naCountOnMerge

Siwanart Ma

2024-12-20

Initial setting up

read the merged csv file

```
df = read.csv("/scratch_tmp/grp/msc_appbio/DCDM_group6/outputs/tables/Merged_df_final.csv")
df = df[, -1]
head(df)
```

```
##       analysis_id gene_accession_id gene_symbol mouse_strain  mouse_life_stage
## 1 0003b50qbwu5484       MGI:2444120       Spag4        C57BL       Early adult
## 2 0003klt8412603i       MGI:2385619      Klhl12        C57BL Middle aged adult
## 3 00052qirbv4o3tm       MGI:1098826        Coq4        C57BL       Early adult
## 4 00062gd65s37385       MGI:1917237      Spaca9        C57BL       Early adult
## 5 0007p3a8djq350z       MGI:2448580       Myo3b        C57BL       Early adult
## 6 0007psb08ytc999       MGI:2155705      P2ry14        C57BL        Late adult
##         parameter_id
## 1  ESLIM_005_001_702
## 2 KMPCLA_CSD_008_001
## 3  M-G-P_006_001_026
## 4   IMPC_IMM_033_002
## 5  M-G-P_016_001_004
## 6   IMPC_GEO_009_002
##                                             parameter_name     pvalue
## 1                                          Lean/Body weight 0.70844160
## 2                                 Coat - color pattern - back 0.42109680
## 3                                         Fusion of vertebrae 0.08293449
## 4 NaÃ\u0083Â¯ve CD8+ T cells - % of live leukocytes (Panel A) 0.30371250
## 5                                              Haematocrit 0.17170630
## 6                                               Hemorrhage 0.85469090
```

Count NA values in each column

```
na_counts = colSums(is.na(df))
print(na_counts)
```

```
##       analysis_id gene_accession_id       gene_symbol      mouse_strain
##               485               505               505               512
##  mouse_life_stage      parameter_id    parameter_name            pvalue
##               487               503               535               536
```

Read sop

```
sop = read.csv("/scratch_tmp/grp/msc_appbio/DCDM_group6/metadata/IMPC_SOP.csv")
sop
```

```
##           dataField dataType minValue maxValue
```

```
## 1       analysis_id   String       15       15
## 2 gene_accession_id   String        9       11
## 3       gene_symbol   String        1       13
## 4      mouse_strain   String        3        5
## 5  mouse_life_stage   String        4       17
## 6      parameter_id   String       15       18
## 7    parameter_name   String        2       74
## 8            pvalue    Float        0        1
##                                      exampleValues
## 1                                    0a1yei0604s8275
## 2                                       MGI:2679336
## 3                                             Satb2
## 4                                             C57BL
## 5                                             E15.5
## 6                                    IMPC_GRS_009_001
## 7 Forelimb and hindlimb grip strength measurement mean
## 8                                         0.8004226
##
## 1                                                                    Unique 15 character string containing al
## 2 Unique alphanumeric character string that is used to unambiguously identify a particular record in
## 3                                                                                                  Offi
## 4                                                                                       Name of mouse stra
## 5             Developmental stage of mice being analysed. Values are E12.5; E15.5; E18.5; E9.5; Early
## 6
## 7
## 8
```

Set the ranges

```
allFields = sop[,1]
allMins = sop[,3]
allMaxs = sop[,4]
print(allFields)
```

```
## [1] "analysis_id"       "gene_accession_id" "gene_symbol"
## [4] "mouse_strain"      "mouse_life_stage"  "parameter_id"
## [7] "parameter_name"    "pvalue"
```

```
print(allMins)
```

```
## [1] 15  9  1  3  4 15  2  0
```

```
print(allMaxs)
```

```
## [1] 15 11 13  5 17 18 74  1
```

Count how many NA values are there in each row

```
# Identify String and p-value columns based on SOP
string_columns <- allFields[sop$dataType == "String"]
pvalue_columns <- allFields[sop$dataType != "String"]

# Filter for valid columns in df
string_columns <- intersect(string_columns, colnames(df))
pvalue_columns <- intersect(pvalue_columns, colnames(df))

# Create a data frame to store NA counts
na_counts <- data.frame(
```

```
  analysis_id = if ("analysis_id" %in% colnames(df)) df$analysis_id else seq_len(nrow(df)),
  na_count_string = rowSums(is.na(df[string_columns])),
  na_count_pvalue = rowSums(is.na(df[pvalue_columns]))
)

# Filter rows where total NA count is greater than 1
filtered_na_counts <- na_counts[
  na_counts$na_count_string + na_counts$na_count_pvalue > 1,
]

# Output the filtered data frame
print(head(filtered_na_counts))
```

```
##          analysis_id na_count_string na_count_pvalue
## 162             <NA>               5               1
## 386             <NA>               5               1
## 749  01dg3wp1kqyh88u               2               0
## 1481 0339irju94z69yi               5               0
## 1781            <NA>               6               1
## 2032            <NA>               7               1
```

How many data have at least 1 NA

```
print(nrow(filtered_na_counts))
```

```
## [1] 765
```

‘‘‘