Decomposition Visualization Function

```julia
using GCPDecompositions, CairoMakie, LinearAlgebra
```

The Function

```
plot_factors (generic function with 1 method)
```

This function automatically creates a generic plot for tensor decomposition. It only needs the tensor, its decomposition, and the plot types for each mode. There are optional arguments for more customization (more will be added).

Usage and Examples

```
 0.464386   0.529142    0.980621  0.59815       0.753148    0.256854  0.647765
 0.227505   0.73233     0.793704  0.458057      0.890274    0.71319   0.015474
 0.555037   0.00164243  0.29403   0.95469       0.00147592  0.917474  0.100486
 0.493676   0.976642    0.168977  0.894165   …  0.993621    0.813131  0.406193
 0.405733   0.618328    0.432084  0.465658      0.816485    0.42063   0.729605

[:, :, 3] =
 0.604603    0.78821     0.110808  …  0.717584  0.37419   0.312431   0.185377
 0.316953    0.516362    0.497558     0.957066  0.549795  0.0601777  0.97043
 0.218988    0.929636    0.738119     0.120431  0.776774  0.535856   0.18237
 0.540931    0.865974    0.201563     0.639513  0.65234   0.819245   0.512026
 0.381727    0.561393    0.380827     0.795431  0.842602  0.602485   0.697681
 0.00259335  0.668311    0.335275  …  0.245741  0.355975  0.296618   0.988647
 0.7846      0.106639    0.662216     0.714821  0.768067  0.620135   0.545162
 ⋮                                 ⋱  ⋮
 0.889191    0.00173448  0.12136      0.419409  0.384872  0.611267   0.904067
 0.677227    0.667241    0.259911     0.755645  0.64853   0.524946   0.329733
 0.141588    0.0377558   0.235581     0.160012  0.970574  0.25537    0.760696
 0.135245    0.281683    0.803019     0.918332  0.726266  0.0484353  0.311037
 0.707619    0.725648    0.787052  …  0.04817   0.598222  0.960163   0.656111
 0.0258461   0.787323    0.595707     0.726022  0.541672  0.109713   0.306948

;;; …

[:, :, 89] =
 0.200488   0.556973   0.700112   0.150859   …  0.675673  0.276741  0.318923
 0.912615   0.794244   0.889633   0.830709      0.522989  0.825271  0.0955385
 0.596955   0.84176    0.365061   0.247461      0.356885  0.160212  0.288124
 0.264001   0.108471   0.941702   0.444781      0.31786   0.688232  0.725277
 0.0428483  0.366309   0.66466    0.435939      0.872365  0.824127  0.235768
 0.638927   0.145437   0.471803   0.59559    …  0.569891  0.512963  0.712752
 0.716507   0.0837831  0.270139   0.931122      0.108002  0.866799  0.902818
 ⋮                                           ⋱
 0.360205   0.374271   0.0462626  0.0355199     0.55997   0.126099  0.618824
```

```julia
X = rand(Float64,77,44,91) # random tensor
```

