CourseKata Class Data

CourseKata Data Download README

This file is included in the post-class data download of class data for a CourseKata class.

If you are reading this as a PDF or on GitHub, this file was created from the RMarkdown notebook `README.Rmd`. If you open that file in RStudio you will be able to run and interact with the code blocks below.

All of the code blocks are also included in the `process_data.R` file that was included in the data download zip folder. If you already know how the data is formatted and what the purpose of this code is for, then you can probably just use that processing script.

Happy reading!

Required Packages and Helper Functions

To use the code in these examples you will need a handful of useful packages. The following code loads those packages and some helper functions, and below the code you can find a description of why each package is needed.

Package	Description
dplyr	Exposes the pipe (`%>%`) and has a bunch of functions that simplify table manipulation
tidyr	Has functions for tidying data
readr	Simplifies and standardises file reading functions
stringr	Simplifies and standardises string manipulation functions
purrr	Makes working with loops much easier
future	Allows for multicore processing for parallel computing
furrr	Parallelized `purrr` functions
lubridate	Makes working with dates easier
jsonlite	Parses JSON string objects into lists
pacman	Simplifies requiring, installing, and loading packages

Here is the code to install and load them, along with some helper functions that will be useful later:

```
# load required packages, installing if needed
if (!require("pacman", quietly = TRUE)) install.packages("pacman")
pacman::p_load(
  jsonlite, dplyr, tidyverse, readr, purrr, stringr, lubridate,
  future, furrr
)

# setup the future/furrr package
future::plan("future::multiprocess")

# define some helper functions that will be useful later
json_to_list <- function(x) {
  if (is.null(x) | is.na(x)) list() else parse_json(x)
}

# read in dates as UTC timezone and then convert to system timezone
convert_date_string <- function(x) {
  ymd_hms(x, tz = "UTC") %>% with_tz(Sys.timezone())
```

```
}
```

Loading Class Data

Data from the class comes in four tables with each table in its own file. Below is the code to load each table, and a description of the contents of each table.

Response Data

Description

The response data is organized into a table with a variable number of columns (depending on `lrn_option_<n>`, see column description below) and a number of rows equivalent to the number of responses made to questions in the course. This table will likely be very large (200 students will yield around 300,000 responses), so in the next section (**Working With Class Data**) we will break things in to useable parts. Here is a description of each variable in the responses table:

Column	Description
class_id	a unique identifier for this particular class
course_name	the GitHub repository for the course
release	the GitHub branch for the course
student_id	a unique identifier for each student on CourseKata
item_id	a unique identifier for this particular question
item_type	whether this is a *learnosity* or *datacamp* item
chapter	the chapter that the item appears in
page	the page that the item appears on
prompt	the question prompt for this response
response	the value of the response: either the value (for shorttext, plaintext, ratings, datacamp, etc.) or an array of numbers indicating the position of the multiple choice answers chosen and which correspond to the columns `lrn_option_<number>`.
points_possible	the number of points possible if the completely correct answer is given
points_earned	the number of points earned
dt_submitted	a datetime object indicating when the response was submitted (timezone: GMT/UTC)
attempt	the number of times the question has been attempted, including the current attempt
user_agent	the browser user agent string for the user (see for details: TODO)
lrn_session_id	the unique ID for this user session on Learnosity
lrn_response_id	the unique ID for this particular response on Learnosity
lrn_items_api_version	the version of the Learnosity Items API for this item
lrn_response_api_version	the version of the Learnosity Response API for this response
lrn_activity_id	the unique ID for the activity on Learnosity
lrn_question_reference	the unique ID for the question on Learnosity
lrn_question_position	for multi-question items, the position of the question in the item
lrn_type	the Learnosity type of the question (e.g. mcq, shorttext, plaintext, rating, etc.)
lrn_dt_started	a datetime object indicating when the responses was started (timezone: GMT/UTC)
lrn_dt_saved	a datetime object indicating when the responses was saved (timezone: GMT/UTC)
lrn_status	the status of the question response on Learnosity (e.g. “Completed”)

Column	Description
lrn__option__	for multiple choice questions (`lrn_type`: "mcq"), the value of the option at position ***n***
lrn__response__json	the fully-detailed JSON response object

For more about the distinction between Learnosity Activities, Items, and Questions, see the documentation at https://authorguide.learnosity.com/hc/en-us.

