Package 'wellknown'

October 22, 2015

Title Convert Between 'WKT' and 'GeoJSON'

Description Convert 'WKT' to 'GeoJSON' and 'GeoJSON' to 'WKT'. Functions included for converting between 'GeoJSON' to 'WKT', creating both 'GeoJSON' features, and non-features, creating WKT from R objects (e.g., lists, data.frames, vectors), and linting 'WKT'.

Version 0.1.0

License MIT + file LICENSE

LazyData true

URL https://github.com/ropensci/wellknown

BugReports https://github.com/ropensci/wellknown/issues

VignetteBuilder knitr

Imports methods, utils, jsonlite (>= 0.9.17), magrittr

Suggests knitr, leaflet (>= 1.0.0), testthat

NeedsCompilation no

Author Scott Chamberlain [aut, cre]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2015-10-22 08:24:51

R topics documented:

`wellknown-package` *WKT to GeoJSON and vice versa*

Description

WKT to GeoJSON and vice versa

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

`as_featurecollection` *As featurecollection*

Description

As featurecollection

Usage

```
as_featurecollection(x)
```

Arguments

`x`	Input

Details

Helper function to make a FeatureCollection list object for use in vizualizing, e.g., with `leaflet`

Examples

```
str <- 'MULTIPOINT ((100.000 3.101), (101.000 2.100), (3.140 2.180),
(31.140 6.180), (31.140 78.180))'
# wkt2geojson(str, fmt = 2) %>%
#   as_featurecollection() %>%
#   lawn::view()
```

as_json *Convert geojson R list to JSON*

Description

Convert geojson R list to JSON

Usage

```
as_json(x, pretty = TRUE, auto_unbox = TRUE, ...)
```

Arguments

x	Output from wkt2geojson
pretty	(logical) Adds indentation whitespace to JSON output. Can be TRUE/FALSE or a number specifying the number of spaces to indent. See prettify Default: TRUE Having TRUE as default makes it easy to copy paste to a text editor, etc.
auto_unbox	(logical) Automatically unbox all atomic vectors of length 1. Default: TRUE
...	Further args passed on to toJSON

Examples

```
str <- "POLYGON ((100 0.1, 101.1 0.3, 101 0.5, 100 0.1),
   (103.2 0.2, 104.8 0.2, 100.8 0.8, 103.2 0.2))"
as_json(wkt2geojson(str))
as_json(wkt2geojson(str), FALSE)
```

circularstring *Make WKT circularstring objects*

Description

Make WKT circularstring objects

Usage

```
circularstring(..., fmt = 16)
```

Arguments

...	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
fmt	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

See Also

Other R.objects: geometrycollection; linestring; multilinestring; multipoint; multipolygon; point; polygon

Examples

```
## empty circularstring
circularstring("empty")
# circularstring("stuff")

# Character string
circularstring("CIRCULARSTRING(1 5, 6 2, 7 3)")

# data.frame
df <- data.frame(lon = c(-116.4, -118), lat = c(45.2, 47))
circularstring(df, fmt=1)
df <- data.frame(lon=c(-116.4, -118, -120), lat=c(45.2, 47, 49))
circularstring(df, fmt=1)

# matrix
mat <- matrix(c(-116.4,-118, 45.2, 47), ncol = 2)
circularstring(mat, fmt=1)
mat2 <- matrix(c(-116.4, -118, -120, 45.2, 47, 49), ncol = 2)
circularstring(mat2, fmt=1)

# list
x <- list(c(1, 5), c(6, 2), c(7, 3))
circularstring(x, fmt=2)
```

`geojson2wkt` *Convert GeoJSON-like objects to WKT.*

Description

Convert GeoJSON-like objects to WKT.

Usage

```
geojson2wkt(obj, fmt = 16, ...)
```

Arguments

`obj`	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates.
`...`	Further args passed on to `fromJSON` only in the event of json passed as a character string.

