Special relativity in R: introducing the lorentz package

Robin K. S. Hankin
Auckland University of Technology

Abstract

Here I present the **lorentz** package for working with relativistic physics. The package includes functionality for Lorentz transforms and relativistic velocity addition, which is noncommutative and nonassociative.

Keywords: Lorentz transform, Lorentz group, Lorentz law, Lorentz velocity addition, special relativity, relativistic physics, Einstein velocity addition, Wigner rotation, gyrogroup, gyromorphism, gyrocommutative, gyroassociative, four velocity, three-velocity, nonassociative, noncommutative.

1. Introduction

In special relativity, the Lorentz transformations supercede their classical equivalent, the Galilean transforms (Goldstein 1980). Lorentz transforms operate on four-vectors such as the four-velocity or four-potential and are usually operationalised as multiplication by a 4×4 matrix. A Lorentz transform takes the components of an arbitrary four-vector as observed in one coordinate system and returns the components observed in another system which is moving at constant velocity with respect to the first.

The **lorentz** package provides R-centric functionality for Lorentz transformations. It deals with Lorentz boosts, converts between three-velocities and four-velocities, and provides computational support for the gyrogroup structure of three-velocities under successive addition.

2. Lorentz transforms

Although the most natural way to consider relative movement between coordinate systems is the four-velocity, it is often the case that analysis will begin by defining a three-velocity. In the **lorentz** package, three velocities are defined by the `as.3vel()` function, for example:

```
> u <- as.3vel(c(0.3,-0.4,0.8))
> u

      x    y   z
[1,] 0.3 -0.4 0.8
```

(note that the package uses units in which $c = 1$ by default, although this can be changed). The three-velocity u may be expressed as a four-velocity using the `as.4vel()` function:

```
> as.4vel(u)
```

```
           t        x         y        z
[1,] 3.015113 0.904534 -1.206045 2.412091
```

The corresponding coordinate transformation is given by `boost()`:

```
> boost(u)
```

```
          t          x          y          z
t  3.015113 -0.9045340  1.2060454 -2.4120908
x -0.904534  1.2037755 -0.2717007  0.5434014
y  1.206045 -0.2717007  1.3622676 -0.7245352
z -2.412091  0.5434014 -0.7245352  2.4490703
```

This matrix corresponds to the Lorentz transformation between a frame at rest, and one with a three velocity of `u` as defined above. Transforming an arbitrary four-vector from the rest frame to the moving one has straightforward R idiom:

```
> boost(u) %*% c(4,5,-2,3)
```

```
       [,1]
t -2.110579
x  4.574347
y -1.432463
z  1.864925
```

Note that coordinate transformation is effected by standard R operator `%*%`. Composition of two Lorentz transforms is also ordinary matrix multiplication:

```
> v <- as.3vel(c(0.4,0.2,-0.1))
> L <- boost(u) %*% boost(v)
> L
```

```
          t          x          y          z
t  3.256577 -2.2327055  0.5419596 -2.0800479
x -1.437147  1.6996791 -0.0237489  0.4194255
y  1.091131 -0.7581795  1.1190282 -0.6029155
z -2.519789  1.5878378 -0.2023170  2.1879612
```

But observe that the resulting transform is not a pure boost, as the spatial components are not symmetrical. We may decompose the matrix into a pure translation composed with an orthogonal matrix, which represents a coordinate rotation:

```
> (U <- orthog(L))
```

```
              t             x            y             z
t  1.000000e+00 -1.332268e-14 3.219647e-15 -1.509903e-14
x -1.054712e-14  9.458514e-01 1.592328e-01 -2.828604e-01
y  8.659740e-15 -1.858476e-01 9.801022e-01 -6.971587e-02
z -1.953993e-14  2.661311e-01 1.185098e-01  9.566241e-01
```

```
> (P <- pureboost(L))
```

```
           t          x          y          z
t  3.2565770 -2.2327055  0.5419596 -2.0800479
x -2.2327055  2.1711227 -0.2842745  1.0910491
y  0.5419596 -0.2842745  1.0690039 -0.2648377
z -2.0800479  1.0910491 -0.2648377  2.0164504
```

In the above, U should be orthogonal and L symmetric:

```
> crossprod(U) - diag(4)
```

```
              t             x             y             z
t -3.907985e-14 -3.010826e-14  7.711957e-15 -3.141176e-14
x -3.010826e-14 -2.575717e-14  6.314393e-15 -2.325917e-14
y  7.711957e-15  6.314393e-15 -1.554312e-15  5.384582e-15
z -3.141176e-14 -2.325917e-14  5.384582e-15 -2.187139e-14
```

```
> P - t(P)
```

```
  t x y z
t 0 0 0 0
x 0 0 0 0
y 0 0 0 0
z 0 0 0 0
```

(that is, zero to numerical precision).

3. Three velocities

The **lorentz** package includes functionality to compose three-velocities. Three velocities do not form a group under composition as the velocity addition law is not associative. Ungar (2006) shows that the velocity addition law is

$$\mathbf{u}\oplus\mathbf{v}=\frac{1}{1+\mathbf{u}\cdot\mathbf{v}}\left\{\mathbf{u}+\frac{\mathbf{v}}{\gamma_{\mathbf{u}}}+\frac{\gamma_{\mathbf{u}}\left(\mathbf{u}\cdot\mathbf{v}\right)\mathbf{u}}{1+\gamma_{\mathbf{u}}}\right\} \tag{1}$$

where $\gamma_{\mathbf{u}} = (1 - \mathbf{u}\cdot\mathbf{u})^{-1/2}$ and we are assuming $c = 1$. Ungar shows that, in general, $\mathbf{u}\oplus\mathbf{v} \neq \mathbf{v}\oplus\mathbf{u}$ and $(\mathbf{u}\oplus\mathbf{v})\oplus\mathbf{w} \neq \mathbf{u}\oplus(\mathbf{v}\oplus\mathbf{w})$. He also defines the binary operator $\ominus$ as $\mathbf{u}\ominus\mathbf{v} = \mathbf{u}\oplus(-\mathbf{v})$, and implicitly defines $\ominus\mathbf{u}\oplus\mathbf{v}$ to be $(-\mathbf{u})\oplus\mathbf{v}$. If we have

$$\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x} = -\left(\mathbf{u}\oplus\mathbf{v}\right)\oplus\left(\mathbf{u}\oplus\left(\mathbf{v}\oplus\mathbf{x}\right)\right) \tag{2}$$

