

# Package: shapefiles (via r-universe)

August 23, 2024

**Version** 0.7.2

**Date** 2013-01-25

**Title** Read and Write ESRI Shapefiles

**Author** Ben Stabler <benstabler@yahoo.com>

**Maintainer** Ben Stabler <benstabler@yahoo.com>

**Depends** R (>= 1.6.0), foreign

**Description** Functions to read and write ESRI shapefiles.

**License** GPL

**Date/Publication** 2022-08-25 12:46:08 UTC

**NeedsCompilation** no

**Repository** <https://bstabler.r-universe.dev>

**RemoteUrl** <https://github.com/cran/shapefiles>

**RemoteRef** HEAD

**RemoteSha** e6cbde3f12087957b739d273ed16a0c177959e40

## Contents

|                      |          |
|----------------------|----------|
| shapefiles . . . . . | 1        |
| <b>Index</b>         | <b>6</b> |

---

|            |                                       |
|------------|---------------------------------------|
| shapefiles | <i>Read and write ESRI shapefiles</i> |
|------------|---------------------------------------|

---

## Description

This package includes functions to read and write ESRI shapefiles.

**Usage**

```

read.shapefile(shape.name)
read.shp(shp.name)
read.shx(shx.name)
read.dbf(dbf.name, header=FALSE)
write.shapefile(shapefile, out.name, arcgis=FALSE)
write.shp(shp, out.name)
write.shx(shx, out.name)
write.dbf(dbf, out.name, arcgis=FALSE)
calc.header(shapefile)
add.xy(shapefile)
scaleXY(shapefile, scale.factor)
convert.to.shapefile(shpTable, attTable, field, type)
convert.to.simple(shp)
change.id(shpTable, newFieldAsVector)
dp(points, tolerance)

```

**Arguments**

|                  |  |
|------------------|--|
| shape.name       | String of the shapefile file name without an extension           |
| shp.name         | String of the shp file name with an extension                    |
| shx.name         | String of the shx file name with an extension                    |
| dbf.name         | String of the dbf file name with an extension                    |
| shapefile        | The shapefile object of lists created by read.shapefile          |
| out.name         | Filename to write the data to                                    |
| shp              | shp portion (list) of the shapefile object of lists              |
| shx              | shx portion (list) of the shapefile object of lists              |
| dbf              | dbf portion (list) of the shapefile object of lists              |
| scale.factor     | Number to divide the shapefile geography by                      |
| arcgis           | Replace "." with "_" in column names for ArcGIS                  |
| shpTable         | data.frame with columns in order Id, X, and Y                    |
| attTable         | data.frame with first column names "Id" - polygon id (key)       |
| type             | ESRI Shape type 1=point, 3=polyLine, 5=polygon                   |
| field            | A field name in the attTable                                     |
| newFieldAsVector | A vector of Ids to replace to the Ids in the shpTable            |
| points           | A named list of two vectors (x and y) representing points        |
| tolerance        | A tolerance setting for the DP polyLine simplification algorithm |
| header           | Should read.dbf return the header?                               |

## Details

ESRI shapefiles consist of three files. The first file (\*.shp) contains the geography of each shape. The second file (\*.shx) is an index file which contains record offsets. The third file (\*.dbf) contains feature attributes with one record per feature.

`read.shapefile` calls `read.shp`, `read.shx`, and `read.dbf` to read in an entire shapefile. The result of `read.shapefile` is a list of many more lists. The sublists are `shp`, `shx`, and `dbf`. Each sublist contains a header list and some sort of data list. The `shp` list is a list of `$shp$header` and `$shp$shp`. The `shx` list is a list of `$shx$header` and `$shx$index`. The `dbf` list is a list of `$dbf$header` and `$dbf$dbf`.

The write functions write out a `shp`, `shx`, and `dbf` file from the shapefile list structure. To write out a shapefile from simple R data, you need to run `convert.to.shapefile`. The inputs to this function are a simple data frame of points (for points, `polyLines`, or polygons) and a data frame representing the `dbf` file. Examples are below.

The package reads shape types 1 (point), 3 (`polyLine`), 5 (polygon), 13 (`polyLineZ`), and 15 (`polygonZ`). Reading of shape type 13 and 15 from Don MacQueen, <macq@lln1.gov>

The package writes shape types 1 (point), 3 (`polyLine`), 5 (polygon), 13 (`polyLineZ`), and 15 (`polygonZ`). Conversion of simple polygons to shapefile format from Manuel Chirouze, <Manuel.Chirouze@benfieldgroup.com>

For simple features, the only difference between `polyLines` and polygons is that the first and last point is the same for a polygon. The `convert.to.simple` function can be used to simplify the `shp` file to a simple `data.frame`. The `change.id` function can then be used to change the `Id` field for the simple `shp data.frame` to a field from a `data.frame (dbf)`.

For details about the ESRI shapefile structure refer to <https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf>. A detailed description of DBF files can be found at <https://www.e-bachmann.dk/docs/xbase.htm>. The `arcgis` argument to `write.dbf` replaces "." with "\_" in field names since ArcGIS does not allow the former. Note that the `read.dbf` and `write.dbf` functions in the `foreign` package are now used for reading and writing dbfs, which greatly improves the speed of reading/writing dbfs.

