Regular readers will recall the “[utility belt](https://rud.is/b/2018/04/08/dissecting-r-package-utility-belts/)” post from back in April of this year. This is a follow-up to a request made asking for a list of all the % infix functions in those files.

We’re going to:

* collect up all of the sources
* parse them
* find all the definitions of % infix functions
* write them to a file

We’ll start by grabbing the data from the previous post and look at it as a refresher:

library(stringi)

library(tidyverse)

utils <- read\_rds(url("https://rud.is/dl/utility-belt.rds"))

utils

## # A tibble: 1,746 x 13

## permsissions links owner group size month day year\_hr path date pkg fil file\_src

## 1 -rw-r--r-- 0 hornik users 1658 Jun 05 2016 AHR/R… 2016-06-05 AHR util… "## \\int f(x)dg(x) …

## 2 -rw-r--r-- 0 ligges users 12609 Dec 13 2016 ALA4R… 2016-12-13 ALA4R util… "## some utility fun…

## 3 -rw-r--r-- 0 hornik users 0 Feb 24 2017 AWR.K… 2017-02-24 AWR.… util… ""

## 4 -rw-r--r-- 0 ligges users 4127 Aug 30 2017 Alpha… 2017-08-30 Alph… util… "#\n#' Assign API ke…

## 5 -rw-r--r-- 0 ligges users 121 Jan 19 2017 Amylo… 2017-01-19 Amyl… util… "make\_decision <- fu…

## 6 -rw-r--r-- 0 herbrandt herbrandt 52 Aug 10 2017 BANES… 2017-08-10 BANE… util… "#' [@importFrom](http://twitter.com/importFrom) dply…

## 7 -rw-r--r-- 0 ripley users 36977 Jan 06 2015 BEQI2… 2015-01-06 BEQI2 util… "#' \tRemove Redunda…

## 8 -rw-r--r-- 0 hornik users 34198 May 10 2017 BGDat… 2017-05-10 BGDa… util… "# A more memory-eff…

## 9 -rwxr-xr-x 0 ligges users 3676 Aug 14 2016 BGLR/… 2016-08-14 BGLR util… "\n readBinMat=funct…

## 10 -rw-r--r-- 0 ripley users 2547 Feb 04 2015 BLCOP… 2015-02-04 BLCOP util… "###################…

## # ... with 1,736 more rows

Note that we somewhat expected the file source to potentially come in handy at a later date and also expected the need to revisit that post, so the [R data file](https://rud.is/dl/utility-belt.rds) [←*direct link to RDS*] included a file\_src column.

Now, let’s find all the source files with at least one infix definition, collect them together and parse them so we can do more code spelunking:

filter(utils, stri\_detect\_fixed(file\_src, "`%")) %>% # only find sources with infix definitions

pull(file\_src) %>%

paste0(collapse="\n\n") %>%

parse(text = ., keep.source=TRUE) -> infix\_src

str(infix\_src, 1)

## length 1364 expression(dplyr::`%>%`, `%||%` <- function(a, b) if (is.null(a)) b else a, get\_pkg\_path <- function(ctx) { pkg\_| \_\_truncated\_\_ ...

## - attr(\*, "srcref")=List of 1364

## - attr(\*, "srcfile")=Classes 'srcfilecopy', 'srcfile'

## - attr(\*, "wholeSrcref")= 'srcref' int [1:8] 1 0 15768 0 0 0 1 15768

## ..- attr(\*, "srcfile")=Classes 'srcfilecopy', 'srcfile'

We can now take all of that lovely parsed source and tokenize it to work with the discrete elements in a very tidy manner:

infix\_parsed <- tbl\_df(getParseData(infix\_src)) # tbl\_df() is mainly for pretty printing

