Package: soils (via r-universe)

November 5, 2024

Title Visualize and Report Soil Health Data

Version 1.0.0

Description Collection of soil health data visualization and reporting tools, including a RStudio project template with everything you need to generate custom HTML and Microsoft Word reports for each participant in your soil health survey.

License MIT + file LICENSE

URL https://github.com/WA-Department-of-Agriculture/soils, https://wa-department-of-agriculture.github.io/soils/

BugReports https://github.com/WA-Department-of-Agriculture/soils/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, flextable (>= 0.9.6), fs, ggiraph, ggplot2 (>= 3.5.0), ggtext, glue, htmltools, htmlwidgets, knitr, leaflet, officer, purrr, rstudioapi, testthat (>= 3.2.0), tidyr, utils, usethis, quarto

Suggests downloadthis, here, magick, ragg, rlang, rmarkdown, webshot2

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/testthat/edition 3

Config/pak/sysreqs libcairo2-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev git make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libx11-dev

Repository https://wa-department-of-agriculture.r-universe.dev

RemoteUrl https://github.com/WA-Department-of-Agriculture/soils

RemoteRef HEAD

RemoteSha c32ca71e01d07f8e9aac915f67b36da75233019b

Contents

add_legend . . . 2
add_texture_points . . . 4
calculate_mode . . . 5
convert_ggiraph . . . 6
create_soils . . . 7
data_dictionary . . . 8
format_ft_colors . . . 8
get_n_texture_by_var . . . 9
get_table_headers . . . 10
make_ft . . . 10
make_leaflet . . . 11
make_strip_plot . . . 12
make_texture_triangle . . . 14
prep_for_map . . . 15
pull_unique . . . 15
set_scales . . . 16
soils_example . . . 17
style_ft . . . 18
summarize_by_project . . . 19
summarize_by_var . . . 19
theme_facet_strip . . . 20
unit_hline . . . 21
washi_data . . . 22

Index **24**

`add_legend` *Add a legend to the texture triangle*

Description

Add a legend to the texture triangle

Usage

```
add_legend(
  x = 1,
  y = 0.7,
  box = FALSE,
  legend = c("Your fields", "Same county", "Same crop", "Other fields"),
  color = c("#a60f2dCC", "#3E3D3D99", "#3E3D3D99", "#ccc29c80"),
  pch = c(15, 17, 18, 19),
  size = c(2.4, 2.16, 2.16, 1.36),
  vertical_spacing = 1.5,
  ...
)
```

Arguments

x, y	X and Y coordinates used to position the legend. Location may also be specified by setting x to a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", and "center".
box	Boolean. TRUE to draw a box around the legend. Defaults to FALSE for no box.
legend	Character vector to appear in legend.
color	Character vector of the color of the points.
pch	Numeric vector of plotting symbols. See graphics::points() for options and details.
size	Numeric expansion factor for points.
vertical_spacing	Numeric spacing factor for vertical line distances between each legend item.
...	Other arguments passed to graphics::legend().

Value

A list with list components for the legend's box and legend's text(s).

Examples

```
texture <- washi_data |>
  dplyr::select(
    sand = sand_percent,
    silt = silt_percent,
    clay = clay_percent
  )

make_texture_triangle(body_font = "sans")

# Add gray points
add_texture_points(
  tail(texture, 5),
  color = "#3E3D3D90",
  pch = 19
)

# Add red points
add_texture_points(
  head(texture, 5),
  color = "#a60f2dCC",
  pch = 15
)

# Add legend
add_legend(
  legend = c("Red squares", "Gray circles"),
  color = c("#a60f2dCC", "#3E3D3D90"),
  pch = c(15, 19),
  vertical_spacing = 2
```

```
)

# Note the text appears squished in this example since the width, height,
# and resolution have been optimized to print the figure 6 in wide in the
# report.
```

`add_texture_points` *Add points to texture triangle*

Description

To vary color, symbol, and size of points by a grouping variable, call this function once for each value of the grouping variable. Add layers from bottom to top. The below example adds the red points last so they are plotted on top of the gray points.

