Package: tccquickr (via r-universe)

June 6, 2026

Title Experimental R Code Transformation Framework on Top of 'Rtinycc'

Version 0.0.0.9000

Description Provides an experimental and intentionally narrow declare()-annotated R code transformation and R-to-C compilation framework built on top of 'Rtinycc'. The package contains frontend parsing, typed IR, middle-end kernel rewrites, target C emission, and backend-neutral compilation helpers while keeping the underlying TinyCC toolchain and FFI runtime in 'Rtinycc' <https://github.com/sounkou-bioinfo/Rtinycc>.

License GPL (>= 3)

Depends R (>= 4.4.0)

Imports Rtinycc

Suggests bench, tinytest, quickr

Remotes sounkou-bioinfo/Rtinycc

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/sounkou-bioinfo/tccquickr, https://sounkou-bioinfo.r-universe.dev/tccquickr, https://github.com/sounkou-bioinfo/Rtinycc

BugReports https://github.com/sounkou-bioinfo/tccquickr/issues

Config/pak/sysreqs make

Repository https://sounkou-bioinfo.r-universe.dev

Date/Publication 2026-06-06 17:50:24 UTC

RemoteUrl https://github.com/sounkou-bioinfo/tccquickr

RemoteRef HEAD

RemoteSha 7be115abe14a0b1296e1d9322bf9703ae8f5cec0

Contents

format.tccq_type 2
print.tccq_module 2
print.tccq_type 3
tccq_analyze 3
tccq_backends 4
tccq_compile 5
tccq_jit 6

Index **8**

`format.tccq_type` *Format a* `tccq` *type*

Description

Format a `tccq` type

Usage

```
## S3 method for class 'tccq_type'
format(x, ...)
```

Arguments

`x`	A `tccq_type` object.
`...`	Unused.

Value

A compact character representation.

`print.tccq_module` *Print a* `tccq` *module summary*

Description

Print a `tccq` module summary

Usage

```
## S3 method for class 'tccq_module'
print(x, ...)
```

Arguments

x	A tccq_module object.
...	Unused.

Value

x, invisibly.

---

print.tccq_type *Print a* tccq *type*

Description

Print a tccq type

Usage

```
## S3 method for class 'tccq_type'
print(x, ...)
```

Arguments

x	A tccq_type object.
...	Unused.

Value

x, invisibly.

---

tccq_analyze *Analyze an R function for typed-compilation reconnaissance*

Description

tccq_analyze() collects high-level AST facts and optional compiler-package observations for a function. This is intentionally a side-band analysis to improve compiler diagnostics and help with boundary-reachability, while keeping the typed-IR path as the optimization frontier.

Usage

```
tccq_analyze(fn)
```

Arguments

fn	R function.

Details

The output is intentionally conservative and mostly advisory.

Value

A list of class `tccq_analysis` with fields:

- `formals`: formals metadata
- `ast`: AST observations (calls, symbols, assignments, loops)
- `compiler`: `compiler` package metadata when available
- `recommendations`: conservative suggestions for lowering success

See Also

`tccq_compile()`, `tccq_jit()`

Examples

```
f <- function(x, y = 1) {
  declare(type(x = double(NA), y = double(NA)))
  sum((x + y) * y)
}
tccq_analyze(f)
```

---

tccq_backends *tccq backend factories*

Description

Backend objects control how emitted C is handled by `tccq_compile()`.

Usage

```
tccq_backend_source()

tccq_backend_tinycc()

tccq_backend_shlib()
```

Details

- `tccq_backend_source()` returns generated C source without compiling it.
- `tccq_backend_tinycc()` compiles and loads the emitted C in memory through `Rtinycc`.
- `tccq_backend_shlib()` compiles the emitted C as a shared library through `R CMD SHLIB` and loads it with `dyn.load()`. This lives in the same general deployment space as `callme`.

Backends declare capabilities internally, and `tccq_compile()` validates the selected backend against the current target, compile context, and explicit boundary APIs before compiling.

