PAELLA:

Personalized student **A**ctivation in **E**ngineering-education: **L**everaging **L**earning **A**nalytics for an engaging blended learning course design

PerActiLA:
Personalized Student Activation through Learning Analytics-based insights about students' learning processes

Progress Report R2

Extracting indicators from Canvas data

Version 1: course 1 in Quarter 4

19.09.2022

Eindhoven University of Technology

Authors: Dr. Rianne Conijn & drs. M.Sc. Sonja Kleter

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Introduction

In the PerActiLA project, we aim at providing helpful interventions to students in ongoing courses who have been identified as having a high likelihood to fail the course. In report R1, we describe more details and a motivation of this approach. The identification of students, as well as the evaluation of the effects of the intervention, make use of students' Learning Analytics, that is the clickstream data of the TU/e Learning Management System (LMS) Canvas.

In this report, we show how to create a list of Canvas indicators (variables) that describe students' use of the LMS. Some of these variables, especially those that make use of Canvas data in the first few weeks of the course, can be of value for the creation of predictive models that identify students who may fail the course. Other variables, especially those that make use of Canvas data in later weeks of the course, may be of use to examine whether the intervention has led to a change in the students' use of the LMS. Of course, for a comprehensive analysis of the effects of the intervention, other data should be used in addition.

In the following, we provide the complete R script that can be used to pre-process the raw Canvas data tables into data sets that researchers or teachers can analyze in any software that allows importing .csv files. The script leads to the generation of the following Canvas variables:

General course indicators

- time between course publication and first log in-session
- time between course publication and first opening of study guide
- time between course publication and first opening of schedule
- time between course publication and first opening of course information page
- time between last lecture day and last log in-session (last Friday of course (23.59).

Session indicators

- The number of sessions
- The number of clicks per session
- The total session time in minutes
- The average session time in seconds
- The standard deviation of session times in seconds
- The average time between two sessions in seconds
- The standard deviation of the time between two sessions in seconds
- The maximum time between two sessions in seconds
- The average start time of session

Assignment indicators

- Number of clicks in assignments
- Number of unique submitted assignments
- Number of unique submitted assignments after deadline (submissions submitted after the due_date)

All assignment indicators are calculated over the full course and per week of the course.

Quiz indicators

- number of clicks in a quiz
- number of unique submitted practice quizzes (quiz_type = practice_quiz/survey OR time_limit = -1)
- number of unique submitted exam-relevant quizzes (quiz_type = assignment/graded_survey)
- number of unique quizzes attempted more than once (using quizzes that allowed more than one attempt)
- mean percentage of time taken exam-relevant quiz (divided by total allowed time)
- mean time taken practice quiz (min)
- mean percentage of correct quiz answers (for most successful attempts; note that multiple points may be earned per question)

All quiz indicators are calculated over the full course and per week of the course.

Discussion forum indicators

- Number of forum clicks
- Number of announcement clicks
- Number of forum topics posts
- Number of forum reply posts

File and video indicators

- number of clicks on files
- number of unique file accessed (including inline views & downloads)
- number of unique file downloads (via 'download' in url requests)
- number of clicks on video files (mediafiles directly uploaded on canvas with mp4 type of extension (file_type in file_dim))
- number of unique accessed video files
- number of clicks on livestreamed videos (media accessed via bigbluebutton, Canvas conferences, or panopto)
- number of unique accessed livestreamed videos (canvas conferences)

All file indicators are calculated over the full course and per week of the course.

This script should be useful to others who are interested in preparing Canvas click stream data for further data analyses. As it is not always useful to post code in a plain pdf- or word-document, we also made an HTML version of it that can be found here: https://rpubs.com/RianneConijn/CanvasIndicators

Preparing the data

The following describes the code for extracting learning analytics indicators from data obtained from the learning management system Canvas. The dataset can be retrieved from the Canvas API. This dataset consists of a total of > 100 tables. An overview of all the tables can be found at https://portal.inshosteddata.com/docs. In the following, we assume that the data are stored as .parquet-files. Accordingly, the package arrow is needed to load these data. To transform the data, two packages are used: tidyverse and lubridate.

```
library(tidyverse)
library(lubridate)
library(arrow)
options(scipen=999)
path <- "your_path_here"
```

Loading the requests table

For the most part, we will work with the requests-table. This table includes all application server requests, which can be seen as the raw clickstreams of students interacting with the Canvas learning management system. In this case, we will work with a requests table made from a subset of courses in the fourth quartile of the academic year 2021-2022, which ended at June 24th, 2022.

```
requests <- read_parquet(paste0(path, "requests0.snappy.parquet"),
col_types = cols(user_id = col_character(),
course_id = col_character(),
discussion_id = col_character(),
assignment_id = col_character(),
quiz_id = col_character()),
as_tibble = TRUE) %>%
na_if("\\N")
quartile <- 4
year <- 21
endcourse <- "2021-06-24 23:59:99"
```

Filtering the requests table

Now, we will first filter the requests table based on the information that we need. Some possible filters include:

- Removing non-clicks (server pings)
- Filtering a subset of courses
- Filtering a subset of users
- Filtering clicks made in a specific time range (e.g., two weeks before the course start up to two weeks after the final exam)

