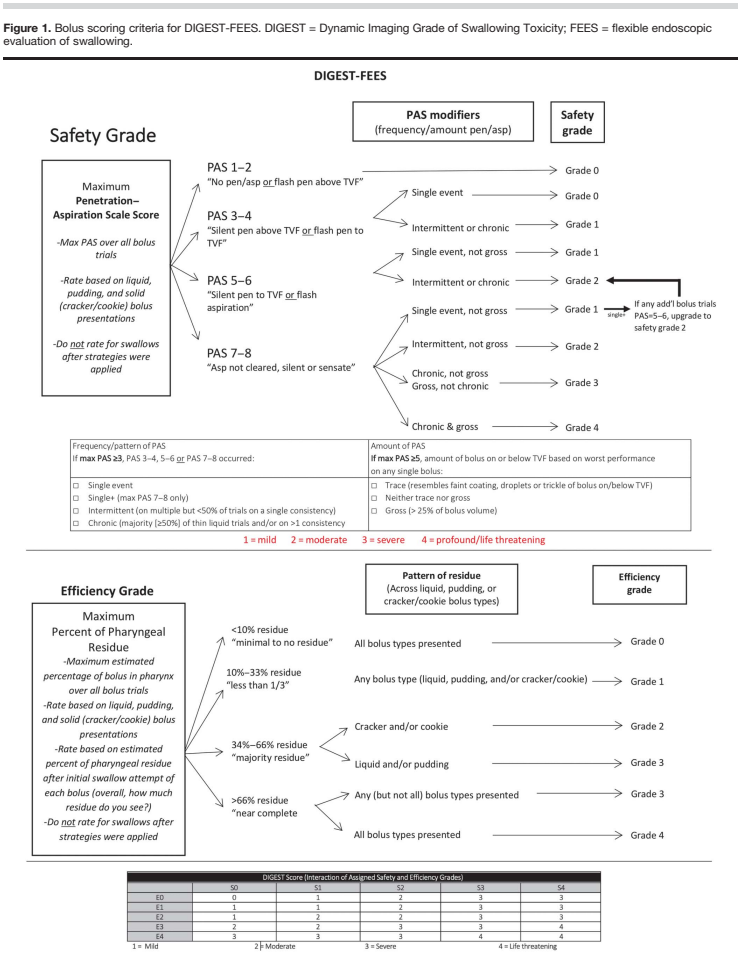
Calculating DIGEST Scores in R

James C. Borders

Background

The Dynamic Imaging Grade of Swallowing Toxicity (DIGEST) is a method to grade the severity of pharyngeal swallowing impairment. The scale can be used with either videofluoroscopic swallowing studies (Hutcheson et al., 2016; 2022) or flexible endoscopic evaluations of swallowing (i.e., DIGEST-FEES; Starmer et al., 2021). DIGEST aligns with the Common Terminology Criteria for Adverse Events (CTCAE) commonly employed during clinical trials to grade the severity of cancer toxicity (Grade 1 = mild, 2 = moderate, 3 = severe, 4 = life threatening). DIGEST evaluates the frequency and severity of airway invasion and pharyngeal residue, resulting in grades for swallowing safety and efficiency impairments. Grades from these two domains are used to derive an overall severity grade (Grade 1 = no dysphagia, 2 = moderate, 3 = severe, 4 = profound/life threatening).

Figure 1. Bolus scoring criteria for DIGEST-FEES. DIGEST = Dynamic Imaging Grade of Swallowing Toxicity; FEES = flexible endoscopic evaluation of swallowing.



Our goal in this vignette is to describe how to use an R function to automatically calculate safety, efficiency, and total DIGEST-FEES scores.

Setting Up Your Data

First, we need to load the necessary R packages.

```
library(tidyverse) # data wrangling
library(simstudy) # simulate data
set.seed(2022) # reproducibility
```

Next, we will simulate some fake swallowing data that we will use as an example for the function. Below we have simulated 20 participants and can view the variable names.

```
data_frame <- sim_swallowing_data(
  between_variance = 1,
  sample_size = 20)

head(data_frame)
#> # A tibble: 6 x 6
#> # Groups:   id [2]
#>      id pen_asp vocal_folds_severity_rating subglottis_severity_~1 perce~2 IDDSI
#>   <int>   <dbl>                       <dbl>                   <dbl>   <dbl> <dbl>
#> 1     1       4                          NA                      NA    4.42       0
#> 2     1       6                          NA                      NA    1.51       0
#> 3     1       7                          NA                       0.0390  1.28       0
#> 4     1       7                          NA                      11.5     0.0899     0
#> 5     2       3                          NA                      NA    6.83       0
#> 6     2       3                          NA                      NA    4.17       0
#> # ... with abbreviated variable names 1: subglottis_severity_rating,
#> #   2: percent_pharyngeal_residue
```