Then Ungar shows that

$$\begin{aligned}
\mathbf{u}\oplus\mathbf{v} &= \mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\left(\mathbf{v}\oplus\mathbf{u}\right) && (3)\\
\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}\cdot\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{y} &= \mathbf{x}\cdot\mathbf{y} && (4)\\
\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\left(\mathbf{x}\oplus\mathbf{y}\right) &= \mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}\oplus\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{y} && (5)\\
\left(\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\right)^{-1} &= \left(\mathrm{gyr}\left[\mathbf{v},\mathbf{u}\right]\right) && (6)\\
\mathbf{u}\oplus\left(\mathbf{v}\oplus\mathbf{w}\right) &= \left(\mathbf{u}\oplus\mathbf{v}\right)\oplus\mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{w} && (7)\\
\left(\mathbf{u}\oplus\mathbf{v}\right)\oplus\mathbf{w} &= \mathbf{u}\oplus\left(\mathbf{v}\oplus\mathrm{gyr}\left[\mathbf{v},\mathbf{u}\right]\mathbf{w}\right) && (8)
\end{aligned}$$

Consider the following R session:

```
> library(lorentz)
> u <- as.3vel(c(-0.7,+0.2,-0.3))
> v <- as.3vel(c(+0.3,+0.3,+0.4))
> w <- as.3vel(c(+0.1,+0.3,+0.8))
> x <- as.3vel(c(-0.2,-0.1,-0.9))
> u

        x   y    z
[1,] -0.7 0.2 -0.3
```

Here we have three-vectors `u` etc. We can see that `u` and `v` do not commute:

```
> u+v

              x         y            z
[1,] -0.5454028 0.4815421 -0.004538856

> v+u

              x         y         z
[1,] -0.4292804 0.5723133 0.1324513
```

(the results differ). We can use equation 3

```
> (u+v)-gyr(u,v,v+u)

                x             y            z
[1,] 1.769252e-16 -7.077008e-16 1.230183e-16
```

showing agreement to within numerical error. It is also possible to use the functional idiom:

```
> f <- gyrfun(u,v)
> (u+v)-f(v+u)    # should be zero
```

```
                x             y            z
[1,] 1.769252e-16 -7.077008e-16 1.230183e-16
```

Function `gyrfun()` is vectorized, which means that it plays nicely with (R) vectors. Consider

```
> u9 <- r3vel(9)
> u9
```

```
                x           y           z
 [1,] -0.88140983 -0.13689387  0.04145539
 [2,]  0.29562685 -0.46267853  0.59150295
 [3,]  0.03128981 -0.28968888  0.20441596
 [4,]  0.40043893  0.39344933 -0.14176847
 [5,] -0.19973294  0.67263992  0.12945349
 [6,]  0.10911267 -0.39067525 -0.45961497
 [7,] -0.09587628 -0.00849401 -0.94458262
 [8,]  0.45247104 -0.42223214  0.15428704
 [9,] -0.68028666 -0.13739381  0.67547246
```

Then we can create a vectorized gyrofunction:

```
> f <- gyrfun(u9,v)
> f(x)
```

```
               x            y          z
 [1,]  0.0990291 -0.057356744 -0.9202736
 [2,] -0.1622890  0.139743907 -0.9022937
 [3,] -0.1775260  0.001733855 -0.9102096
 [4,] -0.3025603 -0.200663049 -0.8533414
 [5,] -0.1072810 -0.250100150 -0.8865330
 [6,] -0.3161515 -0.063063628 -0.8695236
 [7,] -0.4006386 -0.343987937 -0.7623392
 [8,] -0.2768451  0.055186735 -0.8833522
 [9,]  0.1743395  0.101339563 -0.9051718
```

Note that the package vectorization is transparent when using syntatic sugar:

```
> u9+x
```

```
                x           y          z
 [1,] -0.92328225 -0.17032095 -0.3117470
```

```
 [2,]  0.16753276 -0.79843619 -0.3137449
 [3,] -0.18864236 -0.42801449 -0.7793705
 [4,]  0.23723893  0.31114697 -0.8691034
 [5,] -0.37722498  0.63735777 -0.5980971
 [6,] -0.01582057 -0.39396531 -0.8954656
 [7,] -0.11878386 -0.02433181 -0.9906339
 [8,]  0.30829318 -0.55927199 -0.6829988
 [9,] -0.88695458 -0.20653920  0.3755772
```

(here, the addition operates using R's standard recycling rules).

3.1. Associativity

Three velocity addition is not associative:

```
> (u+v)+w
```

```
              x        y         z
[1,] -0.4646213 0.655437 0.5008326
```

```
> u+(v+w)
```

```
              x         y         z
[1,] -0.5489863 0.6672455 0.4160947
```

But we can use equations 7 and 8:

```
> (u+(v+w)) - ((u+v)+gyr(u,v,w))
```

```
                x             y              z
[1,] 6.916191e-16 -1.383238e-15 -6.916191e-16
```

```
> ((u+v)+w) - (u+(v+gyr(v,u,w)))
```

```
     x y            z
[1,] 0 0 5.353251e-16
```

3.2. Visualization of noncommutativity and nonassociativity of three-velocities

Consider the following three-velocities:

```
> u <- as.3vel(c(0.4,0,0))
> v <- seq(as.3vel(c(0.4,-0.2,0)), as.3vel(c(-0.3,0.9,0)),len=20)
> w <- as.3vel(c(0.8,-0.4,0))
```

```
> comm_fail1(u=u, v=v)
```

