

{xmap}: Unified Tools for Data Harmonisation

Presented at UseR! 2024, 9 July 2024

Cynthia A. Huang 

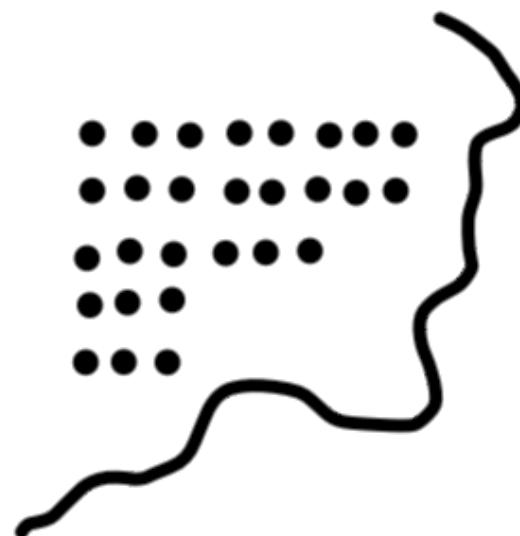
cynthia.huang@monash.edu

Department of Econometrics and Business Statistics

supervised by Rob J Hyndman, Sarah Goodwin and Simon Angus

Overview

Motivation



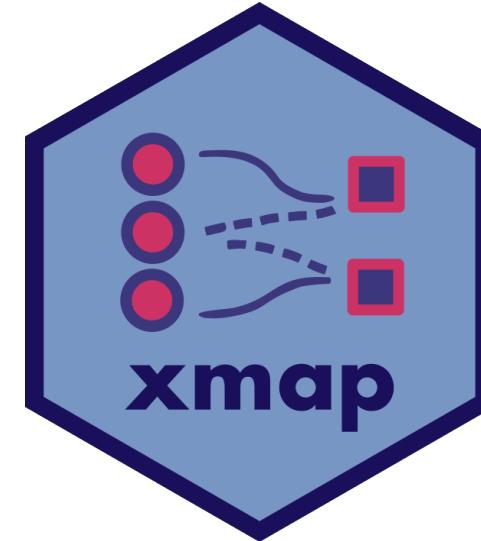
Official Statistics+

Conceptual Solution



Crossmaps
Framework

Implementation



[xmap](#) Toolkit and
Workflows

Motivation

Ex-Post Harmonisation of Survey Statistics

Harmonisation of Aggregate Statistics

Combining **semantically related** data collected under **distinct survey instruments** into a single analysis dataset.

COLLECT & CLEAN

AUS

NAME1	VALUE1
◊	5
○	10
□	50
△	45
○	20
140	

TRANSFORM

NAME2	VALUE2
K	15
T	75
L	40

MERGE

CTRY	NAME2	VALUE2
AUS	K	15
AUS	T	75
AUS	L	40

CTRY	NAME2	VALUE2
USA	K	50
USA	T	60
USA	L	30

USA

NAME2	VALUE2
K	50
T	60
L	30

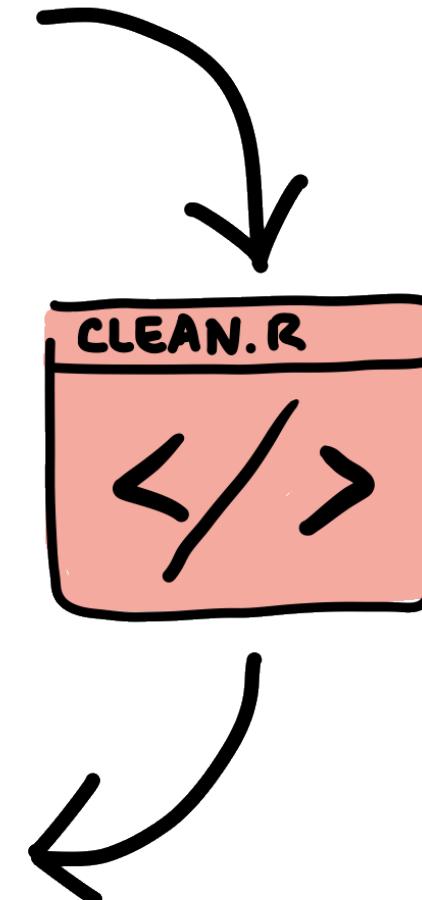
Current Workflow

AUS

NAME 1	VALUE 1
◇	5
○	10
□	50
△	45
○	20

AUS'

