format_check_cell_suppression

Claire Boulange

2023-05-12

```
# Load any necessary packages
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(readr)
library(knitr)
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.2.3
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Part 1: - open a connection to the output file - Set the working directory - folder with data cleaned ready for QA

- define a list of acceptable values for the first column
- define a function to check if column names are in snake case format

- define a regex pattern to check age_group format
- define regex patterns for age_group, year_range, and calendar_year
- get a list of CSV files in the working directory + subfolders
- print the number of files in the folder and their names
- initialize an empty list to store data frame
- read each CSV file into a data frame and store it in the df_list
- append the data frame to the list using the file name as the key

```
cat("Number of files in folder:", length(csv_files), "\n")
```

```
## Number of files in folder: 5
```

```
for (file in csv_files) {
  if (!grepl("\\d{3,}", file)) {
    cat("WARNING! File(s) without indicator code:", file, "\n")
  }
}
```

```
## WARNING! File(s) without indicator code: ABS_schools_Attendance_at_primary_school_year_5_STE.csv
```

```
cat(paste("File with indicator code:",csv_files, collapse = "\n"), "\n")
```

```
## File with indicator code: ABS_schools_461_retention_rate_STE.csv
## File with indicator code: ABS_schools_462_school_completion_year_12.csv.csv
## File with indicator code: ABS_schools_463_continuation_rates_STE.csv
## File with indicator code: ABS_schools_473_full_time_and_part_time_students_STE.csv
## File with indicator code: ABS_schools_Attendance_at_primary_school_year_5_STE.csv
```

```
cat("Finished checking CSV files.", "\n")
```

```
## Finished checking CSV files.
```

```
for (file in csv_files) {
  if (!grepl("_STE|_SA3|_SA2|_SA4|_national|_Australia", file)) {
    cat("WARNING! File(s) without geography suffix:", file, "\n")
  }
}
```

```
## WARNING! File(s) without geography suffix: ABS_schools_462_school_completion_year_12.csv.csv
```

```
cat(paste("File with indicator code:", csv_files, collapse = "\n"), "\n")
```

```
## File with indicator code: ABS_schools_461_retention_rate_STE.csv
## File with indicator code: ABS_schools_462_school_completion_year_12.csv.csv
## File with indicator code: ABS_schools_463_continuation_rates_STE.csv
## File with indicator code: ABS_schools_473_full_time_and_part_time_students_STE.csv
## File with indicator code: ABS_schools_Attendance_at_primary_school_year_5_STE.csv
```

```r
cat("Finished checking CSV files.", "\n")
```

```
## Finished checking CSV files.
```

Part 2 - loop through each data frame in the list and print the head

```r
for (df_name in names(df_list)) {
  cat(paste("Head of", df_name, ":\n"))

  # Get the data frame
  df <- df_list[[df_name]]

  # Generate the HTML table for the head of the data frame
  html_table <- kable(head(df), format = "html") %>%
    kable_styling(bootstrap_options = "striped", full_width = FALSE)

