Package: nwsrfsr (via r-universe)

July 22, 2026

Title NWS Hydrology Models: SAC-SMA, SNOW17, UH, CONSUSE, CHANLOSS

Version 1.0.3

Description Interface to the National Weather Service operational hydrology models, Sacramento Soil Moisture Accounting (SAC-SMA), Snow Accumulation and Ablation (SNOW17). Also provides an interface to the unit hydrograph routing model (UH), consumptive use (CONSUSE) and channel loss/gain modules (CHANLOSS). The Fortran code used in this package is considered ``legacy'' and is not supported officially by NWS, but it should not have any significant differences from current operational models.

License Apache License (>= 2)

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Suggests testthat

URL https://github.com/NOAA-NWRFC/nwsrfs-hydro-models, https://noaa-nwrfc.github.io/nwsrfs-hydro-models/

BugReports https://github.com/NOAA-NWRFC/nwsrfs-hydro-models/issues

Config/testthat/edition 3

Depends R (>= 4.1.0)

NeedsCompilation yes

Author Cameron Bracken [aut, cre] (ORCID: <https://orcid.org/0000-0003-1917-402X>), Geoffrey Walters [aut] (ORCID: <https://orcid.org/0009-0009-3804-8898>), Eric Anderson [ctb] (Original SNOW-17 and SAC-SMA Fortran code (NOAA/HRL)), George F. Smith [ctb] (Original LAG-K Fortran code (NOAA/HRL)), Janice M. Lewis [ctb] (Interpolation and LAG-K Fortran code (NOAA/HRL)), John E. Pask [ctb] (Original sorting routines (LLNL); see inst/COPYRIGHTS), Ondrej Certik [ctb]

(MIT-licensed fortran-utils sorting and types modules; see inst/COPYRIGHTS), John Burkardt [ctb] (MIT-licensed implementation of Brent's zero finder; see inst/COPYRIGHTS), National Oceanic and Atmospheric Administration [cph] (International-rights holder for U.S. Government-authored components, including legacy NWSRFS Fortran and contributions by NOAA employees (C. Bracken through 2022-06-30, G. Walters); see inst/COPYRIGHTS), Battelle Memorial Institute [cph] (Copyright holder for PNNL-authored contributions by C. Bracken from 2022-07-01 onward; see inst/COPYRIGHTS)

Maintainer Cameron Bracken <cameron.bracken@pnnl.gov>

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

Date/Publication 2026-07-22 08:10:02 UTC

RemoteUrl https://github.com/cran/nwsrfsr

RemoteRef HEAD

RemoteSha 8b6c85c43cc2b2d92649dd3374963d36de032a28

Contents

---

adc3 *Areal depeletion curve using a 3 parameter model*

Description

adc=a*x^b+(1.0-a)*x^c

Usage

```
adc3(a, b, c)
```

Arguments

a	a parameter (0<a<1)
b	b parameter (b>=0)
c	c parameter (c>=0)

Value

11 element vector representing the ADC

Examples

```
adc <- adc3(.5, 0.1, 2)
```

`adjustq` *AdjustQ: Create upstream flow timeseries from observations and simulation*

Description

Replicates the CHPS FEWS AdjustQ tool. Adjusts observed mean daily discharges using observed instantaneous and simulated discharges. If both observed mean daily and instantaneous data are missing, simulated discharge is used.

Usage

```
adjustq(
  daily_flow,
  inst_flow,
  sim = NULL,
  blend = 10L,
  interp_type = "ratio",
  error_tol = 0.01,
  max_iterations = 15L
)
```

Arguments

`daily_flow`	Data.frame with columns year, month, day, flow_cfs (daily obs)
`inst_flow`	Data.frame with columns year, month, day, hour, flow_cfs (instantaneous obs)
`sim`	Data.frame with columns year, month, day, hour, flow_cfs (simulated 6hr flow), or NULL. If provided, small/large gap filling uses simulation as fallback.
`blend`	Integer; threshold for small vs large gap in timesteps. Default 10.
`interp_type`	"ratio" or "difference". Default "ratio".
`error_tol`	Numeric; daily volume matching tolerance (fraction). Default 0.01.
`max_iterations`	Integer; max daily correction iterations. Default 15.

Value

A data.frame with columns: datetime (POSIXct), flow_cfs (adjusted flow)

adjustq_load_example *Load AdjustQ example*

Description

Runs AdjustQ using bundled NRKW1 data.

