AstraZeneca

From R to SAS and Back Again

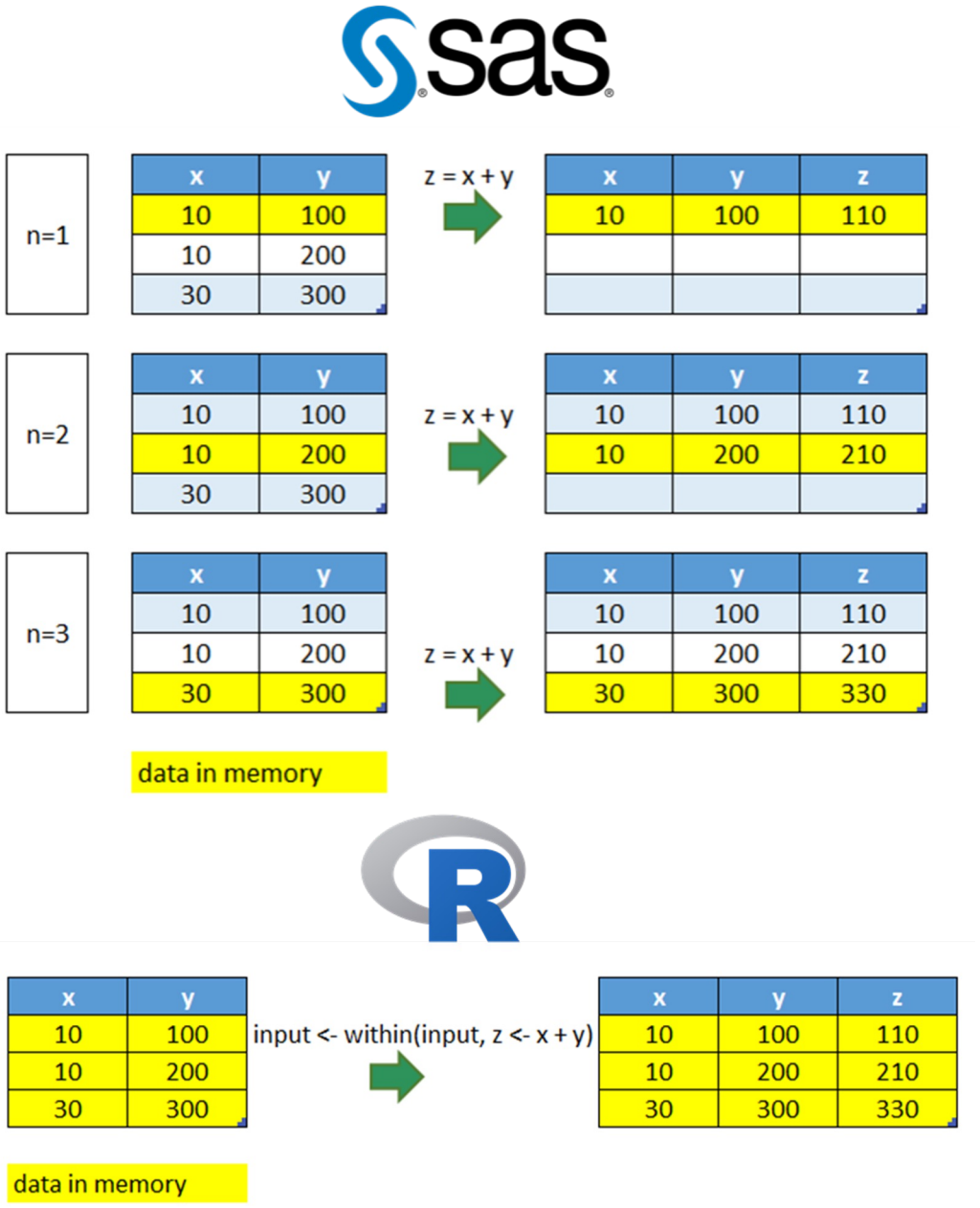
Jasmine Sadler, Keane Cooke, Isabelle Coates

25th May 2023

Learning SAS as an R user

Stage 1: understanding the fundamentals

SAS	R
Procedural Language	Functional Language with support for object-oriented programming
Compiles and then executes	No compilation just execution
Data step works by looping over a dataset row by row	R functions are vectorized
Well documented	Variation in documentation standards
Technical support offered	Community based support



