balance

Balancing biased data samples with the *'balance'* Python package

ISA conference 2023-06-01

Presentors: Tal Galili & Tal Sarig

With: Roee Eilat, Daniel Haimovich, Steve Mandala

Central Applied Science

Meta

Outline of the talk

1. Market ecosystem
2. Survey Methodology context
    a. Total survey error
    b. Assumptions
    c. Estimation: Post-stratification
    d. Propensity Score and Estimation
3. Diagnostics tools
4. balance End-to-End workflow
5. Hands-on example

Acknowledgements / People

The *balance* package is actively maintained by people from the Central Applied Science team (in Tel Aviv and Boston), by Tal Sarig, Tal Galili and Steve Mandala.

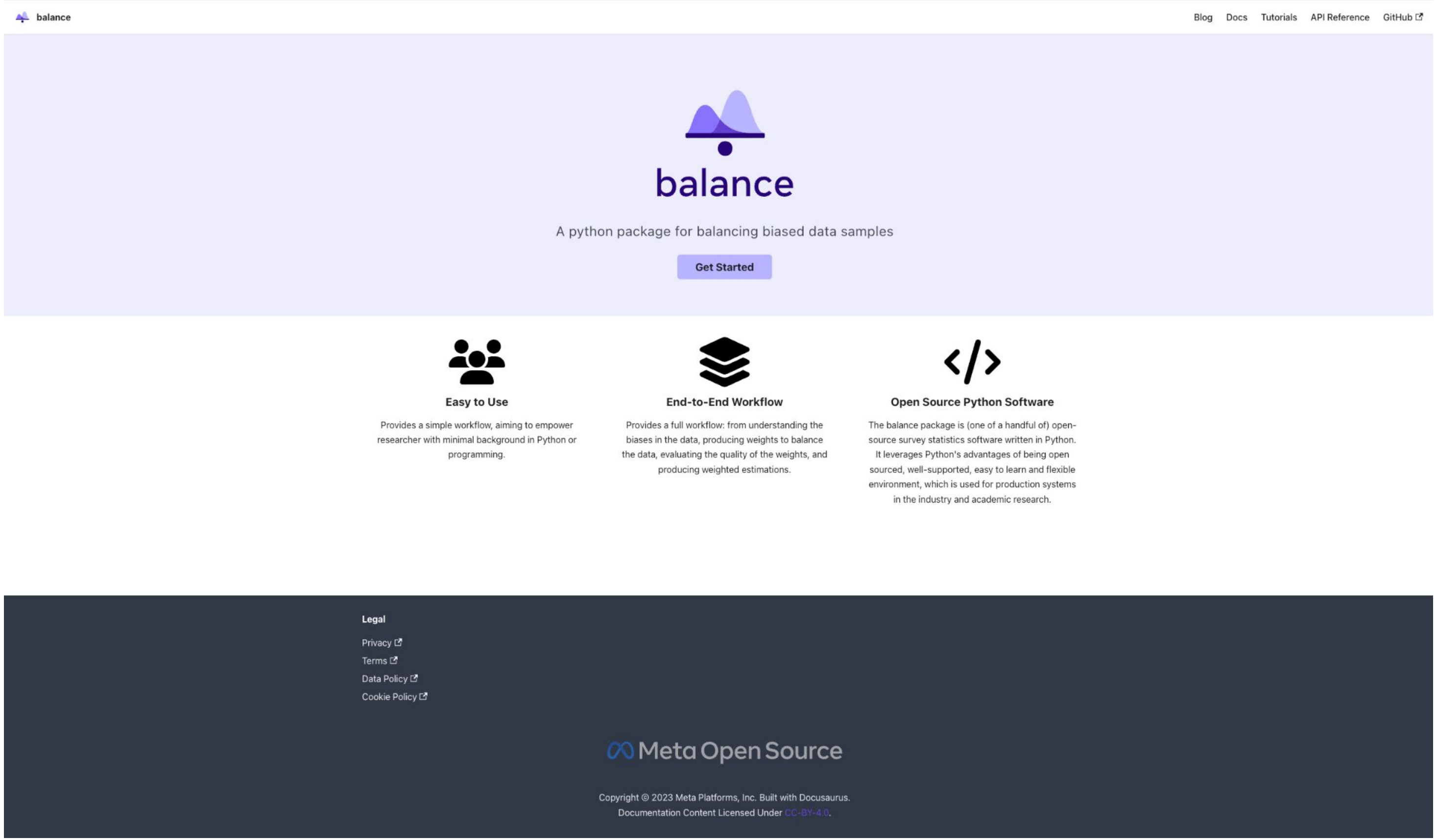
The *balance* package was (and is) developed by many people, including: Roee Eilat, Tal Galili, Daniel Haimovich, Kevin Liou, Steve Mandala, Adam Obeng (author of the initial internal Meta version), Tal Sarig, Luke Sonnet, Sean Taylor, Barak Yair Reif, and others. If you worked on balance in the past, please email us to be added to this list.

The *balance* package was open-sourced by Tal Sarig, Tal Galili and Steve Mandala in late 2022.

Branding created by Dana Beaty, from the Meta AI Design and Marketing Team. For logo files, see here.

Released to github on Nov 2022 (>600 stars)

