

# Package: Orcs (via r-universe)

September 8, 2024

**Title** Omnidirectional R Code Snippets

**Version** 1.2.3

**Maintainer** Florian Detsch <fdetsch@web.de>

**Description** I tend to repeat the same code chunks over and over again.

At first, this was fine for me and I paid little attention to such redundancies. A little later, when I got tired of manually replacing Linux filepaths with the referring Windows versions, and vice versa, I started to stuff some very frequently used work-steps into functions and, even later, into a proper R package. And that's what this package is - a hodgepodge of various R functions meant to simplify (my) everyday-life coding work without, at the same time, being devoted to a particular scope of application.

**License** MIT + file LICENSE

**URL** <https://github.com/fdetsch/Orcs>

**BugReports** <https://github.com/fdetsch/Orcs/issues>

**LazyData** TRUE

**Depends** methods, R (>= 2.10)

**Imports** bookdown, grDevices, grid, knitr, lattice, latticeExtra, plotrix, Rcpp (>= 0.11.3), remotes, sf, sp, stats, terra

**Suggests** checkmate, raster, rmarkdown, tinytest

**LinkingTo** Rcpp

**RoxygenNote** 7.2.3

**Roxygen** list(markdown = TRUE)

**SystemRequirements** GNU make, 7zip, unix2dos

**VignetteBuilder** knitr

**Encoding** UTF-8

**Repository** <https://fdetsch.r-universe.dev>

**RemoteUrl** <https://github.com/fdetsch/orcs>

**RemoteRef** HEAD

**RemoteSha** 6658fc39747eac3dab0f2b33b9ef1306bcb00670

## Contents

Orcs-package	2
assignSSH	3
buildBook	4
bumpVersion	4
compareDistributions	5
coords2Lines	7
coords2Polygons	8
evalMetrics	9
ext2spy	10
ifMissing	11
latticeCombineGrid	12
latticeCombineLayer	13
lineEnding	14
list2df	15
loadFromGit	16
loadPkgs	17
meanDifference	17
merge	18
multiKnit	19
offsetGridText	20
OrcsCppFun	21
par7zip	22
pureBasename	23
pvalue	24
rgb2spLayout	25
rmDuplCols	26
setwdOS	27
stextGrob	28
substrC	29
trimImages	29
unlistStrsplit	31
unsortedFactor	31
<b>Index</b>	<b>33</b>

---

Orcs-package

*Omnidirectional R Code Snippets.*

---

### Description

Omnidirectional R Code Snippets

### Details

The package provides a variety of functions which I regularly use during my everyday work.

**Author(s)**

Florian Detsch, Tim Appelhans, Baptiste Auguie, OpenStreetMap contributors

*Maintainer:* Florian Detsch <fdetsch@web.de>

---

assignSSH

*Assign SSH Key to Local Git Repository*

---

**Description**

Assign an SSH key to a local Git repository to bypass user/password prompts during ‘git push’. See [Generating an SSH Key](#) for further information on how to generate an SSH key and add it to your GitHub account.

**Usage**

```
assignSSH(user, repo)
```

**Arguments**

user	GitHub user name as character. If not specified, information on GitHub user and repository name is taken from the current working environment.
repo	GitHub repository name as character, see ‘user’.

**See Also**

<https://docs.github.com/articles/generating-an-ssh-key/>.

**Examples**

```
## Not run:  
## for an arbitrary git repository  
assignSSH()  
  
## for this very git repository  
assignSSH(user = "fdetsch", repo = "Orcs")  
  
## End(Not run)
```

---

buildBook	<i>Build a Book without Underscores</i>
-----------	---

---

### Description

Since the use of underscores ('\_') is not permitted when streaming **bookdown** documents via [GitHub Pages](#), this wrapper function serves to remove any unwanted underscores from subfolders and link .html documents created by `bookdown::render_book()`.

### Usage

```
buildBook(output_dir = "book", ...)
```

### Arguments

output_dir	Output directory as character.
...	Arguments passed to <code>bookdown::render_book()</code> .

### Note

While all remaining arguments passed to `bookdown::render_book()` remain untouched, and hence their specification is freely up to the user, the default value of 'output\_dir' is explicitly set to "book" here. If this were not the case (i.e. if the default value were used), the output document would be created in "\_book" which is not desirable for obvious reasons.

### Author(s)

Florian Detsch

---

bumpVersion	<i>Bump Package 'Version:' and 'Date:' in DESCRIPTION File</i>
-------------	--

---

### Description

This function let's you bump the version number and creation date of your package's DESCRIPTION file. Additionally, it bumps the version numbers of a NEWS.md file and automatically generates a corresponding plain NEWS file (for R-help pages). Supported versioning system is **MAJOR.MINOR.PATCH**.

### Usage

```
bumpVersion(
  element = "patch",
  pkg.repo = ".",
  news = file.path(pkg.repo, "NEWS.md"),
  plain_news = TRUE
)
```

**Arguments**

element	character, one of "major", "minor", "patch" (default) to be bumped.
pkg.repo	Path to package repository folder. Default is current working directory, i.e. ".".
news	The NEWS.md file of the repo (assumed to be in top level path). If this exists, the first line of that file will be rewritten to be "\<packagename\> \<major.minor.patch\>". Note that the current implementation assumes that the NEWS file is in .md format. A plain NEWS file (for R-help pages) will be generated automatically.
plain_news	whether to generate a plain NEWS file in the package root directory from the NEWS.md file supplied to argument 'news'.

