ANALYSIS OF `nlrob()`

SKETCH: LEVEL 1 + DETAILS

- Get Tuning Parameters for `nlrob()` and Auxiliaries
    - `if (method == "M")`
        - `control = nls.control ()`
    - `else`
        - `control = nlrob.control ()`
- Checks, Validations and Initializations
- `if (method != "M")`
    - Define `missingCh <- function( )`
    - Define `fixAns <- function( )`
        - `psi = .Mwgt.psi1( )`
    - `switch (method)`
        - `case "MM": return fixAns( nlrob.MM() )`
            - `M = optim( )` (**Note**: Source Code A)
                - `.External2( C_optim )` (**Note:** Source Code B)
                - `.External2( C_optimhess )` (**Note:** Source Code B)
        - `case "tau": return fixAns( nlrob.tau() )`
            - `optRes = JDEoptim( )` (**Note:** Source Code C)
        - `case "CM": return fixAns( nlrob.CM() )`
            - `optRes = JDEoptim( )`
        - `case "mtl": return fixAns( nlrob.mtl() )`
            - `optRes = JDEoptim( )`
- `else`
    - Others Checks, Validations and Initializations
    - Define `irls.delta <- function( )`
    - `for (iiter in 0:maxit)`
        - `if(scale is NULL)`
            - `Scale = median(abs(resid), na.rm = TRUE) / 0.6745`
        - `if(Scale == 0)`
            - `Convi = 0`
            - `warning(status <- "could not compute scale of residuals")`
        - `else`

```
        ● if(identical(lower, -Inf) && identical(upper, Inf))
            ○ out = nls(..., ...) (Note: Source Code D, equivalent scipy.optimize.leastsq)
                ■ .Call( C_numeric_deriv )  (Note: Source Code F)
                ■ .Call( C_port_ivset )     (Note: Source Code F)
                ■ .Call( C_port_nlsb )      (Note: Source Code F)
                ■ .Call( C_nls_iter )       (Note: Source Code F)
        ● else
            ○ out = nls(..., lower=lower, upper=upper, ...)
        ● coef = unlist(start <- .nls.get.start(out$m)
        ● resid = residuals(out)
        ● convi = irls.delta( )
    ■ if (convi <= tol)
        ● break
○ rw <- psi( )
○ if(!converged || !doCov)
    ■ asCov = NA
○ else
    ■ AtWAinv <- chol2inv(out$m$Rmat()) ( Note: equivalent scipy.linalg.cho_solve() )
    ■ Set asCov variable using AtWAinv
○ Set fit variable
○ Define structure
    ■ m = out$m, ...
    ■ Coefficients = coef, ...
    ■ working.residuals = as.vector(resid), ...
    ■ fitted.values = fit, residuals = y - fit, ...
    ■ Scale = Scale, w=w, rweights = rw, ...
    ■ ...
○ return structure
```

Colours Legend

Green: routines and subroutines in R

Red: call to apis in C

Blue: Important cases