Value

A tccq_backend object suitable for the backend = argument of tccq_compile().

Examples

```
tccq_backend_source()$name
tccq_backend_tinycc()$name
tccq_backend_shlib()$name
```

---

tccq_compile *Experimental tccq R-to-C compiler entry point*

Description

This is the package's current compiler path. The current milestone supports declared scalar and vector arithmetic, generic fold-style reducers, local bindings, indexed reads, contiguous slices, and local indexed/range writes.

Usage

```
tccq_compile(
  fn,
  mode = c("compile", "code", "ir"),
  fallback = c("hard", "auto"),
  backend = tccq_backend_tinycc(),
  target = tccq_target_c_rapi(),
  extlibs = list(),
  debug = FALSE
)
```

Arguments

fn	R function with a leading declare(type(...)) annotation.
mode	One of "compile", "code", or "ir".
fallback	One of "hard" or "auto". In "hard" mode unsupported calls are rejected. In "auto" mode unsupported calls may lower to explicit r_eval boundary nodes.
backend	Backend object. Use tccq_backend_source(), tccq_backend_tinycc(), or tccq_backend_shlib(). Defaults to TinyCC via Rtinycc.
target	Target object. Defaults to C + R C API emission.
extlibs	Optional list of tccq_external_library descriptors.
debug	Print generated C source before compiling.

Details

Current statement semantics are intentionally strict: rebinding a local name is rejected, direct mutation of formal arguments is rejected, and indexed or range assignment currently requires a scalar right-hand side.

Value

A compiled callable, C source string, or module IR depending on mode.

`tccq_jit` *Runtime-specialized JIT dispatcher for declared-typed* `tccq` *functions*

Description

`tccq_jit()` wraps a declared `tccq` function with a per-call dispatch cache keyed by observed argument signatures. Each distinct signature is compiled at most once (optionally with exact shape guards) and then reused on subsequent calls.

Usage

```
tccq_jit(
  fn,
  fallback = c("hard", "auto"),
  backend = tccq_backend_tinycc(),
  target = tccq_target_c_rapi(),
  extlibs = list(),
  debug = FALSE,
  exact = TRUE
)
```

Arguments

`fn`	R function with a leading `declare(type(...))` annotation.
`fallback`	One of "hard" or "auto". In "hard" mode unsupported calls are rejected. In "auto" mode unsupported calls may lower to explicit `r_eval` boundary nodes.
`backend`	Backend object. Use `tccq_backend_source()`, `tccq_backend_tinycc()`, or `tccq_backend_shlib()`.
`target`	Target object. Defaults to C + R C API emission.
`extlibs`	Optional list of `tccq_external_library()` descriptors.
`debug`	Print generated C source before each (first-time) signature compilation.
`exact`	If `TRUE`, each signature is specialized on exact observed shape (`length` for rank-1 and `nrow`/`ncol` for rank-2) and corresponding argument checks are emitted into C entry setup.

Details

This is intentionally a thin layer over `tccq_compile()` and is aimed at low-latency interactive workflows where many calls share the same runtime shape, e.g. fixed-size vectors/matrices.

Value

A callable R function with signature-cache semantics. Repeated calls with the same observed signature reuse the compiled callable.

See Also

tccq_compile(), tccq_backend_tinycc(), tccq_backend_shlib()

Examples

```
f <- function(x) {
  declare(type(x = double(NA)))
  x + 1
}
jf <- tccq_jit(f, exact = TRUE)
jf(c(1, 2, 3))
```

Index

format.tccq_type, 2

print.tccq_module, 2
print.tccq_type, 3

tccq_analyze, 3
tccq_backend_shlib *(*tccq_backends*)*, 4
tccq_backend_shlib(), *7*
tccq_backend_source *(*tccq_backends*)*, 4
tccq_backend_tinycc *(*tccq_backends*)*, 4
tccq_backend_tinycc(), *7*
tccq_backends, 4
tccq_compile, 5
tccq_compile(), *4*, *5*, *7*
tccq_jit, 6