PollEverywhere_Import

Miles Vaelncia

November 8th, 2024

Load Packages

```
pack <- c("tidyverse",    # Main processing package
          "tinytex")      # Knit to pdf
package.check <- lapply(
  pack, # List of packages to load
  FUN = function(x) { # Making a function
    if (!require(x, character.only = TRUE)) { # If you can't find the package
      install.packages(x, dependencies = TRUE) # First install it
      library(x, character.only = TRUE) # Then load it
    }
  }
)
rm(list = ls()) # Clear
```

Only the tidyverse package is necessary for modifying the PollEverywhere surveys. I prefer using tidyverse functions to manipulate the data, and tinytex is needed to produce this rmarkdown file but isn't necessary for manipulating the gradebook.

Identify File Names For Batch Processing

```
PE_List <- list.files("PollEverywhere/Raw", # Folder with Raw PE csv files
                      pattern = ".csv", full = TRUE) %>%
  # Create named list of files with shortened names for reference
  set_names(str_sub(., start = 20, end = -5)) %>% # Isolate name by character position
  print()
```

```
##                           GradebookExample
## "PollEverywhere/Raw/GradebookExample.csv"
```

All of your raw csv files from PollEverywhere (PE) need to be stored within the same folder. Ideally, these are the only files in the folder, but you could leave non- ".csv" files within the folder, and not affect the code. Each of your file names should be associated with the name of the canvas assignment. If you are combining all of your PE surveys into a single canvas assignment, I'll have separate code to accomplish that. Personally, my file pattern would shows Week and Day; i.e., PollEverywhere Week 7, day 2 = "PE W07D2". Depending on your file pattern, you can change the position to isolate your variable names.

Visualize Raw Imported Data

```
df <- read_csv("PollEverywhere/Raw/GradebookExample.csv", show_col_types = FALSE) %>% print()
```

```
## # A tibble: 13 x 30
##    `First name`         `Last name` `Screen name` `Custom report ID` Email  Rank
##    <chr>                <chr>       <chr>         <lgl>              <chr> <dbl>
##  1 Miles                Valencia    Miles         NA                 SID0~     1
##  2 Kilometers           Guadalajara Kilo          NA                 SID0~     2
##  3 Hectometers          Puebla      Hecto         NA                 SID0~     7
##  4 Decameters           Durango     Deca          NA                 SID0~     4
##  5 Meters               Cancun      Meter         NA                 SID0~     9
##  6 Decimeters           Cebu        Deci          NA                 SID0~    11
##  7 Centimeters          Baguio      Centi         NA                 SID0~     3
##  8 Milimeters           Manila      Mili          NA                 SID0~    10
##  9 Inches               Palayan     Inch          NA                 SID0~     5
## 10 Feet                 Escalante   Foot          NA                 SID1~     8
## 11 Yard                 Borogan     Yard          NA                 SID1~     6
## 12 Average grade        <NA>        <NA>          NA                 <NA>     NA
## 13 Average participati~ <NA>        <NA>          NA                 <NA>     NA
## # i 24 more variables: Attendance <dbl>, Grade <chr>, Participation <chr>,
## #   `Total points possible` <dbl>, `Total points earned` <dbl>,
## #   `Total answered` <dbl>, Question_01 <chr>, Question_01_Points <chr>,
## #   Question_01_CheckIn <chr>, Question_02 <chr>, Question_02_Points <dbl>,
## #   Question_02_CheckIn <chr>, Question_03 <chr>, Question_03_Points <chr>,
## #   Question_03_CheckIn <chr>, Question_04 <chr>, Question_04_Points <dbl>,
## #   Question_04_CheckIn <chr>, Question_05 <chr>, Question_05_Points <dbl>, ...
```

The gradebook loads with ease, but the headers are not exactly how we want to see them. If you export a PollEverywhere (PE) survey with no modifications, the questions, points, and check-ins, are not written in this standard format. The questions are the entire question in quotations, points are "Points earned", and check-ins are "Responded at (PST)". Both points and check-ins are exactly the same without clarifying, which question it's associated with. For this code, it's essential that you rename these columns of each PE survey. You can use this example file to easily copy the column names over to your files. All of the other columns can be left as is.