  # Print the HTML table
  cat(as.character(html_table))
  cat("\n")
}
```

Head of ABS_schools_461_retention_rate_STE.csv :

STE_CODE16

calendar_year

sex

age_group

school_grade

apparent_retention_rate

total_rentention_rate

1

2022

male

13-14

year 7 - year 8

99.2

99.4

1

2022

male

14-15

year 8 - year 9

99.7

99.7

1

2022

male

15-16

year 9 - year 10

99.4

99.2

1

2022

male

16-17

year 10 - year 11

79.8

77.8

1

2022

male

17-18

year 11 - year 12

79.0

77.7

1

2022

female

13-14

year 7 - year 8

98.9

99.1

Head of ABS_schools_462_school_completion_year_12.csv.csv :

STE_CODE16

calendar_year

sex

age_group

school_grade

affiliation_abs_schools

n_full_time_student

n_part_time_student

1

2022

male

16

year 12

government

15

0

1

2022

male

17

year 12

government

646

28

1

2022

male

18

year 12

government

248

19

1

2022

male

16

year 12

government

330

9

1

2022

male

17

year 12

government

11941

313

1

2022

male

18

year 12

government

3966

159

Head of ABS_schools_463_continuation_rates_STE.csv :

STE_CODE16

calendar_year

age_group

sex

p_apparent_continuation_rate

1

2022

14-15

male

98.1

1

2022

15-16

male

88.3

1

2022

16-17

male

79.9

1

2022

17-18

male

26.5

1

2022

18-19

male

2.6

1

2022

14-15

female

98.9

Head of ABS_schools_473_full_time_and_part_time_students_STE.csv :

STE_CODE16

calendar_year

age_group

sex

affiliation_abs_schools

n_full_time_student

n_part_time_student

1

2022

0-4

male

government

38

0

1

2022

5

male

government

2076

0

1

2022

6

male

government

1225

0

1

2022

7

male

government

9

0

1

2022

5

male

government

17

0

1

2022

6

male

government

1949

0

Head of ABS_schools_Attendance_at_primary_school_year_5_STE.csv :

STE_CODE16

calendar_year

sex

age_group

school_grade

affiliation_abs_schools

n_full_time_student

n_part_time_student

1

2022

male

9

year 5

government

20

0

1

2022

male

10

year 5

government

2034

0

1

2022

male

11

year 5

government

1022

0

1

2022

male

12

year 5

government

10

0

1

2022

male

9

year 5

government

466

0

1

2022

male

10

year 5

government

22694

0

- print overview of all CSV files and their variables

```r
# Create a new data frame with only the variables column
csv_info <- data.frame(variables = sapply(df_list, function(x) paste(names(x), collapse = ", ")))

# Generate the HTML table using kable and kableExtra functions
html_table <- kable(csv_info, format = "html", col.names = NULL) %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE)

# Print the HTML table
cat(as.character(html_table))
```

ABS_schools_461_retention_rate_STE.csv

STE_CODE16, calendar_year, sex, age_group, school_grade, apparent_retention_rate, total_rentention_rate

ABS_schools_462_school_completion_year_12.csv.csv

STE_CODE16, calendar_year, sex, age_group, school_grade, affiliation_abs_schools, n_full_time_student, n_part_time_student

ABS_schools_463_continuation_rates_STE.csv

STE_CODE16, calendar_year, age_group, sex, p_apparent_continuation_rate

ABS_schools_473_full_time_and_part_time_students_STE.csv

STE_CODE16, calendar_year, age_group, sex, affiliation_abs_schools, n_full_time_student, n_part_time_student

ABS_schools_Attendance_at_primary_school_year_5_STE.csv

STE_CODE16, calendar_year, sex, age_group, school_grade, affiliation_abs_schools, n_full_time_student, n_part_time_student

- print a data dictionary for each data frame

```r
library(knitr)
library(kableExtra)

# Loop for generating data dictionary
for (df_name in names(df_list)) {
  cat(paste("Data Dictionary for", df_name, ":\n"))

  # Get the data frame
  df <- df_list[[df_name]]

  # Create a data frame with variable name, class, range, unique values, and count of missing values
  var_info <- data.frame(
    variable = names(df),
    class = sapply(df, class),
    range = sapply(df, function(x) if (is.numeric(x)) paste(range(x, na.rm = TRUE), collapse = " - ") el
    unique_values = sapply(df, function(x) if (is.character(x)) paste(unique(x), collapse = ", ") else
    n_missing_values = sapply(df, function(x) sum(is.na(x)))
  )

  # Generate the HTML table using kable and kableExtra functions
  html_table <- kable(var_info, format = "html") %>%
    kable_styling(bootstrap_options = "striped", full_width = FALSE)