NAME 2	VALUE 2
K	15
T	75
L	40



BESPOKE

```
IF ◇. THEN K = ...  
ASSERT THAT  
  sum(VALUE 1) ==  
  sum(VALUE 2),  
  nrow(AUS) >  
  nrow(AUS')  
)
```

Current Workflow

```
schott_algorithm_28.do [800+ lines]
```

```
1 /*  
2  
3 HS-SIC-NAICS- Concordance Project  
4  
5 This program  
6  
7 1. reads in the hs-sic and hs-naics concordances from the monthly trade cd files and  
8      from Census and uses two mechanical matches to fill in naics matches prior to 2000  
9      matches after 2001.  
10 2. Although step 1 succeeds in generating many matches, there are some that remain unmatched
```

Conceptual Solution

Crossmaps Framework

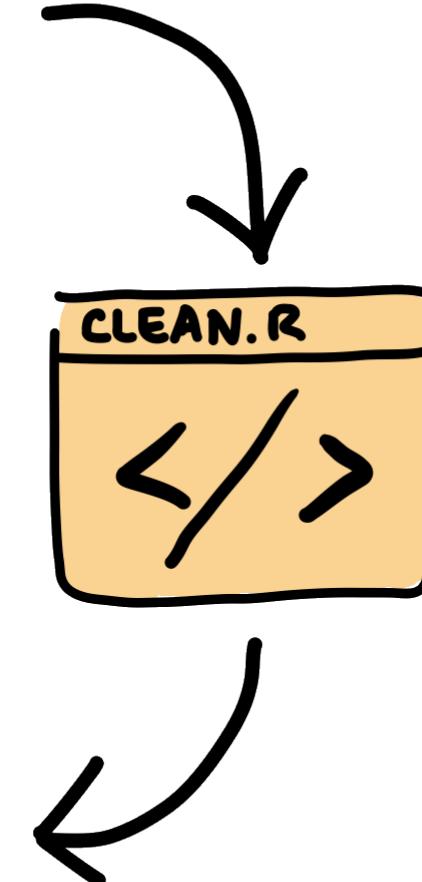
Proposed Solution

AUS

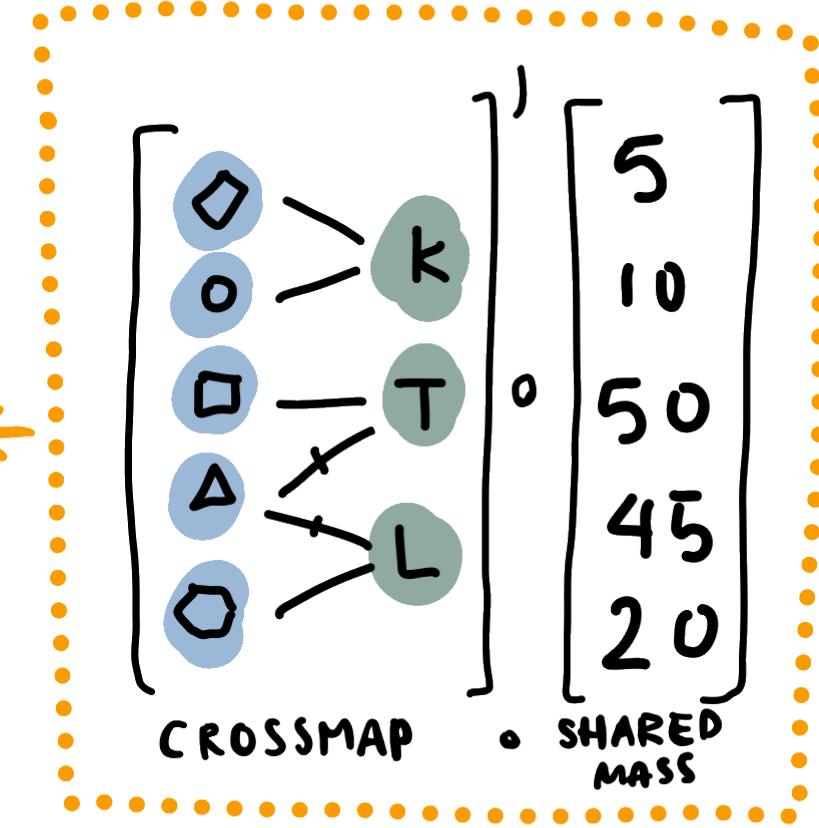
NAME 1	VALUE 1
□	5
○	10
□	50
△	45
○	20

AUS'