infix\_parsed

## # A tibble: 118,242 x 9

## line1 col1 line2 col2 id parent token terminal text

## 1 1 1 1 24 1 -10 COMMENT TRUE #' @impor…

## 2 2 1 2 10 4 -10 COMMENT TRUE #' @export

## 3 3 1 3 12 10 0 expr FALSE ""

## 4 3 1 3 5 7 10 SYMBOL\_PACKAGE TRUE dplyr

## 5 3 6 3 7 8 10 NS\_GET TRUE ::

## 6 3 8 3 12 9 10 SYMBOL TRUE `%>%`

## 7 5 1 5 49 51 0 expr FALSE ""

## 8 5 1 5 6 16 18 SYMBOL TRUE `%||%`

## 9 5 1 5 6 18 51 expr FALSE ""

## 10 5 8 5 9 17 51 LEFT\_ASSIGN TRUE <-

## # ... with 118,232 more rows

We just need to find a sequence of tokens that make up a function definition, then whittle those down to ones that look like our % infix names:

pat <- c("SYMBOL", "expr", "LEFT\_ASSIGN", "expr", "FUNCTION") # pattern for function definition

# find all of ^^ sequences (there's a good twitter discussion on this abt a month ago)

idx <- which(infix\_parsed$token == pat[1]) # find location of match of start of seq

# look for the rest of the sequences starting at each idx position

map\_lgl(idx, ~{

all(infix\_parsed$token[.x:(.x+(length(pat)-1))] == pat)

}) -> found

f\_defs <- idx[found] # starting indices of all the places where functions are defined

# filter ^^ to only find infix ones

infix\_defs <- f\_defs[stri\_detect\_regex(infix\_parsed$text[f\_defs], "^`\\%")]

# there aren't too many, but remember we're just searching `util` functions

length(infix\_defs)

## [1] 106

Now, write it out to a file so we can peruse the infix functions:

# nuke a file and fill it with the function definition

cat("", sep="", file="infix\_functions.R")

walk2(

getParseText(infix\_parsed, infix\_parsed$id[infix\_defs]), # extract the infix name

getParseText(infix\_parsed, infix\_parsed$id[infix\_defs + 3]), # extract the function definition body

~{

cat(.x, " <- ", .y, "\n\n", sep="", file="infix\_functions.R", append=TRUE)

}

)

