# Package 'approxOT'

July 22, 2025

**Type** Package

**Title** Approximate and Exact Optimal Transport Methods

**Version** 1.2

**Date** 2025-01-06

**Maintainer** Eric Dunipace <edunipace@mail.harvard.edu>

**Description** R and C++ functions to perform exact and approximate optimal transport. All C++ methods can be linked to other R packages via their header files.

**License** GPL (== 3.0)

**Imports** Rcpp (>= 1.0.3), stats

**LinkingTo** Rcpp, RcppEigen, RcppCGAL, BH

**BugReports** https://github.com/ericdunipace/approxOT/issues

**Suggests** testthat (>= 2.1.0), transport

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**SystemRequirements** C++17

**URL** https://github.com/ericdunipace/approxOT

**NeedsCompilation** yes

**Author** Eric Dunipace [aut, cre] (ORCID: <https://orcid.org/0000-0001-8909-213X>),
Andrew Johnson [ctb] (ORCID: <https://orcid.org/0000-0001-7000-8065>),
Espen Bernton [ctb] ('Hilbert Sort' adapted from their code),
Mathieu Gerber [ctb] ('Hilbert Sort' adapted from their code),
Pierre Jacob [ctb] ('Hilbert Sort' adapted from their code),
Dominic Schuhmacher [ctb] ('Shortsimplex' optimal transport method adapted from their code),
Nicolas Bonneel [ctb] ('network simplex' algorithm adapted from their code)

**Repository** CRAN

**Date/Publication** 2025-01-09 12:30:06 UTC

# Contents



---

`approxOT` *An R package to perform exact and approximate optimal transport.*

---

### Description

R and C++ functions to perform exact and approximate optimal transport. All C++ methods are linkable to other R packages via their header files.

### Author(s)

Eric Dunipace

### See Also

Useful links:

- `https://github.com/ericdunipace/approxOT`
- Report bugs at `https://github.com/ericdunipace/approxOT/issues`

---

`as.matrix.transport.plan` *Transform transportation plan to transportation matrix*

---

### Description

Transform transportation plan to transportation matrix

### Usage

```
## S3 method for class 'transport.plan'
as.matrix(x, ...)
```

### Arguments

x — An object of class ‘transport.plan‘. See output of (transport_plan)[transport_plan()]

... — Unused arguments

### Value

A matrix specifying the minimal joint distribution between samples. Margins will be equal to the marginal distributions of the samples

### Examples

```
set.seed(203987)
n <- 5
d <- 2
x <- matrix(rnorm(d*n), nrow=d, ncol=n)
y <- matrix(rnorm(d*n), nrow=d, ncol=n)
#get hilbert sort orders for x in backwards way
trans_plan <- transport_plan(X=x, Y=x, ground_p = 2, p = 2,
                         observation.orientation =  "colwise",
                         method = "hilbert")
trans_matrix <- as.matrix(trans_plan)
print(trans_matrix)
```

---

## as.transport.plan — *Transform transportation matrix to transportation plan*

---

### Description

Transform transportation matrix to transportation plan

### Usage

```
as.transport.plan(transport_matrix, ...)
```

### Arguments

transport_matrix — A matrix that is a transportation matrix, i.e. the minimal joint distribution for two samples.

... — Unused arguments

### Value

An object of class ‘transport.plan‘. See output of (transport_plan)[transport_plan]

## Examples

```
set.seed(203987)
n <- 5
d <- 2
x <- matrix(stats::rnorm(d*n), nrow=d, ncol=n)
y <- matrix(stats::rnorm(d*n), nrow=d, ncol=n)
#get hilbert sort orders for x in backwards way
trans_plan <- transport_plan(X=x, Y=x, ground_p = 2, p = 2,
                         observation.orientation =  "colwise",
                         method = "hilbert")
trans_matrix <- as.matrix(trans_plan$tplan)
tplan2 <- as.transport.plan(trans_matrix)
all.equal(tplan2, trans_plan$tplan)
```

---

# cost_calc *Calculate cost matrix*

---

## Description

Calculate cost matrix

## Usage

```
cost_calc(X, Y, ground_p)
```

## Arguments

| | |
|---|---|
| X | matrix of values in first sample. Observations should be by column, not rows. |
| Y | matrix of Values in second sample. Observations should be by column, not rows. |
| ground_p | power of the Lp norm to use in cost calculation. |

## Value

matrix of costs

## Examples

```
X <- matrix(rnorm(10*100), 10, 100)
Y <- matrix(rnorm(10*100), 10, 100)
# the Euclidean distance
cost <- cost_calc(X, Y, ground_p = 2)
```