Code to Load

```
# for responses, the number of columns is variable so we don't know the types
# readr will try to guess the types, but col_guess() is unreliable
# instead just read everything in as a character column and then convert after
responses <- "responses.csv" %>%
  read_csv(col_types = cols(.default = col_character())) %>%
  mutate_at(vars(attempt, lrn_question_position), parse_integer) %>%
  mutate_at(vars(points_possible, points_earned), parse_number) %>%
  # read in dates as UTC timezone and then convert to system timezone
  mutate_at(vars(matches("^(?:lrn_)?dt_")), convert_date_string) %>%
  # convert response JSON to a nested list-column then move it to the end
  mutate(lrn_response_json = map(lrn_response_json, json_to_list)) %>%
  select(-lrn_response_json, lrn_response_json) %>%
  drop_na(student_id, prompt, response)
```

Page Views

Description

The page view data comes in a table of student-page-datetime cases. That is, there is a row for every time a student accessed a page in the course. The table has five variables:

Column	Description
class__id	a unique identifier for this particular class
student__id	a unique identifier for each student on CourseKata
chapter	the chapter the page view occurred in
page	the page that was viewed
dt__accessed	a datetime object indicating when the page was accessed (timezone: GMT/UTC)

Code to Load

```
page_views <- "page-views.csv" %>%
  read_csv(col_types = cols(.default = col_character())) %>%
  # read in dates as UTC timezone and then convert to system timezone
  mutate(dt_accessed = convert_date_string(dt_accessed))
```

Items

Items are organized in a table where each row represents all of the data for a particular question in the class. There are 18 columns in the table, though they will never *all* be relevant for a particular item. Columns prefixed with "dcl_" are only filled for DataCamp-Light items (the R-sandboxes) and columns prefixed with "lrn_" are only filled for Learnosity items. Here are descriptions of the columns in the table:

Column	Description
class_id	a unique identifier for this particular class
item_id	a unique identifier for this particular question
item_type	whether this is a **learnosity** or **datacamp** item
chapter	the chapter that the item appears in
page	the page that the item appears on
dcl_pre_exercise_code	the code run invisibly to set up the module
dcl_sample_code	the code in the module when it first loads
dcl_solution	the code that appears in the solution tab
dcl_sct	the solution checking code (see the `testwhat` package for details)
dcl_hint	the text that appears in the hint box
lrn_activity_reference	the unique ID for the activity on Learnosity
lrn_question_reference	the unique ID for the question on Learnosity
lrn_question_position	for multi-question items, the position of the question in the item
lrn_template_name	the template used to create the item
lrn_template_reference	a unique ID for the item template on Learnosity
lrn_item_status	the status of the item on Learnosity (e.g. "published")
lrn_question_data	the fully-detailed JSON object that sets up the item

For more about the distinction between Learnosity Activities, Items, and Questions, see the documentation at https://authorguide.learnosity.com/hc/en-us.

Code to Load

```
items <- "items.csv" %>%
  read_csv(col_types = cols(.default = col_character())) %>%
  mutate(
    lrn_question_position = parse_integer(lrn_question_position),
    lrn_question_data = map(lrn_question_data, json_to_list)
  )
```

Tags

Description

Tags are not currently utilized heavily within CourseKata but may have a larger role in the future (e.g. tagging specific learning outcomes). For completeness, the table is described here, but it will likely not be of much use.