Infix\_Functions.R

|  |
| --- |
|  |
| `%||%` <- function(a, b) if (is.null(a)) b else a |
|  |  |
|  | `%diag\*%` <- function(d, X) d \* X |
|  |  |
|  | `%\*diag%` <- function(X, d) t(t(X) \* d) |
|  |  |
|  | `%nin%` <- function(x, table) !(x %in% table) |
|  |  |
|  | `%sub\_in%` <- function(x, table) x[x %in% table] |
|  |  |
|  | `%sub\_nin%` <- function(x, table) x[x %nin% table] |
|  |  |
|  | `%notchin%` <- function(lhs, rhs) { |
|  | !{lhs %chin% rhs} |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) ifelse(is.null(x), y, x) |
|  |  |
|  | `%||%` <- function(x, y) if (is.null(x)) y else x |
|  |  |
|  | `%||%` <- function(x, y) if (is.null(x)) y else x |
|  |  |
|  | `%||%` <- function(x, default\_val) { |
|  | if (is.null(x)) return(default\_val) |
|  | x |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%p%` <- function(e1,e2) return(paste0(e1,e2)) |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (!is.null(x)) { |
|  | x |
|  | } else { |
|  | y |
|  | } |
|  | } |
|  |  |
|  | `%==%` <- function(x, y) identical(x, y) |
|  |  |
|  | `%!=%` <- function(x, y) !identical(x, y) |
|  |  |
|  | `%:::%` <- function(pkg, fun) { |
|  | getFromNamespace(fun, asNamespace(pkg)) |
|  | } |
|  |  |
|  | `%||%` <- function(left, right){ |
|  | if (!is.null(left)){ |
|  | return(left) |
|  | } |
|  | right |
|  | } |
|  |  |
|  | `%pin%` <- function(x, table) pmatch(x, table, nomatch = 0L) > 0L |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x) || length(x) <= 0) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%??%` <- function(lhs, rhs) |
|  | if (is\_empty(lhs)) rhs else lhs |
|  |  |
|  | `%In%` <- function(x,Intv) |
|  | { |
|  | if (is.integer(x)) x <- as.double(x) |
|  | if (is.matrix(Intv)) |
|  | { |
|  | Intv <- Intervals(Intv) |
|  | } |
|  | distance\_to\_nearest(x,Intv) == 0 |
|  | } |
|  |  |
|  | `%<<%` <- function(a, b) bitwShiftL(a, b) |
|  |  |
|  | `%>>%` <- function(a, b) bitwShiftR(a, b) |
|  |  |
|  | `%&%` <- function(a, b) bitwAnd(a, b) |
|  |  |
|  | `%notin%` <- function(x, table) match(x, table, nomatch = 0L) == 0L |
|  |  |
|  | `%or%` <- function(lhs, rhs) if (is.null(lhs)) rhs else lhs |
|  |  |
|  | `%||%` <- function (x, y) if (is.null(x)) y else x |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) return(y) |
|  | x |
|  | } |
|  |  |
|  | `%~~%` <- function(x, y) { |
|  | if (length(x) == 0L) return(y) |
|  | x |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) y else x |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) if(is.null(x)) y else x |
|  |  |
|  | `%||%` <- function(x, y) if (is.null(x)) y else x |
|  |  |
|  | `%||%` <- function(lhs, rhs) { |
|  | if (is.null(lhs)) { |
|  | rhs |
|  | } else { |
|  | lhs |
|  | } |
|  | } |
|  |  |
|  | `%R%` <- function(lhs, rhs){ |
|  | if(length(lhs)) lhs else rhs |
|  | } |
|  |  |
|  | `%M%` <- function(lhs, rhs) { |
|  | if (lhs < rhs) { |
|  | old.lhs <- lhs |
|  | lhs <- rhs |
|  | rhs <- old.lhs |
|  | } |
|  | x <- lhs %/% rhs |
|  | y <- lhs %% rhs |
|  | return(c(quotient = x, remainder = y)) |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%notin%` <- function(x, y) { |
|  | !(x %in% y) |
|  | } |
|  |  |
|  | `%||%` <- function (a, b) { |
|  | if (is.null(a)) b else a |
|  | } |
|  |  |
|  | `%||%` <- function(a, b) if (!is.null(a)) a else b |
|  |  |
|  | `%fin%` <- function(a, tbl) fmatch(a, tbl, 0L, NULL) > 0L |
|  |  |
|  | `%||%` <- function(x, y){ |
|  | if (is.null(x)) y else x |