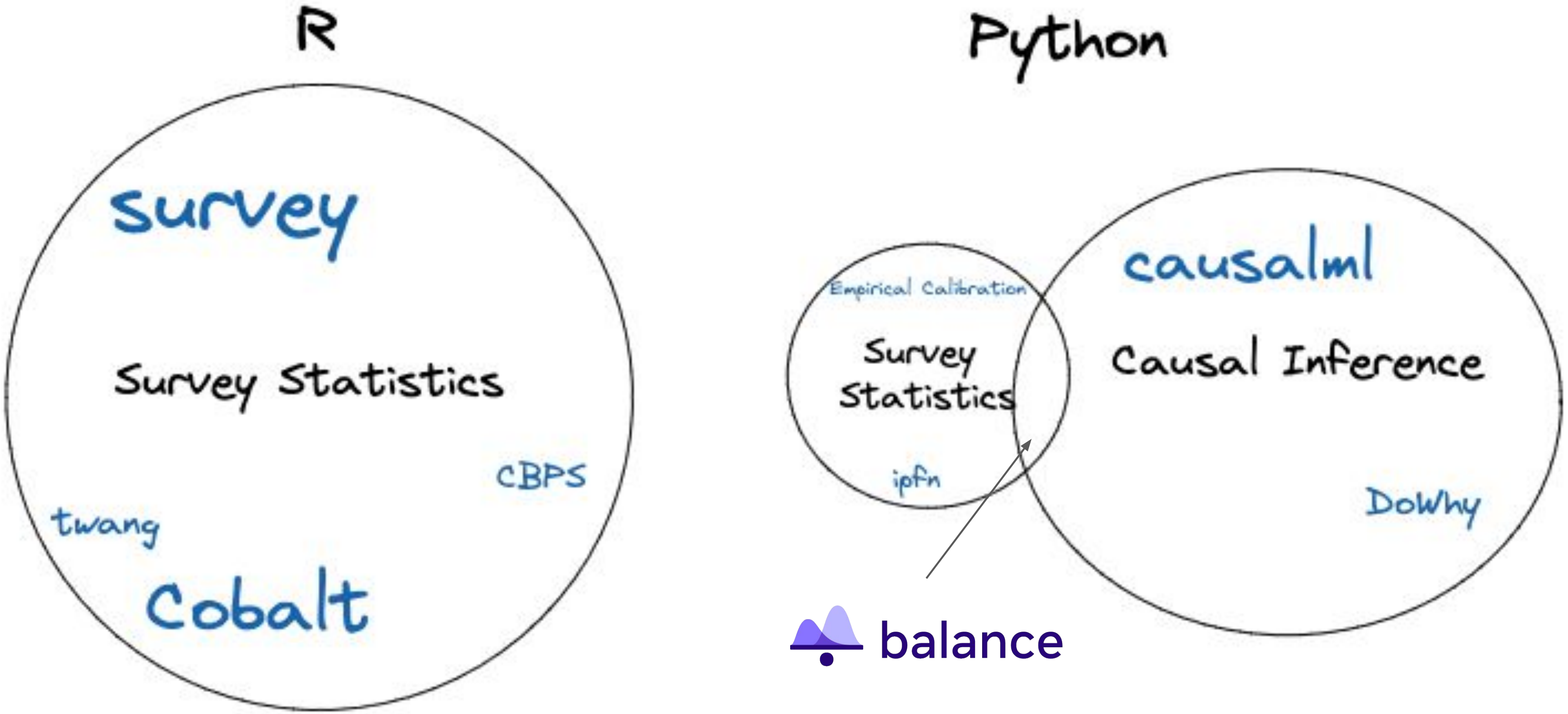
https://import-balance.org



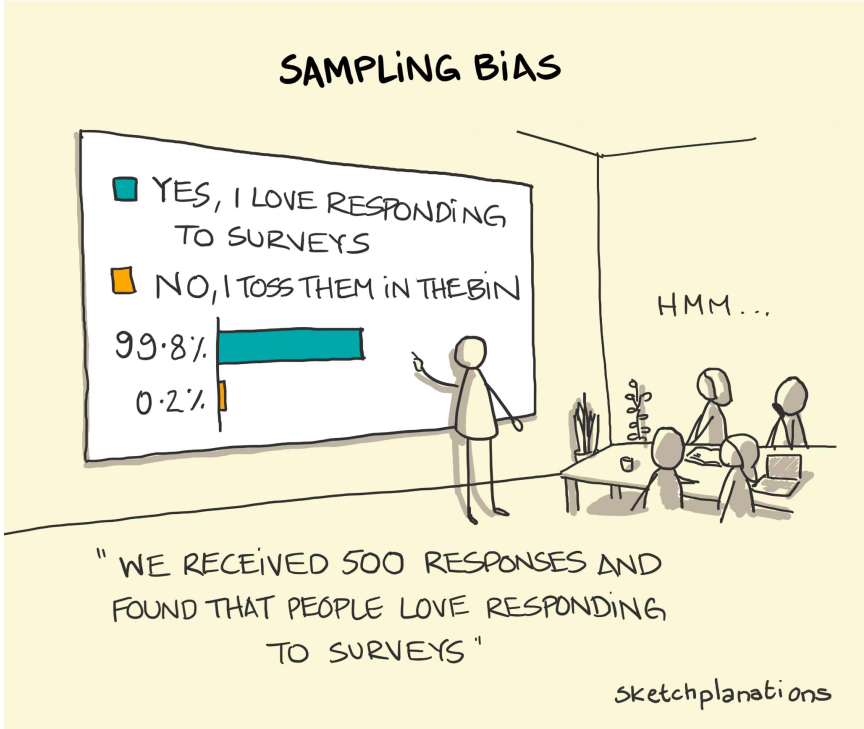
balance offers a simple workflow and methods for dealing with biased data samples when looking to infer from them to some population of interest.

Biased samples often occur in survey statistics when respondents present non-response bias or survey suffers from sampling bias (that are not missing completely at random).

Survey weighting alternatives - market ecosystem

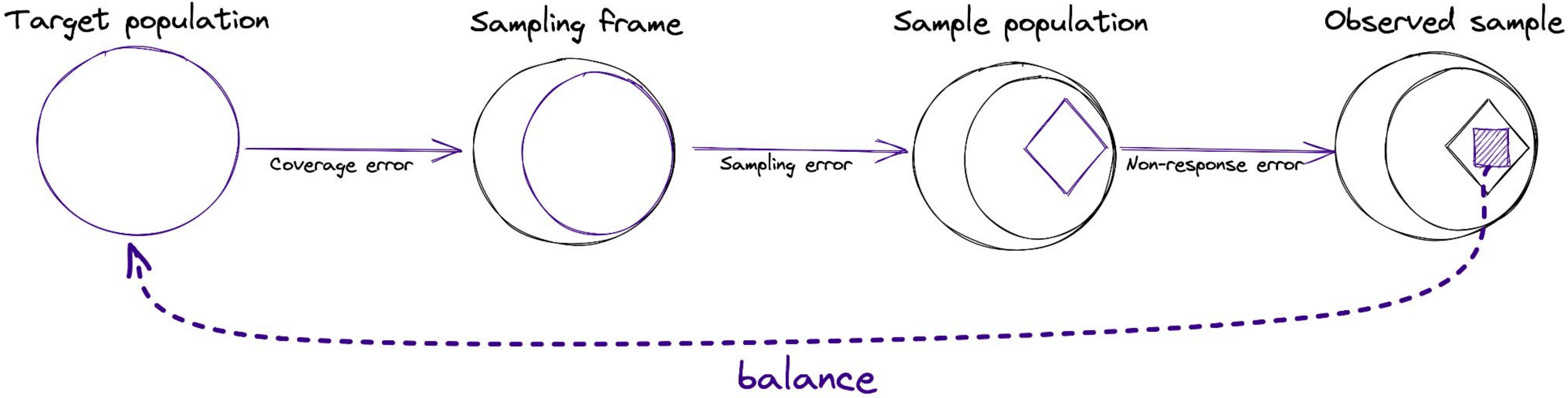


Survey Bias



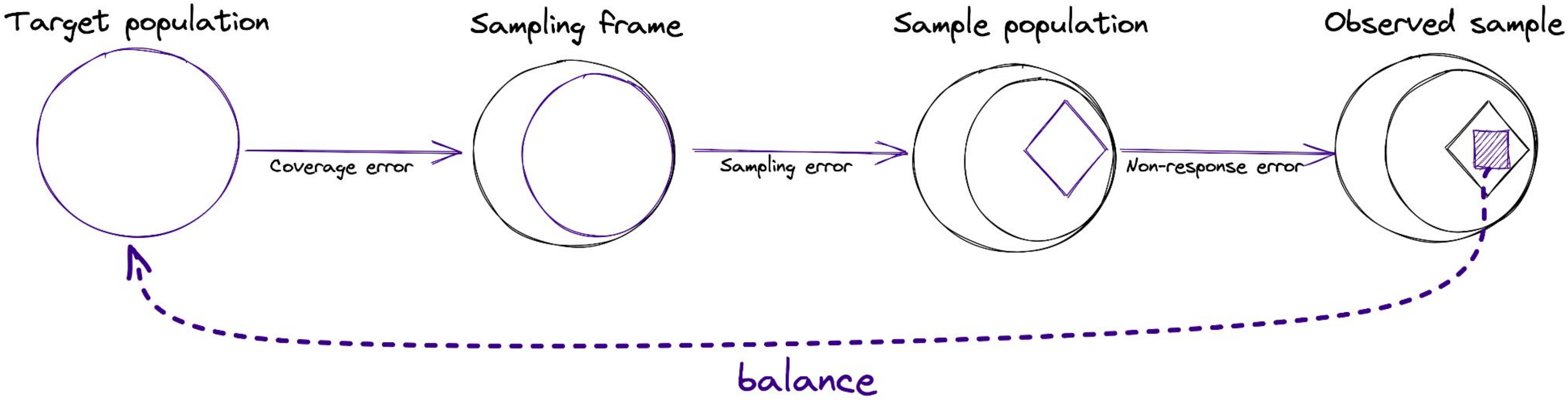
SAMPLING BIAS
YES, I LOVE RESPONDING TO SURVEYS
NO, I TOSS THEM IN THE BIN
99.8%
0.2%
HMM...
"WE RECEIVED 500 RESPONSES AND FOUND THAT PEOPLE LOVE RESPONDING TO SURVEYS"
sketchplanations

Total survey error framework - representation error: [Groves et al. 2010]



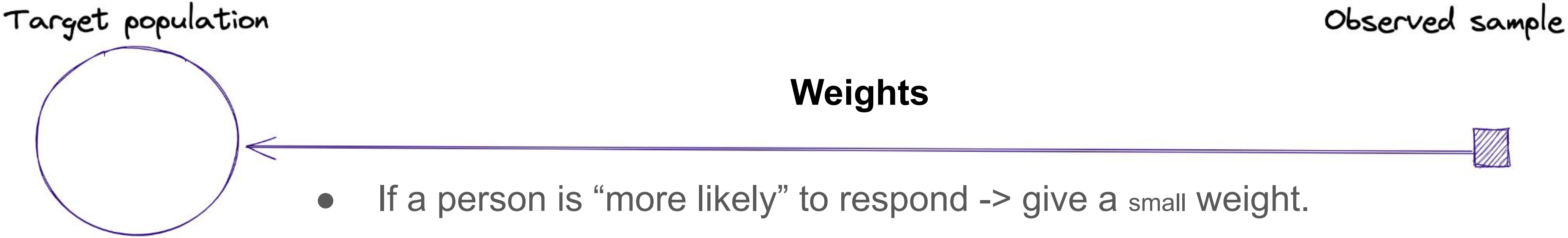
Research goal: Estimate descriptive statistics of interest for a population.

Notations



- Let **Y** be the survey response (observed only for the sample of respondents).
- Let **R** be an indicator of whether unit i responded to the survey (inclusion in sample).
- Let **X** be an auxiliary data (observed for sample and target!)

