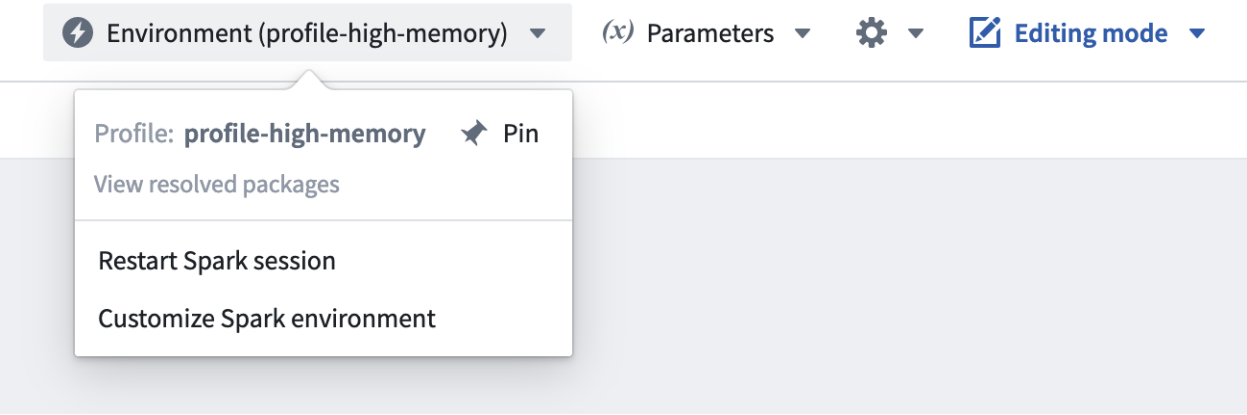
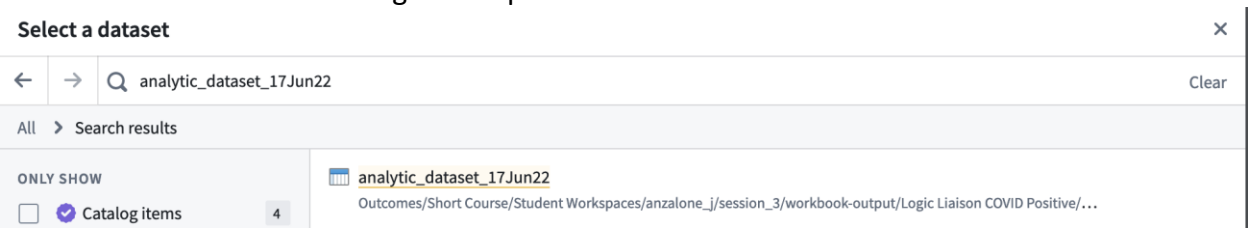
Session 4 Exercises: Data Analysis in R and Python

