

# Package ‘arcpullr’

September 2, 2021

**Type** Package

**Title** Pull Data from an 'ArcGIS REST' API

**Version** 0.2.1

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Description** Functions to efficiently query 'ArcGIS REST' APIs

<<https://developers.arcgis.com/rest/>>.

Both spatial and SQL queries can be used to retrieve data.

Simple Feature (sf) objects are utilized to perform spatial queries.

This package was neither produced nor is maintained by Esri.

**Depends** R (>= 3.6.0), sf (>= 0.9.7),

**Imports** httr (>= 1.4.1), jsonlite (>= 1.6.1), dplyr (>= 1.0.2),  
ggplot2 (>= 3.3.0), tidyr (>= 1.0.2), rlang (>= 0.4.7), raster  
(>= 3.4.5), methods

**RoxygenNote** 7.1.1

**Suggests** testthat (>= 3.0.1), knitr (>= 1.30), rmarkdown (>= 2.6),  
rvest (>= 0.3.6), xml2 (>= 1.3.2), stringr (>= 1.4.0), cowplot  
(>= 1.1.1), magick (>= 2.5.2)

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Paul Frater [aut, cre] (<<https://orcid.org/0000-0002-7237-6563>>),  
Zac Driscoll [aut] (<<https://orcid.org/0000-0002-8233-0980>>)

**Maintainer** Paul Frater <paul.frater@wisconsin.gov>

**Repository** CRAN

**Date/Publication** 2021-09-02 14:50:54 UTC

## R topics documented:

arcpullr-package . . . . . 2

format_coords . . . . .	3
get_geometry_type . . . . .	4
get_image_layer . . . . .	5
get_layers_by_spatial . . . . .	6
get_layer_html . . . . .	7
get_layer_legend . . . . .	8
get_map_layer . . . . .	8
get_raster_layer . . . . .	10
get_service_type . . . . .	11
get_sf_crs . . . . .	12
get_spatial_layer . . . . .	12
match_raster_colors . . . . .	13
plot_layer . . . . .	14
plot_layer,RasterBrick-method . . . . .	15
plot_layer,RasterLayer-method . . . . .	16
plot_layer,RasterStack-method . . . . .	17
plot_layer,sf-method . . . . .	18
raster_colors . . . . .	18
raster_colors,RasterBrick-method . . . . .	19
raster_colors,RasterLayer-method . . . . .	20
raster_colors,RasterStack-method . . . . .	21
sf_example_polys . . . . .	21
sf_objects . . . . .	22
sp_rel_lookups . . . . .	23
sql_where . . . . .	24
valid_sp_rel . . . . .	25
<b>Index</b>	<b>26</b>

---

arcpullr-package	<i>arcpullr</i>
------------------	-----------------

---

## Description

A package for pulling spatial data from an ArcGIS REST API

## Details

The role of the arcpullr package is simple...to pull spatial data from an ArcGIS REST API. These APIs are housed by various different agencies, organizations, entitites, etc., but allow a consistent format for storing and retrieving spatial data

### [get\\_spatial\\_layer](#)

This function makes up the core of the package. It allows users to pull spatial data given a URL of an ArcGIS REST API. There are many additional query parameters that can (and probably should) be added; however, we've simplified many of these out for you with the functions below.

### [get\\_layer\\_by\\_spatial](#) family of functions

These functions allow you to pull layers using a spatial query. The abstract syntax is wrapped into the functions, so all you have to do is pass these functions an sf object of the spatial area, line, or point you want to query by. These functions include `get_layer_by_poly`, `get_layer_by_point`, `get_layer_by_line`, `get_layer_by_multipoint`, and `get_layer_by_envelope`. It should be fairly obvious what type of spatial layer each function takes with the exception of `get_layer_by_envelope` except that it isn't particularly useful for a single point.

### Helper functions

There are a few utility functions to help you along the way. The first is [plot\\_layer](#), which is a useful way to plot the spatial layer you've tried to pull just to make sure it works. If you want fancier maps you'd be better served with `ggplot2` or `tmaps`, though.

Other helpers include the [sf\\_objects](#) functions, which allow you to easily create sf points, lines, and polygons with a few coordinates.