Usage

```
add_texture_points(
  texture_df = NULL,
  color = "#a60f2dCC",
  pch = 19,
  size = 1.5,
  ...
)
```

Arguments

`texture_df`	Data frame or matrix where each row is a soil sample and three numeric columns contain sand, silt, and clay percentages or proportions. The order of sand, silt, clay is required for correct plotting.
`color`	Color of the points. Defaults to WaSHI red.
`pch`	Numeric value of plotting symbol. See `graphics::points()` for options and details. Defaults to 19, which is a filled-in circle.
`size`	Numeric expansion factor for points. Defaults to 1.5.
`...`	Other arguments passed to `graphics::points()`.

Value

A list of x, y coordinates of the soil textures plotted.

Source

Adapted from plotrix: https://github.com/plotrix/plotrix/blob/0d4c2b065e2c2d327358ac8cdc0b0d46b89bea7f/R/soil.texture.R

Examples

```
texture <- soils::washi_data |>
  dplyr::select(
    sand = sand_percent,
    silt = silt_percent,
    clay = clay_percent
  )

make_texture_triangle(body_font = "sans")

# Add gray points
add_texture_points(
  tail(texture, 5),
  color = "#3E3D3D90",
  pch = 19
)

# Add red points
add_texture_points(
  head(texture, 5),
  color = "#a60f2dCC",
  pch = 15
)

# Note the text appears squished in this example since the width, height,
# and resolution have been optimized to print the figure 6 in wide in the
# report.
```

`calculate_mode` *Calculate the mode of categorical variable*

Description

Calculate the mode of categorical variable

Usage

```
calculate_mode(x)
```

Arguments

`x`	Character vector to calculate mode from.

Value

The value that occurred most often.

Examples

```
calculate_mode(washi_data$crop)
```

---

convert_ggiraph *Convert a* ggplot2 *plot to an interactive* ggiraph

---

Description

Convert a ggplot2 plot to an interactive ggiraph

Usage

```
convert_ggiraph(plot, ..., body_font = "Poppins", width = 6, height = 4)
```

Arguments

plot	ggplot2 plot to convert to interactive ggiraph. plot must contain ggiraph::geom_<plot_type>_inter
...	Other arguments passed to ggiraph::girafe_options().
body_font	Font family to use throughout plot. Defaults to "Poppins".
width	Width of SVG output in inches. Defaults to 6.
height	Height of SVG output in inches. Defaults to 4.

Value

Facetted strip plots with classes of girafe and htmlwidget.

Examples

```
# Read in wrangled example plot data
df_plot_path <- soils_example("df_plot.RDS")
df_plot <- readRDS(df_plot_path)

# Make strip plot with all measurements and set scales based on
# the category column and then apply theme.

# Subset df to just biological measurement group
df_plot_bio <- df_plot |>
  dplyr::filter(measurement_group == "biological")

# NOTE: the plot gets piped into the `set_scales()` function, which gets
# added to `theme_facet_strip()`.
plot <- make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
```

```
    tooltip = label,
    color = category,
    size = category,
    alpha = category,
    shape = category
  ) |>
    set_scales() +
    theme_facet_strip(body_font = "sans")

  # Convert static plot to interactive `ggiraph`
  convert_ggiraph(plot)
```

`create_soils` *Create a project directory for generating soil health reports*

Description

Creates an RStudio project containing Quarto template and resources (images, style sheets, render.R script).

Usage

```
create_soils(path, template = "English", overwrite = FALSE, open = TRUE)
```

Arguments

`path`	Name of project directory to be created.
`template`	Template type. Either "English" or "Spanish".
`overwrite`	Boolean. Overwrite the existing project?
`open`	Boolean. Open the newly created project?

Value

A new project directory containing template and resources.

Source

Adapted from `golem::create_golem()`.

