# Package: dots (via r-universe)

October 23, 2026

**Title** Dot Density Maps

**Version** 0.0.3

**Description** Generate point data for representing people within spatial data. This collects a suite of tools for creating simple dot density maps. Several functions from different spatial packages are standardized to take the same arguments so that they can be easily substituted for each other.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Imports** ggplot2, rlang, rmapshaper, sf, terra, purrr, dplyr, sp

**URL** https://github.com/christopherkenny/dots, http://christophertkenny.com/dots/

**BugReports** https://github.com/christopherkenny/dots/issues

**Depends** R (>= 4.1.0)

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), wacolors

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/Needs/website** christopherkenny/ctktemplate

**Config/pak/sysreqs** libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libnode-dev libx11-dev

**Repository** https://christopherkenny.r-universe.dev

**Date/Publication** 2026-05-01 16:15:00 UTC

**RemoteUrl** https://github.com/christopherkenny/dots

**RemoteRef** HEAD

**RemoteSha** ed874678ede2f02372ecf90acdd43ca423a57f67

# Contents

boston_water . . . 2
clip_water . . . 3
dots . . . 3
dots_points . . . 4
engine_sf_hexagonal . . . 5
engine_sf_random . . . 5
engine_sf_regular . . . 6
engine_sp_clustered . . . 7
engine_sp_hexagonal . . . 7
engine_sp_nonaligned . . . 8
engine_sp_random . . . 9
engine_sp_regular . . . 9
engine_sp_stratified . . . 10
engine_terra . . . 11
filter_pts . . . 11
suffolk . . . 12

**Index** **13**

---

`boston_water` *Boston Water*

---

## Description

This data contains the largest named water within Suffolk County MA, with geographies simplified.

## Usage

```
data("boston_water")
```

## Format

An sf dataframe with 10 observations

## Examples

```
data('boston_water')
```

## clip_water *Remove Water*

### Description

Remove Water

### Usage

```
clip_water(shp, water, filter_islands = FALSE, ...)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| water | water shapes to remove with sf geometry |
| filter_islands | logical. Should additional filtering be done to remove small areas? |
| ... | additional arguments to pass to rmapshaper::ms_filter_islands(). Only used if filter_islands = TRUE. |

### Value

tibble with sf geometries

### Examples

```
# time to run varies greatly, depending on machine
data(suffolk)
data(boston_water)
clip_water(suffolk, boston_water[10, ])
```

## dots *Make dot density plots*

### Description

Make dot density plots

### Usage

```
dots(
  shp,
  cols,
  engine = engine_terra,
  divisor = 250,
  min_point = 0.1 * divisor
)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| cols | <tidy-select> columns to produce dots for. |
| engine | backend to use. Default is engine_terra. |
| divisor | Number of people per dot. Default is 250. |
| min_point | Minimum number of people to generate one dot. Defaults to 10% of the divisor. |

### Value

A ggplot

### Examples

```
data('suffolk')
# subset to first 20 rows for speed on CRAN
dots(suffolk[1:20, ], c(vap_black), divisor = 2000)
```

---

## dots_points *Make dot density points*

---

### Description

Make dot density points

### Usage

```
dots_points(
  shp,
  cols,
  engine = engine_terra,
  divisor = 250,
  min_point = 0.1 * divisor
)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| cols | <tidy-select> columns to produce dots for. |
| engine | backend to use. Default is engine_terra. |
| divisor | Number of people per dot. Default is 250. |
| min_point | Minimum number of people to generate one dot. Defaults to 10% of the divisor. |

### Value

tibble with sf geometries

### Examples

```
data('suffolk')
# subset to first 20 rows for speed on CRAN
dots_points(suffolk[1:20, ], c(vap_black))
```

---

## engine_sf_hexagonal *Generate Hexagonal Points with* sf

---

### Description

Uses sf::st_sample() to produce points and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sf_hexagonal(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_sf_hexagonal(suffolk[16:20,], 'pop_nhpi')
```

---

## engine_sf_random *Generate Random Points with* sf

---

### Description

Uses sf::st_sample() to produce points and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs.