|  | } |
|  |  |
|  | `%||%` <- function (lhs, rhs) { |
|  | lres <- withVisible(eval(lhs, envir = parent.frame())) |
|  | if (is.null(lres$value)) { |
|  | eval(rhs, envir = parent.frame()) |
|  | } else { |
|  | if (lres$visible) { |
|  | lres$value |
|  | } else { |
|  | invisible(lres$value) |
|  | } |
|  | } |
|  | } |
|  |  |
|  | `%&&%` <- function(lhs, rhs) { |
|  | lres <- withVisible(eval(lhs, envir = parent.frame())) |
|  | if (!is.null(lres$value)) { |
|  | eval(rhs, envir = parent.frame()) |
|  | } else { |
|  | if (lres$visible) { |
|  | lres$value |
|  | } else { |
|  | invisible(lres$value) |
|  | } |
|  | } |
|  | } |
|  |  |
|  | `%+%` <- function(x, y) { |
|  | stopifnot(is.character(x), is.character(y)) |
|  | paste0(x, y) |
|  | } |
|  |  |
|  | `%inr%` <- function(x,range) |
|  | { |
|  | if (!is.numeric(range) || length(range) != 2) |
|  | { |
|  | stop("Range must be a vector of 2 numeric values") |
|  | } |
|  | if (!is.numeric(x)) |
|  | { |
|  | stop("x must be numeric") |
|  | } |
|  | else |
|  | { |
|  | if (diff(range) < 0) |
|  | { |
|  | stop('Range must be increasing') |
|  | } |
|  |  |
|  | x >= range[1] & x <= range[2] |
|  | } |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%diag\*%` <- function(d, X) d \* X |
|  |  |
|  | `%\*diag%` <- function(X, d) t(t(X) \* d) |
|  |  |
|  | `%||%` <- function(a,b) if(is.null(a)) b else a |
|  |  |
|  | `%+%` <- function(a, b) paste0(a, b) |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x) || length(x) <= 0) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%==%` <- function(x, y) { |
|  | identical(x, y) |
|  | } |
|  |  |
|  | `%!=%` <- function(x, y) { |
|  | !identical(x, y) |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%contains%` <- function(x,y)contains(y,x) |
|  |  |
|  | `%||%` <- function(x, y) if (is.null(x)) y else x |
|  |  |
|  | `%din%` <- function(x, y) { |
|  | by <- intersect(names(x), names(y)) |
|  | nx <- nrow(x <- as.data.frame(x)) |
|  | ny <- nrow(y <- as.data.frame(y)) |
|  | bx <- x[,by,drop=FALSE] |
|  | by <- y[,by,drop=FALSE] |
|  | names(bx) = names(by) <- paste("V", seq\_len(ncol(bx)), sep="") |
|  | bz <- do.call(paste, c(rbind(bx, by), sep="\r")) |
|  | bx <- bz[seq\_len(nx)] |
|  | by <- bz[nx + seq\_len(ny)] |
|  | comm <- match(bx, by, 0) |
|  | x[comm > 0,] |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) y else x |
|  | } |
|  |  |
|  | `%nin%` <- function(x, y) { |
|  | !(x %in% y) |
|  | } |
|  |  |
|  | `%+%` <- function(lhs, rhs) { |
|  | check\_string(lhs) |
|  | check\_string(rhs) |
|  | paste0(lhs, rhs) |
|  | } |
|  |  |
|  | `%||%` <- function(lhs, rhs) { |
|  | if (!is.null(lhs)) { lhs } else { rhs } |
|  | } |
|  |  |
|  | `%s%` <- function(lhs, rhs) { |
|  | assert\_that(is.string(lhs)) |
|  | list(lhs) %>% |
|  | c(as.list(rhs)) %>% |
|  | do.call(what = sprintf) |
|  | } |
|  |  |
|  | `%+%` <- function(lhs, rhs) { |
|  | paste0(lhs, rhs) |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%@%` <- function(x, name) attr(x, name, exact = TRUE) |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%notin%` <- function(needle, haystack) { |
|  | ! (needle %in% haystack) |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%||%` <- function(a, b) if (is.null(a)) b else a |
|  |  |
|  | `%||%` <- function (a, b) if (!is.null(a)) a else b |
|  |  |
|  | `%:::%` <- function (p, f) get(f, envir = asNamespace(p)) |
|  |  |
