9a. Overview and Goals

Martin Alfaro
PhD in Economics

INTRODUCTION

In the previous chapter, we began our exploration of high performance in Julia by focusing on type stability. We now shift our attention to memory allocations, a critical aspect of performance optimization.

Memory allocations occur whenever a new object is created, involving the reservation of memory space to store its values. The aspect is crucial for performance, since the approach selected to handle the process can significantly slow down computations. In particular, memory allocations on the heap, simply referred to as *memory allocations*, incur a notable cost due to the additional CPU instructions required for memory management.

Despite this, the interplay between memory allocation and performance is complex. In fact, **reducing memory allocation is neither necessary nor sufficient for speeding up computations**—we'll present instances where the approach allocating more memory turns out to be faster. This apparent paradox arises from a trade-off involved when creating a new object: although allocations can lead to a significant overhead, the resulting objects store their data in contiguous blocks of memory, enabling the CPU to access information more efficiently.

From a practical perspective, it's essential to closely monitor memory usage if performance is critical. **Excessive memory allocation often serves as a red flag**: if two approaches exhibit large differences in memory allocation, their execution speeds are likely to differ significantly as well.