Package 'rdwplus'

October 14, 2022

Title An Implementation of IDW-PLUS

Version 0.1.0

Author Alan Pearse [aut, cre],
Grace Heron [aut],
Erin Peterson [aut]

Maintainer Alan Pearse <apearse9@gmail.com>

Description Compute spatially explicit land-use metrics for stream survey sites in GRASS GIS and R as an open-source implementation of IDW-PLUS (Inverse Distance Weighted Percent Land Use for Streams). The package includes functions for preprocessing digital elevation and streams data, and one function to compute all the spatially explicit land use metrics described in Peterson et al. (2011) <doi:10.1111/j.1365-2427.2010.02507.x> and previously implemented by Peterson and Pearse (2017) <doi:10.1111/1752-1688.12558> in ArcGIS-Python as IDW-PLUS.

Depends R (>= 3.5.0), raster, rgrass7

Imports methods, utils

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-07 11:00:02 UTC

R topics documented:

`burn_in` *Burn in streams to a digital elevation model*

Description

Burning-in streams (also called 'drainage reinforcement') ensures flow direction and accumulation grids based on the digital elevation model will correctly identify the stream path.

Usage

```
burn_in(dem, stream, out, burn = 10, overwrite = FALSE)
```

Arguments

`dem`	Digital elevation model raster in the GRASS mapset.
`stream`	Binary stream raster in the GRASS mapset.
`out`	Name of output to be created in the GRASS mapset.
`burn`	The magnitude of the drainage reinforcement in elevation units. Defaults to `10` elevation units.
`overwrite`	A logical indicating whether the file `out` should be overwritten in the mapset and on disk. Defaults to `FALSE`.

Value

Nothing. A raster with the name `out` will be written to the current GRASS mapset.

Examples

```
# Will only run if a GRASS session is initialised
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")
set_envir(dem)
raster_to_mapset(dem)
vector_to_mapset(stream_shp)

# Create binary stream
rasterise_stream("streams", "streams_rast")
reclassify_streams("streams_rast", "streams_binary", out_type = "binary")

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary",
out = "dem_burn", burn = 10, overwrite = FALSE)

# Plot
plot_GRASS("dem_burn", col = topo.colors(10))
}
```

---

check_running *Check whether a valid GRASS session is running*

Description

This function is mostly used internally by other functions in the package. However, users may call this function to check whether they have correctly established a GRASS session prior to using the other functions in the package.

Usage

```
check_running()
```

Value

A logical. The logical indicates whether a valid GRASS session is currently running.

Examples

```
check_running()
```

compute_metrics *Compute spatially explicit watershed attributes for survey sites on streams*

Description

Workhorse function for `rdwplus`. This function computes the spatially explicit landuse metrics in IDW-Plus (Peterson and Pearse, 2017).

Usage

```
compute_metrics(
  metrics = c("iFLO", "iFLS", "HAiFLO", "HAiFLS"),
  landuse,
  sites,
  elevation,
  flow_dir,
  flow_acc,
  streams,
  idwp = -1,
  max_memory = 300,
  lessmem = FALSE
)
```

Arguments

`metrics`	A character vector. This vector specifies which metric(s) should be calculated. Your options are lumped, iFLO, iFLS, iEDO, iEDS, HAiFLO and HAiFLS. The default is to calculate all except for lumped, iEDO and iEDS.
`landuse`	Names of binary landuse or landcover rasters in the current GRASS mapset for which spatially explicit watershed metrics should be computed. Binary means land use cells are coded 1 and all other cells are given a value of 0.
`sites`	A shapefile of sites; either a file path to the shapefile or a `SpatialPoints*` object.
`elevation`	File name of a filled (hydrologically-conditioned) digital elevation model in the current GRASS mapset.
`flow_dir`	A 'Deterministic 8' (D8) flow direction grid derived from `derive_flow`.
`flow_acc`	File name of a flow accumulation grid derived from `derive_flow` in the current GRASS mapset.
`streams`	File name of a streams raster in the current GRASS mapset. Optional if you are not asking for the iFLS, iEDS, and/or HAiFLS metrics.
`idwp`	The inverse distance weighting parameter. Default is `-1`.
`max_memory`	Max memory used in memory swap mode (MB). Defaults to `300`.
`lessmem`	A logical indicating whether to use the less memory modified watershed module. Defaults to `FALSE`.

