Time to Event Analysis Imputation: Efficacy Endpoints

List of Figures

1 Missing Values Heatmap 4
2 Missing Values Heatmap - Post Imputation 6

List of Tables

1 Baseline Characteristics of Imputed and Original Datasets 9

Load the Data

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.experimental import enable_iterative_imputer  # noqa
from sklearn.impute import IterativeImputer


adtte = pd.read_csv(r"C:\Users\PGr\Documents\Experiments\Python\Pharma_Analysis\Data\adtte.csv")

#Create Event - If no Censor then they must have and Event
adtte["EVT"] = np.where(adtte["CNSR"] == 1, 0, 1)

# Convert avl to months
adtte["AVAL"] = adtte["AVAL"] /30.475

adtte = adtte[adtte["AVALU"] != "COUNT"].copy()


columns = ['USUBJID', 'AGE', 'SEX', 'RACE', 'ETHNIC', 'TRT01P', 'PARAM','PARAMCD', 'EVT', 'CNSR',
adtte_i = adtte[columns].copy()

# Convert to numeric


columns_numeric = ["AGE", "SEX", "RACE", "AVAL"]

for col in columns_numeric:
    adtte_i[col] = pd.Categorical(adtte_i[col]).codes
```

Missing Value Creation

```python
import random

def add_random_na_by_subject(df, prob=0.2):
    """
    Randomly sets subject-level columns (AGE, SEX, RACE, etc.) to NaN
    for all rows belonging to the same subject.

    prob: probability that a given column (per subject) will be set to NaN
    """
    df_copy = df.copy()

    # All columns except ID
    eligible_cols = [col for col in df_copy.columns if col != "USUBJID"]

    for subject in df_copy["USUBJID"].unique():
        subj_mask = df_copy["USUBJID"] == subject

        # Decide which subject-level cols become NaN for this subject
        cols_to_nan = [col for col in eligible_cols if random.random() < prob]

        # Set entire column to NaN for all rows of this subject
        df_copy.loc[subj_mask, cols_to_nan] = np.nan

    return df_copy


cols_4_random = ["USUBJID", "AGE", "SEX", "RACE", "AVAL"]

# Apply the function
adtte_m = add_random_na_by_subject(adtte_i[cols_4_random], prob=0.2)
```

Visualisation of the missings

```
plt.figure()
sns.heatmap(adtte_m.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values Before Imputation")
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```

