Package ‘GISTools’

February 19, 2015

Title  Some further GIS capabilities for R
Version 0.7-4
Date 2014-10-06
Author Chris Brunsdon and Hongyan Chen
Maintainer Chris Brunsdon <christopher.brunsdon@nuim.ie>
Description Some mapping and spatial data manipulation tools - in particular
drawing choropleth maps with nice looking legends, and aggregation of point
data to polygons.
Depends R (>= 2.15.0), maptools, sp, RColorBrewer, MASS, rgeos
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2014-10-06 17:37:06

R topics documented:

GISTools-package .......................... 2
Add masking around an image .................. 2
auto.shading .................................. 3
choro.legend ................................. 5
choropleth ................................... 6
Computational Inference from Point Data .......... 7
Create a ‘mask’ polygon ....................... 8
Create Transparency .......................... 9
cut function .................................. 9
generalize.polys ............................. 10
gorgia ....................................... 11
Kernel Density Estimates From Points ............. 12
level.plot ................................... 13
map.scale .................................... 14
newhaven .................................... 15
North Arrow .................................. 16
Add masking around an image

Add a number of utilities for handling and visualising geographical data - for example choropleth mapping with 'nice' legends.

**Examples**

```r
# Load up the libraries needed
library(maptools)
library(RColorBrewer)

# Read in map data and compute a rate for mapping
sids <- readShapePoly(system.file("shapes/sids.shp", package="maptools")[[1]],
                      proj4string=CRS("+proj=longlat +ellps=clrk66"))
sids.rate <- 10000*sids$SID74/sids$BIR74

# Create the shading scheme, plot a choropleth map and add a legend
shades <- auto.shading(sids.rate)
choropleth(sids,sids.rate,shades)
choro.legend(c(-83.77,37.87),shades,fmt="%4.1f",title='Rate per 10,000')
```

**Description**

Takes an ‘mask’ type polygon object - basically a rectangle with a polygon hole cut through it - and draws this over an image. This has the effect of only showing the image inside the hole. This is useful for plotting surfaces defined over a study area, but masking the values outside of the area.

**Usage**

```r
add.masking(maskPoly,color)
```
**auto.shading**

**Arguments**

- **maskPoly**: A masking polygon as described above.
- **color**: Colour of the mask. Defaults to white, but for example, sea could be shown as blue.

**Details**

Returns no value, but draws a mask on the current graphics device as a side effect.

**Value**

None

**Author(s)**

Chris Brunsdon

**See Also**

poly.outer, kde.points.

**Examples**

```r
# Data for New Haven to use in example
data(newhaven)
# Do the KDE
breach.dens = kde.points(breach,lims=tracts)
# Plot the result
level.plot(breach.dens)
# Block out the part outside the study area
masker = poly.outer(breach.dens,tracts,extend=100); add.masking(masker)
# Plot census tract boundaries
plot(tracts,add=TRUE)
```

---

**auto.shading**

**auto.shading**

**Description**

Creates an object of class shading automatically, given a choropleth variable to be mapped.

**Usage**

```r
auto.shading(x, digits = 2, cutter = quantileCuts, n = 5,
params = NA, cols = brewer.pal(n, "Reds"))
```
auto.shading

Arguments

- **x**: The variable to be mapped.
- **digits**: The number of significant digits to round the class limits to.
- **cutter**: Function used to create the break points. Can be user defined or a supplied cut function.
- **n**: The number of classes. The should be one more than the number of break points.
- **params**: Other parameters to be passed to the cut function.
- **cols**: List of colours for shading each class. length(cols) should be equal to n.

Details

Returns an object of class shading, as set out below:

Value

An object of class shading, having the following list elements:

- **breaks**: Break points between choropleth classes. length(cols)
- **cols**: Colours to shade in each class. length(cols) should be one more than length(breaks)

Author(s)

Chris Brunsdon

See Also

choropleth, shading, choro.legend.

Examples

```r
# Read in map data and compute a rate for mapping
sids <- readShapePoly(system.file("shapes/sids.shp", package="maptools")[[1]],
  proj4string=CRS("+proj=longlat +ellps=clrk66"))
sids.rate=10000*sids$SID74/sids$BIR74
# Create the shading scheme, plot a choropleth map and add a legend
shades = auto.shading(sids.rate,n=6)
choropleth(sids,sids.rate,shades)
choro.legend(-83.77,37.87,shades,fmt="%4.1f",title='Rate per 10,000')
# Now again with a different set of class intervals and colours
shades = auto.shading(sids.rate,n=6,cutter=rangeCuts,cols=brewer.pal(6,'Greens'))
choropleth(sids,sids.rate,shades)
choro.legend(-83.77,37.87,shades,fmt="%4.1f",title='Rate per 10,000')
```
choro.legend

Description

Draw a legend for a choropleth map.

