

Generative AI

A case study highlighting the potentials and the risks

Nikolay Manchev
Domino Data Lab

▶RS-10211 SEARCH_A01
▶RS-70211 SEARCH_A01

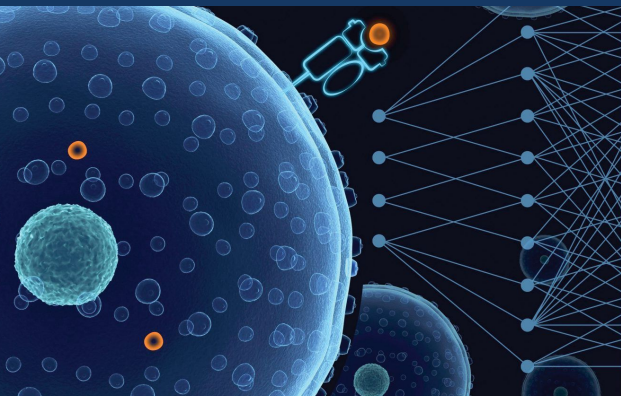
▶SEARCH▶TR/01▶03
▶SEARCH▶TR/01▶03

▶TR-010N ▶TR/01▶03
▶TR/010N ▶TR/01▶03



Domino unleashes data science

for the world's most advanced enterprises



BNP PARIBAS



Principal™



Bristol Myers Squibb™



Carnival™



CISCO



moderna™



Honeywell



Johnson & Johnson



JOHN DEERE



LLOYDS BANKING GROUP



LOCKHEED MARTIN



MOODY'S ANALYTICS



NEW YORK LIFE



GSK



S&P Global



Lilly



SUNCORP



UBS



Vanguard



vmware®



ZURICH

The changing technology landscape

- SAS has long dominated the pharmaceutical industry
- Open-source ecosystems are challenging this

Not only Python remains the No. 1 ranking language in 2023, but it also "widens its lead"

IEEE Spectrum

What about pharma? | SAS/R/Python use

- Dilorio & Abolafia (N=10), 2008-2018

Language	10 Years Ago (N)	Today (N)
SAS	10	10
Some R	1	4*
Extensive use of R	0	0
Other (SPSS, JMP)	1	0

- Domino Data Lab (N=15), March 2023

Language	Number of responses
SAS	12
R	12
Python	11

Source-to-source (S2S) translation

S2S translator (S2S compiler): a language processor that takes source code from one programming language and produces an equivalent program in a different programming language.

- They have been around for ages
 - CONV86 , TRANS86.COM
 - PIPS
- They rely on pattern matching (and are broken)

MeS-86™ *ASSEMBLY LANGUAGE CONVERTER OPERATING INSTRUCTIONS FOR ISIS-II USERS*, Manual Order No. 9800642-0, 1979, Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051, <https://www.nj7p.info/Manuals/PDFs/Intel/9800642B.pdf>

Roger Taylor, Phil Lemmons, *Upward migration – Part 1: Translators – Using translation programs to move CP/M-86 programs to CP/M and MS-DOS*, 1982, BYTE, Vol. 7, no. 6. BYTE Publications Inc., pp. 321–344, <https://www.tech-insider.org/personal-computers/research/acrobat/8206-b.pdf>

R. Keryell, C. Ancourt, F. Coelho, B. Creusillet, F. Irigoin, P., *Pips: a Workbench for Building Interprocedural Parallelizers, Compilers and Optimizers Technical paper*, May 1996, Centre de Recherche en Informatique, Ecole Nationale Supérieure des Mines de Paris

Broken translation

- Undefined behaviour
- Unsupported features
- Not taking advantage of the full capabilities of the target
- Inability to recognise intent

```
#include <iostream>

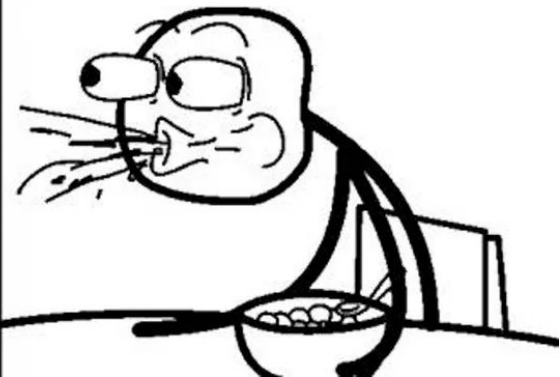
int main() {
    while (1)
        ;

    void unreachable() {
        std::cout << "Hello world!" << std::endl;
    }
}
```

This code will
never do
anything!



```
$ clang++ loop.cpp -O1 -Wall -o loop
$ ./loop
Hello world!
```



