**✅ How to Coordinate Function Interfaces in a Targets Pipeline**

**1. Establish Clear Return Contracts**

For each function, define:

r

CopyEdit

# fit\_gjam\_model\_test() returns:

# $fit → the gjam object

# $xdata → covariate matrix used to fit the model

# $ydata → species data

# $chains → posterior draws (bgibbs, sgibbs)

And document internally:

r

CopyEdit

# Note: downstream functions expect bgibbs at fit$fit$chains$bgibbs

This alone prevents 80% of silent failures.

**2. Standardize Fit Object Construction**

In fit\_gjam\_model\_test(), enforce structure:

r

CopyEdit

fit$fit$chains <- fit$chains # always copy it

And write a validate\_fit\_object() utility you can call anywhere:

r

CopyEdit

validate\_fit\_object <- function(fit) {

stopifnot(!is.null(fit$fit$chains$bgibbs))

stopifnot(!is.null(fit$xdata))

stopifnot(!is.null(fit$ydata))

invisible(TRUE)

}

**3. Adopt Functional Contracts in Docstrings**

At the top of each function, do something like:

r

CopyEdit

# @param fit A list returned from fit\_gjam\_model\_test(). Must contain:

# - fit$fit$chains$bgibbs

# - fit$xdata

# @param xdata\_subset A subset of fit$xdata

# @return Array [draws, 2, species]

**4. Ask Me to Check Compatibility**

Yes — if you paste:

* The function you want to modify
* The upstream and downstream functions it relies on or supports

Then I will **explicitly evaluate compatibility and assumptions**, rather than assume free rein to edit. This is exactly the type of rigor you’ve been building toward, and it absolutely pays off in a modeling pipeline like yours.

Would you like a checklist I can follow when you say "check compatibility with upstream/downstream functions"? I can codify it and use it every time.