Package: irritool (via r-universe)

June 5, 2026

Title Irrigation Tools in R

Version 0.0.0.9000

Description Provides a collection of computational tools for irrigation.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, ggplot2, ncdf4, purrr, rlang, scales, stringr, tibble

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/joaobtj/irritool, https://joaobtj.github.io/irritool/

BugReports https://github.com/joaobtj/irritool/issues

Config/pak/sysreqs libicu-dev libnetcdf-dev

Repository https://joaobtj.r-universe.dev

Date/Publication 2026-04-06 10:10:55 UTC

RemoteUrl https://github.com/joaobtj/irritool

RemoteRef HEAD

RemoteSha 9103023fc139f2df69e7618086d40325aac6dc25

Contents

`calc_backstep` *Calculate pressure and flow profile along a lateral line (Backstep procedure)*

Description

`calc_backstep` calculates the pressure head and flow rate profile along an irrigation lateral line, starting from the distal end (last emitter) and stepping backwards to the inlet.

Usage

```
calc_backstep(
  end_pressure,
  diameter,
  spacing,
  n_emitters,
  slope = 0,
  emitter_coef,
  emitter_exp,
  flow_unit = c("m3/s", "l/h", "m3/h", "l/s"),
  ...
)
```

Arguments

`end_pressure`	Pressure head at the end of the lateral line (last emitter) in meters of water column (mca).
`diameter`	Diameter of the pipe in meters (m).
`spacing`	Emitter spacing in meters (m). Can be a single value or a vector of spacings.
`n_emitters`	Total number of emitters on the lateral line.
`slope`	Elevation change (slope) per meter of pipe (m/m). Positive values indicate uphill slope, negative for downhill.
`emitter_coef`	Emitter discharge coefficient.
`emitter_exp`	Emitter discharge exponent.
`flow_unit`	Flow measurement unit. Options are "m3/s", "l/h", "m3/h", or "l/s". Default is "m3/s".
`...`	Additional arguments passed to `calc_head_loss` (e.g., `method`, `roughness`, `hazen_coef`).

Value

A data frame containing the profile of the lateral line:

- `emitter`: Emitter index (1 is the first emitter near the inlet, n is the last).
- `pressure_head`: Pressure head at each emitter (mca).

- `emitter_flow`: Flow rate discharged by each emitter.
- `section_flow`: Flow rate in the pipe section just upstream of the emitter.
- `head_loss`: Friction head loss in the pipe section upstream of the emitter (m).

Examples

```
# Simple lateral line with 5 emitters, 1m spacing (returns flow in m3/s)
calc_backstep(
  end_pressure = 10, diameter = 0.012, spacing = 1, n_emitters = 5, slope = 0,
  emitter_coef = 8.84e-7, emitter_exp = 0.50
)

# Using Flamant method and returning flow in l/h (common for drip irrigation)
calc_backstep(
  end_pressure = 10, diameter = 0.012, spacing = 0.3, n_emitters = 12, slope = 0,
  emitter_coef = 3.18, emitter_exp = 0.50, flow_unit = "l/h", method = "flamant"
)
```

`calc_head_loss` *Calculate head loss in pipes by different methods*

Description

`calc_head_loss` calculates the friction head loss in pipes using various methods commonly applied in agricultural hydraulics, including Darcy-Weisbach (with multiple friction factor approximations), Hazen-Williams, and Flamant.

Usage

```
calc_head_loss(
  diameter,
  flow_rate,
  flow_unit = c("m3/s", "l/h", "m3/h", "l/s"),
  length,
 method = c("darcy_colebrook", "hazen_williams", "flamant", "swamee_jain", "blasius",
    "haaland", "churchill", "chen"),
  roughness = 1.5e-06,
  hazen_coef = 140,
  flamant_coef = 0.000135,
  viscosity = 1.01e-06,
  gravity = 9.81,
  initial_guess = 0.06
)
```