**Author(s)**

Tim Appelhans

**See Also**

<https://semver.org/>

---

compareDistributions *Compare Two Density Distributions Side by Side*

---

**Description**

This function will produce a plot of two density functions displayed side by side.

**Usage**

```
compareDistributions(  
  left,  
  right,  
  add.spread = TRUE,  
  print.stats = TRUE,  
  xlim = NULL,  
  ylim = NULL,  
  clrs = c("purple", "lightblue"),  
  xlab = "density",  
  ylab = "value",  
  ...  
)
```

**Arguments**

<code>left</code>	numeric vector.
<code>right</code>	numeric vector.
<code>add.spread</code>	logical, whether to plot the spread (q25 to q75 and the median).
<code>print.stats</code>	logical, whether to print summary statistics for each distribution.
<code>xlim, ylim</code>	numeric axis limits, see <code>lattice::xyplot()</code> .
<code>clrs</code>	A character vector of length 2 specifying the colors for the filled density regions.
<code>xlab, ylab</code>	character axis labels, see <code>graphics::plot()</code> .
<code>...</code>	Additional arguments passed to <code>stats::density()</code> .

**Value**

A trellis object.

**Author(s)**

Tim Appelhans

**Examples**

```
compareDistributions(rnorm(1000, 2, 3), rnorm(1000, -5, 1))
compareDistributions(rnorm(1000, 2, 3), rnorm(1000, -5, 1),
  add.spread = FALSE)
compareDistributions(rnorm(1000, 2, 3), rnorm(1000, -5, 1),
  add.spread = TRUE, clrs = c("red", "brown"))
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = FALSE, add.spread = FALSE)

## pass additional parameters to stats::density()
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = FALSE, add.spread = FALSE, bw = 5)
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = FALSE, add.spread = FALSE, bw = 8,
  kernel = "rectangular")
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = FALSE, add.spread = TRUE, bw = 8,
  n = 3)
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = TRUE, add.spread = TRUE, bw = 0.1)
compareDistributions(rnorm(1000, 2, 5), rnorm(1000, -5, 4),
  print.stats = TRUE, add.spread = TRUE, bw = 0.5)
```

---

coords2Lines	<i>Convert Points to SpatialLines*</i>
--------------	--

---

### Description

Create a `SpatialLines*` object from a `Line` object or set of point coordinates in one go, i.e. without being required to run through the single steps outlined in `sp::SpatialLines()`.

### Usage

```
## S4 method for signature 'matrix'  
coords2Lines(coords, ID, data, match.ID = TRUE, ...)  
  
## S4 method for signature 'Line'  
coords2Lines(coords, ID, data, match.ID = TRUE, ...)
```

### Arguments

<code>coords</code>	Line object or 2-column numeric matrix with x and y coordinates.
<code>ID</code>	character, see <code>sp::Lines()</code> .
<code>data</code>	<code>data.frame</code> with data to add to the output <code>SpatialLines*</code> object (optional).
<code>match.ID</code>	logical, see <code>sp::SpatialLinesDataFrame()</code> .
<code>...</code>	Further arguments passed on to <code>sp::SpatialLines()</code> , i.e. <code>proj4string</code> .

### Value

If 'data' is missing, a `SpatialLines` object; else a `SpatialLinesDataFrame` object.

### Examples

```
library(sp)  
  
coords1 <- cbind(c(2, 4, 4, 1, 2), c(2, 3, 5, 4, 2))  
sln1 <- coords2Lines(coords1, ID = "A")  
  
coords2 <- cbind(c(5, 4, 2, 5), c(2, 3, 2, 2))  
sln2 <- coords2Lines(coords2, ID = "B")  
  
plot(sln1, col = "grey75")  
plot(sln2, col = "grey25", add = TRUE)
```

---

`coords2Polygons`*Convert Points to SpatialPolygons\**

---

## Description

Create a `SpatialPolygons*` object from a `Polygon` object or set of point coordinates in one go, i.e. without being required to run through the single steps outlined in `sp::SpatialPolygons()`.

## Usage

```
## S4 method for signature 'matrix'
coords2Polygons(coords, hole = NA, ID, data, match.ID = TRUE, ...)

## S4 method for signature 'Polygon'
coords2Polygons(coords, ID, data, match.ID = TRUE, ...)
```

## Arguments

<code>coords</code>	Polygon object or 2-column numeric matrix with x and y coordinates.
<code>hole</code>	logical, see <code>sp::Polygon()</code> .
<code>ID</code>	character, see <code>sp::Polygons()</code> .
<code>data</code>	<code>data.frame</code> with data to add to the output <code>SpatialPolygons*</code> object (optional).
<code>match.ID</code>	logical, see <code>sp::SpatialPolygonsDataFrame()</code> .
<code>...</code>	Further arguments passed on to <code>sp::SpatialPolygons()</code> , i.e. <code>p0</code> and <code>proj4string</code> .

## Value

If 'data' is missing, a `SpatialPolygons` object; else a `SpatialPolygonsDataFrame` object.

