

Package: soils (via r-universe)

September 30, 2026

Title Visualize and Report Soil Health Data

Version 1.0.1.9000

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 sampling project.

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, gdtools,
ggplot2 (>= 3.5.0), ggrepel, ggtext, glue, htmltools,
htmlwidgets, knitr, lifecycle, leaflet, maptiles, officer,
purrr, quarto, rstudioapi, sf, tidyr, utils, usethis, rlang

Suggests downloadthis, here, magick, ragg, rmarkdown, testthat

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs libabsl-dev libcairo2-dev cmake 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 libuv1-dev libwebp-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

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

Date/Publication 2026-03-31 19:14:00 UTC

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

RemoteRef HEAD

RemoteSha 139c7e04a3a458c0f303e87b063a905df8ca9962

Contents

add_field_count_note	2
add_legend	3
add_texture_points	4
calculate_mode	6
classify_texture	6
convert_ggiraph	9
convert_to_numeric	10
create_soils	11
data_dictionary	12
format_ft_colors	12
get_n_texture_by_var	13
get_table_headers	14
has_complete_row	14
is_column_empty	15
make_ft	15
make_interactive_map	16
make_static_map	17
make_strip_plot	18
make_texture_triangle	20
prep_for_map	21
pull_unique	22
set_scales	22
soils_example	24
style_ft	24
summarize_by_project	25
summarize_by_var	26
theme_facet_strip	26
unit_hline	28
washi_data	29
Index	30

add_field_count_note	<i>Add field count footnote to a flextable</i>
----------------------	--

Description

Adds a footnote explaining that field counts reflect the total number of samples in the crop/county/project group, and that some measurements may be based on fewer samples if results were not available for all fields.

Usage

```
add_field_count_note(ft, language = "English")
```

Arguments

ft	A flextable object
language	"English" (default) or "Spanish"

add_legend	<i>Add a legend to the texture triangle</i>
------------	---

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	<i>Add points to texture triangle</i>
--------------------	---------------------------------------

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 <code>graphics::points()</code> 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 <code>graphics::points()</code> .

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	<i>Calculate the mode of a categorical variable</i>
----------------	---

Description

Returns the most frequently occurring value in a character vector, ignoring missing values.

Usage

```
calculate_mode(x)
```

Arguments

`x` Character vector to calculate the mode from.

Value

A single value (character) that occurs most often in `x`.

Examples

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

classify_texture	<i>Classify USDA soil texture from particle-size fractions</i>
------------------	--

Description

Validates soil particle-size fractions (sand, silt, and clay), completes missing values when possible, and assigns a USDA soil texture class.

Usage

```
classify_texture(df)
```

Arguments

`df` A data frame containing the columns `sample_id`, `sand_percent`, `silt_percent`, and `clay_percent`. An optional texture column can also be provided.

Details

classify_texture() applies the following validation rules and assumptions:

- **Eligibility**
 - Each sample must contain values for at least two of sand_percent, silt_percent, and clay_percent to be eligible for fraction-based texture classification.
- **Errors** (cause validation to fail)
 - All fraction values must fall within the range 0–100.
 - Samples with all three fractions must sum to 100 with a ± 1 tolerance (allowable range 99–101).
- **Warnings** (validation continues)
 - When exactly one fraction is missing, it is calculated as $100 - (\text{sum of the other two})$.
 - Samples with fewer than two provided fractions **and no** provided texture do not have sufficient data for texture classification. These samples are retained and returned with an NA texture value.
- **Special cases** (no warning or error)
 - Samples with fewer than two provided fractions **and** a provided texture will preserve the texture without modification.

Value

A data frame with a texture column containing USDA soil texture classes (if enough soil fraction data are provided). Soil fractions are rounded to whole numbers.

Source

Thresholds for texture classification are based on the [USDA NRCS Soil Texture Calculator](#).