Lastly, there is a [sql\\_where](#) function to help aid in building more complex SQL WHERE clauses used to query by the where argument in the retrieval functions above.

---

format_coords	<i>Convert coordinates from an 'sf' object to formatted well-known text</i>
---------------	---

---

### Description

Use this function to convert the coordinates of a sf polygon object to a string of well known text. The output can be passed to an ArcGIS REST API to perform a spatial query.

### Usage

```
format_polygon_coords(sf_obj)

format_line_coords(sf_obj)

format_multipoint_coords(sf_obj)

format_point_coords(sf_obj)

format_envelope_coords(sf_obj)

format_coords(sf_obj, geom_type)
```

### Arguments

sf_obj	An sf object
geom_type	Either "points", "paths", or "rings". Choose wisely

**Details**

Spatial queries from an ArcGIS REST API require specific text inputs formatted in a way called well-known text (WKT). ArcGIS REST APIs have their own syntax for how the text is taken. These functions will format sf objects in the correct way to be able to make spatial queries from a ArcGIS REST API

**Value**

String of well known text

**Examples**

```
mke_polygon_coords <- format_polygon_coords(mke_county)
```

---

get_geometry_type	<i>Get Geometry Type</i>
-------------------	--------------------------

---

**Description**

Get Geometry Type

**Usage**

```
get_geometry_type(url)
```

**Arguments**

url                    A character string of a feature services URL

**Value**

A character string of the layers geometry type

**Examples**

```
## Not run:  
get_geometry_type(reykjanes_lava_flow_url)  
  
## End(Not run)
```

---

get_image_layer	<i>Retrieve an image service layer from an ArcGIS REST API</i>
-----------------	--

---

### Description

This function retrieves image service layers from an ArcGIS REST services API and returns them as a RasterStack object

### Usage

```
get_image_layer(
    url,
    sf_object = NULL,
    bbox = NULL,
    token = "",
    clip_raster = TRUE,
    format = "png",
    transparent = TRUE,
    ...
)
```

### Arguments

url	A character string of the url for the layer to pull
sf_object	An sf object used for the bounding box
bbox	Character string of the bounding box
token	A character string of the token (if needed)
clip_raster	Logical. Should the raster be clipped to contain only the pixels that reside in the sf_object? By default, ArcGIS returns some overlapping edge pixels. Setting clip_raster to TRUE (default) will remove these using <a href="#">mask</a> from the raster package
format	The raster format desired. Default is "png"
transparent	Logical. Retrieve a raster with a transparent background (TRUE, default) or not (FALSE)
...	Additional arguments to pass to the ArcGIS REST API

### Details

This is one of the core functions of the package. It retrieves image service layers from an ArcGIS REST API designated by the URL. These layers require a bounding box to query the map layer, which is either taken from the sf\_object argument or optionally can be passed via the bbox argument. Either sf\_object or bbox are optional, but one of them must be present.

All of the querying parameters are sent via a POST request to the URL, so if there are issues with passing additional parameters via ... first determine how they fit into the POST request and make adjustments as needed. This syntax can be tricky if you're not used to it.

**Value**

A "RasterStack" object

**Examples**

```
## Not run:
wi_leaf_off_layer <- get_image_layer(wi_leaf_off_url, wis_poly)
plot_layer(wi_leaf_off_layer, outline_poly = wis_poly)

## End(Not run)
```

---

get\_layers\_by\_spatial *Retrieve ArcGIS REST API spatial layer by spatial query*

---

**Description**

These functions are wrappers around [get\\_spatial\\_layer](#) that are specialized for querying by a spatial layer. They will make a POST request to the query URL which returns data (if available) based on the appropriate spatial feature (geometry) and relationship (sp\_rel).