For the latter, additional information is needed about the course structure in the university, to identify the lecture and exam weeks. The yearly week numbers need be to manually mapped to the week numbers as used in the university's academic calendar.
Below, you can find an example function that identifies the week labels according to the academic calendar: calculate_weeklabel(). Here the week labels are defined

as follows: w0 indicates the two weeks prior to the course, w1 indicates the first lecture week, w2 indicates the second lecture week (and so on), ew1 indicates the first exam week, and ew3 indicates the two weeks after the exam period. The suffix 'a' indicates a holiday week (e.g., Christmas holiday). For example, w3a, is a break after the 3rd lecture week.

```
# weeklabels checked for Q3/Q4 2019-2020 and Q1/Q2/Q3/Q4 2020-2021 and Q4 2021-2022
calculate_weeklabel <- function(weekno, quartile, year){
weeklabel =
case_when(((weekno == "15" & quartile == "3") |
(weekno == "26" & quartile == "4") |
(weekno == "44" & quartile == "1") |
(weekno == "03" & quartile == "2")) ~ "ew1",
((weekno == "16" & quartile == "3") |
(weekno == "27" & quartile == "4") |
(weekno == "45" & quartile == "1") |
(weekno == "04" & quartile == "2")) ~ "ew2",
((weekno %in% c("17", "18") & quartile == "3") |
(weekno %in% c("28", "29") & quartile == "4") |
(weekno %in% c("46", "47") & quartile == "1") |
(weekno %in% c("05", "06") & quartile == "2")) ~ "ew3",
((weekno %in% c("04", "05") & quartile == "3") |
(weekno %in% c("15", "16") & quartile == "4") |
(weekno %in% c("34", "35") & quartile == "1") |
(weekno %in% c("44", "45") & quartile == "2")) ~ "w0",
((weekno == "06" & quartile == "3") |
(weekno == "17" & quartile == "4") |
(weekno == "36" & quartile == "1") |
(weekno == "46" & quartile == "2")) ~ "w1",
((weekno == "07" & quartile == "3") |
(weekno == "18" & quartile == "4") |
(weekno == "37" & quartile == "1") |
(weekno == "47" & quartile == "2")) ~ "w2",
((weekno == "08" & quartile == "3" & year == 19) |
(weekno == "09" & quartile == "3" & year == 20) |
(weekno == "19" & quartile == "4") |
(weekno == "38" & quartile == "1") |
(weekno == "48" & quartile == "2")) ~ "w3",
((weekno == "09" & quartile == "3" & year == 19)) ~ "w3a",
((weekno == "08" & quartile == "3" & year == 20)) ~ "w2a",
((weekno == "10" & quartile == "3") |
(weekno == "20" & quartile == "4") |
(weekno == "39" & quartile == "1") |
(weekno == "49" & quartile == "2")) ~ "w4",
((weekno == "11" & quartile == "3") |
(weekno == "21" & quartile == "4") |
(weekno == "40" & quartile == "1") |
(weekno == "50" & quartile == "2")) ~"w5",
((weekno == "12" & quartile == "3") |
(weekno == "22" & quartile == "4") |
(weekno == "41" & quartile == "1") |
(weekno == "51" & quartile == "2")) ~ "w6",
((weekno %in% c("52", "53") & quartile == "2")) ~ "w6a",
((weekno == "13" & quartile == "3") |
(weekno == "23" & quartile == "4") |
(weekno == "42" & quartile == "1") |
```

```
(weekno == "01" & quartile == "2")) ~ "w7",
((weekno == "14" & quartile == "3") |
(weekno == "24" & quartile == "4") |
(weekno == "43" & quartile == "1") |
(weekno == "02" & quartile == "2")) ~ "w8",
((weekno == "25" & quartile == "4")) ~ "w9")
}
```

Below, we filter the requests table in two ways:
- Removing clicks outside course pages
- Removing non-clicks (server pings)
- Filtering clicks made in a specific time range (e.g., two weeks before the course start up to two weeks after the final exam)

```
requests1 <- requests %>%
#remove clicks outside course pages
filter(!is.na(course_id)) %>%
group_by(course_id, user_id) %>%
#remove non-clicks (server pings)
filter(web_application_action != "ping") %>%
mutate(
timestamp = as.POSIXlt(timestamp),
quartile = quartile,
) %>%
arrange(timestamp, .by_group = TRUE) %>%
mutate(
weekno = strftime(timestamp, "%V"),
weeklabel = calculate_weeklabel(weekno, quartile, year)) %>%
# remove clicks outside timeframe
filter(!is.na(weeklabel))
```

Extending the requests table

The next step is to extend the clickstream data by adding add information on the session in which a server request was made. Within the requests-table there is already an indicator for session (session_id). Here, a new session is created when the user is prompted to log in to the learning management system again. However, with single-sign-on, there are usually only a few sessions, which might stretch over a couple of weeks. Accordingly, we define a learning session as follows:
A sequence of activities from a single user within the learning management system, without the user being inactive for more than 30 minutes.

```
requests2 <- requests1 %>%
group_by(course_id, user_id) %>%
mutate(
diff_time = timestamp - lag(timestamp),
# 30 minutes of idle time starts a new session
session_start = (diff_time >= 1800),
session_start = ifelse(row_number() == 1, 1,
ifelse(is.na(session_start), FALSE,
session_start)),
session_no = cumsum(session_start),
time = strftime(timestamp,"%H:%M:%S"))
```

