# Introduction to mobilityIndexR

This vignette provides a guide for using the `mobilityIndexR` package. `mobilityIndexR` allows users to both calculate transition matrices (relative, mixed, and absolute) as well as calculate and compare mobility indices (*Prais-Bibby*, *Average Movement*, *Weighted Group Mobility*, and *Origin Specific*). Below are examples and brief explanations of each.

# Transition Matrices

Two dimensional transition matrices are a tool for representing mobility within a group between two points in time with respect to some observable value such as income of workers or grades of students. Generally, this approach in some way ranks the observable value at each point in time and displays a contingency table with aggregated counts or proportions of each combination of ranks. `mobilityIndexR` offers three methods of constructing ranks using the `getTMatrix` function: relative, mixed, and absolute.

As example data, we include three datasets: `mobilityIndexR::incomeMobility`, `mobilityIndexR::incomeZeroInfMobility`, and `mobilityIndexR::gradeMobility`. These datasets each contain 125 records, an "id" column, and observations of some value at ten points in time, denoted "t0", …, "t9". `mobilityIndexR::incomeMobility` simulates income data over ten years; `mobilityIndexR::incomeZeroInfMobility` simulates income data over ten years that has an inflated number of zeros; and `mobilityIndexR::gradeMobility` simulates student grade data over ten assignments.

```
head(incomeMobility)
#>     id cohort    t0       t1       t2       t3       t4       t5        t6
#> 1 1001   2003 39235 39132.26 38988.60 23614.84 23629.59 23981.55  24032.27
#> 2 1002   2001 42533 70862.75 69655.04 70187.41 42896.49 42434.42  42790.18
#> 3 1003   2002 81463 81950.41 49711.08 49542.85 49628.86 82498.21 133827.51
#> 4 1004   2004 34363 33788.56 33744.43 33556.58 33658.70 20477.59  19994.63
#> 5 1005   2000 96627 97232.96 96614.23 95153.38 57226.46 57846.22  57361.49
#> 6 1006   2000 81552 81219.28 80966.39 81258.77 80573.05 79874.92  79569.12
#>          t7        t8        t9       t10
#> 1  39364.19  64585.58  64018.87  63758.16
#> 2  71126.93  71197.74 117711.75 116259.49
#> 3 133013.99 133126.18  80077.67  80559.27
#> 4  19939.64  19850.30  19869.06  31870.81
#> 5  94513.92  95099.74  94393.98  93521.19
#> 6  79847.03  80492.59  80553.66  80703.09
```

## Relative

Types of transition matrices differ by how values are binned into ranks. With relative transition matrices, values in each of the two specified columns (`col_x`, `col_y`) are binned into ranks as quantiles. The number of quantiles is specified in the `num_ranks` argument. Note: if one value in the data occurs more often than the size of the quantile ranks, then `getTMatrix` will throw an error; see Exclude Value.

`getTMatrix` returns a list containing the transition matrix as well as the bounds used to bin the data into ranks.

```
getTMatrix(dat = incomeMobility, col_x = "t0", col_y = "t5",
           type = "relative", num_ranks = 5, probs = FALSE)
#> $tmatrix
```

```
#>
#>      1  2  3  4  5
#>   1 19  5  1  0  0
#>   2  6 10  6  3  0
#>   3  0  6  8  7  4
#>   4  0  1  3 15  6
#>   5  0  3  7  0 15
#>
#> $col_x_bounds
#>      0%     20%     40%     60%     80%    100%
#>   462.0 21543.4 42469.8 64061.6 77888.4 99557.0
#>
#> $col_y_bounds
#>          0%         20%         40%         60%         80%        100%
#>    340.2705  18204.9969  39710.3062  58494.6271  78178.6713 262909.3195
```

The above example produces a $5 \times 5$ contingency table. Setting the argument `probs = TRUE`, we obtain a transition matrix with unconditional probabilities rather than counts.

```
getTMatrix(dat = incomeMobility, col_x = "t0", col_y = "t5",
           type = "relative", num_ranks = 5, probs = TRUE)
#> $tmatrix
#>
#>         1     2     3     4     5
#>   1 0.152 0.040 0.008 0.000 0.000
#>   2 0.048 0.080 0.048 0.024 0.000
#>   3 0.000 0.048 0.064 0.056 0.032
#>   4 0.000 0.008 0.024 0.120 0.048
#>   5 0.000 0.024 0.056 0.000 0.120
#>
#> $col_x_bounds
#>      0%     20%     40%     60%     80%    100%
#>   462.0 21543.4 42469.8 64061.6 77888.4 99557.0
#>
#> $col_y_bounds
#>          0%         20%         40%         60%         80%        100%
#>    340.2705  18204.9969  39710.3062  58494.6271  78178.6713 262909.3195
```