Usage

```
adjustq_load_example(sim = TRUE, ...)
```

Arguments

sim	Logical; include simulation in AdjustQ. Default TRUE.
...	Additional arguments passed to adjustq()

Value

A data.frame (see adjustq())

area_elev_curve *Area-elevation curve for the zones of SFLN2*

Description

A dataset containing the percent area covered at a complete range of elevations for the two zones of Salmon Falls Creek (SFLN2). Used as input to rsnwelev().

Usage

```
area_elev_curve
```

Format

A data.frame with 21 rows and 3 columns:

quantile Cumulative area fraction of the basin below the reference elevation

SFLN2-1 Reference elevations for zone 1 (ft)

SFLN2-2 Reference elevations for zone 2 (ft)

chanloss *Seasonal chanloss*

Description

Seasonal chanloss

Usage

```
chanloss(flow, forcing, dt_hours, pars)
```

Arguments

flow	streamflow vector
forcing	forcing data
dt_hours	timestep in hours
pars	parameters

Value

Vector of flow modified by the chanloss pattern

consuse *Daily consuse model*

Description

Daily consuse model

Usage

```
consuse(input, pars, cfs = TRUE)
```

Arguments

input	A data frame (or matrix with col names), must have columns: flow, pet (units of mm), year, month, day
pars	model parameters in the same format as the sac and snow models, with type=='consuse'
cfs	if TRUE, then flow units of cfs are expected, if FALSE then cms are expected.

Value

data frame with consuse variables

`fa_adj_nwrfc` *Conduct NWRFC style forcing adjustments*

Description

Conduct NWRFC style forcing adjustments

Usage

```
fa_adj_nwrfc(
  dt_hours,
  forcing,
  pars,
  climo = NULL,
  dry_run = FALSE,
  return_climo = FALSE
)
```

Arguments

`dt_hours`	timestep in hours
`forcing`	data frame with with columns for forcing inputs
`pars`	sac parameters
`climo`	climotology matrix
`dry_run`	Do a run without any forcing adjustments, only compute pet and etd
`return_climo`	Return the computed climo, instead of adjustments

Value

Matrix (1 column per zone) of unrouted channel inflow

Examples

```
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
adj <- fa_adj_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
```

`fa_nwrfc` *Conduct NWRFC style forcing adjustments*

Description

Conduct NWRFC style forcing adjustments

Usage

```
fa_nwrfc(
  dt_hours,
  forcing,
  pars,
  climo = NULL,
  dry_run = FALSE,
  return_adj = FALSE,
  return_climo = FALSE
)
```

Arguments

`dt_hours`	timestep in hours
`forcing`	data frame with with columns for forcing inputs
`pars`	sac parameters
`climo`	climotology matrix
`dry_run`	Do a run without any forcing adjustments, only compute pet and etd
`return_adj`	return monthly adjustment factors only
`return_climo`	return the computed monthly climo

Value

Matrix (1 column per zone) of unrouted channel inflow

Examples

```
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
```

`forcing_adjust_map_pet_ptps` *Adjust monthly climo based on 4 parameters*

Description

description

Usage

```
forcing_adjust_map_pet_ptps(
  climo,
  pars,
  ll = 0.9 * climo,
  ul = 1.1 * climo,
  return_climo = FALSE
)
```

Arguments

`climo`	blah
`pars`	blah
`ll`	blah
`ul`	blah
`return_climo`	blah

Value

stuff

Examples

```
climo <- rep(2, 12)
pars <- c(.5, 0, 10, 0)
forcing_adjust_map_pet_ptps(climo, pars)
```

forcing_adjust_mat *Adjust monthly climo based on 4 parameters*

Description

description

Usage

```
forcing_adjust_mat(
  climo,
  pars,
  ll = climo * ifelse(climo > 0, 0.9, 1.1),
  ul = climo * ifelse(climo > 0, 1.1, 0.9),
  return_climo = FALSE
)
```

Arguments

climo	blah
pars	blah
ll	blah
ul	blah
return_climo	blah

Value

stuff

Examples

```
climo <- rep(2, 12)
pars <- c(.5, 0, 10, 0)
forcing_adjust_mat(climo, pars)
```

format_states *Format state output from sac_snow_states*

Description

Format state output from sac_snow_states

Usage

```
format_states(x)
```

Arguments

x	output list from sac_snow_states

Value

a data.frame with formatted output

---

interp_fa *Interpolate forcing adjustment factors*

Description

This function interpolates 12 forcing adjustment factors (1 per month) by placing them at the 15th of the month then interpolating between the previous and next months values for every time step in between.

