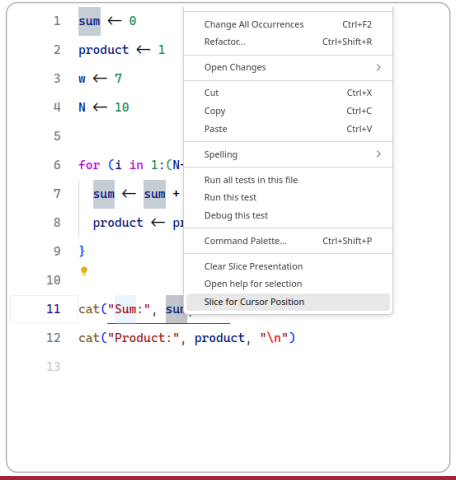
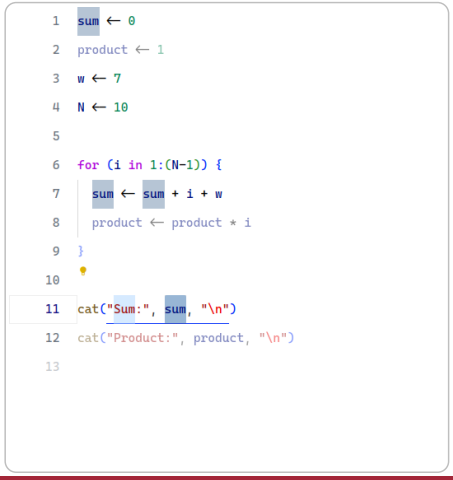




Statically Analyzing the Dataflow of R Programs With *flowR*

IRSER '24 | Ulm University | Florian Sihler | January 16, 2024

Software Engineering
Programming Languages

universität ulm

The R Programming Language

- R is mainly designed for statistical computing[1]
  - Heavily used in research (e.g., social science)[2]
  - Ranks 6th on PYPL[3]
  - Over 20 000 packages on CRAN[1]
- Most users are from non-computer-science domains
- Several problems in practice[2, 4]
  - Replication
  - Program comprehension
  - Missing tool support
- Demand for software engineering practices[5]

[5] Thimbleby, “Improving Science That Uses Code” (2023, Oxford University Press)
[4] Wonsil et al., “Reproducibility as a Service” (2023, Software: Practice and Experience)
[3] https://pypl.github.io/ [archived]
[2] Trisovic et al., “A Large-Scale Study on Research Code Quality and Execution” (2022, Nature Publishing Group)
[1] https://cran.r-project.org/

R Scripts

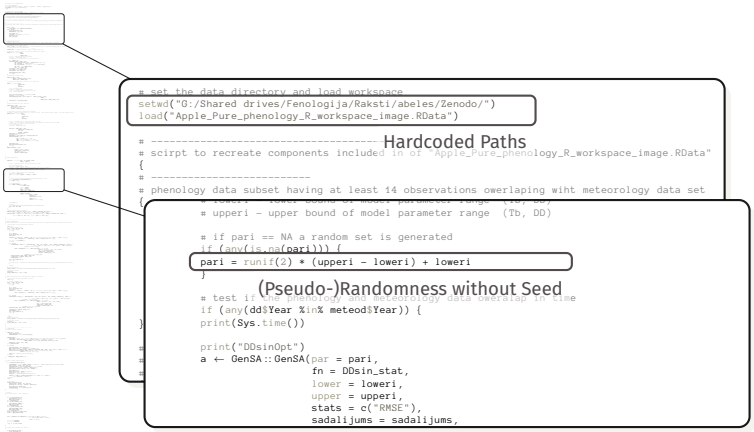
We analyzed 4 083 R-files

```
# 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
```

[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![2]