Value

A `data.frame` object, which is a table with rows corresponding to those from the `sites` argument plus columns for each combination of land use and metric type.

References

Peterson, E.E. & Pearse, A.R. (2017). IDW-Plus: An ArcGIS toolset for calculating spatially explicit watershed attributes for survey sites. *JAWRA*, *53*(5), 1241-1249.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
landuse <- system.file("extdata", "landuse.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data into GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem, landuse), as_integer = c(FALSE, TRUE),
overwrite = TRUE)
vector_to_mapset(vectors = c(sites, stream_shp), overwrite = TRUE)

# Create binary stream
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
out = "dem_burn.tif", burn = 10, overwrite = TRUE)

# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)

# Derive flow accumulation and direction grids
derive_flow(dem = "dem_fill.tif", flow_dir = "fdir.tif",
flow_acc = "facc.tif", overwrite = TRUE)

# Snap sites to pour points (based on flow accumulation)
out_snap <- paste0(tempdir(), "/snapsite.shp")
snap_sites(sites = "site", flow_acc = "facc.tif", max_move = 2,
out = out_snap, overwrite = TRUE)

# Compute metrics
lu_metrics <- compute_metrics(metrics = c("iFLO", "iFLS", "HAiFLO", "HAiFLS"),
                              landuse = "landuse.tif",
                              sites =  out_snap,
                              elevation = "dem_fill.tif",
```

```
                              flow_dir = "fdir.tif",
                              flow_acc = "facc.tif",
                              streams = "streams_rast.tif")
print(lu_metrics)
}
```

---

convert_to_integer *Convert a raster to integer format*

Description

Given a raster in float, double or any other format, this function will convert it to integer format. This can be important because it is often an unstated requirement of GRASS modules such as the one for zonal statistics.

Usage

```
convert_to_integer(x)
```

Arguments

x	A raster layer in the current GRASS mapset.

Value

Nothing. A raster with the same name as x (it may overwrite it) but without the file extension, if one exists.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters
set_envir(dem)
raster_to_mapset(rasters = dem)
vector_to_mapset(vectors = stream_shp)

# Rasterise the streams
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)

# Convert to integer
convert_to_integer("streams_rast.tif")
}
```

coord_to_raster *Turn coordinates of outlets into rasters*

Description

Given a set of coordinates in space (x, y), this function will return a rasterised version of that point in space.

Usage

```
coord_to_raster(outlet, out, overwrite = FALSE)
```

Arguments

outlet	A single pair of Easting, Northing or long, lat coordinates as a numeric vector.
out	The file name of the output outlet raster in the current GRASS mapset.
overwrite	Whether the output files should be allowed to overwrite existing files. Defaults to FALSE.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")

# Set environment parameters
set_envir(dem)

# Read in data
raster_to_mapset(dem)

# Set coordinates to rasterise
coord_df <-  c(1098671, 6924794)

# Convert to raster
coord_to_raster(outlet = coord_df, out = "coords", overwrite = TRUE)

# Plot
plot_GRASS("dem.tif", col = topo.colors(15))
plot_GRASS("coords", col = "red", add = TRUE)
}
```

`derive_flow` *Obtain flow direction and accumulation over a digital elevation model (DEM)*

Description

This function computes flow direction and accumulation (among other things) from a DEM. This is done using the `r.watershed` tool in GRASS.

Usage

```
derive_flow(dem, flow_dir, flow_acc, overwrite = FALSE, max_memory = 300, ...)
```

Arguments

`dem`	A digital elevation model that has been hydrologically corrected.
`flow_dir`	The name of the output flow direction file in the current GRASS mapset.
`flow_acc`	The name of the output flow accumulation file in the current GRASS mapset.
`overwrite`	Whether any of the outputs should be allowed to overwrite existing files.
`max_memory`	Max memory used in memory swap mode (MB). Defaults to `300`.
`...`	Additional arguments to `r.watershed`.

Value

Nothing. Files are written in the current GRASS mapset.

Examples

