

# Package ‘stream.net’

April 19, 2009

**Version** 1.0.6

**Date** 2007-10-22

**Title** Building and analyzing binary stream networks

**Author** Denis White <white.denis@epa.gov>

**Maintainer** Denis White <white.denis@epa.gov>

**Depends** R (>= 2.4)

**Description** Functions with example data for creating, importing, attributing, analyzing, and displaying stream networks represented as binary trees. Capabilities include upstream and downstream distance matrices, stochastic network generation, segmentation of network into reaches, adding attributes to reaches with specified statistical distributions, interpolating reach attributes from sparse data, analyzing autocorrelation of reach attributes, and creating maps with legends of attribute data. Target applications include dynamic fish modeling.

**License** Unlimited

**URL** <http://www.epa.gov/wed/pages/staff/white/>

**Repository** CRAN

**Date/Publication** 2007-10-27 15:32:42

## R topics documented:

marys.aat . . . . .	2
marys.elevslope . . . . .	3
marys.lin . . . . .	3
marys.pol . . . . .	4
net.addatt . . . . .	5
net.addsegs . . . . .	7
net.arcinput . . . . .	8
net.autocorr.att . . . . .	10
net.autocorr.onelag . . . . .	12
net.correlogram . . . . .	13

net.dir . . . . .	15
net.dist . . . . .	16
net.group . . . . .	17
net.interp . . . . .	18
net.lengths . . . . .	20
net.map . . . . .	21
net.map.key . . . . .	22
net.maxupslope . . . . .	24
net.object . . . . .	25
net.orders . . . . .	26
net.prox . . . . .	27
net.qmodel . . . . .	28
net.read . . . . .	29
net.shortlengths . . . . .	30
net.total.dist . . . . .	31
net.write . . . . .	32
read.arcgenlin . . . . .	33

## Index 35

---

marys.aat	<i>Topology Data for Marys River, Oregon</i>
-----------	--

---

### Description

The "from" and "to" pointers for each arc in the Marys River Arc/Info coverage. The Marys River drains an 800 square kilometer watershed in western Oregon.

### Usage

marys.aat

### Format

This is an extraction from the Arc Attribute Table (AAT) of the coverage of the Marys River derived from the source below. The format is comma separated values (.csv) and the three columns are, as stipulated in [net.arcinput](#):

arcid	integer Arc/Info User-Id for each arc (segment)
from	pointer to the preceding arc (either up or downstream)
to	pointer to the succeeding arc (likewise)

### Source

US Geological Survey 1:100,000 scale Digital Line Graph

### See Also

[net.arcinput marys.lin](#)

---

marys.elevslope      *Elevation and Slope Data for Marys River, Oregon*

---

### Description

The elevation and slope values for each segment (reach) in the Marys River stream network.

### Usage

marys.elevslope

### Format

These data are derived from digital elevation models to describe topographic characteristics of the Marys River stream network:

arcid	integer Arc/Info User-Id for each arc (segment)
elev	elevation in meters
slope	slope in percent

### Source

Derived from 30 meter and/or 10 meter Digital Elevation Models

---

marys.lin      *Line Coordinate Data for Marys River, Oregon*

---

### Description

The projected map coordinates for the Arc/Info coverage of the Marys River. The Marys River drains an 800 square kilometer watershed in western Oregon.

### Usage

marys.lin

### Format

These data are derived from a "generate" format representation of the coordinates of the arcs in the coverage of the Marys River, that are in turn derived from the source below. The format is modified S/R lines in a table (.tab) and the two columns are, as stipulated in [net.arcinput](#):

x	x-coordinate
y	y-coordinate

Modified S/R format lines consist of two columns, the "x" coordinate and the "y" coordinate. Prior to the first coordinate pair of each line is a row containing the arcid in the "x" column and NA in the "y" column.

The map projection for these coordinates is Universal Transverse Mercator, Zone 10 using the Clarke 1866 Spheroid.

### Source

US Geological Survey 1:100,000 scale Digital Line Graph

### See Also

[read.arcgenlin net.arcinput marys.aat](#)

---

marys.pol

*Boundary Coordinate Data for Marys River Watershed, Oregon*

---

### Description

The projected map coordinates for the Arc/Info coverage of the boundary of the Marys River Watershed. The Marys River drains an 800 square kilometer watershed in western Oregon.

### Usage

marys.pol

### Format

These data are derived from a "generate" format representation of the coordinates of the single arc in the coverage of the boundary of the watershed of the Marys River. The format is modified S/R lines in a table (.tab) and the two columns are, as stipulated in [net.arcinput](#):

x	x-coordinate
y	y-coordinate

Modified S/R format lines consist of two columns, the "x" coordinate and the "y" coordinate. Prior to the first coordinate pair of each line is a row containing the line or polygon identifier in the "x" column and NA in the "y" column.

For a polygon, the first and last coordinate pairs are identical.

The map projection for these coordinates is Universal Transverse Mercator, Zone 10 using the Clarke 1866 Spheroid.

### Source

Derived from 30 meter Digital Elevation Model

---

net.addatt

---

*Add Attributes to Segments of Stream Network*


---

### Description

Add attributes to segments of a `net.object`, generated by either a functional relationship with optional noise added, or a distributional property.

### Usage

```
net.addatt (net, name=NULL, func=NULL, dist=NULL,
           ind="shreve", sd=0.02, arg1=NULL, arg2=NULL,
           boundscaling=FALSE, outscaling=FALSE, prob=0.01,
           min=0, max=1, vector=TRUE)
```

### Arguments

net	A <code>net.object</code> .
name	name of the attribute to be added (if ! <code>vector</code> ).
func	name of the functional form, see details.
dist	name of the distribution function, see details.
ind	the independent variable for functional values.
sd	standard deviation for added noise to <code>func</code> output. Used as the proportion of the range of the produced values.
arg1	first argument to distribution functions, e.g., <code>shape1</code> , <code>mean</code> , etc. See details.
arg2	second argument to distribution functions.
boundscaling	if TRUE, scale distribution values to either absolute bounds of distribution or to the <code>prob</code> and/or <code>1-prob</code> quantile; implies <code>outscaling</code> is true.
outscaling	if TRUE, scale return values to <code>[min, max]</code> .
prob	probability value for scaling distributions.
min	minimum value for scaling output values.
max	maximum value for scaling output values.
vector	if TRUE, return a vector else a field in <code>\$segs</code> .