[2] 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

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

- String-based code evaluation

```
[8] 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

[8] Robertson, *Social hierarchy reveals thermoregulatory trade- offs in response to repeated stressors* (2020, Zenodo) [L. 68ff]
[7] 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

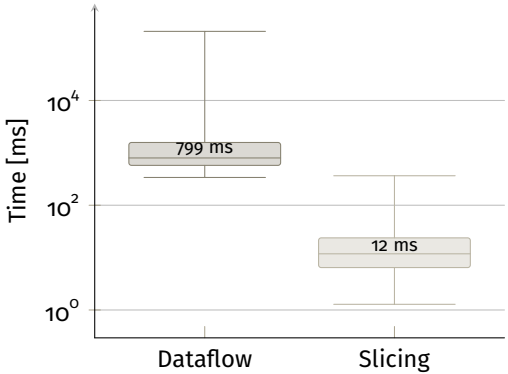
The Goal of flowR

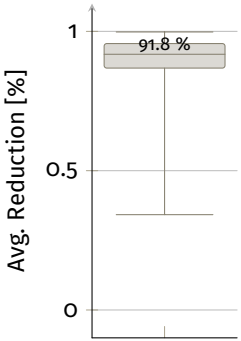
- Interested in a single figure
- ≈ *70 %* reduction

[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)

Performance Measurements

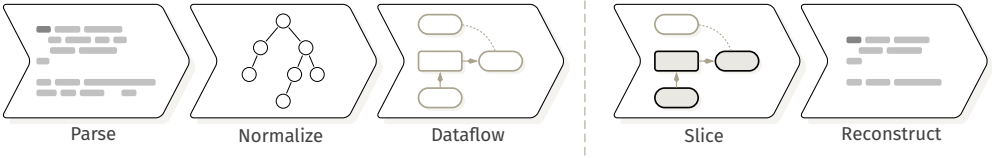
- We generated *every* possible variable of interest
- Dataflow results can be cached





99^{th} percentile

The Architecture

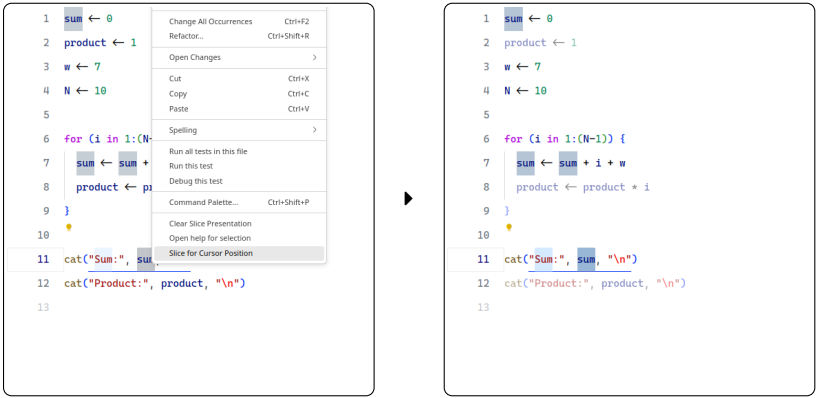


[9] Sihler, “Constructing a static program slicer for R programs” (2023, Ulm University)
[10] Weiser, “Program Slicing” (1984, IEEE Transactions on Software Engineering)

Visual Studio Code Integration

Rudimentary VSCode integration at:

github.com/eagleoutice/vscode-R



Using flowR

github.com/Code-Inspect/flowr
github.com/eagleoutice/vscode-R
hub.docker.com/r/eagleoutice/flowr
npmjs.com/package/@eagleoutice/flowr

Server

```
{
  "type": "request-file-analysis",
  "id":         "1",
  "filetoken": "123",
  "content":    "x←1;x*y"
}

{
  "type": "request-slice",
  "id":         "2",
  "filetoken": "123",
  "criterion": ["1@x"]
}
```

Library

```
const s = new SteppingSlicer({
  shell, tokenMap,
  request:
     requestFromInput("x←1; x*y"),
  criterion: ['1@x'],
})
const slice =
   await s.allRemainingSteps()
```

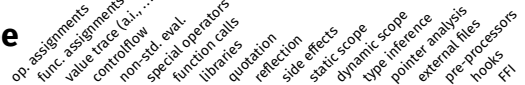
REPL

```
R> :parse "x←1; x*y"
exprlist
├ expr
│ ├ expr
│ │ └ SYMBOL "x" (1:1)
[...]