See Also

wkt2geojson

Examples

```
# point
point <- list('type' = 'Point', 'coordinates' = c(116.4, 45.2))
geojson2wkt(point)

# multipoint
mp <- list(type = 'MultiPoint', coordinates=list(c(100.0, 3.101), c(101.0, 2.1), c(3.14, 2.18)))
geojson2wkt(mp)

# linestring
st <- list(type = 'LineString',
           coordinates = list(c(0.0, 0.0, 10.0), c(2.0, 1.0, 20.0),
                              c(4.0, 2.0, 30.0), c(5.0, 4.0, 40.0)))
geojson2wkt(st)

# multilinestring
multist <- list(type = 'MultiLineString',
     coordinates = list(
       list(c(0.0, -1.0), c(-2.0, -3.0), c(-4.0, -5.0)),
       list(c(1.66, -31023.5), c(10000.9999, 3.0), c(100.9, 1.1), c(0.0, 0.0))
     ))
geojson2wkt(multist)

# polygon
poly <- list(type = 'Polygon',
     coordinates=list(
      list(c(100.001, 0.001), c(101.12345, 0.001), c(101.001, 1.001), c(100.001, 0.001)),
       list(c(100.201, 0.201), c(100.801, 0.201), c(100.801, 0.801), c(100.201, 0.201))
))
geojson2wkt(poly)
geojson2wkt(poly, fmt=6)

# multipolygon
mpoly <- list(type = 'MultiPolygon',
              coordinates=list(
                list(list(c(100.001, 0.001), c(101.001, 0.001), c(101.001, 1.001),
                          c(100.001, 0.001)),
                     list(c(100.201, 0.201), c(100.801, 0.201), c(100.801, 0.801),
                          c(100.201, 0.201))),
                list(list(c(1.0, 2.0, 3.0, 4.0), c(5.0, 6.0, 7.0, 8.0),
                          c(9.0, 10.0, 11.0, 12.0), c(1.0, 2.0, 3.0, 4.0)))
))
geojson2wkt(mpoly, fmt=2)

mpoly2 <- list(type = "MultiPolygon",
              coordinates = list(list(list(c(30, 20), c(45, 40), c(10, 40), c(30, 20))),
                          list(list(c(15, 5), c(40, 10), c(10, 20), c(5 ,10), c(15, 5))))
)
```

```
geojson2wkt(mpoly2, fmt=1)

# geometrycollection
gmcoll <- list(type = 'GeometryCollection',
 geometries = list(
     list(type = "Point", coordinates = list(0.0, 1.0)),
     list(type = 'LineString', coordinates = list(c(-100.0, 0.0), c(-101.0, -1.0))),
     list(type = 'Polygon',
       'coordinates' = list(list(c(100.001, 0.001),
                        c(101.1235, 0.001),
                        c(101.001, 1.001),
                        c(100.001, 0.001)),
                       list(c(100.201, 0.201),
                        c(100.801, 0.201),
                        c(100.801, 0.801),
                        c(100.201, 0.201)))
     ),
     list(type = 'MultiPoint',
       'coordinates' = list(c(100.0, 3.101), c(101.0, 2.1), c(3.14, 2.18))
     ),
     list(type = 'MultiLineString',
       coordinates = list(list(c(0.0, -1.0), c(-2.0, -3.0), c(-4.0, -5.0)),
                       list(c(1.66, -31023.5, 1.1),
                        c(10000.9999, 3.0, 2.2),
                        c(100.9, 1.1, 3.3),
                        c(0.0, 0.0, 4.4)))
     ),
     list(type = 'MultiPolygon',
          coordinates = list(
            list(list(c(100.001, 0.001),
                      c(101.001, 0.001),
                      c(101.001, 1.001),
                      c(100.001, 0.001)),
                 list(c(100.201, 0.201),
                      c(100.801, 0.201),
                      c(100.801, 0.801),
                      c(100.201, 0.201)) ),
            list(list(c(1.0, 2.0, 3.0, 4.0),
                      c(5.0, 6.0, 7.0, 8.0),
                      c(9.0, 10.0, 11.0, 12.0),
                      c(1.0, 2.0, 3.0, 4.0))))
     )
 )
)
geojson2wkt(gmcoll, fmt=0)

# Convert geojson as character string to WKT
str <- '
{
   "type": "Point",
   "coordinates": [
       -105.01621,
       39.57422
```

```
   ]
}'
geojson2wkt(str)

str <- '{"type":["LineString"],"coordinates":[[0,0,10],[2,1,20],[4,2,30],[5,4,40]]}'
geojson2wkt(str)

# From a jsonlite json object
library("jsonlite")
json <- toJSON(list(type="Point", coordinates=c(-105,39)))
geojson2wkt(json)
```

`geometrycollection` *Make WKT geometrycollection objects*

Description

Make WKT geometrycollection objects

Usage

```
geometrycollection(...)
```

Arguments

...	Character string WKT objects representing a Point, LineString, Polygon, etc.