In addition, we want to add information about the user who made the request. Here we specifically want to add

whether the user was enrolled as a teacher or student in the course, to be able to distinguish between student
and teacher clicks. The information about student enrollments can be found in the enrollment_dim-table.

```
# check who is enrolled as student
enrollment_dim <- read_parquet(paste0(path, "enrollment_dim.snappy.parquet"),
as_tibble = TRUE, col_types = cols(.default = "c")) %>%
select(course_id, user_id, type) %>%
distinct() %>%
group_by(course_id, user_id) %>%
summarize(enrollment_type = paste(type, collapse = ",")) %>%
ungroup()
# add enrollment to requests table
requests3 <- requests2 %>%
left_join(enrollment_dim, by = c("user_id", "course_id") )
```

Write the final requests file

Finally, we remove redundant columns and write the filtered and extended requests table to a new file.

```
#remove redundant columns
requests4 <- requests3 %>%
select(timestamp, user_id, course_id, quiz_id,
discussion_id, assignment_id, url,
web_application_controller,
web_application_action, session_id,
quartile:enrollment_type)
write.csv(requests4, paste0(path, "requests_ext.csv"), row.names = FALSE)
```

The data is now fully prepared and we can focus on extracting the indicators from the Canvas data. The feature extraction can be done per category, depending on the type of learning activities available in the course.

Creating session indicators

In this part, we extract indicators to the general log in behavior in the learning management system. This includes several summary statistics for the different sessions the users had during the course. A session is defined in the extended log file (see #Preparing the data). The function below extracts the following session indicators, and may be customized depending on the researcher's needs:

More general course indicators:
- time between course publication and first log in-session
- time between course publication and first opening of study guide
- time between course publication and first opening of schedule
- time between course publication and first opening of course information page
- time between last lecture day and last log in-session (last Friday of course (23.59). Note negative values indicate that the course was not accessed after the last lecture day.)

Session indicators:
- The number of sessions
- The number of clicks per session

- The total session time in minutes
- The average session time in seconds
- The standard deviation of session times in seconds
- The average time between two sessions in seconds
- The standard deviation of the time between two sessions in seconds
- The maximum time between two sessions in seconds
- The average start time of session

All session indicators are calculated over the full course, per week, and for the first half vs. second half of the course.

```
getsession_info <- function(requests_df, quartile, year, outfile){
# calculate summary statistics per session
sessioninfo <- requests_df %>%
group_by(course_id) %>%
mutate(
start_course = min(timestamp),
end_course = endcourse) %>%
group_by(course_id, user_id, session_no) %>%
summarize(
n_clicks = n(),
start_course = first(start_course),
end_course = first(end_course),
firsttime = (hour(first(timestamp)) + minute(first(timestamp))/60) - 6,
firsttime = ifelse(firsttime < 6, firsttime + 18, firsttime - 6),
# change first time of the day to numerical, shifted to start at 6 am
# (6:00 = 0, 6:30 = 0.50, 22:00 = 16, 02:00 = 20)
starttime = min(timestamp),
endttime = max(timestamp),
totaltime = endttime - starttime,
totaltime_min = ifelse(!is.na(first(totaltime)), first(totaltime)/60, NA),