**Usage**

```
get_layer_by_poly(url, geometry, sp_rel = "esriSpatialRelContains", ...)

get_layer_by_line(url, geometry, sp_rel = "esriSpatialRelIntersects", ...)

get_layer_by_point(url, geometry, sp_rel = "esriSpatialRelIntersects", ...)

get_layer_by_multipoint(
  url,
  geometry,
  sp_rel = "esriSpatialRelIntersects",
  ...
)

get_layer_by_envelope(url, geometry, sp_rel = "esriSpatialRelIntersects", ...)

get_layer_by_spatial(
  url,
  geometry,
  geom_type,
  sp_ref = NULL,
  sp_rel = "esriSpatialRelIntersects",
  ...
)
```

**Arguments**

url	A character string of the url for the layer to pull
geometry	An sf object used for the spatial query
sp_rel	Character. The type of relationship to query by.
...	Additional arguments to pass to <a href="#">get_spatial_layer</a>
geom_type	A character of the geometry type to be used. This param is automatically specified in all get_layer_by_* functions except get_spatial_layer
sp_ref	The spatial reference value

**Value**

An object of class "sf" of the appropriate layer

**Examples**

```
base_wdnr_url <- "https://dnrm.wi.gov/arcgis/rest/services/"
hydro_path <- "WT_SWDV/WT_Inland_Water_Resources_WTM_Ext_v2/MapServer/2"
hydro_url <- paste0(base_wdnr_url, hydro_path)
mke_waters <- get_layer_by_poly(url = hydro_url, mke_county)
```

---

get_layer_html	<i>Pull the HTML body from a web page</i>
----------------	---

---

**Description**

Used internally to pull HTML for a layer's web page so that the call doesn't have to be made twice in [get\\_geometry\\_type](#) if the url provided there is for a raster layer.

**Usage**

```
get_layer_html(url)
```

**Arguments**

url	Character. The URL of the web page
-----	------------------------------------

**Value**

A character string of the HTML body

---

get_layer_legend	Returns a legend for a raster layer
------------------	-------------------------------------

---

### Description

Raster layers are accompanied with legends to identify what the colors mean. This function retrieves those legend values and returns them as a data.frame with the associated RGB color values. This will likely be most useful for plotting and analysis of map layers.

### Usage

```
get_layer_legend(url)
```

### Arguments

url                    A URL to a Map or Image Service layer

### Value

A data.frame with two columns (color, values) and the number of rows equal to the number of values in a layer

### Examples

```
## Not run:  
get_layer_legend(wi_landcover_url)  
  
## End(Not run)
```

---

get_map_layer	Retrieve a map service layer from an ArcGIS REST API
---------------	--

---

### Description

This function retrieves map service layers from an ArcGIS REST services API and returns them as a RasterLayer object

### Usage

```
get_map_layer(  
  url,  
  sf_object = NULL,  
  bbox = NULL,  
  token = "",  
  clip_raster = TRUE,  
  format = "png",
```



```

    transparent = TRUE,
    add_legend = TRUE,
    ...
)

```

### Arguments

url	A character string of the url for the layer to pull
sf_object	An sf object used for the bounding box
bbox	Character string of the bounding box
token	A character string of the token (if needed)
clip_raster	Logical. Should the raster be clipped to contain only the pixels that reside in the sf_object? By default, ArcGIS returns some overlapping edge pixels. Setting clip_raster to TRUE (default) will remove these using <a href="#">mask</a> from the raster package
format	The raster format desired. Default is "png"
transparent	Logical. Retrieve a raster with a transparent background (TRUE, default) or not (FALSE)
add_legend	Logical. Pull legend and match to color values (TRUE, default) or not (FALSE)
...	Additional arguments to pass to the ArcGIS REST API

### Details

This is one of the core functions of the package. It retrieves map service layers from an ArcGIS REST API designated by the URL. These layers require a bounding box to query the map layer, which is either taken from the sf\_object argument or optionally can be passed via the bbox argument. Either sf\_object or bbox are optional, but one of them must be present.

All of the querying parameters are sent via a POST request to the URL, so if there are issues with passing additional parameters via ... first determine how they fit into the POST request and make adjustments as needed. This syntax can be tricky if you're not used to it.

