The free group in R: introducing the freegroup package

Robin K. S. Hankin
University of Stirling

Abstract

Here I present the `freegroup` package for working with the free group on a finite set of symbols. The package is vectorised; internally it uses an efficient matrix-based representation for free group objects but uses a configurable print method. A range of R-centric functionality is provided. It is available on CRAN at `https://CRAN.R-project.org/package=freegroup`. To cite the **freegroup** package, use Hankin (2022).

Keywords: Free group, Tietze form.

1. Introduction

The free group is an interesting and instructive mathematical object with a rich structure that illustrates many concepts of elementary group theory. The **freegroup** package provides some functionality for manipulating the free group on a finite list of symbols. Informally, the *free group* $(X, \circ)$ on a set $S = \{a, b, c, \ldots, z\}$ is the set X of *words* that are objects like $W = c^{-4}bb^2aa^{-1}ca$, with a group operation of string juxtaposition. Usually one works only with words that are in "reduced form", which has successive powers of the same symbol combined, so W would be equal to $c^{-4}b^3ca$; see how b appears to the third power and the a term in the middle has vanished. The group operation of juxtaposition is formally indicated by $\circ$, but this is often omitted in algebraic notation; thus, for example $a^2b^{-3}c^2 \circ c^{-2}ba = a^2b^{-3}c^2c^{-2}ba = a^2b^{-2}ba$.

1.1. Formal definition

If X is a set, then a group F is called *the free group on* X if there is a set map $\Psi\colon X \longrightarrow F$, and for any group G and set map $\Phi\colon X \longrightarrow G$, there is a unique homomorphism $\alpha\colon F \longrightarrow G$ such that $\alpha \circ \Psi = \Phi$, that is, the diagram below commutes:

$$\begin{array}{ccc} X & \xrightarrow{\Psi} & F \\ & \searrow^{\Phi} & \downarrow{\alpha} \\ & & G \end{array}$$

It can be shown that F is unique up to group isomorphism; every group is a quotient of a free group.

1.2. Existing work

Computational support for working with the free group is provided as part of a number of algebra systems including GAP, Sage (The Sage Developers 2019), and sympy (Meurer *et al.* 2017) although in those systems the emphasis is on finitely presented groups, not in scope for the **freegroup** package. There are also a number of closed-source proprietary systems which are of no value here.

2. The package in use

In the **freegroup** package, a word is represented by a two-row integer matrix; the top row is the integer representation of the symbol and the second row is the corresponding power. For example, to represent $a^2b^{-3}ac^2a^{-2}$ we would identify a as 1, b as 2, etc and write

```
> (M <- rbind(c(1,2,3,3,1),c(2,-3,2,3,-2)))

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    3    1
[2,]    2   -3    2    3   -2
```

(see how negative entries in the second row correspond to negative powers). Then to convert to a more useful form we would have

```
> library("freegroup")
> (x <- free(M))

[1] a^2.b^-3.c^5.a^-2
```

The representation for R object `x` is still a two-row matrix, but the print method is active and uses a more visually appealing scheme. The default alphabet used is `letters`. We can coerce strings to free objects:

```
> (y <- as.free("aabbbcccc"))

[1] a^2.b^3.c^4
```

The free group operation is simply juxtaposition, represented here by the plus symbol:

```
> x + y

[1] a^2.b^-3.c^5.b^3.c^4
```

(see how the a "cancels out" in the juxtaposition).

2.1. Notation

The package generally uses additive notation but also, as an experimental feature, supports multipicative notation. Thus `x+y == x*y`. One motivation for the use of "+" rather than "*" is that Python uses "+" for appending strings:

```
>>> "a" + "abc"
'aabc'
>>>
```

However, note that the "+" symbol is usually reserved for commutative and associative operations; string juxtaposition is associative.

Multiplication by integers—denoted in **freegroup** idiom by "*"—is also defined. Suppose we want to concatenate 5 copies of x:

```
> x*5

[1] a^2.b^-3.c^5.b^-3.c^5.b^-3.c^5.b^-3.c^5.b^-3.c^5.a^-2
```

This operation is vectorized:

```
> x*(0:3)

[1] 0                                    a^2.b^-3.c^5.a^-2
[3] a^2.b^-3.c^5.b^-3.c^5.a^-2           a^2.b^-3.c^5.b^-3.c^5.b^-3.c^5.a^-2
```

There are a few methods for creating free objects, for example:

```
> abc(1:9)

[1] a                  a.b                a.b.c              a.b.c.d
[5] a.b.c.d.e          a.b.c.d.e.f        a.b.c.d.e.f.g      a.b.c.d.e.f.g.h
[9] a.b.c.d.e.f.g.h.i
```

And we can also generate random free objects:

```
> rfree(10,4)

 [1] a^4.d^3.b^-8      d^-3.a^6          b^-4.c^-4.a^4.c^3 c^3.d^2.b^4.a^-4
 [5] 0                 d^-4.a^4.d^6      a^-1.b^8          d^-3.c^2.b^-2
 [9] a^3.d^3.c^-1.b^4  c.d^-2.b
```