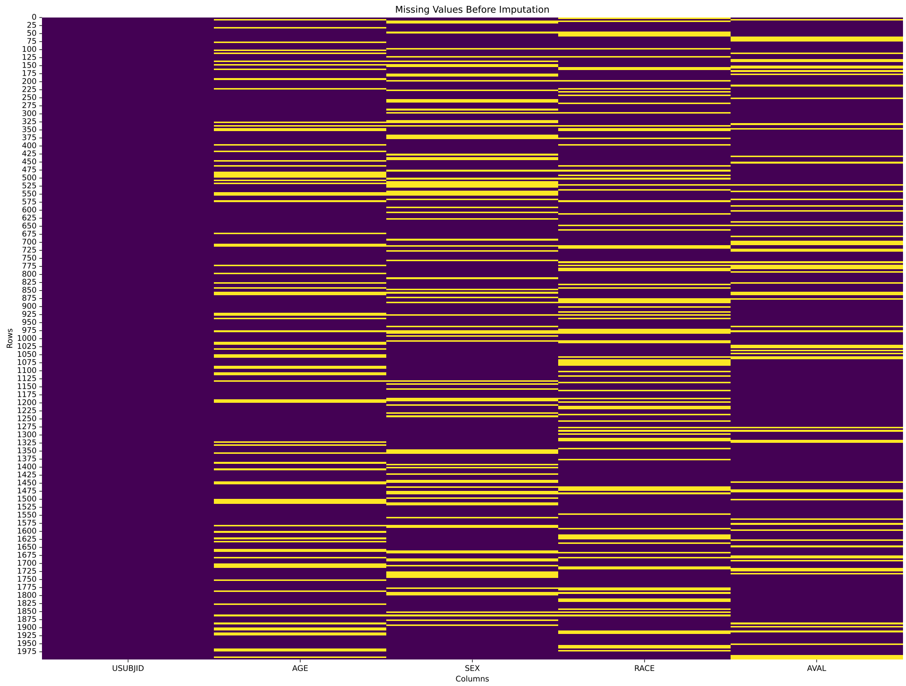


Figure 1: Missing Values Heatmap

Imputation

We will use Multiple Imputation with Chained Equations with 100 iterations to impute the data. We will use the package - "miceforest" for this.

```
import miceforest as mf


cols_to_impute = ["AGE", "SEX", "RACE", "AVAL"]

# Extract ID column separately
usubjid = adtte_m["USUBJID"].reset_index(drop=True)

# Extract columns to impute
df_to_impute = adtte_m[cols_to_impute].reset_index(drop=True)

# Initialize and run miceforest
kernel = mf.ImputationKernel(df_to_impute, random_state=1991)
kernel.mice(100)
```

```
C:\Users\PGr\DOCUME~1\EXPERI~1\Python\PHARMA~1\myvenv\Lib\site-packages\miceforest\imputed_data.p
  self.imputation_values[variable].loc[:, (iteration, dataset)] = newitem
C:\Users\PGr\DOCUME~1\EXPERI~1\Python\PHARMA~1\myvenv\Lib\site-packages\miceforest\imputed_data.p
  self.imputation_values[variable].loc[:, (iteration, dataset)] = newitem
C:\Users\PGr\DOCUME~1\EXPERI~1\Python\PHARMA~1\myvenv\Lib\site-packages\miceforest\imputed_data.p
  self.imputation_values[variable].loc[:, (iteration, dataset)] = newitem
C:\Users\PGr\DOCUME~1\EXPERI~1\Python\PHARMA~1\myvenv\Lib\site-packages\miceforest\imputed_data.p
  self.imputation_values[variable].loc[:, (iteration, dataset)] = newitem
```

```
# Get completed data and reattach ID
adtte_imp = kernel.complete_data()
adtte_imp.insert(0, "USUBJID", usubjid)



adtte_final = adtte_i.copy().reset_index(drop=True)

# Step 2: Replace the imputed columns in the original dataset
for col in cols_to_impute:
    adtte_final[col] = adtte_imp[col]
```

Visualisation of Missing Data - Post Imputation

```
plt.figure()
sns.heatmap(adtte_imp.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values After Imputation")
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```

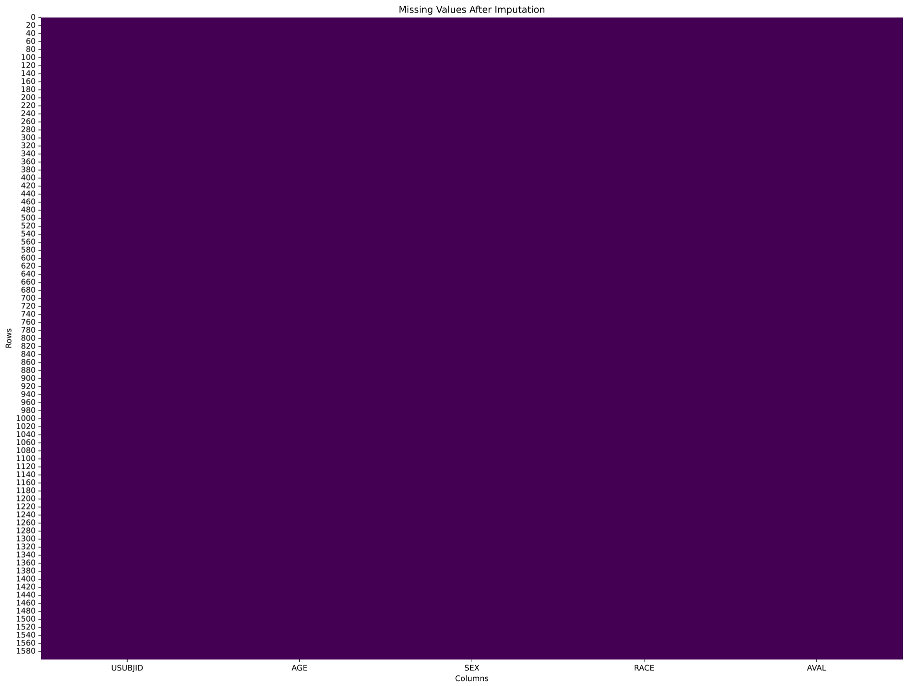


Figure 2: Missing Values Heatmap - Post Imputation