## Examples

```
library(sp)

coords1 <- cbind(c(2, 4, 4, 1, 2), c(2, 3, 5, 4, 2))
spy1 <- coords2Polygons(coords1, ID = "A")

coords2 <- cbind(c(5, 4, 2, 5), c(2, 3, 2, 2))
spy2 <- coords2Polygons(coords2, ID = "B")

plot(spy1, col = "grey75")
plot(spy2, col = "grey25", add = TRUE)
```



---

`evalMetrics`*Compute Selected Evaluation Metrics*

---

**Description**

Compute selected evaluation metrics for binary (i.e. two-class) confusion matrices.

**Usage**

```
evalMetrics(mat, type = c("accuracy", "precision", "recall"))
```

**Arguments**

<code>mat</code>	Binary confusion matrix (2-by-2; see Examples).
<code>type</code>	Target evaluation metric as character, defaults to "accuracy". Other available options are "precision" and "recall".

**Value**

A single numeric.

**Author(s)**

Florian Detsch

**References**

University of Michigan (2017) Applied Machine Learning in Python. Available online: <https://www.coursera.org/learn/python-machine-learning/home/welcome>.

**Examples**

```
in1 = matrix(c(96, 4, 8, 19), nc = 2L, byrow = TRUE)
rownames(in1) = c("Condition Positive", "Condition Negative")
colnames(in1) = c("Predicted Positive", "Predicted Negative")

evalMetrics(in1) # default: "accuracy"
evalMetrics(in1, "precision")
evalMetrics(in1, "recall")

in2 = matrix(c(26, 17, 7, 400), nc = 2, byrow = TRUE)
evalMetrics(in2, "precision")
evalMetrics(in2, "recall")
```

---

ext2spy	<i>Convert Spatial Extent to Polygon</i>
---------	--

---

### Description

Convert a spatial extent to polygons.

### Usage

```
ext2spy(x, crs = "EPSG:4326", as_sf = TRUE)
```

### Arguments

x	A <code>SpatExtent</code> object, or any object from which such an object can be extracted, e.g. <code>SpatRaster</code> .
crs	Coordinate reference system set via <code>terra::crs()</code> .
as_sf	logical. If <code>TRUE</code> (default), the returned object is of class <code>sf</code> rather than <code>Spatial*</code> .

### Value

Depending on 'as\_sf', either a `c(sf, data.frame)` or `SpatVector` object.

### Author(s)

Florian Detsch

### See Also

`terra::ext()`.

### Examples

```
ext = terra::ext(c(25, 70, -5, 30))
ext2spy(ext) # 'sf' (default)
ext2spy(ext, as_sf = FALSE) # 'Spatial*'
```

---

`ifMissing`*Take Measures in Case of Nonexisting Target Files*

---

### Description

If a target file already exists, it is simply being imported into R. However, if the specified target file does not exist, it is first created by a user-defined function and subsequently returned, thus rendering explicit calls to `file.exists()` unnecessary.

### Usage

```
ifMissing(of1, fun0 = terra::rast, fun1 = terra::writeRaster, arg1, ...)
```

### Arguments

<code>of1</code>	Target file name as character.
<code>fun0</code>	If 'of1' exists, function to be applied to it. Defaults to <code>terra::rast()</code> .
<code>fun1</code>	If 'of1' does not exist, function used to create it. Defaults to <code>terra::writeRaster()</code> .
<code>arg1</code>	Argument in 'fun1' (as character) that corresponds to 'of1', e.g. 'filename' in <code>terra::writeRaster()</code> or 'file' in <code>utils::write.table()</code> . If missing (default), the target file name passed to 'fun1' needs to be explicitly included via '...'. ...
<code>...</code>	Additional arguments passed to 'fun0' and 'fun1'.

### Value

If 'of1' has already existed, the contents of 'of1' derived from 'fun0'; else the output resultant from 'fun1'.

### Author(s)

Florian Detsch

### See Also

`file.exists()`, `do.call()`.

### Examples

```
# simply import existing file
logo <- system.file("ex/logo.tif", package = "terra")
s <- ifMissing(logo)

# create nonexisting file and import it afterwards
logo2 <- file.path(tempdir(), "rlogo.tif")
s2 <- ifMissing(logo2, arg1 = "filename", x = s, datatype = "INT1U")
```

```
# this also works with text files and more sophisticated custom functions
fun = function(x, file = "", ...) {
  write.csv(x, file, ...)
  read.csv(file)
}

data(iris)
of1 <- file.path(tempdir(), "iris.csv")
iris2 <- ifMissing(of1, fun1 = fun, x = iris, file = of1, quote = FALSE, row.names = FALSE)
```

---

`latticeCombineGrid`      *Combine Multiple Lattice Plots in a Faceted Grid (Panels)*

---

### Description

This function combines multiple **lattice** plot objects in a faceted grid. Note that the global plot settings (e.g. 'xlim', 'ylim', ...) are taken from the first object though the user can specify whether 'scales' should be identical or not. This is particularly useful when looping over large amounts of data using `lapply()` or the like (see Examples).

### Usage

```
latticeCombineGrid(
  trellis.list,
  between = list(y = 0.3, x = 0.3),
  as.table = TRUE,
  ...
)
```

### Arguments

<code>trellis.list</code>	A list containing <b>lattice</b> plot objects.
<code>between</code>	Space between panels.
<code>as.table</code>	If TRUE (default), drawing is top left to bottom right.
<code>...</code>	Additional arguments passed to <code>latticeExtra::c.trellis()</code> .