Usage

choro.legend(px, py, sh, under = "under", over = "over",
             between = "to", fmt = "%g", cex=1, ...)

Arguments

px  x coordinate of legend location
py  y coordinate of legend location
sh  Shading scheme object used as basis for the legend
under  What to write in front of the lowest choropleth class upper limit.
over    What to write in front of the highest choropleth class lower limit.
between  What to write between the upper and lower limits of intermediate choropleth classes.
fmt  C style format for values stated in above choropleth class limits.
cex  Relative size of text in the legend.
...  Other arguments, passed on to the generic legend function.

Details

Returns no value, but draws a choropleth map legend on the current graphics device as a side effect

Value

None (see above)

Author(s)

Chris Brunsdon

See Also

choropleth, auto.shading, shading.
Examples

```r
# Read in map data and compute a rate for mapping
sids <- readShapePoly(system.file("shapes/sids.shp", package="maptools")[[1]],
                      proj4string=CRS("+proj=longlat +ellps=clrk66"))
sids.rate <- 10000*sids$data[,10]/sids$data[,9]
# Create the shading scheme, plot a choropleth map and add a legend
shades = auto.shading(sids.rate)
choropleth(sids, sids.rate, shades)
choro.legend(-83.77,37.87,shades,fmt="%4.1f",cex=0.8,title='Rate per 10,000')
```

Description

Draws a choropleth map given a spatialPolygons object, a variable and a shading scheme.

Usage

```r
choropleth(sp, v, shading = auto.shading(v), ...)
```

Arguments

- **sp**: A spatialPolygons or spatialPolygonsDataFrame object.
- **v**: The variable to be mapped. Must have the same number of elements as `sp` has polygons.
- **shading**: A shading scheme created by `shading` or `auto.shading`.
- **...**: Additional parameters to be passed on to the `plot` method for `sp`.

Details

The function returns no value, but draws a choropleth map on the current graphics device as a side effect.

Value

None (see above).

Author(s)

Chris Brunsdon

See Also

`choro.legend`, `auto.shading`, `shading`
Examples

# Read in map data and compute a rate for mapping
sids <- readShapePoly(system.file("shapes/sids.shp", package="maptools")[[1]],
  proj4string=CRS("+proj=longlat +ellps=clrk66"))
sids.rate = 10000 * sids$SID74 / sids$BIR74
# Create the shading scheme, plot a choropleth map
shades = auto.shading(sids.rate, cols=brewer.pal(5, 'Blues'))
choropleth(sids, sids.rate, shades)

Computational Inference from Point Data

Bootstrap and Kernel Bootstrap from Points

Description

Operations for bootstrapping and kernel bootstrapping based on point data. `bootstrap.points` samples `n` points with replacement from a sample - and `jitter.points` adds a Gaussian displacement to each point in a data set. Applying a jitter to a bootstrap effectively creates a kernel bootstrap operation.

Usage

```
jitter.points(pts, scl)
bootstrap.points(pts)
```

Arguments

- `pts`: A SpatialPointsDataFrame
- `scl`: A scale parameter - basically the standard deviation of the random Gaussian displacement

Value

A SpatialPointsDataFrame - with either a sample without replacement or a replica of the input data with displacements.

Author(s)

Chris Brunsdon

Examples

```
data(newhaven)
plot(blocks)
for (i in 1:20) plot(jitter.points(breach, 150), add=TRUE, pch=1, col='red')
```
Description

Takes a polygon object and creates a new polygon whose outline is rectangular, but has a hole shaped like the input polygon cut into it. This is useful for plotting surfaces defined over a study area, but masking the values outside of the area. It is designed to work with pixel images, so that the mask covers up all parts of the image not in the input polygon.

Usage

poly.outer(exo.object,input.poly,extend=0)

Arguments

exo.object The object extending beyond input.poly that is to be masked. This is required to ensure that the external rectangle of the mask will be large enough.

input.poly The polygon used to make the hole in the mask.

extend A buffer used to extend the mask if it is required to be larger than exo.object

Value

A polygon object whose outline is rectangular, but having holes cut into it in the shape of input.poly

Author(s)

Chris Brunsdon

See Also

add.masking.kde.points.

