

# Circus of Python in Real World Evidence

Sucharita Pilli  
(Novo Nordisk)

PHUSE US Connect 2022, Atlanta

2<sup>nd</sup> May 2022

[scrp@novonordisk.com](mailto:scrp@novonordisk.com)



# Disclaimer

This presentation is intended to exhibit the theoretical concept of Python and its framework and no data or confidential information belonging to Novo Nordisk has been used in this presentation

- ❖ Python experimentation in Real World Evidence (RWE)
- ❖ Packages required
- ❖ Methods to analyze the data
- ❖ How to perform summary statistics
- ❖ How to validate the two DataFrames for equal observations
- ❖ How to validate the two DataFrames for unequal observations and export the validation dataframe?
- ❖ Conclusion

# Agenda

# Python experimentation in Real World Evidence (RWE)



Learnt Python language from Datacamp - [Learn R, Python & Data Science Online | DataCamp](#)



How can I implement in a short period

# Python

- ❖ Python was named after British comedy group Monty Python's Flying Circus
- ❖ It is a high-level programming language that can be used for doing data analysis and data visualization



# Packages required:

**pandas:** It is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool

**pyreadstat:** Python package to read sas, spss and stata files into pandas data frames

**saspy:** SASPy is an open-source Python package that allows Python coders to access SAS data and analytics capabilities

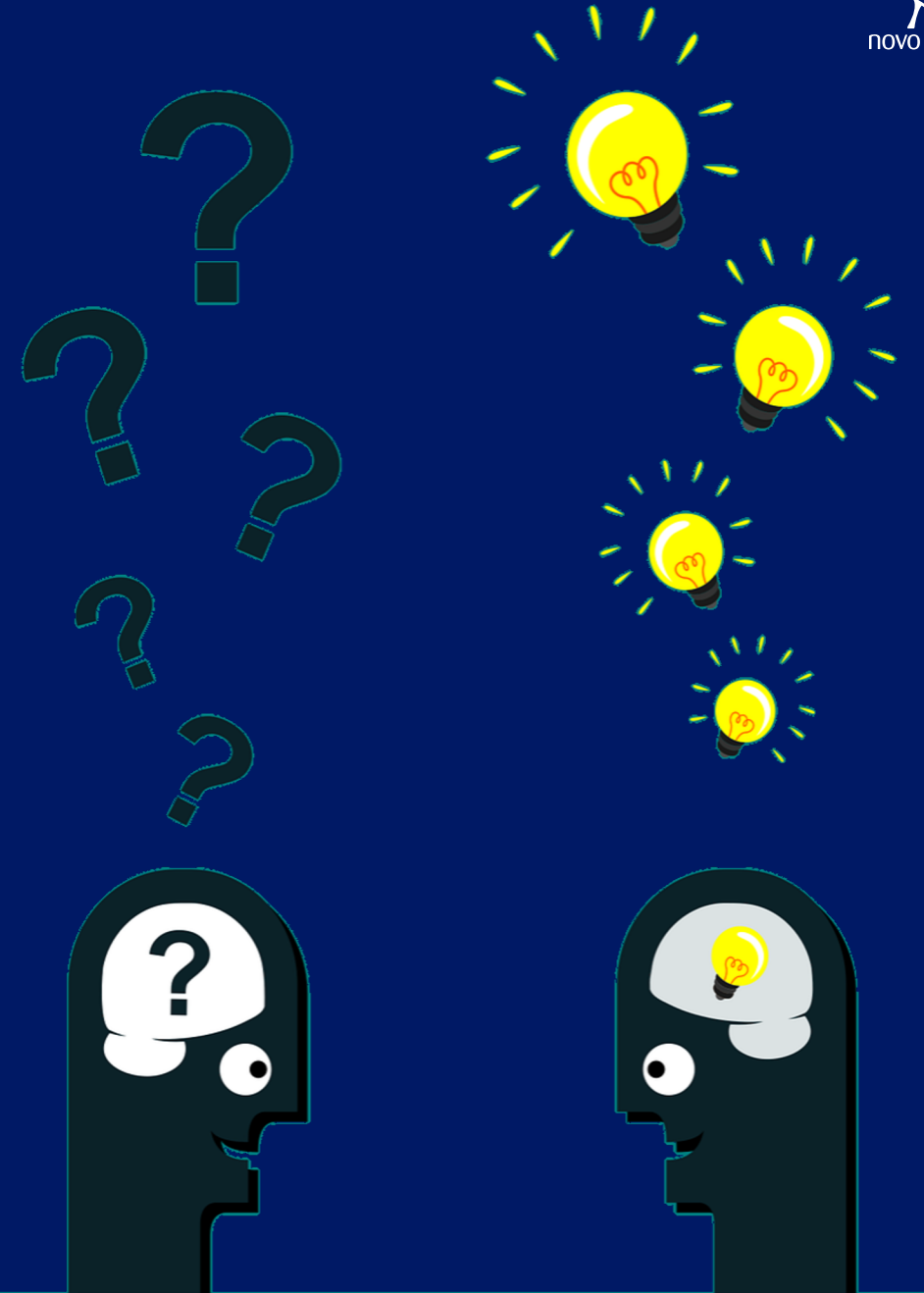
**pyjanitor:** It is a Python implementation of the R package janitor, and provides a clean API for cleaning data

