Session 4 Exercises: Data Analysis in R and Python

Exercise 1: R

1. Create a new folder in your workspace and name it ‘Session 4’
2. Create a new code workbook in the Session 4 folder and name it ‘Malnutrition Data Analysis’
3. Change the environment to ‘profile-high-memory’

Graphical user interface, application

Description automatically generated

1. Import the dataset titled, ‘analytic\_dataset\_17Jun22.’ You can do this by searching for it in the search bar after clicking the ‘Import dataset’ button:

Graphical user interface, application, Teams

Description automatically generated

1. Rename the analytics\_dataset\_17June22 analytic\_dataset.
2. Select analytic\_dataset and select “new transform” then select “R code.” Name the transform exercise\_1.
3. Copy the code below into the global code section of the code workbook:

Graphical user interface, application, Word

Description automatically generated

Global code:

library(gtsummary)

library(dplyr)

# Use dplyr to select data elements of interest

local\_df <- analytic\_dataset %>%

dplyr::select(age, age\_group, gender, race\_ethnicity, BMI, BMI\_category,

CCI, CCI\_category, diabetes, MI, CHF, PVD, CVD, dementia, pulmonary,

AIRD, liver, PUD, paralysis, renal, cancer, HIV, smoking, COVID\_death, IMV,

ARDS, ECMO, HAPI, LOS, malnutrition\_status)

# Set factors: this defines the order of appearance

local\_df$malnutrition\_status <- factor(local\_df$malnutrition\_status, levels = c("No Documented Malnutrition",

"Hx of Malnutrition", "Hospital-Acquired Malnutrition"))

local\_df$age\_group <- factor(local\_df$age\_group, levels = c("<30", "30-49",

"50-64", "65+"))

local\_df$gender <- factor(local\_df$gender, levels = c("FEMALE", "MALE"))

local\_df$race\_ethnicity <- factor(local\_df$race\_ethnicity, levels = c("White Non-Hispanic",

"Black or African American Non-Hispanic", "Hispanic or Latino Any Race",

"Other", "Unknown"))

local\_df$BMI\_category <- factor(local\_df$BMI\_category, levels = c("<18.5",

"18.5-24.9", "25.0-29.9", "30+"))

local\_df$CCI\_category <- factor(local\_df$CCI\_category, levels = c("<1",

"1-3", ">3"))

# Set references

local\_df$race\_ethnicity=relevel(as.factor(local\_df$race\_ethnicity),ref="White Non-Hispanic")

local\_df$gender=relevel(as.factor(local\_df$gender),ref="FEMALE")

local\_df$malnutrition\_status=relevel(as.factor(local\_df$malnutrition\_status),ref="No Documented Malnutrition")

1. Copy the code below into the exercise\_1 transformation. Toggle the ‘save as dataset’ button and run the transformation.

R code:

descriptive\_statistics\_by\_malnutrition\_status <- function(analytic\_dataset) {

# Use gtsummary to create descriptive statistics, stratified by

# malnutrition status

table1 <- local\_df %>%

tbl\_summary(by = malnutrition\_status) %>%

add\_p()

print(table1)

# Convert to tibble to coerce to a dataframe

table1 <- as\_tibble(table1, col\_labels = FALSE)

# Return summary stats. Note: column labels are excluded because they

# cannot be coerced into a data frame in this platform.

return(table1)

}

Exercise 2: Python

1. In the Malnutrition Data Analysis code workbook, select new transform and click ‘Python code.’
2. Copy the code below into the new transform and toggle ‘Save as dataset.’

def processed\_dataset(analytic\_dataset, manifest\_safe\_harbor):

"Visualizes the distribution of numeric values and the density of non-numeric values."

# data\_partner\_id field is mandatory in input dataframe

if 'data\_partner\_id' not in manifest\_safe\_harbor.columns:

raise Exception('Input dataframe must contain "data\_partner\_id" column')

# cdm\_name will be joined in later, so drop if it exists

df = analytic\_dataset.drop('cdm\_name')

# Convert non-numeric columns to binary (null/notnull)

nonnumeric\_cols = [c for c,d in df.dtypes if 'int' not in d]

for c in nonnumeric\_cols:

df = df.withColumn(c, F.when(F.col(c).isNotNull(),1).otherwise(0))

# Aggregate mean values at the site level

cols = list(df.columns)

cols.remove('data\_partner\_id')

df = (df

.groupBy('data\_partner\_id')

.agg(\*[F.avg(c).alias(c) for c in cols])

.toPandas()

)

# Score the overall density of indicator columns

numeric\_df = df.select\_dtypes('number')

indicator\_cols = numeric\_df.columns[numeric\_df.max(axis=0)<1]

print('indicator\_cols',indicator\_cols)

df.insert(0, 'overall', df[indicator\_cols].sum(axis=1))

df['overall'] = df['overall'] / df['overall'].max()

cols = cols + ['overall']

for c in cols:

diff\_to\_median\_col = c + "\_diff\_to\_median"

dmd\_col = c + "\_dmd"

# Calculate median absolute deviation (MAD)

median = df[c].median()

print(c, 'median =\t', median)

df[diff\_to\_median\_col] = (df[c] - median)

MAD = df[diff\_to\_median\_col].abs().median()

print(c, 'MAD =\t', MAD)

# Calculate how many MADs value is from median

df[dmd\_col] = (df[diff\_to\_median\_col]/MAD) if MAD > 0 else 0

df.drop(columns=[diff\_to\_median\_col], errors='ignore', inplace=True)

# Join in cdm\_name for visualizations

sites = (manifest\_safe\_harbor

.select(F.col('data\_partner\_id').cast('int'),'cdm\_name')

.toPandas()

)

return df.merge(sites, on='data\_partner\_id')

#################################################

## Global imports and functions included below ##

#################################################

import pyspark.sql.functions as F

from pyspark.sql.functions import abs as \_abs

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.patches import Patch

1. Import the manifest\_safe\_harbor table from the data catalog, which is located in the De-Identified Data folder.
2. Select new transform and click ‘Python code.’
3. Copy the code below into the new transform and toggle ’Save as dataset.’

def Standardized\_Density(processed\_dataset):

# Create heatmap of grades

df = (processed\_dataset

.set\_index('data\_partner\_id')

.drop(columns=['cdm\_name'])

)

# Sort rows by overall site density

cols = [c for c in df.columns if '\_dmd' not in c]

df = df[cols]

df = df.divide(df.max()).fillna(0).sort\_values('overall')

# Sort columns by sum

df = df[df.sum().sort\_values(ascending=False).index]

# Plot clustermap

cbar\_kws = dict(label='Standardized values/density')#,location="top"

sns.heatmap(df, cmap=sns.color\_palette("Blues"), cbar\_kws=cbar\_kws, xticklabels=True, yticklabels=True)

plt.title('Dataset Density by Site', fontsize=32)

plt.xlabel('Columns', fontsize=20)

plt.ylabel('Sites', fontsize=20)

plt.show()

return df

#################################################

## Global imports and functions included below ##

#################################################

import pyspark.sql.functions as F

from pyspark.sql.functions import abs as \_abs

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.patches import Patch

1. Go to the import tab in the transform and change the input type to Pandas dataframe.

Graphical user interface, text, application, chat or text message

Description automatically generated

Code for python transform courtesy of Evan French. Source: https://unite.nih.gov/workspace/module/view/latest/ri.workshop.main.module.3ab34203-d7f3-482e-adbd-f4113bfd1a2b?id=KO-9901C7E&view=focus