```
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem), as_integer = c(FALSE))
vector_to_mapset(vectors = c(stream_shp))

# Create binary stream
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif", out = "dem_burn.tif",
 burn = 10, overwrite = TRUE)
```

```
# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)

# Derive flow accumulation and direction grids
derive_flow(dem = "dem_fill.tif",
flow_dir = "fdir.tif",
flow_acc = "facc.tif",
overwrite = TRUE)

# Plot
plot_GRASS("fdir.tif", col = topo.colors(15))
plot_GRASS("facc.tif", col = topo.colors(15))
}
```

`derive_streams` *Extract streams from a flow accumulation raster*

Description

Derive a set of stream lines from a flow accumulation raster.

Usage

```
derive_streams(
  dem,
  flow_acc,
  out,
  min_acc = 1000,
  min_length = 0,
  overwrite = FALSE,
  ...
)
```

Arguments

`dem`	Name of the elevation raster in the current GRASS mapset.
`flow_acc`	Name of the flow accumulation raster in the current GRASS mapset.
`out`	File path to the output vector dataset of stream lines. Should be WITHOUT .shp extension.
`min_acc`	The minimum accumulation value (in upstream cells) that a cell needs to have in order to be classified as a stream. Defaults to `1000`.
`min_length`	The minimum length of a stream segment in cells. Defaults to `0`.
`overwrite`	A logical indicating whether the output should be allowed to overwrite existing files. Defaults to `FALSE`.
`...`	Additional arguments to `r.stream.extract`.

Value

Nothing. A vector dataset with the name `basename(out)` will appear in the current GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters
set_envir(dem)

# Set
raster_to_mapset(rasters = c(dem), as_integer = c(FALSE))
vector_to_mapset(vectors = c(stream_shp))

# Create binary stream
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
 out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
out = "dem_burn.tif", burn = 10, overwrite = TRUE)

# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)

# Derive flow accumulation and direction grids
derive_flow(dem = "dem_fill.tif", flow_dir = "fdir.tif",
flow_acc = "facc.tif", overwrite = TRUE)

# Derive streams
derive_streams(dem = "dem_fill.tif", flow_acc = "facc.tif", out = "stream_lines", overwrite = TRUE)
}
```

`fill_sinks` *Fill sinks in a digital elevation model (DEM)*

Description

A sink is a depression in a DEM. Water flows into these depressions but not out of them. These depressions, although often real features of landscapes, are problematic for flow direction and accumulation algorithms. Therefore, it is common practice to remove these depressions. This function removes depressions (sinks) in a DEM. Note that this function calls `r.hydrodem`, which is a GRASS GIS add-on. It can be installed through the GRASS GUI.

Usage

```
fill_sinks(dem, out, flags, overwrite = FALSE, max_memory = 300, ...)
```

Arguments

dem	The name of a DEM in the current GRASS mapset.
out	Name of the output, which is a hydrologically corrected (sink-filled) DEM.
flags	Optional. A vector of flags that should be passed to r.hydrodem. See details for more on the possible flags.
overwrite	A logical indicating whether the output should be allowed to overwrite existing files. Defaults to FALSE.
max_memory	Max memory used in memory swap mode (MB). Defaults to 300.
...	Optional additional parameters to r.hydrodem.

Details

The following flags may be supplied (as strings):

- "a": The nuclear option. Vigorously remove all sinks.
- "verbose": Lots of module output.
- "quiet": Barely any module output.