### Value

A single **lattice** plot object.

### Author(s)

Tim Appelhans

### See Also

`latticeExtra::c.trellis()`.

**Examples**

```

library(lattice)

p1 = xyplot(1:10 ~ 1:10)
p2 = xyplot(10:1 ~ 1:10)

(
  p = latticeCombineGrid(
    list(p1, p2)
  )
)

if (requireNamespace("raster", quietly = TRUE)) {
  # load data
  # Use a probability map assuming high potential for city expansion is just
  # resulting from proximity to current urban area:
  prd = raster::raster(system.file("extdata/probability.rst", package = "Orcs"))

  # observed city growth between 1990 and 2006
  obs = raster::raster(system.file("extdata/citygrowth.tif", package = "Orcs"))

  # masking current urban area since these pixels have no potential for change
  msk = raster::raster(system.file("extdata/citymask.tif", package = "Orcs"))

  # create data list
  dat <- list(prd, obs, msk)

  # create list of lattice plots
  plist <- lapply(dat, raster::splot, scales = list(draw = TRUE))

  # # draw individually
  # plist[[1]]
  # plist[[2]]
  # plist[[3]]

  # combine to grid, using c(1, 3) layout
  p = latticeCombineGrid(plist, layout = c(1, 3))
  print(p)
}

```

---

latticeCombineLayer      *Combine Multiple Lattice Plots Layerwise*

---

**Description**

This function combines multiple **lattice** plot objects drawing each as a layer on top of the previous plots. Note that the global plot settings (e.g. 'xlim', 'ylim', ...) are taken from the first object. This is particularly useful when looping over large amounts of data using [lapply\(\)](#) (see Examples).

**Usage**

```
latticeCombineLayer(trellis.list, ...)
```

**Arguments**

`trellis.list` A list containing **lattice** plot objects.  
`...` Additional arguments passed to `latticeExtra::as.layer()`.

**Value**

A single **lattice** plot object.

**Author(s)**

Tim Appelhans

**See Also**

[latticeExtra::as.layer\(\)](#).

**Examples**

```
library(latticeExtra)
dat <- list(1:10,
           10:1,
           3:7,
           7:3)

plist <- lapply(seq(dat), function(i) {
  tmp <- xyplot(dat[[i]] ~ seq(dat[[i]]),
               type = "l", col = i)
})

p <- latticeCombineLayer(plist)

print(p)
```

---

lineEnding

*Convert Between DOS and UNIX Line Endings*

---

**Description**

This function converts between DOS and UNIX style line endings by invoking `unix2dos` (or `dos2unix`) upon a text file (see also `system("unix2dos --help")`). Note that `'unix2dos'` must be installed on your local system, see Source.

**Usage**

```
lineEnding(infile, pattern = NULL, outfile = NULL, to = c("dos", "unix"), ...)
```

**Arguments**

infile	Input filename(s).
pattern	See <a href="#">list.files()</a> . This will be ignored if 'infile' is specified.
outfile	Output filename. If not supplied, 'infile' will be overwritten.
to	Either "dos" or "unix".
...	Additional arguments passed to <a href="#">list.files()</a> , only applicable if 'infile' is not specified.

**Author(s)**

Florian Detsch

**Source**

[Dos2Unix/Unix2Dos Text file format converters.](#)

**See Also**

[list.files\(\)](#), [system\(\)](#).

**Examples**

```
## input file
infile <- paste(system.file(package = "Orcs"), "DESCRIPTION", sep = "/")

## convert to dos line endings and write to output file
of1 = file.path(tempdir(), "DESCRIPTION4wd")
lineEnding(infile, outfile = of1, to = "dos")
```

---

list2df	<i>Create data.frame from list</i>
---------	------------------------------------

---

**Description**

Create a `data.frame` from a `list` directly, i.e. without being required to explicitly call [rbind\(\)](#) first.

**Usage**

```
list2df(x, bind = c("rows", "cols"), ...)
```

**Arguments**

x	A list object.
bind	Binding direction. Available options are "rows" (default) and "cols" for <code>rbind()</code> and <code>cbind()</code> , respectively.
...	Additional arguments passed to <code>data.frame()</code> .

**Value**

A `data.frame` object.

**Examples**

```
lst <- list(letters[1:3], letters[4:6], letters[7:9])

do.call("rbind", lst) # results in matrix
list2df(lst)          # results in data.frame created using rbind()
list2df(lst, bind = "cols") # same for cbind()
```

---

loadFromGit

*Install and Load a Package from GitHub*


---

**Description**

This function comprises multiple steps required to install and load a package directly from GitHub.

**Usage**

```
loadFromGit(repo = "fdetsch/Orcs", ...)
```

**Arguments**

repo	Repository address as character, defaults to "fdetsch/Orcs".
...	Additional arguments passed to <code>remotes::install_github()</code> .

**Author(s)**

Florian Detsch

**Examples**

```
## Not run:
## install 'Orcs' from GitHub
loadFromGit("fdetsch/Orcs")

## End(Not run)
```



---

loadPkgs	<i>Load Multiple Packages</i>
----------	-------------------------------

---

**Description**

Load and attach multiple packages at once.

