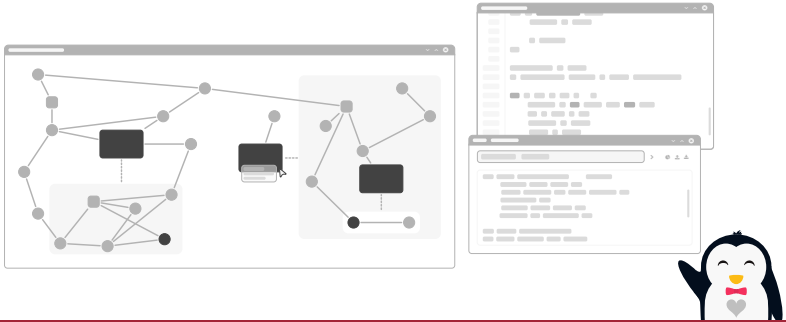


Assisting Data Analysis using Program Slicing with flowR

deRSE '25 | Ulm University | Florian Sihler and Matthias Tichy | February 26, 2025

Software Engineering
Programming Languages

universität ulm

The R Programming Language

Is Used
in Research[1]

Is Used
for Statistical Computing[2]

Is Used (mostly)
by Non-Programmers

- Lacks sophisticated static-analysis tools[3]
- *Many* powerful reflective capabilities[4]
- Incomplete language specification[5]
- Do not reproduce[1]

[1] Trisovic et al., "A Large-Scale Study on Research Code Quality and Execution" (2022, Nature Publishing Group)
[2] https://cran.r-project.org/
[3] Sihler et al., "Statically Analyzing the Dataflow of R Programs" (2024, ASE [submitted])
[4] Flückiger et al., "R melts brains: an IR for first-class environments and lazy effectful arguments" (2019, ACM DLS)
[5] R Core Team, *R Language Definition* (2023, online)

Problems of R

[1] Reproducibility
74 % do not run

```
library(ggplot2)
library(magrittr)
```

Under-specified Versions

```
setwd("C:/Users/Example/Study")
load("data.RData")
penguins ← data %>%
      dplyr::filter(species == "penguin")

t.test(size ~ species, data=penguins)

ggplot(penguins, aes(x=size, y=weight)) +
   geom_histogram(bins=42)
```

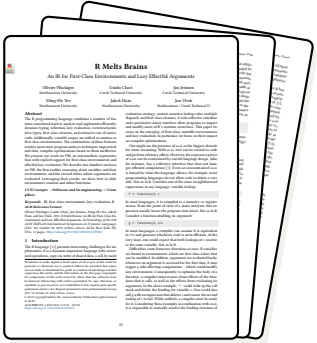
Hardcoded Paths

[7] Sen et al., *ROSA: R Optimizations with Static Analysis* (2017, arXiv)
[4] Flückiger et al., "R melts brains: an IR for first-class environments and lazy effectful arguments" (2019, ACM DLS)
[1] Trisovic et al., "A Large-Scale Study on Research Code Quality and Execution" (2022, Nature Publishing Group)

Problems of R

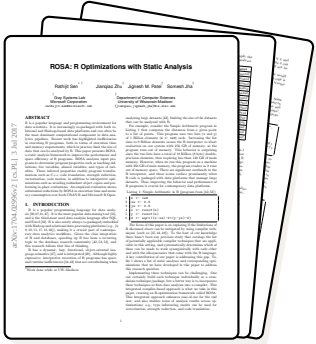
[1] Reproducibility

74 % do not run



[4] Dynamic Features

Reflexiveness, …



[7] Little Tool Support

No sophisticated analysis

[7] Sen et al., *ROSA: R Optimizations with Static Analysis* (2017, arXiv)
[4] Flückiger et al., "R melts brains: an IR for first-class environments and lazy effectful arguments" (2019, ACM DLS)
[1] Trisovic et al., "A Large-Scale Study on Research Code Quality and Execution" (2022, Nature Publishing Group)