### Value

A "RasterLayer" object

### Examples

```

## Not run:
wi_landcover<- get_map_layer(wi_landcover_url, wis_poly)
plot_layer(wi_landcover, outline_poly = wis_poly)

## End(Not run)

```

---

get_raster_layer	<i>Pull a raster layer from a map service or image service layer of an ArcGIS REST API</i>
------------------	--

---

### Description

This is an internal function to pull raster layers from either a map service or an image service of an ArcGIS REST API. This function is the engine that drives [get\\_map\\_layer](#) and [get\\_image\\_layer](#)

### Usage

```
get_raster_layer(
    url,
    sf_object = NULL,
    bbox = NULL,
    token = "",
    clip_raster = TRUE,
    format = "png",
    transparent = TRUE,
    export_type = "map",
    add_legend = FALSE,
    ...
)
```

### Arguments

url	A character string of the url for the layer to pull
sf_object	An sf object used for the bounding box
bbox	Character string of the bounding box
token	A character string of the token (if needed)
clip_raster	Logical. Should the raster be clipped to contain only the pixels that reside in the sf_object? By default, ArcGIS returns some overlapping edge pixels. Setting clip_raster to TRUE (default) will remove these using <a href="#">mask</a> from the raster package
format	The raster format desired. Default is "png"
transparent	Logical. Retrieve a raster with a transparent background (TRUE, default) or not (FALSE)
export_type	Character. Either "map" or "image" for the respective service layer desired
add_legend	Logical. Pull legend and match to color values (TRUE, default) or not (FALSE)
...	Additional arguments to pass to the ArcGIS REST API

### Value

An object of type RasterLayer if export\_type = "map" or an object of type RasterStack if export\_type = "image"

---

get_service_type	<i>Get elements of a Service or Layer from an ArcGIS REST endpoint</i>
------------------	--

---

### Description

This family of functions is meant to pull attributes from a particular service or layer hosted on an ArcGIS REST API. If the service is an ImageServer or MapServer, then the behavior will be slightly different than for a Feature Layer (see details).

### Usage

```
get_service_type(url, ...)
```

### Arguments

url	A character string of a valid layer URL
...	Only used internally, but html can be passed

### Details

get\_service\_type will return the type of service or layer for the respective URL (or html) that is passed to the function. For a feature layer the function should return "feature\_layer", for a Image or Map Server the function will return "image" or "map", respectively.

get\_geometry\_type will return the geometry type of feature service layers housed on an ArcGIS REST API server. If a URL is provided that points to a map or image layer the function will return an error (i.e. only feature layers have geometry types).

get\_supported\_operations will simply return a character vector that lists the supported operations for url.

### Value

A character string defining the layer type

### Examples

```
## Not run:  
get_service_type(reykjanes_lava_flow_url)  
  
## End(Not run)
```

---

get_sf_crs	<i>Return CRS value of an sf object</i>
------------	---

---

**Description**

Return CRS value of an sf object

**Usage**

```
get_sf_crs(sf_obj)
```

**Arguments**

sf\_obj            An object of class sf

**Value**

A numeric value referring to the coordinate reference system

**Examples**

```
get_sf_crs(iceland_poly)
```

---

get_spatial_layer	<i>Retrieve a feature service layer from an ArcGIS REST API</i>
-------------------	---

---

**Description**

This function retrieves spatial layers present in Feature Service layers of an ArcGIS REST services API and returns them as an sf object

**Usage**

```
get_spatial_layer(  
  url,  
  out_fields = c("*"),  
  where = "1=1",  
  token = "",  
  sf_type = NULL,  
  ...  
)
```

**Arguments**

url	A character string of the url for the layer to pull
out_fields	A character string of the fields to pull for each layer
where	A character string of the where condition. Default is 1=1
token	A character string of the token (if needed)
sf_type	A character string specifying the layer geometry to convert to sf ("esriGeometryPolygon", "esriGeometryPoint", "esriGeometryPolyline"), if NULL (default) the server will take its best guess
...	Additional arguments to pass to the ArcGIS REST POST request

**Details**

This is one of the core functions of this package. It retrieves spatial layers from feature services of an ArcGIS REST API designated by the URL. Additional querying features can be passed such as a SQL WHERE statement (where argument) or spatial queries as well as any other types of queries that the ArcGIS REST API accepts (using ...). However, for easier spatial querying see [get\\_layers\\_by\\_spatial](#).

All of the querying parameters are sent via a POST request to the URL, so if there are issues with passing additional parameters via ... first determine how they fit into the POST request and make adjustments as needed. This syntax can be tricky if you're not used to it.