**Usage**

```
loadPkgs(pkgs, ...)
```

**Arguments**

pkgs	Packages to load as character.
...	Additional arguments passed to <code>library()</code> , except for 'character.only' which is set to TRUE.

**Note**

Package start-up messages are automatically disabled.

**Author(s)**

Florian Detsch

**Examples**

```
loadPkgs(c("terra", "lattice"))
```

---

meanDifference	<i>Calculate Mean Difference Between Two Datasets</i>
----------------	---

---

**Description**

Calculate the mean difference between two datasets as suggested by Wang *et al.* (2012).

**Usage**

```
## S4 method for signature 'SpatRaster'  
meanDifference(x, y)  
  
## S4 method for signature 'numeric'  
meanDifference(x, y)
```

**Arguments**

`x, y` Pairwise objects of class `SpatRaster`, `RasterLayer` or numeric.

**Value**

The mean difference between the two inputs as numeric.

**Source**

Wang *et al.* (2012) Impact of sensor degradation on the MODIS NDVI time series. *Remote Sensing of Environment* 119, 55-61, doi:10.1016/j.rse.2011.12.001.

Detsch *et al.* (2016) A Comparative Study of Cross-Product NDVI Dynamics in the Kilimanjaro Region - A Matter of Sensor, Degradation Calibration, and Significance. *Remote Sensing* 8(2), 159, doi:10.3390/rs8020159.

**Examples**

```
x <- 1:10
y <- 2:11
meanDifference(x, y)
```

---

merge

---

*Merge Objects Stored in a List*


---

**Description**

Complementing existing merge methods, e.g. `terra::merge()` for `Spat*` objects, which typically work with one or two inputs only, this function accepts a list of objects that are to be merged together.

**Usage**

```
## S4 method for signature 'list,missing'
merge(x, by = 1L, all = TRUE, ...)
```

**Arguments**

`x` A list of objects of the same type (e.g. `Spat*` or `data.frame`).

`by, all` See `merge.data.frame()`. Ignored if data stored in `'x'` is not of class `data.frame`.

`...` Additional arguments passed to the underlying merge method (e.g. arguments compatible with `terra::merge()` and `terra::writeRaster()` for `Spat*` input). Ignored if data stored in `'x'` is of class `data.frame`.

**Value**

A merged object (e.g. a new `Spat*` object with a larger spatial extent).

**Author(s)**

Florian Detsch

**See Also**[do.call\(\)](#), [Reduce\(\)](#).**Examples**

```
## `SpatRaster` input
dms = list.files(system.file("extdata", package = "Orcs")
                 , pattern = "ASTGTM2.*dem.tif$", full.names = TRUE)
dms = lapply(dms, terra::rast)

dem = merge(dms[3:4])
terra::plot(dem)

## data.frame input
mrg = merge(list(iris, iris, iris)
            , by = c("Species", "Sepal.Length", "Petal.Width"))
head(mrg)
```

---

`multiKnit`*Convert Multiple R Markdown Files to Ordinary Markdown*

---

**Description**

This function is a convenient wrapper around `knitr::knit()` as it automatically converts multiple R Markdown files (.Rmd) located in a specified folder (and, optionally, matching a particular pattern) to standard Markdown (.md).

**Usage**

```
multiKnit(path_in = ".", path_out = path_in, pattern = "*.Rmd$", ...)
```

**Arguments**

<code>path_in</code>	Input file path as character, defaults to the current working directory.
<code>path_out</code>	Output file path as character, defaults to <code>path_in</code> .
<code>pattern</code>	Passed to <code>list.files()</code> , defaults to <code>"*.Rmd\$"</code> .
<code>...</code>	Additional arguments passed to <code>knitr::knit()</code> .

**Value**

Output filenames as character.

**Author(s)**

Florian Detsch

offsetGridText

*Insert Offset Text Annotation into trellis Plot***Description**

This is a wrapper function around `Orcs:::calcOffsetGridText` and **grid** based text drawing functions (currently including `grid::grid.text()` and `grid.stext()`) that automatically adds offset text annotations to a trellis plot.

**Usage**

```
offsetGridText(
  x,
  y = NULL,
  labels,
  xlim = NULL,
  ylim = NULL,
  pos = NULL,
  stext = FALSE,
  offset = 0.02,
  ...
)
```

**Arguments**

<code>x</code>	A numeric vector containing x coordinates, or a 2-column matrix containing x and y coordinates.
<code>y</code>	A numeric vector containing y coordinates, or NULL if 'x' is a two-column matrix.
<code>labels</code>	The text to be written as character.
<code>xlim, ylim</code>	X and Y-axis limits (c(min, max)) of the current plot. If not supplied, limits are automatically calculated from supplied x and y coordinates.
<code>pos</code>	Text position specifier(s) as integer used by <code>graphics::text()</code> . If not supplied, optimal text positions will be determined with respect to neighboring locations using <code>plotrix::thigmophobe()</code> .
<code>stext</code>	logical, defaults to FALSE. If TRUE, shadow text will be drawn around 'labels'.
<code>offset</code>	A numeric offset in normalized parent coordinates ("npc", see <code>grid::unit()</code> ).
<code>...</code>	Additional arguments passed to the respective <b>grid</b> text drawing function (depends on 'stext').