## hilbert.projection *Get order along the Hilbert curve*

### Description

Get order along the Hilbert curve

### Usage

```
hilbert.projection(X, Sigma = NULL)
```

### Arguments

| | |
|---|---|
| `X` | matrix of values. Observations are unique by rows. |
| `Sigma` | Covariance of the data. If provided, uses a Mahalanobis distance. |

### Value

Index of orders

### Examples

```
X <- matrix(rnorm(10*3), 3, 10)
idx <- hilbert.projection(X)
print(idx)
```

## is.transport.plan *Check if function is a transport.plan*

### Description

Check if function is a transport.plan

### Usage

```
is.transport.plan(tplan)
```

### Arguments

| | |
|---|---|
| `tplan` | An object of class ‘transport.plan‘. See output of (transport_plan)[transport_plan] |

### Value

Logical

### Examples

```
set.seed(203987)
n <- 5
d <- 2
x <- matrix(rnorm(d*n), nrow=d, ncol=n)
y <- matrix(rnorm(d*n), nrow=d, ncol=n)
#get hilbert sort orders for x in backwards way
trans_plan <- transport_plan(X=x, Y=x, ground_p = 2, p = 2,
                         observation.orientation =  "colwise",
                         method = "hilbert")
print(is.transport.plan(trans_plan))
```

---

## transport_options *Function returning supported optimal transportation methods.*

---

### Description

Function returning supported optimal transportation methods.

### Usage

```
transport_options()
```

### Details

The currently supported methods are

- exact, networkflow: Utilize the networkflow algorithm to solve the exact optimal transport problem
- shortsimplex: Use the shortsimplex algorithm to solve the exact optimal transport problem
- sinkhorn: Use Sinkhorn's algorithm to solve the approximate optimal transport problem
- sinkhorn_log: Use Sinkhorn's algorithm on a log-scale for added stability to solve the approximate optimal transport problem
- greenkhorn: Use the Greenkhorn algorithm to solve the approximate optimal transport problem
- hilbert: Use hilbert sorting to perform approximate optimal transport
- rank: use the average covariate ranks to perform approximate optimal transport
- univariate: Use appropriate optimal transport methods for univariate data
- swapping: Utilize the swapping algorithm to perform approximate optimal transport
- sliced: Use the sliced optimal transport distance

### Value

Returns a vector of supported transport methods transport_plan *Optimal transport plans*

## Description

Optimal transport plans

## Usage

```
transport_plan(
  X,
  Y,
  a = NULL,
  b = NULL,
  p = 2,
  ground_p = 2,
  observation.orientation = c("rowwise", "colwise"),
  method = transport_options(),
  ...
)
```

## Arguments

| | |
|---|---|
| X | The covariate data of the first sample. |
| Y | The covariate data of the second sample. |
| a | Optional. Empirical measure of the first sample |
| b | Optional. Empirical measure of the second sample |
| p | The power of the Wasserstein distance |
| ground_p | The power of the Lp norm |
| observation.orientation | Are observations by row ("rowwise") or column ("colwise"). |
| method | Which transportation method to use. See [transport_options][transport_options] |
| ... | Additional arguments for various methods<br>• "niter": The number of iterations to use for the entropically penalized optimal transport distances<br>• "epsilon": The penalty for entropy penalized optimal transport distances. Should be > 0.<br>• "unbiased": If using Sinkhorn distances, should the distance be de-biased? (TRUE/FALSE)<br>• "nboot": If using sliced Wasserstein distances, specify the number of Monte Carlo samples |

### Value

a list with slots "tplan" and "cost". "tplan" is the optimal transport plan and "cost" is the optimal transport distance.

### Examples

```
set.seed(203987)
n <- 100
d <- 10
x <- matrix(stats::rnorm(d*n), nrow=d, ncol=n)
y <- matrix(stats::rnorm(d*n), nrow=d, ncol=n)
#get hilbert sort orders for x in backwards way
transx <- transport_plan(X=x, Y=x, ground_p = 2, p = 2,
                         observation.orientation =  "colwise",
                         method = "hilbert")
```