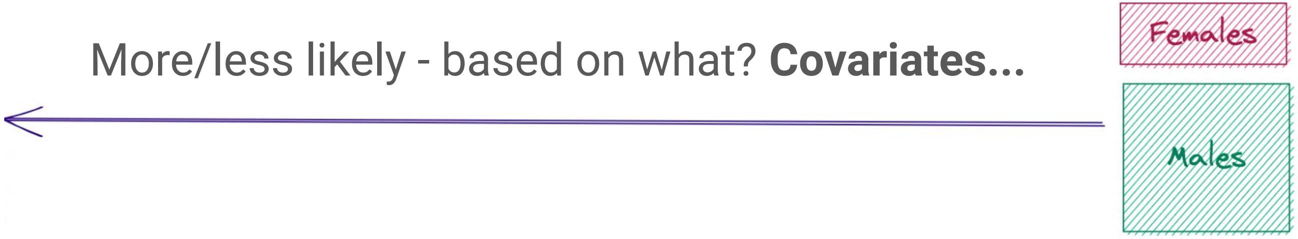
How can we mitigate survey bias? **weights**



- If a person is “more likely” to respond -> give a small weight.
- If a person is “less likely” to respond -> give a large weight.

$$\bar{y}_w = \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i}$$

How do we estimate weights? Assumptions



(1) **MAR (Missing At Random) assumption** [Rubin, 1976]**:** The response mechanism is independent of the survey responses conditional on the auxiliary data:

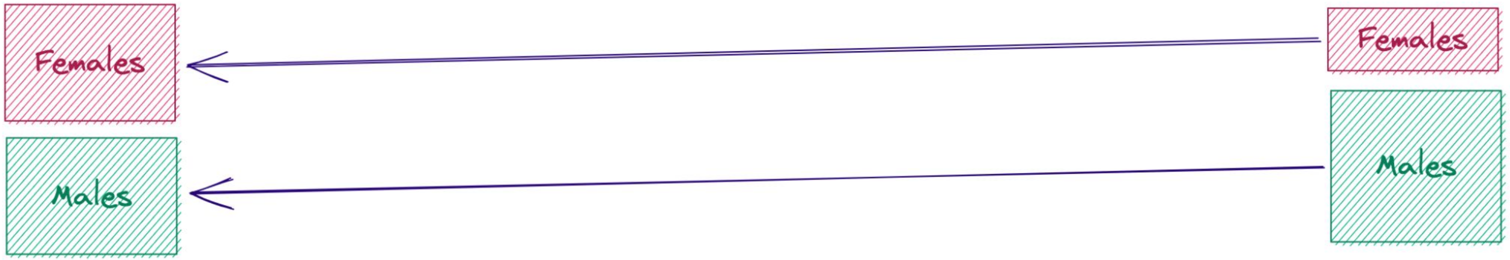
$$Y \perp\!\!\!\perp R \mid X$$

(2) **Positivity**:

$$0 < Pr(R = 1|X) < 1$$

(Also known as **strong ignorability** (or unconfoundedness in causal inference))

Estimation: Post-Stratification [Little, 1993]



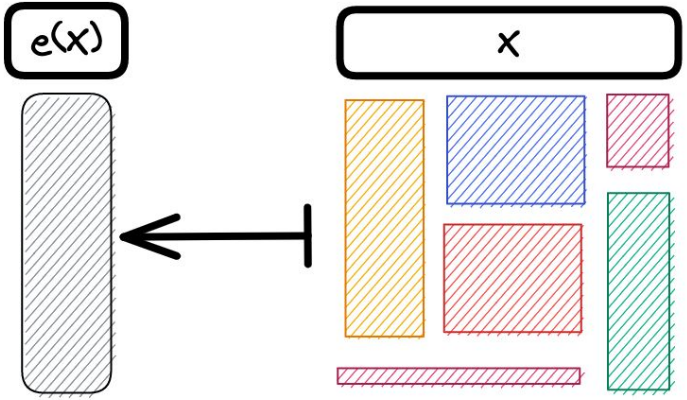
- Can be used with several covariates given the joint distribution (E.g.: Age, Gender, State)

Limitation:

- We must have “enough” respondents in each bucket.
- Must know the joint distribution→ raking
  (iterative process based on marginal distributions)

Propensity Score [Rosenbaum & Rubin, 1983]

$$e(x) = Pr(R = 1|x)$$



The assumptions:

(1) Strong ignorability: $Y \perp\!\!\!\perp R \mid X$

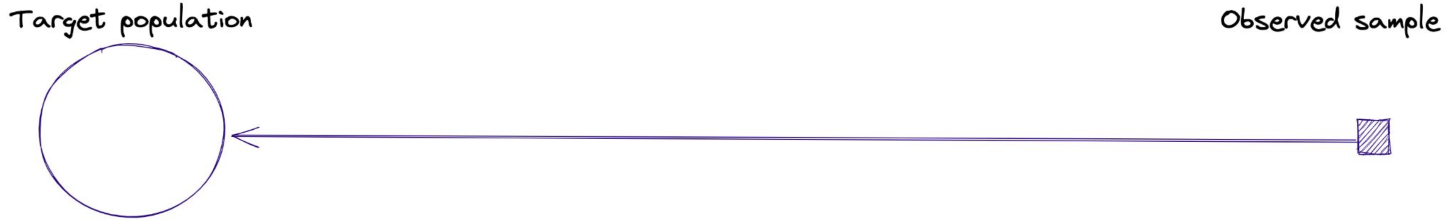
(2) Positivity: $0 < Pr(R = 1|x) < 1$

Implies:

(1) Strong ignorability given the propensity score: $Y \perp\!\!\!\perp R \mid e(X)$

(2) The propensity score is the "coarsest" balancing score (e.g. the lowest dimension score B(X) such that $Y \perp\!\!\!\perp R \mid B(X)$)

Estimation: Inverse Propensity Score Weighting (IPW or IPSW)



- Estimate the propensity score using logistic regression:

$$\ln(\frac{p_i}{1-p_i}) = \beta_0 + \overrightarrow{\beta_1} X_i$$

- ***balance*** uses **regularized logistic regression (LASSO)** (to remove covariates that are not predictive for non-response).
- Alternative model is **C**ovariate **B**alancing **P**ropensity **S**core [Imai & Ratkovic, 2014]**,** optimizing both the propensity score and the balance of the covariates.
- Estimate the weights:

$$w_i = \frac{1-\hat{p_i}}{\hat{p_i}}$$

When do survey weights help?

When:

(a) the non-response pattern is strongly related to the measurable covariates,

(b) the covariates are accurately represented in (and fixed by using) the fitted propensity score model, and

(c) the survey weights correlate (strongly “enough”) with the outcome of interest

Diagnostics measure are available to (at least partially) measure the above assumptions.

When do survey weights help?

When:

(a) the non-response pattern is strongly related to the measurable covariates,

ASMD values, and distribution plots

(b) the covariates are accurately represented in (and fixed by using) the fitted propensity score model, and

ASMD values, and distribution plots (before/after fitting the weights)

(c) the survey weights correlate (strongly “enough”) with the outcome of interest

Kish’s design effect (other quantiles of the weights), and the weighted mean (and their CI)

Diagnostics measure are available to (at least partially) measure the above assumptions.

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data relative to a target we would like to infer about

(2) **Adjusting** the data to correct for the bias by producing weights for each unit in the sample based on propensity scores

(3) **Evaluating** the final bias and the variance inflation after applying the fitted weights

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, ASMD) + weights diagnostics
+ the outcome (with/without weights)

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (**plots**, ASMD)

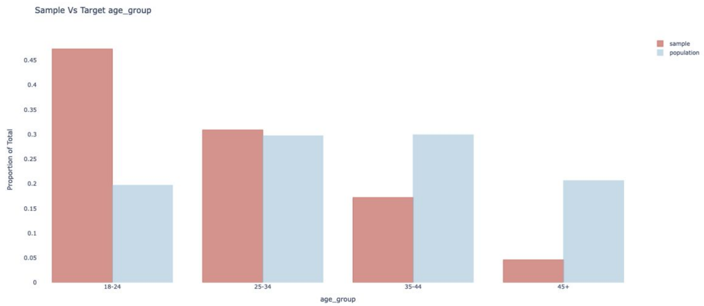
(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

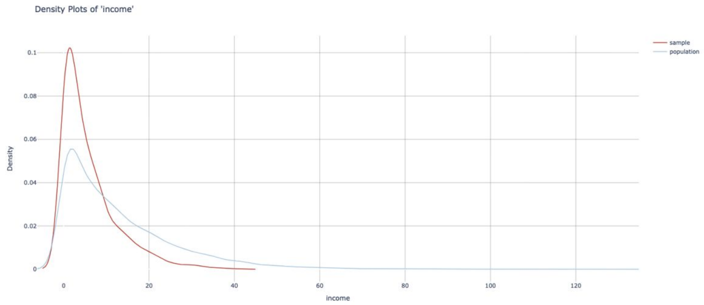
(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, ASMD) + weights diagnostics
+ the outcome (with/without weights)

Visualizing Distributions with Distribution Plots



(b) Bar plot for age group



(c) KDE for income

Un-weighted sample   Weighted sample

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, ASMD) + weights diagnostics
+ the outcome (with/without weights)

Adjustment

Researcher may choose between 4 adjustment models (IPW, CBPS, raking and post-stratification) - also - *balance* utilizes a few best practices when modeling:

1. **Feature engineering** - *balance* will automatically apply transformations to the covariates to better adjust their distribution.
2. **“Model selection”** - *balance* applies regularized logistic regression in IPW in order to reduce the “non-significant” inflation of the variance created by the weights.
3. **Weights trimming** - *balance* applies trimming of the weights to reduce the influence of extreme observations.

