



Analysis and Reporting Technology

risk.assessr: extracting OOP function details

Edward Gillian, Hugo Bottois,
Paulin Charliquart, Andre Couturier

November, 2025

Introduction



Extracting details from Object Oriented Programming (OOP) functions in R has proved challenging because:

- the number of OOP systems in R (S3, S4, R6, and S7)
- their varying mechanisms for registration, visibility, and interaction.
- each system has its own naming, exporting, and structuring function conventions - this affects how these functions can be discovered and analyzed.

Extracting function details



Extracting details from Object Oriented Programming (OOP) functions in R has proved challenging because:

- NAMESPACE file is the primary entry point for identifying exported functions but it does not capture all OOP constructs
- the interplay between different OOP systems (e.g. S3 methods invoking R6 internals) further complicates the extraction process.

Extraction solution



Thus, extracting details from Object Oriented Programming (OOP) functions in R requires:

- a layered and adaptive approach
- detection based on:
 - visibility and registration clarity,
 - S3 methods inferred from naming conventions,
 - R6 components that require inspection of class definitions

The solution is found in:

get_exports – the function engine for extracting details



get_exports() calls two functions for getting function details

1. extract_exported_function_info()

- Extracts data about exported functions

2. classify_function_body()

- classifies the function types in a package and extracts the function body

extract_exported_function_info()



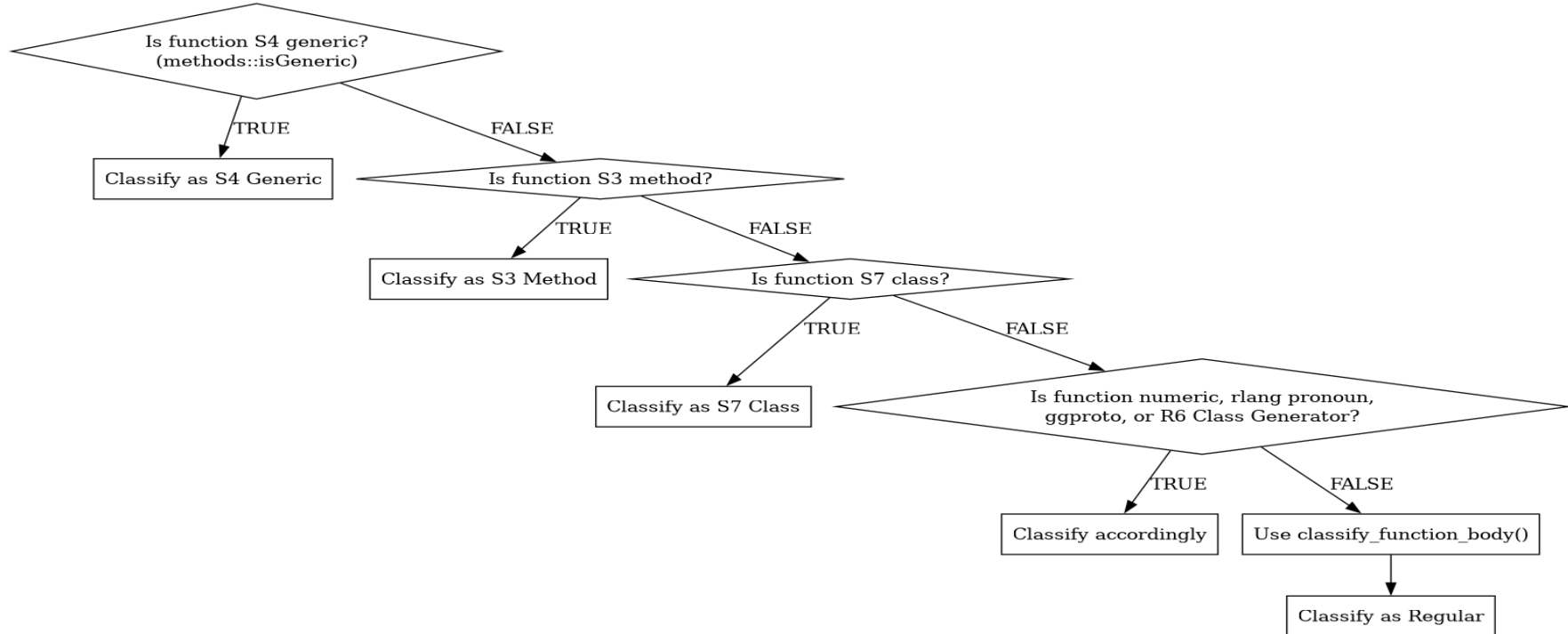
extract_exported_function_info() - tasks

1. starts with parsing the package's namespace using *parseNamespaceFile()* from base R.
 - This may not retrieve all the exported functions as the NAMESPACE file may not contain all the entries or be malformed
2. fallback function - *getNamespaceExports()* –
 - requires a loaded target package and shows actual runtime exports

classify_function_body()



classify_function_info() follows this processing sequence



Samples of the results



teal.code::get_code classification package version 0.5.0

Traceability Matrix - All functions

Search:

	Exported_function ⚡	Function_type ⚡	Code_script ⚡	Documentation ⚡	Description ⚡	Test_Coverage ⚡
4	get_code	S4 method, S4 generic	R/qenv- get_code.R	qenv.Rd	Retrieves the code stored in the qenv.	95

Samples of the results



R6::as.list code classification package version 2.5.1

Traceability Matrix - All functions

Search:

	Exported_function	Function_type	Code_script	Documentation	Description	Test_Coverage
1	as.list	S3 generic + method	R/aslist.R	as.list.R6.Rd	NA	0

Summary



Extracting details from OOP functions in R
has proved demanding and rewarding because:

- of the number of OOP systems in R
- each system has its own naming, exporting, and structuring function conventions - these influence how they are registered and to how interact with them.
- this will enable analysis of function bodies to see what functions come from which dependencies and function types such as deprecated and experimental – enabling fine grained validation analysis.



Thank you!