### Details

Functional relationships create a new attribute as a function of an existing attribute. Supported functions:

- Linear
- NegLinear
- Exponential
- NegExponential

Sigmoid  
 RevSigmoid  
 Logarithmic  
 RevLogarithmic  
 Sinusoidal  
 Quadratic

Distributions create a new attribute with designated distributional properties. The assignment of distributed values is not controlled for spatial autocorrelation. Supported distributions:

Uniform	= runif (n, min=arg1, max=arg2)
Normal	= rnorm (n, mean=arg1, sd=arg2)
Ramp	= rbeta (n, shape1=arg1, shape2=arg2)
NegRamp	= rbeta (n, shape1=arg1, shape2=arg2)
Exponential	= rbeta (n, shape1=arg1, shape2=arg2)
NegExponential	= rbeta (n, shape1=arg1, shape2=arg2)
Lognormal	= rlnorm (n, meanlog=arg1, sdlog=arg2)
Weibull	= rweibull (n, shape=arg1, scale=arg2)
Unimodal	= rbeta (n, shape1=arg1, shape2=arg2)
Logistic	= rlogis (n, location=arg1, scale=arg2)

where the default values of `arg1` and `arg2` are:

Uniform	= c(0, 1)
Normal	= c(0, 1)
Ramp	= c(2, 1)
NegRamp	= c(1, 2)
Exponential	= c(100, 1)
NegExponential	= c(1, 100)
Lognormal	= c(0, 0.4)
Weibull	= c(100, 100)
Unimodal	= c(2, 2)
Logistic	= c(0, 1)

and the `Boundscaling` values are:

Uniform	= c(0, 1)
Normal	= c(qnorm (prob, mean=arg1, sd=arg2), qnorm (1-prob, mean=arg1, sd=arg2))
Ramp	= c(0, 1)
NegRamp	= c(0, 1)
Exponential	= c(0, 1)
NegExponential	= c(0, 1)
Lognormal	= c(0, qlnorm (1-prob, meanlog=arg1, sdlog=arg2))
Weibull	= c(0, qweibull (1-prob, shape=arg1, scale=arg2))
Unimodal	= c(0, 1)
Logistic	= c(qlogis (prob, location=arg1, scale=arg2), qlogis (1-prob, location=arg1, scale=arg2))

**Value**

If `vector` is `TRUE`, then a vector of the attribute values in segment order, else a `net.object` with the attribute added to `$segs`.

**Author(s)**

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

**See Also**

[net.object](#) [net.addsegs](#) [runif](#) [rnorm](#) [rlnorm](#) [rbeta](#) [rweibull](#) [rlogis](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# add attribute based on normal distribution
net <- net.addatt (net, name="normal", dist="Normal",
  boundscaling=TRUE, vector=FALSE)
classes <- net.group (net, segatt="normal", ngroups=7,
  method="equalInterval")
net.map (net, group=classes$group, lwd=2)
net.map.key (0.8, 0.1, labels=round(classes$cuts, 2),
  sep=0, head="normal", horizontal=FALSE)

# make attribute based on log of shreve order
att <- net.addatt (net, name="slope", func="Logarithmic",
  ind="shreve", outscaling=TRUE, vector=TRUE)
att <- (1 - att) * 10
classes <- net.group (net, att, ngroups=7,
  method="equalInterval")
net.map (net, group=classes$group, lwd=2)
net.map.key (0.6, 0.1, labels=round(classes$cuts, 0),
  sep=0, head="log.shreve", horizontal=TRUE)
```

---

net.addsegs

*Add Segments to Links of Stream Network*

---

**Description**

Add segments to a `net.object`, according to either a uniform or a gamma distribution.

**Usage**

```
net.addsegs (net, dist="Uniform", min=1, max=2,
  shape=1.35, scale=2.3)
```

## Arguments

net	A <code>net.object</code> .
dist	If "Uniform", use uniform distribution for number of segments to add, else if "Gamma", use gamma distribution.
min	min argument of <code>runif</code> .
max	max argument of <code>runif</code> .
shape	shape argument of <code>rgamma</code> .
scale	scale argument of <code>rgamma</code> .

## Details

Add segments to a (presumably random) network. Number of segments to add per link is either a random draw from a uniform distribution between `min` and `max`, inclusive, or a random draw from a gamma distribution with parameters `shape` and `scale`. Does not handle segment attributes, so should be used before attributes are added. The number of segments added for any particular link can be zero.

Primarily designed to be used with `net.qmodel`.

## Value

A `net.object` with segments added.

## Author(s)

Denis White, (white.denis@epa.gov)

## See Also

`net.object` `net.qmodel` `net.addatt` `runif` `rgamma`

## Examples

```
net <- net.qmodel (10)
net <- net.addsegs (net)
net.map (net, segatt="sid")
```

---

net.arcinput

*Read ESRI-Format Stream Networks*

---

## Description

Read in a stream network from ESRI Arc/Info Arc Attribute Table (AAT) data and from Arc/Info "generate" format lines.

**Usage**