```
from tableone import TableOne, load_dataset
import pandas as pd

#Create a column in each table called "Method

adtte_i["Method"] = "Original"
adtte_final["Method"] = "Imputed"



# make a big data frame
```

```
adtte_combined = pd.concat([adtte_i, adtte_final])

# fix the column contexts

adtte_combined["SEX"] = np.where(adtte_combined["SEX"] == 1, "Male", "Female")
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == 2, "Black or African American", adtte
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == "5.0", "White", adtte_combined["RACE"
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == "1.0", "Asian", adtte_combined["RACE"
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == "0.0", "American Indian Or Alaska Nat
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == "3.0", "Native Hawaiian Or Other Paci
adtte_combined["RACE"] = np.where(adtte_combined["RACE"] == "4.0", "Multipler", adtte_combined["R
```

Baseline Characteristics

This section is done in R because there are better packages available to make nicer tabls

```
if(!require(reticulate)){install.packages("reticulate");library(reticulate)}
```

```
Lade nötiges Paket: reticulate
```

```
if(!require(gtsummary)){install.packages("gtsummary");library(gtsummary)}
```

```
Lade nötiges Paket: gtsummary
```

```
if(!require(kableExtra)){install.packages("kableExtra");library(kableExtra)}
```

```
Lade nötiges Paket: kableExtra
```

```
df <- py$adtte_combined
```

```
Warning in py_to_r.pandas.core.frame.DataFrame(x): index contains duplicated
values: row names not set
```

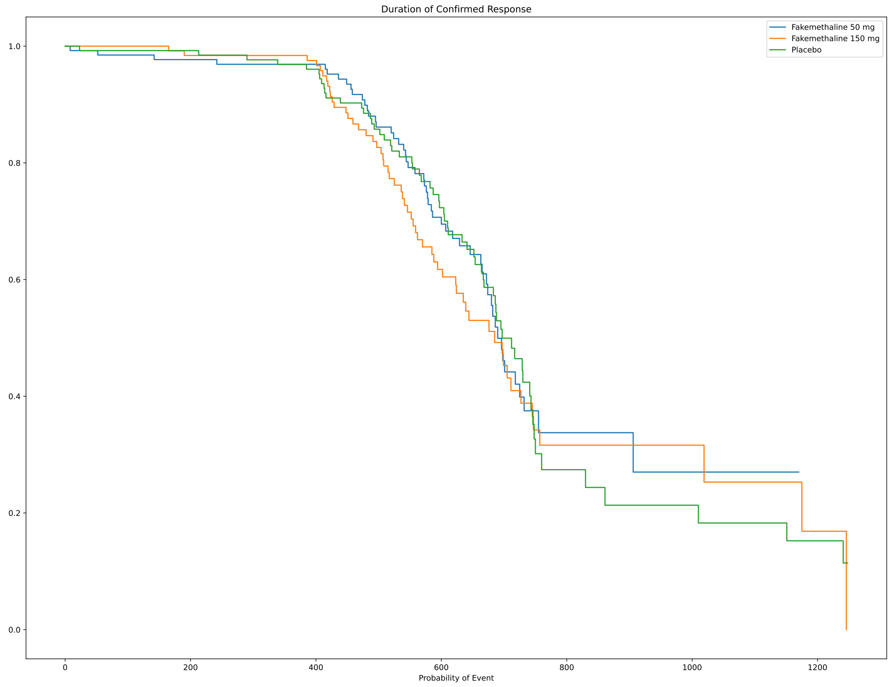
```
tbl <- df %>%
  select(AGE, SEX, RACE, TRT01P, Method) %>%
  tbl_summary(
    by = Method,
    sort = all_categorical() ~ "alphanumeric",
    type = list(
      AGE ~ "continuous2"
    ),
    statistic = list(
      all_continuous() ~ c("{mean}", "{median} ({p25}, {p75})", "{min}, {max}")
    ),
    label = list(
      AGE ~ "Age",
      SEX ~ "Sex",
      RACE ~ "Race",
      TRT01P ~ "Treatment"
    )
  ) %>%
  add_overall(last = TRUE) %>%
  bold_labels() %>%