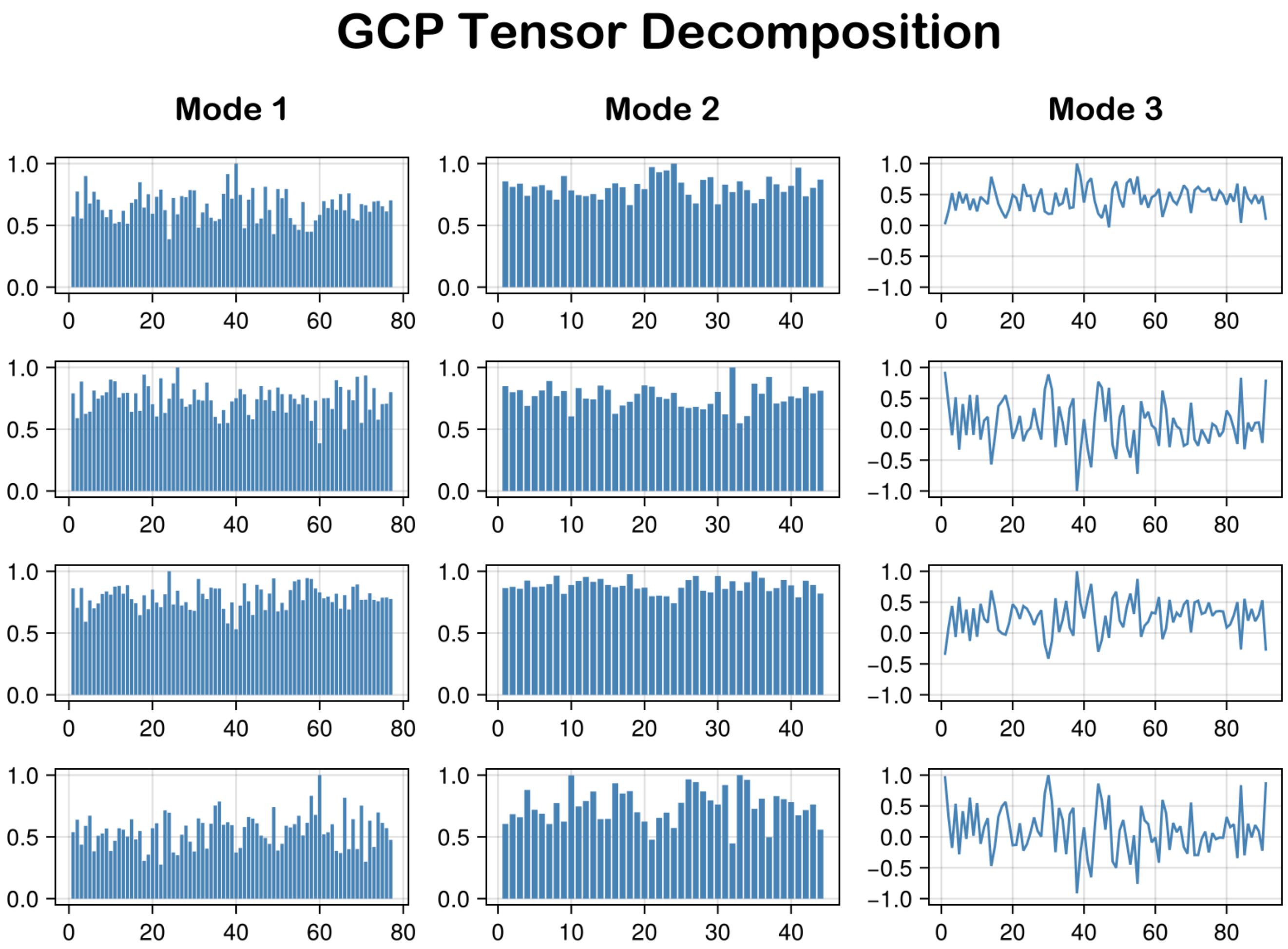
```
M = 77×44×91 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 4 components
    λ weights:
    4-element Vector{Float64}: …
    U[1] factor matrix:
    77×4 Matrix{Float64}: …
    U[2] factor matrix:
    44×4 Matrix{Float64}: …
    U[3] factor matrix:
    91×4 Matrix{Float64}: …
```

```julia
M = gcp(X,4) # rank 6 decomposition
```

```
N = 77×44×91 CPD{Float64, 3, Vector{Float64}, Matrix{Float64}} with 7 components
    λ weights:
    7-element Vector{Float64}: …
    U[1] factor matrix:
    77×7 Matrix{Float64}: …
    U[2] factor matrix:
    44×7 Matrix{Float64}: …
    U[3] factor matrix:
    91×7 Matrix{Float64}: …
```