---

## transport_plan_given_C

*Optimal transport plans given a pre-specified cost*

---

### Description

Optimal transport plans given a pre-specified cost

### Usage

```
transport_plan_given_C(
  mass_x,
  mass_y,
  p = 2,
  cost = NULL,
  method = "exact",
  cost_a = NULL,
  cost_b = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| mass_x | The empirical measure of the first sample |
| mass_y | The empirical measure of the second sample. |
| p | The power of the Wasserstein distance |
| cost | Specify the cost matrix in advance. |
| method | The transportation method to use, one of "exact", "networkflow","shortsimplex", "sinkhorn", "greenkhorn" |
| cost_a | The cost matrix for the first sample with itself. Only used for unbiased Sinkhorn |

cost_b — The cost matrix for the second sample with itself. Only used for unbiased Sinkhorn

... — Additional arguments for various methods

- "niter": The number of iterations to use for the entropically penalized optimal transport distances
- "epsilon": The penalty for entropy penalized optimal transport distances. Should be > 0.
- "unbiased": If using Sinkhorn distances, should the distance be de-biased? (TRUE/FALSE)

## Value

A transportation plan as an object of class "transport.plan", which is a list with slots "from","to", and "mass".

## Examples

```
n <- 32
d <- 5
set.seed(293897)
A <- matrix(stats::rnorm(n*d),nrow=d,ncol=n)
B <- matrix(stats::rnorm(n*d),nrow=d,ncol=n)
transp.meth <- "sinkhorn"
niter <- 1e2
test <- transport_plan_given_C(rep(1/n,n),
rep(1/n,n),  2, cost = cost_calc(A,B,2),
"sinkhorn", niter = niter)
```

---

# transport_plan_multimarg

*Multimarginal optimal transport plans*

---

## Description

Multimarginal optimal transport plans

## Usage

```
transport_plan_multimarg(
  ...,
  p = 2,
  ground_p = 2,
  observation.orientation = c("rowwise", "colwise"),
  method = c("hilbert", "univariate", "sliced"),
  nsim = 1000
)
```

### Arguments

`...` Either data matrices as separate arguments or a list of data matrices. Arguments after the data must be specified by name.

`p` The power of the Wasserstein distance to use

`ground_p` The power of the Euclidean distance to use

`observation.orientation`
Are observations by rows or columns

`method` One of "hilbert", "univariate", or "sliced"

`nsim` Number of simulations to use for the sliced method

### Value

transport plan

### Examples

```
set.seed(23423)
n <- 100
d <- 10
p <- ground_p <- 2 #euclidean cost, p = 2
x <- matrix(stats::rnorm((n + 11)*d), n + 11 , d)
y <- matrix(stats::rnorm(n*d), n, d)
z <- matrix(stats::rnorm((n +455)*d), n +455, d)

# make data a list
data <- list(x,y,z)

tplan <- transport_plan_multimarg(data, p = p, ground_p = ground_p,
observation.orientation = "rowwise", method = "hilbert")

#' #transpose data works too
datat <- lapply(data, t)

tplan2 <- transport_plan_multimarg(datat, p = p, ground_p = ground_p,
observation.orientation = "colwise",method = "hilbert")
```

---

## wasserstein *Calculate the Wasserstein distance*

---

### Description

Calculate the Wasserstein distance

### Usage

```
wasserstein(
  X = NULL,
  Y = NULL,
  a = NULL,
  b = NULL,
  cost = NULL,
  tplan = NULL,
  p = 2,
  ground_p = 2,
  method = transport_options(),
  cost_a = NULL,
  cost_b = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| `X` | The covariate data of the first sample. |
| `Y` | The covariate data of the second sample. |
| `a` | Optional. Empirical measure of the first sample |
| `b` | Optional. Empirical measure of the second sample |
| `cost` | Specify the cost matrix in advance. |
| `tplan` | Give a transportation plan with slots "from", "to", and "mass", like that returned by the [tranportation_plan()] function. |
| `p` | The power of the Wasserstein distance |
| `ground_p` | The power of the Lp norm |
| `method` | Which transportation method to use. See [transport_options()] |
| `cost_a` | The cost matrix for the first sample with itself. Only used for unbiased Sinkhorn |
| `cost_b` | The cost matrix for the second sample with itself. Only used for unbiased Sinkhorn |
| `...` | Additional arguments for various methods:<br>• "niter": The number of iterations to use for the entropically penalized optimal transport distances<br>• "epsilon": The multiple of the median cost to use as a penalty in the entropically penalized optimal transport distances<br>• "unbiased": If using Sinkhorn distances, should the distance be de-biased? (TRUE/FALSE)<br>• "nboot": If using sliced Wasserstein distances, specify the number of Monte Carlo samples |

### Value

The p-Wasserstein distance, a numeric value

## Examples

```
set.seed(11289374)
n <- 100
z <- stats::rnorm(n)
w <- stats::rnorm(n)
uni <- approxOT::wasserstein(X = z, Y = w,
p = 2, ground_p = 2,
observation.orientation = "colwise",
method = "univariate")
```

# Index