Mata’s built-in function list contains many useful matrix operations, but I recently came upon one that was lacking: the ability to flip a matrix along its rows or columns. Either of those operations can readily be done as a Mata statement, but I’d rather not remember the syntax—or have to remember what it is meant to do when I reread the code. So I wrote these two simple functions:

```mata
matrix function flipud(matrix X) {
    return(rows(X)>1 ? X[rows(X)..1,.] : X)
}

matrix function fliplr(matrix X) {
    return(cols(X)>1 ? X[.,cols(X)..1] : X)
}
end
```

These functions will flip a matrix ud—upside down (the first row becomes the last)—or lr, left to right (the first column becomes the last). Because the functions take a matrix argument, they may be applied to any of Mata’s matrix types, including string matrices.

Users have asked why one would want to flip a matrix “upside down”. As it happens, doing so becomes a handy tool when creating a two-sided linear filter. Say that we have defined a vector \( x \), containing a declining set of weights: a one-sided linear filter. We can turn \( x \) into a two-sided set of weights by using `flipud()`:

```
mata:
    matrix x = (1\0.5\0.25\0.125\0.0625)
    flipud(x)
```

I thank Mata’s principal architect, William Gould, for improvements he suggested that make the code more general.

1. I thank Mata’s principal architect, William Gould, for improvements he suggested that make the code more general.
To decipher that statement, note that 2..rows(x) refers to the second through last rows of vector x. The statement thus flips those rows of x upside down and concatenates them to the original x by using the column-join operator (see [M-2] op_join).

As a second example, consider applying both functions to a string matrix:

```
mata:
: Mata (type end to exit)
: Greek2me = ("alpha","beta","gamma","delta","epsilon","zeta","eta","theta",
> "iota","kappa","lambda","mu","nu","xi","omicron","pi",
> "rho","sigma","tau","upsilon","phi","chi","psi","omega")
: Greek2me
:   1  2  3
:   a b g
d e z
:   e t i
:   k l m
:   n x o
:   p r s
:   t u p
:   c p o
: lastFirst = fliplr(flipud(Greek2me)); lastFirst
:   1  2  3
:   o p c
:   p u t
:   s r p
:   x o m
:   m a k
:   i t e
:   z e d
:   g b a
: end
```