```
net.arcinput (aatname, linname, lineformat="R")
```

**Arguments**

aatname	name of AAT object; if mode(aatname)=="character", then character string of the AAT file name, else R object with AAT data.
linname	name of lines object; if mode(linname)=="character", then character string of the lines file name, else R object with lines data.
lineformat	"arcgen" for Arc/Info generate format lines; "R" for S/R format lines; only applicable to file input.

**Details**

Arc/Info input consists of two objects, either from files in text format, or existing R objects.

First, information from the coverage AAT, unloaded from the tables command, and containing at least the user-id for each arc and the "from" and "to" node numbers. Any number of arc attributes can follow. After unloading, the file should be edited so that the first record, the "header", has the names of the fields in the file, in order, separated by commas. The user-id is expected to have the name "arcid", the "from" node "from", and the "to" node "to". Otherwise the names can be any legal R names (see the R reference manual).

Second, the arc coordinates from an ungenerate command. The ungenerate command should use the "fixed" option (and the "line" option, of course). If the lines have already been converted to (modified) S/R format, then argument "lineformat" should be set to "R". The arcids in the line file have to correspond with the arcids in the AAT file, of course.

Modified S/R format lines consist of two columns, the "x" coordinate and the "y" coordinate. Prior to the first coordinate pair of each line is a row containing the arcid in the "x" column and NA in the "y" column.

It is essential that each arc have a unique user-id ("arcid"), that there be no nodes of degree greater than 2, that there be no cycles in the network (loops), and that the topology be clean for this process to work correctly.

Arcs (the units in both the AAT file and the lines file) become segments in package 'stream.net' and links are determined by those arc junctions (nodes) that are of degree greater than two.

Up or downstream order for arcs is determined by finding the mouth as the only singleton "from" arc (up order), or only singleton "to" arc (down order).

**Value**

A [net.object](#).

**Author(s)**

Denis White, <[white.denis@epa.gov](mailto:white.denis@epa.gov)>

**See Also**

[net.object read.arcgenlin](#)

## Examples

```
data (marys.aat, marys.lin, marys.pol, marys.elevslope)

net <- net.arcinput (marys.aat, marys.lin)
net.map (net, outline=marys.pol, col="blue")
hist (marys.elevslope$elev, col="gray")
hist (marys.elevslope$slope, col="gray")
classes <- net.group (net, marys.elevslope$elev, ngroups=7,
  method="equalInterval")
net.map (net, group=classes$group, lwd=2, outline=marys.pol)
net.map.key (0.8, 0.1, labels=round(classes$cuts, 0),
  sep=0, head="elev", horizontal=FALSE)
classes <- net.group (net, marys.elevslope$slope, ngroups=7,
  method="equalInterval")
net.map (net, group=classes$group, lwd=2, outline=marys.pol)
net.map.key (0.8, 0.1, labels=round(classes$cuts, 2),
  sep=0, head="slope", horizontal=FALSE)
```

---

net.autocorr.att    *Create Autocorrelated Attribute for Stream Segments*

---

## Description

Create a network autocorrelated attribute for stream segments.

## Usage

```
net.autocorr.att (net, dist, target=0.5, lag=1,
  outscaling=TRUE, min=0, max=1, eps=1e-6,
  vector=TRUE, name=NULL)
```

## Arguments

net	A <a href="#">net.object</a> .
dist	an upstream/downstream distance matrix from <a href="#">net.dist</a> , probably using <code>ends=0.5</code> , and possibly using <code>method="segment"</code> .
target	goal for autocorrelation.
lag	separation distance (in method units) at which to produce autocorrelation effect.
outscaling	if TRUE, scale return values to <code>[min, max]</code> .
min	minimum value for scaling output values.
max	maximum value for scaling output values.
eps	precision of calculating lag distance.
vector	if TRUE, return a vector else a field in <code>\$segs</code> .
name	name of the attribute to be added (if ! <code>vector</code> ).

## Details

Uses total distances, ignoring upstream/downstream. The algorithm for calculating autocorrelation, from the reference below, is

$$r(x) = \text{sum}(2 * z[i] * z[j]) / \text{sum}(z[i]^2 + z[j]^2)$$

for all segment pairs  $i, j$  at lag  $x$ . Values are in  $[-1, 1]$ .

## Value

If `vector` is TRUE, then a vector of the attribute values in segment order, else a `net.object` with the attribute added to `$segs`.

## Author(s)

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

## References

Henley S. 1975. Autocorrelation coefficients from irregularly spaced areal data. *Computers and Geosciences* 2(4):437-438.

## See Also

[net.object](#) [net.dist](#)

## Examples

```
# Q model random net
net <- net.qmodel (5)
net <- net.addsegs (net)
dmat <- net.dist (net, ends=0.5, method="segment")

# make autocorrelated attributes
att.autoc <- net.autocorr.att (net, dist=dmat, vector=TRUE,
  outscaling=TRUE) * 10

# check autocorrelation
net.autocorr.onelag (net, dist=dmat, segatt=att.autoc)
```

---

```
net.autocorr.onelag
```

*Calculate Autocorrelation Coefficient for Stream Segment Attribute*

---

### Description

Compute the autocorrelation coefficient of an attribute for a specified separation distance (lag) between stream segments.

### Usage

```
net.autocorr.onelag (net, dist, segatt, lag=1, eps=1e-6)
```

### Arguments

net	A <code>net.object</code> .
dist	an upstream/downstream distance matrix from <code>net.dist</code> , probably using <code>ends=0.5</code> , and possibly using <code>method="segment"</code> .
segatt	attribute associated with each segment. If <code>segatt</code> is a character vector of length one, then get attribute from <code>net\$segs\$segatt</code> , else assume <code>segatt</code> is a numeric vector of length equal to number of segments and in correct order.
lag	separation distance (in <code>method</code> units) at which to calculate coefficient.
eps	precision of calculating lag.

### Details

Uses total distances, ignoring upstream/downstream. The algorithm for autocorrelation coefficients, from the reference below, is

$$r(x) = \text{sum}(2 * z[i] * z[j]) / \text{sum}(z[i]^2 + z[j]^2)$$

for all segment pairs  $i, j$  at lag  $x$ . Values are in  $[-1, 1]$ .

### Value

A list with the following components:

r	autocorrelation coefficient
num	number of pairs used

### Author(s)

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

## References

Henley S. 1975. Autocorrelation coefficients from irregularly spaced areal data. *Computers and Geosciences* 2(4):437-438.

## See Also

[net.object](#) [net.dist](#) [net.autocorr.att](#) [net.correlogram](#)

## Examples