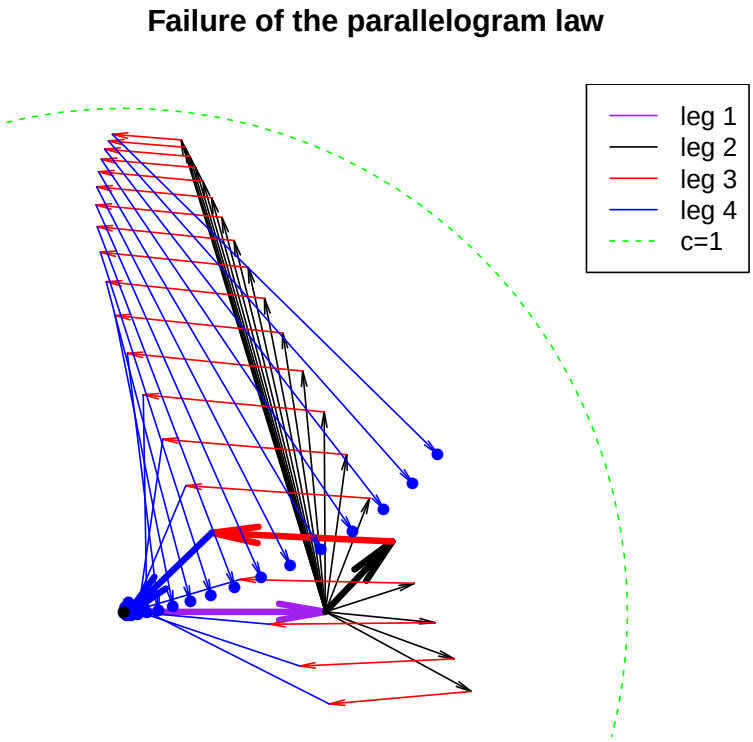

Figure 1: Failure of the commutative law for velocity composition in special relativity. The arrows show successive velocity boosts of $+\mathbf{u}$ (purple), $+\mathbf{v}$ (black), $-\mathbf{u}$ (red), and $-\mathbf{v}$ (blue) for $\mathbf{u}, \mathbf{v}$ as defined above. Velocity $\mathbf{u}$ is constant, while $\mathbf{v}$ takes a sequence of values. If velocity addition is commutative, the four boosts form a closed quadrilateral; the thick arrows show a case where the boosts almost close and the boosts nearly form a parallelogram. The blue dots show the final velocity after four successive boosts; the distance of the blue dot from the origin measures the combined velocity, equal to zero in the classical limit of low speeds. The discrepancy becomes larger and larger for the faster elements of the sequence $\mathbf{v}$

```
> comm_fail2(u=u, v=v)
```

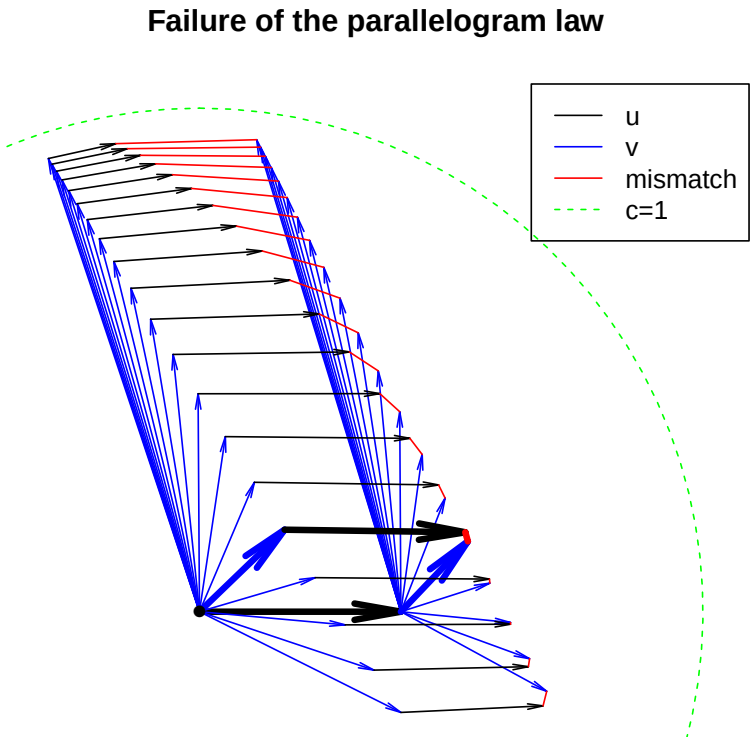

Figure 2: Another view of the failure of the commutative law in special relativity. The black arrows show velocity boosts of $\mathbf{u}$ and the blue arrows show velocity boosts of $\mathbf{v}$, with $\mathbf{u}, \mathbf{v}$ as defined above; $\mathbf{u}$ is constant while $\mathbf{v}$ takes a sequence of values. If velocity addition is commutative, then $\mathbf{u}+\mathbf{v} = \mathbf{v}+\mathbf{u}$ and the two paths end at the same point: the parallelogram is closed. The red lines show the difference between $\mathbf{u}+\mathbf{v}$ and $\mathbf{v}+\mathbf{u}$

```
> ass_fail(u=u, v=v, w=w, bold=10)
```