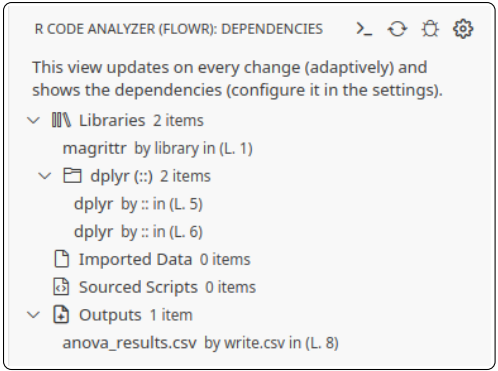
The Goal of flowR

- Interested in a single figure
- ≈ *80 % to 90 %* reduction

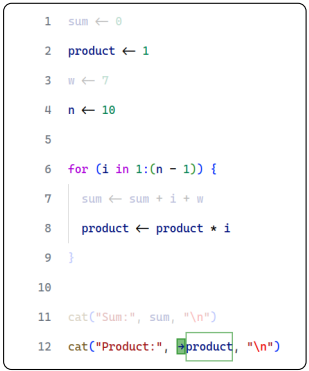
[6] Drudze et al., *Apple phenology data set and R script, related to publication "Full flowering phenology of apple tree (Malus domestica) in Pūre orchard, Latvia from 1959 to 2019"* (2021, Zenodo)

Using flowR… in VS Code!

github.com/flowr-analysis/flowr
github.com/flowr-analysis/vscode-flowr



Dependency Analysis



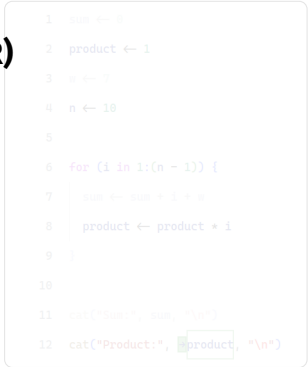
Program Slicing

Using flowR… in VS Code!

github.com/flowr-analysis/flowr
github.com/flowr-analysis/vscode-flowr

Dependency Analysis

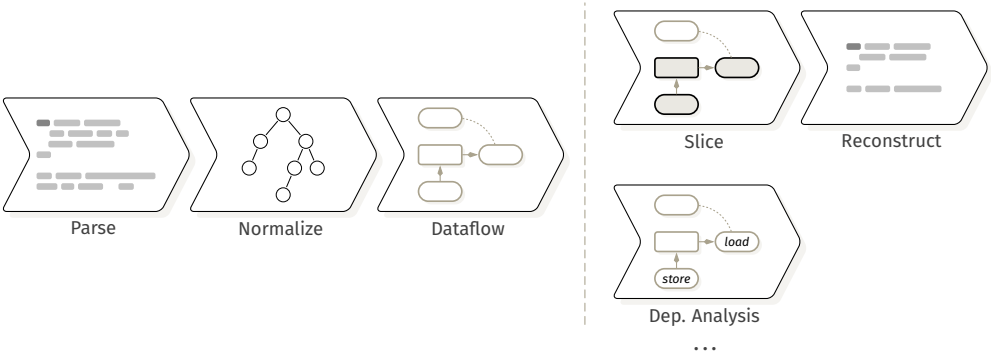
Extension
R Code Analyzer (flowR)
Works in *vscode.dev*



Program Slicing

tinyurl.com/flowr-derse25 (requires GH Account)

The Architecture



[8] Sihler, "Constructing a static program slicer for R programs" (2023, Ulm University)
[9] Weiser, "Program Slicing" (1984, IEEE Transactions on Software Engineering)

R Code Analysis

github.com/flowr-analysis/flowr

Florian Sihler • florian.sihler@uni-ulm.de

Appendix

R Scripts

We analyzed 4 083 R-files[10]

```
# set the data directory and load workspace
setwd("G:/Shared drives/Fenologija/Raksti/abeles/Zenodo/")
load("Apple_Pure_phenology_R_workspace_image.RData")

# ----------------------------------------
# scirpt to recreate components included in of "Apple_Pure_phenology_R_workspace_image.RData"
{
# ---------------------------
# phenology data subset having at least 14 observations owerlaping wiht meteorology data set
{
    dM ← d %>%
       filter(Year %in% (meteoH$e_obs$Year))
   dM ← dM %>%
       group_by(Variety) %>%
       summarize(N = n()) %>%
       arrange(N) %>%
       filter(N >= 14) %>% # 12 skirnes
       select(-N) %>%
       left_join(dM)
}

# -----------------------
# FUNCTION definitinos
# phenology model deifned as functino, see Kalvans et al, 2015, DDcos model for details
```