Examples

```
# Three samples classified without error

df <- data.frame(
  sample_id = c(1, 2, 3),
  sand_percent = c(20, 45, 75),
  silt_percent = c(65, 35, 15),
  clay_percent = c(15, 20, 10)
)

classify_texture(df)

# Error when any fraction is outside the allowable range (0-100)

df <- data.frame(
  sample_id = c(1, 2, 3),
  sand_percent = c(40, 0, 65),
  silt_percent = c(40, 55, 5),
  clay_percent = c(20, 110, 30)
```

```
)

try(classify_texture(df))

# Error when fractions do not sum to 100 +/- 1 (allowable range: 99-101)

df <- data.frame(
  sample_id = c(1, 2, 3),
  sand_percent = c(40, 0, 90),
  silt_percent = c(40, 55, 5),
  clay_percent = c(20, 45, 30)
)

try(classify_texture(df))

# Warn when one fraction is missing
# The missing fraction is calculated as 100 minus the other two

df <- data.frame(
  sample_id = c(1, 2, 3),
  sand_percent = c(NA, 60, 25),
  silt_percent = c(45, 10, 40),
  clay_percent = c(50, 30, 35)
)

classify_texture(df)

# Warn when a sample has insufficient data and classification is skipped
# One sample is missing two soil fractions

df <- data.frame(
  sample_id = c(1, 2, 3),
  sand_percent = c(40, NA, 65),
  silt_percent = c(40, 55, 5),
  clay_percent = c(20, NA, 30)
)

classify_texture(df)

# No fractions provided, but texture is supplied
# No warning; existing texture is preserved

df <- data.frame(
  sample_id = c(1, 2),
  sand_percent = c(NA, NA),
  silt_percent = c(NA, NA),
  clay_percent = c(NA, NA),
  texture = c("Loam", "Sandy loam")
)

classify_texture(df)
```

convert_ggiraph	<i>Convert a ggplot2 plot to an interactive ggiraph</i>
-----------------	---

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 <code>ggiraph::geom_<plot_type>_inter</code>
...	Other arguments passed to <code>ggiraph::girafe_options()</code> .
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,
  fill = category,
  size = category,
```

```

    alpha = category,
    shape = category
  ) |>
    set_scales() +
    theme_facet_stripe(body_font = "sans")

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

```

convert_to_numeric	<i>Convert measurement columns to numeric and warn</i>
--------------------	--

Description

Intended for use during data ingestion and validation in the reporting workflow.

Usage

```
convert_to_numeric(data, measurement_cols)
```

Arguments

data	A data frame containing both metadata and quantitative measurement columns.
measurement_cols	A character vector of column names to convert to numeric.

Details

Non-numeric values in specified measurement columns, such as character values commonly used to represent missing or censored measurements (e.g. "ND", "<1", "-", or ""), are converted to NA. Warnings are emitted when non-missing values are converted to NA, and when a column contains only NA values after coercion, indicating it may be omitted from downstream summaries or visualizations.

Value

A data frame where specified measurement columns have been converted to numeric.

Examples

```

# Example data
example_data <- data.frame(
  year      = c(2023, 2023, 2024),
  sample_id = c("S1", "S2", "S3"),
  field_id  = c("A", "A", "B"),
  ph        = c("6.5", "ND", "7.1"), # partial NA (1 ND -> NA)
  nh4_n_mg_kg = c("12.3", "<1", ""), # partial NA (2 non-numeric -> NA)
  no3_n_mg_kg = c("NA", "NA", "NA"), # fully NA
  stringsAsFactors = FALSE
)

```

```

)