## Mixed

With mixed transition matrices, values in `col_x` are first binned into ranks as quantiles, then the bounds for `col_x` are used to bin `col_y` into ranks. In the example below, observe that `col_x_bounds` and `col_y_bounds` are equal except for the minimum and maximum values.

```
getTMatrix(dat = incomeMobility, col_x = "t0", col_y = "t5",
           type = "mixed", num_ranks = 5, probs = FALSE)
#> $tmatrix
#>
#>      1  2  3  4  5
#>   1 21  3  1  0  0
#>   2  8 11  4  2  0
#>   3  1  6 13  1  4
#>   4  0  2  2 15  6
```

```
#>   5  1  2  7  0 15
#>
#> $col_x_bounds
#>      0%     20%     40%     60%     80%    100%
#>   461.0 21543.4 42469.8 64061.6 77888.4 99558.0
#>
#> $col_y_bounds
#>          0%         20%         40%         60%         80%        100%
#>    339.2705  21543.4000  42469.8000  64061.6000  77888.4000 262910.3195
```

## Absolute

With absolute transition matrices, values in `col_x` and `col_y` are binned into ranks using user-specified bounds with the `bounds` argument.

```
getTMatrix(dat = gradeMobility, col_x = "t0", col_y = "t5",
           type = "absolute", probs = FALSE, bounds = c(0, 0.6, 0.7, 0.8, 0.9, 1.0))
#> $tmatrix
#>
#>      1  2  3  4  5
#>   1 10  3  0  0  0
#>   2  6 13  8  0  0
#>   3  0 12 17  8  3
#>   4  0  3 13 11 13
#>   5  0  0  1  0  4
#>
#> $col_x_bounds
#> [1] 0.0 0.6 0.7 0.8 0.9 1.0
#>
#> $col_y_bounds
#> [1] 0.0 0.6 0.7 0.8 0.9 1.0
```

# Mobility Indices

Mobility indices are measures of mobility within a group between two points in time with respect to some observable value such as income of workers or grades of students. `mobilityIndexR` calculates indices using the `getMobilityIndices` function. By default, the `getMobilityIndices` returns all available indices. Note that `getMobilityIndices` can additionally return bootstrapped intervals for each of the indices; see Bootstrapped Intervals.

```
getMobilityIndices(dat = incomeMobility, col_x = "t0", col_y = "t5",
                   type = "relative", num_ranks = 5)
#> $average_movement
#> [1] 0.64
#>
#> $os_far_bottom
#> [1] 0.04
#>
#> $os_far_top
#> [1] 0.4
#>
#> $os_total_bottom
```

```
#> [1] 0.24
#>
#> $os_total_top
#> [1] 0.4
#>
#> $prais_bibby
#> [1] 0.464
#>
#> $wgm
#> [1] 0.58
```

Below is a brief description of the indices. Let $s(i)$ be the event that one is in rank $i$ at the first or starting point in time and $e(j)$ be the event that one is in rank $j$ at the second or ending point in time. Suppose $k$ is the largest rank in the data.

The *Prais-Bibby* index (`type = "prais_bibby"`) is the proportion of records that change ranks, i.e. $\Sigma_{i=1}^{k} Pr[s(i) \neq e(i)]$.

The *Average Mobility* index (`type = "average_movement"`) is the average number of ranks records move.

The *Weighted Group Mobility* index (`type = "wgm"`) is a weighted version of the proportion of records that change ranks, i.e. $k^{-1}\Sigma_{i=1}^{k} \frac{Pr[\neg e(i)|s(i)]}{Pr[\neg e(i)]}$.

There are four *Origin Specific* indices (`type = "origin_specific"`): top, bottom, top far, and bottom far. The top (bottom) index is the proportion of records that begin in the top (bottom) and end outside of the top (bottom). The far versions of these indices are the proportions of records that begin in the top (bottom) and end at least two ranks away from the top (bottom).

```
getMobilityIndices(dat = incomeMobility, col_x = "t0", col_y = "t5",
                   type = "relative", num_ranks = 5, indices = "wgm")
#> $wgm
#> [1] 0.58
```

A single index or subset of the indices can be returned by passing a string or vector of strings in the `indices` argument.