Details

This is different from the other functions that create WKT from R objects, in that we can't do the same thing for GeometryCollection's since many different WkT object could be created from the same input. So, this function accepts WKT strings already formed and attempts to creat a c GeommetryCollection from them.

See Also

Other R.objects: `circularstring`; `linestring`; `multilinestring`; `multipoint`; `multipolygon`; `point`; `polygon`

Examples

```
## empty geometrycollection
geometrycollection("empty")
# geometrycollection("stuff")

# Character string, returns itself
geometrycollection("GEOMETRYCOLLECTION(POINT(4 6), LINESTRING(4 6, 7 10))")

# From a point
```

```
geometrycollection(point(-116.4, 45.2))

# From two points
geometrycollection(point(-116.4, 45.2), point(-118.4, 49.2))

# From various object types
geometrycollection(point(-116.4, 45.2),
 linestring("LINESTRING (-116.4 45.2, -118.0 47.0)"),
 circularstring(list(c(1, 5), c(6, 2), c(7, 3)), fmt = 2)
)
```

`linestring` *Make WKT linestring objects*

Description

Make WKT linestring objects

Usage

```
linestring(..., fmt = 16)
```

Arguments

`...`	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

See Also

Other R.objects: `circularstring`; `geometrycollection`; `multilinestring`; `multipoint`; `multipolygon`; `point`; `polygon`

Examples

```
## empty linestring
linestring("empty")
# linestring("stuff")

## character string
linestring("LINESTRING (-116.4 45.2, -118.0 47.0)")

# numeric
## 2D
linestring(c(100.000, 0.000), c(101.000, 1.000), fmt=2)
linestring(c(100.0, 0.0), c(101.0, 1.0), c(120.0, 5.00), fmt=2)
## 3D
linestring(c(0.0, 0.0, 10.0), c(2.0, 1.0, 20.0),
```

```
           c(4.0, 2.0, 30.0), c(5.0, 4.0, 40.0), fmt=2)
## 4D
linestring(c(0.0, 0.0, 10.0, 5.0), c(2.0, 1.0, 20.0, 5.0),
           c(4.0, 2.0, 30.0, 5.0), c(5.0, 4.0, 40.0, 5.0), fmt=2)

# data.frame
df <- data.frame(lon=c(-116.4,-118), lat=c(45.2,47))
linestring(df, fmt=1)
df <- data.frame(lon=c(-116.4,-118,-120), lat=c(45.2,47,49))
linestring(df, fmt=1)

# matrix
mat <- matrix(c(-116.4,-118, 45.2, 47), ncol = 2)
linestring(mat, fmt=1)
mat2 <- matrix(c(-116.4, -118, -120, 45.2, 47, 49), ncol = 2)
linestring(mat2, fmt=1)

# list
linestring(list(c(100.000, 0.000), c(101.000, 1.000)), fmt=2)
```

`lint` *Validate WKT strings*

Description

Validate WKT strings

Usage

```
lint(str)
```

Arguments

`str`	A WKT string

Value

A logical (`TRUE` or `FALSE`)

Examples

```
lint("POINT (1 2)")
lint("POINT (1 2 3)")
lint("LINESTRING EMPTY")
lint("LINESTRING (100 0, 101 1)")
lint("MULTIPOINT EMPTY")
lint("MULTIPOINT ((1 2), (3 4))")
lint("MULTIPOINT ((1 2), (3 4), (-10 100))")
lint("POLYGON ((1 2, 3 4, 0 5, 1 2))")
lint("POLYGON((20.3 28.6, 20.3 19.6, 8.5 19.6, 8.5 28.6, 20.3 28.6))")
```

```
lint("MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))")
lint("TRIANGLE ((0 0, 0 1, 1 1, 0 0))")
lint("TRIANGLE ((0.1 0.1, 0.1 1.1, 1.1 1.1, 0.1 0.1))")
lint("CIRCULARSTRING (1 5, 6 2, 7 3)")
lint("CIRCULARSTRING (1 5, 6 2, 7 3, 5 6, 4 3)")
lint('COMPOUNDCURVE (CIRCULARSTRING (1 0, 0 1, -1 0), (-1 0, 2 0))')
```

`multilinestring` *Make WKT multilinestring objects*

Description

Make WKT multilinestring objects

Usage

```
multilinestring(..., fmt = 16)
```

Arguments

`...`	A GeoJSON-like object representing a Point, LineString, Polygon, multilinestring, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

Details

There is no `numeric` input option for multilinestring. There is no way as of yet to make a nested multilinestring with `data.frame` input, but you can do so with list input. See examples.