**Value**

An object of class "sf" of the appropriate layer

**Examples**

```
## Not run:
# lava flows on Reykjanes (pr. 'rake-yah-ness') peninsula in Iceland
lava_flows <- get_spatial_layer(reykjanes_lava_flow_url)
plot_layer(lava_flows, outline_poly = reykjanes_poly)
plot_layer(lava_flows, outline_poly = iceland_poly)

## End(Not run)
```

---

match\_raster\_colors     *Match colors in RasterLayer color space to the provided legend values*

---

**Description**

Colors provided by the legend do not always correspond exactly with the colors in the colortable of a RasterLayer object. They are usually pretty close, though, so this function finds the closest colors, maps them to the appropriate colors in the Raster\* object, and applies that to the legend.

**Usage**

```
match_raster_colors(legend, x)
```

## Arguments

legend            An object of class raster\_legend as returned by [get\\_layer\\_legend](#)  
x                 A RasterLayer object as returned by [get\\_map\\_layer](#)

## Details

Raster colors in x are mapped to those in legend by converting the RGB hexadecimal values to a 3D vector of values for red, green and blue. The closest values are then assigned using 3D Pythagorean theorem to compute the distance among all colors. The minimum distance in three dimensional space is the color in x that gets mapped to the appropriate color in legend.

## Value

A raster\_legend object with corrected colors to match those in x

## Examples

```
## Not run:  
wi_landcover <- get_map_layer(wi_landcover_url, wis_poly)  
legend <- get_layer_legend(wi_landcover_url)  
new_legend <- match_raster_colors(legend, wi_landcover_url)  
  
## End(Not run)
```

---

plot\_layer

*Plot a spatial layer*

---

## Description

This function plots a spatial layer as returned from [get\\_spatial\\_layer](#).

## Usage

```
plot_layer(x, ...)  
  
plot_layer.sf(  
  x,  
  outline_poly = NULL,  
  outline_size = 1.2,  
  outline_color = "gray30",  
  plot_pkg = "ggplot",  
  ...  
)
```

**Arguments**

x	An sf or Raster* object as returned from a get_*_layer function
...	Additional arguments to plot_layer
outline_poly	Optional. An sf polygon to outline sf_data for context
outline_size	Numeric argument that controls width of parameter
outline_color	A character vector of a valid color
plot_pkg	Character. The plotting environment to use. Either "ggplot" (default) or "base"

**Value**

Either a ggplot object, or simply plots x if plot\_pkg = "base"

**Examples**

```
## Not run:
plot_layer(iceland_poly)
plot_layer(portage_county, outline_poly = wis_poly)

## End(Not run)
```

---

plot\_layer,RasterBrick-method  
*Plot a RasterBrick object*

---

**Description**

Plot a RasterBrick object

**Usage**

```
## S4 method for signature 'RasterBrick'
plot_layer(
  x,
  outline_poly = NULL,
  outline_size = 1.2,
  outline_color = "gray30",
  plot_pkg = "ggplot",
  ...
)
```

**Arguments**

x	An sf or Raster* object as returned from a get_*_layer function
outline_poly	Optional. An sf polygon to outline sf_data for context
outline_size	Numeric argument that controls width of parameter
outline_color	A character vector of a valid color
plot_pkg	Character. The plotting environment to use. Either "ggplot" (default) or "base"
...	Additional arguments to plot_layer

**Examples**

```
## Not run:
wi_aerial <- get_map_layer(wi_leaf_off_url, wis_poly)
plot_layer(wi_aerial, outline_poly = wis_poly)

## End(Not run)
```

---

plot\_layer,RasterLayer-method

*Plot a RasterLayer object*

---

**Description**

Plot a RasterLayer object

**Usage**

```
## S4 method for signature 'RasterLayer'
plot_layer(
  x,
  outline_poly = NULL,
  outline_size = 1.2,
  outline_color = "gray30",
  legend = TRUE,
  plot_pkg = "ggplot",
  ...
)
```

**Arguments**

x	An sf or Raster* object as returned from a get_*_layer function
outline_poly	Optional. An sf polygon to outline sf_data for context
outline_size	Numeric argument that controls width of parameter
outline_color	A character vector of a valid color
legend	Logical. Only valid when plotting RasterLayers retrieved from <a href="#">get_map_layer</a> where legend was also retrieved
plot_pkg	Character. The plotting environment to use. Either "ggplot" (default) or "base"
...	Additional arguments to plot_layer