Inability to recognise intent

- Like-for-like often breaks

```
while (*m++ = *p++);

int main() {
    char source[] = "Hello, World!"; // Source string
    char destination[20];           // Destination string

    char *p = source;                // Pointer to source string
    char *m = destination;           // Pointer to destination string

    // Copy the source string to the destination string
    while (*m++ = *p++);

    printf("Source: %s\n", source);
    printf("Destination: %s\n", destination);

    return 0;
}
```

Translation to R | CodePal.ai

```
source <- "Hello, World!" # Source string
destination <- character(20) # Destination string

p <- source # Pointer to source string
m <- destination # Pointer to destination string

# Copy the source string to the destination string
while (m <- m + 1) {
  m <- p <- p + 1
}

cat("Source: ", source, "\n")
cat("Destination: ", destination, "\n")
```

What it had to do: `destination <- source`

Does it really work?

Yes! The theoretical basis for the translation has been developed at ETH Zurich, as part of the PhD thesis of Dr. Marco Trudel

```
package demo;
import static com.mtsystems.coot.String8.s8;
import com.mtsystems.coot.String8;

public class DemoTranslation {
    public static void main(String[] args) {
        String8 source = s8("Hello, World!"); // Source string
        String8 destination = new String8(20); // Destination string
        String8 p = source; // Pointer to source string
        String8 m = destination; // Pointer to destination string
```

What it had to do:

```
        System.out.println("Source: " + source);
        System.out.println("Destination: " + destination);
    }
}
```

Abstract Syntax Trees | ASTs

Generative AI: A Case Study Highlighting the Potential and the Risks

Translating legacy SAS code to modern open-source alternatives

Nikolay Manchev *

Domino Data Lab

October 13, 2023

Abstract

In this paper we look at how the use of permissive-licensed languages like Python and R has been steadily increasing in the pharmaceutical industry, often at the expense of proprietary technologies like SAS. We consider the large amount of technical know-how that is currently implemented in SAS and investigate how with the help of Large Language Models (LLMs) such existing code can be migrated to more open and widely used alternatives like R and Python. We cover some key challenges around source to source translation in detail and provide a blueprint for building a modern translation pipeline, which is flexible, cost efficient, compliant, and produces reproducible results.

1 The changing technology landscape

When it comes to statistical analysis and data management, the Statistical Analysis System (SAS) software suite has long dominated the pharmaceutical industry. Some of the reasons, which have historically driven the adoption of SAS, are its maturity, regulatory acceptance, and strong user community. SAS has indeed a long-established presence as the software suite has been around since the 1960s. It is widely accepted by regulatory agencies as the U.S. Food and Drug Administration and the European Medicines Agency, and its popularity in academia has maintained a pool of trained resources that the industry can draw from.

It is worth noting, however, that the technology landscape is rapidly evolving and although SAS has had a dominant position historically, its widespread use has been increasingly challenged by open-source ecosystems centred around languages like R and Python. The 10th annual ranking of the top programming languages published by IEEE Spectrum [1] points out that not only Python remains the No. 1 ranking language in 2023, but it also “widens its lead”. In this ranking Python holds first place with the maximum score of 1.0 while R is ranked 11th with a score of 0.1316 and SAS is put in 15th place¹.

*nikolay.manchev@dominodatalab.com

¹The methodology uses 8 metrics to judge language popularity and provides a normalised score in [0,1] for the first 13 entries. No index value is given for SAS as it currently sits in 15th place. More details on the methodology are available here: <https://spectrum.ieee.org/top-programming-languages-methodology>

Large Language Models| LLMs

Generative AI: A Case Study Highlighting the Potential and the Risks

Translating legacy SAS code to modern open-source alternatives

Nikolay Manchev *

Domino Data Lab

October 13, 2023

Abstract

In this paper we look at how the use of permissive-licensed languages like Python and R has been steadily increasing in the pharmaceutical industry, often at the expense of proprietary technologies like SAS. We consider the large amount of technical know-how that is currently implemented in SAS and investigate how with the help of Large Language Models (LLMs) such existing code can be migrated to more open and widely used alternatives like R and Python. We cover some key challenges around source to source translation in detail and provide a blueprint for building a modern translation pipeline, which is flexible, cost efficient, compliant, and produces reproducible results.

1 The changing technology landscape

When it comes to statistical analysis and data management, the Statistical Analysis System (SAS) software suite has long dominated the pharmaceutical industry. Some of the reasons, which have historically driven the adoption of SAS, are its maturity, regulatory acceptance, and strong user community. SAS has indeed a long-established presence as the software suite has been around since the 1960s. It is widely accepted by regulatory agencies as the U.S. Food and Drug Administration and the European Medicines Agency, and its popularity in academia has maintained a pool of trained resources that the industry can draw from.