```
# Q model random net
net <- net.qmodel (5)
net <- net.addsegs (net)
dmat <- net.dist (net, ends=0.5, method="segment")

# make rnorm and autocorrelated attributes
att.rnorm <- net.addatt (net, dist="Normal",
  boundscaling=TRUE, vector=TRUE)
att.autoc <- net.autocorr.att (net, dist=dmat, vector=TRUE,
  outscaling=TRUE) * 10

# check autocorrelation
net.autocorr.onelag (net, dist=dmat, segatt=att.rnorm)
net.autocorr.onelag (net, dist=dmat, segatt=att.autoc)
```

---

`net.correlogram`      *Calculate Correlogram for Stream Segment Attribute*

---

## Description

Compute the autocorrelation coefficients of an attribute for a sequence of separations in distance (lags) between stream segments.

## Usage

```
net.correlogram (net, dist, segatt, nlags=10)
```

## Arguments

<code>net</code>	A <a href="#">net.object</a> .
<code>dist</code>	an upstream/downstream distance matrix from <a href="#">net.dist</a> , probably using <code>ends=0.5</code> .
<code>segatt</code>	attribute associated with each segment. If <code>segatt</code> is a character vector of length one, then get attribute from <code>net\$segs\$segatt</code> , else assume <code>segatt</code> is a numeric vector of length equal to number of segments and in correct order.
<code>nlags</code>	number of lags in the output function.

**Details**

Uses total distances, ignoring upstream/downstream. The algorithm for autocorrelation coefficients, from the reference below, is

$$r(x) = \text{sum}(2 * z[i] * z[j]) / \text{sum}(z[i]^2 + z[j]^2)$$

for all segment pairs  $i, j$  at lag  $x$ . Values are in  $[-1, 1]$ .

Lag distances are in method units used in `net.dist`, and are in range  $[0, \text{max}(\text{dist})]$ .

**Value**

A list with the following elements:

r	vector of autocorrelation coefficients at each lag
lag	vector of lag distances
num	vector of number of pairs at each lag

**Author(s)**

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

**References**

Henley S. 1975. Autocorrelation coefficients from irregularly spaced areal data. *Computers and Geosciences* 2(4):437-438.

**See Also**

[net.object](#) [net.dist](#) [net.autocorr.att](#) [net.autocorr.onelag](#)

**Examples**

```
# Q model random net
net <- net.qmodel (5)
net <- net.addsegs (net)
dmat <- net.dist (net, ends=0.5, method="segment")

# make rnorm and autocorrelated attributes
att.rnorm <- net.addatt (net, dist="Normal",
  boundscaling=TRUE, vector=TRUE)
att.autoc <- net.autocorr.att (net, dist=dmat, vector=TRUE,
  outscaling=TRUE) * 10

# plot correlograms
cg1 <- net.correlogram (net, dist=dmat, segatt=att.rnorm)
cg2 <- net.correlogram (net, dist=dmat, segatt=att.autoc)
plot (cg1$lag, cg1$r, pch=19, type="b", ylim=c(-1,1),
  main="Random Normal")
plot (cg2$lag, cg2$r, pch=19, type="b", ylim=c(-1,1),
  main="Autocorrelated")
```

---

net.dir *Calculate Direction Matrix for Stream Segments*

---

**Description**

Compute the upstream/downstream/mixed direction matrix for stream segments .

**Usage**

```
net.dir (dist)
```

**Arguments**

dist                    an upstream/downstream distance matrix from [net.dist](#).

**Details**

Directions are +1 for upstream, -1 for downstream, and 0 for both. The matrix is skew-symmetric and is read from rows to columns, i.e., row 1, column 3 is +1 if segment 3 is upstream of segment 1 and -1 if downstream. Upstream and downstream are defined as being in direct linkage; any pair that have mixed upstream and downstream directions have direction 0.

**Value**

A square matrix of dimension the number of segments.

**Author(s)**

Denis White, <[white.denis@epa.gov](mailto:white.denis@epa.gov)>

**See Also**

[net.dist](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# distance matrix
dmat <- net.dist (net, ends=0.5, method="coordinate")

# directions
tdmat <- net.dir (dmat)
net.map (net, segatt="sid")
table (tdmat[5,]) # includes self in zero count
```

---

`net.dist`*Calculate Distance Matrix for Stream Segments*

---

**Description**

Compute the asymmetric distance matrix for stream segments. Upstream and downstream distances are included separately.

**Usage**

```
net.dist (net, ends=0.5, method="coordinate", digits=10)
```

**Arguments**

<code>net</code>	A <a href="#">net.object</a> .
<code>ends</code>	how to handle from and to segments if <code>ends=0</code> , do not use from and to lengths if <code>ends=1</code> , use total of from and to lengths if <code>ends=0.5</code> , use half of from and to lengths.
<code>method</code>	<code>method="coordinate"</code> means network coordinate distance; <code>method="segment"</code> means distance in number of segments.
<code>digits</code>	if <code>digits=NULL</code> , do not round output matrix, else precision of rounding.

**Details**

Upstream distances are found by indexing the source segment by its row and the destination segment by its column. Downstream distances are the opposite; the source is the column and the destination is the row.

Algorithm adapted from that of SG Leibowitz. This algorithm does produce roundoff residue and thus the matrix can be purged of very small non-zero values with rounding.

**Value**

A square matrix of dimension the number of segments.