**Examples**

```
## Not run:
wi_landcover <- get_map_layer(wi_landcover_url, wis_poly)
plot_layer(wi_landcover, outline_poly = wis_poly)

## End(Not run)
```

---

plot\_layer,RasterStack-method

*Plot a RasterStack object*


---

**Description**

Plot a RasterStack object

**Usage**

```
## S4 method for signature 'RasterStack'
plot_layer(
  x,
  outline_poly = NULL,
  outline_size = 1.2,
  outline_color = "gray30",
  plot_pkg = "ggplot",
  ...
)
```

**Arguments**

x	An sf or Raster* object as returned from a get_*_layer function
outline_poly	Optional. An sf polygon to outline sf_data for context
outline_size	Numeric argument that controls width of parameter
outline_color	A character vector of a valid color
plot_pkg	Character. The plotting environment to use. Either "ggplot" (default) or "base"
...	Additional arguments to plot_layer

**Examples**

```
## Not run:
wi_aerial <- get_map_layer(wi_leaf_off_url, wis_poly)
plot_layer(wi_aerial, outline_poly = wis_poly)

## End(Not run)
```

---

plot\_layer, sf-method *Plot an sf object*

---

### Description

Plot an sf object

### Usage

```
## S4 method for signature 'sf'
plot_layer(
  x,
  outline_poly = NULL,
  outline_size = 1.2,
  outline_color = "gray30",
  plot_pkg = "ggplot",
  ...
)
```

### Arguments

x	An sf or Raster* object as returned from a get_*_layer function
outline_poly	Optional. An sf polygon to outline sf_data for context
outline_size	Numeric argument that controls width of parameter
outline_color	A character vector of a valid color
plot_pkg	Character. The plotting environment to use. Either "ggplot" (default) or "base"
...	Additional arguments to plot_layer

### Examples

```
## Not run:
plot_layer(wis_poly)

## End(Not run)
```

---

raster_colors	<i>Convert RasterLayer into data.frame of colors for each pixel that can be used for plotting</i>
---------------	---

---

### Description

This function is used internally by [plot\\_layer](#) to convert a Raster\* object to a data.frame of colors for each pixel that can be used for plotting with ggplot2

**Usage**

```
raster_colors(x)
```

**Arguments**

x                    A Raster\* object

**Value**

A data.frame with 3 columns and `length(raster_object)` rows. Two of these columns are the x-y coordinates of each pixel, and one is a value for color that can be used for plotting

**Examples**

```
## Not run:
wi_landcover <- get_map_layer(wi_landcover_url, wis_poly)
wi_landcover_data <- raster_colors(wi_landcover)
head(wi_landcover_data)

## End(Not run)
```

---

raster\_colors,RasterBrick-method

*Convert RasterBrick into data.frame of colors that can be used for plotting*

---

**Description**

This function is used internally by [plot\\_layer](#) to convert a RasterBrick object to a data.frame of colors for each pixel that can be used for plotting with ggplot2. Note that this function assumes that the first three bands in the RasterBrick objects are the RGB values and all additional bands are ignored.

**Usage**

```
## S4 method for signature 'RasterBrick'
raster_colors(x)
```

**Arguments**

x                    A RasterBrick object

**Value**

A data.frame with 3 columns and `length(raster_object)` rows

## Examples

```
## Not run:
wi_leaf_off_layer <- get_image_layer(wi_leaf_off_url, wis_poly)
wi_leaf_off_data <- raster_colors(wi_leaf_off_layer)

## End(Not run)
```

---

raster\_colors,RasterLayer-method

*Convert RasterLayer into data.frame of colors that can be used for plotting*

---

## Description

This function is used internally by [plot\\_layer](#) to convert a RasterLayer object to a data.frame of colors for each pixel that can be used for plotting with ggplot2

## Usage

```
## S4 method for signature 'RasterLayer'
raster_colors(x)
```

## Arguments

x                    A RasterLayer object

## Value

A data.frame with 3 columns and length(raster\_object) rows

## Examples

```
## Not run:
wi_landcover <- get_map_layer(wi_landcover_url, wis_poly)
wi_landcover_data <- raster_colors(wi_landcover)

## End(Not run)
```

---

raster\_colors,RasterStack-method

*Convert RasterStack into data.frame of colors that can be used for plotting*

---

### Description

This function is used internally by [plot\\_layer](#) to convert a RasterStack object to a data.frame of colors for each pixel that can be used for plotting with ggplot2. Note that this function assumes that the first three bands in the RasterStack objects are the RGB values and all additional bands are ignored.