[10] Sihler et al., "On the Anatomy of Real-World R Code for Static Analysis" (2024, MSR)

[6] Drudze et al., *Apple phenology data set and R script, related to publication "Full flowering phenology of apple tree (Malus domestica) in Pūre orchard, Latvia from 1959 to 2019"* (2021, Zenodo)

R Scripts Fail to Replicate

74 % even fail to complete![1]

```
# set the data directory and load workspace
setwd("G:/Shared drives/Fenologija/Raksti/abeles/Zenodo/")
load("Apple_Pure_phenology_R_workspace_image.RData")

# ----------------------------------------
# scirpt to recreate components included in of "Apple_Pure_phenology_R_workspace_image.RData"
{
# ---------------------------
# phenology data subset having at least 14 observations overlaping wiht meteorology data set
{
```

Hardcoded Paths

```
        # lowerI - lower bound of model parameter range  (Tb, DD)
        # upperI - upper bound of model parameter range  (Tb, DD)

        # if pari == NA a random set is generated
        if (any(is.na(pari))) {
        pari = runif(2) * (upperI - lowerI) + lowerI
        }
        # test if the phenology and meteorology data overlap in time
        if (any(dd$Year %in% meteod$Year)) {
        print(Sys.time())
}

        print("DDsinOpt")
        a <- GenSA::GenSA(par = pari,
                          fn = DDsin_stat,
                          lower = lowerI,
                          upper = upperI,
                          stats = c("RMSE"),
                          sadalijums = sadalijums,
```

(Pseudo-)Randomness without Seed

[1] Trisovic et al., "A Large-Scale Study on Research Code Quality and Execution" (2022, Nature Publishing Group)
[6] Drudze et al., *Apple phenology data set and R script, related to publication "Full flowering phenology of apple tree (Malus domestica) in Pūre orchard, Latvia from 1959 to 2019"* (2021, Zenodo)

R Scripts Do Too Much

- Several analyses in one script
- Hard to comprehend
- Hard to extract/re-use parts

[6] Drudze et al., *Apple phenology data set and R script, related to publication "Full flowering phenology of apple tree (Malus domestica) in Pūre orchard, Latvia from 1959 to 2019"* (2021, Zenodo)

R Scripts Are Hard to Analyze

```
[11] mod.proj_wc ← ecospat.ESM.Projection(ESM.modeling.output=mod,
          new.env=eval(parse(text = paste("env_",period,"_wc",sep=""))))
```

- String-based code evaluation

```
[12] pull.cat ← function(x) {
        bins ← up_bins # (e.g., 6)
        increments ← (range(x)[2] - range(x)[1])/(bins - 1)
        to_return ← seq(range(x)[1], range(x)[2], increments)
        return(to_return)
     }

     up.cat ← function(new_bins) {
        up_bins = new_bins
        body(pull.cat)[[2]] ←← substitute(bins ← up_bins)
     }
```

- Self-modifying code

[12] Robertson, *Social hierarchy reveals thermoregulatory trade- offs in response to repeated stressors* (2020, Zenodo) [L. 68ff]
[11] Ma et al., *Predicting range shifts of pikas (Mammalia, Ochotonidae) in China under scenarios incorporating land-use change, climate change, and dispersal limitations* (2021, Zenodo) [L. 135f]

R Misses Sophisticated Analysis Tools

- RStudio IDE posit.co
  - Syntax-highlighting and auto-completion
  - ~~Refactorings (rename, extract functions and variables)~~ ← Often wrong (simple heuristics)

- R language server github.com/REditorSupport
  - Syntax-highlighting and auto-completion
  - ~~Reference tracing & Refactorings (rename)~~ ← Often wrong (XPath-Expressions)

- {lintr} github.com/r-lib/lintr
  - Style & syntax errors
  - ~~Potential semantic errors~~

    ↑

    XPath-Expressions, packages

- {CodeDepends} github.com/duncantl/CodeDepends
  - ~~Dependency analysis~~ ← Only top scope
  - Creation of call-graphs

R Scripts ...