|  | `%::%` <- function (p, f) get(f, envir = asNamespace(p)) |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) { |
|  | y |
|  | } else { |
|  | x |
|  | } |
|  | } |
|  |  |
|  | `%==%` <- function(x, y) { |
|  | identical(x, y) |
|  | } |
|  |  |
|  | `%||%` <- function(a,b) if(is.null(a)) b else a |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%:::%` <- function(p, f) { |
|  | get(f, envir = asNamespace(p)) |
|  | } |
|  |  |
|  | `%||%` <- function(a, b) { |
|  | if (is.null(a)) b else a |
|  | } |
|  |  |
|  | `%++%` <- function(l, r) { append(l, r) } |
|  |  |
|  | `%||%` <- function(a,b) if(is.null(a)) b else a |
|  |  |
|  | `%+%` <- function(a,b) paste(a, b, sep = '') |
|  |  |
|  | `%+|%` <- function(a,b) paste(a, b, sep = '|') |
|  |  |
|  | `%+&%` <- function(a,b) paste(a, b, sep = '&') |
|  |  |
|  | `%||%` <- function(l, r) { |
|  | if (is.null(l)) r else l |
|  | } |
|  |  |
|  | `%notin%` <- function(a, b) !(a %in% b) |
|  |  |
|  | `%AND%` <- function(x, y) { |
|  | if (!is.null(x) && !is.na(x)) |
|  | if (!is.null(y) && !is.na(y)) |
|  | return(y) |
|  | return(NULL) |
|  | } |
|  |  |
|  | `%AND%` <- function (x, y) |
|  | { |
|  | if (!is.null(x) && !is.na(x)) |
|  | if (!is.null(y) && !is.na(y)) |
|  | return(y) |
|  | return(NULL) |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (!is.null(x)) x else y |
|  | } |
|  |  |
|  | `%OR%` <- function(x, y) { |
|  | if (is.null(x) || isTRUE(is.na(x))) |
|  | y |
|  | else |
|  | x |
|  | } |
|  |  |
|  | `%AND%` <- function(x, y) { |
|  | if (!is.null(x) && !is.na(x)) |
|  | if (!is.null(y) && !is.na(y)) |
|  | return(y) |
|  | return(NULL) |
|  | } |
|  |  |
|  | `%.%` <- function(x, y) { |
|  | paste(x, y, sep='') |
|  | } |
|  |  |
|  | `%OR%` <- function(x, y) { |
|  | if (is.null(x) || isTRUE(is.na(x))) |
|  | y |
|  | else |
|  | x |
|  | } |
|  |  |
|  | `%AND%` <- function(x, y) { |
|  | if (!is.null(x) && !is.na(x)) |
|  | if (!is.null(y) && !is.na(y)) |
|  | return(y) |
|  | return(NULL) |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%||%` <- function(a, b) { |
|  | if (is.null(a)) b else a |
|  | } |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x)) y else x |
|  | } |
|  |  |
|  | `%identical%` <- function(x, y) identical(x, y) |
|  |  |
|  | `%assert\_class%` <- function(dat, class){ |
|  | assert\_class(dat = dat, class = class) |
|  | } |
|  |  |
|  | `%||%` <- function(a, b) if (is.null(a)) b else a |
|  |  |
|  | `%||%` <- function(x, y) { |
|  | if (is.null(x) || length(x) == 0L) y else x |
|  | } |
|  |  |
|  | `%+%` <- function(chr1, chr2){ |
|  | paste0(chr1, chr2) |
|  | } |
|  |  |
|  | `%||%` <- function(l, r) if (is.null(l)) r else l |
|  |  |
|  | `%+%` <- function(l, r) { |
|  | assert\_string(l) |
|  | assert\_string(r) |
|  | paste0(l, r) |
|  | } |
|  |  |
|  | `%||%` <- function(a, b) if (is.null(a)) b else a |
|  |  |

Here's an overview of what you can expect to find:

# A tibble: 39 x 2

name n

1 `%||%` 47

2 `%+%` 7

3 `%AND%` 4

4 `%notin%` 4

5 `%:::%` 3

6 `%==%` 3

7 `%!=%` 2

8 `%\*diag%` 2

9 `%diag\*%` 2

10 `%nin%` 2

11 `%OR%` 2

12 `%::%` 1

13 `%??%` 1

14 `%.%` 1

15 `%@%` 1

16 `%&&%` 1

17 `%&%` 1

18 `%+&%` 1

19 `%++%` 1

20 `%+|%` 1

21 `%<<%` 1

22 `%>>%` 1

23 `%~~%` 1

24 `%assert\_class%` 1

25 `%contains%` 1

26 `%din%` 1

27 `%fin%` 1

28 `%identical%` 1

29 `%In%` 1

30 `%inr%` 1

31 `%M%` 1

32 `%notchin%` 1

33 `%or%` 1

34 `%p%` 1

35 `%pin%` 1

36 `%R%` 1

37 `%s%` 1

38 `%sub\_in%` 1

39 `%sub\_nin%` 1