Stage 2: Basic functionality and syntax

Syntax:

- R functions are often more intuitive and show more similarity to other languages than SAS functions

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Weight	9	70.4444444	3.1169340	65.0000000	73.5000000
Height	9	154.8888889	17.2514090	135.0000000	190.0000000

```
> summary(bmi)
     Weight          Height
 Min.   :65.00   Min.   :135.0
 1st Qu.:69.00   1st Qu.:140.0
 Median :71.50   Median :155.0
 Mean   :70.44   Mean   :154.9
 3rd Qu.:73.00   3rd Qu.:164.0
 Max.   :73.50   Max.   :190.0
```

Functionalities:

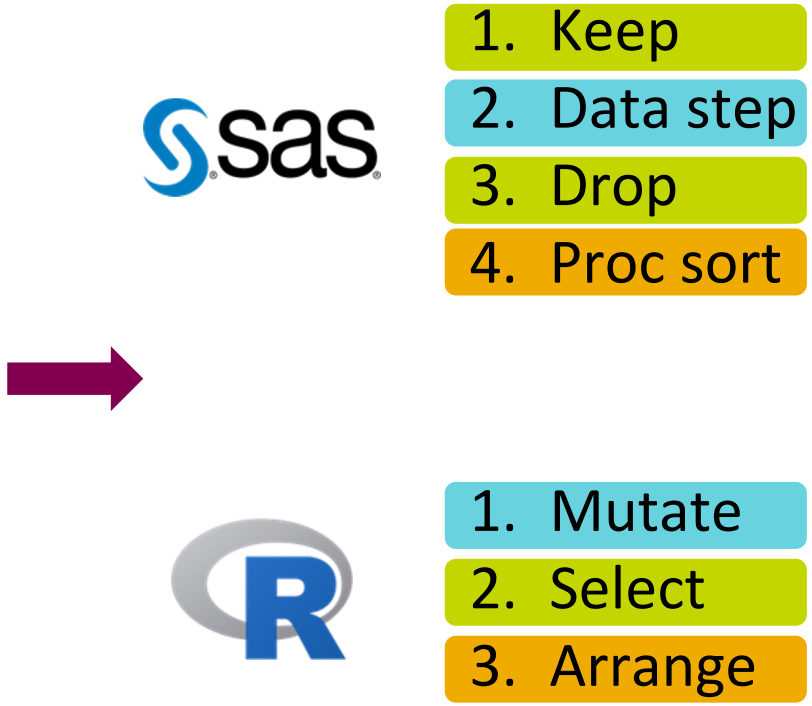
- SAS shows less flexibility to achieve certain outcomes
- R's extensive packages give multiple ways to achieve an outcome