1. fail to replicate
2. **do too much**
3. are hard to analyze
4. are not well supported by tools

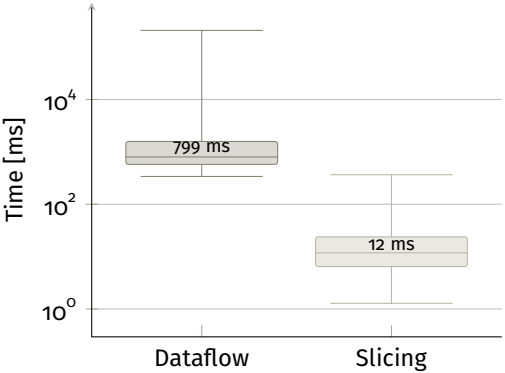
Better Software, Better Research

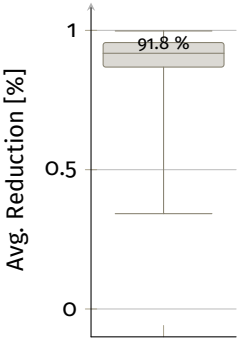
The R Code Static Analysis Landscape

	goal	method	impl. lang.	op. assignments	func. assignments	value trace(a1, ...)	controlflow	non-std. eval.	special operators	function calls	libraries	quotation	reflection	side effects	static scope	dynamic scope	type inference	pointer analysis	external files	pre-processors	hooks	FFI
[13] {CodeDepends}	static analysis	AST visitor	R	●	○			○	○	○	○		○	●								
[14] {codetools}	static analysis	AST visitor	R	●	○	○	○	○	○	○		○		●								
[15] {checkglobals}	missing libs.	AST visitor	R, C	●	○			○	○	○	○	○		○								○
[16] {rstatic}	static analysis	AST visitor	R	●		○	●		○					●								
[17] {CodeAnalysis}	static analysis	AST visitor	R	●	○	○	●	○	○	○	○	○		●			○		○			
[18] {RTypeInference}	type inference	AST visitor	R	●		○	●		○					●			●					
[19] {pkgstats}	package insight	ctags & gtags	R, C++	○					○	○				○								
[20] {globals}	distributed env.	AST visitor	R	●		○	○	○	○	○		○		●								
[21] {Rclean}	debug/refactor	PDG traversal	R	●	○			○	○	○	○			●								
[22] {lintr}	linting	XPath, visitor	R	●	○		○		○		○			●								
[23] {SimilaR}	plagiarism	PDG, visitor	R, C++	○			○			○				●								
[24] {rco}	optimization	AST visitor	R	●		●	○							●								
[25] {cyclocomp}	code complexity	AST visitor	R				○							●								
[26] {flow}	visualize, debug	AST visitor	R, C	○			●		○		○	○						○				
[27] {PaRe}	code review	Regex	R	○			○			○												
[28] {dfgraph}	static analysis	AST visitor	R	○			○			○												
[29] {rflowgraph}	call graph	AST visitor	R							○	○											
[30] {languageserver}	editor support	XPath, visitor	R, C	●	○					○	○			●								
[31] RStudio	editor support	AST visitor	Java, C++, TS, ...	●	○					○	○			●								
[7] ROSA	optimization	visitor	C++, R	●			○			○			○	●			●	○				
[32] Random	abstract int.	trace & visitor	R	○	○	○	○				○			○								
[33] RaaS	reproducibility	AST visitor	Python, R	●	○			○	○	○	○			●								
[34] GNU R	execute R	bytecode	C, Fortran, R	●	○	●	○	○	●	○		○		●								
[35] FastR	execute R	AST visitor	Java, C, R, ...	●			○		○	○	○			●								
[36] R̃	execute R	SSA, bytecode	C++, R, C, ...	●	●	●	●	○	●	○		○		●	○	●						
[37] renjin	execute R	SSA, CFG	R, Java, C, ...	●	○	○	●		○	○			○	●		●						
[38] pqR	execute R	bytecode	C, R, ...	●	○	●	○	○	●	○		○		●								
[39] MRO	execute R	bytecode	R, C, Fortran, ...	●	○	○	○															
[40] RCC	transpile C	CFG, bytecode	C/C++, Fortran, R, ...	●		●	○	○		○			○	●								

Performance Measurements

- We generated *every* possible variable of interest
- Dataflow results can be cached