**Author(s)**

Florian Detsch

**Examples**

```

KiLi = terra::rast(system.file("extdata/KiLi.tif", package = "Orcs"))

# kilimanjaro peaks
peaks = data.frame(Peak = c("Kibo", "Mawenzi", "Shira")
                  , Lon = c(37.359031, 37.455061, 37.210408)
                  , Lat = c(-3.065053, -3.095436, -3.038222))

peaks = sf::st_as_sf(peaks, crs = 4326, coords = c("Lon", "Lat"))

# visualization
xlim_kili <- c(37.15, 37.55)
ylim_kili <- c(-3.25, -2.9)

if (requireNamespace("raster", quietly = TRUE)) {
  p = raster::spplot(KiLi[[1]], col.regions = "transparent", colorkey = FALSE,
                    xlim = xlim_kili, ylim = ylim_kili,
                    scales = list(draw = TRUE, y = list(rot = 90)),
                    sp.layout = rgb2spLayout(KiLi, quantiles = c(0, 1), alpha = .8)) +
    latticeExtra::layer(sp.points(as(peaks, "Spatial"), cex = 1.5, pch = 20, col = "black"))

  print(p)

  grid::downViewport(lattice::trellis.vpname(name = "figure"))
  offsetGridText(x = sf::st_coordinates(peaks), labels = peaks$Peak,
                xlim = xlim_kili, ylim = ylim_kili, stext = TRUE, offset = .02,
                gp = grid::gpar(fontsize = 16))
}

```

OrcsCppFun

*Dimensions of a data.frame***Description**

Similar to base-R `nrow()`, `ncol()` and `dim()`, this set of functions let's you retrieve the number of rows and columns of a `data.frame`.

**Usage**

```
nrowC(x)
```

```
ncolC(x)
```

```
dimC(x)
```

**Arguments**

`x` A `data.frame`.

**Value**

`dimC()` returns an integer vector of length 2 (number of rows and columns); `nrowC()` (or `ncolC()`) returns the number of rows (or columns) as a single integer.

**Functions**

- `nrowC()`:
- `ncolC()`:
- `dimC()`:

**Author(s)**

Florian Detsch

**Examples**

```
dat <- data.frame(a = 1:4, b = 2:5, c = 3:6)
nrowC(dat)
```

---

par7zip

*Parallelized 7-Zip Compression*

---

**Description**

By calling the Unix terminal or Windows command prompt, this function performs parallelized 7-zip compression of selected files based on the built-in **parallel** package.

**Usage**

```
par7zip(outfile, nodes = 1L, ...)
```

**Arguments**

<code>outfile</code>	Target file for compression as character. A file extension compatible with 7-zip needs to be included, see <b>supported formats</b> . If missing, this defaults to the found input file names with a <code>.7z</code> extension attached.
<code>nodes</code>	Number of cores to use for parallelization as integer, defaults to 1L.
<code>...</code>	Additional arguments passed to <code>list.files()</code> .

**Value**

Output filename(s) as character.

**Author(s)**

Florian Detsch

**See Also**

[list.files\(\)](#), [system\(\)](#)

---

pureBasename

*Return File Name without Extension*

---

**Description**

As opposed to [basename\(\)](#), this function returns the pure basename of one or multiple file names, i.e. without extension.

**Usage**

```
pureBasename(path, slash = FALSE)
```

**Arguments**

path	File name(s) as character.
slash	A logical determining whether to add a leading slash ("/") to the returned file name.

**Value**

File name(s) without extension as character.

**Author(s)**

Florian Detsch

**See Also**

[tools::file\\_path\\_sans\\_ext\(\)](#).

**Examples**

```
if1 = tempfile(fileext = ".tif")
pureBasename(if1)
pureBasename(if1, slash = TRUE)
```

---

pvalue	<i>Get p-Value from 'lm' Object</i>
--------	-------------------------------------

---

**Description**

Retrieve the  $p$ -value associated with a univariate linear regression.

**Usage**

```
pvalue(mod)
```

**Arguments**

mod                    An object of class lm.

**Value**

A numeric  $p$ -value.

**Source**

[retrieving p-values in lm](#) on R-help mailing list.

**See Also**

[stats::lm\(\)](#).

**Examples**

```
## taken from ?lm
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)

pvalue(lm.D9)
```



---

rgb2spLayout	<i>Convert an RGB Raster* to Use with spplot()</i>
--------------	--

---

### Description

This function takes a red-green-blue SpatRaster or Raster\* object and produces a list with color information that can be passed on as 'sp.layout' to [sp::spplot\(\)](#).

### Usage

```
rgb2spLayout(x, quantiles = c(0.02, 0.98), alpha = 1)
```

### Arguments

x	A 3-layered SpatRaster* or Raster* object.
quantiles	Upper and lower quantiles used for color stretching.
alpha	Level of transparency.

### Author(s)

Tim Appelhans, Florian Detsch

### See Also

[terra::plotRGB\(\)](#).