interval_time = first(diff_time),
schedule_time = first(timestamp[grepl("schedule", url)]),
courseinfo_time = first(timestamp[grepl("course-information", url)]),
studyguide_time = first(timestamp[web_application_action == "syllabus"])) %>%
ungroup() %>%
# at least 2 clicks per session
filter(n_clicks >= 2)
# Summarize the session statistics over the full course
session_sum <- sessioninfo %>%
group_by(course_id, user_id) %>%
summarize(
time_to_first_login = as.numeric(first(starttime) -
first(start_course)),
time_to_first_schedule = as.numeric(first(schedule_time) -
first(start_course)),
time_to_first_courseinfo = as.numeric(first(courseinfo_time) -
first(start_course)),
time_to_first_studyguide = as.numeric(first(studyguide_time) -
first(start_course)),
time_to_last_login = as.numeric(last(starttime) -
first(as.POSIXct(end_course))),
n_clicks = sum(n_clicks, na.rm = T),
n_sessions = n(),
totalsessiontime_min = sum(totaltime_min, na.rm = T),
m_sessiontime = as.numeric(mean(totaltime, na.rm = T)),
```

```
sd_sessiontime = sd(totaltime, na.rm = T),
m_intervaltime = mean(interval_time, na.rm = T),
sd_intervaltime = sd(interval_time, na.rm = T),
max_intervaltime = max(interval_time, na.rm = T),
# map all times to one day, to
m_starttime = mean(firsttime, na.rm = T)
) %>%
ungroup()
# calculate summary statistics per session per week
session_weekinfo <- requests_df %>%
group_by(course_id, user_id, weeklabel, session_no) %>%
summarize(
n_clicks = n(),
firsttime = (hour(first(timestamp)) + minute(first(timestamp))/60) - 6,
firsttime = ifelse(firsttime < 6, firsttime + 18, firsttime - 6),
starttime = min(timestamp),
endttime = max(timestamp),
totaltime = endttime - starttime,
totaltime_min = ifelse(!is.na(totaltime), totaltime/60, NA),
interval_time = first(diff_time)) %>%
ungroup() %>%
# at least 2 clicks per session
filter(n_clicks >= 2)
# Summarize the weekly session statistics over the full course
session_sumweek <- session_weekinfo %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_clicks = sum(n_clicks, na.rm = T),
n_sessions = n(),
totalsessiontime_min = sum(totaltime_min, na.rm = T),
m_sessiontime = as.numeric(mean(totaltime, na.rm = T)),
sd_sessiontime = sd(totaltime, na.rm = T),
m_intervaltime = mean(interval_time, na.rm = T),
sd_intervaltime = sd(interval_time, na.rm = T),
max_intervaltime = max(interval_time, na.rm = T),
m_starttime = mean(firsttime, na.rm = T)
) %>%
ungroup()
# sum per first vs. second half of the course
session_sumweekhalf <- session_weekinfo %>%
mutate(
half = case_when(
(weeklabel %in% c("w1", "w2", "w3", "w4", "w3a") |
(weeklabel %in% c("w5") & quartile == 4)) ~ "half1",
(weeklabel %in% c("w5", "w6", "w7", "w8", "w9", "w6a")) ~ "half2")) %>%
filter(!is.na(half)) %>%
group_by(course_id, user_id, half) %>%
summarize(
sd_sessiontime = sd(totaltime, na.rm = T),
sd_intervaltime = sd(interval_time, na.rm = T)
) %>%
pivot_wider(id_col = c(course_id, user_id),
names_from = half,
values_from = c(sd_sessiontime, sd_intervaltime))
# convert to datawide and merge indicators
session_weekwide <- session_sumweek %>%
pivot_wider(id_col = c(course_id, user_id), names_from = weeklabel,
```

```
values_from = c(n_clicks, n_sessions, totalsessiontime_min,
m_sessiontime, sd_sessiontime, m_intervaltime,
sd_intervaltime,
max_intervaltime, m_starttime)) %>%
left_join(session_sum) %>%
left_join(session_sumweekhalf) %>%
mutate_at(vars(starts_with("n_"),
starts_with("m_sessiontime"),
starts_with("totalsessiontime_min")),
~replace_na(., 0))
# save session indicators to a separate file
write.csv(session_weekwide, paste0(path, outfile),
row.names = FALSE)
session_weekwide
}
```