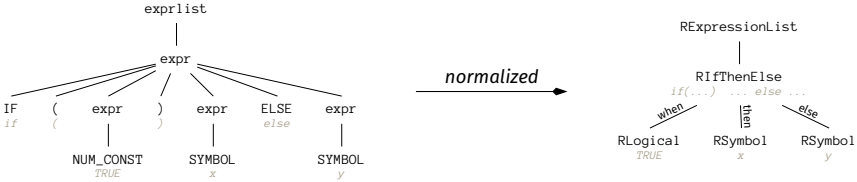




99th percentile

Parse & Normalize

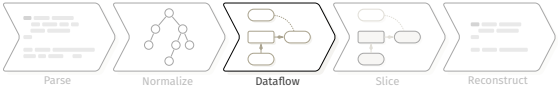

```
parse(text="if(TRUE) x else y")
```



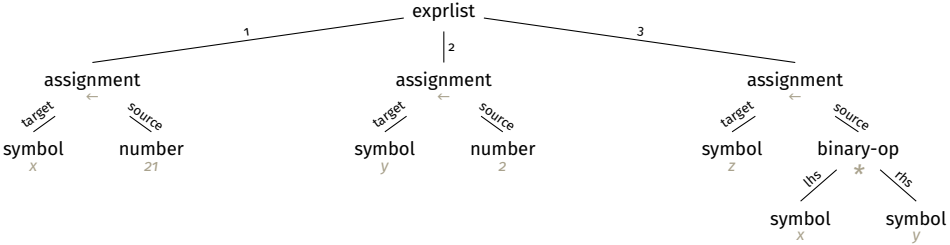
- Normalizing constants, namespacing, operators, …
- We use the “R language definition”[5] as a basis

[5] R Core Team, *R Language Definition* (2023)

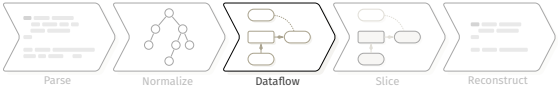
Dataflow



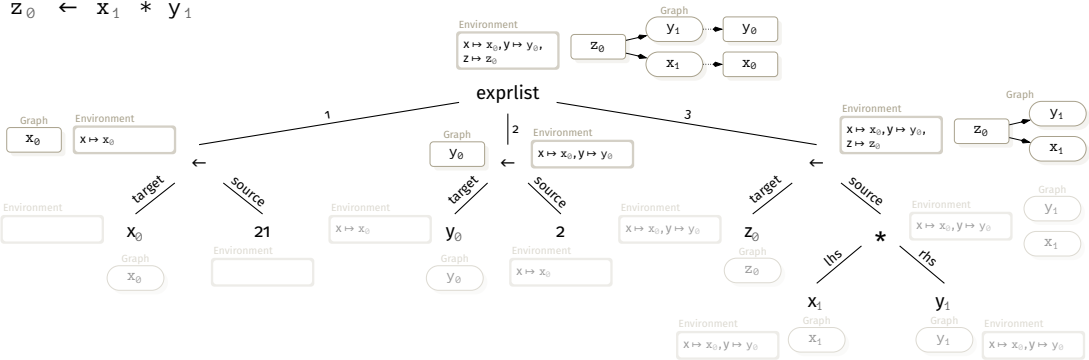
$x_0 \leftarrow 21$
$y_0 \leftarrow 2$
$z_0 \leftarrow x_1 * y_1$



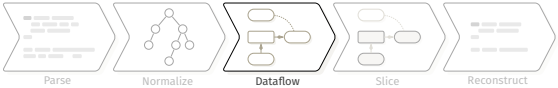
Dataflow



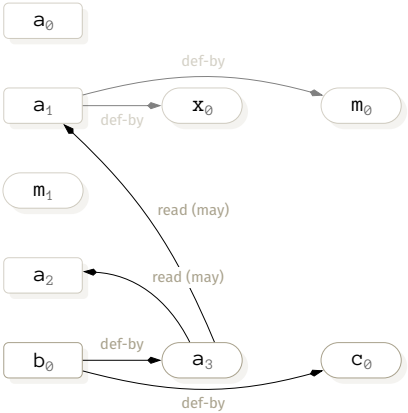
$x_0 \leftarrow 21$
$y_0 \leftarrow 2$
$z_0 \leftarrow x_1 * y_1$



