



```
> expandChain(output$plot)
```

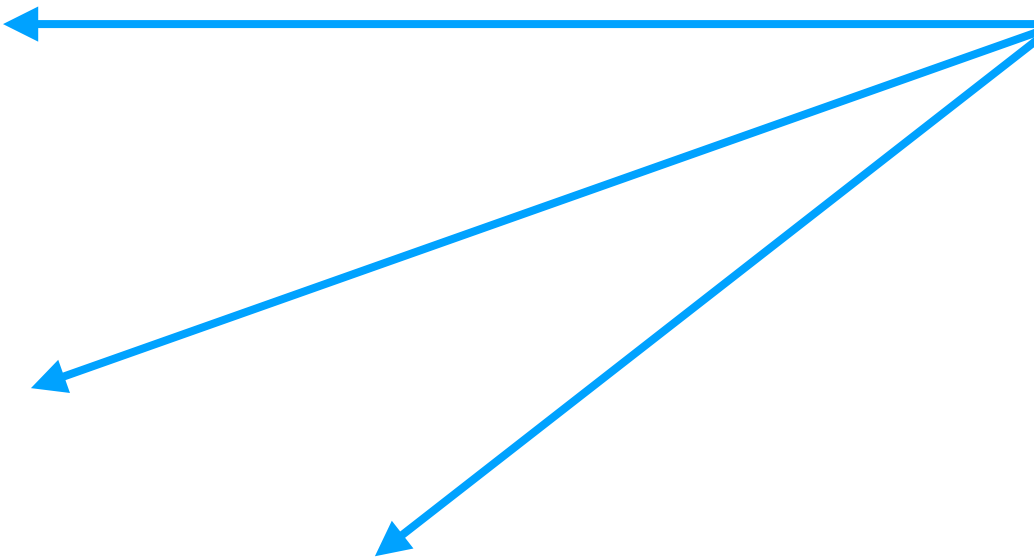
```
downloads <-  
  cranlogs::cran_downloads(  
    "shiny",  
    from = ..(format(Sys.Date() - 365)),  
    to = Sys.Date()  
  )
```

```
downloads_rolling <-  
  downloads %>%  
    mutate(count = zoo::rollapply(count, 7, mean, fill = "extend"))
```

```
ggplot(downloads_rolling, aes(date, count)) + geom_line()
```

**Step 3: Generate code with xpandchain()**

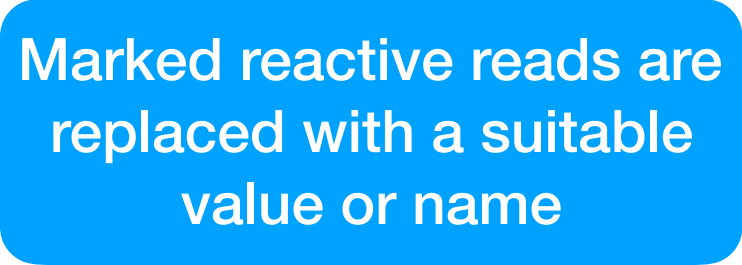
Marked reactive reads are  
replaced with a suitable  
value or name



### Step 3: Generate code with `expandChain()`

```
> expandChain(output$plot)
```

```
downloads <-  
  cranlogs::cran_downloads(  
    "shiny",  
    from = ..(format(Sys.Date() - 365)),  
    to = Sys.Date()  
  )  
  
downloads_rolling <-  
  downloads %>%  
    mutate(count = zoo::rollapply(count, 7, mean, fill = "extend"))  
  
ggplot(downloads_rolling, aes(date, count)) + geom_line()
```



Marked reactive reads are replaced with a suitable value or name

The diagram illustrates the replacement of reactive reads in the provided R code. A blue box on the right contains the text "Marked reactive reads are replaced with a suitable value or name". Three blue arrows originate from this box and point to specific parts of the code: the first arrow points to the string "shiny" in the `cran_downloads()` function call; the second arrow points to the `downloads` variable in the `downloads_rolling <- downloads %>%` line; and the third arrow points to the `downloads_rolling` variable in the `ggplot(downloads_rolling, ...)` function call.

### Step 3: Generate code with `expandChain()`

```
> expandChain(output$plot)
```

```
downloads <-  
  cranlogs::cran_downloads(  
    "shiny",  
    from = ..(format(Sys.Date() - 365)),  
    to = Sys.Date()  
  )
```

```
downloads_rolling <-  
  downloads %>%  
    mutate(count = zoo::rollapply(count, 7, mean, fill = "extend"))
```

```
ggplot(downloads_rolling, aes(date, count)) + geom_line()
```

Other code wrapped in `..()` is evaluated (i.e. unquoted)