### Usage

```
## S4 method for signature 'RasterStack'  
raster_colors(x)
```

### Arguments

x                    A RasterStack object

### Value

A data.frame with 3 columns and length(raster\_object) rows

### Examples

```
## Not run:  
wi_leaf_off_layer <- get_image_layer(wi_leaf_off_url, wis_poly)  
wi_leaf_off_data <- raster_colors(wi_leaf_off_layer)  
  
## End(Not run)
```

---

sf\_example\_polys

*Various example sf polygons*

---

### Description

These are sf polygons that are used for examples throughout the package

**Usage**

```
iceland_poly  
mke_county  
portage_county  
reykjanes_poly  
wis_counties  
wis_poly
```

**Format**

An object of class `sf` and `data.frame`:

An object of class `sf` (inherits from `data.frame`) with 1 rows and 3 columns.

An object of class `sf` (inherits from `data.frame`) with 1 rows and 3 columns.

An object of class `sf` (inherits from `data.frame`) with 1 rows and 2 columns.

An object of class `sf` (inherits from `data.frame`) with 72 rows and 3 columns.

An object of class `sf` (inherits from `data.frame`) with 1 rows and 2 columns.

**Source**

[map\\_data](#)

---

sf\_objects

*Create sf objects from coordinates*

---

**Description**

These are simple wrapper functions for creating `sf` objects from points

**Usage**

```
sf_line(..., crs = 4326)  
sf_point(..., crs = 4326)  
sf_points(..., crs = 4326)  
sf_polygon(..., crs = 4326)
```

**Arguments**

...                   The coordinates of the object  
 crs                   The coordinate reference system. Defaults to 4326

**Value**

An sf object of the appropriate type

**Examples**

```
pt_a <- c(-90, 45)
pt_b <- c(-89, 44)
pt <- sf_points(pt_a)
line <- sf_line(pt_a, pt_b)
```

---

sp_rel_lookups	<i>Spatial relationship descriptor and lookup tables</i>
----------------	--

---

**Description**

These data.frames are used to lookup and explain which spatial relation types go with different spatial queries.

**Usage**

```
sp_rel_valid
sp_rel_lookup
```

**Format**

sp\_rel\_valid is a data.frame with 105 rows and 3 variables as follows:

**feature\_class** A feature class to be queried

**query\_feature\_class** The feature class used to do a spatial query

**sp\_rel** The spatial relationships that are valid for the feature class and query\_feature\_class combination

sp\_rel\_lookup is a data.frame with 9 rows and 2 variables as follows:

**sp\_rel** The spatial relationship being described

**description** A description of the sp\_rel

**Details**

sp\_rel\_lookup explains the various different types of spatial relationships available through ArcGIS REST APIs. sp\_rel\_valid shows which spatial relationships are valid with different geometry types being queried and used to do spatial queries

**Source**

sp\_rel\_valid–Independent tests done specifically by and for arcpullr

sp\_rel\_lookup – <https://help.arcgis.com/en/webapi/wpf/apiref/ESRI.ArcGIS.Client~ESRI.ArcGIS.Client.Tasks.SpatialRelationship.html>

---

sql\_where

*Format a SQL where clause from arguments*

---

**Description**

This function will create a where statement that is compatible with `get_spatial_layer`). This statement can then be passed to the where argument in this function.

**Usage**

```
sql_where(..., rel_op = "=")
```

**Arguments**

...            Named objects to be queried by

rel\_op        Character. The relational operator in the SQL clause (i.e. "=", "IN", "NOT IN", etc.). If a single rel\_op is provide with multiple ... parameters then it will be recycled `length(...)` times.