Resulting Dataflow

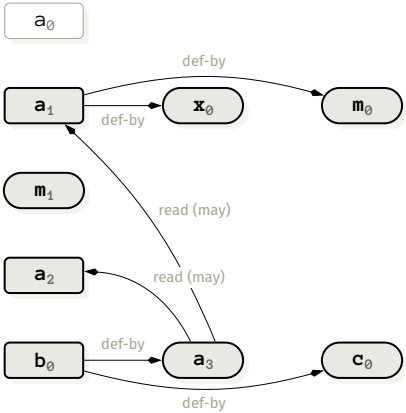



```
> a0 ← 3
  a1 ← x0 * m0

  if(m1 > 3) {
    a2 ← 5
  }

  b0 ← a3 + c0
```



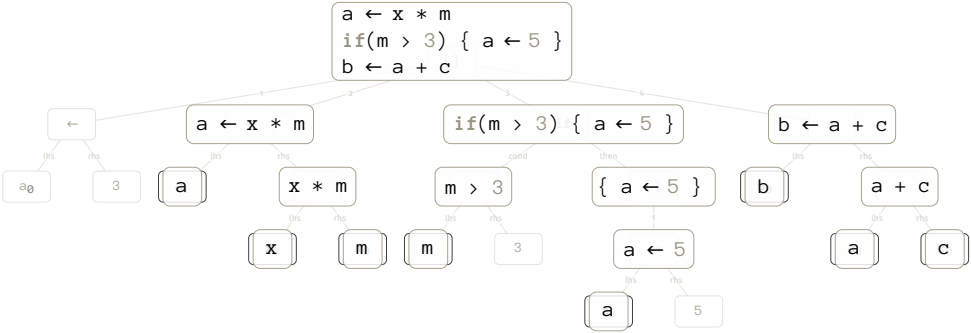
Slicing, I


```
a0 ← 3
a1 ← x0 * m0

if(m1 > 3) {
  a2 ← 5
}

b0 ← a3 + c0
```



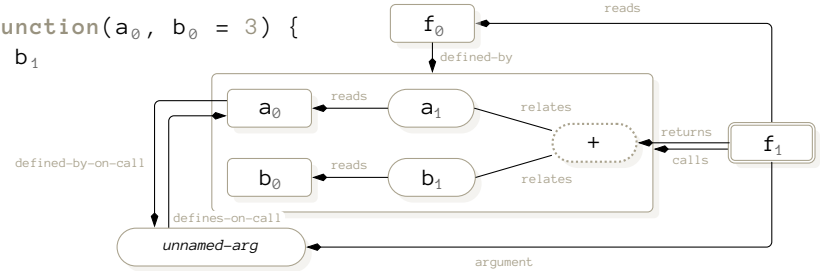
Slicing, II


Parse
Normalize
Dataflow
Slice
Reconstruct


a ← x * m
if(m > 3) { a ← 5 }
b ← a + c
1
2
3
4
←
a ← x * m
if(m > 3) { a ← 5 }
b ← a + c
lhs
rhs
a0
3
a
x * m
m > 3
{ a ← 5 }
b
a + c
cond
then
x
m
m
3
a ← 5
a
c
a
5

There Is More…

```
f₀ ← function(a₀, b₀ = 3) {
    a₁ + b₁
}
f₁(39)
```



Definition-Retrieval

```
paste(
   "(*|descendant-or-self::exprlist/*)[self::FUNCTION or self::OP-LAMBDA]/
       following-sibling::SYMBOL_FORMALS[text() = '{token_quote}' and @line1 <= {
       row}]",
   "(*|descendant-or-self::exprlist/*)[LEFT_ASSIGN[preceding-sibling::expr[count
       (*)=1]/SYMBOL[text() = '{token_quote}' and @line1 <= {row}] and following-
       sibling::expr[@start > {start} or @end < {end}]]]",
   "(*|descendant-or-self::exprlist/*)[RIGHT_ASSIGN[following-sibling::expr[count
       (*)=1]/SYMBOL[text() = '{token_quote}' and @line1 <= {row}] and preceding-
       sibling::expr[@start > {start} or @end < {end}]]]",
   "(*|descendant-or-self::exprlist/*)[EQ_ASSIGN[preceding-sibling::expr[count(*)=
       1]/SYMBOL[text() = '{token_quote}' and @line1 <= {row}] and following-
       sibling::expr[@start > {start} or @end < {end}]]]",
   "forcond/SYMBOL[text() = '{token_quote}' and @line1 <= {row}]",
sep = "|")
```