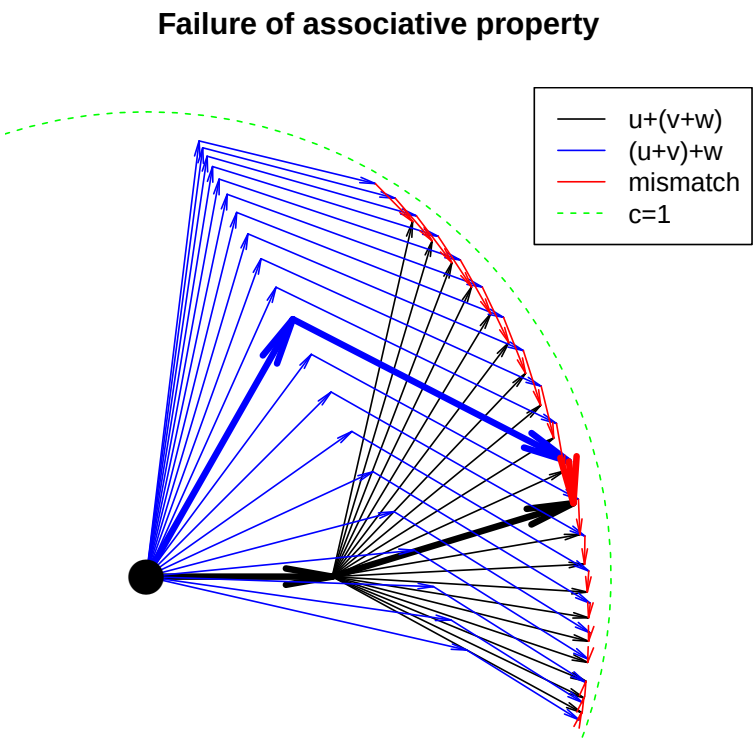

Figure 3: Failure of the associative law for velocity composition in special relativity. The arrows show successive boosts of $\mathbf{u}$ followed by $\mathbf{v}+\mathbf{w}$ (black lines), and $\mathbf{u}+\mathbf{v}$ followed by $\mathbf{w}$ (blue lines), for $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ as defined above; $\mathbf{u}$ and $\mathbf{w}$ are constant while $\mathbf{v}$ takes a sequence of values. The mismatch between $\mathbf{u}+(\mathbf{v}+\mathbf{w})$ and $(\mathbf{u}+\mathbf{v})+\mathbf{w}$ is shown in red

Objects **v** and **w** are single three-velocities, and object **v** is a vector of three velocities. We can see the noncommutativity of three velocity addition in figures 1 and 2, and the nonassociativity in figures 3.

3.3. The magrittr package: pipes

The **lorentz** package works nicely with **magrittr**. If we define

```
>  u <- as.3vel(c(+0.5,0.1,-0.2))
>  v <- as.3vel(c(+0.4,0.3,-0.2))
>  w <- as.3vel(c(-0.3,0.2,+0.2))
```

Then pipe notation operates as expected:

```
> jj1 <- u %>% add(v)
> jj2 <- u+v
> speed(jj1-jj2)

[1] 2.206363e-16
```

The pipe operator is left associative:

```
> jj1 <- u %>% add(v) %>% add(w)
> jj2 <- (u+v)+w
> speed(jj1-jj2)

[1] 7.392965e-17
```

If we want right associative addition, the pipe operator needs brackets:

```
> jj1 <- u %>% add(v %>% add(w))
> jj2 <- u+(v+w)
> speed(jj1-jj2)

[1] 3.446154e-17
```

3.4. Functional notation

It is possible to replace calls like `gyr(u,v,x)` with functional notation which can make for arguably more natural R idiom. If we have

```
> u <- as.3vel(c(0, 0.8, 0))
> v <- r3vel(5,0.9)
> x <- as.3vel(c(0.7, 0, -0.7))
> y <- as.3vel(c(0.1, 0.3, -0.6))
```

Then we can define `f()` to be the map $\mathbf{x} \mapsto \mathrm{gyr}\left[\mathbf{u},\mathbf{v}\right]\mathbf{x}$. In R:

```
> f <- gyrfun(u,v)
> f(w)

              x           y         z
[1,] -0.2522052  0.13944464 0.2948690
[2,] -0.3308892 -0.03740011 0.2431329
[3,] -0.3666863  0.11058449 0.1526835
[4,] -0.3498135 -0.02860336 0.2163617
[5,] -0.2872348  0.25698598 0.1464731
```

Then numerical verification of equations 4 and 6 is straightforward:

```
> prod3(f(x),f(y)) - prod3(x,y)

[1]  1.665335e-15 -4.218847e-15 -8.326673e-16 -1.998401e-15 -9.436896e-16
```

and

```
> f <- gyrfun(u,v)
> g <- gyrfun(v,u)
> f(g(x)) - g(f(x))

                  x             y             z
[1,]  5.551115e-14  1.957480e-14 -5.551115e-14
[2,] -7.771561e-14 -2.955804e-14  6.661338e-14
[3,]  1.665335e-14  9.089923e-15 -3.330669e-14
[4,] -4.440892e-14  1.424088e-14  4.440892e-14
[5,] -3.330669e-14  1.463596e-14  4.440892e-14
```

(zero to numerical precision). It is possible to use pipes together with the functional notation:

```
> x %<>% f %>% add(y)      # x <- f(x)+y
> x

             x          y          z
[1,] 0.7036623  0.1973652 -0.6791218
[2,] 0.5733663  0.5775843 -0.5772440
[3,] 0.6628360  0.1060987 -0.7380503
[4,] 0.5394796  0.4980063 -0.6758028
[5,] 0.6791534 -0.1629317 -0.7119076
```

4. SI units

The preceding material used units in which $c = 1$. Here I show how the package deals with units such as SI in which $c = 299792458 \neq 1$. For obvious reasons we cannot have a function called `c()` so the package gets and sets the speed of light with function `sol()`:

```
> sol(299792458)
> sol()

[1] 299792458
```

The speed of light is now 299792458 until re-set by `sol()` (an empty argument queries the speed of light). We can now consider speeds which are fast by terrestrial standards but involve only a small relativistic correction to the Galilean result:

```
> u <- as.3vel(c(100,200,300))
> u

       x   y   z
[1,] 100 200 300
```

The gamma correction term γ is only very slightly larger than 1 and indeed R's default print method suppresses the difference:

```
> gam(u)

[1] 1
```

However, we can display more significant figures by subtracting one:

```
> gam(u)-1

[1] 7.789325e-13
```

or alternatively we can use the `gamm1()` function which calculates $\gamma - 1$ more accurately for speeds $\ll c$:

```
> gamm1(u)

[1] 7.78855e-13
```

The Lorentz boost is again calculated by the `boost()` function:

```
> boost(u)

     t             x            y             z
t    1 -1.112650e-15 -2.22530e-15 -3.337950e-15
x -100  1.000000e+00  1.11265e-13  1.668975e-13
y -200  1.112650e-13  1.00000e+00  3.337950e-13
z -300  1.668975e-13  3.33795e-13  1.000000e+00
```