Value

Nothing. A file with the name out will be created in the current GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
landuse <- system.file("extdata", "landuse.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem, landuse), as_integer = c(FALSE, TRUE))
vector_to_mapset(vectors = c(sites, stream_shp))

# Create binary stream
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
 out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
out = "dem_burn.tif", burn = 10, overwrite = TRUE)
```

```
# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)

# Plot
plot_GRASS("dem_fill.tif", col = topo.colors(15))
}
```

`get_distance` *Compute Euclidean distance to a survey site or stream line within a watershed*

Description

This function is needed to compute Euclidean distance from a feature of interest in a watershed raster.

Usage

```
get_distance(target, out, overwrite = FALSE)
```

Arguments

`target`	File name of the watershed outlet or streams (as a raster) in the current GRASS mapset.
`out`	File path for the result to be written.
`overwrite`	A logical indicating whether the outputs of this function should be allowed to overwrite existing files.

Value

Nothing. A file with the name `basename(out)` will be created in the current GRASS mapset.

`get_flow_length` *Derive a flow length to streams and outlets*

Description

Given a (hydrologically corrected, see `fill_sinks`) DEM, this function produces a flow accumulation grid which shows the upstream area that flows into each cell in the DEM. Note that this function calls `r.stream.distance`, which is a GRASS GIS add-on. It can be installed through the GRASS GUI.

Usage

```
get_flow_length(
  str_rast,
  flow_dir,
  out,
  to_outlet = FALSE,
  overwrite = FALSE,
  max_memory = 300
)
```

Arguments

str_rast	Rasterized unary streams file.
flow_dir	Flow direction raster.
out	A file name for the output raster of flow lengths.
to_outlet	Calculate parameters for outlets flag. Defaults to FALSE for streams.
overwrite	Overwrite flag. Defaults to FALSE.
max_memory	Max memory used in memory swap mode (MB). Defaults to 300.

Value

Nothing. A file with the name out will be written to GRASS's current workspace.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = dem, as_integer = FALSE)
vector_to_mapset(vectors = c(sites, stream_shp))

# Create binary stream
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
 out = "dem_burn.tif", burn = 10, overwrite = TRUE)

# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)
```

```
# Derive flow accumulation and direction grids
derive_flow(dem = "dem_fill.tif", flow_dir = "fdir.tif",
 flow_acc = "facc.tif", overwrite = TRUE)

# Snap sites to pour points (based on flow accumulation)
snap_sites(sites = "site", flow_acc = "facc.tif", max_move = 2,
out = "snapsite.shp", overwrite = TRUE)

# Get pour points / outlets as raster cells
rowID <- 1
site_coords <- readVECT("site")
coords_i <- site_coords@coords[rowID, 1:2]
coords_i_out <- paste0("pour_point_", rowID)
coord_to_raster(coords_i, coords_i_out, TRUE)

# Name for flow length raster
current_flow_out <- paste0("flowlenOut_", rowID, ".tif")

# Create it
get_flow_length(str_rast = coords_i_out,
flow_dir = "fdir.tif",
out = current_flow_out,
to_outlet = TRUE,
overwrite = TRUE)

# Plot
plot_GRASS(current_flow_out, col = topo.colors(15))
}
```

`get_watershed` *Delineate watershed for a survey site*

Description

This function delineates a watershed around the `i`th site from a set of survey sites.

Usage

```
get_watershed(
  sites,
  i,
  flow_dir,
  out,
  write_file = FALSE,
  overwrite = FALSE,
  lessmem = FALSE
)
```

Arguments

sites	A file path to a shapefile of points or a SpatialPoints* object.
i	An integer which indexes one row of the sites' attribute table.
flow_dir	The name of a flow direction grid in the current GRASS mapset.
out	The name of the output raster. A raster with the name basename(out) will be imported into the GRASS mapset. If write_file is true, then a file with the name out will be written into the user's current working directory.
write_file	A logical indicating whether the output file should be stored as a file in the user's current working directory. Defaults to FALSE.
overwrite	A logical indicating whether the output should be allowed to overwrite existing files. Defaults to FALSE.
lessmem	A logical indicating whether to use the less memory modified watershed module. Defaults to FALSE.