Creating assignment indicators

In this part, we extract indicators which relate to the assignments in the learning management system. For this, information is used from a variety of tables which store information about assignments in Canvas. Specifically, we use the assignment_dim and assignment_fact tables, which contain information on assignment characteristics such as its name, deadline and date published. In addition, we use the assignment_submission_fact and the assignment_submission_dim tables, as these contain specific information on student submissions for a specific assignment.

The function below extracts the following assignment indicators, and may be customized depending on the researcher's needs:

- Number of clicks in assignments
- Number of unique submitted assignments
- Number of unique submitted assignments after deadline (submissions submitted after the due_date)

All assignment indicators are calculated over the full course and per week of the course.

```
getassignment_info <- function(requests_df, quartile, year, outfile){
requests4 <- requests_df
# load assignments
assign_dim <- read_parquet(paste0(path, "assignment_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
assign_fact <- read_parquet(paste0(path, "assignment_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
assign_submission_fact <- read_parquet(paste0(path,
"submission_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
```

```
na_if("\\N")
assign_submission_fact2 <- assign_submission_fact %>%
#remove quizzes & wiki submissions
filter(!is.na(quiz_id) | !is.na(wiki_id)) %>%
mutate_at(c("score"), as.numeric)
assign_submission_dim <- read_parquet(paste0(path,
"submission_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
####### transform & merge tables
##########################################
# merge assignment_fact and assignment_dim
assign_fact_dim <- assign_dim %>%
# remove unpublished assignments
filter(workflow_state != "unpublished") %>%
left_join(assign_fact,by = c("id" = "assignment_id", "course_id",
"points_possible")) %>%
select(-updated_at, -created_at, -workflow_state)
# filter only interaction with assignments from requests
requests_assignment <- requests4 %>%
filter(!is.na(assignment_id)) %>%
mutate(assignment_id = as.numeric(assignment_id),
# get canvas_id from URL in requests
assignment_canvas_id = gsub(".*/assignments/","", url),
assignment_canvas_id = as.numeric(gsub("/.*", "",
assignment_canvas_id)))
# merge assignment submission dim and fact
assign_subm_fact_dim <- assign_submission_dim %>%
# only submissions after 2019
filter(!is.na(submitted_at), submitted_at > "2020") %>%
left_join(assign_submission_fact2, by = c("assignment_id",
"id" = "submission_id",
"user_id")) %>%
rename("submission_id" = "id")
# merge submissions with assignment fact dim table
assignment_submission <- assign_subm_fact_dim %>%
select(-canvas_id) %>%
left_join(assign_fact_dim, by = c("assignment_id"= "id","course_id"))
# extract courses and add weeklabels
assignment_submission2 <- assignment_submission %>%
mutate(date_submission = as.POSIXlt(submitted_at),
quartile = quartile,
weekno = strftime(date_submission,"%V"),
weeklabel = calculate_weeklabel(weekno, quartile, year))
# remove clicks outside timeframe & filter courses
assignment_submission3 <- assignment_submission2 %>%
filter(!is.na(weeklabel), course_id %in% requests4$course_id)
####### Summarize assignment info
##########################################
### assignment CLICKS ONLY (via requests)
# sum over full course
assignment_sum <- requests_assignment %>%
group_by(course_id, user_id) %>%
summarize(
n_assignmentclicks = n()
) %>%
```

```
ungroup()
# sum per week
assignment_sumweek <- requests_assignment %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_assignmentclicks = n()
) %>%
ungroup()
### assignment SUBMISSION INFO
# create one submission entry per (unique) assignment per user per course
assignment_submission4 <- assignment_submission3 %>%
group_by(course_id, assignment_id, user_id) %>%
arrange(as.POSIXct(submitted_at)) %>%
summarize(
firstattempt_date = min(submitted_at),
weeklabel = first(weeklabel), # weeklabel of first attempt
lastattempt_date = last(submitted_at),
late = ifelse(!is.na(due_at), firstattempt_date > first(due_at), 0)
) %>%
ungroup()
# summarize assignment indicators for full course
assignment_submission4_sum <- assignment_submission4 %>%
group_by(course_id, user_id) %>%
summarize(
n_assignment = n(),
n_late_assignment = sum(late)
) %>% ungroup()
# summarize assignment indicators per week
assignment_submission4_sumweek <- assignment_submission4 %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_assignment = n(),
n_late_assignment = sum(late)
) %>% ungroup()
# convert to datawide
assignment_weekwide <- assignment_submission4_sumweek %>%
full_join(assignment_sumweek) %>%
pivot_wider(id_col = c(course_id, user_id), names_from = weeklabel,
values_from = c(n_assignment, n_late_assignment,
n_assignmentclicks)) %>%
left_join(assignment_submission4_sum, by = c("course_id", "user_id")) %>%
left_join(assignment_sum, by = c("course_id", "user_id")) %>%
mutate_at(vars(starts_with("n_")), ~replace_na(., 0))
write.csv(assignment_weekwide, paste0(path, outfile),
row.names = FALSE)
assignment_weekwide
}
```

Creating quiz indicators

In this part, we extract indicators which relate to the quizzes in the learning management system. For this, information is used from a variety of tables which store information about assignments in Canvas. Specifically, we use the quiz_dim and quiz_fact tables, which contain information on quiz characteristics such as its

name, and points possible. In addition, we use the quiz_submission_dim and the quiz_submission_fact tables, as these contain specific information on student submissions for a specific quiz.

The function below extracts the following quiz indicators, and may be customized depending on the researcher's needs:

