## [1] "Passed"
```

```
# check that columns have the correct names, because function references these names internally

# change class of output_dataLoad$dateTime back to POSIXct
output_dataLoad$dateTime <- lubridate::ymd_hm(paste(output_dataLoad$dateTime, "00:00"))

# change name of "WaterTemp_C" column
names(output_dataLoad)[names(output_dataLoad) == "WaterTemp_C"] <- "wrongName"

# display output of script to user, should return detailed error message
output_dataPrep <- dataPrep(output_dataLoad)
```

```
## Error: Check that column names are correct. Data file must contain columns labeled:
##  'Lat', 'Long', dateTime', 'Discharge_m3s', 'WaterTemp_C'
##  'DO_mgL', 'DOsat_pct'
```

```
# test is passed if script returns error
if (tryCatch(dataPrep(output_dataLoad), error = function(c) "Error") == "Error"){
  print("Passed")
} else {
  print("Failed")
}
```

```
## [1] "Passed"
```

```
# process dummyData through dataLoad and dataPrep
output_dataLoad <- dataLoad(filepath, lat = 50, long = 100, tz = "UTC")
output_dataPrep <- dataPrep(output_dataLoad)
```

```
## PAR estimated based on latitude and time.
```

```
source("metabolismModeling.R")
```

```
# return warning if na present in any row of input dataframe, this will break model
# this will catch if the data was pre-processed with dataPrep, so a secondary check of
```

```r
# correct column names is unnecessary.

# introduce NA into DO.obs column
output_dataPrep$DO.obs[1] <- NA

# display output of script to user, should return detailed error message
output_metabolismModeling <- metabolismModeling(output_dataPrep)
```

```
## Error: Data contains NA values, which break streamMetabolizer. Process data with
##  dataPrep function first, then use metabolismModeling.
```

```r
# test is passed if script returns error
if (tryCatch(metabolismModeling(output_dataPrep), error = function(c) "Error") == "Error"){
  print("Passed")
} else {
  print("Failed")
}
```

```
## [1] "Passed"
```