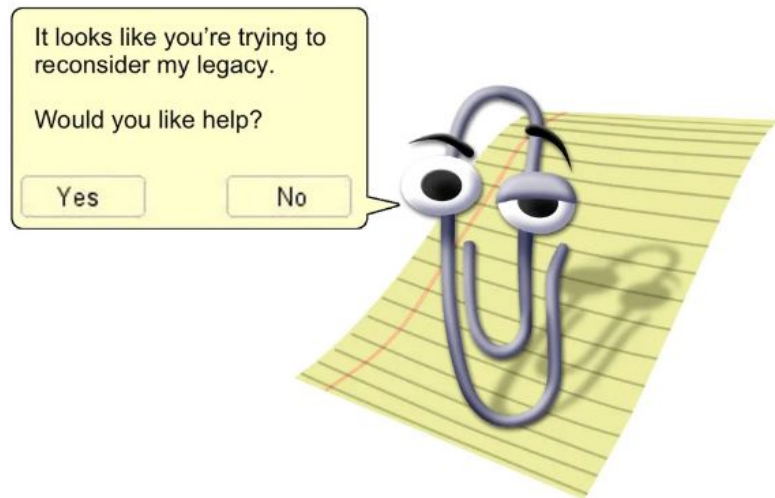
It is worth noting, however, that the technology landscape is rapidly evolving and although SAS has had a dominant position historically, its widespread use has been increasingly challenged by open-source ecosystems centred around languages like R and Python. The 10th annual ranking of the top programming languages published by IEEE Spectrum [1] points out that not only Python remains the No. 1 ranking language in 2023, but it also “widens its lead”. In this ranking Python holds first place with the maximum score of 1.0 while R is ranked 11th with a score of 0.1316 and SAS is put in 15th place¹.

*nikolay.manchev@dominodatalab.com

¹The methodology uses 8 metrics to judge language popularity and provides a normalised score in [0,1] for the first 13 entries. No index value is given for SAS as it currently sits in 15th place. More details on the methodology are available here: <https://spectrum.ieee.org/top-programming-languages-methodology>

S2S with LLMs

- High-Level Abstraction
- Flexibility
- No Need for Language-Specific Grammar Rules
- Context Awareness and Intent Understanding
- Simpler Deployment



ChatGPT example | Pointer to string copy routine

Prompt: What does the following line of C code do: `while (*m++ = *p++);`

Completion: The line of C code `while (*m++ = *p++);` is a concise way to perform a common operation in C: copying a null-terminated string (character array) from one location to another. Let's break down what it does step by step:

1. `*m++` and `*p++`: `*m++` and `*p++` are postfix increment operations. They first use the value of `m` and `p`, respectively, and then increment the pointers `m` and `p`. In this context, `m` and `p` are assumed to be pointers to characters (`char*`).
2. `*m = *p`: This part of the code assigns the character pointed to by `p` to the character pointed to by `m`.
3. `while (...)`: The code is enclosed in a while loop. It continues executing as long as the result of the assignment `*m = *p` is non-zero (true). In C, the result of an assignment operation is the value assigned, so as long as characters are being copied (i.e., not a null terminator), the loop continues.
4. Null termination: The loop will terminate when it encounters a null terminator `'\0'` in the string pointed to by `p`. This is because the null terminator has a value of 0, so when the assignment `*m = *p` copies the null terminator, the loop condition becomes false, and the loop exits.

In summary, this line of code copies the characters from the memory location pointed to by `p` to the memory location pointed to by `m` until it reaches the end of the string (null terminator). It's a concise way to perform string copying in C and is commonly used when you want to copy one string into another.

ChatGPT example | Translate to R

Prompt: Translate this code to R

Completion: In R, which is primarily a language for statistical computing and data analysis, direct memory manipulation and pointer-based operations like the C code you provided are not typically used. Instead, R provides higher-level abstractions for working with data.

If you want to copy elements from one vector to another in R, you can use standard indexing and assignment operations. Here's an example of how you might copy elements from one vector `p` to another vector `m` in R:

```
m <- character(length(p)) # Create a character vector m of the same length as p
```

```
m <- p # Copy the elements from p to m
```

This code creates a new character vector `m` of the same length as `p` and then assigns the values from `p` to `m`, effectively copying the elements. R takes care of the memory management and doesn't require explicit memory manipulation like in C.

R is designed for high-level data analysis and manipulation, so it abstracts away many low-level memory operations found in languages like C.