Examples

# Data for New Haven to use in example
data(newhaven)
# Do the KDE
breach.dens = kde.points(breach,lims=tracts)
# Plot the result
level.plot(breach.dens)
# Block out the part outside the study area
masker = poly.outer(breach.dens,tracts,extend=100); add.masking(masker)
# Plot census tract boundaries
plot(tracts,add=TRUE)
Create Transparency

Add transparency to a hex-defined colour

Description

Takes a colour defined in hex format as \#XXXXXX and adds a two transparency bytes XX based on a number from 0 to 1. Its main use is to make RColorBrewer palettes transparent.

Usage

add.alpha(hex.color.list, alpha)

Arguments

hex.color.list A list of strings defining solid colors in six byte format.
alpha A value (or list of values) from 0 to 1 specifying transparency.

Value

A list of strings defining transparent colours in eight byte format.

Author(s)

Chris Brunsdon

Examples

# Make a list of semi-transparent RColorBrewer colours, based on Brewer's Red palette with 5 shades
add.alpha(brewer.pal(5,'Reds'),0.5)

Cut function

Cut functions

Description

Helper functions for auto.shading. Given a variable to be mapped, a number of classes and possibly some more params, returns a list of break values. There should be one less break value than the number of classes.

Usage

quantileCuts(x, n = 5, params = NA)
sdCuts(x, n = 5, params = NA)
rangCuts(x, n = 5, params = NA)
generalize.polys

Arguments

x    The variable to be mapped.
n    The number of classes.
params Extra params for individual cut functions.

Value

An ordered list of the break values between classes

Note

The only cut function using params is quantileCuts, where it is used to specify a list of quantile values - useful if they are not evenly spaced.

Author(s)

Chris Brunsdon

See Also

auto.shading

generalize.polys  
generalize.polys

description

Generalises a SpatialPolygons or SpatialPolygonsDataFrame object using the Douglas-Peuker algorithm

Usage

generalize.polys(sp, tol)

Arguments

sp A SpatialPolygons or SpatialPolygonsDataFrame object.
tol The weeding tolerance for the generalisation algorithm.

Details

Returns an object of the same class as sp. Note that the algorithm is applied on a polygon-by-polygon, not edge-by-edge basis. Thus edges in generalised polygons may not match perfectly.

Value

An object of class SpatialPolygons or SpatialPolygonsDataFrame. Each polygon shape has been generalized using the Douglas-Peuker algorithm.
Author(s)

Chris Brunsdon

Examples

```r
# Data for Georgia to use in example
data(georgia)
# Create an outline of Georgia
georgia.outline <- unionSpatialPolygons(georgia,rep(1,159))
plot(georgia.outline)
georgia.generalised <- generalize.polys(georgia.outline,0.1)
plot(georgia.generalised,add=TRUE,border='red')
```

---

data(georgia)  
```
# Georgia Social and Economic Data by County

Description

Polygon Data Frame as used in the Brunsdon, Fotheringham & Charlton GWR book, with further variable median income (MedInc)

Usage

data(georgia)
```
georgia
georgia2

Format

- **georgia** Georgia polygons SpatialPolygonsDataFrame - geographical projection
- **georgia2** Georgia polygons SpatialPolygonsDataFrame - equal area projection
- **georgia.polys** Georgia polygons in list format - equal area projection

Examples

```r
# Read in the data
data(georgia)
# Make a map of median income
choropleth(georgia2,georgia2$MedInc)
```
Kernel Density Estimates From Points

Description

Given a set of points, a bandwidth, a grid density and a frame, produce a kernel density estimate

Usage

kde.points(pts,h,n=200,lims=NULL)

Arguments

pts A SpatialPoints or SpatialPointsDataFrame object.
h A real number - the bandwidth of the KDE
n An integer, the output grid density - ie result is nxn grid
lims A spatial object - the KDE grid will cover this, if provided

Value

A SpatialPixelsDataFrame containing the KDE.