• number of clicks in a quiz
• number of unique submitted practice quizzes (quiz_type = practice_quiz/survey OR time_limit = -1)
• number of unique submitted exam-relevant quizzes (quiz_type = assignment/graded_survey)
• number of unique quizzes attempted more than once (using quizzes that allowed more than one attempt)
• mean percentage of time taken exam-relevant quiz (divided by total allowed time)
• mean time taken practice quiz (min)
• mean percentage of correct quiz answers (for most successful attempts; note that multiple points may be earned per question)
All quiz indicators are calculated over the full course and per week of the course.

```
getquiz_info <- function(requests_df, quartile, year, outfile){
requests4 <- requests_df
# load quizzes
quiz_dim <- read_parquet(paste0(path, "quiz_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
quiz_fact <- read_parquet(paste0(path, "quiz_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
quiz_submission_fact <- read_parquet(paste0(path,
"quiz_submission_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N") %>%
mutate_at(c("total_attempts", "score", "quiz_points_possible"), as.numeric)
quiz_submission_dim <- read_parquet(paste0(path,
"quiz_submission_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
####### transform & merge tables
##################################################
# merge quiz_fact and quiz_dim
quiz_fact_dim <- quiz_dim %>%
# remove unpublished quizzes
filter(workflow_state != "unpublished") %>%
left_join(quiz_fact,by = c("id" = "quiz_id", "course_id","assignment_id",
"points_possible")) %>%
select(-updated_at, -created_at, -due_at, -workflow_state)
# filter only interaction with quizzes from requests
requests_quiz <- requests4 %>%
```

```
filter(!is.na(quiz_id)) %>%
mutate(quiz_id = as.numeric(quiz_id),
# get canvas_id from URL in requests
quiz_canvas_id = gsub(".*/quizzes/","", url),
quiz_canvas_id = as.numeric(gsub("/.*", "", quiz_canvas_id)))
# merge quiz submission dim and fact
quiz_subm_fact_dim <- quiz_submission_fact %>%
filter(!is.na(date)) %>%
select(-enrollment_term_id, -course_account_id, -assignment_id,
-enrollment_rollup_id, -fudge_points) %>%
right_join(quiz_submission_dim, by = c("quiz_id",
"quiz_submission_id" = "id",
"user_id","submission_id"))
# merge submissions with quiz fact dim table
quiz_submission <- quiz_subm_fact_dim %>%
select(-canvas_id) %>%
left_join(quiz_fact_dim, by = c("quiz_id"= "id","course_id"))
# extract courses and add weeklabels
quiz_submission2 <- quiz_submission %>%
mutate(date_submission = as.POSIXlt(date),
quartile = quartile,
weekno = strftime(date_submission,"%V"),
weeklabel = calculate_weeklabel(weekno, quartile, year))
# remove clicks outside timeframe & filter courses
quiz_submission3 <- quiz_submission2 %>%
filter(!is.na(weeklabel), course_id %in% requests4$course_id)
####### Summarize quiz info
#############################################################
### QUIZ CLICKS ONLY (via requests)
# sum over full course
quiz_sum <- requests_quiz %>%
group_by(course_id, user_id) %>%
summarize(
n_quizclicks = n()
) %>%
ungroup()
# sum per week
quiz_sumweek <- requests_quiz %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_quizclicks = n()
) %>%
ungroup()
### ALL other quiz indicators via quiz_submissions
### QUIZ SUBMISSION INFO
# get max score per quiz
## Note: points_possible is often 0, while score > 0
## max score can be higher than max points possible (and vice versa)
## so we take the highest score of both
quiz_submission_sum <- quiz_submission3 %>%
group_by(course_id, quiz_id) %>%
summarize(
max_quiz_score = max(quiz_points_possible, score)
) %>%
ungroup()
# create one submission entry per (unique) quiz per user per course
quiz_submission4 <- quiz_submission3 %>%
```

```
left_join(quiz_submission_sum) %>%
group_by(course_id, quiz_id, user_id) %>%
arrange(as.POSIXct(date)) %>%
summarize(
firstattempt_date = first(date),
weeklabel = first(weeklabel), # weeklabel of first attempt
lastattempt_date = max(date),
late = ifelse(!is.na(due_at), firstattempt_date > first(due_at), 0),
quiz_type = first(quiz_type),
total_duration = sum(as.numeric(time_taken)),
first_duration = sum(as.numeric(time_taken)),
m_duration = mean(as.numeric(time_taken)/60),
nof_questions = first(question_count),
time_limit = first(as.numeric(time_limit)),
perc_time_taken = ifelse(time_limit > 0, m_duration/time_limit, NA),
allowed_attempts = first(allowed_attempts),
total_attempts = first(total_attempts),
max_score_perc = max(score)/first(max_quiz_score),
mean_score_perc = mean(score)/first(max_quiz_score)
) %>%
ungroup()
# summarize quiz indicators for full course
quiz_submission4_sum <- quiz_submission4 %>%
group_by(course_id, user_id) %>%
summarize(
n_practicequiz = sum(quiz_type == "practice_quiz" | time_limit == -1),
n_examquiz = sum(quiz_type %in% c("assignment", "graded_survey")
& time_limit != -1),
n_quiz_retry = sum(total_attempts > 1 & allowed_attempts > 1),
m_examquiz_duration_perc = ifelse(n_examquiz > 0, mean(perc_time_taken,
na.rm = T), NA),
m_practicequiz_duration = mean(m_duration, na.rm = T),
max_quizscore_perc = mean(max_score_perc, na.rm = T)
) %>% ungroup()
# summarize quiz indicators per week
quiz_submission4_sumweek <- quiz_submission4 %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_practicequiz = sum(quiz_type == "practice_quiz" | time_limit == -1),
n_examquiz = sum(quiz_type %in% c("assignment", "graded_survey")
& time_limit != -1),
n_quiz_retry = sum(total_attempts > 1 & allowed_attempts > 1),
m_examquiz_duration_perc = ifelse(n_examquiz > 0, mean(perc_time_taken,
na.rm = T), NA),
m_practicequiz_duration = mean(m_duration, na.rm = T),
max_quizscore_perc = mean(max_score_perc, na.rm = T)
)%>% ungroup()
# convert to datawide
quiz_weekwide <- quiz_submission4_sumweek %>%
full_join(quiz_sumweek) %>%
pivot_wider(id_col = c(course_id, user_id), names_from = weeklabel,
values_from = c(n_practicequiz, n_examquiz, n_quiz_retry,
m_examquiz_duration_perc, m_practicequiz_duration,
max_quizscore_perc,
n_quizclicks)) %>%
left_join(quiz_submission4_sum, by = c("course_id", "user_id")) %>%
left_join(quiz_sum, by = c("course_id", "user_id")) %>%
```

```
mutate_at(vars(starts_with("n_")), ~replace_na(., 0))
write.csv(quiz_weekwide, paste0(path, outfile),
row.names = FALSE)
quiz_weekwide
}
```

Creating discussion forum indicators

In this part, we extract indicators which relate to the discussion forum in the learning management system. For this, information is used from a variety of tables which store information about discussions in Canvas. Specifically, we use the discussion_topic_dim and discussion_topic_fact tables, which contain information on the discussion topics. In addition, we use the discussion_entry_fact table, as this table contains specific information on student replies to a specific discussion topic.