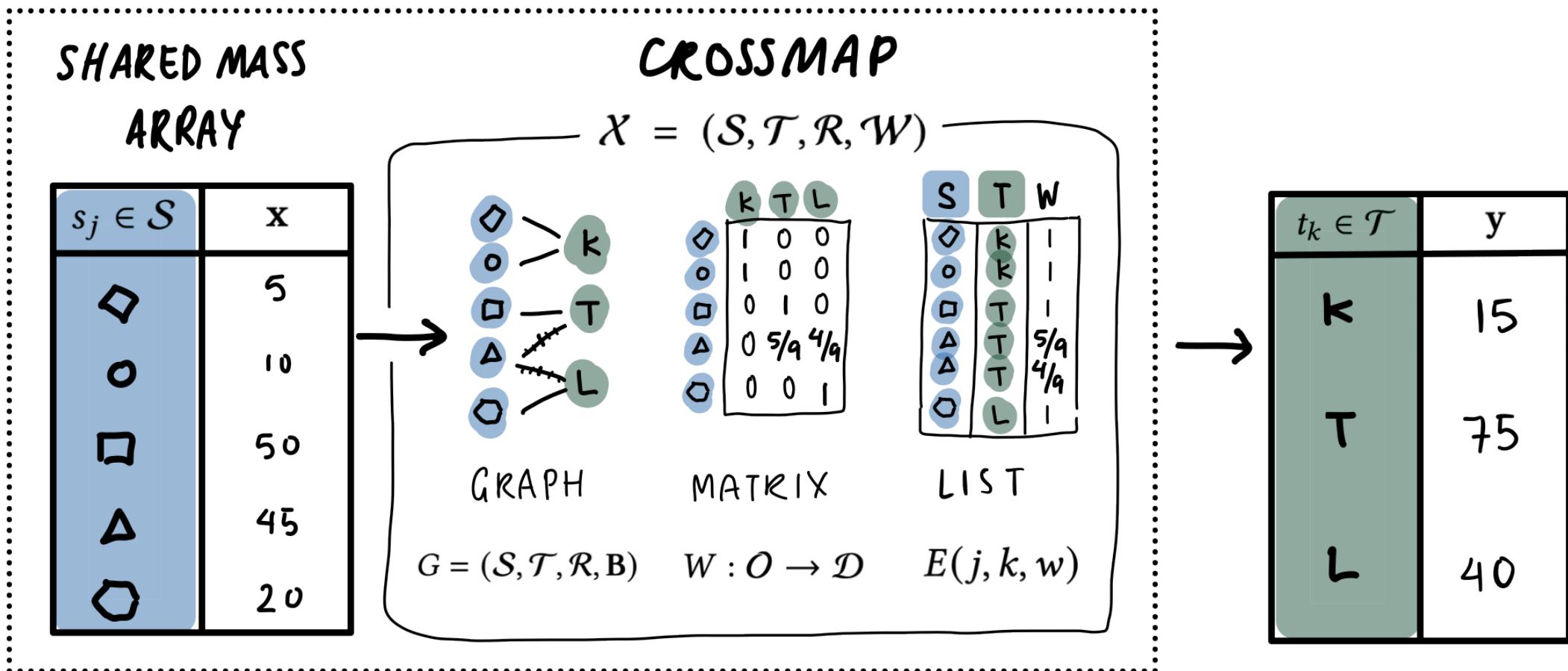
NAME 2	VALUE 2
K	15
T	75
L	40



FRAMEWORK



Crossmaps Framework



For details see preprint: [A Unified Statistical And Computational Framework For Ex-Post Harmonisation Of Aggregate Statistics <\[arxiv.org/abs/2406.14163\]\(https://arxiv.org/abs/2406.14163\)>](#)

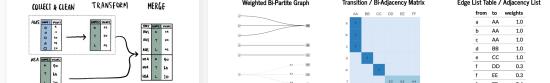
Unified and Principled Workflows

Using *Graphs, Matrices and Edge Lists* to investigate, illuminate and improve *Ex-Post Data Harmonisation*