  # Print the HTML table
  cat(as.character(html_table))
  cat("\n")
}
```

Data Dictionary for ABS_schools_461_retention_rate_STE.csv :

variable

class

range

unique_values

n_missing_values

STE_CODE16

STE_CODE16

integer

0 - 8

0

calendar_year

calendar_year

integer

2011 - 2022

0

sex

sex

character

male, female, persons

0

age_group

age_group

character

13-14, 14-15, 15-16, 16-17, 17-18

0

school_grade

school_grade

character

year 7 - year 8, year 8 - year 9, year 9 - year 10, year 10 - year 11, year 11 - year 12

0

apparent_retention_rate

apparent_retention_rate

numeric

55.4 - 100

0

total_rentention_rate

total_rentention_rate

numeric

54.1 - 100

0

Data Dictionary for ABS_schools_462_school_completion_year_12.csv.csv :

variable

class

range

unique_values

n_missing_values

STE_CODE16

STE_CODE16

integer

1 - 8

0

calendar_year

calendar_year

integer

2006 - 2022

0

sex

sex

character

male, female

0

age_group

age_group

character

16, 17, 18, 19, 20, 21+, 15, 14, 12

0

school_grade

school_grade

character

year 12

0

affiliation_abs_schools

affiliation_abs_schools

character

government, catholic, independent

0

n_full_time_student

n_full_time_student

integer

0 - 15194

0

n_part_time_student

n_part_time_student

integer

0 - 669

0

Data Dictionary for ABS_schools_463_continuation_rates_STE.csv :

variable

class

range

unique_values

n_missing_values

STE_CODE16

STE_CODE16

integer

0 - 8

0

calendar_year

calendar_year

integer

2011 - 2022

0

age_group

age_group

character

14-15, 15-16, 16-17, 17-18, 18-19

0

sex

sex

character

male, female, persons

0

p_apparent_continuation_rate

p_apparent_continuation_rate

numeric

1.8 - 100

0

Data Dictionary for ABS_schools_473_full_time_and_part_time_students_STE.csv :

variable

class

range

unique_values

n_missing_values

STE_CODE16

STE_CODE16

integer

1 - 8

0

calendar_year

calendar_year

integer

2006 - 2022

0

age_group

age_group

character

0-4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21+

0

sex

sex

character

male, female

0

affiliation_abs_schools

affiliation_abs_schools

character

government, catholic, independent

0

n_full_time_student

n_full_time_student

integer

0 - 25572

0

n_part_time_student

n_part_time_student

integer

0 - 1552

0

Data Dictionary for ABS_schools_Attendance_at_primary_school_year_5_STE.csv :

variable

class

range

unique_values

n_missing_values

STE_CODE16

STE_CODE16

integer

1 - 8

0

calendar_year

calendar_year

integer

2006 - 2022

0

sex

sex

character

male, female

0

age_group

age_group

integer

8 - 15

0

school_grade

school_grade

character

year 5

0

affiliation_abs_schools

affiliation_abs_schools

character

government, catholic, independent

0

n_full_time_student

n_full_time_student

integer

0 - 24368

0

n_part_time_student

n_part_time_student

integer

0 - 350

0

Part 3 - iterate through each data frame in df_list and perform checks:

1. check if age_group, sex, calendar year or year_range columns exist
2. check if age_group values are in the correct format
3. check if year_range values are in the correct format
4. check if calendar_year values are in the correct format
5. check if geography column is one of the acceptable values

```
for (df_name in names(df_list)) {
  df <- df_list[[df_name]]
  col_names <- names(df)
  first_col <- colnames(df)[1]


  if (!("age_group" %in% col_names)) {
    cat("Error: age_group column not found in", df_name, "\n")
  }
  if (!("sex" %in% col_names)) {
    cat("Error: sex column not found in", df_name, "\n")
  }
  if (!("calendar_year" %in% col_names) && !("year_range" %in% col_names)) {
    cat("Error: either calendar_year or year_range column must be present in", df_name, "\n")
  } else if (("calendar_year" %in% col_names) && ("year_range" %in% col_names)) {
    cat("Error: both calendar_year and year_range columns cannot be present in", df_name, "\n")
  } else if ("calendar_year" %in% col_names && !all(grepl("\\d{4}", df$calendar_year))) {
    cat("Error: calendar_year values in", df_name, "are not in the correct format (expected format: \\d
  } else if ("year_range" %in% col_names && !all(grepl("\\d{4}-\\d{4}", df$year_range))) {
    cat("Error: year_range values in", df_name, "are not in the correct format (expected format: \\d{4}
  }