You can also import your own data. You can download a csv template here.

```
data <- read.csv("your data path here.csv")
```

Below we have the DIGEST-FEES function to calculate safety, efficiency, and total grades.

```
digest_fees_function <- function(data, # put data frame name here
                          id, # put name of id grouping variable here
                          IDDSI, # put name of consistency variable here
                          pas, # put name of PAS outcome variable here
                          percent_pharyngeal_residue, # put name of % pharyngeal residue score here
                          vocal_folds_severity_rating, # put name of VF severity rating here
                          subglottis_severity_rating # put name of subglottis severity rating here
) {

  ### Calculate SAFETY Grade
  data_x <- {{data}} |>
    group_by(id, {{IDDSI}}) |>
    summarise(num_trials = n()) # calculate # of trials for each consistency
```

```
x <- inner_join(data, data_x) # join # of trials summary with full data set

x2 <- x |>
  group_by(id, {{IDDSI}}, .drop = F) |> # calculate % of PAS for each grouping by consistency
  mutate(perc_3_4 = sum({{pas}} == 3 | {{pas}} == 4, na.rm = T)/num_trials,
         perc_5_6 = sum({{pas}} == 3 | {{pas}} == 4, na.rm = T)/num_trials,
         perc_7_8 = sum({{pas}} == 3 | {{pas}} == 4, na.rm = T)/num_trials) |>
  group_by(id) |>
  mutate(binary_airway_invasion = case_when({{pas}} > 2 ~ 1, # counts number of PAS > 2 events per pa
                                            {{pas}} <= 2 ~ 0),
         max_pas = max({{pas}}, na.rm = T), # max PAS
         freq_3_4 = sum({{pas}} == 3 | {{pas}} == 4, na.rm = T), # overall frequencies for PAS 3 & 4
         freq_5_6 = sum({{pas}} == 5 | {{pas}} == 6, na.rm = T), # overall frequencies for PAS 5 & 6
         freq_7_8 = sum({{pas}} == 7 | {{pas}} == 8, na.rm = T)) # overall frequency of PAS 7 or 8

x3 <- x2 |> # calculates airway invasion across multiple consistencies
  group_by(id, {{IDDSI}}) |>
  mutate(freq_airway_invasion = sum(binary_airway_invasion, na.rm = T),
         airway_invasion_by_consistency = case_when(freq_airway_invasion >= 1 ~ 1,
                                                    freq_airway_invasion < 1 ~ 0)
  ) |>
  dplyr::slice(1) |>
  group_by(id) |>
  summarise(sum(airway_invasion_by_consistency, na.rm = T)) |>
  mutate(airway_invasion_multiple_consistencies = case_when(
    `sum(airway_invasion_by_consistency, na.rm = T)` > 1 ~ "yes",
    `sum(airway_invasion_by_consistency, na.rm = T)` <= 1 ~ "no")) |>
  dplyr::select(id, airway_invasion_multiple_consistencies)

x4 <- inner_join(x2, x3)

x5 <- x4 |> # summary info on whether someone had intermittent or chronic frequency on at least 1 con
  group_by(id, {{IDDSI}}) |>
  dplyr::slice(1) |>
  group_by(id) |>
  mutate(pas_3_4_chronic_pre = case_when(perc_3_4 < 50 ~ 0, # count number of times >= 50% events
                                         perc_3_4 >= 50 ~ 1),
         pas_5_6_chronic_pre = case_when(perc_5_6 < 50 ~ 0,
                                         perc_5_6 >= 50 ~ 1),
         pas_7_8_chronic_pre = case_when(perc_7_8 < 50 ~ 0,
                                         perc_7_8 >= 50 ~ 1),
         pas_3_4_chronic_pre_sum = sum(pas_3_4_chronic_pre, na.rm = T), # add up freq of these events
         pas_5_6_chronic_pre_sum = sum(pas_5_6_chronic_pre, na.rm = T),
         pas_7_8_chronic_pre_sum = sum(pas_7_8_chronic_pre, na.rm = T)
  ) |>
  dplyr::slice(1) |>
  mutate(pas_3_4_chronic = case_when(pas_3_4_chronic_pre_sum <= 1 ~ "intermittent",
                                     pas_3_4_chronic_pre_sum >1 ~ "chronic"),
         pas_5_6_chronic = case_when(pas_5_6_chronic_pre_sum <= 1 ~ "intermittent",
                                     pas_5_6_chronic_pre_sum >1 ~ "chronic"),
         pas_7_8_chronic = case_when(pas_7_8_chronic_pre_sum <= 1 ~ "intermittent",
                                     pas_7_8_chronic_pre_sum >1 ~ "chronic")
  ) |>
```

```
  dplyr::select(id, pas_3_4_chronic, pas_5_6_chronic, pas_7_8_chronic)

x6 <- inner_join(x4, x5)

df_safety <- x6 |>
  dplyr::select(id, {{pas}}, max_pas, {{vocal_folds_severity_rating}},
                {{subglottis_severity_rating}}, freq_3_4, freq_5_6, freq_7_8,
                airway_invasion_multiple_consistencies, pas_3_4_chronic,
                pas_5_6_chronic, pas_7_8_chronic) |>
  drop_na({{pas}}) |>
  group_by(id) |>
  top_n(1, {{pas}}) |>  # keep rows based on max PAS
  mutate( # calculate max values for VF and subglottis for when > 2 trials with same max PAS score
    vocal_folds_severity_rating_max = case_when(
      max_pas >= 5 | max_pas <= 6 ~ max({{vocal_folds_severity_rating}})),
    subglottis_severity_rating_max = case_when(max_pas >= 7 | max_pas <= 8 ~ max({{subglottis_severity
  ) |>
  dplyr::slice(1) |>
  mutate(safety_grade = case_when(
    max_pas == 1 | max_pas == 2 ~ 0,
    freq_3_4 == 1 & max_pas == 3 | max_pas == 4 ~ 0, # PAS 3-4, single event
    freq_3_4 > 1 & max_pas == 3 | max_pas == 4 ~ 1, # PAS 3-4, multiple events
