Package 'rdwplus'

February 12, 2025

Date 2025-02-06

Title Inverse Distance Weighted Percent Land Use for Streams

Version 1.0.1

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

Maintainer Alan Pearse <alan.pearse@unimelb.edu.au>

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 (>= 4.0.0), rgrass

Imports methods, utils, stars, sf, stringr

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-02-12 11:40:02 UTC

Contents

burn_in 2
check_running 3
clear_mask 4
compute_iFLO_weights 4
compute_iFLS_weights 6
compute_metrics 8

compute_metrics_precomputed . . . 11
convert_to_integer . . . 14
coord_to_raster . . . 15
derive_flow . . . 16
derive_streams . . . 17
fill_sinks . . . 19
get_distance . . . 20
get_flow_length . . . 21
get_watersheds . . . 23
install_extensions . . . 24
plot_GRASS . . . 25
point_to_raster . . . 26
rasterise_stream . . . 27
raster_to_mapset . . . 28
rast_calc . . . 29
reclassify_streams . . . 29
report_mapset . . . 30
retrieve_raster . . . 31
retrieve_vector . . . 32
search_for_grass . . . 33
set_envir . . . 33
set_mask . . . 34
silence . . . 35
snap_sites . . . 36
toggle_silence . . . 38
vector_to_mapset . . . 39
vibe_check . . . 40

Index **41**

`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()
```

---

clear_mask *Clear current raster mask*

Description

This function has no parameters. It can be used to clear an existing raster mask.

Usage

```
clear_mask()
```

Examples

```
if(check_running()) clear_mask()
```

---

compute_iFLO_weights *Compute iFLO weights*

Description

Compute an iFLO weight raster outside of the `compute_metrics()` function.

Usage

```
compute_iFLO_weights(
  pour_point,
  watershed,
  null_streams,
  flow_dir,
  out_flow_length,
  out_iFLO,
  out_iFLO_no_stream,
  idwp = -1,
  remove_streams = FALSE,
  ...
)
```

Arguments

pour_point	Pour point raster containing a single pour point (i.e., the outlet).
watershed	Watershed raster to use as a mask for the flow-path calculations.
null_streams	A streams raster with NoData for the stream cells and 1s everywhere else
flow_dir	A flow direction raster.
out_flow_length	Name of the output flow length raster.
out_iFLO	Name of the output weights raster.
out_iFLO_no_stream	Name of the output weights raster excluding cells on the stream line (ignored inf remove_streams = FALSE).
idwp	An inverse distance weighting power. This should be negative. The value idwp = -1 is the default.
remove_streams	A logical indicating whether cells corresponding to the stream line should be removed from the weights raster. Defaults to FALSE.
...	Optional extra arguments to get_flow_length().

Value

Nothing.

Examples

```
if(check_running()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)
```

```
# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Recode streams
reclassify_streams("new_stm.tif", "null_stm.tif", "none")

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)

# Get watersheds
get_watersheds("snapsite", "fd.tif", "wshed.tif", T)

#  Get pour points
coord_to_raster("snapsite", which = 1, out = "pour_point")

# Get iFLO weights
compute_iFLO_weights(
"pour_point",
"wshed.tif",
"null_stm.tif",
"fd.tif",
"fl_outlet.tif",
"iFLO_weights.tif",
idwp = -1,
remove_streams = FALSE
)
plot_GRASS("iFLO_weights.tif", col = topo.colors(12))
}
```

`compute_iFLS_weights` *Compute iFLS weights*

Description

Compute an iFLO weight raster outside of the `compute_metrics()` function.

Usage

```
compute_iFLS_weights(
  streams,
  null_streams,
  flow_dir,
  out_flow_length,
  out_iFLS,
  out_iFLS_no_stream,
  watershed,
```