References I

[1] Ana Trisovic et al. “A Large-Scale Study on Research Code Quality and Execution”. 2022

[2] *The Comprehensive R Archive Network — cran.r-project.org*. 2024

[3] Florian Sihler and Matthias Tichy. “Statically Analyzing the Dataflow of R Programs”. 2024

[4] Olivier Flückiger et al. “R melts brains: an IR for first-class environments and lazy effectful arguments”. 2019

[5] R Core Team. *R Language Definition*. 2023

[6] Inese Drudze et al. *Apple phenology data set and R script, related to publication "Full flowering phenology of apple tree (Malus domestica) in Pūre orchard, Latvia from 1959 to 2019"*. June 2021

[7] Rathijit Sen et al. *ROSA: R Optimizations with Static Analysis*. 2017

[8] Florian Sihler. “Constructing a static program slicer for R programs”. 2023

[9] Mark Weiser. “Program Slicing”. July 1984

[10] Florian Sihler et al. “On the Anatomy of Real-World R Code for Static Analysis”. 2024

References II

[11] Liang Ma et al. *Predicting range shifts of pikas (Mammalia, Ochotonidae) in China under scenarios incorporating land-use change, climate change, and dispersal limitations.* Aug. 2021

[12] Joshua Robertson. *Social hierarchy reveals thermoregulatory trade- offs in response to repeated stressors.* Oct. 2020

[13] Duncan Lang et al. *CodeDepends. Analysis of R Code for Reproducible Research and Code Comprehension.* 2018

[14] Luke Tierney. *codetools: Code Analysis Tools for R.* 2023

[15] Joris Chau. *checkglobals: Static Analysis of R-Code Dependencies.* 2023

[16] Nick Ulle and Duncan Temple Lang. *rstatic: Low-level Static Analysis Tools for R Code.* 2019

[17] Duncan Lang et al. *CodeAnalysis. Tools for static analysis of R code.* 2023

[18] Nick Ulle and Duncan Temple Lang. *RTypeInference: Infer Types of Inputs and Outputs for R Expressions.* 2021

[19] Mark Padgham. *pkgstats.* 2021

[20] Henrik Bengtsson. *globals: Identify Global Objects in R Expressions.* 2022

References III

[21] Matthew Lau. *Rclean: A Tool for Writing Cleaner, More Transparent Code*. 2022

[22] Jim Hester et al. *lintr: A 'Linter' for R Code*. 2023

[23] Maciej Bartoszuk and Marek Gagolewski. *SimilaR: R Source Code Similarity Evaluation*. 2020

[24] Juan Cruz Rodriguez. *rco: The R Code Optimizer*. 2021

[25] Gabor Csardi. *cyclocomp: Cyclomatic Complexity of R Code*. 2023

[26] Antoine Fabri. *flow: View and Browse Code Using Flow Diagrams*. 2023

[27] Maarten van Kessel. *PaRe: A Way to Perform Code Review or QA on Other Packages*. 2023

[28] Dan Kary. *dfgraph: Visualize R Code with Data Flow Graphs*.

[29] Evan Patterson. *The algebra and machine representation of statistical models*. 2020

[30] Randy Lai. *languageserver: Language Server Protocol*. 2023

[31] Posit team. *RStudio: Integrated Development Environment for R*. 2023

[32] Gianluca Amato and Francesca Scozzari. "Random: R-Based Analyzer for Numerical Domains". 2012

[33] Joseph Wonsil et al. "Reproducibility as a Service". 2023

References IV

[34] R Core Team. *R: A Language and Environment for Statistical Computing*. 2023

[35] Tomas Kalibera et al. “A fast abstract syntax tree interpreter for R”. 2014

[36] Olivier Flückiger et al. “Sampling optimized code for type feedback”. 2020

[37] Alexander Bertram. “Renjin: A new r interpreter built on the jvm”. 2013

[38] Radford M Neal. “Speed Improvements in pqR: Current Status and Future Plans”. 2014

[39] *Microsoft R Open Source*. 2019

[40] John Garvin. *RCC: A compiler for the R language for statistical computing*. 2004