R> :dataflow* "x ← 1; x * y"
https://mermaid.live/edit#base64:eyJj...
```

```
docker run -it --rm eagleoutice/flowr
```

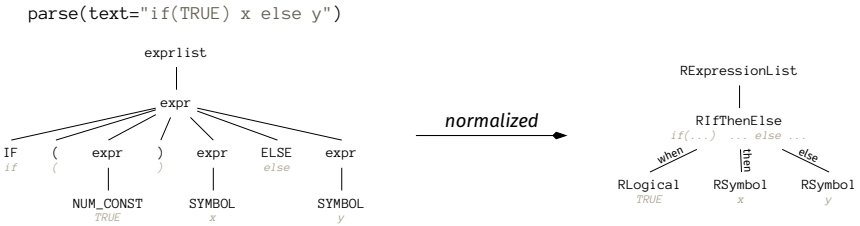
Appendix

The R Code Static Analysis Landscape



	goal	method	impl. lang.
[11] {CodeDepends}	static analysis	AST visitor	R
[12] {codetools}	static analysis	AST visitor	R
[13] {checkglobals}	missing libs.	AST visitor	R, C
[14] {rstatic}	static analysis	AST visitor	R
[15] {CodeAnalysis}	static analysis	AST visitor	R
[16] {RTypeInference}	type inference	AST visitor	R
[17] {pkgstats}	package insight	ctags & gtags	R, C++
[18] {globals}	distributed env.	AST visitor	R
[19] {Rclean}	debug/refactor	PDG traversal	R
[20] {lintr}	linting	XPath, visitor	R
[21] {SimilaR}	plagiarism	PDG, visitor	R, C++
[22] {rco}	optimization	AST visitor	R
[23] {cyclocomp}	static analysis	AST visitor	R
[24] {flow}	visualize, debug	AST visitor	R, C
[25] {PaRe}	code review	Regex	R
[26] {dfgraph}	static analysis	AST visitor	R
[27] {rflowgraph}	call graph	AST visitor	R
[28] {languageserver}	editor support	XPath, visitor	R, C
[29] RStudio	editor support	AST visitor	Java, C++, TS, ...
[30] ROSA	optimization	visitor	C++, R
[31] Random	abstract int.	trace & visitor	R
[4] RaaS	reproducibility	AST visitor	Python, R
[32] GNU R	execute R	bytecode	C, Fortran, R
[33] FastR	execute R	AST visitor	Java, C, R, ...
[34] Ř	execute R	SSA, bytecode	C++, R, C, ...
[35] renjin	execute R	SSA, CFG	R, Java, C, ...
[36] pqR	execute R	bytecode	C, R, ...
[37] MRO	execute R	bytecode	R, C, Fortran, ...
[38] RCC	transpile C	CFG, bytecode	C/C++, Fortran, R, ...

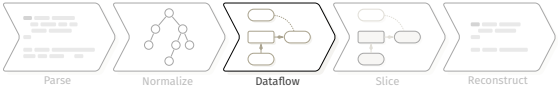
Parse & Normalize



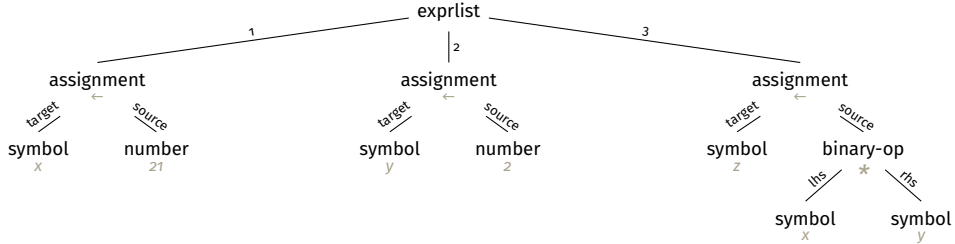
- Normalizing constants, namespacing, operators, …
- We use the “R language definition”[39] as a basis

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

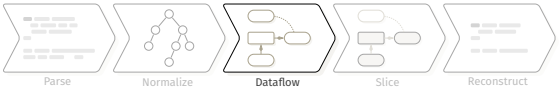
Dataflow



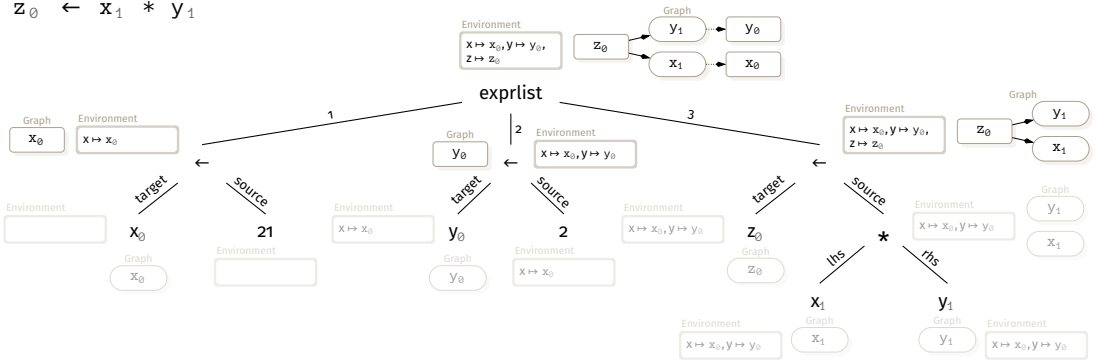
$x_{\emptyset} \leftarrow 21$
$y_{\emptyset} \leftarrow 2$
$z_{\emptyset} \leftarrow x_1 * y_1$



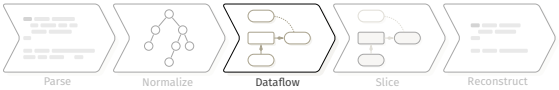
Dataflow



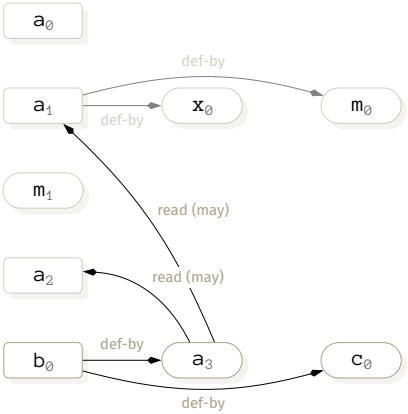
$x_\emptyset \leftarrow 21$
$y_\emptyset \leftarrow 2$
$z_\emptyset \leftarrow x_1 * y_1$



Resulting Dataflow

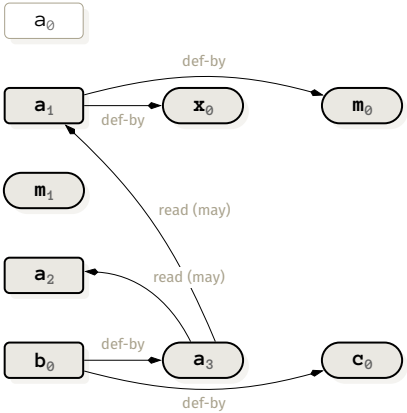