The function below extracts the following discussion forum indicators, and may be customized depending on the researcher's needs:

- Number of forum clicks
- Number of announcement clicks
- Number of forum topics posts
- Number of forum reply posts

All discussion forum indicators are calculated over the full course and per week of the course.

```
getforum_info <- function(requests_df, quartile, year, outfile){
requests4 <- requests_df
# load discussion tables
discussion_entry_fact <- read_parquet(paste0(path,
"discussion_entry_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N") %>%
filter(!is.na(course_id))
discussion_topic_fact <- read_parquet(paste0(path,
"discussion_topic_fact.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N") %>%
# remove discussion in groups
filter(!is.na(course_id))
discussion_topic_dim <- read_parquet(paste0(path,
"discussion_topic_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N") %>%
filter(!is.na(course_id))
####### transform & merge tables
##############################################
# merge discussion_topic dim & fact
disc_topics <- discussion_topic_dim %>%
left_join(discussion_topic_fact, by = c("id" = "discussion_topic_id",
"course_id", "group_id"))
```

```
# filter courses and add weeklabels
disc_topics2 <- disc_topics %>%
filter(!is.na(posted_at)) %>%
mutate(date_submission = as.POSIXlt(posted_at),
quartile = quartile,
weekno = strftime(date_submission,"%V"),
weeklabel = calculate_weeklabel(weekno, quartile, year))
# remove clicks outside timeframe & filter courses
disc_topics3 <- disc_topics2 %>%
filter(!is.na(weeklabel), course_id %in% requests4$course_id)
#merge discussion entry fact
disc_entries <- disc_topics %>%
select(-user_id) %>%
right_join(discussion_entry_fact, by = c("id" = "topic_id", "course_id"))
# filter courses and add weeklabels
disc_entries2 <- disc_entries %>%
filter(!is.na(posted_at)) %>%
mutate(date_submission = as.POSIXlt(posted_at),
quartile = quartile,
weekno = strftime(date_submission,"%V"),
weeklabel = calculate_weeklabel(weekno, quartile, year))
# remove clicks outside timeframe & filter courses
disc_entries3 <- disc_entries2 %>%
filter(!is.na(weeklabel), course_id %in% requests4$course_id)
# merge with discussion topic dim to get discussion type
requests_discussions <- requests4 %>%
filter(!is.na(discussion_id)) %>%
left_join(discussion_topic_dim, by = c("discussion_id" = "id", "course_id"))
####### Summarize discussion info
##########################################
### Discussion CLICKS ONLY (via requests)
# sum over full course
disc_sum <- requests_discussions %>%
group_by(course_id, user_id) %>%
summarize(
n_forumclicks = sum(is.na(type)),
n_announcementclicks = sum(type == "Announcement")
) %>%
ungroup()
# sum per week
disc_sumweek <- requests_discussions %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_forumclicks = sum(is.na(type)),
n_announcementclicks = sum(type == "Announcement")
) %>%
ungroup()
### ALL other discussion indicators via discussion_submissions
# post in topics
topic_sum <- disc_topics3 %>%
group_by(course_id, user_id) %>%
summarize(
n_forumtopic_post = n_distinct(id)
) %>% ungroup()
# summarize assignment indicators per week
topic_sumweek <- disc_topics3 %>%
group_by(course_id, user_id, weeklabel) %>%
```

```
summarize(
n_forumtopic_post = n_distinct(id)
) %>% ungroup()
# post in topics
entry_sum <- disc_entries3 %>%
group_by(course_id, user_id) %>%
summarize(
n_forumreply_post = n_distinct(id)
) %>% ungroup()
# summarize assignment indicators per week
entry_sumweek <- disc_entries3 %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_forumreply_post = n_distinct(id)
) %>% ungroup()
# convert to datawide
disc_weekwide <- topic_sumweek %>%
full_join(entry_sumweek) %>%
full_join(disc_sumweek) %>%
pivot_wider(id_col = c(course_id, user_id), names_from = weeklabel,
values_from = c(n_forumreply_post, n_forumtopic_post,
n_forumclicks, n_announcementclicks)) %>%
left_join(topic_sum, by = c("course_id", "user_id")) %>%
left_join(entry_sum, by = c("course_id", "user_id")) %>%
left_join(disc_sum, by = c("course_id", "user_id")) %>%
mutate_at(vars(starts_with("n_")), ~replace_na(., 0)) %>%
filter(!is.na(user_id))
write.csv(disc_weekwide, paste0(path, outfile),
row.names = FALSE)
disc_weekwide
}
```

Creating file and video indicators

In this part, we extract indicators which relate to the files in the learning management system. For this, information is used from a a single table which includes specific information on files in Canvas: file_dim. Note that the code discusses all file types, including videos, pictures and presentations. In case one wants to exclude any file type for generating more specific indicators, other filter operations can be applied to the requests_files2 table specifying which file types to include or exclude.