Function `dp` is an implementation of the Douglas-Peucker `polyLine` simplification algorithm. Douglas, D. and Peucker, T. (1973). "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." *The Canadian Cartographer* 10(2). 112-122. `dp` currently uses the line, not the line segment to determine the distance of the points from the line. This can result in the omission of extreme "outlier-like" points. See [doi:10.3138/FM576770U75U7727](https://doi.org/10.3138/FM576770U75U7727) for more information.

## Value

|                              |      |                       |
|------------------------------|------|-----------------------|
| <code>read.shapefile</code>  | list | shapefile list object |
| <code>read.shp</code>        | list | shp list object       |
| <code>read.shx</code>        | list | shx list object       |
| <code>read.dbf</code>        | list | DBF list object       |
| <code>write.shapefile</code> | NA   | Nothing returned      |

|                      |      |                            |
|----------------------|------|----------------------------|
| write.shp            | NA   | Nothing returned           |
| write.shx            | NA   | Nothing returned           |
| write.dbf            | NA   | Nothing returned           |
| calc.header          | list | shapefile list object      |
| add.xy               | list | shapefile list object      |
| scaleXY              | list | shapefile list object      |
| convert.to.shapefile | list | shapefile list object      |
| convert.to.simple    | list | data.frame list data.frame |
| change.id            | list | data.frame list data.frame |
| dp                   | list | data.frame list data.frame |

### Author(s)

Ben Stabler <<benstabler@yahoo.com>>

### Examples

```
## Not run:
#Read entire shapefile
shapefile <- read.shapefile("links")

#Write entire shapefile
write.shapefile(shapefile, "temp", T)

#Read shp, shx, or dbf file
dbf <- read.dbf("links.dbf")

#Write shp, shx, or dbf file
write.dbf(dbf, "links.dbf", T)

#Calculate header (to clean up GeoMedia shapefile exports)
shapefile <- calc.header(shapefile)

#Add the X and Y coordinates to the dbf list of the shapefile list object
shapefile <- add.xy(shapefile)

#Scale the shapefile by scale.factor
shapefile <- scaleXY(shapefile, scale.factor)

#Samples of using the convert.to.shapefile function to write out simple shapefiles
#from basic R data.frames

#Point
dd <- data.frame(Id=c(1,2),X=c(3,5),Y=c(9,6))
ddTable <- data.frame(Id=c(1,2),Name=c("Item1","Item2"))
ddShapefile <- convert.to.shapefile(dd, ddTable, "Id", 1)
write.shapefile(ddShapefile, "c:/test", arcgis=T)
```

```
#PolyLine
dd <- data.frame(Id=c(1,1,1,2,2,2),X=c(3,5,8,6,7,8),Y=c(9,8,3,6,7,4))
ddTable <- data.frame(Id=c(1,2),Name=c("Item1","Item2"))
ddShapefile <- convert.to.shapefile(dd, ddTable, "Id", 3)
write.shapefile(ddShapefile, "c:/test", arcgis=T)

#Polygon
dd <- data.frame(Id=c(1,1,1,1,2,2,2,2),X=c(3,5,8,3,6,7,8,6),Y=c(9,8,3,9,6,7,4,6))
ddTable <- data.frame(Id=c(1,2),Name=c("Item1","Item2"))
ddShapefile <- convert.to.shapefile(dd, ddTable, "Id", 5)
write.shapefile(ddShapefile, "c:/test", arcgis=T)

#Convert to list of shapes
ddAsList <- by(dd,dd$Id, function(x) x)

#Convert to data.frame
dd <- do.call(rbind, ddAsList)

#Read in shp file and convert to simple format
shpTest <- read.shp("c:/test.shp")
simpleShpFormat <- convert.to.simple(shpTest)
simpleShpFormat <- change.id(simpleShpFormat, c("a","b"))
simpleAsList <- by(simpleShpFormat, simpleShpFormat[,1], function(x) x)
backToShape <- convert.to.shapefile(simpleShpFormat,
  data.frame(index=c("a","b")), "index", 5)
write.shapefile(backToShape, "c:/test", arcgis=T)

#Polyline simplification with dp algorithm
x <- c(5,3,4,1,8,9,10,11)
y <- c(6,4,2,1,1,5,2,3)
points <- list(x=x,y=y)
plot(points, type="l")
simpleLine <- dp(points, 2)
lines(simpleLine, type="l", col="blue")

## End(Not run)
```

# Index

## \* programming

shapefiles, 1

add.xy (shapefiles), 1

calc.header (shapefiles), 1

change.id (shapefiles), 1

convert.to.shapefile (shapefiles), 1

convert.to.simple (shapefiles), 1

dp (shapefiles), 1

read.dbf (shapefiles), 1

read.shapefile (shapefiles), 1

read.shp (shapefiles), 1

read.shx (shapefiles), 1

scaleXY (shapefiles), 1

shapefiles, 1

write.dbf (shapefiles), 1

write.shapefile (shapefiles), 1

write.shp (shapefiles), 1

write.shx (shapefiles), 1