Examples

```
## Not run:
# Create temporary directory
dir <- tempdir()

# Create soils project
create_soils(dir, overwrite = TRUE)
```

```
# Delete temporary directory
unlink(dir, recursive = TRUE)

## End(Not run)
```

`data_dictionary` *Data dictionary*

Description

An example data dictionary for the Washington Soil Health Initiative (WaSHI) State of the Soils Assessment anonymized data.

Usage

```
data_dictionary
```

Format

`data_dictionary` **A data frame with 32 rows and 7 columns.:**

measurement_group Name to group measurements by

measurement_group_label Label of measurement group to be used as heading

column_name Name of column in data set, used for joining

order Order of how measurements are presented within each measurement_group

abbr Abbreviated measurement name for labels

unit Measurement unit

abbr_unit HTML formatted abbreviation with unit for plots and tables ...

Source

WaSHI State of the Soils

`format_ft_colors` *Conditional formatting of flextable background cell colors*

Description

Color the background cells based on how the value compares to the project average. The project average must be the last row of the table. A footnote is added to the table describing what the dark and light colors mean.

Usage

```
format_ft_colors(
  ft,
  lighter_color = "#F2F0E6",
  darker_color = "#ccc29c",
  language = "English"
)
```

Arguments

ft	Flextable object
lighter_color	Lighter background color. Defaults to WaSHI cream.
darker_color	Darker background color. Defaults to WaSHI tan.
language	Language of the footnote. "English" (default) or "Spanish".

Examples

```
# Read in wrangled example table data
tables_path <- soils_example("tables.RDS")
tables <- readRDS(tables_path)

# Make the table
ft <- flextable::flextable(tables$biological)
ft

# Conditionally format background cell colors
format_ft_colors(ft)
```

---

get_n_texture_by_var *Calculate n samples and most frequent texture by a grouping variable*

Description

This function is used in summarize_by_var.

Usage

```
get_n_texture_by_var(results_long, producer_info, var)
```

Arguments

results_long	Dataframe in tidy, long format with columns: sample_id, texture.
producer_info	Vector of producer's values for the grouping variable.
var	Variable to group and summarize by.

get_table_headers *Get table headers for flextable*

Description

This function uses the data dictionary to create a new dataframe of the abbreviations and units for each measurement group for flextable

Usage

```
get_table_headers(dictionary, group)
```

Arguments

dictionary	Dataframe containing columns measurement_group, abbr, unit.
group	Character measurement_group value.

make_ft *Make a flextable with column names from another dataframe*

Description

Make a flextable with column names from another dataframe

Usage

```
make_ft(table, header)
```

Arguments

table	A dataframe with the contents of the desired flextable output.
header	Another dataframe with three columns: • First column contains what the top header row should be. In our template, this is the abbreviation of the measurement (i.e. Organic Matter). • Second column, called "key", contains the join key. In our template, this is the same as the first column. • Third column contains the second header row. In our template, this is the unit (i.e. %).

Value

Formatted flextable object.

Examples

```
# Read in wrangled table data
headers_path <- soils_example("headers.RDS")
headers <- readRDS(headers_path)

tables_path <- soils_example("tables.RDS")
tables <- readRDS(tables_path)

# Input dataframes
headers$chemical

tables$chemical

# Make the flextable
make_ft(
  table = tables$chemical,
  header = headers$chemical
) |>
  # Style the flextable
  style_ft() |>
  # Add the white line under the columns with the same units
  unit_hline(header = headers$chemical)
```

`make_leaflet` *Make leaflet map*

Description

Make leaflet map

Usage

```
make_leaflet(df, primary_color = "#a60f2d")
```

Arguments

`df`	Dataframe containing columns: `longitude`, `latitude`, `label`, `popup`. See `prep_for_map()` for details.
`primary_color`	Color of points. Defaults to WaSHI red.

Value

Leaflet map.

Source

JavaScript code adapted from leaflet.extras.