Options for Calculating Points

```
df %>%
  # filter out summary data at the bottom of each file
  filter(!is.na(Rank)) %>%
  # Create new columns without keeping old columns
  transmute(Email,
            PointsPossible = `Total points possible`,
            Answered = `Total answered`,
            TotalPoints = `Total points earned`,
            # No changes
            Points_1 = TotalPoints,
            # Require a threshold of correct answers for full credit
            Points_2 = if_else(TotalPoints > 3, 7, 0),
```

```
            # Allot more points for participation, but incentivize correctness
            Points_3 = (Answered * 0.8) + (TotalPoints * 0.2))
```

```
## # A tibble: 11 x 7
##    Email          PointsPossible Answered TotalPoints Points_1 Points_2 Points_3
##    <chr>                   <dbl>    <dbl>       <dbl>    <dbl>    <dbl>    <dbl>
##  1 SID01@email.e~              6        6           6        6        7     6
##  2 SID02@email.e~              6        5           5        5        7     5
##  3 SID03@email.e~              6        6         5.8      5.8        7     5.96
##  4 SID04@email.e~              6        3           4        4        7     3.2
##  5 SID05@email.e~              6        4           5        5        7     4.2
##  6 SID06@email.e~              6        6           6        6        7     6
##  7 SID07@email.e~              6        6         5.8      5.8        7     5.96
##  8 SID08@email.e~              6        6         5.8      5.8        7     5.96
##  9 SID09@email.e~              6        5           5        5        7     5
## 10 SID10@email.e~              6        5         5.8      5.8        7     5.16
## 11 SID11@email.e~              6        6           6        6        7     6
```

There are many ways that you can calculate points for a PollEverywhere survey, but I listed out 3 options that have been used in my classes. First, you can reward points as listed if PE calculates points as you want before exporting. Second, you can award full points as long as they hit a threshold of correct answers. For this example, as long as students get at least 50% correct, they get full credit. Lastly, you can award majority of points to participation, but also award points for being correct. For my example, I award 80% for participation and 20% for correctness; this is assuming each question is 1 point each. You'll need to adjust your calculations if questions are more points than 1.

If you'd like to utilize the geolocation tracking or check-in PE feature, you can only count points for questions they answered in class. You'll have to do some extra processing to filter out questions with no check-in. I have incorporated this method within my main example for batch processing all files.

Batch Process Files

```
# 1. Import data
# Process each file and combine into one dataset
PE_clean <- imap_dfr(
  PE_List, # List of file names for batch processing
  ~ read_csv(.x, show_col_types = FALSE) %>%  # Load each file in list
    add_column(Day = .y) %>%  # Add Day column from file name for context
    filter(!is.na(Rank)) %>%  # Remove summary rows (usually marked by NA in Rank)

# 2. Wrangle data
  # Separate Email column to extract UCINet ID only (before "@")
  separate_wider_delim(Email, "@", names = c("UCINet", NA)) %>%

  # Standardize columns, including converting Points columns to double type
  mutate(
    # Keep standardized name columns
    FirstName = `First name`, LastName = `Last name`,
    # Retain Day for pivoting later
    Day,
    # Convert Points columns to double
    across(matches("^Question_\\d+_Points$"), as.double)) %>%
```

```
    # Columns that start with "Question_" and end with "_Points"

# Reshape data to long format for question-related columns
  pivot_longer(
    # Select columns with question data
    cols = matches("^Question_\\d+_(Points|CheckIn)$"),
    # Create new columns "Question" and "value"
    names_to = c("Question", ".value"),
    # Split names into Question and Check-In
    names_pattern = "^(Question_\\d+)_(Points|CheckIn)$") %>%
    # Retain only relevant columns
    select(FirstName, LastName, UCINet, Question, Points, CheckIn, Day) %>%

# 3. Calculate points
  # Calculate total checked-in points per UCINet ID for each day
  filter(!is.na(CheckIn)) %>%  # Only include rows where CheckIn is recorded
  group_by(FirstName, LastName, UCINet, Day) %>%
  # Sum points and ungroup
  summarise(TotalPoints = sum(Points, na.rm = TRUE), .groups = "drop")) %>%

# 4. Prepare for export
  # Reshape to wide format with days as columns and total points per day as values
  pivot_wider(
    # Use Day column names for new columns
    names_from = Day,
    # Populate with TotalPoints values
    values_from = TotalPoints) %>%
# Sort final data by UCINet ID for clarity
  arrange(UCINet) %>%
  print()  # Display final output
```

```
## # A tibble: 11 x 4
##    FirstName   LastName    UCINet GradebookExample
##    <chr>       <chr>       <chr>             <dbl>
##  1 Miles       Valencia    SID01               6
##  2 Kilometers  Guadalajara SID02               6
##  3 Hectometers Puebla      SID03               3.8
##  4 Decameters  Durango     SID04               3
##  5 Meters      Cancun      SID05               4
##  6 Decimeters  Cebu        SID06               5
##  7 Centimeters Baguio      SID07               4.8
##  8 Milimeters  Manila      SID08               4.8
##  9 Inches      Palayan     SID09               4
## 10 Feet        Escalante   SID10               3.8
## 11 Yard        Borogan     SID11               4
```