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

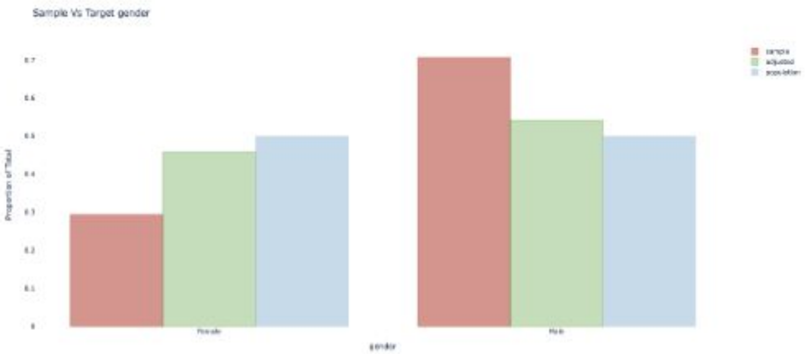
(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

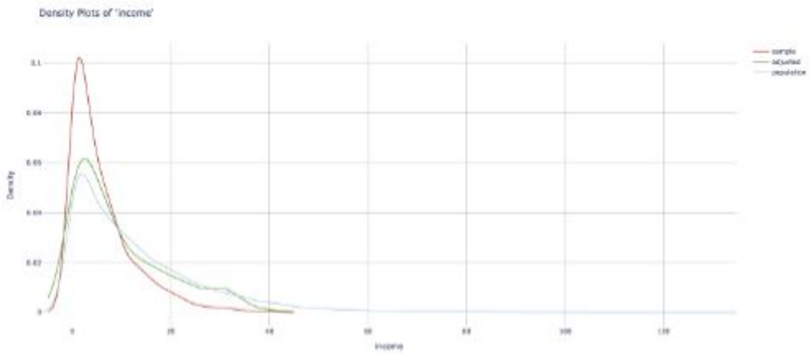
(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (**plots**, ASMD) + weights diagnostics
+ the outcome (with/without weights)

Visualizing Distributions with Distribution Plots



(a) Bar plot for gender



(b) KDE for income

Un-weighted sample   Weighted sample   True value

Examples (from simulated data) of diagnostic plots for covariates

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, **ASMD**) + weights diagnostics
+ the outcome (with/without weights)

Absolute Standardized Mean Deviation (ASMD)

It is computed as follows:

$$ASMD = \frac{\left|\bar{X}_{Sample} - \bar{X}_{Target}\right|}{SD_{Target}}$$

where $\bar{X}_{Sample}$ and $\bar{X}_{Target}$ are the means of the sample and target, and SD_{Target} is the standard deviation of the target population.

Absolute Standardized Mean Deviation (ASMD)

It is computed as follows:

$$ASMD = \frac{\left|\bar{X}_{Sample} - \bar{X}_{Target}\right|}{SD_{Target}}$$

where $\bar{X}_{Sample}$ and $\bar{X}_{Target}$ are the means of the sample and target, and SD_{Target} is the standard deviation of the target population.

$$ASMD_{diff} = ASMD_{unadjusted} - ASMD_{weighted}$$

Absolute Standardized Mean Deviation (ASMD)

Source	Weighted	Unadjusted	Unadjusted - Adjusted
age_group[T.25-34]	0.001085	0.005688	0.004602
age_group[T.35-44]	0.037455	0.312711	0.275256
age_group[T.45+]	0.129304	0.378828	0.249525
gender[Female]	0.133970	0.375699	0.241730
gender[Male]	0.109697	0.379314	0.269617
gender[_NA]	0.042278	0.006296	-0.035983
income	0.243762	0.494217	0.250455
mean(asmd)	0.131675	0.326799	0.195124

Examples (from simulated data) of an ASMD table for covariates of a sample. Columns are for cases the data is weighted, unadjusted, and their difference ($ASMD_{diff}$)

Absolute Standardized Mean Deviation (ASMD)

- Pros
  - Standardized measure (comparable across features)
  - Easy comparison over many features and alternative weighting options

- Cons
  - Hard to interpret in absolute terms (how much ASMD is “good”?)
  - Sensitive to outliers while being indifferent to distributional differences
  - Less applicable to categorical variables
    (we currently use dummy variables,
    but aggregation leads to an even less obvious interpretation)

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, ASMD) + **weights diagnostics**
+ the outcome (with/without weights)

Weights diagnostics: Kish’s design effect

A design effect measures the increase in variance of an estimate due to the use of survey weights compared to an equal probability sample of the same size. Theoretically, it is calculated as follows:

$$D_{eff} = \frac{Var_{weighted}}{Var_{unweighted}}$$

(e.g.: Deff=2 means the variance
is double when using weights
As compared to NOT using weights)

Weights diagnostics: Kish's design effect

A design effect measures the increase in variance of an estimate due to the use of survey weights compared to an equal probability sample of the same size. Theoretically, it is calculated as follows:

$$D_{eff} = \frac{Var_{weighted}}{Var_{unweighted}}$$

(e.g.: Deff=2 means the variance
is double when using weights
As compared to NOT using weights)

The effective sample size is:

$$n_{eff} = \frac{n}{D_{eff}}$$

Weights diagnostics: Kish’s design effect

Kish’s design effect

assumes no correlation between the weights and the outcome variable, also known as "haphazard weights." The formula

$$D_{eff} = \frac{n\sum_{i=1}^{n} w_i^2}{\left(\sum_{i=1}^{n} w_i\right)^2} = \frac{\frac{1}{n}\sum_{i=1}^{n} w_i^2}{\left(\frac{1}{n}\sum_{i=1}^{n} w_i\right)^2} = \frac{\overline{w^2}}{\overline{w}^2}$$

Weights diagnostics: quantiles of weights

For weights diagnostics, it's helpful to look at weights that are normalized to sum to the sample size, i.e.:

$$w_i^* = w_i / \bar{w}$$

This allows us to check:

1. Quantiles
2. Proportion of weights above/below some values
   (this helps **identify extreme values, or odd behaviors**)

Weights diagnostics

Var	Val
design_effect	1.90
effective_sample_proportion	0.53
effective_sample_size	527.04
sum	10000.00
describe_count	1000.00
describe_mean	1.00
describe_std	0.95
describe_min	0.31
describe_25%	0.38
describe_50%	0.64
describe_75%	1.20
describe_max	11.65
prop(w < 0.1)	0.00
prop(w < 0.2)	0.00
prop(w < 0.333)	0.11
prop(w < 0.5)	0.29
prop(w < 1)	0.65
prop(w >= 1)	0.35
prop(w >= 2)	0.12
prop(w >= 3)	0.03
prop(w >= 5)	0.01
prop(w >= 10)	0.00
nonparametric_skew	0.38
weighted_median_breakdown_point	0.21

Examples (from simulated data) of diagnostics statistics for weights

Workflow in practice

The main workflow of *balance* includes three steps:

(1) **Understanding** the initial bias in the data
relative to a target we would like to infer about

Checking the covariates of sample vs target (plots, ASMD)

(2) **Adjusting** the data to correct for the bias by producing
weights for each unit in the sample based on
propensity scores

Fitting a weighting model (IPW, CBPS, post-stratification, raking)

(3) **Evaluating** the final bias and the variance inflation
after applying the fitted weights

Checking the covariates of sample (with/without weights) vs target (plots, ASMD) + weights diagnostics
+ **the outcome (with/without weights)**

Outcome diagnostics

Weighted mean:

$$\bar{y}_w = \frac{\sum\limits_{i=1}^{n} w_i y_i}{\sum\limits_{i=1}^{n} w_i}$$

Variance of the weighted mean:

$$\widehat{V(\bar{y}_w)} = \frac{1}{(\sum_{i=1}^{n} w_i)^2} \sum_{i=1}^{n} w_i^2 (y_i - \bar{y}_w)^2$$

Confidence intervals for the weighted mean:

$$CI(\mu) : \bar{y}_w \pm z_{\alpha/2} \sqrt{\widehat{V(\bar{y}_w)}}$$

Outcome diagnostics

Metric	Happiness
Weighted (mean)	53.389
Target (mean)	56.278
Unadjusted (mean)	48.559
Weighted CI	(52.183, 54.595)
Target CI	(55.961, 56.595)
Unadjusted CI	(47.669, 49.449)

Examples (from simulated data) of weighted means and their confidence intervals (CI)

Outcome diagnostics

The **variance** of the weighted mean

(required for the CI)

Assumes:

1. y is fixed and known.
2. w is fixed and known.

$$\widehat{V(\bar{y}_w)} = \frac{1}{(\sum_{i=1}^n w_i)^2} \sum_{i=1}^n w_i^2 (y_i - \bar{y}_w)^2$$

The randomness comes from the selection indicator.