Value

Nothing. A raster file with the name out may be written to file if you have set the write_file argument accordingly. A raster with the name basename(out) will be imported into the current GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
landuse <- system.file("extdata", "landuse.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem, landuse), as_integer = c(FALSE, TRUE))
vector_to_mapset(vectors = c(sites, stream_shp))

# Create binary stream
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
out = "dem_burn.tif", burn = 10, overwrite = TRUE)

# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif",
size = 1, overwrite = TRUE)

# Derive flow accumulation and direction grids
```

```
derive_flow(dem = "dem_fill.tif", flow_dir = "fdir.tif",
flow_acc = "facc.tif", overwrite = TRUE)

# Snap sites to pour points (based on flow accumulation)
snap_sites(sites = "site", flow_acc = "facc.tif", max_move = 2,
out = "snapsite.shp", overwrite = TRUE)
point_to_raster(outlets = "site", out = "sites_rast.tif", overwrite = TRUE)

# Compute current site's watershed
rowID <- 1
current_watershed <- paste0("watershed_", rowID, ".tif")
get_watershed(sites = "snapsite.shp",
i =  rowID,
flow_dir = "fdir.tif",
out = current_watershed,
write_file = FALSE,
overwrite = TRUE)

# Plot
plot_GRASS(current_watershed, col = topo.colors(2))
plot_GRASS("streams_rast.tif", col = "white", add = TRUE)
plot_GRASS("sites_rast.tif", col = "red", add = TRUE)
}
```

`plot_GRASS` *A function to plot a raster from the current GRASS mapset*

Description

Given the name of a raster in the current GRASS mapset, this function will plot it as a `RasterLayer` object.

Usage

```
plot_GRASS(x, out_x, ...)
```

Arguments

`x`	The name of an object in the current GRASS mapset.
`out_x`	Optional. If supplied, the function makes a call to `retrieve_raster` and writes out the raster to the file path `out_x`. Otherwise the function will write the layer to `tempdir`.
`...`	Additional arguments to `plot`.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
landuse <- system.file("extdata", "landuse.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem, landuse), as_integer = c(FALSE, TRUE))
vector_to_mapset(vectors = c(sites, stream_shp))

# Rasterise streams and sites
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)
point_to_raster(outlets = "site", out = "sites_rast.tif", overwrite = TRUE)

# Plot
# Set number 1
par(mfrow = c(1,2))
plot_GRASS("dem.tif", col = topo.colors(15))
plot_GRASS("sites_rast.tif", col = heat.colors(1), add = TRUE)

# Set number 2
plot_GRASS("landuse.tif", col = topo.colors(2))
plot_GRASS("streams_rast.tif", col = heat.colors(1), add = TRUE)

# Reset plotting device parameters
par(mfrow = c(1, 1))
}
```

---

plot_layer *A function to plot a raster from a file name*

Description

Given a path to a file raster, this function will plot it as a `RasterLayer` object.

Usage

```
plot_layer(x, ...)
```

Arguments

`x`	A file path to a raster stored on disk. Can also be a `Raster*` object.
`...`	Additional arguments to `plot`.

Value

Nothing.

Examples

```
# Load data set file path
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
# Plot
# MAY NEED TO HAVE rgdal INSTALLED FOR .TIF FILES
plot_layer(dem)
```

---

point_to_raster *Convert outlet of a watershed from shapefile format into raster format*

Description

Given a shapefile of outlet(s), this function will convert its contents into a raster.

Usage

```
point_to_raster(outlets, out, overwrite = FALSE, max_memory = 300)
```

Arguments

outlets	A shapefile of outlets in the current GRASS mapset.
out	The name of the output raster.
overwrite	A logical indicating whether the output should be allowed to overwrite existing files. Defaults to FALSE.
max_memory	Max memory used in memory swap mode (MB). Defaults to 300.