  as_kable(booktabs = TRUE)

tbl
```

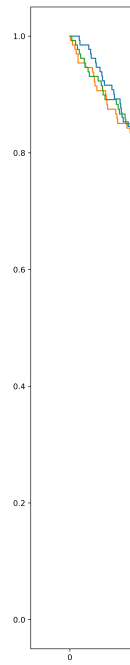
Table 1: Baseline Characteristics of Imputed and Original Datasets

Characteristic	**Imputed** N = 1,600	**Original** N = 1,600	**Overall** N = 3,200
Age			
Mean	14	14	14
Median (Q1, Q3)	13 (8, 19)	13 (8, 18)	13 (8, 19)
Min, Max	0, 37	0, 37	0, 37
Sex			
Female	890 (56%)	924 (58%)	1,814 (57%)
Male	710 (44%)	676 (42%)	1,386 (43%)
Race			
American Indian Or Alaska Native	103 (6.4%)	100 (6.3%)	203 (6.3%)
Asian	806 (50%)	832 (52%)	1,638 (51%)
Black or African American	382 (24%)	364 (23%)	746 (23%)
Multipler	7 (0.4%)	4 (0.3%)	11 (0.3%)
Native Hawaiian Or Other Pacific Islander	0 (0%)	4 (0.3%)	4 (0.1%)
White	302 (19%)	296 (19%)	598 (19%)
Treatment			
Fakemethaline 150 mg	528 (33%)	528 (33%)	1,056 (33%)
Fakemethaline 50 mg	536 (34%)	536 (34%)	1,072 (34%)
Placebo	536 (34%)	536 (34%)	1,072 (34%)

Time to Event Analysis - for the Imputed Data

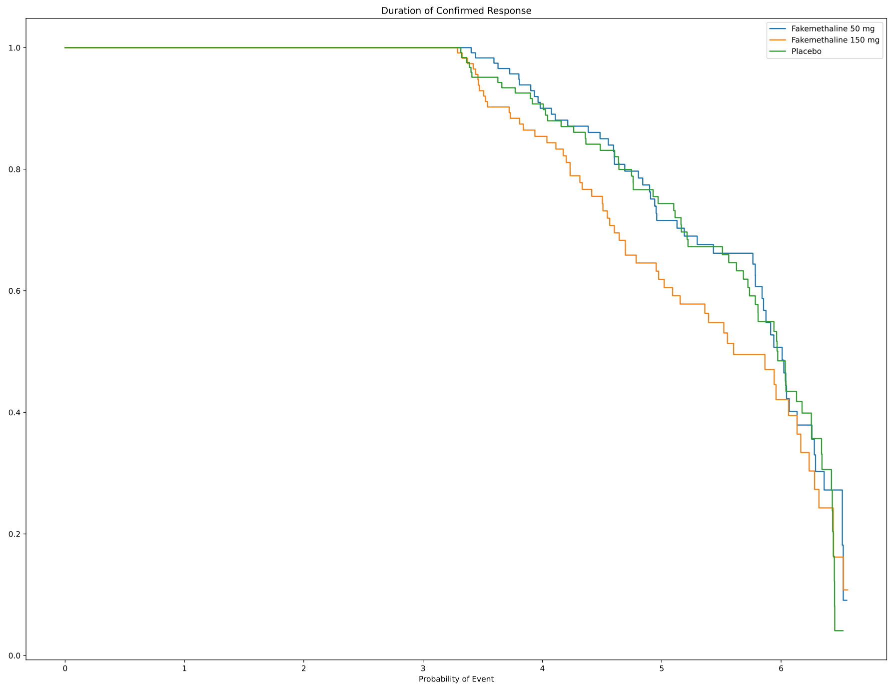
```
from lifelines import KaplanMeierFitter



