DMPAF.m file

New functions:

- kaiserNormalization - scales the factor loadings by their associated eigenvalues to standardize the factors
  - this is typically applied before factor loading rotation because the rotations work best when the loadings are scaled to keep variances balanced across factors
  - makes the resulting factors easier to interpret because it reduces the chance of one factor overwhelming the others, which leads to a clearer picture of underlying structure
- varimaxRotation (gamma = 1, focus strictly on orthogonality):
  - correlation: NO (orthogonal)
  - designed to create completely uncorrelated factors
  - maximizes the variance of squared loadings within each factor while minimizing the loadings across factors
  - minimizes the number of variables that have high loadings on each factor
- quartimaxRotation (gamma = 0, focus on reduce factors rather than maintain orthogonal):
  - correlation: can lead to uncorrelated factors (orthogonal) but its focus is more on reducing the number of significant factors rather than maintaining orthogonality. It could still allow for correlated factors as it focuses on factor loadings
  - simplifies the loadings by maximizing the variance of the squared loadings across factors, but can lead to one factor dominating the others
  - minimizes the number of factors needed to explain each variable
- promaxRotation (gamma > 1):
  - correlation: allows for correlation among factors (oblique)
  - first applies an orthogonal rotation (varimax) and then obliquely transforms the factors to allow for correlation
  - transfors by raising the factor loadings to a power of gamma, usually between 1-4, but default is set to 4 to achieve the oblique rotation
  - this rotation is calculated quicker than the direct oblimin method (not using this in ours because it is the same as promax just slower)

- equamaxRotation (gamma = 0.5):
  - correlation: allows for correlation among factors (oblique) but does not emphasize as much as promax does given the 0.5 gamma
  - a combination of varimax and quartimax, aiming to balance the simplicity of factors (varimax) and the variance of loadings across both rows and columns (simplicity of variables) (quartimax)
  - the number of variables that load highly on a factor and the number of factors needed to explain a variable are minimized
- orthomaxRotation (0 < gamma < 1):
  - correlation: customizable (gamma = 0 <— quartimax) (gamma = 1 <— varimax)
  - optimizes factor loadings for interpretability with an adjustable gamma for controlling the level of obliqueness

new %% params:

- kaiserNormalizeLoadings = true;
  - true or false (do you want to use kaiser normalization for your factor loadings?)
- rotationType = ‘varimax’;
  - set your rotation type as a string
  - valid: ‘’, ‘varimax’, ‘quartimax’, ‘promax’, ‘equamax’, ‘orthomax’ (‘’ = no rotation)
- orthoGamma = 0.75;
  - set your gamma value or orthomax rotation (0 < orthoGamma < 1)
  - coefficient that controls the level of correlation target between factors
  - 1 = varimax (focus on orthogonality)
  - 0 = quartimax (reduce the number of significant factors rather than focusing on strict orthogonality)
- builtInNormalizeLoadings = false;
  - true or false (do you want to use the built-in rotation function’s rotation method?)
- visualizeBeforeAfterRotation = ‘both’;
  - valid: ‘’, ‘before’, ‘after’, ‘both’ (‘’ = no factor loading visualizations)
  - do you want to display your visualizations (heatmap and bar chart) of your loadings

    - do you want to display your visualizations (heatmap and bar chart) of your factor loadings before, after, or before and after performing your rotations of your loadings? (good for comparison of before rotating and after rotating)
- numVariablesToShow = 15;
    - how many variables do you want to show in your visualizations?
    - the function sorts the absolute value of factor loadings to determine the most significant
    - a higher magnitude in the heatmap = variable has greater positive relationship with that factor

new code implemented:

- while loop to iteratively calculate communalities
    - added disp([‘Convergence reached after ‘ num2str(iter) ‘ iterations’]);
    - in the if commDiff < convergenceThreshold then break statement
- kaiser normalization
    - if true, perform kaiser normalization
- before rotation visualizations
    - if ‘before’ or ‘both’, display heatmap and bar chart prior to rotation
- factor loadings rotation
    - in each function there are specific parameters unique to each rotation
    - all have ‘Normalize’ - flag indicated whether the loadings matrix should be row-normalized for rotation or no. If ‘on’ (default), rows of matrix are normalized prior to rotation to have unit Euclidean norm and unnormalized right after the rotation occurs. If ‘off’, the raw loadings are rotated and returned
    - if rotationType == ‘…’, call the rotation function
    - else, return error
- after rotation visualizations
    - if ‘after’ or ‘both’, display heatmap and bar chart post rotation