See Also

Other R.objects: `circularstring`; `geometrycollection`; `linestring`; `multipoint`; `multipolygon`; `point`; `polygon`

Examples

```
## empty multilinestring
multilinestring("empty")
# multilinestring("stuff")

# character string
x <- "MULTILINESTRING ((30 20, 45 40, 10 40), (15 5, 40 10, 10 20))"
multilinestring(x)

# data.frame
df <- data.frame(long = c(30, 45, 10), lat = c(20, 40, 40))
df2 <- data.frame(long = c(15, 40, 10), lat = c(5, 10, 20))
multilinestring(df, df2, fmt=0)
```

```
multilinestring(df, df2, fmt=0) %>% lint
multilinestring(df, df2) %>% wktview(zoom=3)

# matrix
mat <- matrix(c(df$long, df$lat), ncol = 2)
mat2 <- matrix(c(df2$long, df2$lat), ncol = 2)
multilinestring(mat, mat2, fmt=0)

# list
x1 <- list(c(30, 20), c(45, 40), c(10, 40))
x2 <- list(c(15, 5), c(40, 10), c(10, 20))
multilinestring(x1, x2, fmt=2)

polys <- list(
  list(c(30, 20), c(45, 40), c(10, 40)),
  list(c(15, 5), c(40, 10), c(10, 20))
)
multilinestring(polys, fmt=2) %>%
  wktview(zoom=3)
```

`multipoint` *Make WKT multipoint objects*

Description

Make WKT multipoint objects

Usage

```
multipoint(..., fmt = 16)
```

Arguments

`...`	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

See Also

Other R.objects: `circularstring`; `geometrycollection`; `linestring`; `multilinestring`; `multipolygon`; `point`; `polygon`

Examples

```
## empty multipoint
multipoint("empty")
# multipoint("stuff")
```

```
# numeric
multipoint(c(100.000, 3.101), c(101.000, 2.100), c(3.140, 2.180))

# data.frame
df <- us_cities[1:25, c('long', 'lat')]
multipoint(df)

# matrix
mat <- matrix(c(df$long, df$lat), ncol = 2)
multipoint(mat)

# list
multipoint(list(c(100.000, 3.101), c(101.000, 2.100), c(3.140, 2.180)))
```

`multipolygon` *Make WKT multipolygon objects*

Description

Make WKT multipolygon objects

Usage

```
multipolygon(..., fmt = 16)
```

Arguments

`...`	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

Details

There is no `numeric` input option for multipolygon. There is no way as of yet to make a nested multipolygon with `data.frame` input, but you can do so with list input. See examples.

See Also

Other R.objects: `circularstring`; `geometrycollection`; `linestring`; `multilinestring`; `multipoint`; `point`; `polygon`

Examples

```
## empty multipolygon
multipolygon("empty")
# multipolygon("stuff")

# data.frame
```

```
df <- data.frame(long = c(30, 45, 10, 30), lat = c(20, 40, 40, 20))
df2 <- data.frame(long = c(15, 40, 10, 5, 15), lat = c(5, 10, 20, 10, 5))
multipolygon(df, df2, fmt=0)
multipolygon(df, df2, fmt=0) %>% lint
multipolygon(df, df2) %>% wktview(zoom=3)

# matrix
mat <- matrix(c(df$long, df$lat), ncol = 2)
mat2 <- matrix(c(df2$long, df2$lat), ncol = 2)
multipolygon(mat, mat2, fmt=0)

# list
multipolygon(list(c(30, 20), c(45, 40), c(10, 40), c(30, 20)),
  list(c(15, 5), c(40, 10), c(10, 20), c(5, 10), c(15, 5)), fmt=2)

polys <- list(
  list(c(30, 20), c(45, 40), c(10, 40), c(30, 20)),
  list(c(15, 5), c(40, 10), c(10, 20), c(5, 10), c(15, 5))
)
multipolygon(polys, fmt=2) %>%
  wktview(zoom=3)

## nested polygons
polys <- list(
  list(c(40, 40), c(20, 45), c(45, 30), c(40, 40)),
  list(
    list(c(20, 35), c(10, 30), c(10, 10), c(30, 5), c(45, 20), c(20, 35)),
    list(c(30, 20), c(20, 15), c(20, 25), c(30, 20))
  )
)
multipolygon(polys, fmt=0)
multipolygon(polys, fmt=0) %>% lint
```