The boost matrix is not symmetrical, even though it is a pure boost, because $c \neq 1$. Note how the transformation is essentially the Galilean result, which may be calculated in the package exactly by setting the speed of light to infinity:

```
> sol(Inf)
> boost(u)

     t x y z
t    1 0 0 0
x -100 1 0 0
y -200 0 1 0
z -300 0 0 1
```

We can compose boosts as expected:

```
> sol(299792458)
> v <- as.3vel(c(400,-200,300))
> boost(u) %*% boost(v)

              t             x             y             z
t  1.000000e+00 -5.563250e-15  4.704692e-27 -6.675900e-15
x -5.000000e+02  1.000000e+00 -5.563250e-13  1.168283e-12
y -2.559179e-10  5.563250e-13  1.000000e+00  6.675900e-13
z -6.000000e+02  2.169668e-12 -6.675900e-13  1.000000e+00

> boost(v) %*% boost(u)

              t             x             y             z
t  1.000000e+00 -5.563250e-15 -2.847319e-27 -6.675900e-15
x -5.000000e+02  1.000000e+00  5.563250e-13  2.169668e-12
y  4.228102e-10 -5.563250e-13  1.000000e+00 -6.675900e-13
z -6.000000e+02  1.168283e-12  6.675900e-13  1.000000e+00
```

Again this is very close to the Galilean result (observe that the spatial components of the boost are almost equal to the identity matrix).

Appendix A. Active and passive transforms

Passive transforms are the usual type of transforms taught and used in relativity. However, sometimes active transforms are needed and it is easy to confuse the two. Here I will discuss passive transforms first, then active transforms.

Passive transforms

Consider the following canonical Lorentz transform in which we have only motion in the x-direction at speed v. The motion is from left to right, and this means that $v > 0$. The physical

interpretation is that I am at rest, and my friend is in his spaceship moving at speed v past me; and we are wondering what vectors which I measure in my own rest frame look like to him. The (passive) Lorentz transform is:

$$\begin{pmatrix} \gamma & \gamma v \\ \gamma v & \gamma \end{pmatrix}$$

And the canonical example of that is:

$$\begin{pmatrix} \gamma & -\gamma v \\ -\gamma v & \gamma \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \gamma \\ -\gamma v \end{pmatrix}$$

where the vectors are four velocities (recall that $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ is the four-velocity of an object at rest). What this means is that I measure the four-velocity of an object to be $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and he measures the same object as having a four-velocity of $\begin{pmatrix} \gamma \\ -\gamma v \end{pmatrix}$. So I see the object at rest, and he sees it as moving at speed $-v$; that is, he sees it moving to the left (it moves to the left because he is moving to the right relative to me).

Null vectors: light

Let's try it with light. Recall that we describe a photon in terms of its four momentum, not four-velocity, which is undefined for a photon. Specifically, we *define* the four-momentum of a photon to be

$$\begin{pmatrix} E/c \\ Ev_x/c^2 \\ Ev_y/c^2 \\ Ev_z/c^2 \end{pmatrix}$$

So if we consider unit energy and keep $c = 1$ we get $p = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ in our one-dimensional world (for a rightward-moving photon) and the Lorentz transform is then

$$\begin{pmatrix} \gamma & -\gamma v \\ -\gamma v & \gamma \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \gamma - \gamma v \\ \gamma - \gamma v \end{pmatrix}$$

So I see a photon with unit energy, and he sees the photon with energy $\gamma(1-v) = \sqrt{1-v} < 1$: the photon has less energy in his frame than mine because of Doppler redshifting.

It's worth doing the same analysis with a leftward-moving photon:

$$\begin{pmatrix} \gamma & -\gamma v \\ -\gamma v & \gamma \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} \gamma(1+v) \\ -\gamma(1+v) \end{pmatrix}$$

Here the photon has more energy for him than me because of blue shifting: he is moving to the right and encounters a photon moving to the left.

The above analysis uses *passive* transforms: There is a single physical reality, and we are just describing that one physical reality using two different coordinate systems. One of the coordinate systems uses a set of axes that are *boosted* relative to the axes of the other.

This is why it makes sense to use prime notation as in $x \longrightarrow x'$ and $t \longrightarrow t'$ for a passive Lorentz transformation: the prime denotes measurements made using coordinates that are defined with respect to the boosted system, and we see things like

$$\begin{pmatrix} t' \\ x' \end{pmatrix} = \begin{pmatrix} \gamma & -\gamma v \\ -\gamma v & \gamma \end{pmatrix} \begin{pmatrix} t \\ x \end{pmatrix}$$

These are the first two elements of a displacement four-vector. It is the same four-vector but viewed in two different reference frames. This is the default mode in the package:

```
> sol(1)
> boost(as.3vel(c(0.6,0,0)))
```

```
      t     x y z
t  1.25 -0.75 0 0
x -0.75  1.25 0 0
y  0.00  0.00 1 0
z  0.00  0.00 0 1
```

(note that element $[1,2]$ is negative).

Active transforms

In the passive view, there is a single physical reality, and we are just describing that one physical reality using two different coordinate systems. Now we will consider active transforms: there are two physical realities, but one is boosted with respect to another.

Suppose me and my friend are at rest with respect to one another, but my friend is in a spaceship and I am outside it, in free space, at rest. He constructs a four-vector in his spaceship; for example, he could fire bullets out of a gun which is fixed in the spaceship, and then calculate their four-velocity. We both agree on this four-velocity as our reference frames are identical: we have no relative velocity.

Now his spaceship acquires a constant velocity, leaving me stationary. My friend continues to fire bullets out of his gun and sees that their four-velocity, as viewed in his spaceship coordinates, is the same as when we were together.

Now he wonders what the four-velocity of the bullets is in my reference frame. This is an *active* transform: we have two distinct physical realities, one in the spaceship when it was at rest with respect to me, and one in the spaceship when moving. And both these realities, by construction, look the same to my friend in the spaceship.