Usage

```
interp_fa(factors, month, day, hour)
```

Arguments

factors	12 element vector of monthly adjustment factors
month	a vector of month values for each time step
day	a vector of day values for each time step
hour	a vector of hour values for each time step

Value

A vector matching the length of month, containing interpolated adjustment factors

Examples

```
d1 <- as.POSIXct("2001-01-01 00:00:00", tz = "UTC")
d2 <- as.POSIXct("2001-12-31 18:00:00", tz = "UTC")
dates <- seq.POSIXt(d1, d2, by = "6 hours")

month <- as.integer(format(dates, "%m"))
day <- as.integer(format(dates, "%d"))
hour <- as.integer(format(dates, "%H"))
factors <- c(.5, 2, 1, 1.5, 2, .5, 1, 2.5, 3, -1.5, 0, 1)

ifa <- interp_fa(factors, month, day, hour)

plot(dates, ifa, t = "l")
points(as.POSIXct(paste0("2001-", 1:12, "-15"), tz = "UTC"), factors, col = "red")
```

lagk *Lag-K Routing for any number of upstream points*

Description

Lag-K Routing for any number of upstream points

Usage

```
lagk(dt_hours, uptribs, pars, sum_routes = TRUE, return_states = FALSE)
```

Arguments

dt_hours	timestep in hours
uptribs	a matrix where each column contains flow data (in cfs) for an upstream point
pars	parameters
sum_routes	add all routed values together or leave separate
return_states	return the lagk states

Value

vector of routed flows

Examples

```
NULL
```

load_example *Load and run a bundled example dataset*

Description

Loads NRKW1 or SFLN2 bundled example data and runs the full NWSRFS model chain.

Usage

```
load_example(lid = "NRKW1", forcing_adj = TRUE, shift_sf = TRUE)
```

Arguments

lid	Station identifier: "NRKW1" or "SFLN2"
forcing_adj	Logical; apply forcing adjustments. Default TRUE.
shift_sf	Logical; shift UH flow forward one timestep. Default TRUE.

Value

A list with class "nwsrfs_run" (see `nwsrfs_run()`)

Examples

```
run = load_example("NRKW1")
plot(run$sim, type = "l")
```

`nrkw1_daily_flow` *NRKW1 observed daily flow*

Description

Daily average observed streamflow at NRKW1.

Usage

```
nrkw1_daily_flow
```

Format

A data.frame with columns:

year,month,day Date columns

flow_cfs Daily average streamflow (cfs)

Source Data source identifier

`nrkw1_forcing` *NRKW1 forcing data*

Description

6-hour forcing data for Nooksack River at North Cedarville (NRKW1), 2 zones, period of record.

Usage

```
nrkw1_forcing
```

Format

A named list of 2 data.frames (NRKW1-1, NRKW1-2), each with columns:

year,month,day,hour Datetime columns

map_mm Mean areal precipitation (mm)

mat_degc Mean areal temperature (deg C)

ptps Fraction of precipitation as snow

nrkw1_inst_flow *NRKW1 observed instantaneous flow*

Description

Instantaneous (6-hour) observed streamflow at NRKW1.

Usage

```
nrkw1_inst_flow
```

Format

A data.frame with columns:

year,month,day,hour Datetime columns

flow_cfs Instantaneous streamflow (cfs)

Source Data source identifier

nrkw1_pars *NRKW1 optimal parameters*

Description

Optimal parameters from NWRFC autocalibration for Nooksack River at North Cedarville, WA (USGS-12210700). 2-zone SAC-SMA/SNOW17/UH model with 3-reach Lag-K routing (6 hour timestep).

Usage

```
nrkw1_pars
```

Format

A data.frame with 5 columns:

p_name Unique parameter identifier

name Parameter name

type Model type (sac, snow, uh, fa, lagk)

zone Zone or upstream reach name

value Parameter value

nrkw1_upflow *NRKW1 upstream flow data*

Description

6-hour upstream flow timeseries for 3 upstream reaches (MFNW1, NFNW1, NSSW1) routed via Lag-K to Nooksack River at North Cedarville.