Stage 3: Linking steps for data transformation

In R, the package tidyverse is the go-to for data transformation

- SAS programmers find the tidyverse syntax more familiar to SAS's proc steps

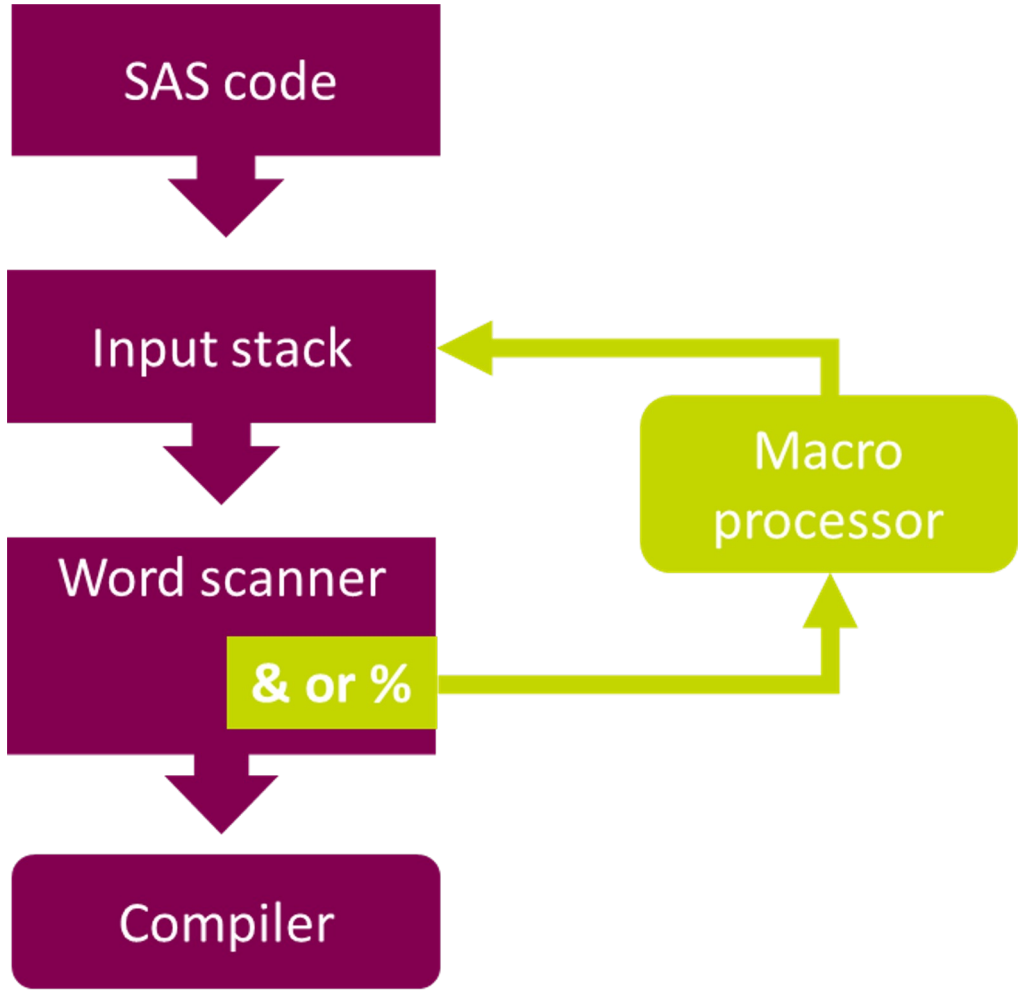
sex	wt_lb	ht_cm
M	145.2	140
F	158.4	145
M	161.7	160
M	160.6	190
F	151.8	155



sex	weight	unit
M	66	kg
F	72	kg
M	73.5	kg
M	73	kg
F	69	kg

Stage 4: Applying learnings to new contexts

SAS Macro	R Function
Macro is compiled differently to normal SAS code	R functions are executed like normal R code
Has different syntax to base SAS	Same syntax as R
SAS macro variables are just character strings	R functions arguments can take any object type



R for Statistical Programming

General benefits to using R in Pharma

1. Open source and free
2. Updated frequently
3. Validation process in place

Opportunities – Statistical Capabilities

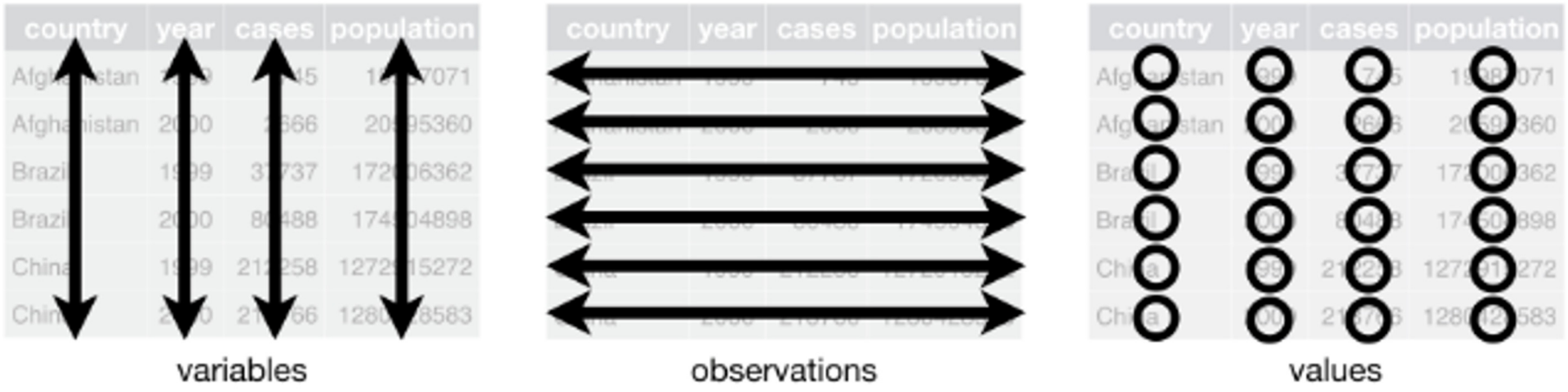
1. Includes more recent cutting-edge statistical techniques than SAS