param = adtte_final["PARAM"].unique()
treat  = adtte_final["TRT01P"].unique()

for par in param:

  data = adtte_final[adtte_final["PARAM"] == par]

  kmf = KaplanMeierFitter()

  T = data["AVAL"]
  E = data["CNSR"]


  trt_50mg = data["TRT01P"] == 'Fakemethaline 50 mg'
  trt_150mg = data["TRT01P"] == 'Fakemethaline 150 mg'
  placebo = data["TRT01P"] == 'Placebo'

  plt.clf()
  ax = plt.subplot(111)

  kmf.fit(T[trt_50mg], event_observed=E[trt_50mg], label= "Fakemethaline 50 mg")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  kmf.fit(T[trt_150mg], event_observed=E[trt_150mg], label="Fakemethaline 150 mg")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  kmf.fit(T[placebo], event_observed=E[placebo], label="Placebo")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  plt.title(par)
  plt.xlabel("Time (Months)")
  plt.xlabel("Probability of Event")
  plt.show()
```

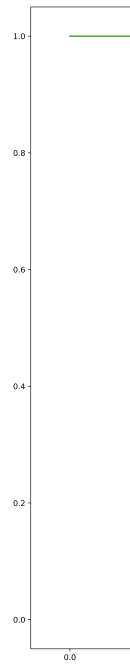

Duration of Confirmed Response
Fakemethaline 50 mg
Fakemethaline 150 mg
Placebo
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
Probability of Event


1.0
0.8
0.6
0.4
0.2
0.0
0

Time to Event Analysis - for the "Original Data"

```python
from lifelines import KaplanMeierFitter



param = adtte["PARAM"].unique()
treat  = adtte["TRT01P"].unique()

for par in param:

  data = adtte[adtte["PARAM"] == par]

  kmf = KaplanMeierFitter()

  T = data["AVAL"]
  E = data["CNSR"]


  trt_50mg = data["TRT01P"] == 'Fakemethaline 50 mg'
  trt_150mg = data["TRT01P"] == 'Fakemethaline 150 mg'
  placebo = data["TRT01P"] == 'Placebo'

  plt.clf()
  ax = plt.subplot(111)

  kmf.fit(T[trt_50mg], event_observed=E[trt_50mg], label= "Fakemethaline 50 mg")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  kmf.fit(T[trt_150mg], event_observed=E[trt_150mg], label="Fakemethaline 150 mg")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  kmf.fit(T[placebo], event_observed=E[placebo], label="Placebo")
  kmf.plot_survival_function(ax=ax, ci_show = False)
  plt.title(par)
  plt.xlabel("Time (Months)")
  plt.xlabel("Probability of Event")
  plt.show()
```





When comparing graphs from original data and the imputed data can see a difference in the reults.