Exercise 1: R

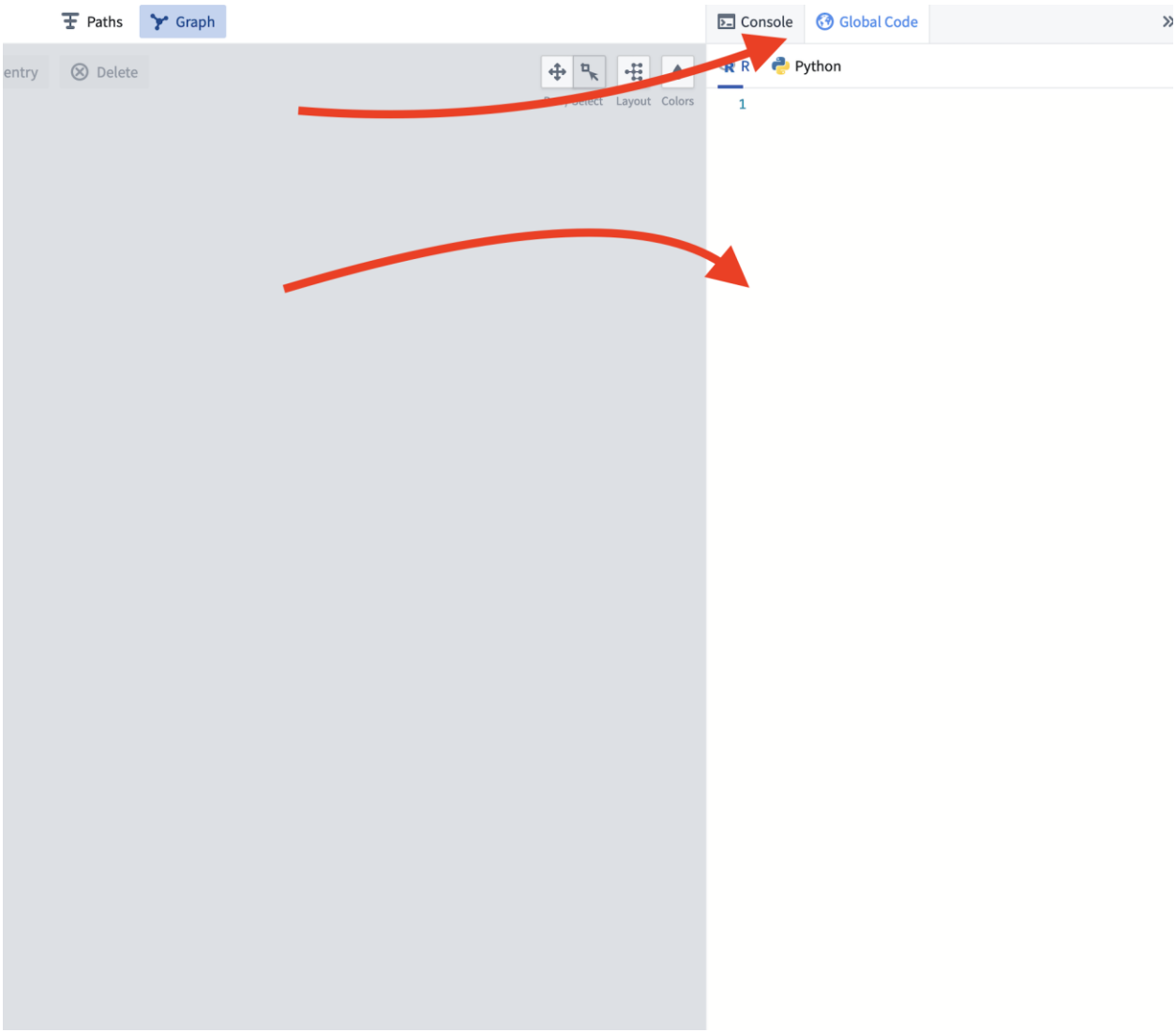
1. Create a new folder in your workspace and name it ‘Session 4’
2. Create a new code workbook in the Session 4 folder and name it ‘Malnutrition Data Analysis’
3. Change the environment to ‘profile-high-memory’



4. 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:



5. Rename the analytics_dataset_17June22 analytic_dataset.
6. Select analytic_dataset and select “new transform” then select “R code.” Name the transform exercise_1.
7. Copy the code below into the global code section of the code workbook:



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")
```

8. 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
```

3. Import the manifest_safe_harbor table from the data catalog, which is located in the De-Identified Data folder.
4. Select new transform and click ‘Python code.’
5. 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)
```

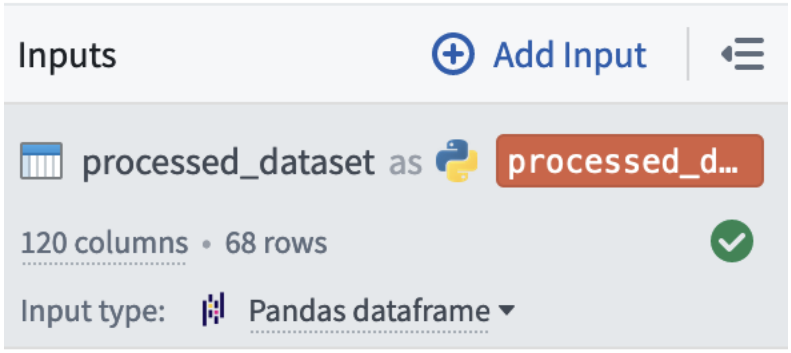
```
    plt.title('Dataset Density by Site', fontsize=32)
    plt.xlabel('Columns', fontsize=20)
    plt.ylabel('Sites', fontsize=20)
    plt.show()

    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
```

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



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