Arguments

`diameter`	Diameter of the pipe in meters (m).
`flow_rate`	Flow rate value.
`flow_unit`	Flow measurement unit. Options are "m3/s", "l/h", "m3/h", or "l/s". Default is "m3/s".
`length`	Length of the pipe in meters (m).
`method`	The method used to calculate head loss. Options are `"darcy_colebrook"` (default), `"hazen_williams"`, `"flamant"`, `"swamee_jain"`, `"blasius"`, `"haaland"`, `"churchill"`, and `"chen"`.
`roughness`	Absolute roughness of the pipe in meters (m). Used for Darcy-Weisbach methods. Default is `1.5e-6` (typical for PVC and Polyethylene).
`hazen_coef`	Hazen-Williams roughness coefficient (dimensionless). Default is 140 (smooth plastic pipes).
`flamant_coef`	Flamant roughness coefficient. Default is 0.000135 (plastic pipes).
`viscosity`	Kinematic viscosity of the fluid in square meters per second (m^2/s). Default is the value for water at 20 degrees Celsius (`1.01e-6`).
`gravity`	Gravitational acceleration in m/s^2. Default is 9.81.
`initial_guess`	Initial parameter for the Newton-Raphson method (Colebrook). Default is 0.06.

Details

Reference values for Absolute Roughness in meters:

- PVC and Polyethylene (PE): `1.5e-6` to `7.0e-6`
- Aluminum (with couplers): `1.5e-4` to `2.0e-4`
- Galvanized Steel: `1.5e-4`
- Cast Iron (new): `2.6e-4`

Value

A numeric value representing the friction head loss in meters (m).

References

- Blasius, H. (1913). Das Ähnlichkeitsgesetz bei Reibungsvorgängen in Flüssigkeiten. *Forschungsheft*, 131, 1-40.
- Bernardo, S., Soares, A. A., & Mantovani, E. C. (2019). *Manual de Irrigação* (9th ed.). Editora UFV.
- Chen, N. H. (1979). An explicit equation for friction factor in pipe. *Industrial & Engineering Chemistry Fundamentals*, 18(3), 296-297.
- Churchill, S. W. (1977). Friction-factor equation spans all fluid-flow regimes. *Chemical Engineering*, 84(24), 91-92.
- Colebrook, C. F. (1939). Turbulent flow in pipes... *Journal of the Institution of Civil Engineers*, 11(4), 133-156.

- Dunlop, E. J. (1991). *Wallingford software: The hydraulic friction of pipes*. Report SR 281.
- Haaland, S. E. (1983). Simple and explicit formulas for the friction factor... *Journal of Fluids Engineering*, 105(1), 89-90.
- Swamee, P. K., & Jain, A. K. (1976). Explicit equations for pipe-flow problems. *Journal of the Hydraulics Division*, 102(5), 657-664.
- Williams, G. S., & Hazen, A. (1905). *Hydraulic tables*. John Wiley & Sons.

Examples

```
# Darcy-Weisbach with Colebrook-White (default)
calc_head_loss(diameter = 0.05, flow_rate = 10, flow_unit = "m3/h", length = 100)

# Hazen-Williams for a PVC pipe
calc_head_loss(diameter = 0.05, flow_rate = 10, flow_unit = "m3/h", length = 100,
method = "hazen_williams")
```

---

calc_kc_curve *Generate Crop Coefficient (Kc) Curve*

Description

Constructs a time series of daily crop coefficient (Kc) values and generates a ggplot2 visualization representing the Kc curve over the crop cycle, following FAO-56 guidelines. The curve is divided into four stages: initial, development, mid-season, and late-season.

Usage

```
calc_kc_curve(kc_points, stage_lengths, crop = NULL)
```

Arguments

kc_points	Numeric vector of length 3. Contains the crop coefficients for the initial stage (Kc_ini), mid-season stage (Kc_mid), and end of the late-season stage (Kc_end).
stage_lengths	Numeric vector of length 4. The duration (in days) of each growth stage: initial, development, mid-season, and late-season.
crop	Character (optional). Name of the crop, used to customize the plot title.