# Hypothesis Testing

The function `getHypothesisTest` compares mobility indices across datasets. The user specifies two datasets (`dat_A`, `dat_B`) as well as a pair of columns from each dataset as a vector (`cols_A`, `cols_B`). This function performs a non-parametric one-sided hypothesis test that the index value for `dat_A` is greater than the index value for `dat_B`. The parameter `bootstrap_iter` specifies the number of bootstrap iterations; the default value is `100`.

```
getHypothesisTest(dat_A = incomeMobility, dat_B = incomeMobility,
                  cols_A = c("t0", "t3"), cols_B = c("t5", "t8"),
                  type = "relative", num_ranks = 5, bootstrap_iter = 100)
#> $prais_bibby
#> [1] 0.38
#>
#> $average_movement
#> [1] 0.6
#>
#> $wgm
#> [1] 0.39
```

```
#>
#> $os_total_top
#> [1] 0.51
#>
#> $os_far_top
#> [1] 0.92
#>
#> $os_total_bottom
#> [1] 0.32
#>
#> $os_far_bottom
#> [1] 0.03
```

By default, all available indices are returned. A single index or subset of the indices can be returned by passing a string or vector of strings in the `indices` argument.

# Advanced Options

## Bootstrapped Intervals

The function `getMobilityIndices` produces non-parametric bootstrapped confidence intervals using the percentile method by setting `intervals = TRUE`. By default, `intervals = FALSE`. The parameter `interval_pct` specifies the size of the confidence interval in terms of the proportion of the data covered; the default value is `0.95`. The parameter `bootstrap_iter` specifies the number of bootstrap iterations; the default value is `100`.

```
getMobilityIndices(dat = incomeMobility, col_x = "t0", col_y = "t5",
                   type = "relative", num_ranks = 5, indices = "wgm",
                   intervals = TRUE, interval_pct = 0.95, bootstrap_iter = 100)
#> $wgm
#> [1] 0.58
#>
#> $wgm_intervals
#> $wgm_intervals$lower
#>      2.5%
#> 0.4250779
#>
#> $wgm_intervals$upper
#>     97.5%
#> 0.7296108
#>
#> $wgm_intervals$interval_bounds
#> [1] 0.025 0.975
```

## Exclude Value

For `type = "relative"` or `type = "mixed"`, if one value in the data occurs more often than the size of the ranks then an error will result. This error will occur with `getTMatrix`, `getMobilityIndices`, and `getHypothesisTest` since binning into quantile ranks is not well defined in this case. See the example below where $0$ makes up more than $20\%$ of the values in the “t0” column.

```
getTMatrix(dat = incomeZeroInfMobility, col_x = "t0", col_y = "t5",
           type = "relative", num_ranks = 5, probs = FALSE)
#> Error in checkTooManyRanks(df = df, col_in = col_in, num_ranks = num_ranks, : One of the values
         represents more than 0.2 of the data. Try using fewer ranks or setting strict to FALSE.
table(incomeZeroInfMobility$t0)
#>
#>     0  1701  3009  8528 12094 12900 13404 13636 20507 25257 25698 27887 28486
#>    61     1     1     1     1     1     1     1     1     1     1     1     1
#> 29316 29558 29961 30216 31460 32665 32789 35269 37638 38635 39021 40496 41443
#>     1     1     1     1     1     1     1     1     1     1     1     1     1
#> 45106 45500 47188 47776 51703 52084 55799 56134 59943 60689 61592 63729 64877
#>     1     1     1     1     1     1     1     1     1     1     1     1     1
#> 66356 67509 68428 68683 72093 72611 74418 76879 77174 78929 80121 80562 82322
#>     1     1     1     1     1     1     1     1     1     1     1     1     1
#> 85666 85932 90068 90668 93007 96565 97100 97143 97907 98537 98768 98771 99406
#>     1     1     1     1     1     1     1     1     1     1     1     1     1
```