Value

Nothing. A file called out will be created in the current GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
vector_to_mapset(vectors = sites)
```

```
# Point to raster
point_to_raster(outlets = "site", out = "sites_rast.tif", overwrite = TRUE)

# Check conversion success
vibe_check()

}
```

`rasterise_stream` *Turn a shapefile of stream edges into a raster*

Description

Given a shapefile of lines representing the channels of a stream network, this function will return a rasterised version of the shapefile. The raster will have the parameters of the current GRASS mapset.

Usage

```
rasterise_stream(streams, out, overwrite = FALSE, max_memory = 300, ...)
```

Arguments

`streams`	A file name for a shapefile of stream edges in the current GRASS mapset.
`out`	The filename of the output.
`overwrite`	A logical indicating whether the output is allowed to overwrite existing files. Defaults to `FALSE`.
`max_memory`	Max memory used in memory swap mode (MB). Defaults to `300`.
`...`	Additional arguments to `v.to.rast`.

Value

Nothing. A file will be written to `out`. Note that `out` can be a full file path to any location in your file system. A raster with the name `basename(out)` will be written to the current GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
vector_to_mapset(vectors = stream_shp)
```

```
# Create rasterised stream
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)

# Plot
plot_GRASS("streams_rast.tif", col = topo.colors(2))

}
```

`raster_to_mapset` *Import rasters into GRASS mapset*

Description

GRASS can only deal with raster and vector data in a GRASS mapset. This function takes external rasters and imports them into the current GRASS mapset.

Usage

```
raster_to_mapset(
  rasters,
  as_integer = rep(FALSE, length(rasters)),
  overwrite = FALSE,
  max_memory = 300,
  ...
)
```

Arguments

`rasters`	A character vector of filenames of rasters to import.
`as_integer`	A logical vector indicating whether each raster should be imported strictly in integer format. Defaults to `FALSE`.
`overwrite`	A logical indicating whether the overwrite flag should be used. If `FALSE`, then the corresponding raster is allowed to retain its original format. Defaults to `FALSE`. May cause value truncation if improperly used.
`max_memory`	Max memory used in memory swap mode (MB). Defaults to `300`.
`...`	Additional arguments to `r.import`.

Value

A vector of raster layer names in the GRASS mapset.

Examples

```
# Will only run if a GRASS session is initialised
if(check_running()){
   dem <- system.file("extdata", "dem.tif", package = "rdwplus")
   raster_to_mapset(dem)
}
```

`rdwplus` *An Implementation of IDW-PLUS (Inverse Distance Weighted Percent Land Use for Streams) in R*

Description

Use R to call the hydrological toolboxes and functions in GRASS GIS to compute spatially explicit land-use metrics for stream survey sites. The package includes functions for preprocessing digital elevation and streams data, and one function to compute all the spatially explicit land use metrics described in Peterson et al. (2011) Freshwater Biology, 56(3), 590-610.

`reclassify_streams` *Reclassify streams into the format required for the land use metric calculations*

Description

Given a streams raster, this function will either create a binary streams raster (0 for non-stream cells and 1 for stream cells) or a unary streams raster (1 for stream cells and NoData for all other cells). Another option is to reclassify the streams raster such that stream cells are given the value NoData and non-stream cells are given the value 1.

Usage

```
reclassify_streams(stream, out, out_type = "binary", overwrite = FALSE)
```

Arguments

`stream`	Name of a streams raster in the current GRASS mapset. Typically this is the result of `rasterise_stream`. The raster should have NoData values for all non-stream cells. Stream cells can have any other value.
`out`	The output file.
`out_type`	Either 'binary', 'unary', or 'none'. See the Description above.
`overwrite`	A logical indicating whether the output should be allowed to overwrite any existing files. Defaults to `FALSE`.

Value

Nothing. A file with the name out will be written to the current GRASS mapset. This raster will be in unsigned integer format.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data
set_envir(dem)
vector_to_mapset(vectors =  stream_shp)

# Create binary stream
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif", out_type = "binary", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_unary.tif", out_type = "unary", overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_none.tif", out_type = "none", overwrite = TRUE)

# Plot
plot_GRASS("streams_rast.tif", col = topo.colors(2), main = "Rasterized Streams")
plot_GRASS("streams_binary.tif", col = topo.colors(2), main = "Binary Streams")
plot_GRASS("streams_unary.tif", col = topo.colors(2), main = "Unary Streams")
plot_GRASS("streams_none.tif", col = topo.colors(2), main = "Null (none) Streams")
}
```

---

report_mapset *Identify current mapset or list all possible mapsets*

Description

GRASS GIS uses a system of mapsets.

Usage