# Specify the measurement columns
measurement_cols = c("ph", "nh4_n_mg_kg", "no3_n_mg_kg")

# Convert measurement columns to numeric
clean_data <- convert_to_numeric(example_data, measurement_cols)

```

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	<i>Data dictionary</i>
-----------------	------------------------

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	<i>Conditional formatting of flextable background cell colors</i>
------------------	---

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 number of samples and most frequent texture (deprecated)*

Description

[Deprecated]

Usage

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

Arguments

results_long	Data frame in tidy, long format containing sample_id and texture.
producer_info	Vector of producer values for the grouping variable.
var	Variable to group and summarize by.

Details

This function is deprecated. Its functionality has been incorporated directly into `summarize_by_var()`. Do not use directly in new code.

Value

Deprecated. Previously returned a data frame with n and Texture by group.

See Also

`summarize_by_var`

get_table_headers	<i>Get table headers for flextable</i>
-------------------	--

Description

Internal helper that uses the data dictionary to construct flextable column headers for a single measurement group.

Usage

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

Arguments

dictionary	Data frame containing columns measurement_group, abbr, unit.
group	Character measurement_group value.

has_complete_row	<i>Check if a data frame has at least one complete row for the specified columns</i>
------------------	--

Description

Returns TRUE if there is at least one row in the data frame where all specified columns are non-missing (no NA values).

Usage

```
has_complete_row(df, cols)
```

Arguments

df	A data frame.
cols	A character vector of column names to check.

Value

Logical scalar. TRUE if at least one row has all specified columns complete, FALSE otherwise.

Examples

```
has_complete_row(washi_data, c("crop", "county"))
```

is_column_empty	<i>Check if a column is empty</i>
-----------------	-----------------------------------

Description

Returns TRUE if all values in a vector are missing (NA) or blank strings ("").

Usage

```
is_column_empty(column)
```

Arguments

column	A vector to check.
--------	--------------------

Value

Logical scalar. TRUE if all values are NA or empty strings, FALSE otherwise.

make_ft	<i>Make a flextable with column names from another dataframe</i>
---------	--

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_interactive_map *Make an interactive map of soil sample locations with leaflet*

Description

Create an interactive web map with point locations and popups using **leaflet**. This function is designed to work with data prepared using `prep_for_map()`.

Usage

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

Arguments

<code>df</code>	Dataframe containing longitude, latitude, label, and popup columns. See <code>prep_for_map()</code> for details.
<code>primary_color</code>	A character string specifying the color used for point features on the map (hex code or color name). Defaults to WaSHI red (#a60f2d).

Value

A **leaflet** map widget.

Source

JavaScript code adapted from [leaflet.extras](#).

See Also

`make_static_map()`

Examples

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

dplyr::glimpse(gis_df)

make_interactive_map(gis_df)
```

make_static_map	<i>Make a static map of soil sample locations with ggplot2</i>
-----------------	--

Description

Create a static map using basemap tiles and sample point locations. This function is intended as a non-interactive alternative to `make_interactive_map()`.

Usage

```
make_static_map(
  df,
  label = field_id,
  provider = "Esri.WorldImagery",
  body_font = "sans",
  primary_color = "#a60f2d"
)
```

Arguments

<code>df</code>	Dataframe containing at minimum longitude and latitude columns.
<code>label</code>	Name of the column in <code>df</code> used to label soil sample points.
<code>provider</code>	Character string specifying the basemap tile provider, passed to <code>maptiles::get_tiles()</code> . See details of <code>maptiles::get_tiles()</code> for available providers. Defaults to "Esri.WorldImagery".
<code>body_font</code>	Character string specifying the font family used for map labels. Defaults to "sans".
<code>primary_color</code>	A character string specifying the color used for point features on the map (hex code or color name). Defaults to WaSHI red (#a60f2d).

Value

A {ggplot2} object representing the static map.

Examples

```
gis_df <- washi_data |>
  dplyr::distinct(latitude, longitude, .keep_all = TRUE) |>
  dplyr::select(c(latitude, longitude, farm_name)) |>
  head(3)

dplyr::glimpse(gis_df)

## Not run:
static_map <- make_static_map(gis_df, label = farm_name)
static_map

## End(Not run)
```

make_strip_plot	<i>Make a facettted strip plot</i>
-----------------	------------------------------------

Description

Make a facettted 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 <code>graphics::points()</code> .
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 legend. "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,
  fill = 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. The dummy variable is what centers the points within the subplot.
make_strip_plot(
  df_plot_bio,
  x = abbr_unit,
  y = value,
```

```

    id = sample_id,
    group = abbr_unit,
    tooltip = label,
    color = category,
    fill = 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
```