Examples

```
gis_df <- washi_data |>
  dplyr::distinct(latitude, longitude, .keep_all = TRUE) |>
  head(3) |>
  prep_for_map(label_heading = field_name, label_body = crop)

dplyr::glimpse(gis_df)

# Make leaflet
make_leaflet(gis_df)
```

`make_strip_plot` *Make a facetted strip plot*

Description

Make a facetted strip plot

Usage

```
make_strip_plot(
  df,
  ...,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  language = "English"
)
```

Arguments

`df`	Data frame to plot.
`...`	Other arguments passed to `graphics::points()`.
`x`	Column for x-axis. For these strip plots, we recommend using a dummy variable to act as a placeholder. Defaults to a column named `dummy` with only one value ("dummy") for all rows.
`y`	Column for y-axis. Defaults to `value`.
`id`	Column with unique identifiers for each sample to use as `data_id` for interactive plots. Defaults to `sample_id`.
`group`	Column to facet by. Defaults to `abbr_unit`.
`tooltip`	Column with tooltip labels for interactive plots.
`language`	Language of the footnote. "English" (default) or "Spanish".

Value

Facetted ggplot2 strip plots.

Examples

```
# Read in wrangled example plot data
df_plot_path <- soils_example("df_plot.RDS")
df_plot <- readRDS(df_plot_path)

# Subset df to just biological measurement group
df_plot_bio <- df_plot |>
  dplyr::filter(measurement_group == "biological")

# Make strip plot with all measurements and set scales based on
# the category column and then apply theme.

# NOTE: the plot gets piped into the `set_scales()` function, which gets
# added to `theme_facet_strip()`.

make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  color = category,
  size = category,
  alpha = category,
  shape = category
) |>
  set_scales() +
  theme_facet_strip(body_font = "sans")

# Example of strip plot without scales or theme functions
make_strip_plot(df_plot_bio)

# Example of strip plot with `x` set to the facet group instead of a
# dummy variable.
make_strip_plot(
  df_plot_bio,
  x = abbr_unit,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  color = category,
  size = category,
  alpha = category,
  shape = category
) |>
  set_scales() +
```

```
  theme_facet_strip(body_font = "sans")
```

`make_texture_triangle` *Make a textural class triangle*

Description

Make a texture triangle with USDA textural classes.

Usage

```
make_texture_triangle(
  body_font = "Poppins",
  show_names = TRUE,
  show_lines = TRUE,
  show_grid = FALSE
)
```

Arguments

`body_font`	Font family to use throughout plot. Defaults to `"Poppins"`.
`show_names`	Boolean. Defaults to `TRUE` to show USDA textural class names.
`show_lines`	Boolean. Defaults to `TRUE` to show boundaries of USDA textural classes.
`show_grid`	Boolean. Defaults to `FALSE` to hide grid lines at each 10 level of each soil component.

Value

Opens the graphics device with a triangle plot containing USDA textural classes.

Source

Adapted from plotrix: https://github.com/plotrix/plotrix/blob/0d4c2b065e2c2d327358ac8cdc0b0d46b89bea7f/R/soil.texture.R

Examples

```
# Note the text appears squished in this example since the width, height,
# and resolution have been optimized to print the figure 6 in wide in the
# report.

make_texture_triangle(body_font = "sans")
```

prep_for_map *Prep data to gis df*

Description

Prep data to gis df

Usage

```
prep_for_map(df, label_heading, label_body)
```

Arguments

df	Dataframe containing columns: longitude, latitude, and two columns with values you want to appear in the map label and popup.
label_heading	Column in df that you want to appear as the bold point label on your map, as well as the first line of the popup when the user clicks a point.
label_body	Column in df that you want to appear as body text below the label_heading in the popup.

Value

Dataframe to be input into make_leaflet().