One approach is to bin some elements of the problematic value in one rank and the others in another rank. This is done by setting the parameter `strict = FALSE`. By default, `strict = TRUE`. This setting adds a negligible amount of random noise to the problematic values to make binning the values into quantile ranks well defined.

```
getTMatrix(dat = incomeZeroInfMobility, col_x = "t0", col_y = "t5",
           type = "relative", num_ranks = 5, probs = FALSE, strict = FALSE)
#> $tmatrix
#>
#>      1  2  3  4  5
#>   1  6  8  7  4  0
#>   2  4  6 12  3  0
#>   3  6  8  4  7  0
#>   4  3  2  0 10 10
#>   5  6  1  2  1 15
#>
#> $col_x_bounds
#>      0%     20%     40%     60%     80%    100%
#>     0.0     0.0     0.0 29719.2 66586.6 99406.0
#>
#> $col_y_bounds
#>        0%       20%       40%       60%       80%      100%
#>      0.00   2899.76  14254.91  25601.36  44479.91 121396.47
```

A second approach is to exclude the problematic value from binning the values into quantile ranks then either give the problematic value its own rank, add the problematic value to an existing rank, or exclude the problematic value altogether. A problematic value can be specified in the `exclude_value` argument. Note: currently, `mobilityIndexR` only supports one exclude value.

If an exclude value is specified, the user must also specify how to handle the exclude value using the `rerank_exclude_value` parameter. Setting `rerank_exclude_value = "as_new_rank"` creates an new rank for the exclude value and modifies the other ranks to remain ordinally consistent. Setting `rerank_exclude_value = "as_existing_rank"` bins the exclude value into an existing rank. Setting `rerank_exclude_value = "exclude"` drops any row that has the exclude value from the calculation.

```
getTMatrix(dat = incomeZeroInfMobility, col_x = "t0", col_y = "t5",
           type = "relative", num_ranks = 5, probs = FALSE,
```

```
          exclude_value = 0, rerank_exclude_value = "as_new_rank")
#> $tmatrix
#>
#>      1  2  3  4  5  6
#>   1 12 14 17 13  5  0
#>   2  1  4  3  4  1  0
#>   3  2  0  0  0 11  0
#>   4  1  2  0  1  2  6
#>   5  4  0  0  1  1  7
#>   6  2  1  0  2  0  8
#>
#> $col_x_bounds
#> exclude_value            0%           20%           40%           60%
#>           0.0        1701.0       29461.2       45184.8       66060.2
#>           80%          100%
#>       83659.6       99406.0
#>
#> $col_y_bounds
#> exclude_value            0%           20%           40%           60%
#>          0.00       1480.00      10664.72      19406.75      28154.18
#>           80%          100%
#>      57525.96     121396.47

getTMatrix(dat = incomeZeroInfMobility, col_x = "t0", col_y = "t5",
          type = "relative", num_ranks = 5, probs = FALSE,
          exclude_value = 0, rerank_exclude_value = "as_existing_rank")
#> $tmatrix
#>
#>      1  2  3  4  5
#>   1 31 20 17  6  0
#>   2  2  0  0 11  0
#>   3  3  0  1  2  6
#>   4  4  0  1  1  7
#>   5  3  0  2  0  8
#>
#> $col_x_bounds
#> exclude_value            0%           20%           40%           60%
#>           0.0        1701.0       29461.2       45184.8       66060.2
#>           80%          100%
#>       83659.6       99406.0
#>
#> $col_y_bounds
#> exclude_value            0%           20%           40%           60%
#>          0.00       1480.00      10664.72      19406.75      28154.18
#>           80%          100%
#>      57525.96     121396.47

getTMatrix(dat = incomeZeroInfMobility, col_x = "t0", col_y = "t5",
          type = "relative", num_ranks = 5, probs = FALSE,
          exclude_value = 0, rerank_exclude_value = "exclude")
#> $tmatrix
#>
#>      1  2  3  4  5
#>   1  4  3  4  1  0
#>   2  0  0  0 11  0
```

```
#>   3  2  0  1  2  6
#>   4  0  0  1  1  7
#>   5  1  0  2  0  8
#>
#> $col_x_bounds
#> exclude_value            0%          20%          40%          60%
#>           0.0        1701.0       29461.2       45184.8       66060.2
#>           80%          100%
#>       83659.6       99406.0
#>
#> $col_y_bounds
#> exclude_value            0%          20%          40%          60%
#>          0.00       1480.00      10664.72      19406.75      28154.18
#>           80%          100%
#>      57525.96     121396.47
```