```
  idwp,
  remove_streams,
  ...
)
```

Arguments

streams	Pour point raster containing a single pour point (i.e., the outlet).
null_streams	A streams raster with NoData for the stream cells and 1s everywhere else
flow_dir	A flow direction raster.
out_flow_length	Name of the output flow length raster.
out_iFLS	Name of the output weights raster.
out_iFLS_no_stream	Name of the output weights raster excluding cells on the stream line (ignored inf remove_streams = FALSE).
watershed	Watershed raster to use as a mask for the flow-path calculations. This is optional for the iFLS weight calculations.
idwp	An inverse distance weighting power. This should be negative. The value idwp = -1 is the default.
remove_streams	A logical indicating whether cells corresponding to the stream line should be removed from the weights raster. Defaults to FALSE.
...	Optional extra arguments to get_flow_length().

Value

Nothing.

Examples

```
if(check_running()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)
```

```
# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Recode streams
reclassify_streams("new_stm.tif", "null_stm.tif", "none")

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)

# Get watersheds
get_watersheds("snapsite", "fd.tif", "wshed.tif", T)

# Get iFLS weights
compute_iFLS_weights(
"new_stm.tif",
"null_stm.tif",
"fd.tif",
"fl_streams.tif",
"iFLS_weights.tif",
idwp = -1,
watershed = "wshed.tif",
remove_streams = FALSE,
overwrite = T
)
plot_GRASS("iFLS_weights.tif", col = topo.colors(12))
}
```

`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("lumped", "iFLO", "iFLS", "HAiFLO", "HAiFLS"),
```

```
  landuse,
  sites,
  out_fields,
  watersheds,
  flow_dir,
  flow_acc,
  streams,
  idwp = -1,
  percentage = TRUE,
  remove_streams = TRUE,
  max_memory = 300
)
```

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 the lumped, iFLO, iFLS, HAiFLO, and HAiFLS metrics.
landuse	Names of landuse or landcover rasters in the current GRASS mapset. These can be continuous (e.g., percentage cover or NDVI) or binary, with a value of 1 for cells with a particular land use category and a value of 0 otherwise.
sites	A set of survey sites in the current GRASS mapset.
out_fields	A character vector of output field names to store the metrics. Note that `length(out_fields)` must be the same as `length(landuse) * length(metrics)`.
watersheds	A vector of watershed raster names in the current GRASS mapset.
flow_dir	Name of a flow direction raster produced by `derive_flow` in the current GRASS mapset.
flow_acc	Name of a flow accumulation raster produced by `derive_flow` in the current GRASS mapset.
streams	Name of a streams raster in the current GRASS mapset. The stream raster must have NoData values in cells that do not fall along the stream line. Optional if you are not asking for the iFLS, iEDS, and/or HAiFLS metrics.
idwp	The inverse distance weighting parameter. Default is `-1`.
percentage	A logical indicating whether the result should be expressed as a percentage. Defaults to `TRUE`. Set to `FALSE` if the landuse/landcover raster is continuous.
remove_streams	A logical indicating whether cells falling on the stream line should be removed from iEDS, iFLS, and HAiFLS metrics. Defaults to `TRUE`, which is in line with the behaviour of IDWPLUS.
max_memory	Max memory used in memory swap mode (MB). Defaults to `300`.

Value

A `sf` object of the snapped survey sites that also contains the computed landscape metrics.

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
# You should load rdwplus and initialise GRASS via the initGRASS function
if(check_running()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)

# Get watersheds
get_watersheds("snapsite", "fd.tif", "wshed.tif", T)

compute_metrics(
  metrics = c("lumped", "iFLO", "iEDO", "HAiFLO", "iFLS", "iEDS", "HAiFLS"),
  landuse = "landuse.tif",
  sites = "snapsite",
  out_fields = c("lumped", "iFLO", "iEDO", "HAiFLO", "iFLS", "iEDS", "HAiFLS"),
  watersheds = "wshed.tif",
  flow_dir = "fd.tif",
```