2. Multiple packages and methods available to achieve the same type of analysis

Packages capable of fighting linear models

Opportunities – The Tidyverse

- The Tidyverse – Perhaps the greatest strength of R
- A set of tools for data transformation and visualisation
- Focused on the tidy data principles
  - Each variable must have its own column
  - Each observation must have its own row
  - Each value must have its own cell



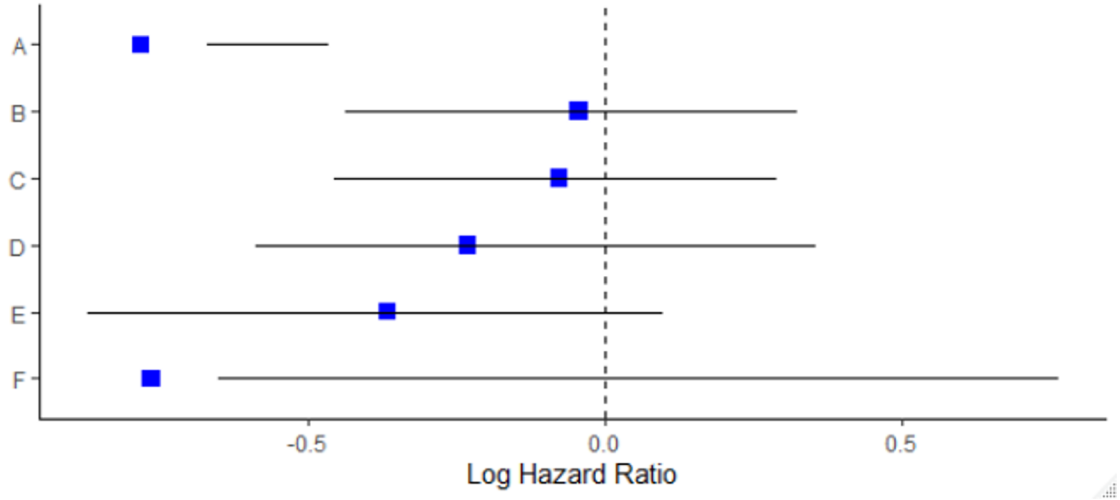
Tidy Data & CDISC Standards

- The tidy data principles are compatible with CDISC standards

Row	STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC	RFENDTC	RFXSTDTC	RFXENDTC
1	ABC12	DM			12	10		-10
2	ABC123	DM	ABC12301002	01002	2006-01-15	2006-02-28	2006-01-15	2006-02-28
3	ABC123	DM	ABC12301003	01003	2006-01-16	2006-03-19	2006-01-16	2006-03-19
4	ABC123	DM	ABC12301004	01004				
5	ABC123	DM	ABC12302001	02001	2006-02-02	2006-03-31	2006-02-02	2006-03-31
6	ABC123	DM	ABC12302002	02002	2006-02-03	2006-04-05	2006-02-03	2006-04-05

Opportunities – Ggplot2

- Aside from SDTMs and ADaMs, the real ‘endgame’ of statistical programming is generating TLFs
- This is where R could shine – ggplot allows a user to create publication-ready plots with ease and flexibility




```r
library(tidyverse)

##Create forest plot
ggplot(data=df, aes(y = fct_rev(model))) +
  theme_classic() +
  geom_point(aes(x=log_est), shape=15, size=3, colour='blue') +
  geom_linerange(aes(xmin=log_conf_l, xmax=log_conf_h)) +
  geom_vline(xintercept = 0, linetype="dashed") +
  labs(x="Log Hazard Ratio", y="")
```

Opportunities – Harnessing Shiny

- Use of R opens the door to leverage powerful tools such as shiny
- Shiny allows a user to create interactive web apps
- Potential use cases:
  - Interactive exploratory data analysis
  - Automation
  - More possibilities are likely to arise in the future!