Author(s)

Chris Brunsdon

Examples

# Data for New Haven to use in example
data(newhaven)
# Do the KDE
breach.dens = kde.points(breach,lims=tracts)
# Plot the result
level.plot(breach.dens)
# Block out the part outside the study area
masker = poly.outer(breach.dens,tracts,extend=100); add.masking(masker)
# Plot census tract boundaries
plot(tracts,add=TRUE)
**level.plot**

**Level plot for gridded data**

**Description**

Draws a level plot given a SpatialPixelsDataFrame, an index and a shading scheme.

**Usage**

```r
level.plot(grd, shades, index=1, add=FALSE)
```

**Arguments**

- `grd`: A `SpatialPixelsDataFrame` object.
- `shades`: A shading scheme created by `shading` or `auto.shading`. If omitted, chosen automatically from `grd`.
- `index`: Index giving the variable in `grd` to plot.
- `add`: Whether to add the level plot to an existing plot.

**Details**

The function returns no value, but draws a level plot on the current graphics device as a side effect.

**Value**

None (see above).

**Author(s)**

Chris Brunsdon

**Examples**

```r
# Data for New Haven to use in example
data(newhaven)
# Do the KDE
breach.dens = kde.points(breach,lims=tracts)
# Plot the result
level.plot(breach.dens)
# Block out the part outside the study area
masker = poly.outer(breach.dens,tracts,extend=100); add.masking(masker)
# Plot census tract boundaries
plot(tracts,add=TRUE)
```
**Description**

Draws a scale bar on a map.

**Usage**

```r
map.scale(xc, yc, len, units, ndivs, subdiv = 1, tcol = 'black', scol = 'black', sfcol = 'black')
```

**Arguments**

- `xc`: The x-centre (in map units) of the scale bar
- `yc`: The y-centre (in map units) of the scale bar
- `len`: The length (in map units) of the scale bar
- `units`: String specifying the name of the units for the scale bar
- `ndivs`: The number of divisions (units marked) on the scale
- `subdiv`: The fraction of `units` used to step along the divisions
- `tcol`: The colour of text on the scale bar.
- `scol`: The colour of the scale bar itself.
- `sfcol`: The colour of the filled rectangles in the scale bar.

**Details**

Draws an alternating bar scale on a map. Returns no value.

**Value**

None (see above)

**Author(s)**

Chris Brunsdon

**See Also**

`choro.legend`
Examples

# Read in map data for New Haven
data(newhaven)
# Plot census block boundaries
plot(blocks)
# Add a map scale
map.scale(534750,1520000,miles2ft(2),"Miles",4,0.5,sfcol='red')
# ... and a title
title('New Haven (CT)')

Description

Data set from New Haven (CT) crime web site containing point sources of some crimes, plus roads, railways and census block spatial data frames.

Usage

data(newhaven)
blocks
breach
famdisp
burgres.f
burgres.n
places
roads
tracts

Format

- blocks Census blocks SpatialPolygonsDataFrame
- roads Roads SpatialLinesDataFrame
- places Place names SpatialPointsDataFrame
- breach Breach of peace SpatialPointsDataFrame
- famdisp Family dispute SpatialPointsDataFrame
- tracts Census tracts SpatialPolygonsDataFrame
- burgres.f Residential Burglary (Forced) SpatialPointsDataFrame
- burgres.n Residential Burglary (Non-Forced) SpatialPointsDataFrame

Source

http://www.newhavencrimelog.org/
Examples

# Read in map data for New Haven
data(newhaven)
# Plot census block boundaries
plot(blocks)
# Add a map scale
map.scale(534750, 152000, miles2ft(2), "Miles", 4, 0.5, sfcol='red')
# ... and a title
title('New Haven (CT)'

---

North Arrow  
Add a north arrow to a map

Description

Draws a north arrow on a map.

Usage

north.arrow(xb, yb, len, lab='NORTH', cex.lab=1, tcol='black', ...)

Arguments

xb  The x-centre (in map units) of the arrow base.
yb  The y-centre (in map units) of the arrow base.
len The length (in map units) of the arrow base.
lab The label for the arrow.
cex.lab Scale factor for the label for the arrow.
tcol The colour of the label text.
... Other graphical parameters passed to the drawing of the arrow.

Details

Draws a north arrow on a map. The arrow itself is drawn using polygon and any extra parameters are passed to this call.

Value

None.