Suppose, for example, he sees the bullets at rest in his spaceship; they have a four-velocity of $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and my friend says to himself: "I see bullets with a four velocity of $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and I know what that means. The bullets are at rest. What are the bullets' four velocities in Robin's reference frame?". This is an *active* transform:

$$\begin{pmatrix} \gamma & \gamma v \\ \gamma v & \gamma \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \gamma \\ \gamma v \end{pmatrix}$$

(we again suppose that the spaceship moves at speed $v > 0$ from left to right). So he sees a four velocity of $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and I see $\begin{pmatrix} \gamma \\ \gamma v \end{pmatrix}$, that is, with a positive speed: the bullets move from left to right (with the spaceship).

Here we use SI units to show the asymmetry of the boost matrix. The R idiom would be:

```
> sol(299792458)
> (B <- boost(as.3vel(c(1000,0,0))))  # 1km/s left to right

      t            x y z
t     1 -1.11265e-14 0 0
x -1000  1.00000e+00 0 0
y     0  0.00000e+00 1 0
z     0  0.00000e+00 0 1

> solve(B) %*% c(1,0,0,0)             # active transform ~= Galilean

  [,1]
t    1
x 1000
y    0
z    0
```

5. Vectorization

Here I discuss vectorized operations. The issue is difficult because a vector of four-velocities is arranged so that each row is a four-velocity, and each column is a component. To avoid confusion between boost matrices and their transposes we will use $c = 10$.

```
> sol(10)
> options(digits=3)

> u <- 1:7  # speed in the x-direction [c=10]
> jj <- cbind(gam(u),gam(u)*u,0,0)
> (U <- as.4vel(jj))

        t    x y z
[1,] 1.01 1.01 0 0
[2,] 1.02 2.04 0 0
[3,] 1.05 3.14 0 0
[4,] 1.09 4.36 0 0
[5,] 1.15 5.77 0 0
[6,] 1.25 7.50 0 0
[7,] 1.40 9.80 0 0
```

Now a boost, also in the x-direction:

```
> (B <- boost(as.3vel(c(6,0,0))))  # 60% speed of light
```

```
      t      x y z
t  1.25 -0.075 0 0
x -7.50  1.250 0 0
y  0.00  0.000 1 0
z  0.00  0.000 0 1
```

Note the asymmetry of B, in this case reflecting the speed of light being 10 (but note that boost matrices are not always symmetrical, even if $c = 1$).

To effect a *passive* boost we need to multiply each row of U by the transpose of the boost matrix B:

```
> U %*% t(B)
```

```
        t     x y z
[1,] 1.18 -6.28 0 0
[2,] 1.12 -5.10 0 0
[3,] 1.07 -3.93 0 0
[4,] 1.04 -2.73 0 0
[5,] 1.01 -1.44 0 0
[6,] 1.00  0.00 0 0
[7,] 1.02  1.75 0 0
```

we can verify that the above is at least plausible:

```
> is.consistent.4vel(U %*% t(B))
```

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

the above shows that the four velocities U, as observed by an observer corresponding to boost B, satisfies $U^i U_i = -c^2$. Anyway, in this context we really ought to use `tcrossprod()`:

```
> tcrossprod(U,B)
```

```
        t     x y z
[1,] 1.18 -6.28 0 0
[2,] 1.12 -5.10 0 0
[3,] 1.07 -3.93 0 0
[4,] 1.04 -2.73 0 0
[5,] 1.01 -1.44 0 0
[6,] 1.00  0.00 0 0
[7,] 1.02  1.75 0 0
```

which would be preferable (because this idiom does not require one to take a transpose) although the speed increase is unlikely to matter much because B is only 4×4.

The above transforms were passive: we have some four-vectors measured in my rest frame, and we want to see what these are four-vectors as measured by my friend Baz, who is moving in the positive x direction at 60% of the speed of light (remember that $c = 10$). See how the x-component of the transformed four-velocity is negative, because in Baz's rest frame, the four velocities are pointing backwards.

To effect an *active* transform we need to take the matrix inverse of B:

```
> solve(B)
```

```
     t     x y z
t 1.25 0.075 0 0
x 7.50 1.250 0 0
y 0.00 0.000 1 0
z 0.00 0.000 0 1
```

and then

```
> tcrossprod(U,solve(B))
```

```
        t     x y z
[1,] 1.33  8.79 0 0
[2,] 1.43 10.21 0 0
[3,] 1.55 11.79 0 0
[4,] 1.69 13.64 0 0
[5,] 1.88 15.88 0 0
[6,] 2.13 18.75 0 0
[7,] 2.49 22.75 0 0
```

In the above, note how the positive x-component of the four-velocity is increased because we have actively boosted it. We had better check the result for consistency:

```
> is.consistent.4vel(tcrossprod(U,solve(B)))
```

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

5.1. Multiple boosts

If we are considering multiple boosts, it is important to put them in the correct order. First we will do some passive boosts.

```
> sol(100)
> B1 <- boost(r3vel(1)) %*% boost(r3vel(1))
> B2 <- boost(r3vel(1)) %*% boost(r3vel(1))
> (U <- r4vel(5))
```

```
        t      x     y       z
[1,] 1.07  -26.5  27.8    5.33
[2,] 1.22   39.1 -55.8   18.33
[3,] 2.32 -183.5  89.3  -45.01
[4,] 1.55 -110.6  11.1  -39.00
[5,] 2.92 -247.2 -64.3 -100.18
```

Successive boosts are effected by matrix multiplication; there are at least four natural R constructions:

```
> U %*% t(B1) %*% t(B2)

         t     x   y    z
[1,]  54.4 -3593 276 4077
[2,]  28.2 -1854 201 2108
[3,] 132.8 -8825 563 9904
[4,]  71.1 -4746 297 5284
[5,] 103.1 -6945 365 7608

> U %*% t(B2 %*% B1)    # note order of operations

         t     x   y    z
[1,]  54.4 -3593 276 4077
[2,]  28.2 -1854 201 2108
[3,] 132.8 -8825 563 9904
[4,]  71.1 -4746 297 5284
[5,] 103.1 -6945 365 7608

> tcrossprod(U, B2 %*% B1)

         t     x   y    z
[1,]  54.4 -3593 276 4077
[2,]  28.2 -1854 201 2108
[3,] 132.8 -8825 563 9904
[4,]  71.1 -4746 297 5284
[5,] 103.1 -6945 365 7608

> U %>% tcrossprod(B2 %*% B1)

         t     x   y    z
[1,]  54.4 -3593 276 4077
[2,]  28.2 -1854 201 2108
[3,] 132.8 -8825 563 9904
[4,]  71.1 -4746 297 5284
[5,] 103.1 -6945 365 7608
```

(in the above, note that the result is the same in each case).