  if ("age_group" %in% col_names) {
    age_group_values <- df$age_group
    if (!all(grepl(age_group_regex, age_group_values))) {
      cat("Error: age_group values in", df_name, "are not in the correct format (expected format: \\d-\
      cat("Invalid values:\n")
      invalid_age_group_values <- age_group_values[!grepl(age_group_regex, age_group_values)]
      cat(paste(unique(invalid_age_group_values), collapse=", "), "\n")
    }
  }

  if ("year_range" %in% col_names) {
    year_range_values <- df$year_range
    if (!all(grepl(year_range_regex, year_range_values))) {
      cat("Error: year_range values in", df_name, "are not in the correct format (expected format: \\d{4
      cat("Invalid values:\n")
      invalid_year_range_values <- year_range_values[!grepl(year_range_regex, year_range_values)]
      cat(paste(unique(invalid_year_range_values), collapse=", "), "\n")
    }
  }

  if ("calendar_year" %in% col_names) {
    calendar_year_values <- df$calendar_year
    if (!all(grepl(calendar_year_regex, calendar_year_values))) {
      cat("Error: calendar_year values in", df_name, "are not in the correct format (expected format: \
      cat("Invalid values:\n")
      invalid_calendar_year_values <- calendar_year_values[!grepl(calendar_year_regex, calendar_year_val
      cat(paste(unique(invalid_calendar_year_values), collapse=", "), "\n")
    }
  }

  if (!is_snake_case(col_names[-1])) {
    cat("Error: column names in", df_name, "are not in snake case format (lowercase words separated by u
  }

  if (!(first_col %in% first_col_check)) {
    cat("Error: geography column is not one of the acceptable values (", paste(first_col_check, collapse
  }

  if (!is.na(first_col) && first_col != "Australia" && any(df[[1]] == 0, na.rm = TRUE)) {
  cat("Error: Values coded as 0 (Australia) found in a dataset that is not national:", df_name, "\n")
  }
}
```

```
## Error: Values coded as 0 (Australia) found in a dataset that is not national: ABS_schools_461_retenti
## Error: age_group values in ABS_schools_462_school_completion_year_12.csv.csv are not in the correct f
## Invalid values:
## 16, 17, 18, 19, 20, 21+, 15, 14, 12
## Error: Values coded as 0 (Australia) found in a dataset that is not national: ABS_schools_463_continu
## Error: age_group values in ABS_schools_473_full_time_and_part_time_students_STE.csv are not in the co
## Invalid values:
## 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21+
## Error: age_group values in ABS_schools_Attendance_at_primary_school_year_5_STE.csv are not in the cor
## Invalid values:
## 9, 10, 11, 12, 13, 8, 15
```

STOP HERE AND CHEC

Did you detect any formatting errors?

YES »»»»»»»»»»»»»»»»»»»»»> GO BACK TO YOUR CODE AND MAKE CORRECTIONS

NO »»»»»»»»»»»»»»»»»»»»»» PROCEED WITH CELL SUPPRESSION

\ Part 4 - cell suppression - define the input directory path - define the output directory path to save cleaned datasets WITH CELL SUPPRESSED (R will automatically create this folder if it does not exist)

```
input_dir <- "C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/"
output_dir <- "C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/c
```

run this code to… - create the output directory if it doesn’t already exist - get a list of all CSV files in the input directory

```
dir.create(output_dir, showWarnings = FALSE)
csv_files <- list.files(input_dir, pattern = ".csv$", full.names = TRUE)
```

Run this code to… check if there is an “uncertainty” column in the data frame

```
for (file in csv_files) {


  df <- read.csv(file, stringsAsFactors = FALSE)


  if ("uncertainty" %in% colnames(df)) {
    message(paste0("Note: The file ", basename(file), " contains an 'uncertainty' column. Make sure to 
  else {
  # Print message indicating that there is no need to apply cell suppression to "uncertainty" column
  cat("You don't have to worry about cell suppression on 'uncertainty' in", basename(file), "\n")
}
}
```