### Examples

```
b = terra::rast(system.file("ex/logo.tif", package="terra"))

## using plotRGB
terra::plotRGB(b)

## convert raster to list
lout = rgb2spLayout(b)
lout_alph = rgb2spLayout(b, alpha = 0.5)

## create random spatial points for plotting
df = data.frame(
  dat = rnorm(100, 2, 1)
  , x = rnorm(100, 50, 20)
  , y = rnorm(100, 50, 25)
)

df = sf::st_as_sf(df, coords = c("x", "y"))

## plot spatial points with rgb background
if (require(sp, quietly = TRUE)) {
  spplot(as(df, "Spatial"), sp.layout = lout)
```

```
  spplot(as(df, "Spatial"), sp.layout = lout_alph)  
}
```

---

**rmDuplCols***Remove Duplicated Columns from data.frame*

---

**Description**

Automatically detect and remove columns from a `data.frame` based on duplicated headers.

**Usage**

```
rmDuplCols(x, keep_first = TRUE, ...)
```

**Arguments**

<code>x</code>	Input <code>data.frame</code> .
<code>keep_first</code>	A logical determining whether the first column of an otherwise duplicated header should be kept, defaults to <code>TRUE</code> .
<code>...</code>	Currently not in use.

**Value**

Revised `data.frame`.

**Author(s)**

Florian Detsch

**See Also**

[duplicated\(\)](#).

**Examples**

```
## sample data  
set.seed(123)  
dat <- data.frame(matrix(rnorm(28), nc = 7))  
names(dat) <- c("Col1", "Col1", "Col1", "Col2", "Col3", "Col3", "Col4")  
  
dat  
rmDuplCols(dat)  
rmDuplCols(dat, keep_first = FALSE)
```

---

`setwdOS`*Set Working Directory Dependent on Current OS*

---

### Description

Similar to `setwd()`, this function sets the working directory to a user-defined path. Rather than supplying a single 'dir' argument, however, both an OS-sensitive path to the desired hard disk partition and, optionally, an extension of this file path are required.

### Usage

```
setwdOS(lin = "/media/permanent/", win = "C:/", ext = NULL)
```

### Arguments

<code>lin, win</code>	Absolute file paths to the Linux and Windows partition as character.
<code>ext</code>	Optional file path extension as character that will be added to 'lin' or 'win' after automatic OS determination.

### Author(s)

Florian Detsch

### See Also

[switch\(\)](#).

### Examples

```
## Not run:  
# desired partition  
setwdOS()  
  
# including file path extension  
setwdOS(ext = "kilimanjaro/nubiscope")  
  
## End(Not run)
```

---

stextGrob

*Draw Shadow Text*


---

### Description

Create and draw shadow text by wrapping a textual expression into a colored framing.

### Usage

```
stextGrob(
  label,
  x = grid::unit(0.5, "npc"),
  y = grid::unit(0.5, "npc"),
  col = "white",
  fill = "black",
  r = 0.1,
  gp = grid::gpar(),
  vp = NULL,
  name = NULL,
  ...
)
```

### Arguments

label	A character or expression vector, see <a href="#">grid::textGrob()</a> .
x, y	Horizontal and vertical text position as <a href="#">grid::unit()</a> objects passed to <a href="#">grid::grid.text()</a> .
col, fill	Framing and fill color passed to <a href="#">grid::gpar()</a> .
r	Blur radius of colored framing as numeric.
name, gp, vp	Graphical parameters passed to <a href="#">grid::gTree()</a> .
...	Additional arguments passed to <a href="#">[grid::grid.text()]</a> .

### Value

A text grob created by [grid::gTree\(\)](#).

### Author(s)

Baptiste Auguie, Florian Detsch

### Examples

```
library(grid)
grid.newpage()
grid.rect(gp = gpar(fill = "grey"))
grid.stext("test")
```

---

**substrC** *Substrings of a Character Vector (C++ Style)*

---

**Description**

Extract substrings from a character vector in C++.

**Usage**

```
substrC(x, pos, len)
```

**Arguments**

x	A character vector.
pos	The start point of the substring as integer. Position indications start from 1L, which is the default in R.
len	The length of the substring as integer.

**Value**

A character vector of the same length as 'x'.

**See Also**

<https://cplusplus.com/reference/string/string/substr/>, [substr\(\)](#).

**Examples**

```
substrC("Hello, world!", pos = 1, len = 5)
```

---

**trimImages** *Remove Whitespace from Images*

---

**Description**

This is a wrapper function around `convert -trim` to automatically remove any whitespace from locally saved images. Note that 'ImageMagick' must be installed on your local system, see [Source](#).

**Usage**

```
trimImages(path = ".", pattern = c(".png$", ".tiff$"))
```

**Arguments**

path	File path leading to image files as character, defaults to the current working directory.
pattern	A regular expression as character accepted by <code>list.files()</code> , defaults to <code>c(".png\$", ".tiff\$")</code> .

**Value**

A character vector containing the names of the processed images.

**Author(s)**

Florian Detsch

**Source**

Ooms J (2018) The **magick** package: Advanced Image-Processing in R. Available online: <https://cran.r-project.org/package=magick/vignettes/intro.html>.