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <b>Install packages</b> | example: pip install pyreadstat                                               |
| <b>Import</b>           | import pandas                                                                 |
| <b>Import</b>           | import pyreadstat as pyr<br>adsl,meta=pyr.read_sas7bdat("path/adsl.sas7bdat") |
| <b>Import</b>           | import saspy                                                                  |
| <b>Import</b>           | import janitor                                                                |

# Methods to analyze the data

| groupby_agg()                                                                                                    | count()                                                                                 | nunique()                                                                                                                   | agg()                                                          | describe()                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Calculate the denominator/ begin using janitor package                                                           | Count non-NA cells for each column or row. The values None, NaN, NaT are considered NA. | Count distinct observations. Can be used for a <u>dataframe</u> or a <u>series</u> . By default, it exclude the NaN values. | Aggregate using one or more operations over the specified axis | Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values |
| data.groupby_agg(<br>by="column",<br>agg="count",<br>agg_column_name="column",<br>new_column_name="new-column" ) | Data.count(<br>axis=0,<br>level=None,<br>numeric_only=False)                            | Data.nunique(axis=0,<br>dropna=True)                                                                                        | Data.agg(<br>func=None,<br>axis=0,<br>*args,<br>**kwargs)      | DataFrame.describe(<br>percentiles=None,<br>include=None,<br>exclude=None,<br>datetime_is_numeric=False)                                         |

# How to perform summary statistics



# Mock shell

## Adverse events - overview - in-study observation period - full analysis set

|                                  | Serious |       |    | Non-serious |        |    | Total |        |    |
|----------------------------------|---------|-------|----|-------------|--------|----|-------|--------|----|
|                                  | N       | (%)   | E  | N           | (%)    | E  | N     | (%)    | E  |
| Number of patients               |         |       |    |             |        |    | xxx   |        |    |
| Adverse events                   | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Severity                         |         |       |    |             |        |    |       |        |    |
| Mild                             | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Moderate                         | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Severe                           | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Causality                        |         |       |    |             |        |    |       |        |    |
| Probable                         | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Possible                         | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Unlikely                         | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Outcome                          |         |       |    |             |        |    |       |        |    |
| Recovered/resolved               | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Recovering/resolving             | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Recovered/resolved with sequelae | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Not recovered/not resolved       | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Fatal                            | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Unknown                          | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Action taken                     |         |       |    |             |        |    |       |        |    |
| Drug interrupted                 | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Drug withdrawn                   | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Dose reduced                     | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Dose increased                   | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |
| Dose not changed                 | x       | (x.x) | xx | xx          | (xx.x) | xx | xx    | (xx.x) | xx |

# Denominator calculation (1/2)

## Using janitor package: groupby\_agg

*#Denominator - FASFL*

```
ads11=ads11.groupby_agg(by="fasfl",  
                        agg="count",  
                        agg_column_name="USUBJID",  
                        new_column_name="total" )
```

|  | OBSDURW | total |
|--|---------|-------|
|  | 38.4286 | 779   |
|  | 39      | 779   |
|  | 37.4286 | 779   |
|  | 37      | 779   |
|  | 35.5714 | 779   |
|  | 37.4286 | 779   |
|  | 37.5714 | 779   |
|  | 36.8571 | 779   |

*Note: "Numbers in tables are randomly example numbers for illustration purposes"*

# Denominator calculation (2/2)

## Using nunique method

---

```
patients=ads11.groupby(  
    "fasfl")["USUBJID"].nunique().reset_index(  
        name="Total").assign(ord=1,  
                             col1="Number of patients").astype(str)
```

| Index | fasfl | total | ord | col1               |
|-------|-------|-------|-----|--------------------|
| 0     | Y     | 779   | 1   | Number of patients |

*Note: "Numbers in tables are randomly example numbers for illustration purposes"*