Author(s)

Chris Brunsdon

See Also

map.scale
Examples

```r
# Read in map data for New Haven
data(newhaven)
# Plot census block boundaries
plot(blocks)
# Add a north arrow
north.arrow(534750,152000,miles2ft(0.5),col='cyan')
# ... and a title
title('New Haven (CT)')
```

Description


Usage

```r
data(phenology)
chinensis
chinensis2
us_states
us_states2
```

Format

- **chinensis** Syringa Chinensis Observation Stations SpatialPointsDataFrame - geographical projection
- **chinensis2** Syringa Chinensis Observation Stations SpatialPointsDataFrame - equal area projection
- **us_states** States of US SpatialPolygonsDataFrame - geographical projection
- **us_states2** States of US SpatialPolygonsDataFrame - equal area projection

Source

```
http://www.ncdc.noaa.gov/paleo/phenology.html
```

Examples

```r
# Read in the data
data(phenology)
# Split the plot in two
par(mfrow=c(2,1))
# Plot US states
```
Point in Polygon Counts

Number of Points in Each Polygon

Description

Given a set of points, and a set of polygons, computes the number of points in each polygon.

Usage

poly.counts(pts, polys)

Arguments

pts       A SpatialPoints or SpatialPointsDataFrame object.
polys     A SpatialPolygons or SpatialPolygonsDataFrame object.

Value

A list of integers of the same length as the number of polygons in polys, giving the number of points from pts.

Author(s)

Chris Brunsdon

Examples

# Data for New Haven to use in example
data(newhaven)
# How many breaches of peace in each census block?
n.breach = poly.counts(breach, blocks)
# Compute densities and map them
choropleth(blocks, n.breach/poly.areas(blocks))
**Polygon Areas**

---

**Area of Each Polygon**

<table>
<thead>
<tr>
<th>Polygon Areas</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Given a set of polygons, returns the area of each polygon.</td>
</tr>
</tbody>
</table>

**Usage**

```
poly.areas(polys)
```

**Arguments**

- `polys` A `SpatialPolygons` or `SpatialPolygonsDataFrame` object.

**Value**

A list of areas of the same length as the number of polygons in `polys`.

**Author(s)**

Chris Brunsdon

**Examples**

```r
# Data for New Haven to use in example
data(newhaven)
# What is the area each census block?
poly.areas(blocks)
```

---

**Polygon Label Points**

---

**Number of Points in Each Polygon**

<table>
<thead>
<tr>
<th>Polygon Label Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Given a set of polygons, returns the label point for each polygon in a <code>SpatialPoints</code> object.</td>
</tr>
</tbody>
</table>

**Usage**

```
poly.labels(polys)
```

**Arguments**

- `polys` A `SpatialPolygons` or `SpatialPolygonsDataFrame` object.
Value

SpatialPoints object containing the label point for each polygon.

Author(s)

Chris Brunsdon

Examples

```
# Data for New Haven to use in example
data(newhaven)
# How many breaches of peace in each census block?
n.breach = poly.counts(breach, blocks)
# Compute densities and map them
choropleth(blocks, n.breach/blocks$AREA)
```

Description

Creates an object of class shading by directly specifying break values and (optionally) colours.

Usage

```
shading(breaks, cols = brewer.pal(length(breaks), "Reds"))
```

Arguments

- **breaks**: The break points
- **cols**: The shading colours - there should be one more of these than break points.

Value

An object of class shading.

Warning

At the moment, it is assumed that the number of shading colours is one more than the break points, but this is not checked.

Author(s)

Chris Brunsdon

See Also

`choropleth, choro.legend`
Examples

# Read in map data and compute a rate for mapping
sids <- readShapePoly(system.file("shapes/sids.shp", package="maptools"))[1],
proj4string=CRS("+proj=longlat +ellps=clrk66")
sids.rate=10000*sids@data[,10]/sids@data[,9]
shades = shading(breaks=c(15,30,45,60,75),cols=brewer.pal(6,'YlGn'))
choropleth(sids,sids.rate,shades)
choro.legend(-83.77,37.87,shades,fmt="%4.0f",title='Rate per 10,000')

---

tornados

US Tornado Touchdown Data

Description

Data set from NOAA's National Weather Service Indianapolis, IN Weather Forecast Office 6900 W. Hanna Ave.