Outcome diagnostics

The **variance** of the weighted mean

(required for the CI)

Assumes:

1. y is fixed and known.
2. w is fixed and known.

The randomness comes from the selection indicator.

$$\bar{y}_w = \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i}$$

$$\widehat{V(\bar{y}_w)} = \frac{1}{(\sum_{i=1}^{n} w_i)^2} \sum_{i=1}^{n} w_i^2 (y_i - \bar{y}_w)^2$$

$$\hat{\bar{Y}} = \frac{\sum_{i=1}^{N} I_i \frac{y_i}{\pi_i}}{\sum_{i=1}^{N} I_i \frac{1}{\pi_i}} = \frac{\sum_{i=1}^{N} \check{y}_i'}{\sum_{i=1}^{N} \check{1}_i'} = \frac{\sum_{i=1}^{N} w_i y_i'}{\sum_{i=1}^{N} w_i 1_i'} = \frac{\sum_{i=1}^{n} w_i y_i'}{\sum_{i=1}^{n} w_i 1_i'} = \bar{y}_w$$

Outcome diagnostics

The variance of the weighted mean formula assumes that the weights are known and fixed quantities.

It does not account for the uncertainty that is introduced from the estimation of the weights.

An end-to-end bootstrap simulation can be performed to account for this variance.

Example: simulated data

Source

https://import-balance.org/docs/tutorials/quickstart/

Example: simulated data

Getting simulated data (stored in *balance*)

```
from balance import load_data
```

```
INFO (2023-05-14 09:00:15,410) [__init__/<module> (line 52)]: Using balance version 0.9.0
```

```
In [2]: target_df, sample_df = load_data()

        print("target_df: \n", target_df.head())
        print("sample_df: \n", sample_df.head())
```

```
target_df: 
       id gender age_group     income  happiness
0  100000   Male       45+  10.183951  61.706333
1  100001   Male       45+   6.036858  79.123670
2  100002   Male     35-44   5.226629  44.206949
3  100003    NaN       45+   5.752147  83.985716
4  100004    NaN     25-34   4.837484  49.339713
sample_df: 
   id  gender age_group     income  happiness
0  0    Male     25-34   6.428659  26.043029
1  1  Female     18-24   9.940280  66.885485
2  2    Male     18-24   2.673623  37.091922
3  3     NaN     18-24  10.550308  49.394050
4  4     NaN     18-24   2.689994  72.304208
```

Example: simulated data

Loading the simulated data into an instance of ‘Sample’ class (from *balance*)

```
from balance import Sample
```

```
sample = Sample.from_frame(sample_df, outcome_columns=["happiness"])
# Often times we don'y have the outcome for the target. In this case we've added it just to validate later
target = Sample.from_frame(target_df, outcome_columns=["happiness"])
```

Example: simulated data

The ‘sample’ object has

Many methods, attributes and

Properties.

E.g.: using “.df” will get us
The DataFrame stored in
the object.

```
sample.df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          1000 non-null   object
 1   gender      912 non-null    object
 2   age_group   1000 non-null   object
 3   income      1000 non-null   float64
 4   happiness   1000 non-null   float64
 5   weight      1000 non-null   int64
dtypes: float64(2), int64(1), object(3)
memory usage: 47.0+ KB
```

Example: simulated data

Each Sample object

has a print-out

with information

about what’s stored.

```
sample
```

```
(balance.sample_class.Sample)

        balance Sample object
        1000 observations x 3 variables: gender,age_group,income
        id_column: id, weight_column: weight,
        outcome_columns: happiness
```

```
target
```

```
(balance.sample_class.Sample)

        balance Sample object
        10000 observations x 3 variables: gender,age_group,income
        id_column: id, weight_column: weight,
        outcome_columns: happiness
```

Example: simulated data

Using “.set_target” we can connect the target Sample with the sample Sample

```
sample_with_target = sample.set_target(target)
```

Looking on `sample_with_target` now, it has the target atteched:

```
sample_with_target
```

```
(balance.sample_class.Sample)

        balance Sample object with target set
        1000 observations x 3 variables: gender,age_group,income
        id_column: id, weight_column: weight,
        outcome_columns: happiness

            target:

                balance Sample object
                10000 observations x 3 variables: gender,age_group,income
                id_column: id, weight_column: weight,
                outcome_columns: happiness

            3 common variables: gender,age_group,income
```

Example: simulated data

Comparing means

```
print(sample_with_target.covars().mean().T)
```

```
source                          self     target
_is_na_gender[T.True]       0.088000   0.089800
age_group[T.25-34]          0.300000   0.297400
age_group[T.35-44]          0.156000   0.299200
age_group[T.45+]            0.053000   0.206300
gender[Female]              0.268000   0.455100
gender[Male]                0.644000   0.455100
gender[_NA]                 0.088000   0.089800
income                      6.297302  12.737608
```

Example: simulated data

Looking at ASMD

```
print(sample_with_target.covars().asmd().T)
```

```
source                    self
age_group[T.25-34]    0.005688
age_group[T.35-44]    0.312711
age_group[T.45+]      0.378828
gender[Female]        0.375699
gender[Male]          0.379314
gender[_NA]           0.006296
income                0.494217
mean(asmd)            0.326799
```

```
print(sample_with_target.covars().asmd(aggregate_by_main_covar = True).T)
```

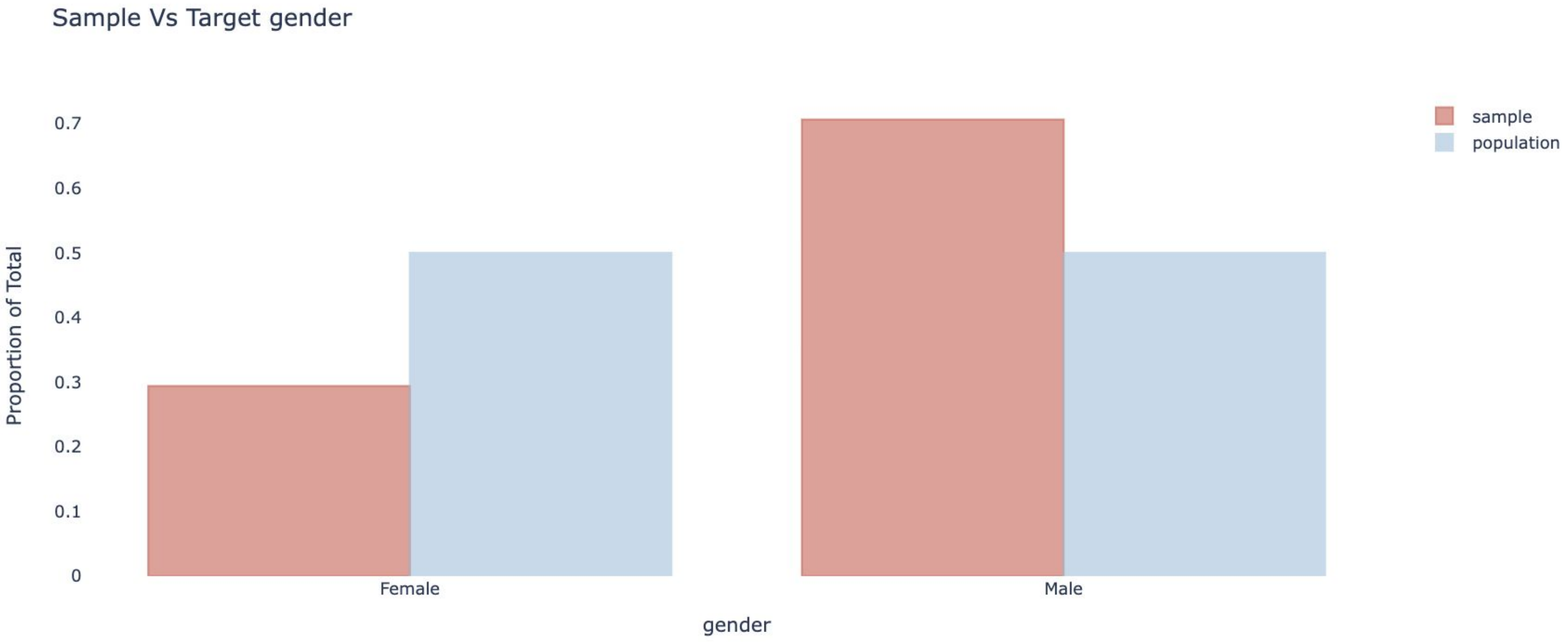
```
source          self
age_group   0.232409
gender      0.253769
income      0.494217
mean(asmd)  0.326799
```