Examples

```
washi_data |>
  dplyr::distinct(latitude, longitude, .keep_all = TRUE) |>
  head(3) |>
  prep_for_map(label_heading = field_name, label_body = crop) |>
  dplyr::glimpse()
```

pull_unique *Pull unique values from one column of dataframe*

Description

Pull unique values from one column of dataframe

Usage

```
pull_unique(df, target)
```

Arguments

df	Dataframe with column to extract unique values from.
target	Variable to pull unique vector of (i.e. crop or county).

Value

Vector of unique values from target column.

Examples

```
washi_data |>
  pull_unique(crop)
```

`set_scales` *Define styles for producer's samples versus all samples*

Description

Define styles for producer's samples versus all samples

Usage

```
set_scales(
  plot,
  primary_color = "#a60f2d",
  secondary_color = "#3E3D3D",
  other_color = "#ccc29c",
  language = "English"
)
```

Arguments

`plot`	`ggplot` object to apply scales to.
`primary_color`	Color of producer's sample points. Defaults to WaSHI red
`secondary_color`	Color of sample points with `"Same crop"` or `"Same county"` values in the `category` column. Defaults to WaSHI gray.
`other_color`	Color of sample points with `"Other fields"` value in `category` column. Defaults to WaSHI tan.
`language`	Language of the footnote. "English" (default) or "Spanish".

Value

`ggplot` object with manual alpha, color, shape, and size scales applied.

Examples

```
# Read in wrangled example plot data
df_plot_path <- soils_example("df_plot.RDS")
df_plot <- readRDS(df_plot_path)
# Subset df to just biological measurement group
df_plot_bio <- df_plot |>
  dplyr::filter(measurement_group == "biological")

# Make strip plot

make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  color = category,
  size = category,
  alpha = category,
  shape = category
) |>
  set_scales() +
  theme_facet_strip(body_font = "sans")

# Example without setting scales
make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label
) +
  theme_facet_strip(body_font = "sans")
```

`soils_example` *Get path to example data*

Description

`soils` comes bundled with some example files in its `inst/extdata` directory. This function make them easy to access.

Usage

```
soils_example(file = NULL)
```

Arguments

file	Name of file. If NULL, the example files will be listed.

Source

Adapted from readxl::readxl_example().

Examples

```
soils_example()

soils_example("df_plot.RDS")
```

---

style_ft *Style a flextable*

Description

Style a flextable

Usage

```
style_ft(
  ft,
  header_font = "Lato",
  body_font = "Poppins",
  header_color = "#023B2C",
  header_text_color = "white",
  border_color = "#3E3D3D"
)
```

Arguments

ft	Flextable object.
header_font	Font of header text. Defaults to "Lato".
body_font	Font of body text. Defaults to "Poppins".
header_color	Background color of header cells. Defaults to WaSHI green.
header_text_color	Color of header text. Defaults to white.
border_color	Color of border lines. Defaults to WaSHI gray.

Value

Styled flextable object.

Examples

```
# Read in wrangled example table data
tables_path <- soils_example("tables.RDS")
tables <- readRDS(tables_path)

# Make the table
ft <- flextable::flextable(tables$biological)
ft

# Style the table
style_ft(ft)
```

`summarize_by_project` *Summarize samples across the project*

Description

Summarize samples across the project

Usage

```
summarize_by_project(results_long)
```

Arguments

`results_long`	Dataframe in tidy, long format with columns: `sample_id`, `texture`, `measurement_group`, `abbr`, `value`.

`summarize_by_var` *Summarize producer's samples with averages by grouping variable*

Description

Summarize producer's samples with averages by grouping variable

Usage

```
summarize_by_var(results_long, producer_samples, var)
```

Arguments

`results_long`	Dataframe in tidy, long format with columns: `sample_id`, `texture`, `measurement_group`, `abbr`, `value`.
`producer_samples`	Dataframe in tidy, long format with columns: `measurement_group`, `abbr`, `value`.
`var`	Variable to summarize by.