Usage

data(tornados)
torn
torn2

Format

- **torn** Tornado Touchdown points SpatialPointsDataFrame - geographical projection
- **torn2** Tornado Touchdown points SpatialPointsDataFrame - equal area projection

Source

http://www.crh.noaa.gov/ind/?n=svrgis

Examples

# Read in the data
data(tornados)
# Split the plot in two
par(mfrow=c(2,1))
# Plot US states
plot(us_states)
# Add Locations of observation stations
plot(torn,add=TRUE,pch=16,col='red')
# Plot a histogram of year of observation next to this
hist(torn$YEAR)
Unit Conversion

Description

Convert between different distance units - all functions take the form \( xx \rightarrow yy \) where \( xx \) is the unit to be converted from and \( yy \) is the unit to be converted to.

Usage

\[
\begin{align*}
\text{ft2miles}(x) \\
\text{miles2ft}(x) \\
\text{ft2km}(x) \\
\text{km2ft}(x)
\end{align*}
\]

Arguments

\( x \) A quantity in units to be converted from

Value

The value of \( x \) converted to the new units. In the example below the conversions are from feet to miles and feet to kilometers (hence functions are \( \text{ft2miles} \) and \( \text{ft2km} \)).

Author(s)

Chris Brunsdon

Examples

\[
\begin{align*}
\# \text{ How many miles is 10,000 feet?} \\
\text{ft2miles}(10000) \\
\# \text{ How about in kilometers?} \\
\text{ft2km}(10000)
\end{align*}
\]

Phenology data for North American lilacs

Description

**vulgaris**

**Usage**

```r
data(vulgaris)
vulgaris2
us_states
us_states2
```

**Format**

- **vulgaris** Syringa Vulgaris Observation Stations SpatialPointsDataFrame - geographical projection
- **vulgaris2** Syringa Vulgaris Observation Stations SpatialPointsDataFrame - equal area projection
- **us_states** States of US SpatialPolygonsDataFrame - geographical projection
- **us_states2** States of US SpatialPolygonsDataFrame - equal area projection

**Source**

http://www.ncdc.noaa.gov/paleo/phenology.html

**Examples**

```r
# Read in the data
data(vulgaris)
# Split the plot in two
par(mfrow=c(2,1))
# Plot US states
plot(us_states)
# Add Locations of observation stations
plot(vulgaris,add=TRUE,pch=16,col='red')
# Plot a histogram of year of observation next to this
hist(vulgaris$Year)
```
Index

Add masking around an image, 2
add.alpha (Create Transparency), 9
add.masking, 8
add.masking (Add masking around an image), 2
auto.shading, 3, 5, 6, 9, 10
blocks (newhaven), 15
breach (newhaven), 15
bstrap.points (Computational Inference from Point Data), 7
burgres.f (newhaven), 15
burgres.n (newhaven), 15

chinensis (phenology), 17
chinensis2 (phenology), 17
choropleth, 4, 5, 6, 20
Computational Inference from Point Data, 7
Create a ‘mask’ polygon, 8
Create Transparency, 9
cut function, 4, 9

famdisp (newhaven), 15
ft2km (Unit Conversion), 22
ft2miles (Unit Conversion), 22

generalize.polys, 10
georgia, 11
georgia2 (georgia), 11
GISTools (GISTools-package), 2
GISTools-package, 2

jitter.points (Computational Inference from Point Data), 7

kde.points, 3, 8
kde.points (Kernel Density Estimates from Points), 12

Kernel Density Estimates From Points, 12
km2ft (Unit Conversion), 22
legend, 5
level.plot, 13
map.scale, 14, 16
miles2ft (Unit Conversion), 22

newhaven, 15
North Arrow, 16
north.arrow (North Arrow), 16

phenology, 17
places (newhaven), 15
Point in Polygon Counts, 18
poly.areas (Polygon Areas), 19
poly.counts (Point in Polygon Counts), 18
poly.labels (Polygon Label Points), 19
poly.outer, 3
poly.outer (Create a ‘mask’ polygon), 8
Polygon Areas, 19
Polygon Label Points, 19
quantileCuts (cut function), 9
rangeCuts (cut function), 9
roads (newhaven), 15

sdCuts (cut function), 9
shading, 4–6, 20

torn (tornados), 21
torn2 (tornados), 21
tornados, 21
tracts (newhaven), 15

Unit Conversion, 22
us_states (vulgaris), 22
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>US States 2 (vulgaris)</td>
<td>22</td>
</tr>
<tr>
<td>vulgaris</td>
<td>22</td>
</tr>
<tr>
<td>vulgaris 2 (vulgaris)</td>
<td>22</td>
</tr>
</tbody>
</table>