```
> a₀ ← 3
  a₁ ← x₀ * m₀

  if(m₁ > 3) {
    a₂ ← 5
  }

  b₀ ← a₃ + c₀
```



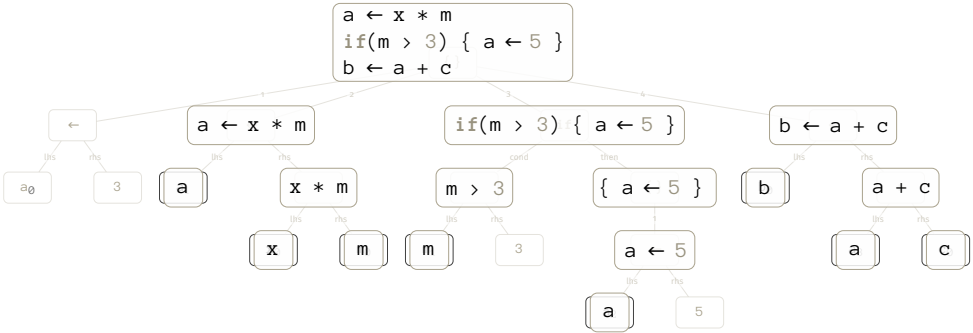
Slicing, I


```
a₀ ← 3
a₁ ← x₀ * m₀

if(m₁ > 3) {
  a₂ ← 5
}

b₀ ← a₃ + c₀
```



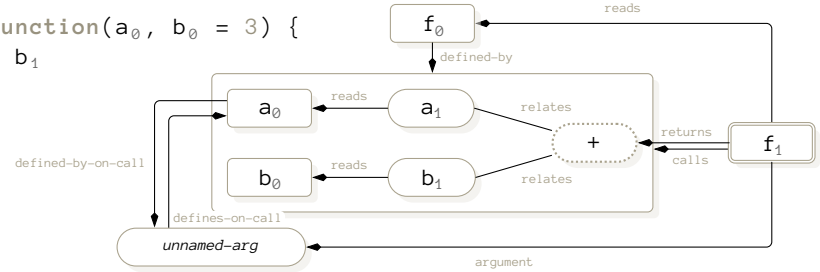
Slicing, II


Parse
Normalize
Dataflow
Slice
Reconstruct


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

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

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

[3] *PYPL — PopularitY of Programming Language index*. 2024

[4] Joseph Wonsil et al. “Reproducibility as a Service”. 2023

[5] Harold Thimbleby. “Improving Science That Uses Code”. 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] 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

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

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

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

References II

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

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

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

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

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

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

[17] Mark Padgham. *pkgstats.* 2021

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

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

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

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

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

References III

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

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

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

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

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

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

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

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

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

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

[33] Tomas Kalibera et al. "A fast abstract syntax tree interpreter for R". 2014

[34] Olivier Flückiger et al. "Sampling optimized code for type feedback". 2020

[35] Alexander Bertram. "Renjin: A new r interpreter built on the jvm". 2013

References IV

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

[37] *Microsoft R Open Source*. 2019

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

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