# Summarize categorical data

Distinct count

```
def stats(var,data,all1,order):

    #Number of patients - distinct
    n=adsl_adae.groupby(["AESER_c",var])["USUBJID"].nunique().reset_index(name='count')

    #Number of events
    eve=adsl_adae.groupby(["AESER_c",var])["USUBJID"].count().reset_index(name='event')

    # Merging both the count and events data
    all1=pd.merge(n,eve, on=["AESER_c",var], how="left").rename(columns={var:'col1'})
    all1.assign(ord=order)

    total=pd.concat([all1,adsl1[["total"]]], axis=1, join='inner')

    total["percent"]=total["count"].astype(str)+"("+round(
        (total["count"]/total["total"])*100,1).apply(str
        )+")"+total["event"].astype(str)

    total1=total[["AESER_c","ord","col1","percent"]].pivot(index=["ord","col1"],
        columns="AESER_c",values="percent"
        ).reset_index()

    return total1

asev1=stats(var="ASEV",data="asev",all1="asev1",order=3)
```

| Index | col1                             | col2      | col3         | total | ord |
|-------|----------------------------------|-----------|--------------|-------|-----|
| 0     | Number of patients               | nan       | nan          | 669   | 1   |
| 1     | Adverse events                   | 21(3.1)22 | 190(28.4)373 | nan   | 2   |
| 2     | Mild                             | 2(0.3)2   | nan          | nan   | 3   |
| 3     | Moderate                         | 11(1.6)11 | 58(8.7)83    | nan   | 3   |
| 4     | Severe                           | 9(1.3)9   | 2(0.3)2      | nan   | 3   |
| 5     | Possible                         | nan       | 70(10.5)99   | nan   | 4   |
| 6     | Probable                         | nan       | 71(10.6)124  | nan   | 4   |
| 7     | Unlikely                         | 21(3.1)22 | nan          | nan   | 4   |
| 8     | Not recovered/not resolved       | 3(0.4)3   | 35(5.2)44    | nan   | 5   |
| 9     | Recovered/resolved               | 16(2.4)16 | 170(25.4)309 | nan   | 5   |
| 10    | Recovered/resolved with sequelae | 2(0.3)2   | 1(0.1)1      | nan   | 5   |
| 11    | Recovering/resolving             | nan       | 10(1.5)19    | nan   | 5   |
| 12    | Dose not changed                 | 16(2.4)17 | nan          | nan   | 6   |
| 13    | Dose reduced                     | nan       | 17(2.5)20    | nan   | 6   |
| 14    | Drug interrupted                 | 3(0.4)3   | 10(1.5)12    | nan   | 6   |
| 15    | Drug withdrawn                   | nan       | 2(0.3)2      | nan   | 6   |
| 16    | N/A                              | nan       | 3(0.4)5      | nan   | 6   |
| 17    | Unknown                          | 2(0.3)2   | 2(0.3)2      | nan   | 6   |

Note: "Numbers in tables are randomly example numbers for illustration purposes"

[scrp@novonordisk.com](mailto:scrp@novonordisk.com)

# Statistical analysis for continuous variable

## Display all descriptive statistics by country:

```
x=adsl.groupby("COUNTRY").describe().reset_index()
```

| Index 0 | 0         | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9     | 10      | 11      | 12  |
|---------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|-------|---------|---------|-----|
| None    | COUNTRY   | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | AGE   | AGE     | AGE     | AGE |
| None    |           | count    | mean     | std      | min      | 25%      | 50%      | 75%      | max      | count | mean    | std     | min |
| 0       | AUSTRALIA | 183      | 1        | 0        | 1        | 1        | 1        | 1        | 1        | 183   | 67.2077 | 10.9243 | 36  |
| 1       | INDIA     | 185      | 2        | 0        | 2        | 2        | 2        | 2        | 2        | 185   | 58.0703 | 10.2756 | 32  |
| 2       | MALAYSIA  | 182      | 3        | 0        | 3        | 3        | 3        | 3        | 3        | 182   | 56.3736 | 11.8789 | 21  |

## Percentiles:

```
y=adsl.groupby("COUNTRY").describe(percentiles=[.10,.90]).reset_index()
```

| Index 0 | 0         | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        |
|---------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|
| None    | COUNTRY   | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN | COUNTRYN |
| None    |           | count    | mean     | std      | min      | 10%      | 50%      | 90%      | max      |
| 0       | AUSTRALIA | 183      | 1        | 0        | 1        | 1        | 1        | 1        | 1        |
| 1       | INDIA     | 185      | 2        | 0        | 2        | 2        | 2        | 2        | 2        |
| 2       | MALAYSIA  | 182      | 3        | 0        | 3        | 3        | 3        | 3        | 3        |