`point` *Make WKT point objects*

Description

Make WKT point objects

Usage

```
point(..., fmt = 16)
```

Arguments

`...`	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
`fmt`	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

See Also

Other R.objects: circularstring; geometrycollection; linestring; multilinestring; multipoint; multipolygon; polygon

Examples

```
## empty point
point("empty")
# point("stuff")

## single point
point(-116.4, 45.2)
point(0, 1)

## single point, from data.frame
df <- data.frame(lon=-116.4, lat=45.2)
point(df)

## many points, from a data.frame
ussmall <- us_cities[1:5, ]
df <- data.frame(long = ussmall$long, lat = ussmall$lat)
point(df)

## many points, from a matrix
mat <- matrix(c(df$long, df$lat), ncol = 2)
point(mat)

## single point, from a list
point(list(c(100.0, 3.101)))

## many points, from a list
point(list(c(100.0, 3.101), c(101.0, 2.1), c(3.14, 2.18)))
```

---

polygon *Make WKT polygon objects*

Description

Make WKT polygon objects

Usage

```
polygon(..., fmt = 16)
```

Arguments

...	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
fmt	Format string which indicates the number of digits to display after the decimal point when formatting coordinates. Max: 20

Details

You can create nested polygons with `list` and `data.frame` inputs, but not from `numeric` inputs. See examples.

See Also

Other R.objects: `circularstring`; `geometrycollection`; `linestring`; `multilinestring`; `multipoint`; `multipolygon`; `point`

Examples

```
## empty polygon
polygon("empty")
# polygon("stuff")

# numeric
polygon(c(100.001, 0.001), c(101.12345, 0.001), c(101.001, 1.001), c(100.001, 0.001), fmt=2)

# data.frame
## single polygon
df <- us_cities[2:5,c('long','lat')]
df <- rbind(df, df[1,])
polygon(df, fmt=2) %>% wktview(zoom=4)
## nested polygons
df2 <- data.frame(long = c(-85.9, -85.9, -93, -93, -85.9),
                  lat = c(37.5, 35.3, 35.3, 37.5, 37.5))
polygon(df, df2, fmt=2) %>% wktview(zoom=4)

# matrix
mat <- matrix(c(df$long, df$lat), ncol = 2)
polygon(mat)

# list
# single list - creates single polygon
ply <- list(c(100.001, 0.001), c(101.12345, 0.001), c(101.001, 1.001), c(100.001, 0.001))
polygon(ply, fmt=2) %>% wktview(zoom=7)
# nested list - creates nested polygon
polygon(list(c(35, 10), c(45, 45), c(15, 40), c(10, 20), c(35, 10)),
   list(c(20, 30), c(35, 35), c(30, 20), c(20, 30)), fmt=2) %>%
   wktview(zoom=3)
# multiple lists nested within a list
polygon(list(list(c(35, 10), c(45, 45), c(15, 40), c(10, 20), c(35, 10)),
   list(c(20, 30), c(35, 35), c(30, 20), c(20, 30))), fmt=2) %>%
   wktview(zoom=3)
```

`properties` *Add properties to a geojson object*

Description

Add properties to a geojson object

Usage

```
properties(x, style = NULL, popup = NULL)
```

Arguments

x	Input
style	List of color, fillColor, etc., or NULL
popup	Popup text, or NULL

Examples

```
str <- "POINT (-116.4000000000000057 45.2000000000000028)"
x <- wkt2geojson(str)
properties(x, style=list(color = "red"))
```

---

us_cities *This is the same data set from the maps library, named differently*

Description

This database is of us cities of population greater than about 40,000. Also included are state capitals of any population size.

Format

A list with 6 components, namely "name", "country.etc", "pop", "lat", "long", and "capital", containing the city name, the state abbreviation, approximate population (as at January 2006), latitude, longitude and capital status indication (0 for non-capital, 1 for capital, 2 for state capital.

---

wkt2geojson *Convert WKT to GeoJSON-like objects.*

Description

Convert WKT to GeoJSON-like objects.