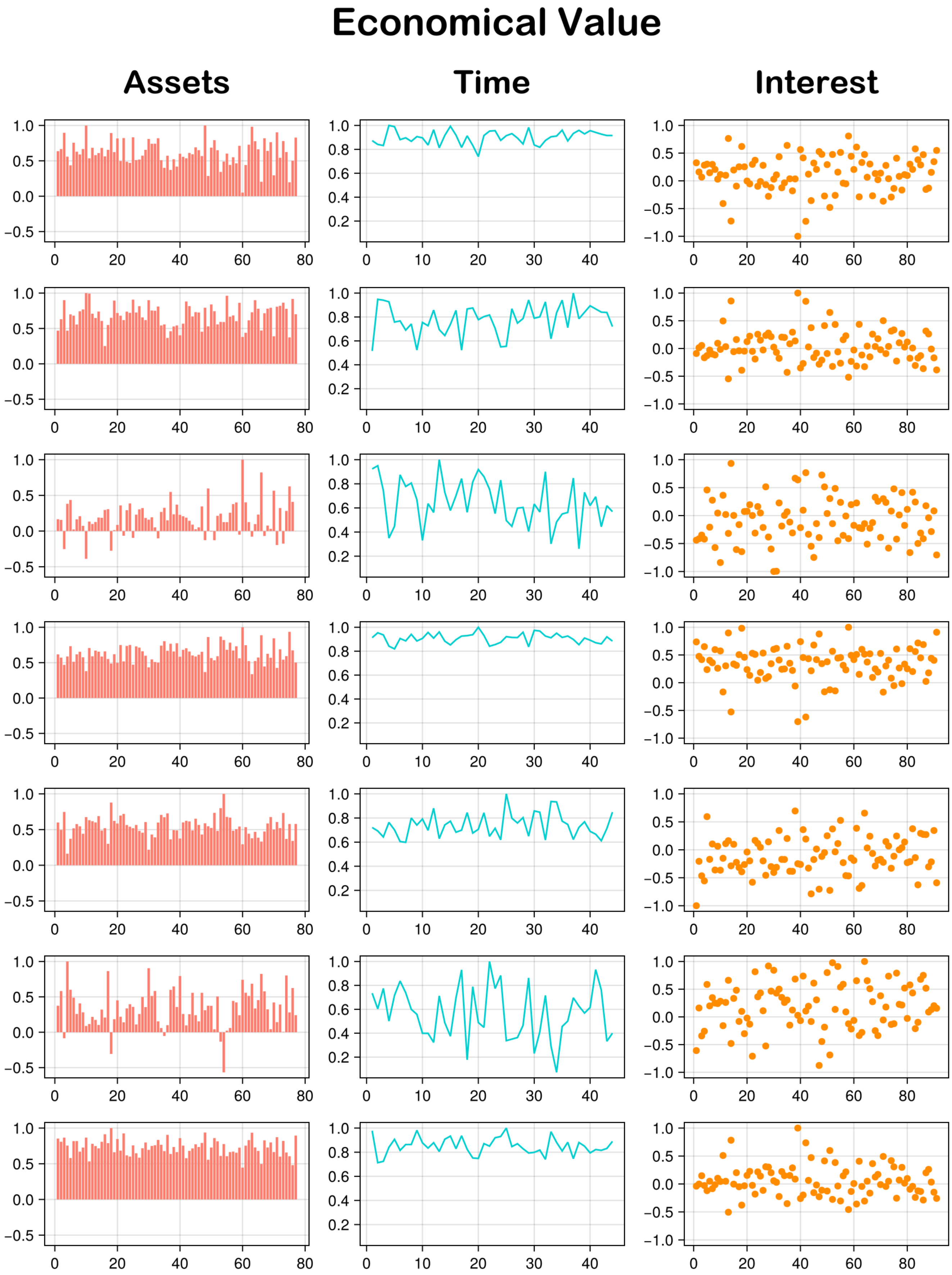
```julia
N = gcp(X,7)
```

Generic Plot



```
plot_factors(M,X,[:barplot,:barplot,:lines])
```

Custom Plot



Economical Value

Assets

Time

Interest

```julia
plot_factors(N,X,[:barplot,:lines,:scatter];graphsize = (900,1200),titlesize = 35, labelsize = 30, colors = [:salmon,:darkturquoise,:darkorange], title = "Economical Value", factor_names = ["Assets","Time","Interest"])
```

Functionality

Time efficiency! This function can either serve as a generic plot to quickly see how the decomposition/data looks, or it can be customized to create a visually appealing graph that can tell a story with the data/highlight essential parts.