"imap_dfr" is a function that applies the following commands to each element within a list, and then combines all of the elements into a single dataframe. This is essential since all of the files need to be processed individually before preparing for export.

First, we want to import the data and filter out the summary data by the rank column; you can use any column that isn't populated. All students are given a rank whereas a student may sign-in without an email, which can cause problems if used as a filter. At this step, we also create a new column populated with the Day associated with the file.

Second, we want to wrangle the data into a format easy for manipulations. We extract the UCINet IDs from the emails, which is needed for importing into Canvas. Depending on your institution, you may need to adjust how you extract student IDs to pair with your Canvas gradebook. For the 3 columns associated with each question (Question, Points, and CheckIn), we want to reshape the columns to be in a long format. We only want to keep the columns needed for merging with your Canvas gradebook, which include: UCINet (or student ID), Question, Points, CheckIn, and Day. We retain first & last names because we don't remember our students by their IDs; at least I don't! This makes it easier for us to read the code's outputs, but we'll have to delete those columns when importing into Canvas.

Third, we want to filter out any points not associated with a check-in. This indicates that students were not located in class if they don't have a value. Then we group rows by student ID and day. Now we can calculate the points as we want.

Lastly, we want to reshape the data into a wide format to easily read how many points students earned for each assignment. The format is also necessary for importing into Canvas. This concludes individually processing files, so we end the "imap_dfr" function to combine each element into 1 dataset.

Export Gradebook

```
SaveDate <- format(Sys.Date(), format = "%Y%b%d")
write_csv(PE_clean,
          na = "", # Leave missing values blank
          file.path("PollEverywhere/Clean", # Folder path to save file
            str_c("Example101_PE_", # Change to your class information
                  SaveDate, ".csv")))
```

The final step is to export the dataset as a new csv file. We auto-generate the date to associate with the new file because that'll help with bookkeeping. Before you run the code for exporting, you'll want to change the file path to match your folder structure. Otherwise, this is the easiest step. You might notice that some students will have zeroes or blanks for a PollEverywhere (PE) survey. Zeroes indicate that students got none of the questions correct. If you integrate a participation component into your grading, then no student should have a zero; manually check if that occurs. If some students do not participate in a PE survey, they will not be found in this csv file. If they participate in 1 PE survey, but not another, then the latter column will have a blank cell for the student. This difference is important when discussing grades with students.

Post Processing

You'll need to complete a few steps manually to organize the csv files specifically how Canvas wants for importing grades. First, delete FirstName and LastName. These were only retained for your convenience. Second, delete the header name "UCINet" (or student ID). Lastly, insert a row below these headers and include 1) "Points Possible" (without the quotations) above UCINet, and 2) the total points possible for each assignment. Preferrably, I save this as a new file, so that I don't have to rerun my code to view the output with student names. As you amass more PollEverywhere (PE) surveys, you can rerun the code and delete the older files because the new file will contain all PE surveys in the folder.