```
## You don’t have to worry about cell suppression on ’uncertainty’ in ABS_schools_461_retention_rate_ST
## You don’t have to worry about cell suppression on ’uncertainty’ in ABS_schools_462_school_completion
## You don’t have to worry about cell suppression on ’uncertainty’ in ABS_schools_463_continuation_rate
## You don’t have to worry about cell suppression on ’uncertainty’ in ABS_schools_473_full_time_and_par
## You don’t have to worry about cell suppression on ’uncertainty’ in ABS_schools_Attendance_at_primary
```

Run this code to… detect columns that are numeric and where you might need to apply cell suppression

```
# Define the exclusion list
exclude_list <- c("STE_CODE16", "SA2_CODE16", "SA3_CODE16", "SA4_CODE16", "LGA_CODE16", "Australia", "s

# Loop through each CSV file and check for columns that are numeric and not in the exclusion list
for (file in csv_files) {

  # Read in the CSV file
  df <- read.csv(file, stringsAsFactors = FALSE)

  # Get the names of columns that are numeric and not in the exclusion list
  num_cols <- names(df)[sapply(df, is.numeric) & !names(df) %in% exclude_list]

  # If there are any such columns, print a message for each file and column
  if (length(num_cols) > 0) {
    for (col in num_cols) {
      message(paste0("For file ", basename(file), ", check values in column '", col, "' for cell suppre
    }
  } else {
    # Print message indicating that there are no columns to check
    cat("You don't have to worry about cell suppression in any numeric columns in", basename(file), "\n
  }

}
```

```
## For file ABS_schools_461_retention_rate_STE.csv, check values in column ’apparent_retention_rate’ fo

## For file ABS_schools_461_retention_rate_STE.csv, check values in column ’total_rentention_rate’ for

## For file ABS_schools_462_school_completion_year_12.csv.csv, check values in column ’n_full_time_stude

## For file ABS_schools_462_school_completion_year_12.csv.csv, check values in column ’n_part_time_stude

## For file ABS_schools_463_continuation_rates_STE.csv, check values in column ’p_apparent_continuation

## For file ABS_schools_473_full_time_and_part_time_students_STE.csv, check values in column ’n_full_ti

## For file ABS_schools_473_full_time_and_part_time_students_STE.csv, check values in column ’n_part_ti

## For file ABS_schools_Attendance_at_primary_school_year_5_STE.csv, check values in column ’n_full_time

## For file ABS_schools_Attendance_at_primary_school_year_5_STE.csv, check values in column ’n_part_time
```

Once you understand what you need to do with cell suppression (which columns represent count values in your series of data set AND if you have uncertainty columns to deal with) customise the code below to apply cell suppression (keep the first 4 columns as they are)

Loop through each CSV file and apply cell suppression - I used the outputs above to specify which column contain the count values

Make sure you un-comment the write csv line

```
for (file in csv_files) {

  # Read in the CSV file
  df <- read.csv(file, stringsAsFactors = FALSE)


  # Check if the n_full_time_student column exists in the data frame
  if ("n_full_time_student" %in% colnames(df)) {
    # Apply cell suppression to n_full_time_student column
    df[df[,"n_full_time_student"] %in% 0:4 & !is.na(df[,"n_full_time_student"]), -c(1:4)] <- 9999999
  } else {
    # Print a message to indicate that the n_full_time_student column was not found
    cat("Skipping file", file, "because it does not contain the n_full_time_student column.\n")
  }

  # Check if the n_part_time_student column exists in the data frame
  if ("n_part_time_student" %in% colnames(df)) {
    # Apply cell suppression to n_part_time_student column
    df[df[,"n_part_time_student"] %in% 0:4 & !is.na(df[,"n_part_time_student"]), -c(1:4)] <- 9999999
  } else {
    # Print a message to indicate that the n_part_time_student column was not found
    cat("Skipping file", file, "because it does not contain the n_part_time_student column.\n")
  }

  # Write the modified data frame to a new CSV file in the output directory
  #write.csv(df, file.path(output_dir, basename(file)), row.names = FALSE)

}
```

```
## Skipping file C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/
## Skipping file C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/
## Skipping file C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/
## Skipping file C:/Users/00095998/OneDrive - The University of Western Australia/acwa_temp/abs_schools/
```