Inverses are calculated using unary or binary minus:

```
> (p <- rfree(10,4))

 [1] 0                 c^-3.d^2.c^3.a^-3 a^-7.c^2          b^3.a^2.c^3.a^-1
 [5] c^3.b^2.d^-2      c^-1.b^-2.a^-1    b^5               c.a^-2.b.c^-4
 [9] d^2.b.d^3         b^-4.d^4

> -p
```

```
 [1] 0                 a^3.c^-3.d^-2.c^3 c^-2.a^7          a.c^-3.a^-2.b^-3
 [5] d^2.b^-2.c^-3     a.b^2.c           b^-5              c^4.b^-1.a^2.c^-1
 [9] d^-3.b^-1.d^-2    d^-4.b^4
```

```
> p-p
```

```
 [1] 0 0 0 0 0 0 0 0 0 0
```

We can take the "sum" of a vector of free objects simply by juxtaposing the elements:

```
> sum(p)
```

```
[1] c^-3.d^2.c^3.a^-10.c^2.b^3.a^2.c^3.a^-1.c^3.b^2.d^-2.c^-1.b^-2.a^-1.b^5.c.a^-2.b.c^-4.
```

Powers are defined as per group conjugation: `x^y == y^{-1}xy` (or, written in additive notation, `-y+x+y`):

```
> p
```

```
 [1] 0                 c^-3.d^2.c^3.a^-3 a^-7.c^2          b^3.a^2.c^3.a^-1
 [5] c^3.b^2.d^-2      c^-1.b^-2.a^-1    b^5               c.a^-2.b.c^-4
 [9] d^2.b.d^3         b^-4.d^4
```

```
> a <- alpha(26)
> p^a
```

```
 [1] 0                         z^-1.c^-3.d^2.c^3.a^-3.z z^-1.a^-7.c^2.z
 [4] z^-1.b^3.a^2.c^3.a^-1.z z^-1.c^3.b^2.d^-2.z      z^-1.c^-1.b^-2.a^-1.z
 [7] z^-1.b^5.z              z^-1.c.a^-2.b.c^-4.z     z^-1.d^2.b.d^3.z
[10] z^-1.b^-4.d^4.z
```

Thus:

```
> sum(p^a) == sum(p)^a
```

```
[1] TRUE
```

The experimental multiplicative notation allows us to have the equivalent of $(xy)^z = x^z y^z$ and $x^{(yz)} = (x^y)^z$:

```
> x <- rfree()
> y <- rfree()
> z <- rfree()
> (x*y)^z == x^z * y^z
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
> x^(y*z) == (x^y)^z

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

In additive notation these manifest, somewhat unappealingly, as `(x+y)^z == x^z + y^z` and `x^(y+z) = (x^y)^z`. Further, note that the distributive law $x * (y + z) = x * y + x * z$ is now incorrect [we have, again somewhat unappealingly, $x * (y + z) = x * y * z = x + y + z$] but it can be resurrected if we reinterpret addition as (vector) juxtaposition:

```
> x * c(y, z) == c(x*y, x*z)

 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

> c(x, y) * z == c(x*z, y*z)

 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

There is also a commutator bracket, defined as $[x, y] = x^{-1}y^{-1}xy$ or in package idiom `.[x,y]=-x-y+x+y`:

```
> .[p,a]

 [1] 0
 [2] a^3.c^-3.d^-2.c^3.z^-1.c^-3.d^2.c^3.a^-3.z
 [3] c^-2.a^7.z^-1.a^-7.c^2.z
 [4] a.c^-3.a^-2.b^-3.z^-1.b^3.a^2.c^3.a^-1.z
 [5] d^2.b^-2.c^-3.z^-1.c^3.b^2.d^-2.z
 [6] a.b^2.c.z^-1.c^-1.b^-2.a^-1.z
 [7] b^-5.z^-1.b^5.z
 [8] c^4.b^-1.a^2.c^-1.z^-1.c.a^-2.b.c^-4.z
 [9] d^-3.b^-1.d^-2.z^-1.d^2.b.d^3.z
[10] d^-4.b^4.z^-1.b^-4.d^4.z
```

If we have more than 26 symbols the print method runs out of letters:

```
> alpha(1:30)

 [1] a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y
[26] z  NA NA NA NA
```

If this is a problem (it might not be: the print method might not be important) it is possible to override the default symbol set:

```
> options(freegroup_symbols = state.abb)
> alpha(1:30)
```

```
 [1] AL AK AZ AR CA CO CT DE FL GA HI ID IL IN IA KS KY LA ME MD MA MI MN MS MO
[26] MT NE NV NH NJ
```

3. Conclusions and further work

The **freegroup** package furnishes a consistent and documented suite of reasonably efficient R-centric functionality. Further work might include the finitely presented groups but it is not clear whether this would be consistent with the precepts of R.

References

GAP (2018). *GAP – Groups, Algorithms, and Programming, Version 4.10.0*. The GAP Group. URL https://www.gap-system.org.

Hankin RKS (2022). "The free group in R." doi:10.48550/ARXIV.2212.05883.

Meurer A, *et al.* (2017). "SymPy: symbolic computing in Python." *PeerJ Computer Science*, **3**, e103. ISSN 2376-5992. doi:10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

The Sage Developers (2019). *SageMath, the Sage Mathematics Software System (Version 8.6)*. URL https://www.sagemath.org.

Affiliation:

Robin K. S. Hankin
University of Stirling
E-mail: hankin.robin@gmail.com