**Author(s)**

Denis White, ([white.denis@epa.gov](mailto:white.denis@epa.gov))

**See Also**

[net.object](#) [net.total.dist](#) [net.dir](#) [net.prox](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# distance matrix
net.map (net, segatt=round (net$segs$length, 2))
dmat <- net.dist (net, ends=0.5, method="coordinate")
hist (dmat, col="gray", main="One Way Distances")
```

---

net.group

*Classify Stream Attribute for Mapping*


---

**Description**

Classify stream segment or stream link attributes for use by `net.map`.

**Usage**

```
net.group (net, segatt=NULL, linkatt=NULL,
           ngroups=5, method="quantile", spread=NULL)
```

**Arguments**

net	A <code>net.object</code> .
segatt	attribute associated with each segment. If <code>segatt</code> is a character vector of length one, then get attribute from <code>net\$segs\$segatt</code> , else assume <code>segatt</code> is a numeric vector of length equal to number of segments and in correct order.
linkatt	attribute associated with each link. If <code>linkatt</code> is a character vector of length one, then get attribute from <code>net\$links\$linkatt</code> , else assume <code>linkatt</code> is a numeric vector of length equal to number of links and in correct order.
ngroups	the number of groups to create.
method	currently must be "quantile" or "equalInterval". <code>method = "quantile"</code> means quartiles, quintiles, etc., depending on <code>ngroups</code> . <code>method = equalInterval</code> means dividing the range of the attribute into <code>ngroups</code> equal intervals.
spread	if not NULL, then, for "equalInterval" classification, set the range of intervals to be the two element spread vector.

**Details**

One of `segatt` or `linkatt` must not be NULL.

**Value**

A list with the following components:

```

group  vector of group numbers for each segment
cuts   vector of break points for the classification

```

The `group` vector is always of length the number of segments. If the attribute is provided by link, the attribute is associated with all segments in the link.

The names of the `group` vector are set to the `sids` so that `net.map` will work correctly.

### Author(s)

Denis White, (white.denis@epa.gov)

### See Also

[net.object](#) [net.map](#)

### Examples

```

# Q model random net
net <- net.qmodel (10)
net <- net.addsegs (net)

# classify some attributes
class1 <- net.group (net, linkatt="shreve",
  method="equalInterval")
class2 <- net.group (net, segatt="length")
att <- seq (nrow (net$segs))
class3 <- net.group (net, segatt=att)

# map
n <- length (table (class1$group))
net.map (net, group=class1$group, col=rev(terrain.colors(n)))
n <- length (table (class2$group))
net.map (net, group=class2$group, col=gray(0:n/n))
n <- length (table (class3$group))
net.map (net, group=class3$group, col=rainbow(n))

```

---

net.interp

*Interpolate Sparse Attribute to all Stream Segments*

---

### Description

Create an attribute for stream segments that is interpolated from data provided for a subset of segments.

### Usage

```

net.interp (net, dist, samples, predict=NULL,
  maxdist=1e32, method="inverseDistance", power=2,
  vector=TRUE, name=NULL)

```

**Arguments**

net	A <code>net.object</code> .
dist	an upstream/downstream distance matrix from <code>net.dist</code> , probably using <code>ends=0.5</code> , and possibly using <code>method="segment"</code> .
samples	two column matrix or data frame with first column/field <code>sids</code> to segments of sampled data and second column/field the sampled data values.
predict	vector of segments to which to predict, unless <code>NULL</code> , in which case predict to all other segments.
maxdist	maximum distance (in method units) for neighborhood.
method	only "inverseDistance" currently implemented.
power	exponent for <code>method="inverse distance"</code> function.
vector	if <code>TRUE</code> , return a vector else a field in <code>\$segs</code> .
name	name of the attribute to be added (if ! <code>vector</code> ).

**Details**

Uses total distances, ignoring upstream/downstream.

**Value**

If `vector` is `TRUE`, then a vector of the attribute values in segment order, else a `net.object` with the attribute added to `$segs`.

**Author(s)**

Denis White, (white.denis@epa.gov)

**See Also**

`net.object` `net.dist`

**Examples**

```
# Q model random net
net <- net.qmodel (10)
dmat <- net.dist (net, ends=0.5, method="segment")

# interpolation
samples <- matrix (c(2, 4, 6, 10, 5, 1), ncol=2)
y <- net.interp (net, dmat, samples, power=0.25)
net.map (net, segatt=round (y, 1))
```

---

`net.lengths`*Calculate Stream Segment Lengths*

---

### Description

Calculate lengths of segments for a `net.object` using distance between all coordinate points along segment.

### Usage

```
net.lengths (net)
```

### Arguments

`net` A `net.object`.

### Details

Intended primarily for internal use. A `net.object` created by `net.arcinput` or `net.qmodel` includes the segment lengths. Length is the sum of linear distances between segment coordinates.

### Value

A vector, in segment order, of lengths.

### Author(s)

Denis White, <white.denis@epa.gov>

### See Also

`net.object`

### Examples

```
net <- net.qmodel (10)
net.map (net, segatt="length")
```

---

net.map

*Draw Map of Stream Network Attribute*


---

### Description

Draw map of stream network by symbolizing segments according to a classification of segment or link attributes obtained from `net.group`, using colors or line widths, or just according to a set of colors or line widths directly associated with the segments.

### Usage