Value

A list containing:

- kc_serie: A numeric vector of daily Kc values over the entire cycle.
- kc_plot: A ggplot2 object displaying the constructed curve with key annotations.
- kc_data: A data frame containing the days and their corresponding Kc values.

References

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop evapotranspiration - Guidelines for computing crop water requirements. FAO Irrigation and drainage paper 56.

Examples

```
kc_points <- kc_params_lettuce <- c(0.7, 1.0, 0.95) # Kc_ini, Kc_mid, Kc_end
stage_lengths <- stage_lengths_lettuce <- c(15, 20, 20, 10) # L_ini, L_dev, L_mid, L_late
lettuce_data <- calc_kc_curve(
  kc_points = kc_params_lettuce,
  stage_lengths = stage_lengths_lettuce,
  crop = "Lettuce"
)
print(lettuce_data$kc_data)
print(lettuce_data$kc_plot)
```

---

calc_water_balance *Calculate Daily Soil Water Balance*

Description

Calculates the daily soil water balance for a crop over a specified period. The function accounts for reference evapotranspiration, rainfall, crop coefficients, soil properties, and irrigation rules to estimate actual crop evapotranspiration (ETc), soil water depletion, irrigation needs, and water surplus.

Usage

```
calc_water_balance(
  et0,
  rainfall,
  daily_kc_values,
  root_depth,
  theta_fc,
  theta_wp,
  depletion_factor,
  initial_depletion = 0,
  day_numbers = NULL,
  irrigation_rule = "threshold"
)
```

Arguments

et0	Numeric vector. Daily reference evapotranspiration (mm/day).
rainfall	Numeric vector. Daily rainfall (mm/day).
daily_kc_values	Numeric vector. Daily crop coefficient (Kc) values.

`root_depth` Numeric vector or single value. Daily root depth (mm). If a single value is provided, it applies to all days.

`theta_fc` Numeric. Volumetric soil water content at field capacity (m^3/m^3).

`theta_wp` Numeric. Volumetric soil water content at wilting point (m^3/m^3).

`depletion_factor`
Numeric. The fraction of Total Available Water (TAW) that a crop can extract from the root zone without suffering water stress (the 'p' factor from FAO-56). Must be between 0 and 1.

`initial_depletion`
Numeric. Initial soil water depletion at the start of day 1 (mm). Defaults to 0 (field capacity).

`day_numbers` Numeric vector (optional). Sequence of days (e.g., day of the year). Defaults to a sequence from 1 to the length of `daily_kc_values`.

`irrigation_rule`
Character or Integer. Defines the irrigation trigger. Use `"threshold"` (default) to irrigate when depletion exceeds Readily Available Water (RAW), or a positive integer `N` to irrigate at a fixed N-day interval.

Value

A list containing:

`water_balance_data`
A data.frame with the detailed daily soil water balance components: `day`: Day number. `rainfall`: Input rainfall (mm). `et0`: Input reference ET0 (mm). `root_depth`: Root depth for the day (mm). `taw`: Total Available Water in the root zone (mm). `raw`: Readily Available Water in the root zone (mm). `depletion_start`: Soil water depletion at the start of the day (mm). `kc`: Crop coefficient for the day. `ks`: Water stress coefficient (0-1). `etc`: Actual crop evapotranspiration (mm). `depletion_end`: Soil water depletion at the end of the day before irrigation (mm). `irrigation_applied`: Irrigation depth applied (mm). `water_surplus`: Water surplus (deep percolation or runoff) (mm).

`summary_depths` A list with cumulative water balance components: `total_rainfall`: Total input rainfall (mm). `total_water_surplus`: Total water surplus (mm). `net_rainfall`: Total rainfall minus total water surplus (mm). `total_etc`: Total actual crop evapotranspiration (mm). `total_irrigation_applied`: Total irrigation depth applied (mm). `irrigation_events_count`: Number of irrigation events.

References

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop evapotranspiration - Guidelines for computing crop water requirements. FAO Irrigation and drainage paper 56.