Crossmaps: A principled approach to ex-post data harmonisation and dataset integration

Harmonising and merging data collected under different statistical classifications, taxonomies or nomenclatures is often required to analyse and compare social, political and economic phenomena across time or countries. Procedures used to achieve comparability are broadly known as **Ex-Post Harmonisation**, and include the transformation of data collected under a **source** taxonomy into harmonised data classified according to a **target** taxonomy. We refer to this sub-task as a **Cross-Taxonomy Transformation**, and encapsulate the transformation logic in a new information structure the **Crossmap**.

Cross-Taxonomy Transformation is an imputation *from source to target* data



Transformation logic can be validated via graph properties instead of ad-hoc assertions or line-by-line code review



Metrics based on crossmap properties can be used to *quantify and compare*:

- How robust are the results of imputation differ between crossmaps?
- How robust are downstream results to alternative harmonisation designs?
- How many observations are transformed on a given dataset with a given crossmap?
- Which observations in a harmonised dataset have undergone the most (or least) transformation?



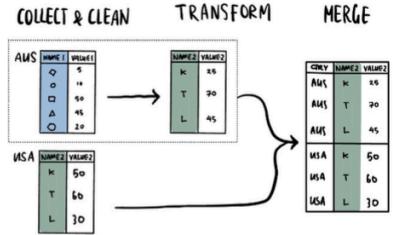
Cynthia A. Huang

Department of Econometrics and Business Statistics, Monash University

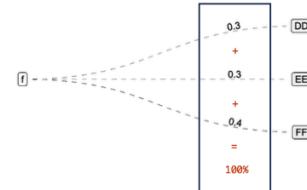
Supervised by Prof Rob Hyndman, Dr Seath Goodwin and Assoc. Prof Simon Angus



Cross-Taxonomy Transformation is an imputation *from source to target* data

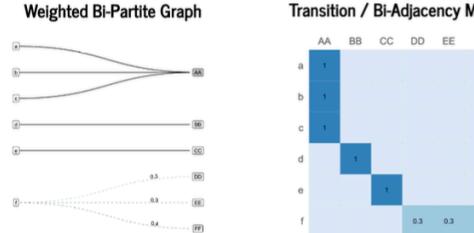


Transformation logic can be validated via **graph properties** instead of ad-hoc assertions or line-by-line code review



Condition for preserving numeric totals

Crossmaps is a unified framework for *specifying, validating, implementing, documenting and analysing cross-taxonomy transformations*



Data transformation can be implemented using validated crossmaps via *matrix multiplication* [1] performed as *database operations* on the edge list [2]

```
# A tibble: 10 x 3
  anzsc02  isc08 weights
  <chr>   <chr>    <dbl>
1 111111  1112  0.333
2 111111  1113  0.333
3 111111  1120  0.333
4 111211  1112  0.333
5 111211  1114  0.333
6 111211  1120  0.333
7 111212  0110  1
8 111212  1111  1
9 111312  1111  1
10 111399  1111  1

# A tibble: 5 x 5
  .x       .y       new_count
  <chr>   <chr>    <dbl>
1 0110    1112      40
2 1111    460
3 1112    500
4 1114    500
5 1120    500

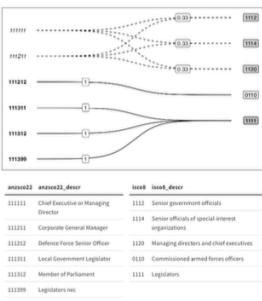
# A tibble: 6 x 2
  anzsc02 count
  <chr>   <dbl>
1 111111  1000
2 111211  500
3 111311  40
4 111312  300
5 111312  150
6 111399  10

apply_xmap(.data = anzsc02_stats,
            .xmap = anzsc0_xmap)

# A tibble: 5 x 5
  .x       .y       new_count
  <chr>   <chr>    <dbl>
1 0110    1112      40
2 1111    460
3 1112    500
4 1114    500
5 1120    500

# mock-up of apply_xmap() function
apply_xmap <- function(.data, .xmap) {
  left_join(
    x = .data,
    y = .xmap,
    by = c("isc08" = "isc02"))
  mutate(part_count = count * weights) |>
  group_by(isc08) |>
  summarise(new_count = sum(part_count))
}
```