```
net.map (net, group=NULL, linkatt=NULL, segatt=NULL,
        col=NULL, lwd=NULL, cex=par("cex"), new=TRUE,
        outline=NULL, uniquegroup=FALSE)
```

### Arguments

net	A <code>net.object</code> .
group	if not NULL, then a vector of color codes for each segment. <code>names(group)</code> must match <code>sids</code> .
segatt	if not NULL then write the attribute value associated with each segment in text format at the midpoint of the segment. If <code>segatt</code> is a character vector of length one, then get attribute from <code>net\$segs\$segatt</code> , else assume <code>segatt</code> is a numeric vector of length equal to number of segments and in correct order.
linkatt	if not NULL then write the attribute value associated with each link in text format at the midpoint of the link. If <code>linkatt</code> is a character vector of length one, then get attribute from <code>net\$links\$linkatt</code> , else assume <code>linkatt</code> is a numeric vector of length equal to number of links and in correct order.
col	<code>par</code> parameter for color codes for segments. Default ramp is yellow-red-brown.
lwd	<code>par</code> parameter for line widths of segments. If <code>lwd="seq"</code> , generate widths in sequence <code>1:(number of groups)</code> .
cex	<code>par</code> parameter for size of text.
new	if TRUE, then create a new plot.
outline	if not NULL, then shade an outline polygon in light gray first before drawing network; outline format is S/R polygon format.
uniquegroup	if TRUE, group categories are not members of the integers <code>1:(number of groups)</code> , assuming <code>! is.null (group)</code> .

### Details

The mapping is by segment. Link attributes can be mapped by grouping them with `net.group` where they will be "stretched" onto the segments.

Text format labeling of attributes for segments or links is separate from (and can be in addition to) color or line width symbolism of attributes.

The names of the group vector must match the sids (as set, for example, in `net.group`).

The values of group are assumed to be integers. If these values are not dense (continuous in 1:(number of groups), then set `uniquegroups=TRUE`.

### Value

If `! is.null (group)` then the colors provided or generated else nothing in `invisible`.

### Author(s)

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

### See Also

[net.object](#) [net.group](#)

### Examples

```
# Q model random net
net <- net.qmodel (10)
net <- net.addsegs (net)

# classify some attributes
class1 <- net.group (net, linkatt="shreve",
  method="equalInterval")
class2 <- net.group (net, segatt="length")
att <- seq (nrow (net$segs))
class3 <- net.group (net, segatt=att)

# map
net.map (net) # just draw segments in black
n <- length (table (class1$group))
net.map (net, group=class1$group, col="black", lwd="seq")
n <- length (table (class2$group))
net.map (net, group=class2$group, col=gray(0:n/n),
  linkatt="strahler")
n <- length (table (class3$group))
net.map (net, group=class3$group, col=rainbow(n))
```

---

net.map.key

*Draw Legend for a Map of a Stream Attribute*

---

### Description

Draw a legend to accompany a map of a stream attribute.

**Usage**

```
net.map.key (x, y, labels=NULL, cex=par("cex"),
            pch=par("pch"), size=2.5*cex, col=NULL, head="",
            sep=0.25*cex, horizontal=FALSE, new=FALSE)
```

**Arguments**

<code>x, y</code>	lower left coordinates of key in proportional units [0, 1].
<code>labels</code>	vector of labels for classes if <code>NULL</code> , use integers <code>1:length(col)</code> , except if <code>is.null(col)</code> then use "1".
<code>cex</code>	<code>par</code> parameter for size of text.
<code>pch</code>	<code>par</code> parameter for type of symbols.
<code>size</code>	size of key symbols in <code>cex</code> units.
<code>col</code>	<code>par</code> parameter for color codes. Default ramp is yellow-red-brown.
<code>head</code>	text heading for key.
<code>sep</code>	separation in <code>cex</code> units between adjacent symbols. If <code>sep=0</code> , assume continuous scale and use <code>pch=15</code> , and put labels at breaks between squares.
<code>new</code>	if <code>TRUE</code> , create a new plot.
<code>horizontal</code>	if <code>TRUE</code> , key runs horizontal.

**Details**

Alternative to [legend](#).

**Value**

A vector of the colors provided or generated returned as [invisible](#).

**Author(s)**

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

**See Also**

[net.object](#) [net.group](#)

**Examples**

```
data (marys.aat, marys.lin, marys.pol, marys.elevslope)

net <- net.arcinput (marys.aat, marys.lin)
net.map (net, outline=marys.pol, col="blue")
hist (marys.elevslope$elev, col="gray")
hist (marys.elevslope$slope, col="gray")
classes <- net.group (net, marys.elevslope$elev, ngroups=7,
                    method="equalInterval")
net.map (net, group=classes$group, lwd=2, outline=marys.pol)
```

```
net.map.key (0.8, 0.1, labels=round(classes$cuts, 0),
  sep=0, head="elev", horizontal=FALSE)
classes <- net.group (net, marys.elevslope$slope, ngroups=7,
  method="equalInterval")
net.map (net, group=classes$group, lwd=2, outline=marys.pol)
net.map.key (0.8, 0.1, labels=round(classes$cuts, 2),
  sep=0, head="slope", horizontal=FALSE)
```

---

net.maxupslope      *Calculate Maximum Upslope Matrix for Stream Segments*

---

## Description

Compute the asymmetric maximum upslope matrix for stream segments.

## Usage

```
net.maxupslope (net, slopes)
```

## Arguments

net	A <a href="#">net.object</a> .
slopes	vector of "slopes" in segment order.

## Details

Uses the distance matrix algorithm to calculate the maximum upstream slope between two segments. The maximum is the maximum of the slopes of all segments between the two segments, not including either of them. The matrix is read from row to column, i.e., the maximum slope from segment 1 to segment 3 is in row 1, column 3.

The function was written for application to topographic slope but any other attribute could be used of course.

## Value

A square matrix of dimension the number of segments.

## Author(s)

Denis White, <white.denis@epa.gov>

## See Also

[net.dist](#)

**Examples**