```
  flow_acc = "fa.tif",
  streams = "new_stm.tif",
  idwp = -1
)
}
```

`compute_metrics_precomputed` *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). In contrast to `compute_metrics()`, this version of the function assumes most of the intermediate data layers (i.e., flow path distance and inverse-distance weight rasters) have been precomputed.

Usage

```
compute_metrics_precomputed(
  metrics = c("lumped", "iFLO", "iFLS", "HAiFLO", "HAiFLS"),
  landuse,
  sites,
  out_fields,
  watersheds,
  flow_dir,
  flow_acc,
  iEDO_weights,
  iFLO_weights,
  HAiFLO_weights,
  iEDS_weights,
  iFLS_weights,
  HAiFLS_weights,
  percentage = TRUE,
  max_memory = 300
)
```

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 the lumped, iFLO, iFLS, HAiFLO, and HAiFLS metrics.
`landuse`	Names of landuse or landcover rasters in the current GRASS mapset. These can be continuous (e.g., percentage cover or NDVI) or binary, with a value of 1 for cells with a particular land use category and a value of 0 otherwise.
`sites`	A set of survey sites in the current GRASS mapset.

out_fields	A character vector of output field names to store the metrics. Note that length(out_fields) must be the same as length(landuse) * length(metrics).
watersheds	A vector of watershed raster names in the current GRASS mapset.
flow_dir	Name of a flow direction raster produced by derive_flow in the current GRASS mapset.
flow_acc	Name of a flow accumulation raster produced by derive_flow in the current GRASS mapset.
iEDO_weights	A vector of names of iEDO weight rasters in the GRASS mapset.
iFLO_weights	A vector of names of iFLO weight rasters in the GRASS mapset.
HAiFLO_weights	A vector of names of HAiFLO weight rasters in the GRASS mapset.
iEDS_weights	A vector of names of iEDS weight rasters in the GRASS mapset.
iFLS_weights	A vector of names of iFLS weight rasters in the GRASS mapset.
HAiFLS_weights	A vector of names of HAiFLS weight rasters in the GRASS mapset.
percentage	A logical indicating whether the result should be expressed as a percentage. Defaults to TRUE. Set to FALSE if the landuse/landcover raster is continuous.
max_memory	Max memory used in memory swap mode (MB). Defaults to 300.

Value

A sf object of the snapped survey sites that also contains the computed landscape metrics.

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
# You should load rdwplus and initialise GRASS via the initGRASS function
if(check_running()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
```

```
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Recode streams
reclassify_streams("new_stm.tif", "null_stm.tif", "none")

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)

# Get watersheds
get_watersheds("snapsite", "fd.tif", "wshed.tif", T)

#  Get pour points
coord_to_raster("snapsite", which = 1, out = "pour_point")

# Get iFLO weights
compute_iFLO_weights(
"pour_point",
"wshed.tif",
"null_stm.tif",
"fd.tif",
"fl_outlet.tif",
"iFLO_weights.tif",
idwp = -1,
remove_streams = FALSE
)

# Get iFLS weights
compute_iFLS_weights(
"new_stm.tif",
"null_stm.tif",
"fd.tif",
"fl_streams.tif",
"iFLS_weights.tif",
idwp = -1,
watershed = "wshed.tif",
remove_streams = FALSE,
overwrite = T
)

# Compute metrics for this site
compute_metrics_precomputed(
```

```
  metrics = c("iFLO", "iFLS"),
  landuse = "landuse.tif",
  sites = "snapsite",
  out_fields = c("iFLO", "iFLS"),
  watersheds = "wshed.tif",
  iFLO_weights = "iFLO_weights.tif",
  iFLS_weights = "iFLS_weights.tif",
  flow_dir = "fd.tif",
  flow_acc = "fa.tif"
)
}
```

`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, out)
```

Arguments

`x`	A raster layer in the current GRASS mapset.
`out`	Name of the output raster. Avoid names with hyphens.

Value

Nothing. A raster with the name `out` will be added 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")

# Set environment
set_envir(dem)

# Make an integer-valued version of 'dem.tif'
convert_to_integer("dem.tif", "int_dem.tif")

# Compare
```

```
plot_GRASS("dem.tif")
plot_GRASS("int_dem.tif")

}
```

`coord_to_raster` *Turn coordinates of outlets into rasters*

Description

Given a set of x-y coordinates, this function will return a raster with a single cell at those coordinates.