Example: simulated data

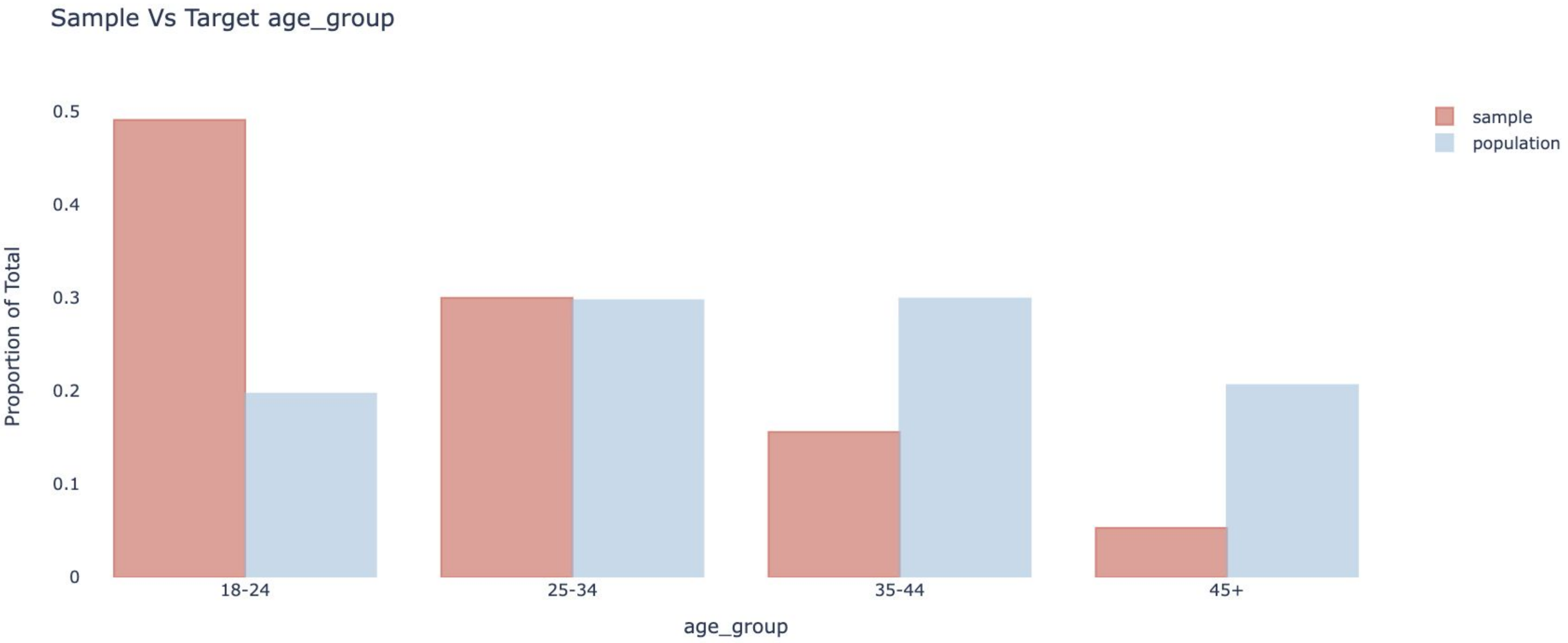
And we can get plots for all covariates

```
sample_with_target.covars().plot()
```

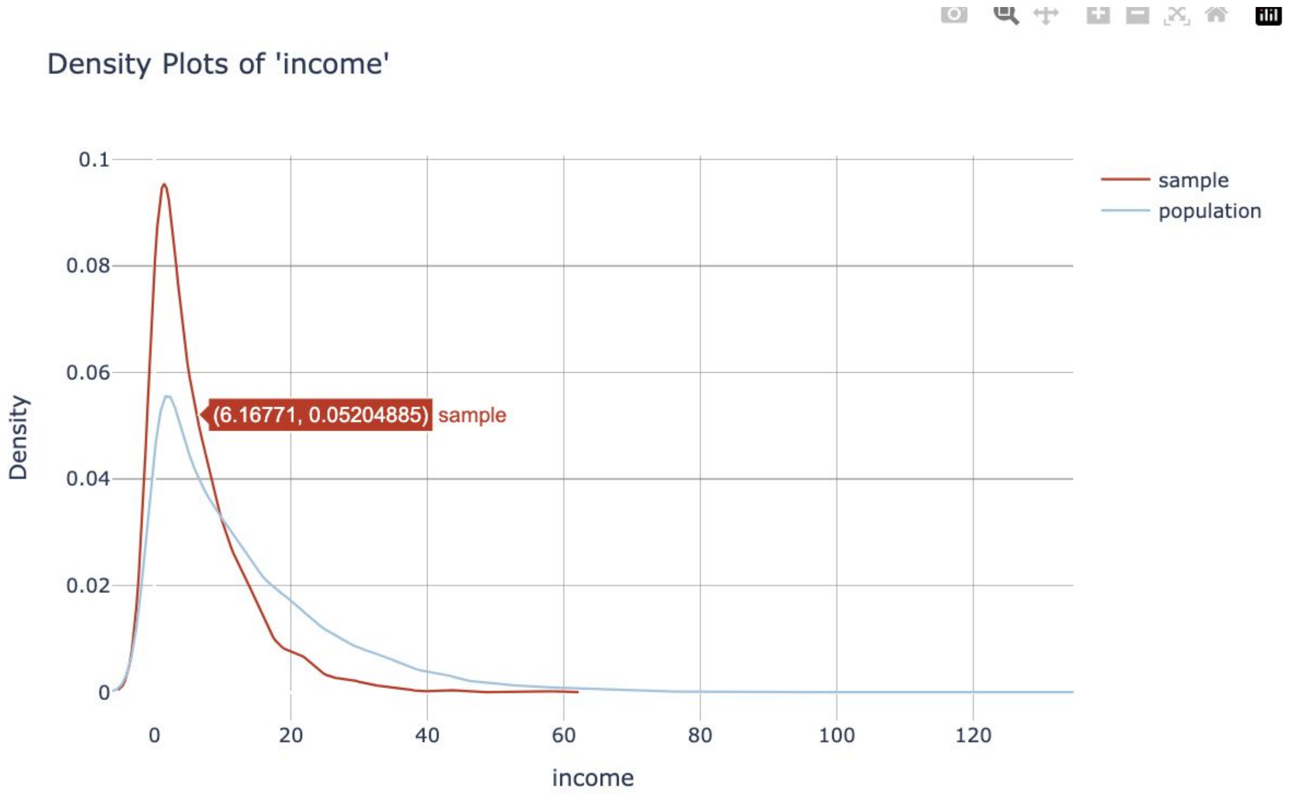
Example: simulated data



Example: simulated data



Example: simulated data



Example: simulated data

We use “.adjust()” to fit a model (e.g.: IPW, CBPS, raking, or post-stratification)

```
]: # Using ipw to fit survey weights
   adjusted = sample_with_target.adjust()
```

```
INFO (2023-05-14 09:00:16,643) [ipw/ipw (line 428)]: Starting ipw function
INFO (2023-05-14 09:00:16,646) [adjustment/apply_transformations (line 257)]: Adding the variables: []
INFO (2023-05-14 09:00:16,647) [adjustment/apply_transformations (line 258)]: Transforming the variables: ['gender', 'age_group', 'income']
INFO (2023-05-14 09:00:16,658) [adjustment/apply_transformations (line 295)]: Final variables in output: ['gender', 'age_group', 'income']
INFO (2023-05-14 09:00:16,667) [ipw/ipw (line 462)]: Building model matrix
INFO (2023-05-14 09:00:16,761) [ipw/ipw (line 486)]: The formula used to build the model matrix: ['income + gender + age_group + _is_na_gender']
INFO (2023-05-14 09:00:16,762) [ipw/ipw (line 489)]: The number of columns in the model matrix: 16
INFO (2023-05-14 09:00:16,763) [ipw/ipw (line 490)]: The number of rows in the model matrix: 11000
INFO (2023-05-14 09:00:16,771) [ipw/ipw (line 521)]: Fitting logistic model
INFO (2023-05-14 09:00:18,481) [ipw/ipw (line 564)]: max_de: None
INFO (2023-05-14 09:00:18,485) [ipw/ipw (line 594)]: Chosen lambda for cv: [0.0131066]
INFO (2023-05-14 09:00:18,487) [ipw/ipw (line 602)]: Proportion null deviance explained [0.17168419]
```