```
Prepare data for interactive mapping
```

Description

Prepare a dataframe for use in interactive maps by creating formatted label and popup columns from user-specified variables. This function is intended for use prior to `make_interactive_map()`.

Usage

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

Arguments

<code>df</code>	A data frame containing at minimum longitude and latitude columns.
<code>label_heading</code>	A column in <code>df</code> used as the bold heading for each map feature. This value appears as the point label and as the first line of the popup.
<code>label_body</code>	A column in <code>df</code> used as body text displayed below the heading in the popup.

Value

A data frame with additional label and popup columns, suitable for input into `make_interactive_map()`.

Examples

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

pull_unique	<i>Extract unique values from a single column of a data frame</i>
-------------	---

Description

Extract unique values from a single column of a data frame

Usage

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

Arguments

df	A data frame containing the column of interest.
target	Column to extract unique values from (string or unquoted symbol).

Value

A vector of unique values from the specified column.

Examples

```
pull_unique(washi_data, "crop")
```

set_scales	<i>Define styles for producer's samples versus all samples</i>
------------	--

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 legend. "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,
  fill = 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	<i>Get path to example data</i>
---------------	---------------------------------

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	<i>Style a flextable</i>
----------	--------------------------

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

<code>ft</code>	Flextable object.
<code>header_font</code>	Font of header text. Defaults to "Lato".
<code>body_font</code>	Font of body text. Defaults to "Poppins".
<code>header_color</code>	Background color of header cells. Defaults to WaSHI green.
<code>header_text_color</code>	Color of header text. Defaults to white.
<code>border_color</code>	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)
```

<code>summarize_by_project</code>	<i>Summarize samples across the project</i>
-----------------------------------	---

Description

Summarize samples across the project

Usage

```
summarize_by_project(results_long)
```

Arguments

<code>results_long</code>	Dataframe in tidy, long format with columns: <code>sample_id</code> , <code>measurement_group</code> , <code>abbr</code> , <code>value</code> .
---------------------------	---

Value

A data frame summarizing the average values across the project by `measurement_group` and `abbr`. Includes a "Field or Average" column and, if present, a `Texture` column.

summarize_by_var	<i>Summarize producer's samples with averages by grouping variable</i>
------------------	--

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, measurement_group, abbr, and value. If a texture column is present, the most frequent texture (mode) is included in the output.
producer_samples	Dataframe in tidy, long format with columns: measurement_group, abbr, value.
var	Variable to summarize by, which should be present in both results_long and producer_samples.

Value

A data frame with a "Field or Average" column summarizing the average values by measurement group, abbr, and the specified grouping variable. If a texture column is present, the data frame will contain a Texture column containing the most frequent texture class per grouping variable.

theme_facet_strip	<i>Theme for faceted strip plots</i>
-------------------	--------------------------------------

Description

Theme for faceted 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,
  fill = 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,
  fill = 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 41 columns::**

year Year of sample

sample_id, producer_id, field_id Anonymized IDs

farm_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, [12](#)
washi_data, [29](#)

add_field_count_note, [2](#)
add_legend, [3](#)
add_texture_points, [4](#)

calculate_mode, [6](#)
classify_texture, [6](#)
convert_ggiraph, [9](#)
convert_to_numeric, [10](#)
create_soils, [11](#)

data_dictionary, [12](#)

format_ft_colors, [12](#)

get_n_texture_by_var, [13](#)
get_table_headers, [14](#)
ggiraph::girafe_options(), [9](#)
ggplot2::theme(), [27](#)
graphics::legend(), [3](#)
graphics::points(), [3](#), [5](#), [18](#)

has_complete_row, [14](#)

is_column_empty, [15](#)

make_ft, [15](#)
make_interactive_map, [16](#)
make_static_map, [17](#)
make_strip_plot, [18](#)
make_texture_triangle, [20](#)

prep_for_map, [21](#)
pull_unique, [22](#)

set_scales, [22](#)
soils_example, [24](#)
style_ft, [24](#)

summarize_by_project, [25](#)
summarize_by_var, [26](#)

theme_facet_strip, [26](#)

unit_hline, [28](#)

washi_data, [29](#)