Usage

```
coord_to_raster(outlets, which, out, overwrite = FALSE)
```

Arguments

`outlets`	The name of a set of sites in the current GRASS mapset.
`which`	A numeric identifier for the site to convert to raster.
`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`.

Details

This function is exposed to the user, and users are welcome to use if convenient for them, this function is intended for internal use in other functions.

Value

Nothing.

Examples

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

# Set environment parameters
set_envir(dem)

# Read in sites
vector_to_mapset(sts)
```

```
# Convert first site to raster
coord_to_raster("site", 1, "coords", overwrite = 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,
  d8 = TRUE,
  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.
`d8`	A logical indicating whether D8 flow direction should be used. If `FALSE`, multiple flow direction is allowed. Defaults to `TRUE`.
`overwrite`	A logical indicating whether any of the outputs should be allowed to overwrite existing files. Defaults to `FALSE`.
`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)
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 =  "dem_fill.tif", out_fd = "fd1.tif", 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(6))
plot_GRASS("facc.tif", col = topo.colors(6))
}
```

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

Description

Derive a raster and a vector layer of stream lines from a flow accumulation raster.

Usage

```
derive_streams(
  dem,
  flow_acc,
  out_rast,
  out_vect,
  min_acc = 1000,
```

```
  min_length = 0,
  overwrite = FALSE,
  ...
)
```

Arguments

dem	Name of an elevation raster in the current GRASS mapset.
flow_acc	Name of a flow accumulation raster in the current GRASS mapset.
out_rast	Name of the output raster dataset of stream lines. File extensions should not matter.
out_vect	Name of the output vector dataset of stream lines. Should be WITHOUT .shp extension.
min_acc	The minimum accumulation value that a cell needs 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()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)
```

```
# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)
}
```

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

Description

Remove sinks in a DEM (see the 'Details' section)

Usage

```
fill_sinks(dem, out_dem, out_fd, out_sinks, overwrite = FALSE, ...)
```

Arguments

`dem`	The name of a DEM in the current GRASS mapset.
`out_dem`	Name of the output DEM, which is a hydrologically corrected (sink-filled) DEM.
`out_fd`	Name of the output flow direction map for the sink-filled DEM.
`out_sinks`	Optional argument giving the name of the output sinks raster. Leave this missing to skip the output.
`overwrite`	A logical indicating whether the output should be allowed to overwrite existing files. Defaults to `FALSE`.
`...`	Optional additional parameters to `r.fill.dir`.

Details

A sink is a depression in a DEM. Water flows into these depressions but does not flow 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.

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()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)
}
```

---

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.

Examples

```
if(check_running()){

# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Get distances
get_distance("new_stm.tif", "dist_from_stream.tif", T)

}
```

---

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)

# 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 = "dem_fill.tif", out_fd = "fd1.tif", 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 watershed
get_watersheds(sites = "site", flow_dir = "fdir.tif", out = "wshed.tif", overwrite = T)

# Set mask
set_mask("wshed.tif")

# Get flow length
get_flow_length(
  str_rast = "streams_rast.tif",
  flow_dir = "fdir.tif",
  out = "flowlength.tif",
  to_outlet = TRUE,
  overwrite = TRUE
)

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

`get_watersheds` *Delineate watersheds for survey sites*

Description

This function delineates watersheds around a set of survey sites.

Usage