Example: simulated data

The print-out tells us what we got in the object

```
print(adjusted)
```

```
Adjusted balance Sample object with target set using ipw
1000 observations x 3 variables: gender,age_group,income
id_column: id, weight_column: weight,
outcome_columns: happiness

    target:

        balance Sample object
        10000 observations x 3 variables: gender,age_group,income
        id_column: id, weight_column: weight,
        outcome_columns: happiness

    3 common variables: gender,age_group,income
```

Example: simulated data

We can evaluate the results

```
print(adjusted.summary())
```

```
Covar ASMD reduction: 59.7%, design effect: 1.897
Covar ASMD (7 variables): 0.327 -> 0.132
Model performance: Model proportion deviance explained: 0.172
```

```
print(adjusted.covars().mean().T)
```

```
source                       self     target  unadjusted
_is_na_gender[T.True]    0.101888   0.089800    0.088000
age_group[T.25-34]       0.297896   0.297400    0.300000
age_group[T.35-44]       0.282048   0.299200    0.156000
age_group[T.45+]         0.153975   0.206300    0.053000
gender[Female]           0.388383   0.455100    0.268000
gender[Male]             0.509730   0.455100    0.644000
gender[_NA]              0.101888   0.089800    0.088000
income                   9.561063  12.737608    6.297302
```

Example: simulated data

We can look at the effect of weighting on the ASMD:

```
print(adjusted.covars().asmd().T)
```

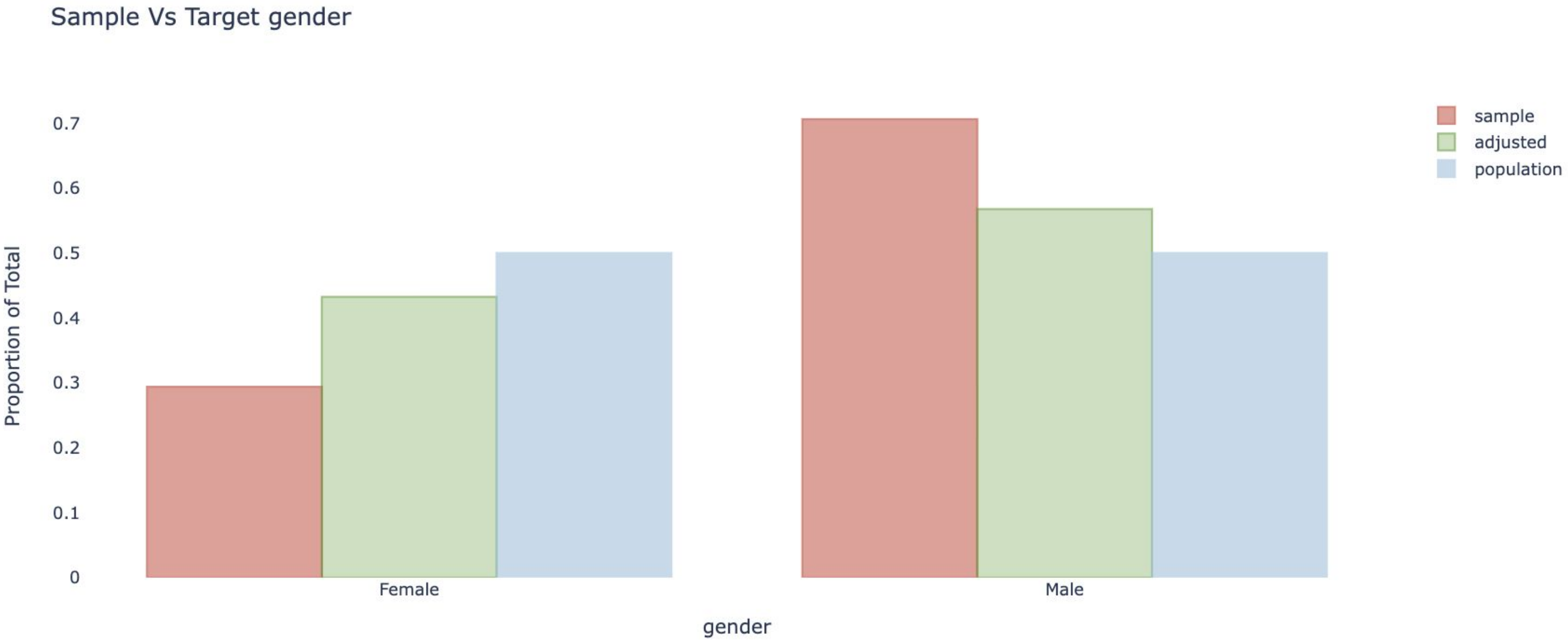
```
source                  self  unadjusted  unadjusted - self
age_group[T.25-34]  0.001085    0.005688           0.004602
age_group[T.35-44]  0.037455    0.312711           0.275256
age_group[T.45+]    0.129304    0.378828           0.249525
gender[Female]      0.133970    0.375699           0.241730
gender[Male]        0.109697    0.379314           0.269617
gender[_NA]         0.042278    0.006296          -0.035983
income              0.243762    0.494217           0.250455
mean(asmd)          0.131675    0.326799           0.195124
```

Example: simulated data

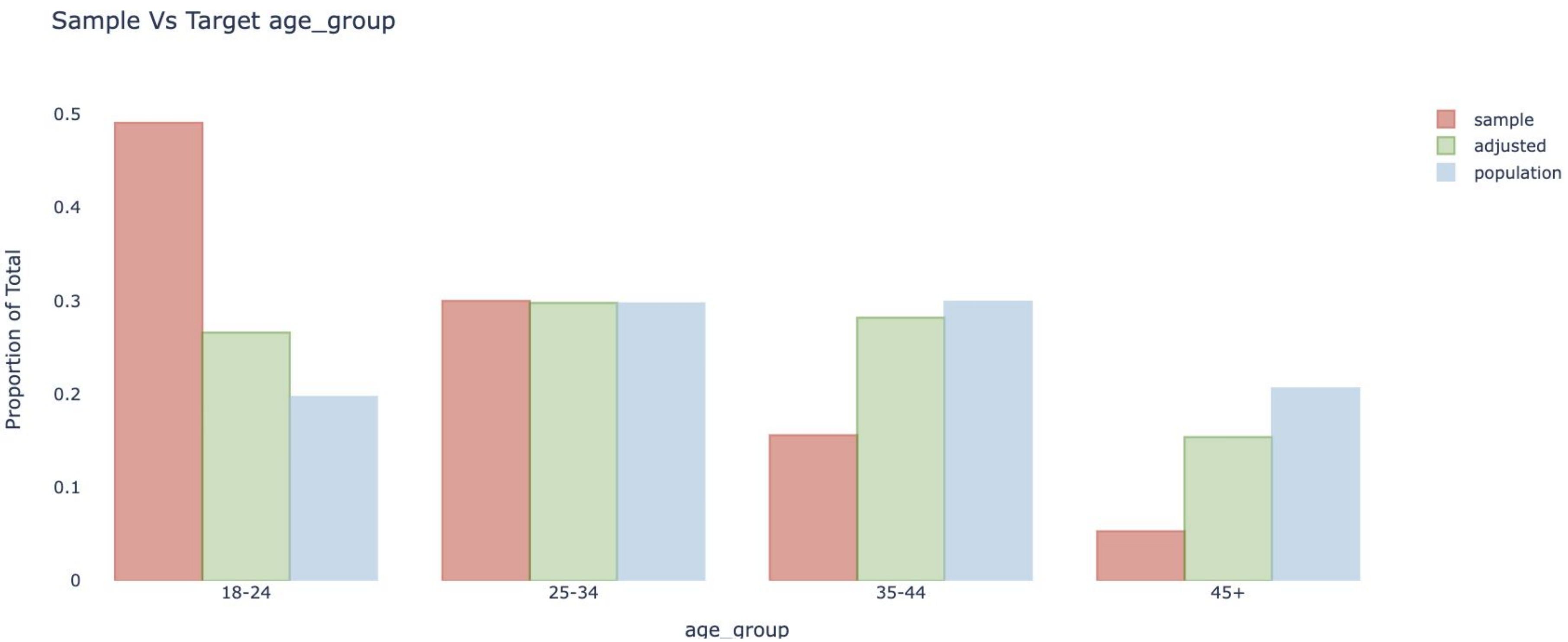
And we can plot our data again (seeing the effect of weighting on the covariates)

```
adjusted.covars().plot()
```