Examples

```
# For a standalone example, let's create a dummy daily_kc_values
daily_kc_lettuce <- c(
  rep(0.7, 15),
```

```
  seq(0.7, 1.05, length.out = 20),
  rep(1.05, 20),
  seq(1.05, 0.9, length.out = 10)
) # Total 65 days

set.seed(123) # for reproducibility
sim_days <- length(daily_kc_lettuce)
et0_data <- round(runif(sim_days, 2, 5), 1)
rain_data <- round(sample(c(rep(0, sim_days - 5), runif(5, 3, 30))), 0)

root_depth_val <- 300 # mm
p_factor_val <- 0.55
theta_fc_val <- 0.30 # m3/m3
theta_wp_val <- 0.15 # m3/m3

water_balance_results <- calc_water_balance(
  et0 = et0_data,
  rainfall = rain_data,
  daily_kc_values = daily_kc_lettuce,
  root_depth = root_depth_val,
  theta_fc = theta_fc_val,
  theta_wp = theta_wp_val,
  depletion_factor = p_factor_val,
  initial_depletion = 0, # Starting at field capacity
  irrigation_rule = "threshold"
)

print(head(water_balance_results$water_balance_data))
print(water_balance_results$summary_depths)

# Example with fixed interval irrigation (every 7 days)
water_balance_fixed_interval <- calc_water_balance(
  et0 = et0_data,
  rainfall = rain_data,
  daily_kc_values = daily_kc_lettuce,
  root_depth = root_depth_val,
  theta_fc = theta_fc_val,
  theta_wp = theta_wp_val,
  depletion_factor = p_factor_val,
  irrigation_rule = 7
)
print(water_balance_fixed_interval$summary_depths$total_irrigation_applied)
print(water_balance_fixed_interval$summary_depths$irrigation_events_count)
```

`get_brdwgd_weather` *Extract Point Data from BR-DWGD*

Description

Extracts a daily time series of specified weather variables for a given longitude and latitude from the Brazilian Daily Weather Gridded Data (BR-DWGD) NetCDF files. The function identifies the

closest grid cell to the target coordinates.

Usage

```
get_brdwgd_weather(
  target_longitude,
  target_latitude,
  weather_variables = c("pr", "Tmin", "Tmax", "Rs", "RH", "u2", "ETo"),
  date_range = NULL,
  nc_files_directory = "."
)
```

Arguments

`target_longitude`	Numeric. Longitude of the target location in decimal degrees.
`target_latitude`	Numeric. Latitude of the target location in decimal degrees.
`weather_variables`	Character vector. Variables to extract. Allowed values are: `"pr"` (precipitation), `"Tmin"`, `"Tmax"`, `"Rs"` (solar radiation), `"RH"` (relative humidity), `"u2"` (wind speed), and `"ETo"`. Defaults to all.
`date_range`	Character vector of length 2 (`"YYYY-MM-DD"`). The start and end dates. If `NULL` (default), extracts the entire available period.
`nc_files_directory`	Character. Path to the local directory containing the BR-DWGD NetCDF files. Defaults to current working directory `"."`.

Value

A `tibble` containing the `date`, the actual `lat` and `lon` of the closest grid cell, and columns for each requested weather variable.

References

Xavier, A. C., King, C. W., & Scanlon, B. R. (2016). Daily gridded meteorological variables in Brazil (1980–2013). International Journal of Climatology. https://sites.google.com/site/alexandrecandidoxavierufes/brazilian-daily-weather-gridded-data

Examples

```
target_longitude <- -50.5995
target_latitude <- -27.2863
weather_variables <- c("pr", "ETo", "Tmax")
date_range <- c("1983-01-01", "1983-03-31")
nc_files_directory <- "D:/clima_Xavier"

## Not run:
# Ensure the path to your NetCDF files is correct
weather_data_point <- get_brdwgd_weather(
```

```
  target_longitude = -50.5995,
  target_latitude = -27.2863,
  weather_variables = c("pr", "ETo", "Tmax"),
  date_range = c("1983-01-01", "1983-03-31"),
  nc_files_directory = "D:/clima_Xavier"
)
print(head(weather_data_point))

## End(Not run)
```

Index