ChatGPT example| SAS

Prompt: Translate the following SAS code into R. Produce just the code.

```
title "Weight Limits on the Y Axis";
proc sgplot data=sashelp.classfit;
  where age > 12;
  band x=name lower=lowermean upper=uppermean;
run;
title;
```

Completion: Here is the equivalent R code for the given SAS code:

```
library(ggplot2)
# Set the title
ggtitle("Weight Limits on the Y Axis")
# Load the data (assuming sashelp.classfit is available in R)
data <- sashelp.classfit
# Filter the data where age > 12
data <- data[data$age > 12, ]
# Create the plot
ggplot(data) + geom_ribbon(aes(x = name, ymin = lowermean, ymax = uppermean),
fill = "blue") + labs(x = "Name") + theme_minimal()
```

Commercial vs. Locally Hosted

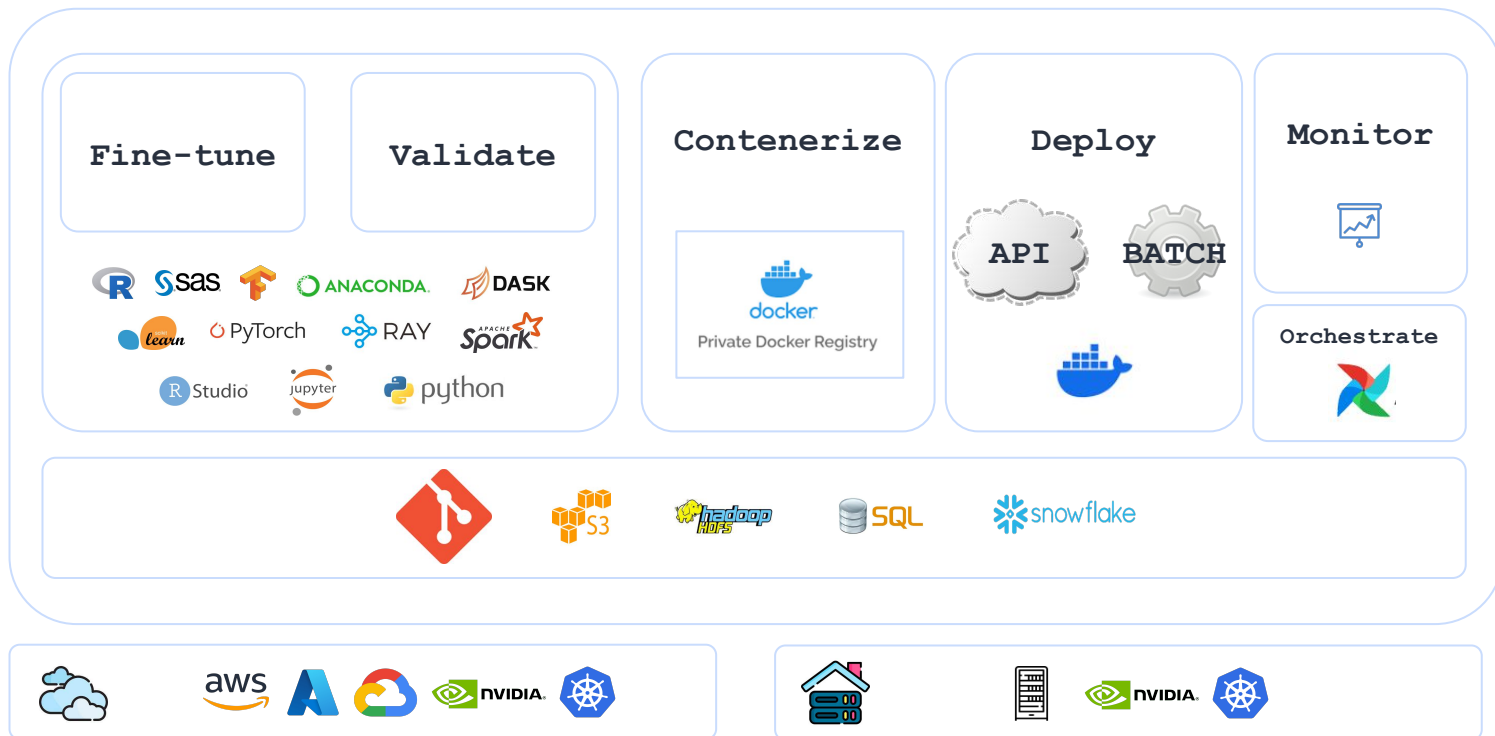
	Commercial APIs	Locally Hosted
Cost to train	😊	😞
Cost to use	😞	😊
Reproducibility	😞	😊
Long-term support	😞	😊
IP Risk	😞	😊

Blueprint| S2S



Foundation Models

StarCoder,
Code Llama,
Code T5 etc.





Thank You

www.dominodatalab.com

nikolay.manchev@dominodatalab.com
www.linkedin.com/in/nikolaymanchev/
@nikolaymanchev