Usage

```
wkt2geojson(str, fmt = 16, feature = TRUE)
```

Arguments

str	A GeoJSON-like object representing a Point, LineString, Polygon, MultiPolygon, etc.
fmt	Number of digits to display after the decimal point when formatting coordinates.
feature	(logical) Make a feature geojson object. Default: TRUE

Details

Should be robust against a variety of typing errors, including extra spaces between coordinates, no space between WKT type and coordinates. However, some things won't pass, includingn lowercase WKT types, no spaces between coordinates.

See Also

geojson2wkt

Examples

```
# point
str <- "POINT (-116.4000000000000057 45.2000000000000028)"
wkt2geojson(str)
wkt2geojson(str, feature=FALSE)

# multipoint
str <- 'MULTIPOINT ((100.000 3.101), (101.000 2.100), (3.140 2.180))'
wkt2geojson(str, fmt = 2)
wkt2geojson(str, fmt = 2, feature=FALSE)

# polygon
str <- "POLYGON ((100 0.1, 101.1 0.3, 101 0.5, 100 0.1),
   (103.2 0.2, 104.8 0.2, 100.8 0.8, 103.2 0.2))"
wkt2geojson(str)
wkt2geojson(str, feature=FALSE)

# multipolygon
str <- "MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)),
   ((20 35, 45 20, 30 5, 10 10, 10 30, 20 35), (30 20, 20 25, 20 15, 30 20)))"
wkt2geojson(str)
wkt2geojson(str, feature=FALSE)

# linestring
str <- "LINESTRING (100.000 0.000, 101.000 1.000)"
wkt2geojson(str)
wkt2geojson(str, feature=FALSE)
wkt2geojson("LINESTRING (0 -1, -2 -3, -4 5)")
wkt2geojson("LINESTRING (0 1 2 3, 4 5 6 7)")

# multilinestring
str <- "MULTILINESTRING ((30 1, 40 30, 50 20)(10 0, 20 1))"
wkt2geojson(str)
```

```
str <- "MULTILINESTRING (
   (-105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5),
   (-105.0 39.5, -105.0 39.5, -105.0 39.5),
   (-105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5),
   (-105.0 39.5, -105.0 39.5, -105.0 39.5, -105.0 39.5))"
wkt2geojson(str)

# Geometrycollection
str <- "GEOMETRYCOLLECTION (
   POINT (0 1),
   LINESTRING (-100 0, -101 -1),
   POLYGON ((100.001 0.001, 101.1235 0.0010, 101.001 1.001, 100.001 0.001),
           (100.201 0.201, 100.801 0.201, 100.801 0.801, 100.201 0.201)),
   MULTIPOINT ((100.000 3.101), (101.0 2.1), (3.14 2.18)),
   MULTILINESTRING ((0 -1, -2 -3, -4 -5),
                    (1.66 -31023.50 1.10, 10001.0 3.0 2.2, 100.9 1.1 3.3, 0.0 0.0 4.4)),
   MULTIPOLYGON (((100.001 0.001, 101.001 0.001, 101.001 1.001, 100.001 0.001),
                 (100.201 0.201, 100.801 0.201, 100.801 0.801, 100.201 0.201)),
                 ((1 2 3 4, 5 6 7 8, 9 10 11 12, 1 2 3 4))))"
wkt2geojson(str)
```

`wktview` *Visualize geojson from a character string or list*

Description

Visualize geojson from a character string or list

Usage

```
wktview(x, center = NULL, zoom = 5, fmt = 16)
```

Arguments

`x`	Input, a geojson character string or list
`center`	(numeric) A length two vector of the form: `longitude, latitude`
`zoom`	(integer) A number between 1 and 18 (1 zoomed out, 18 zoomed in)
`fmt`	Number of digits to display after the decimal point when formatting coordinates.

Value

Opens a map with the geojson object(s) using `leaflet`

See Also

`as_featurecollection`

Examples

```
## Not run:
# point
str <- "POINT (-116.4000000000000057 45.2000000000000028)"
wktview(str)

# multipoint
df <- us_cities[1:5,c('long','lat')]
str <- multipoint(df)
wktview(str)
wktview(str, center = c(-100,40))
wktview(str, zoom = 3)

# linestring
wktview(linestring(c(100.000, 0.000), c(101.000, 1.000), fmt=2))

# polygon
a <- polygon(list(c(100.001, 0.001), c(101.12345, 0.001), c(101.001, 1.001), c(100.001, 0.001)))
wktview(a)
wktview(a, zoom=9)

## End(Not run)
```

Index