    freq_5_6 == 1 & vocal_folds_severity_rating_max <= 25 & max_pas == 5 | max_pas == 6 ~ 1, # PAS 5-
    freq_5_6 == 1 & vocal_folds_severity_rating_max > 25 & max_pas == 5 | max_pas == 6 ~ 2, # PAS 5-6
    freq_5_6 > 1 & max_pas == 5 | max_pas == 6 ~ 2, # PAS 5-6, multiple events
    freq_7_8 == 1 & subglottis_severity_rating_max <= 25 & freq_5_6 < 1 & max_pas == 7 | max_pas == 8
    freq_7_8 == 1 & subglottis_severity_rating_max <= 25 & freq_5_6 >= 1 & max_pas == 7 | max_pas == 8
    freq_7_8 > 1 & pas_7_8_chronic == "intermittent" & subglottis_severity_rating_max <= 25 & max_pas
    freq_7_8 > 1 & pas_7_8_chronic == "chronic" | airway_invasion_multiple_consistencies == "yes" & su
    freq_7_8 >= 1 & pas_7_8_chronic == "intermittent" & subglottis_severity_rating_max > 25 & max_pas
    freq_7_8 > 1 & pas_7_8_chronic == "chronic" | airway_invasion_multiple_consistencies == "yes" & su
  )
  ) |>
  select(id, max_pas, safety_grade)

### Calculate EFFICIENCY Grade
df1 <- {{data}} |> # calculate efficiency groupings by consistency
  group_by(id) |>
  mutate(max_residue_score = max({{percent_pharyngeal_residue}})
  ) |>
  group_by(id, {{IDDSI}}) |>
  mutate(efficiency_group_1 = case_when({{percent_pharyngeal_residue}} < .10 ~ 1),
         efficiency_group_2 = case_when({{percent_pharyngeal_residue}} >= .10 & {{percent_pharyngeal_r
         efficiency_group_3 = case_when({{percent_pharyngeal_residue}} >= .34 & {{percent_pharyngeal_r
         efficiency_group_4 = case_when({{percent_pharyngeal_residue}} > .66 ~ 1),
         efficiency_group_1_freq = sum(efficiency_group_1, na.rm = T),
         efficiency_group_2_freq = sum(efficiency_group_2, na.rm = T),
         efficiency_group_3_freq = sum(efficiency_group_3, na.rm = T),
         efficiency_group_4_freq = sum(efficiency_group_4, na.rm = T),
  ) |>
  dplyr::slice(1) |>
  dplyr::select(id, {{IDDSI}}, max_residue_score, efficiency_group_1_freq,
                efficiency_group_2_freq, efficiency_group_3_freq, efficiency_group_4_freq)
```

```
  df2 <- df1 |> # count number of consistencies administered per participant
    group_by(id) |>
    summarise(number_of_consistencies = n())

  df3 <- inner_join(df1, df2)

  df_efficiency <- df3 |>
    group_by(id) |>
    mutate(efficiency_grade = case_when(max_residue_score < 10 ~ 0,
                                        max_residue_score >= 10 & max_residue_score <= 33 ~ 1,
                                        {{IDDSI}} == 7 & max_residue_score >= 34 & max_residue_score <=
                                        {{IDDSI}} == 0 | {{IDDSI}} == 4 & max_residue_score >= 34 & max
                                        efficiency_group_4_freq/number_of_consistencies > 0 ~ 3,
                                        efficiency_group_4_freq/number_of_consistencies == 1 ~ 4)
    ) |>
    dplyr::slice(1) |>
    dplyr::select(id, max_residue_score, efficiency_grade)

  # Combine safety and efficiency grades into 1 data set
  data_dfs <- inner_join(df_safety, df_efficiency)

  ### CALCULATE TOTAL DIGEST Scores
  data_dfs |>
    group_by(id) |>
    mutate(total_grade = case_when(safety_grade == 0 & efficiency_grade == 0 ~ 0,
                                   safety_grade == 0 & efficiency_grade == 1 ~ 1,
                                   safety_grade == 0 & efficiency_grade == 2 ~ 1,
                                   safety_grade == 0 & efficiency_grade == 3 ~ 2,
                                   safety_grade == 0 & efficiency_grade == 4 ~ 3,
                                   safety_grade == 1 & efficiency_grade == 0 ~ 1,
                                   safety_grade == 1 & efficiency_grade == 1 ~ 1,
                                   safety_grade == 1 & efficiency_grade == 2 ~ 2,
                                   safety_grade == 1 & efficiency_grade == 3 ~ 2,
                                   safety_grade == 1 & efficiency_grade == 4 ~ 3,
                                   safety_grade == 2 & efficiency_grade == 0 ~ 2,
                                   safety_grade == 2 & efficiency_grade == 1 ~ 2,
                                   safety_grade == 2 & efficiency_grade == 2 ~ 2,
                                   safety_grade == 2 & efficiency_grade == 3 ~ 3,
                                   safety_grade == 2 & efficiency_grade == 4 ~ 3,
                                   safety_grade == 3 & efficiency_grade == 0 ~ 3,
                                   safety_grade == 3 & efficiency_grade == 1 ~ 3,
                                   safety_grade == 3 & efficiency_grade == 2 ~ 3,
                                   safety_grade == 3 & efficiency_grade == 3 ~ 3,
                                   safety_grade == 3 & efficiency_grade == 4 ~ 4,
                                   safety_grade == 4 & efficiency_grade == 0 ~ 3,
                                   safety_grade == 4 & efficiency_grade == 1 ~ 3,
                                   safety_grade == 4 & efficiency_grade == 2 ~ 3,
                                   safety_grade == 4 & efficiency_grade == 3 ~ 4,
                                   safety_grade == 4 & efficiency_grade == 4 ~ 4))
}
```