The tags table is organized at the item-tag level where each tag for each item has its own row. There are three columns in the table:

Column	Description
item_id	a unique identifier for this particular question
tag	the tag given to this item
tag_type	the hierarchical parent tag for this tag (e.g. "Chapter" holds all of the chapter name tags)

Code to Load

```
tags <- "tags.csv" %>%
  read_csv(col_types = cols(.default = col_character()))
```

Working With Class Data

Filling Out Multiple Choice Data

You might have noticed that the responses with a `lrn_type` of "mcq" all have response values that look like `["0"]` or `["1", "3"]`. These values correspond to the `lrn_option_<n>` columns, where `["0"]` would correspond to the value of `lrn_option_0` and `["1", "3"]` would correspond to both the value in `lrn_option_1` and `lrn_option_3`. Before we split up the table into parts, we will map out those values so that `response` is filled with the actual value from that response option.

```
# NOTE: this code chunk can take a couple minutes to run

lookup_table <- responses %>%
  filter(lrn_type == "mcq") %>%
  select(reference = lrn_question_reference, starts_with("lrn_option_")) %>%
  distinct()

lookup_response <- function(response, reference, lookup_table) {
  if (response == "[]" | is.na(response)) {
    return(NA)
  }
  if (!str_detect(response, '\\[(?:"\\d+",? ?)\\]')) {
    return(response)
  }
  if (!reference %in% lookup_table$reference) {
    return(response)
  }

  option_nums <- response %>%
    str_split(",", simplify = TRUE) %>%
    parse_number() %>%
    as.integer()

  lookup_table[lookup_table$reference == reference, option_nums + 2] %>%
    as.character() %>%
    paste0(collapse = "; ")
}

responses <- responses %>%
  mutate(response = future_map2_chr(
    response, lrn_question_reference,
    lookup_response,
    lookup_table = lookup_table
  ))

# clean-up
rm(lookup_table, lookup_response)
```

Split the Response Data

Though all of the responses have been read into the `responses` table, there are a lot of different types of responses lumped into the one file. One thing that might helps is to make some semantic splits into the pre- and post-survey items, in-text items, and practice quizzes. Since the pre-/post-survey items are (generally) static trait questions like demographics, those can go into a wide `students` table, and the others into a long `textbook_items` table and a long `practice_quizzes` table.

```
students <- responses %>%
  filter(str_detect(item_id, "Student_Survey_Pre|Post-Survey")) %>%
  drop_na(student_id, prompt, response) %>%
  mutate(prompt = ifelse(
    str_detect(item_id, "Student_Survey"),
    paste("pre:", prompt),
    paste("post:", prompt)
  )) %>%
  # only keep most recent response
  arrange(desc(dt_submitted)) %>%
  distinct(student_id, prompt, .keep_all = TRUE) %>%
  # spread each prompt to its own column
  select(
    -starts_with("lrn_"), -attempt, -release, -dt_submitted,
    -points_possible, -points_earned,
    -chapter, -page, -item_id, -item_type, -user_agent
  ) %>%
  spread(prompt, response) %>%
  # rearrange the columns
  select(
    class_id, course_name, branch, student_id,
    starts_with("pre: "), starts_with("post: ")
  )

textbook_items <- responses %>%
  filter(!str_detect(item_id, "Student_Survey|Post-Survey|Practice Quiz")) %>%
  drop_na(prompt, response)

practice_quizzes <- responses %>%
  filter(str_detect(page, "Practice Quiz"))
```

Saving Data Images

It takes a while to process all of the class data and get it into the tables you feel comfortable working with. Instead of having to run this file or these code chunks every time we want to work with the processed data, we can save the processed data to a *.Rds* file. Further, `readr::write_rds()` makes it easy to compress the files so that they don't take up as much disk space.

```
write_rds(items, "items.Rds", compress = "gz")
write_rds(page_views, "page_views.Rds", compress = "gz")
write_rds(tags, "tags.Rds", compress = "gz")
write_rds(responses, "responses.Rds", compress = "gz")
write_rds(students, "students.Rds", compress = "gz")
write_rds(textbook_items, "textbook_items.Rds", compress = "gz")
write_rds(practice_quizzes, "practice_quizzes.Rds", compress = "gz")
```

Reading the files in again is as simple as using the complement function `readr::read_rds()`. This code block will read in a second copy of the `items` table and then compare to the one we already have loaded.

```
items2 <- read_rds("items.Rds")
identical(items, items2)
```

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