```

# Q model random net
net <- net.qmodel (5)

# add segments
net <- net.addsegs (net)

# upslope distance matrix
slopes <- max (net$links$shreve) -
  net$links$shreve[net$segs$link]
slopes <- sapply (slopes + round (runif (nrow (net$segs)), 2),
  function (x) max (0, x))
net.map (net, segatt=slopes)
slopmat <- net.maxupslope (net, slopes=slopes)

```

---

net.object

*Data Structure for Stream Network Object*


---

**Description**

Components of the `net.object`.

**Format**

The network data structure, as returned from `net.qmodel` and `net.arcinput`, is a list with three components:

links	a data frame of the topological structure
segs	a data frame of the segment structure (pieces of links)
coords	a list of the x and y coordinates for each segment

links has the components

lid	identifier for the link
parent	index of parent link (or 0 if root)
left	index of left child (or 0 if terminal)
right	index of right child (or 0 if terminal)
depth	topological depth of link (number of links from root)
first	index of first segment of this link
last	index of last segment of this link
strahler	Strahler order
shreve	Shreve order

segs has the components

sid	identifier for the segment (for external ref)
link	index to parent link

nxt      index of the next segment for this link  
 prev     index of the previous segment for this link  
 up       if segment (and coordinates) oriented upstream = 1, else = 0  
 length   segment coordinate length

cords has the components

x    x coordinates with a sublist for each segment  
 y    y coordinates with a sublist for each segment

(The names for the x and y sublists are the respective `sids`.)

### Author(s)

Denis White, <white.denis@epa.gov>

### See Also

[net.arcinput](#) [net.qmodel](#)

### Examples

```
net <- net.qmodel (5)
net
```

---

`net.orders`

*Calculate Strahler and Shreve Stream Orders*

---

### Description

Calculate Strahler and Shreve orders for a `net.object` and add to link table.

### Usage

```
net.orders (links)
```

### Arguments

`links`            link table, see [net.object](#).

### Details

Intended primarily for internal use. A `net.object` created by [net.arcinput](#) or [net.qmodel](#) includes the stream orders.

### Value

A link table.

**Author(s)**

Denis White, (white.denis@epa.gov)

**See Also**

[net.object](#)

**Examples**

```
net <- net.qmodel (10)
net.map (net, linkatt="strahler")
net.map (net, linkatt="shreve")
```

---

net.prox

*Calculate Proximity Relation for Stream Segments*

---

**Description**

Calculate the segments that are in a specified neighborhood of a given segment.

**Usage**

```
net.prox (dist, seg, lag=1, direction="both")
```

**Arguments**

dist	an upstream/downstream distance matrix from <a href="#">net.dist</a> .
seg	index to a segment (row number in segment table). Not necessarily the segment identifier ( <i>sid</i> ), except that in networks generated by <a href="#">net.qmodel</a> the index and the <i>sid</i> are always identical.
lag	distance within which neighborhood is computed, either in coordinates or segments (depending on how <i>dist</i> was constructed).
direction	"up", "down", or "both".

**Details**

This returns a vector of segments in proximity of *seg* in the direction "up", "down", or "both". Siblings that are in a combination of upstream and downstream directions are not included in "up" or "down". The distance matrix determines whether the neighborhood is defined by number of segments or by coordinates.

**Value**

A vector of segment indices.

**Author(s)**

Denis White, (white.denis@epa.gov)

**See Also**

[net.total.dist](#) [net.dir](#) [net.prox](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# distance matrix
net.map (net, segatt=round (net$segs$length, 2))
dmat <- net.dist (net, ends=0.5, method="coordinate")

# proximities
net.prox (dmat, seg=2, lag=3)
net.prox (dmat, seg=2, lag=3, direction="up")
```

---

net.qmodel

*Generate Stochastic Stream Network Object*

---

**Description**

Generate a stochastic network according to the Q model (see references).

**Usage**

```
net.qmodel (size=100, Q=0.5)
```

**Arguments**

size	magnitude of the network in number of terminal branches.
Q	probability of an internal branch rather than a terminal branch ( $0 \leq Q \leq 1$ ).

**Details**

Any value of  $Q$  will generate all possible topologies of the given size, however, the likelihood of any topology is controlled by the value provided. If  $Q = 0.01$ , then networks with short lengths to terminal branches are more likely (because branching from existing terminal branches is much more probable than from existing internal branches). If  $Q = 0.99$ , then networks with a few long sequences of branches are more likely.

**Value**

A `net.object`.

**Author(s)**

Denis White, [white.denis@epa.gov](mailto:white.denis@epa.gov)

**References**

Costa-Cabral MC, Burges SJ. 1997. Sensitivity of channel network planform laws and the question of topologic randomness. *Water Resources Research* **33**(9):2179-2197.

**See Also**

`net.object`

**Examples**

```
net <- net.qmodel (10, Q=0.99)
net.map (net)
```

---

`net.read`*Read In a Stream Network Object*

---

**Description**

Read in a stream network object from the tables type, text format files written by `net.write`.

**Usage**

```
net.read (prefix="somenet")
```

**Arguments**

`prefix` character string for the prefix of the input file name. See details.

**Details**

An alternative to `load` or `source` that reads table format text files.

Read in the `$links`, `$segs`, and `$cords` of a `net.object` from three separate files that were written by `net.write`. The prefix is pasted onto suffixes `".links.dat"`, `".segs.dat"`, and `".cords.dat"`, respectively, to make full filenames.

Stream networks created from either `net.qmodel` or `net.arcinput` can be retrieved much more quickly using this method.

**Value**

A `net.object`.