### Usage

```
engine_sf_random(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_sf_random(suffolk[16:20,], 'pop_nhpi')
```

---

## engine_sf_regular *Generate Regular Points with* sf

---

### Description

Uses sf::st_sample() to produce points and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sf_regular(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_sf_regular(suffolk[16:20,], 'pop_nhpi')
```

## engine_sp_clustered *Generate Clustered Points with* sp

### Description

Uses sp::spsample() with method "clustered" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sp_clustered(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_sp_clustered(suffolk[16:20, ], 'pop_nhpi')
```

## engine_sp_hexagonal *Generate Hexagonal Points with* sp

### Description

Uses sp::spsample() with method "hexagonal" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sp_hexagonal(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

## Value

tibble with sf geometries

## Examples

```
set.seed(1)
data('suffolk')
engine_sp_hexagonal(suffolk[16:20, ], 'pop_nhpi')
```

---

# engine_sp_nonaligned *Generate Nonaligned Points with* sp

---

## Description

Uses sp::spsample() with method "nonaligned" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

## Usage

```
engine_sp_nonaligned(shp, col)
```

## Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

## Value

tibble with sf geometries

## Examples

```
set.seed(1)
data('suffolk')
engine_sp_nonaligned(suffolk[16:20, ], 'pop_nhpi')
```

## engine_sp_random *Generate Random Points with* sp

### Description

Uses sp::spsample() with method "random" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sp_random(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_sp_random(suffolk[16:20,], 'pop_nhpi')
```

## engine_sp_regular *Generate Regular Points with* sp

### Description

Uses sp::spsample() with method "regular" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

### Usage

```
engine_sp_regular(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

## Value

tibble with sf geometries

## Examples

```
set.seed(1)
data('suffolk')
engine_sp_regular(suffolk[16:20,], 'pop_nhpi')
```

---

# engine_sp_stratified *Generate Stratified Points with* sp

---

## Description

Uses sp::spsample() with method "regular" to produce points, converts back to sf, and spatial joins with input shp. Each engine function takes the same arguments and produces comparable outputs. Final number of points may be approximate for this method.

## Usage

```
engine_sp_stratified(shp, col)
```

## Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

## Value

tibble with sf geometries

## Examples

```
set.seed(1)
data('suffolk')
engine_sp_stratified(suffolk[16:20, ], 'pop_nhpi')
```

## engine_terra *Generate Points with* terra

### Description

Uses terra::dots() to produce points and transforms back to sf. Each engine function takes the same arguments and produces comparable outputs.

### Usage

```
engine_terra(shp, col)
```

### Arguments

| | |
|---|---|
| shp | input shp with sf geometry. |
| col | character column name to produce points with |

### Value

tibble with sf geometries

### Examples

```
set.seed(1)
data('suffolk')
engine_terra(suffolk, 'pop_nhpi')
```

## filter_pts *Filter Points to a Region*

### Description

Filter Points to a Region

### Usage

```
filter_pts(pts, shp, cond = TRUE)
```

### Arguments

| | |
|---|---|
| pts | points with sf geometry to filter |
| shp | shp to filter to |
| cond | geometry subset to reduce shp to |

## Value

tibble with `sf` geometries

## Examples

```
data(suffolk)
pts <- dots_points(suffolk, pop, divisor = 1000)
filter_pts(pts, suffolk, pop < 1000)
```

---

# suffolk *Suffolk County, MA Voting Districts*

---

## Description

This data contains the voting districts for Suffolk County MA, with geographies simplified.

## Usage

```
data("suffolk")
```

## Format

An sf dataframe with 295 observations

## Examples

```
data('suffolk')
```

# Index

∗ **data**
boston_water, 2
suffolk, 12

boston_water, 2

clip_water, 3

dots, 3
dots_points, 4

engine_sf_hexagonal, 5
engine_sf_random, 5
engine_sf_regular, 6
engine_sp_clustered, 7
engine_sp_hexagonal, 7
engine_sp_nonaligned, 8
engine_sp_random, 9
engine_sp_regular, 9
engine_sp_stratified, 10
engine_terra, 11

filter_pts, 11

suffolk, 12