Exploring R in Stat Programming

Exploratory Use-Cases

- Automation - generating specifications using R
- Fully automated process - calling a single function

R1C1 | Source Library

	1 Source Library	2 Source Dataset	3 Source Variable	4 SDTM Domain	5 SDTM Dataset	6 SDTM Variable Name	7 Variable Label	8 Variable Length	9 Seq	10 Key	11 Origin	12 Format / Controlled Terminology	13 Variable Type	14 Siginificant Digits	15 Core
2	RAW	AE	STUDY	AE	AE	STUDYID	Study Identifier	21	1	1	Protocol		C		Req
3	RAW	EVTLOG	STUDY	AE	AE	STUDYID	Study Identifier	21	1	1	Protocol		C		Req
4	RAW	AE		AE	AE	DOMAIN	Domain Abbreviation	8	2		Derived	(DOMAIN)	C		Req
5	RAW	EVTLOG		AE	AE	DOMAIN	Domain Abbreviation	8	2		Derived	(DOMAIN)	C		Req
6	RAW	AE	STUDYSUBJECT	AE	AE	USUBJID	Unique Subject Identifie	30	3	2	Derived		C		Req
7	RAW	EVTLOG	STUDYSUBJECT	AE	AE	USUBJID	Unique Subject Identifie	30	3	2	Derived		C		Req
8				AE	AE	AESEQ	Sequence Number	8	4		Derived		N		Req
9				AE	AE	AEGRPID	Group ID	40	5		Derived		C		Perm
10	RAW	EVTLOG	LINEMODULE_O	AE	AE	AEREFID	Reference ID	40	6		Derived		C		Perm
11	RAW	AE		AE	AE	AESPID	Sponsor-Defined Identif	40	7		CRF		C		Perm
12	RAW	EVTLOG	EVNO	AE	AE	AESPID	Sponsor-Defined Identif	40	7		CRF		C		Perm

spec_creation.R*

Source on Save

```
1  source("/program/specification_creation_function.r")
2
3  specification_creation(standards_location="/standards",
4                         output_location="/output",
5                         output_file=gsub(" ","_",paste("sdtm_specification",format(Sys.Date(),"%Y%m%d"),".xlsx")),
6                         study_file="/study_file.xlsx",
7                         previous_spec_location = NA,
8                         previous_spec_file = NA)
9
10
```


Exploratory Use Cases

- Automation - generating domain code using R
- Fully automated process - calling a single function

code_gen.R

Source on Save

```
1  source("/program/code_generator_function.r")
2
3
4  code_generator(sdtm_spec="/sdtm_specification_20230512",
5                 output_location="/dev/program")
```

ae	SAS_Pgm	development	10.59 KB
ce	SAS_Pgm	development	12.68 KB
cm	SAS_Pgm	development	20.19 KB
co	SAS_Pgm	development	10.27 KB

Advantages and disadvantages of switching languages

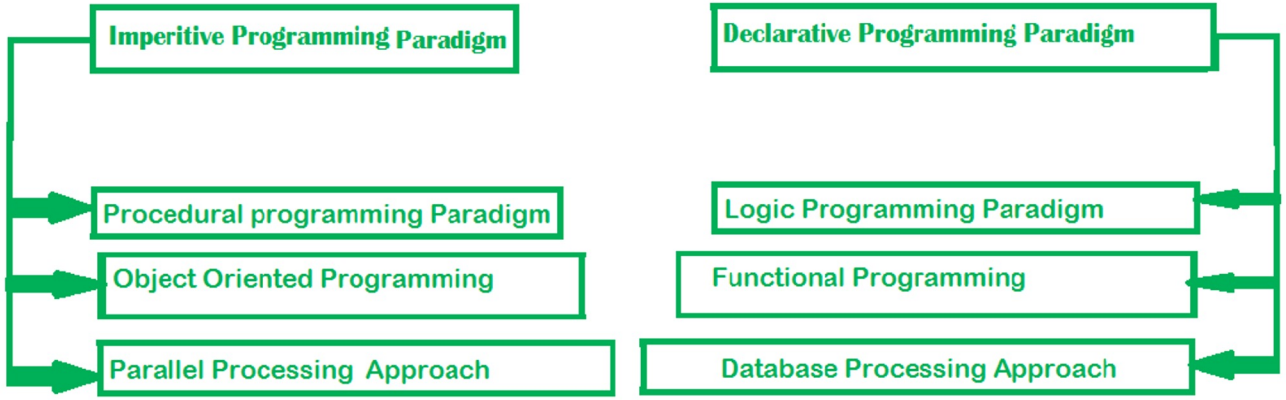
Advantages/Disadvantages of switching between languages