5.2. Multiple boosts of the stress-energy tensor

Multiple boosts work quite nicely with the package:

```
> B1 <- boost(r3vel(1)) %*% boost(r3vel(1))
> B2 <- boost(r3vel(1)) %*% boost(r3vel(1))
> LHS <- transform_uu(transform_uu(dust(1),B1),B2)
> RHS <- transform_uu(dust(1),B2 %*% B1)   # note order
> LHS-RHS  # should be small

          t         x         y         z
t -1.42e-14 -1.82e-12  9.09e-13  4.55e-13
x  0.00e+00  2.33e-10 -2.91e-11 -5.82e-11
y  9.09e-13 -1.16e-10 -2.91e-11 -4.37e-11
z  4.55e-13 -5.82e-11  0.00e+00  0.00e+00
```

Again the magrittr package can be used for more readable idiom:

```
> perfectfluid(3,100) %>% transform_uu(B1) %>% transform_uu(B2)

       t       x       y       z
t    158  -11897    8003    6364
x -11897  893615 -601155 -478020
y   8003 -601155  404410  321575
z   6364 -478020  321575  255707

> perfectfluid(3,100) %>% transform_uu(B2 %*% B1)  # should match

       t       x       y       z
t    158  -11897    8003    6364
x -11897  893615 -601155 -478020
y   8003 -601155  404410  321575
z   6364 -478020  321575  255707
```

A warning

It is easy to misapply matrix multiplication in this context. Note carefully that the following natural idiom is **incorrect**:

```
> U %*% B  # Young Frankstein: Do Not Use This Brain!

        t      x     y       z
[1,]  200  -33.2  27.8    5.33
[2,] -291   48.7 -55.8   18.33
[3,] 1379 -229.5  89.3  -45.01
[4,]  832 -138.4  11.1  -39.00
[5,] 1858 -309.3 -64.3 -100.18
```

It is not clear to me that the idiom above has any meaning at all.

6. Photons

It is possible to define the four-momentum of photons by specifying their three-velocity and energy, and using `as.photon()`:

```
> sol(1)
> (A <- as.photon(as.3vel(cbind(0.9,1:5/40,5:1/40))))

     E   p_x    p_y    p_z
[1,] 1 0.990 0.0275 0.1375
[2,] 1 0.992 0.0551 0.1103
[3,] 1 0.993 0.0828 0.0828
[4,] 1 0.992 0.1103 0.0551
[5,] 1 0.990 0.1375 0.0275
```

above, A is a vector of four-momentum of five photons, all of unit energy, each with a null world line. They are all moving approximately parallel to the x-axis. We can check that this is indeed a null vector:

```
> inner4(A)

[1]  1.45e-16  2.56e-17 -5.55e-17  2.56e-17  1.45e-16
```

showing that the vectors are indeed null to numerical precision. What do these photons look like in a frame moving along the x-axis at $0.7c$?

```
> tcrossprod(A,boost(as.3vel(c(0.7,0,0))))

         t     x      y      z
[1,] 0.430 0.406 0.0275 0.1375
[2,] 0.428 0.409 0.0551 0.1103
[3,] 0.427 0.410 0.0828 0.0828
[4,] 0.428 0.409 0.1103 0.0551
[5,] 0.430 0.406 0.1375 0.0275
```

Above, see how the photons have lost the majority of their energy due to redshifting. Blue shifting is easy to implement as either a passive transform:

```
> tcrossprod(A,boost(as.3vel(c(-0.7,0,0))))

        t    x      y      z
[1,] 2.37 2.37 0.0275 0.1375
[2,] 2.37 2.37 0.0551 0.1103
[3,] 2.37 2.37 0.0828 0.0828
[4,] 2.37 2.37 0.1103 0.0551
[5,] 2.37 2.37 0.1375 0.0275
```

or an active transform:

```
> tcrossprod(A,solve(boost(as.3vel(c(0.7,0,0)))))
```

```
        t    x      y      z
[1,] 2.37 2.37 0.0275 0.1375
[2,] 2.37 2.37 0.0551 0.1103
[3,] 2.37 2.37 0.0828 0.0828
[4,] 2.37 2.37 0.1103 0.0551
[5,] 2.37 2.37 0.1375 0.0275
```

giving the same result.

6.1. Reflection in mirrors

Gjurchinovski (2004) discusses reflection of light from a uniformly moving mirror and here I show how the **lorentz** package can illustrate some of his insights.

We are going to reflect the photons in an oblique mirror which is itself moving at half the speed of light along the x-axis. The first step is to define the mirror `m`, and the boost `B` corresponding to its velocity:

```
> m <- c(1,1,1)
> B <- boost(as.3vel(c(0.5,0,0)))
```

Above, the three-vector m is parallel to the normal vector of the mirror and B shows the Lorentz boost needed to bring it to rest. We are going to reflect these photons in this mirror. The R idiom for the reflection is performed using a sequence of transformations. First, transform the photons' four-momentum to a frame in which the mirror is at rest:

```
> A
```

```
     E   p_x    p_y    p_z
[1,] 1 0.990 0.0275 0.1375
[2,] 1 0.992 0.0551 0.1103
[3,] 1 0.993 0.0828 0.0828
[4,] 1 0.992 0.1103 0.0551
[5,] 1 0.990 0.1375 0.0275
```

```
> (A <- as.4mom(A %*% t(B)))
```

```
          E   p_x    p_y    p_z
[1,] 0.583 0.566 0.0275 0.1375
[2,] 0.582 0.569 0.0551 0.1103
[3,] 0.581 0.569 0.0828 0.0828
[4,] 0.582 0.569 0.1103 0.0551
[5,] 0.583 0.566 0.1375 0.0275
```