```
get_watersheds(sites, flow_dir, out, overwrite = FALSE, lessmem = FALSE)
```

Arguments

`sites`	A file path to a shapefile of points.
`flow_dir`	The name of a flow direction grid in the current GRASS mapset.
`out`	The names of the output watershed rasters.
`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`. If set to `TRUE`, the `r.wateroutlet.lessmem` extension module must be installed. It can be installed using the GRASS GUI.

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()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)

# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Recode streams
reclassify_streams("new_stm.tif", "null_stm.tif", "none")

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)

# Get watersheds
get_watersheds("snapsite", "fd.tif", "wshed.tif", T)
}
```

---

install_extensions *Install required extension(s)*

Description

Some functions in the rdwplus package rely on GRASS extensions that need to be installed prior to use. This function installs those extensions.

Usage

```
install_extensions()
```

Details

This function has no arguments. Simply run it and it will install a pre-set list of GRASS extensions.

Currently, the GRASS extension required are `r.stream.snap`, `r.stream.distance`, and `r.wateroutlet.lessmem`.

Value

Nothing.

Examples

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

---

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 `stars` object.

Usage

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

Arguments

`x`	The name of an object in the current GRASS mapset.
`colours`	Optional. A colour scale. If not supplied, the default settings in `plot.stars` is used. If supplied, `length(colours)` must be less than or equal to the number of unique values in the raster.
`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.stars`.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
# You should load rdwplus and initialise GRASS via the initGRASS function
if(check_running()){

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

# Set environment
set_envir(dem)

# Plot
plot_GRASS("dem.tif") # argument must match name of data set in the mapset
plot_GRASS("dem.tif", heat.colors(10)) # with different colour scale

}
```

`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
rasterise_stream("streams", "streams_rast.tif", overwrite = TRUE)

# Plot
plot_GRASS("streams_rast.tif")

}
```

`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)
}
```

`rast_calc` *Raster calculator (wrapper for "r.mapcalc")*

Description

Raster calculator (wrapper for "r.mapcalc")

Usage

```
rast_calc(ex, overwrite = TRUE)
```

Arguments

`ex`	A valid raster calculator expression for GRASS.
`overwrite`	Defaults to `TRUE`.

Value

Nothing.

`reclassify_streams` *Reclassify streams into various formats*

Description

Re-format a stream raster.

Usage

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

Arguments

`stream`	Name of a streams raster in the current GRASS mapset. This can be the output from `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 'zeros_to_nodata', 'binary', 'unary', or 'none'. See Details below
`overwrite`	A logical indicating whether the output should be allowed to overwrite any existing files. Defaults to `FALSE`.

Details

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.

Do not use raster names containing dashes/hyphens. The underlying call to `r.calc` will crash if the raster name contains these symbols because they are misinterpreted as math symbols.

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()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)
}
```

---

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 projection and computation region from a raster file.*

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(file, ...)
```

Arguments

file	The file path to a raster that should be used to set environment parameters such as the projection, cell size, extent, etc. The file argument will automatically be imported into the mapset as basename(file).
...	Optional arguments for raster_to_mapset(). The main argument of interest for most users will be overwrite, which should be set to true if an object of name basename(file) already exists in the mapset.

Value

Nothing. Displays current environment settings.

Examples

```
# Will only run if GRASS is running
# You should load rdwplus and initialise GRASS with initGRASS
if(check_running()){

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

# Set environment
set_envir(dem)

}
```

---

set_mask *Set a raster mask*

Description

Set a raster mask

Usage

```
set_mask(x, inverse = FALSE, overwrite = TRUE, ...)
```

Arguments

x	Raster to use as a mask
inverse	Whether the inverse of the raster should be used as the mask. Defaults to FALSE.
overwrite	Whether the existing mask should be overwritten. Defaults to TRUE.
...	Optional. Additional parameters for r.mask.

Value

Nothing.

Examples

```
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)

# 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 = "dem_fill.tif", out_fd = "fd1.tif", 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 watershed
get_watersheds(sites = "site", flow_dir = "fdir.tif", out = "wshed.tif", overwrite = T)