- Structure/Syntax/processing
  - Javascript, C++, Python syntax
  - SQL, mySQL syntax
- Compilation and Function logic
  - SAS step and macro compilation

Advantages/Disadvantages of switching between languages

- Paradigms
  - No universal language
- Language specialisation
  - Python
  - Javascript

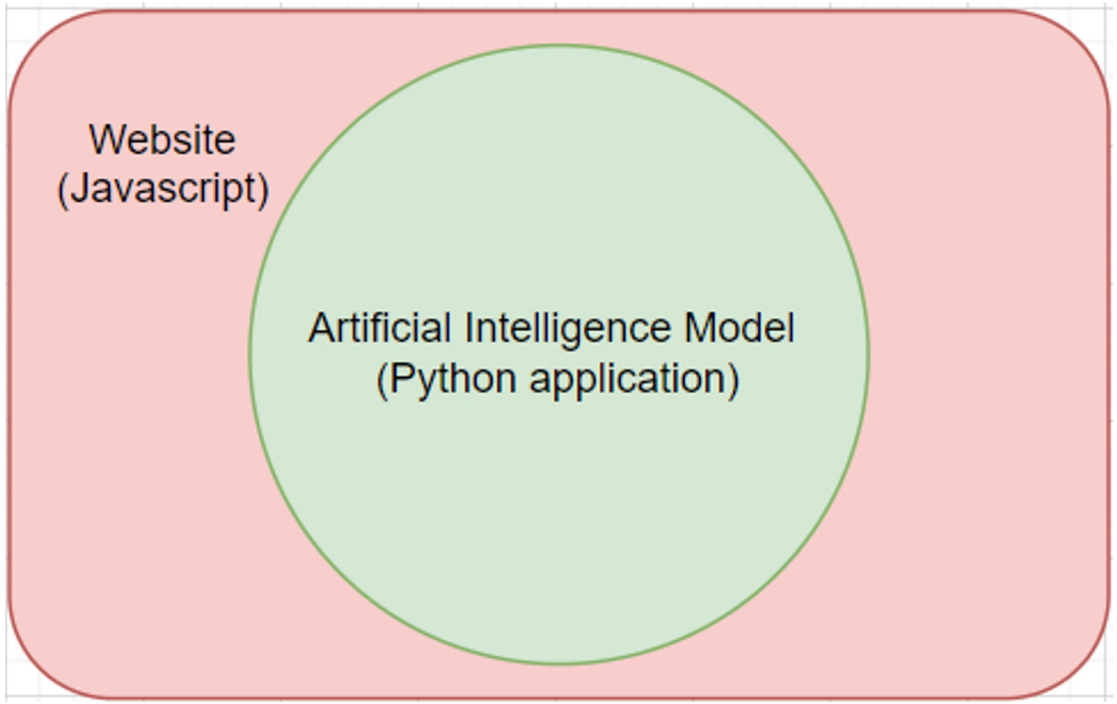
Programming Paradigms



Recommendations of switching languages

Recommendations of switching languages

- Recommendations
  - Switching between languages
  - Exploiting languages for their specialisation.



Conclusion

Acknowledgements

- The R-Initiative Team: George Gibson, Vijay Vagolu, Emmanuel Endeshaw, Tomaz Lebitko, Artur Krupa, Stella Munuo (Sponsor)