Bi-graph visualisation and summary techniques can be used to *design data provenance documentation* [3]



For details see poster on cynthiahqy.com

Implementation in R

Data Structures and Functions for using the Crossmaps Approach

Software Overview

Core Features

Specify and Validate Mappings

- `{dplyr}` verbs
- `as_xmap_tbl()`
- `diagnose_as_xmap_tbl()`

Match and Apply Transformations

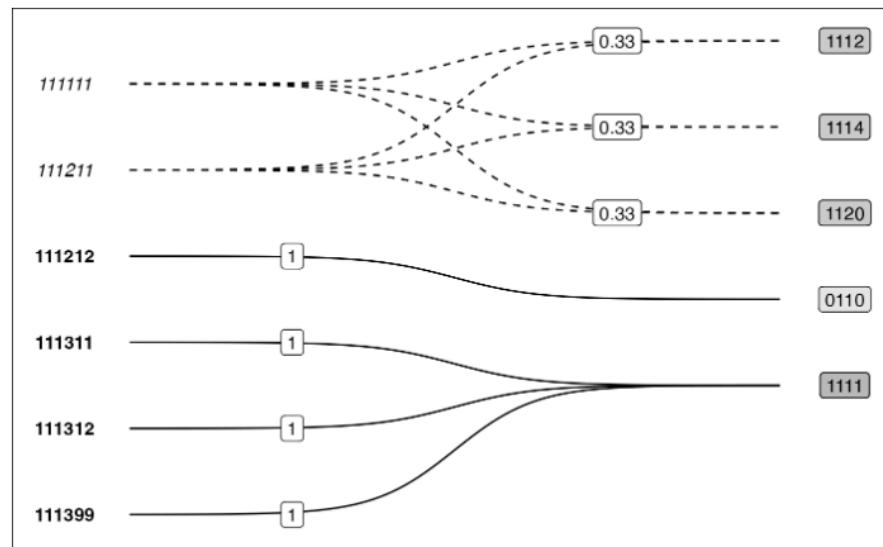
- `apply_xmap()`
- `diagnose_apply_xmap()`

Auxillary Features [WIP]

- extract logic from existing scripts
- graph and matrix classes
- symbolic fractional weights
- manage collections of crossmaps
- analysis of crossmap properties
- visualise and edit crossmap graphs
- ...

Example: Occupation Code Mappings

ANZSCO22 to ISCO8



anzsco22	anzsco22_descr	isco8	isco8_descr
111111	Chief Executive or Managing Director	1112	Senior government officials
111211	Corporate General Manager	1114	Senior officials of special-interest organizations
111212	Defence Force Senior Officer	1120	Managing directors and chief executives
111311	Local Government Legislator	0110	Commissioned armed forces officers
111312	Member of Parliament	1111	Legislators
111399	Legislators nec		

Stylised Occupation Level Counts

anzsco22	count
111111	1000
111211	500
111212	40
111311	300
111312	150
111399	10

Hypothetically aggregated from 2000 individual responses

Creating a Valid Crossmap

Start with *crosswalk*, or *lookup table*:

#	A tibble: 10 × 5	anzsco22	anzsco22_descr	isco8	partial	isco8_descr
		<chr>	<chr>	<chr>	<chr>	<chr>
1	111111 Chief Executive or Managing Director	1112	p	Senior governmen...		
2	111111 Chief Executive or Managing Director	1114	p	Senior officials...		
3	111111 Chief Executive or Managing Director	1120	p	Managing directo...		
4	111211 Corporate General Manager	1112	p	Senior governmen...		
5	111211 Corporate General Manager	1114	p	Senior officials...		
6	111211 Corporate General Manager	1120	p	Managing directo...		
7	111212 Defence Force Senior Officer	0110	p	Commissioned arm...		
8	111311 Local Government Legislator	1111	p	Legislators		
9	111312 Member of Parliament	1111	p	Legislators		
10	111399 Legislators nec	1111	p	Legislators		