**Author(s)**

Denis White, <white.denis@epa.gov>

**See Also**

[net.object](#) [net.write](#) [net.qmodel](#) [net.arcinput](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# save
net.write (net, prefix="test")

# retrieve
net <- net.read (prefix="test")
```

---

net.shortlengths    *Calculate Stream Segment Endpoint-only Lengths*

---

**Description**

Calculate lengths of segments for a [net.object](#) using distance between endpoints.

**Usage**

```
net.shortlengths (net)
```

**Arguments**

net                    A [net.object](#).

**Details**

Intended primarily for internal use. The shortlength is the distances between segment endpoint coordinates only, ignoring any intermediate points.

**Value**

A vector, in segment order, of shortlengths.

**Author(s)**

Denis White, <white.denis@epa.gov>

**See Also**[net.lengths](#)**Examples**

```
data (marys.aat, marys.lin)

net <- net.arcinput (marys.aat, marys.lin)
summary (net.lengths (net) - net.shortlengths (net))
```

---

net.total.dist      *Calculate Distance Matrix for Stream Segments*

---

**Description**

Compute the symmetric total distance matrix for stream segments as the sum of upstream and downstream distances.

**Usage**

```
net.total.dist (dist)
```

**Arguments**

dist                  an upstream/downstream distance matrix from [net.dist](#).

**Value**

A square matrix of dimension the number of segments.

**Author(s)**

Denis White, <[white.denis@epa.gov](mailto:white.denis@epa.gov)>

**See Also**

[net.dist](#) [net.dir](#) [net.prox](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# distance matrix
net.map (net, segatt=round (net$segs$length, 2))
dmat <- net.dist (net, ends=0.5, method="coordinate")
```

```
# total distances
tdmat <- net.total.dist (dmat)
hist (tdmat, col="gray", main="Total Distances")
```

---

`net.write`*Write Out a Stream Network Object*

---

## Description

Write out a stream network object to a set of tables type, text format files.

## Usage

```
net.write (net, prefix="somenet")
```

## Arguments

<code>net</code>	A <a href="#">net.object</a> .
<code>prefix</code>	character string for the prefix of the output file name. See details.

## Details

An alternative to [save](#) or [dump](#) that writes table format text files.

Writes out the `$links`, `$segs`, and `$cords` of the [net.object](#) in R table format to three separate files. The prefix is pasted onto suffixes `".links.dat"`, `".segs.dat"`, and `".cords.dat"`, respectively, to make full filenames.

Stream networks created from either [net.qmodel](#) or [net.arcinput](#) can be saved this way for quicker retrieval for subsequent use.

## Value

[invisible](#).

## Author(s)

Denis White, (white.denis@epa.gov)

## See Also

[net.object](#) [net.read](#) [net.qmodel](#) [net.arcinput](#)

**Examples**

```
# Q model random net
net <- net.qmodel (10)

# add segments
net <- net.addsegs (net)

# save
net.write (net, prefix="test")
```

---

read.arcgenlin	<i>Read In an Arc/Info Generate Format Lines File</i>
----------------	---

---

**Description**

Read in an Arc/Info generate format lines file and convert to modified S/R lines format.

**Usage**

```
read.arcgenlin (filename, coord=c("x", "y"))
```

**Arguments**

filename	character string for the input file name.
coord	the names to be given to the columns for the coordinates. For stream.net, use the default.

**Details**

Any Arc/Info generate format line data can be converted for plotting or other uses in R. In particular, [net.arcinput](#) uses Arc/Info generate format lines to represent the coordinate structure of a stream network.

Modified S/R format lines consist of two columns, the "x" coordinate and the "y" coordinate. Prior to the first coordinate pair of each line is a row containing the arcid in the "x" column and NA in the "y" column.

**Value**

A [net.object](#).

**Author(s)**

Denis White, <white.denis@epa.gov>

**See Also**

[net.object](#) [net.arcinput](#)

**Examples**

```
# No examples provided as there appears to be no way to store  
# the Arc/Info format data directly in R.
```

# Index

## \*Topic **IO**

- net.arcinput, 8
- net.read, 29
- net.write, 32
- read.arcgenlin, 33

## \*Topic **datagen**

- net.addatt, 4
- net.addsegs, 7
- net.qmodel, 28

## \*Topic **datasets**

- marys.aat, 2
- marys.elevslope, 2
- marys.lin, 3
- marys.pol, 4

## \*Topic **data**

- net.object, 24

## \*Topic **hplot**

- net.map, 20
- net.map.key, 22

## \*Topic **manip**

- net.addatt, 4
- net.addsegs, 7
- net.arcinput, 8
- net.autocorr.att, 10
- net.autocorr.onelag, 11
- net.correlogram, 13
- net.dir, 14
- net.dist, 15
- net.group, 17
- net.interp, 18
- net.lengths, 19
- net.maxupslope, 23
- net.orders, 26
- net.prox, 27
- net.read, 29
- net.shortlengths, 30
- net.total.dist, 31
- net.write, 32
- read.arcgenlin, 33

- dump, 32

- invisible, 21, 23, 32

- legend, 22

- load, 29

- marys.aat, 2, 3
- marys.elevslope, 2
- marys.lin, 2, 3
- marys.pol, 4

- net.addatt, 4, 8

- net.addsegs, 6, 7

- net.arcinput, 2–4, 8, 20, 25, 26, 29, 32, 33

- net.autocorr.att, 10, 12, 14

- net.autocorr.onelag, 11, 14

- net.correlogram, 12, 13

- net.dir, 14, 16, 27, 31

- net.dist, 10–14, 15, 15, 18, 19, 24, 27, 31

- net.group, 17, 20, 21, 23

- net.interp, 18

- net.lengths, 19, 30

- net.map, 17, 18, 20

- net.map.key, 22

- net.maxupslope, 23

- net.object, 4–14, 16–21, 23, 24, 24, 26, 28–30, 32, 33

- net.orders, 26

- net.prox, 16, 27, 27, 31

- net.qmodel, 8, 20, 25–27, 28, 29, 32

- net.read, 29, 32

- net.shortlengths, 30

- net.total.dist, 16, 27, 31

- net.write, 29, 32

- par, 21, 22

- rbeta, 6

- read.arcgenlin, 3, 9, 33

rgamma, 7, 8  
rlnorm, 6  
rlogis, 6  
rnorm, 6  
runif, 6–8  
rweibull, 6  
  
save, 32  
source, 29