# Set mask
set_mask("wshed.tif")

# Get flow length
get_flow_length(
  str_rast = "streams_rast.tif",
  flow_dir = "fdir.tif",
  out = "flowlength.tif",
  to_outlet = TRUE,
  overwrite = TRUE
)

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

`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 survey sites to a stream raster and a flow accumulation raster*

Description

This function takes a set of survey site locations and snaps them to the highest-value cell within a flow accumulation raster, within a specified distance. 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,
  stream,
  flow_acc,
  max_move,
  out,
  overwrite = FALSE,
  max_memory = 300,
  ...
)
```

Arguments

sites	File name for a shapefile containing the locations of the survey sites in the current GRASS mapset.
stream	Name of a stream raster in the current GRASS mapset. This can either be formatted to have NoData in non-stream cells or 0s in non-stream cells.
flow_acc	Name of 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	Name of the output in the current GRASS mapset. Note that this function will add a column called snap_dist to the attribute table of the input sites, which indicates how far each site was snapped.
overwrite	Whether the output should be allowed to overwrite any existing files. Defaults to FALSE.
max_memory	Max memory (in) used in memory swap mode. Defaults to 300 Mb.
...	Additional arguments to r.stream.snap.

Value

Nothing.

Examples

```
# Will only run if GRASS is running
# You should load rdwplus and initialise GRASS via the initGRASS function
if(check_running()){
# Retrieve paths to data sets
dem <- system.file("extdata", "dem.tif", package = "rdwplus")
lus <- system.file("extdata", "landuse.tif", package = "rdwplus")
sts <- system.file("extdata", "site.shp", package = "rdwplus")
stm <- system.file("extdata", "streams.shp", package = "rdwplus")

# Set environment
set_envir(dem)

# Get other data sets (stream layer, sites, land use, etc.)
raster_to_mapset(lus)
vector_to_mapset(c(stm, sts))

# Reclassify streams
out_stream <- paste0(tempdir(), "/streams.tif")
rasterise_stream("streams", out_stream, TRUE)
reclassify_streams("streams.tif", "streams01.tif", overwrite = TRUE)

# Burn in the streams to the DEM
burn_in("dem.tif", "streams01.tif", "burndem.tif", overwrite = TRUE)

# Fill dem
fill_sinks("burndem.tif", "filldem.tif", "fd1.tif", "sinks.tif", overwrite = TRUE)
```

```
# Derive flow direction and accumulation grids
derive_flow("dem.tif", "fd.tif", "fa.tif", overwrite = T)

# Derive a new stream raster from the FA grid
derive_streams("dem.tif", "fa.tif", "new_stm.tif", "new_stm", min_acc = 200, overwrite = T)

# Snap sites to streams and flow accumulation
snap_sites("site", "new_stm.tif", "fa.tif", 2, "snapsite", T)
}
```

`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 summarise the computation region, vectors and rasters in the mapset.*

Description

This function takes no inputs. It prints a list of data sets in the current GRASS mapset, as well as the parameters of the current computation region.

Usage

```
vibe_check()
```

Value

Nothing.

Examples

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

Index

burn_in, 2

check_running, 3
clear_mask, 4
compute_iFLO_weights, 4
compute_iFLS_weights, 6
compute_metrics, 8
compute_metrics_precomputed, 11
convert_to_integer, 14
coord_to_raster, 15

derive_flow, 16
derive_streams, 17

fill_sinks, 19, *21*

get_distance, 20
get_flow_length, 21
get_watersheds, 23

install_extensions, 24

plot_GRASS, 25
point_to_raster, 26

rast_calc, 29
raster_to_mapset, 28
rasterise_stream, 27
reclassify_streams, 29
report_mapset, 30
retrieve_raster, *25*, 31
retrieve_vector, 32

search_for_grass, 33
set_envir, 33
set_mask, 34
silence, 35
snap_sites, 36

toggle_silence, 38

vector_to_mapset, 39
vibe_check, 40