`theme_facet_strip` *Theme for facetted strip plots*

Description

Theme for facetted strip plots

Usage

```
theme_facet_strip(
  ...,
  body_font = "Poppins",
  strip_color = "#335c67",
  strip_text_color = "white"
)
```

Arguments

`...`	Other arguments to pass into `ggplot2::theme()`.
`body_font`	Font family to use throughout plot. Defaults to `"Poppins"`.
`strip_color`	Color of facet strip background. Defaults to WaSHI blue.
`strip_text_color`	Color of facet strip text. Defaults to white.

Examples

```
# Read in wrangled example plot data
df_plot_path <- soils_example("df_plot.RDS")
df_plot <- readRDS(df_plot_path)

# Subset df to just biological measurement group
df_plot_bio <- df_plot |>
  dplyr::filter(measurement_group == "biological")

# Make strip plot with all measurements and set scales based on
# the category column and then apply theme.

# NOTE: the plot gets piped into the `set_scales()` function, which gets
# added to `theme_facet_strip()`.

make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  color = category,
```

```
  size = category,
  alpha = category,
  shape = category
) |>
  set_scales() +
  theme_facet_strip(body_font = "sans")

# Example without setting theme
make_strip_plot(
  df_plot_bio,
  x = dummy,
  y = value,
  id = sample_id,
  group = abbr_unit,
  tooltip = label,
  color = category,
  size = category,
  alpha = category,
  shape = category
) |>
  set_scales()
```

`unit_hline` *Add bottom border to specific columns in flextable*

Description

Use when columns with the same units are merged together to add a bottom border to make it more obvious those columns share units.

Usage

```
unit_hline(ft, header)
```

Arguments

`ft`	flextable object
`header`	Another dataframe with three columns: • First column contains what the top header row should be. In our template, this is the abbreviation of the measurement (i.e. `Organic Matter`). • Second column, called `"key"`, contains the join key. In our template, this is the same as the first column. • Third column contains the second header row. In our template, this is the unit (i.e. `%`).

Value

Flextable object with bottom borders added.

Examples

```
# Read in wrangled table data
headers_path <- soils_example("headers.RDS")
headers <- readRDS(headers_path)

tables_path <- soils_example("tables.RDS")
tables <- readRDS(tables_path)

# Input dataframes
headers$chemical

tables$chemical

# Make the flextable
make_ft(
  table = tables$chemical,
  header = headers$chemical
) |>
  # Style the flextable
  style_ft() |>
  # Add the white line under the columns with the same units
  unit_hline(header = headers$chemical)

# Example without `unit_hline()`
make_ft(
  table = tables$chemical,
  header = headers$chemical
) |>
  # Style the flextable
  style_ft()
```

`washi_data` *Example WaSHI data*

Description

A subset of the Washington Soil Health Initiative (WaSHI) State of the Soils Assessment anonymized data. This data set presents each sample in its own row, with columns for each measurement.

Usage

```
washi_data
```

Format

`washi_data` **A data frame with 100 rows and 42 columns::**

year Year of sample

sample_id, producer_id, field_id Anonymized IDs

farm_name, field_name Anonymized names

longitude, latitude Truncated coordinates

texture Measured soil texture

other columns Column name includes measurement and units; value is the measurement result

...

Source

WaSHI State of the Soils

Index

∗ **datasets**
data_dictionary, 8
washi_data, 22

add_legend, 2
add_texture_points, 4

calculate_mode, 5
convert_ggiraph, 6
create_soils, 7

data_dictionary, 8

format_ft_colors, 8

get_n_texture_by_var, 9
get_table_headers, 10
ggiraph::girafe_options(), *6*
ggplot2::theme(), *20*
graphics::legend(), *3*
graphics::points(), *3, 4*, *12*

make_ft, 10
make_leaflet, 11
make_strip_plot, 12
make_texture_triangle, 14

prep_for_map, 15
pull_unique, 15

set_scales, 16
soils_example, 17
style_ft, 18
summarize_by_project, 19
summarize_by_var, 19

theme_facet_strip, 20

unit_hline, 21

washi_data, 22