Below we can call the function `digest_fees_function()` to calculate the scores automatically. However, keep in mind that you need to specify your variable names below. For example, my PAS variable was named

"pen-asp" and my data was called "data_frame" so I had to specify these in the function.

```
digest_fees_function(
  data = data_frame,
  id = id,
  IDDSI = IDDSI,
  pas = pen_asp,
  percent_pharyngeal_residue,
  vocal_folds_severity_rating,
  subglottis_severity_rating
)
#> # A tibble: 20 x 6
#> # Groups:   id [20]
#>       id max_pas safety_grade max_residue_score efficiency_grade total_grade
#>    <int>   <dbl>        <dbl>              <dbl>            <dbl>       <dbl>
#>  1     1       7            2               30.5                1           2
#>  2     2       5            2               23.4                1           2
#>  3     3       5            2               33.8                3           3
#>  4     4       5            2               43.3                3           3
#>  5     5       7            2               38.4                3           3
#>  6     6       5            2               22.9                1           2
#>  7     7       5            2               37.2                3           3
#>  8     8       5            2               74.3                3           3
#>  9     9       6            1               25.7                1           1
#> 10    10       7            2               35.9                3           3
#> 11    11       7            2               43.6                3           3
#> 12    12       7            2               26.7                1           2
#> 13    13       6            1               40.5                3           2
#> 14    14       6            1               25.4                1           1
#> 15    15       7            3               44.2                3           3
#> 16    16       6            1               32.9                1           1
#> 17    17       6            1               36.1                3           2
#> 18    18       7            2               32.6                1           2
#> 19    19       7            2               46.4                3           3
#> 20    20       7            2               22.1                1           2
```