**Value**

A character string that can be passed to the where argument of `get_spatial_layer`

**Examples**

```
## Not run:
wbics <- sql_where(WATERBODY_WBIC = c(805400, 804600), rel_op = "IN")
base_wdnr_url <- "https://dnrmaps.wi.gov/arcgis/rest/services/"
hydro_path <- "WT_SWDV/WT_Inland_Water_Resources_WTM_Ext_v2/MapServer/3"
hydro_url <- paste0(base_wdnr_url, hydro_path)
lakes <- get_spatial_layer(url = hydro_url, where = wbics)
plot_layer(lakes)

## End(Not run)
```



---

valid_sp_rel	<i>Check to see which spatial relation types are applicable to the feature classes being queried and the sf objects use do to a spatial query</i>
--------------	---

---

**Description**

Check to see which spatial relation types are applicable to the feature classes being queried and the sf objects use do to a spatial query

**Usage**

```
valid_sp_rel(fc1, fc2, pull = TRUE)
```

**Arguments**

fc1	Character. The feature class type being queried. Available options are "point", "multipoint", "line", or "area".
fc2	Character. The geometry type of the sf object used to do a spatial query. Available options are "point", "multipoint", "line", or "area".
pull	Logical. Pull the available options (TRUE) or print all columns of the sp_rel_valid data.frame for the appropriate fc1 and fc2

**Value**

Either a vector or filtered data.frame showing the appropriate sp\_rels for the given feature classes

**Examples**

```
valid_sp_rel("line", "line")
```

# Index

## \* datasets

- sf\_example\_polys, 21
- sp\_rel\_lookups, 23
  
- arcpullr (arcpullr-package), 2
- arcpullr-package, 2
  
- format\_coords, 3
- format\_envelope\_coords (format\_coords), 3
- format\_line\_coords (format\_coords), 3
- format\_multipoint\_coords (format\_coords), 3
- format\_point\_coords (format\_coords), 3
- format\_polygon\_coords (format\_coords), 3
  
- get\_geometry\_type, 4, 7
- get\_image\_layer, 5, 10
- get\_layer\_by\_envelope (get\_layers\_by\_spatial), 6
- get\_layer\_by\_line (get\_layers\_by\_spatial), 6
- get\_layer\_by\_multipoint (get\_layers\_by\_spatial), 6
- get\_layer\_by\_point (get\_layers\_by\_spatial), 6
- get\_layer\_by\_poly (get\_layers\_by\_spatial), 6
- get\_layer\_by\_spatial, 3
- get\_layer\_by\_spatial (get\_layers\_by\_spatial), 6
- get\_layer\_html, 7
- get\_layer\_legend, 8, 14
- get\_layers\_by\_spatial, 6, 13
- get\_map\_layer, 8, 10, 14, 16
- get\_raster\_layer, 10
- get\_service\_type, 11
- get\_sf\_crs, 12
- get\_spatial\_layer, 2, 6, 7, 12, 14, 24
  
- iceland\_poly (sf\_example\_polys), 21
  
- map\_data, 22
- mask, 5, 9, 10
- match\_raster\_colors, 13
- mke\_county (sf\_example\_polys), 21
  
- plot\_layer, 3, 14, 18–21
- plot\_layer, RasterBrick-method, 15
- plot\_layer, RasterLayer-method, 16
- plot\_layer, RasterStack-method, 17
- plot\_layer, sf-method, 18
- portage\_county (sf\_example\_polys), 21
  
- raster\_colors, 18
- raster\_colors, RasterBrick-method, 19
- raster\_colors, RasterLayer-method, 20
- raster\_colors, RasterStack-method, 21
- reykjanes\_poly (sf\_example\_polys), 21
  
- sf\_example\_polys, 21
- sf\_line (sf\_objects), 22
- sf\_objects, 3, 22
- sf\_point (sf\_objects), 22
- sf\_points (sf\_objects), 22
- sf\_polygon (sf\_objects), 22
- sp\_rel\_lookup (sp\_rel\_lookups), 23
- sp\_rel\_lookups, 23
- sp\_rel\_valid (sp\_rel\_lookups), 23
- sql\_where, 3, 24
  
- valid\_sp\_rel, 25
  
- wis\_counties (sf\_example\_polys), 21
- wis\_poly (sf\_example\_polys), 21