```
report_mapset(which = "current")
```

Arguments

which	One of either 'current' (the default), which causes the function to return the current mapset, or 'possible', which causes the function to list all possible mapsets.

Value

Nothing.

`retrieve_raster` *Write a raster layer from the current GRASS mapset to file*

Description

This function writes a GRASS mapset raster to file.

Usage

```
retrieve_raster(layer, out_layer, overwrite = FALSE, ...)
```

Arguments

`layer`	The name of the raster in the GRASS mapset that is to be written out.
`out_layer`	The name of the file to be created, with the relevant file extension.
`overwrite`	A logical indicating whether the output from this function should be allowed to overwrite any existing files. Defaults to `FALSE`.
`...`	Additional arguments to `r.out.gdal`.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = dem, as_integer = FALSE)

# Retrieve raster
out_name <- paste0(tempdir(), "/retrieved_dem.tif")
retrieve_raster("dem.tif", out_layer = out_name, overwrite = TRUE)

}
```

`retrieve_vector` *Write a vector layer from the current GRASS mapset to file*

Description

This function writes a GRASS mapset vector layer (like a shapefile) to file.

Usage

```
retrieve_vector(layer, out_layer, overwrite = FALSE, ...)
```

Arguments

`layer`	The name of the vector layer in the GRASS mapset that is to be written out.
`out_layer`	The name of the shapefile to be created (with .shp file extension).
`overwrite`	A logical indicating whether the output from this function should be allowed to overwrite any existing files. Defaults to `FALSE`.
`...`	Additional arguments to `v.out.ogr`.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
vector_to_mapset(vectors = stream_shp)

# Retrieve raster
out_name <- paste0(tempdir(), "/", "retrieved_streams.shp")
retrieve_vector("streams", out_layer = out_name, overwrite = TRUE)

}
```

search_for_grass *Find GRASS installations*

Description

This function finds the path to potential GRASS installations. It does so in a very crude way; that is, by searching for directories that match the string `'GRASS'`.

Warning: this function works by brute force, so it may take a few minutes to find potential GRASS installations.

Note: This is not guaranteed to work. It is not hard to find the path to your computer's GRASS installation yourself. This is the preferred course of action.

Usage

```
search_for_grass(guide)
```

Arguments

`guide`	Optional. A specific folder to search in for the GRASS installation.

Value

A vector of file paths to potential GRASS installations.

Examples

```
my_grass <- search_for_grass()
my_grass
```

set_envir *Set environment parameters from a GIS layer.*

Description

This function simplifies the process of setting up a GRASS environment with parameters such as cell snapping, size and mapset extent.

Usage

```
set_envir(layer)
```

Arguments

layer	A Raster* object or the file path to a GIS layer that should be used to set environment parameters such as cell size, extent, etc.

Value

Nothing. Displays current environment settings.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")

# Set environment
set_envir(dem)

# Get environment metadata
gmeta()

}
```

---

silence *Function to suppress messages, warnings, errors from GRASS commands*

Description

Prevents the printing GRASS warnings, etc. Use with extreme caution. This is only helpful IF AND ONLY IF you are SURE that any printed messages, warnings, and errors are spurious.

Usage

```
silence(value)
```

Arguments

value	A logical indicating whether GRASS messages, warnings, errors should be suppressed. Can be missing, and it is missing by default. Choose "TRUE" or "FALSE".

Value

A logical indicating the current status of the option.

Examples

```
silence(TRUE)
silence(FALSE)
```

`snap_sites` *A function to snap sites in a shapefile to a flow accumulation grid*

Description

This function takes a set of survey site locations and makes sure that they are coincident with the point of highest flow accumulation within a specified distance. This is equivalent to snapping sites to a stream network. Note that this function calls `r.stream.snap`, which is a GRASS GIS add-on. It can be installed through the GRASS GUI.