The function below extracts the following file indicators, and may be customized depending on the researcher's needs:

- number of clicks on files
- number of unique file accessed (including inline views & downloads)
- number of unique file downloads (via 'download' in url requests)
- number of clicks on video files (mediafiles directly uploaded on canvas with mp4 type of extension (file_type in file_dim))

- number of unique accessed video files
- number of clicks on livestreamed videos (media accessed via bigbluebutton, Canvas conferences, or panopto)
- number of unique accessed livestreamed videos (canvas conferences)

All file indicators are calculated over the full course and per week of the course.

```
getfiles_info <- function(requests_df, quartile, year, outfile){
requests4 <- requests_df
# load files
file_dim <- read_parquet(paste0(path, "file_dim.snappy.parquet"),
as_tibble = TRUE,
col_types = cols(.default = "c")) %>%
na_if("\\N")
####### transform & merge tables
##################################################
file_dim2 <- file_dim %>%
mutate(course_id = as.numeric(gsub("^7542", "", course_id))) %>%
filter(course_id %in% requests4$course_id) %>%
mutate(file_type = gsub("\\/.*", "", content_type),
file_type = ifelse(file_type == "binary"| is.na(file_type),
"unknown",
file_type),
file_id = as.numeric(gsub("^7542", "", id)))
requests_files <- requests4 %>%
filter(grepl("files/", url)) %>%
mutate(file_id = gsub(".*/files/", "", url),
file_id = as.numeric(gsub("/.*|\\?.*", "", file_id))) %>%
filter(!is.na(file_id))
#merge with file_dim for file_types
requests_files2 <- requests_files %>%
left_join(select(file_dim2, file_id, file_type, display_name))
# roughly file types include: video (.mp4), text (.docx, csv),
# application (pdf, pptx, zip), image (jpg, png), unknown (.sav, .R, .do, .dta)
# extract video views in canvas conferences
requests_conferences <- requests4 %>%
filter(grepl("conferences/|panopto", url)) %>%
mutate(conference_id = gsub(".*/conferences/", "", url),
conference_id = as.numeric(gsub("/.*|\\?.*", "", conference_id)))
####### Summarize file info
##################################################
# get unique access
# create one entry per (unique) file per user per course
files2 <- requests_files2 %>%
group_by(course_id, file_id, user_id) %>%
arrange(as.POSIXct(timestamp)) %>%
summarize(
n_fileclicks = n(),
weeklabel = first(weeklabel), # weeklabel of first access
downloaded = sum(grepl("download", url)) > 0 ,
video = first(file_type) == "video"
) %>%
ungroup()
# summarize files indicators for full course
files_sum <- files2 %>%
group_by(course_id, user_id) %>%
summarize(
n_fileclicks = sum(n_fileclicks, na.rm = T),
n_fileaccess = n(),
n_video = sum(video),
n_videoclicks = sum(n_fileclicks[video], na.rm = T),
n_filedownload = sum(downloaded)
```

```
) %>% ungroup()
# summarize assignment indicators per week
files_sumweek <- files2 %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_fileclicks = sum(n_fileclicks, na.rm = T),
n_fileaccess = n(),
n_videofile = sum(video),
n_videofileclicks = sum(n_fileclicks[video], na.rm = T),
n_filedownload = sum(downloaded)
) %>% ungroup()
# get unique access
# create one entry per (unique) video per user per course
conferences2 <- requests_conferences %>%
group_by(course_id, url, user_id) %>%
arrange(as.POSIXct(timestamp)) %>%
summarize(
n_confclicks = n(),
weeklabel = first(weeklabel), # weeklabel of first access
) %>%
ungroup()
# summarize video indicators for full course
video_sum <- conferences2 %>%
group_by(course_id, user_id) %>%
summarize(
n_videoclicks_total = sum(n_confclicks, na.rm = T),
n_video_total = n()
) %>% ungroup()
# summarize video indicators per week
video_sumweek <- conferences2 %>%
group_by(course_id, user_id, weeklabel) %>%
summarize(
n_videoclicks = sum(n_confclicks, na.rm = T),
n_video = n()
) %>% ungroup()
# convert to datawide
files_weekwide <- files_sumweek %>%
full_join(video_sumweek) %>%
pivot_wider(id_col = c(course_id, user_id), names_from = weeklabel,
values_from = c(n_fileclicks, n_fileaccess,
n_filedownload, n_videofileclicks, n_videofile,
n_videoclicks, n_video)) %>%
full_join(files_sum) %>%
full_join(video_sum) %>%
mutate_at(vars(starts_with("n_")), ~replace_na(., 0))
write.csv(files_weekwide, paste0(path, outfile),
row.names = FALSE)
files_weekwide
}
```

Merging all Canvas indicators

With the code below, you can run all separate functions described above to extract the Canvas indicators per category.

```
session_weekwide <- getsession_info(requests4, quartile, year,
"session_indicators.csv")
```

```
assignment_weekwide <- getassignment_info(requests4, quartile, year,
"assignment_indicators.csv")
quiz_weekwide <- getquiz_info(requests4, quartile, year,
"quiz_indicators.csv")
disc_weekwide <- getforum_info(requests4, quartile, year,
"discussion_indicators.csv")
files_weekwide <- getfiles_info(requests4, quartile, year,
"file_indicators.csv")
```

Thereafter, you can merge all Canvas indicators into one large data-wide csv file. This file contains one row per student per course, for all indicators.

```
# merge all canvas indicators
canvasall <- session_weekwide %>%
left_join(assignment_weekwide, by = c("course_id", "user_id")) %>%
left_join(quiz_weekwide, by = c("course_id", "user_id")) %>%
left_join(disc_weekwide, by = c("course_id", "user_id")) %>%
left_join(files_weekwide, by = c("course_id", "user_id"))
write.csv(canvasall, paste0(path, "CanvasIndicators.csv"))
```

This csv file can now be combined with additional data sources, such as grades or survey data and can be used for your further analyses.