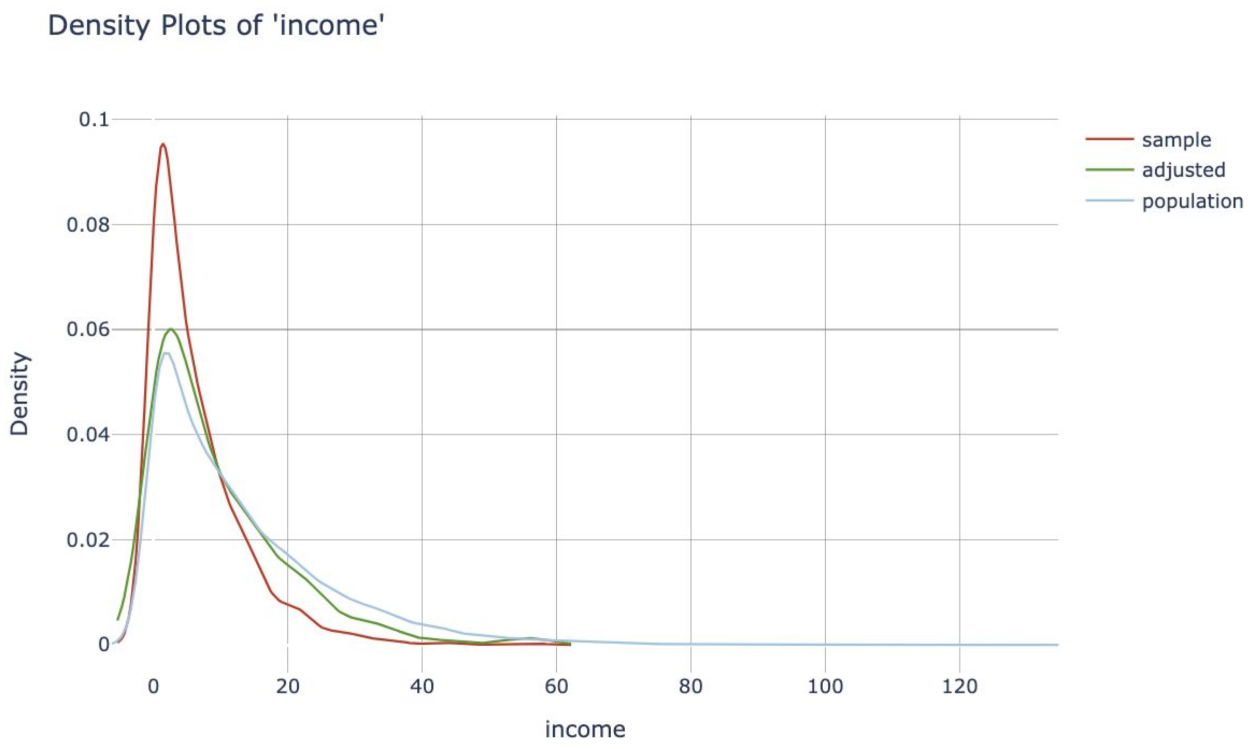
Example: simulated data



Example: simulated data



Example: simulated data



Example: simulated data

Weight diagnostics

```
print(adjusted.weights().summary().round(2))
```

	var	val
0	design_effect	1.90
1	effective_sample_proportion	0.53
2	effective_sample_size	527.04
3	sum	10000.00
4	describe_count	1000.00
5	describe_mean	1.00
6	describe_std	0.95
7	describe_min	0.31
8	describe_25%	0.38
9	describe_50%	0.64
10	describe_75%	1.20
11	describe_max	11.65
12	prop(w < 0.1)	0.00
13	prop(w < 0.2)	0.00
14	prop(w < 0.333)	0.11
15	prop(w < 0.5)	0.29
16	prop(w < 1)	0.65
17	prop(w >= 1)	0.35
18	prop(w >= 2)	0.12
19	prop(w >= 3)	0.03
20	prop(w >= 5)	0.01
21	prop(w >= 10)	0.00

Example: simulated data

Outcome - weighted means

```
print(adjusted.outcomes().summary())
```

```
1 outcomes: ['happiness']
Mean outcomes (with 95% confidence intervals):
source          self  target  unadjusted             self_ci        target_ci     unadjusted_ci
happiness     53.389  56.278      48.559  (52.183, 54.595)  (55.961, 56.595)  (47.669, 49.449)
```

Example: simulated data

Outcome

```
adjusted.outcomes().plot()
```

Example: simulated data

Downloading the results

```
adjusted.to_download()
```

Click here to download: /tmp/tmp_balance_out_f686eeac-53e1-4664-a4d7-384ae031309a.csv

```
# We can prepare the data to be exported as csv - showing the first 500 charaacters for simplicity:
adjusted.to_csv()[0:500]
```

```
'id,gender,age_group,income,happiness,weight\n0,Male,25-34,6.428659499046228,26.043028759747298,6.3736
66.88548460632677,7.348179942660277\n2,Male,18-24,2.6736231547518043,37.09192191668306,3.716606477803
71002,6.509195107938343\n4,,18-24,2.689993854299385,72.30420755038209,6.394774127610416\n5,,35-44,5.995
1593\n6,,18-24,12.63469573898972,31.663293445944596,7.7552609990'
```

balance

Thanks!

Questions?

Tal Sarig

Tal Galili

Central Applied Science

Meta

References

- Donald B Rubin. Inference and missing data. Biometrika, 63(3):581–592, 1976.
- Kosuke Imai and Marc Ratkovic. Covariate balancing propensity score. Journal of the Royal Statistical Society: Series B: Statistical Methodology, pages 243–263, 2014
- Leslie Kish. Survey Sampling. John Wiley Sons, Inc., New York, 1965.
- Leslie Kish and J. Official Stat. Weighting for unequal π. Journal of Official Statistics, 8:183–200, 1992.
- Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. Biometrika, 70(1):41–55, 1983.
- Robert M Groves and Lars Lyberg. Total survey error: Past, present, and future. Public opinion quarterly, 74(5):849–879, 2010.
- Roderick JA Little. Post-stratification: a modeler's perspective. Journal of the American Statistical Association, 88(423):1001–1012, 1993.
- Siegfried Gabler, Sabine H ̈ader, and Partha Lahiri. A model based justification of kish's formula for design effects for weighting and clustering. Survey Methodology, 25:105–106, 1999.

Proof for Kish's design effect

$$var\left(\bar{y}_w\right) \overset{1}{=} var\left(\frac{\sum\limits_{i=1}^{n} w_i y_i}{\sum\limits_{i=1}^{n} w_i}\right) \overset{2}{=} var\left(\sum_{i=1}^{n} w'_i y_i\right) \overset{3}{=} \sum_{i=1}^{n} var\left(w'_i y_i\right)$$

$$\overset{4}{=} \sum_{i=1}^{n} w_i'^2 var\left(y_i\right) \overset{5}{=} \sum_{i=1}^{n} w_i'^2 \sigma^2 \overset{6}{=} \sigma^2 \sum_{i=1}^{n} w_i'^2 \overset{7}{=} \sigma^2 \frac{\sum\limits_{i=1}^{n} w_i^2}{\left(\sum\limits_{i=1}^{n} w_i\right)^2}$$

$$\overset{8}{=} \sigma^2 \frac{\sum\limits_{i=1}^{n} w_i^2}{\left(\sum\limits_{i=1}^{n} w_i \frac{n}{n}\right)^2} \overset{9}{=} \sigma^2 \frac{\sum\limits_{i=1}^{n} w_i^2}{\left(\frac{\sum\limits_{i=1}^{n} w_i}{n}\right)^2 n^2} \overset{10}{=} \frac{\sigma^2}{n} \frac{\frac{\sum\limits_{i=1}^{n} w_i^2}{n}}{\left(\frac{\sum\limits_{i=1}^{n} w_i}{n}\right)^2}$$

$$\overset{11}{=} \frac{\sigma^2}{n} \frac{\overline{w^2}}{\bar{w}^2} \overset{12}{=} var\left(\bar{y}'\right) D_{eff}$$

$$\implies D_{eff(kish)} = \frac{var\left(\bar{y}_w\right)}{var\left(\bar{y}'\right)}$$

Why the relation between the response and the weighting variable is important?

Given a fixed amount of imbalance between the sample of respondents and the population in the weighting variables, the potential effect of correcting the bias depends on the correlation between the variable and the response. The higher the correlation is the larger the effect is.

This is demonstrated in a simple analytic case: Assume we have two groups A and B, where the probability for True in each group is pA and pB, respectively. In addition, assume that the two groups are balanced in the sample, but not in the population, where the percentage of group “A” in the population is p and for group “B” 1-p.

The difference in the responses between the two groups can be measured with pB-pA and is plotted on the X-axis (assuming pA=0.1). The imbalance between the groups is measured by ASMD, and can be written as (0.5-p)/sqrt(p(1-p)). We consider the ratio of the prefect weighted average response pA*p+pB(1-p) and the unweighted average response 0.5pA+0.5pB.

As expected, when the asmd is zero, i.e. when there is no imbalance between the groups, the weighted and unweighted reponses are the same. Similarly, when pA and pB are the same and there is no difference in the response between the groups, the weighted and unweighted responses are also the same. When the asmd increases, or when the difference in response between the groups increases, the ratio becomes larger, meaning the importance of weighting increases.