**See Also**

[system\(\)](#)

**Examples**

```
## Not run:
## trim image of bart simpson
download.file("http://pngimg.com/uploads/simpsons/simpsons_PNG93.png?i=1"
             , destfile = (of1 <- file.path(tempdir(), "bart.png", fsep = "\\"))
             , mode = "wb")

par(mfrow = c(1, 2))

img = brick(of1)
plotRGB(img)

jnk = trimImages(tempdir(), "bart.png")
trm = brick(jnk)
plotRGB(trm)

dev.off()

## End(Not run)
```

---

unlistStrsplit	<i>Unlist the Outcome of strsplit()</i>
----------------	---

---

### Description

Per default, `strsplit()` returns a list, with each entry holding the vector of splits of the initial string(s). This function is a simple wrapper that casts `unlist()` upon the returned list to produce a concatenated character vector consisting of the single split elements.

### Usage

```
unlistStrsplit(x, split, ...)
```

### Arguments

x	A character vector with elements to be split.
split	A character vector used for splitting, see <code>strsplit()</code> .
...	Additional arguments passed to <code>strsplit()</code> .

### Author(s)

Florian Detsch

### Examples

```
## 1st example
x <- "This is a test."
unlistStrsplit(x, " ")

## 2nd example; note that 'split' defaults to 'whitespace'
x2 <- "This is a 2nd test."
unlistStrsplit(c(x, x2))
```

---

unsortedFactor	<i>Factor with Unsorted Levels</i>
----------------	------------------------------------

---

### Description

Casting `factor()` upon a (character) vector usually results in alphabetically ordered factor levels. Although this seems reasonable in most cases, the automated ordering of factor levels is seldomly desirable in the context of visualization, e.g. when working with tiled **lattice** or **ggplot2** figures. This function returns a factor with levels ordered according to their first appearance in the supplied vector.

**Usage**

```
unsortedFactor(x, ...)
```

**Arguments**

`x`                    A character vector with elements to converted to factor.  
`...`                 Additional arguments passed to `factor()`.

**Author(s)**

Florian Detsch

**Examples**

```
mnth <- month.abb

## factor levels are being sorted
fc_mnth <- factor(mnth)
levels(fc_mnth)

## factor levels remain unsorted
fc_mnth2 <- unsortedFactor(mnth)
levels(fc_mnth2)
```



# Index

- \* **grob userlevel**
  - stextGrob, 28
- \* **package**
  - Orcs-package, 2
- assignSSH, 3
- basename(), 23
- bookdown::render\_book(), 4
- buildBook, 4
- bumpVersion, 4
- cbind(), 16
- compareDistributions, 5
- coords2Lines, 7
- coords2Lines, Line-method
  - (coords2Lines), 7
- coords2Lines, matrix-method
  - (coords2Lines), 7
- coords2Polygons, 8
- coords2Polygons, matrix-method
  - (coords2Polygons), 8
- coords2Polygons, Polygon-method
  - (coords2Polygons), 8
- data.frame(), 16
- dim(), 21
- dimC (OrcsCppFun), 21
- do.call(), 11, 19
- duplicated(), 26
- evalMetrics, 9
- ext2spy, 10
- factor(), 31, 32
- file.exists(), 11
- graphics::plot(), 6
- graphics::text(), 20
- grid.stext (stextGrob), 28
- grid.stext(), 20
- grid::gpar(), 28
- grid::grid.text(), 20, 28
- grid::gTree(), 28
- grid::textGrob(), 28
- grid::unit(), 20, 28
- ifMissing, 11
- knitr::knit(), 19
- lapply(), 12, 13
- lattice::xyplot(), 6
- latticeCombineGrid, 12
- latticeCombineLayer, 13
- latticeExtra::as.layer(), 14
- latticeExtra::c.trellis(), 12
- library(), 17
- lineEnding, 14
- list.files(), 15, 19, 22, 23, 30
- list2df, 15
- loadFromGit, 16
- loadPkgs, 17
- meanDifference, 17
- meanDifference, numeric-method
  - (meanDifference), 17
- meanDifference, SpatRaster-method
  - (meanDifference), 17
- merge, 18
- merge, list, missing-method (merge), 18
- merge.data.frame(), 18
- multiKnit, 19
- ncol(), 21
- ncolC (OrcsCppFun), 21
- nrow(), 21
- nrowC (OrcsCppFun), 21
- offsetGridText, 20
- Orcs-package, 2
- OrcsCppFun, 21

orcspackage (Orcs-package), 2

par7zip, 22

plotrix::thigmophobe(), 20

pureBasename, 23

pvalue, 24

rbind(), 15, 16

Reduce(), 19

remotes::install\_github(), 16

rgb2splLayout, 25

rmDuplCols, 26

setwd(), 27

setwdOS, 27

sp::Lines(), 7

sp::Polygon(), 8

sp::Polygons(), 8

sp::SpatialLines(), 7

sp::SpatialLinesDataFrame(), 7

sp::SpatialPolygons(), 8

sp::SpatialPolygonsDataFrame(), 8

sp::spplot(), 25

stats::density(), 6

stats::lm(), 24

stextGrob, 28

strsplit(), 31

substr(), 29

substrC, 29

switch(), 27

system(), 15, 23, 30

terra::crs(), 10

terra::ext(), 10

terra::merge(), 18

terra::plotRGB(), 25

terra::rast(), 11

terra::writeRaster(), 11, 18

tools::file\_path\_sans\_ext(), 23

trimImages, 29

unlist(), 31

unlistStrsplit, 31

unsortedFactor, 31

utils::write.table(), 11