Creating a Valid Crossmap

Add some naive (equal) distribution weights:

```
1 library(dplyr)
2 library(xmap)
3
4 crosswalk <-
5   xmap::demo$anzsco22_isco8_crosswalk |>
6   select(anzsco22, isco8)
7
8 (
9   links <- crosswalk |>
10  group_by(anzsco22) |>
11  mutate(equal = 1 / n_distinct(isco8)) |>
12  ungroup()
13 )
```

```
# A tibble: 10 × 3
  anzsco22 isco8 equal
  <chr>    <chr> <dbl>
1 111111   1112   0.333
2 111111   1114   0.333
3 111111   1120   0.333
4 111211   1112   0.333
5 111211   1114   0.333
6 111211   1120   0.333
7 111212   0110   1
8 111311   1111   1
9 111312   1111   1
10 111399  1111   1
```

Creating a Valid Crossmap

... and coerce to a crossmap:

```
1 (
2 occp_xmap <- links |>
3   xmap::as_xmap_tbl(
4     from = anzsc022,
5     to = isco8,
6     weight_by = equal
7   )
8 )
```

```
# A crossmap tibble: 10 × 3
# with unique keys: [6] anzsc022 -> [5] isco8
  .from$anzsc022 .to$isco8 .weight_by$equal
  <chr>          <chr>          <dbl>
1 111111         1112           0.333
2 111111         1114           0.333
3 111111         1120           0.333
4 111211         1112           0.333
5 111211         1114           0.333
6 111211         1120           0.333
7 111212         0110           1
8 111311         1111           1
9 111312         1111           1
10 111399        1111           1
```

! ERROR: Invalid Weights

What if we try to naively use unit weights?

```
1 crosswalk |>
2   mutate(ones = 1) |>
3   xmap::as_xmap_tbl(
4     from = anzSCO22,
5     to = isco8,
6     weight_by = ones
7   )
```

```
Error in `xmap_tbl()`:
! Invalid `weight_by` found for some links
✖ The total outgoing `weight_by` for some
`from` nodes are not near enough to
  1
ℹ Modify `weight_by` or adjust `tol` and try
again.
ℹ Use `diagnose_xmap_tbl()` for more
information.
```

```
1 crosswalk |>
2   mutate(ones = 1) |>
3   xmap::diagnose_as_xmap_tbl(
4     anzSCO22, isco8,
5     weight_by = ones
6   )
```

```
$bad_dups
NULL
```

```
$miss_weight_by
NULL
```

```
$bad_froms
# A tibble: 2 × 2
  .from$anzSCO22 .sum.weight_by
  <chr>                <dbl>
1 111111                  3
2 111211                  3
```

Applying a Valid Crossmap to Conformable Data

Stylised counts from before...

```
1 occp_stats <-
2   xmap::demo$anzsco22_stats |>
3   mutate(ref = 100)
4 )

# A tibble: 6 × 3
  anzsco22 count    ref
  <chr>     <dbl> <dbl>
1 111111     1000 100
2 111211      500 100
3 111212       40 100
4 111311      300 100
5 111312      150 100
6 111399       10 100
```

Transformed, with redistribution and aggregation:

```
1 occp_stats |>
2   xmap::apply_xmap(
3     .xmap = occp_xmap,
4     values_from = c(count, ref),
5     keys_from = anzsco22
6   )

# A tibble: 5 × 3
  isco8 count    ref
  <chr> <dbl> <dbl>
1 0110     40 100
2 1111     460 300
3 1112     500 66.7
4 1114     500 66.7
5 1120     500 66.7
```

Verifying Crossmaps and Transformations

Helper functions for keeping inside crossmap guardrails:

`diagnose_as_xmap_tbl()`:

- weights from a given `.from` key sum to one, to preserve total mass before and after
- no missing values in the links
- no duplicated links

`diagnose_apply_xmap()`:

- all key-value pairs in `.data` have matching instructions in `.xmap`
- no missing values in `.data`

Thanks for listening!

Try the package:

- Install from GitHub: [cynthiahqy/xmap](https://github.com/cynthiahqy/xmap)
- ...soon to be on CRAN

Ask me about related work and applications:

-  **UseR!, Salzburg (Jul 8-11)**
-  JSM, Portland (Aug 3-9)
-  posit::conf(2024), Seattle (Aug 12-14)
-  UBC, Vancouver, (Jul-Nov)
-  Monash University, Melbourne (Nov onwards)

Or online: [@cynthiahqy & \[cynthiahqy.com\]\(https://cynthiahqy.com\)](https://@cynthiahqy)