Above, see how the photons have lost energy because of a redshift (the `as.4mom()` function has no effect other than changing the column names). Next, reflect the photons in the mirror (which is at rest):

```
> (A <- reflect(A,m))
```

```
         E    p_x    p_y    p_z
[1,] 0.583 0.0786 -0.460 -0.350
[2,] 0.582 0.0793 -0.434 -0.379
[3,] 0.581 0.0795 -0.407 -0.407
[4,] 0.582 0.0793 -0.379 -0.434
[5,] 0.583 0.0786 -0.350 -0.460
```

Above, see how the reflected photons have a reduced the x-component of momentum; but have acquired a substantial y- and z- component. Finally, we transform back to the original reference frame. Observe that this requires an *active* transform which means that we need to use the matrix inverse of B:

```
> (A <- as.4mom(A %*% solve(t(B))))
```

```
         E   p_x    p_y    p_z
[1,] 0.719 0.427 -0.460 -0.350
[2,] 0.718 0.427 -0.434 -0.379
[3,] 0.717 0.427 -0.407 -0.407
[4,] 0.718 0.427 -0.379 -0.434
[5,] 0.719 0.427 -0.350 -0.460
```

Thus in the original frame, the photons have lost about a quarter of their energy as a result of a Doppler effect: the mirror was receding from the source. The photons have imparted energy to the mirror as a result of mechanical work.

It is possible to carry out the same operations in one line:

```
> A <- as.photon(as.3vel(cbind(0.9,1:5/40,5:1/40)))
> A %>% tcrossprod(B) %>% reflect(m) %>% tcrossprod(solve(B)) %>% as.4mom()
```

```
         E   p_x    p_y    p_z
[1,] 0.719 0.427 -0.460 -0.350
[2,] 0.718 0.427 -0.434 -0.379
[3,] 0.717 0.427 -0.407 -0.407
[4,] 0.718 0.427 -0.379 -0.434
[5,] 0.719 0.427 -0.350 -0.460
```

Mirrors and rotation-boost coupling

Consider the following R idiom: we take a bunch of photons which, in a certain reference frame are all moving (almost) parallel to the x-axis. Then we reflect the photons from a mirror which is moving with a composition of pure boosts, and examine the reflected light in their original reference frame:

```
> sol(1)
> light_start <- as.photon(as.3vel(cbind(0.9,1:5/40,5:1/40)))
> m <- c(1,0,0)     # mirror normal to x-axis
> B1 <- boost(as.3vel(c(-0.5, 0.1, 0.0)))
> B2 <- boost(as.3vel(c( 0.2, 0.0, 0.0)))
> B3 <- boost(as.3vel(c( 0.0, 0.0, 0.6)))
> B <- B1 %*% B2 %*% B3   # matrix multiplication is associative!
> light <- light_start %*% t(B)
> light <- reflect(light,m)
> light <- as.4mom(light %*% solve(t(B)))
> light

        E   p_x   p_y   p_z
[1,] 2.30 -2.11 0.119 0.920
[2,] 2.31 -2.13 0.147 0.897
[3,] 2.32 -2.14 0.175 0.874
[4,] 2.32 -2.15 0.203 0.849
[5,] 2.33 -2.16 0.230 0.824
```

See how the photons have picked up momentum in the y- and z- direction, even though the mirror is oriented perpendicular to the x-axis (in its own frame). Again it is arguably preferable to use pipes:

```
> light_start %>% tcrossprod(B) %>% reflect(m) %>% tcrossprod(solve(B)) %>% as.4mom()

        E   p_x   p_y   p_z
[1,] 2.30 -2.11 0.119 0.920
[2,] 2.31 -2.13 0.147 0.897
[3,] 2.32 -2.14 0.175 0.874
[4,] 2.32 -2.15 0.203 0.849
[5,] 2.33 -2.16 0.230 0.824
```

Compare when the speed of light is infinite:

```
> sol(Inf)
> light_start <- as.photon(as.3vel(cbind(0.9,1:5/40,5:1/40)))
> B1 <- boost(as.3vel(c(-0.5, 0.1, 0.0)))
> B2 <- boost(as.3vel(c( 0.2, 0.0, 0.0)))
> B3 <- boost(as.3vel(c( 0.0, 0.0, 0.6)))
> B <- B1 %*% B2 %*% B3
> light_start

     E   p_x    p_y    p_z
[1,] 0 0.990 0.0275 0.1375
[2,] 0 0.992 0.0551 0.1103
[3,] 0 0.993 0.0828 0.0828
[4,] 0 0.992 0.1103 0.0551
[5,] 0 0.990 0.1375 0.0275
```

```
> light_start %>% tcrossprod(B) %>% reflect(m) %>% tcrossprod(solve(B)) %>% as.4mom()
```

```
     E    p_x    p_y    p_z
[1,] 0 -0.990 0.0275 0.1375
[2,] 0 -0.992 0.0551 0.1103
[3,] 0 -0.993 0.0828 0.0828
[4,] 0 -0.992 0.1103 0.0551
[5,] 0 -0.990 0.1375 0.0275
```

Note that, in the infinite light speed case, the energy of the photons is zero (photons have zero rest mass); further observe that in this classical case, the effect of the mirror is to multiply the x-momentum by -1 and leave the ither components unchanged, as one might expect from a mirror perpendicular to $(1, 0, 0)$.

7. Conclusions

The **lorentz** package furnishes some functionality for manipulating four-vectors and three-velocities in the context of special relativity. The R idiom is relatively natural and the package has been used to illustrate different features of relativistic kinematics.

References

Gjurchinovski A (2004). "Reflection of light from a uniformly moving mirror." *American Journal of Physics*, **72**(10), 1316–1324.

Goldstein H (1980). *Classical mechanics*. Second edition. Addison-Wesley.

Ungar AA (2006). "Thomas precession: a kinematic effect of the algebra of Einstein's velocity addition law. Comments on 'Deriving relativistic momentum and energy: II. Three-dimensional case'." *European Journal of Physics*, **27**, L17–L20.

Affiliation:

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