Usage

```
snap_sites(
  sites,
  flow_acc,
  max_move,
  out,
  overwrite = FALSE,
  max_memory = 300,
  use_sp = TRUE,
  ...
)
```

Arguments

`sites`	File name for a shapefile containing the locations of the survey sites in the current GRASS mapset.
`flow_acc`	File name for a flow accumulation raster in the current GRASS mapset.
`max_move`	The maximum distance in cells that any site can be moved to snap it to the flow accumulation grid.
`out`	The output file path.
`overwrite`	Whether the output should be allowed to overwrite any existing files. Defaults to `FALSE`.
`max_memory`	Max memory used in memory swap mode (MB). Defaults to `300`.
`use_sp`	Logical to use 'sf' or 'stars' classes for vector objects. Defaults to `TRUE` to use 'stars' class.
`...`	Additional arguments to `r.stream.snap`.

Value

Nothing. Note that a shapefile of snapped survey sites will be written to the location `out` and a shapefile called `basename(out)` will be imported into the GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){
# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
sites <- system.file("extdata", "site.shp", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters and import data to GRASS
set_envir(dem)
raster_to_mapset(rasters = c(dem), as_integer = c(FALSE))
vector_to_mapset(vectors = c(sites, stream_shp))

# Create binary stream
out_name <- paste0(tempdir(), "/streams_rast.tif")
rasterise_stream("streams", out_name, overwrite = TRUE)
reclassify_streams("streams_rast.tif", "streams_binary.tif",
out_type = "binary", overwrite = TRUE)

# Burn dem
burn_in(dem = "dem.tif", stream = "streams_binary.tif",
 out = "dem_burn.tif", burn = 10, overwrite = TRUE)

# Fill sinks
fill_sinks(dem = "dem_burn.tif", out = "dem_fill.tif", size = 1, overwrite = TRUE)

# Derive flow accumulation and direction grids
derive_flow(dem = "dem_fill.tif", flow_dir = "fdir.tif",
flow_acc = "facc.tif", overwrite = TRUE)

# Snap sites to pour points (based on flow accumulation)
out_snap <- paste0(tempdir(), "/snapsite.shp")
snap_sites(sites = "site", flow_acc = "facc.tif", max_move = 2,
out = out_snap, overwrite = TRUE)

}
```

`toggle_silence` *Toggle between silence on and silence off*

Description

This function detects whether output suppression is on or off, and switches it to its opposite state. Under one setting, this function can be used as an off-switch for the GRASS message/warning/error suppression enforced via the use of `silence(value = TRUE)`.

Usage

```
toggle_silence(stay_off = TRUE)
```

Arguments

stay_off	A logical indicating whether output suppression should be kept off once it is turned off. That is, if this function is called but output suppression is already off, then for stay_off=TRUE output suppression will simply remain off. Defaults to TRUE.

Value

A logical indicating whether output suppression is active.

Examples

```
# Even if silence is currently off, silence will stay off
toggle_silence(TRUE)

# If silence is currently off, silence will be turned on.
toggle_silence(FALSE)
```

---

vector_to_mapset *Import rasters into GRASS mapset*

Description

GRASS can only deal with raster and vector data in a GRASS mapset. This function takes external vectors and imports them into the current GRASS mapset.

Usage

```
vector_to_mapset(vectors, overwrite = FALSE, ...)
```

Arguments

vectors	A character vector of filenames of shapefiles to import.
overwrite	A logical indicating whether the overwrite flag should be used. Defaults to FALSE.
...	Additional arguments to v.import.

Value

A vector of vector layer names in the GRASS mapset.

Examples

```
# Will only run if GRASS is running
if(check_running()){

# Load data set
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
stream_shp <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment parameters
set_envir(dem)

# Import vector data to mapset
vector_to_mapset(vectors = stream_shp)

}
```

`vibe_check` *A function to print current vectors and rasters in the mapset.*

Description

This function takes no inputs. It prints a list of maps in the current GRASS mapset.

Usage

```
vibe_check()
```

Value

Nothing.

Examples

```
if(check_running()) vibe_check()
```

Index