## Descriptive statistics for only required variable to analyze:

```
z=adsl_1.groupby("COUNTRY")["AGE"].agg(['count','mean','median','std','quantile','min','max','sum']).reset_index()
```

| Index | COUNTRY   | count | mean    | median | std     | quantile | min | max | sum   |
|-------|-----------|-------|---------|--------|---------|----------|-----|-----|-------|
| 0     | AUSTRALIA | 183   | 67.2077 | 69     | 10.9243 | 69       | 36  | 96  | 12299 |
| 1     | INDIA     | 185   | 58.0703 | 58     | 10.2756 | 58       | 32  | 84  | 10743 |
| 2     | MALAYSIA  | 182   | 56.3736 | 57     | 11.8789 | 57       | 21  | 94  | 10260 |

Note: "Numbers in tables are randomly example numbers for illustration purposes"

# How to validate the two DataFrames for equal observations?

**compare():** This methods allow you to compare two DataFrame or Series, respectively, and summarize their differences

```
t_ov_sop_fas.compare(t_ov_sop_fas_parp)
```

Difference of the columns

|    | col2      |           | col3         |              |
|----|-----------|-----------|--------------|--------------|
|    | self      | other     | self         | other        |
| 0  |           | NaN       |              | NaN          |
| 1  | 34(4.4)44 | 21(3.1)22 | 261(33.5)525 | 190(28.4)373 |
| 2  | 3(0.4)3   | 2(0.3)2   | 197(25.3)380 | NaN          |
| 3  | 19(2.4)22 | 11(1.6)11 | 95(12.2)142  | 58(8.7)83    |
| 4  | 15(1.9)19 | 9(1.3)9   | 3(0.4)3      | 2(0.3)2      |
| 5  | 1(0.1)3   | NaN       | 100(12.8)141 | 70(10.5)99   |
| 6  | 2(0.3)2   | NaN       | 109(14.0)199 | 71(10.6)124  |
| 7  | 32(4.1)39 | 21(3.1)22 | 125(16.0)185 | NaN          |
| 8  | 8(1.0)10  | 3(0.4)3   | 45(5.8)57    | 35(5.2)44    |
| 9  | 24(3.1)29 | 16(2.4)16 | 234(30.0)443 | 170(25.4)309 |
| 10 | 4(0.5)4   | 2(0.3)2   | 2(0.3)2      | 1(0.1)1      |
| 11 | 1(0.1)1   | NaN       | 12(1.5)23    | 10(1.5)19    |
| 12 | 21(2.7)23 | 16(2.4)17 | 198(25.4)379 | NaN          |
| 13 | 0         | NaN       | 21(2.7)28    | 17(2.5)20    |
| 14 | 7(0.9)12  | 3(0.4)3   | 30(3.9)39    | 10(1.5)12    |
| 15 | 4(0.5)4   | NaN       | 42(5.4)62    | 2(0.3)2      |
| 16 | 1(0.1)2   | NaN       | 9(1.2)15     | 3(0.4)5      |
| 17 | 3(0.4)3   | 2(0.3)2   | NaN          | NaN          |

No difference

|    | col2 |       | col3 |       |
|----|------|-------|------|-------|
|    | self | other | self | other |
| 0  |      | NaN   |      | NaN   |
| 13 | 0    | NaN   | NaN  | NaN   |

Note: "Numbers in tables are randomly example numbers for illustration purposes"

[scrp@novonordisk.com](mailto:scrp@novonordisk.com)

# How to validate the two DataFrames for unequal observations and export the validation dataframe?

- ❖ `eq()` : It is a wrapper used to perform flexible comparison of dataframe object with constant, series or another dataframe object

```
unequal=t_ov_sop_fas_[t_ov_sop_fas_parp.eq(t_ov_sop_fas_).any(axis=1)==True]
```

|   | index | coll             | col2      | col3         | col4         | ord | sort |
|---|-------|------------------|-----------|--------------|--------------|-----|------|
| 0 | 0     | Numberofpatients |           |              | 779          | 0.0 | 0.0  |
| 1 | 1     | Adverseevents    | 34(4.4)44 | 261(33.5)525 | 281(36.1)569 | 1.0 | 1.0  |

- ❖ Export the validated dataframe: `unequal.to_html(r'path/validation.html')`
- ❖ Run the file in terminal/powershell: `python t_ov_sop_fas_parp.py`

*Note: "Numbers in tables are randomly example numbers for illustration purposes"*

# Conclusion

- Python has several libraries to perform statistical analysis
- Code readability facilitates learning the syntax
- Enhanced ipdb debugger in the IPYTHON Console which allows the user to debug the code in easy way
- Programming in Python is much easier to save time
- Python is used as an open-source scripting language and cost-effectiveness
- The production SAS datasets can be validated using Python with clear syntax, simplicity and high efficiency

# Thank You!