Usage

```
nrkw1_upflow
```

Format

A named list of 3 data.frames, each with columns:

year,month,day,hour Datetime columns

flow_cfs Streamflow (cfs)

nwsrfs_run *Run the full NWSRFS model chain*

Description

Reads parameter, forcing, flow, and upflow CSVs from an NWRFC autocalibration directory, auto-detects which model components are present, and runs the full chain: FA -> SAC-SMA/SNOW17 -> UH -> Lag-K -> Chanloss -> Consuse.

Usage

```
nwsrfs_run(autocalb_dir, run_dir = NULL, forcing_adj = TRUE, shift_sf = TRUE)
```

Arguments

autocalb_dir	Path to an NWRFC autocalibration directory
run_dir	Name of results subdirectory (e.g. "results_por_02"). If NULL, uses the first results_* directory found.
forcing_adj	Logical; apply monthly climatological forcing adjustments. Default TRUE.
shift_sf	Logical; shift UH-derived streamflow forward one timestep (required for NWRFC calibrations). Default TRUE.

Details

The returned list contains:

sim Numeric vector of simulated flow (cfs)

sacsnow_sf Numeric vector of SAC-SMA/SNOW17/UH flow per zone (cfs), or NULL

sacsnow_tci Matrix of total channel inflow per zone (mm), or NULL

lagk_route Numeric vector of Lag-K routed flow (cfs), or NULL

forcings List of adjusted forcing data.frames, or NULL

pars Parameter data.frame

forcing_raw List of raw (unadjusted) forcing data.frames

upflow List of upstream flow data.frames, or NULL

daily_flow Daily observed flow data.frame

inst_flow Instantaneous observed flow data.frame, or NULL

dt_hours Model timestep in hours

zone_names Character vector of zone names

upflow_names Character vector of upstream reach names, or NULL

localflow_logic Logical; SAC-SMA/SNOW17/UH present

upflow_logic Logical; Lag-K routing present

chanloss_logic Logical; chanloss present

consuse_logic Logical; consuse present

Value

A list with class "nwsrfs_run" containing model results and metadata. See Details.

`nwsrfsr` *nwsrfsr: NWS River Forecast System Hydrologic Models*

Description

R interface to NWSRFS Fortran hydrologic models used operationally by NOAA's National Weather Service River Forecast Centers. Includes SAC-SMA soil moisture accounting, SNOW-17 snow accumulation/ablation, gamma unit hydrograph routing, Lag-K channel routing, channel loss/gain, and consumptive use modules.

Low-level model wrappers

Individual model components callable via Fortran interface: `sac_snow()`, `uh()`, `lagk()`, `chanloss()`, `consuse()`, `fa_nwrfc()`

High-level orchestration

`nwsrfs_run()` reads NWRFC autocalibration directories and runs the full model chain. `load_example()` provides bundled example data for NRKW1 and SFLN2.

AdjustQ preprocessing

`adjustq()` replicates the CHPS FEWS AdjustQ tool for creating upstream flow timeseries from observed and simulated data.

Author(s)

Maintainer: Cameron Bracken <cameron.bracken@pnnl.gov> (ORCID)

Authors:

- Geoffrey Walters <geoffrey.walters@noaa.gov> (ORCID)

Other contributors:

- Eric Anderson (Original SNOW-17 and SAC-SMA Fortran code (NOAA/HRL)) [contributor]
- George F. Smith (Original LAG-K Fortran code (NOAA/HRL)) [contributor]
- Janice M. Lewis (Interpolation and LAG-K Fortran code (NOAA/HRL)) [contributor]
- John E. Pask (Original sorting routines (LLNL); see inst/COPYRIGHTS) [contributor]
- Ondrej Certik (MIT-licensed fortran-utils sorting and types modules; see inst/COPYRIGHTS) [contributor]
- John Burkardt (MIT-licensed implementation of Brent's zero finder; see inst/COPYRIGHTS) [contributor]
- National Oceanic and Atmospheric Administration (International-rights holder for U.S. Government-authored components, including legacy NWSRFS Fortran and contributions by NOAA employees (C. Bracken through 2022-06-30, G. Walters); see inst/COPYRIGHTS) [copyright holder]
- Battelle Memorial Institute (Copyright holder for PNNL-authored contributions by C. Bracken from 2022-07-01 onward; see inst/COPYRIGHTS) [copyright holder]

See Also

Useful links:

- https://github.com/NOAA-NWRFC/nwsrfs-hydro-models
- https://noaa-nwrfc.github.io/nwsrfs-hydro-models/
- Report bugs at https://github.com/NOAA-NWRFC/nwsrfs-hydro-models/issues

pet_hs *Daily Potential Evapotranspiration using Hargreaves-Semani equations*

Description

Daily Potential Evapotranspiration using Hargreaves-Semani equations

Usage

```
pet_hs(lat, jday, tave, tmax, tmin)
```

Arguments

lat	Latitude in decimal degrees
jday	Julian day (Day of year since Jan 1)
tave	Average daily temperature (C)
tmax	Max daily temperature (C)
tmin	Min daily temerature (C)

Value

Daily PET (vectorized over all inputs)

Examples

```
pet <- pet_hs(42, 200, 20, 25, 15)
```

print.nwsrfs_run *Print method for nwsrfs_run objects*

Description

Print method for nwsrfs_run objects

Usage

```
## S3 method for class 'nwsrfs_run'
print(x, ...)
```

Arguments

x	An nwsrfs_run object
...	Ignored

Value

x (invisibly)

`rsnwelev` *Replace ptps column with ptps derived using rain snow line code (lapse rate + MAT)*

Description

Replace ptps column with ptps derived using rain snow line code (lapse rate + MAT)

Usage

```
rsnwelev(forcing, pars, ae_tbl)
```

Arguments

`forcing`	named list of per-zone forcing data frames (each with a `mat_degc` column). The list names are matched against the parameter `zone` column.
`pars`	parameter data frame that supplies one value per forcing zone for each of `elev` (reference elevation, m), `talr` (temperature lapse rate, degC per 100 m) and `pxtemp` (rain/snow threshold temperature, degC). A `zone` column is used to align values with `forcing` when present.
`ae_tbl`	data.frame with col1 containing quantile info and subsequent col with elev for each zone

Value

The `forcing` list with each zone's `ptps` column replaced by the value derived from the rsnwelev model.

Examples

```
# area_elev_curve is bundled for SFLN2 (2 zones); pair it with SFLN2 data.
data(sfln2_forcing)
data(sfln2_pars)
data(area_elev_curve)
# rsnwelev only needs the zones that match the area-elevation table,
# so drop the SFLN2-CU consumptive-use zone before calling.
forcing_zones <- sfln2_forcing[c("SFLN2-1", "SFLN2-2")]
# rsnwelev needs elev plus a temperature lapse rate (talr, degC/100m) and a
# rain/snow threshold (pxtemp, degC) per zone. The bundled calibration only
# carries elev, so add representative talr/pxtemp rows for the two zones.
zones <- c("SFLN2-1", "SFLN2-2")
rs_pars <- rbind(
  sfln2_pars[sfln2_pars$name == "elev" & sfln2_pars$zone %in% zones, ],
  data.frame(
    p_name = paste0(rep(c("talr_", "pxtemp_"), each = 2), zones),
```

```
    name = rep(c("talr", "pxtemp"), each = 2),
    type = "rsnwelev",
    zone = rep(zones, 2),
    value = c(0.5, 0.5, 1.0, 1.0)
  )
)
forcing_adj <- rsnwelev(forcing_zones, rs_pars, area_elev_curve)
```

---

sac_snow *Execute SAC-SMA, SNOW17, return total channel inflow per zone, and model states*

Description

Execute SAC-SMA, SNOW17, return total channel inflow per zone, and model states

Usage

```
sac_snow(dt_hours, forcing, pars, return_states = FALSE)
```

Arguments

dt_hours	timestep in hours
forcing	data frame with with columns for forcing inputs
pars	sac parameters
return_states	logical value indicating if the states should be output as well as the tci

Value

data.frame (1 column per zone) of unrouted channel inflow (tci), sac states uztwc, uzfwc, lztwc, lzfsc, lzfpc, adimc, and snow water equivalent (swe), and adjusted forcing data.

Examples

```
# Simple: full model chain via the bundled example
run <- load_example("NRKW1")
head(run$sacsnow_tci)

# Low-level: populate etd_mm / pet_mm via fa_nwrfc first,
# then call the SAC-SMA / SNOW17 wrapper directly. sac_snow() requires
# forcing data frames that already contain etd_mm.
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
tci <- sac_snow(dt_hours, forcing_adj, nrkw1_pars)
```

sac_snow_states *Execute SAC-SMA, SNOW17, return total channel inflow per zone, and model states*

Description

Execute SAC-SMA, SNOW17, return total channel inflow per zone, and model states

Usage

```
sac_snow_states(dt_hours, forcing, pars)
```

Arguments

dt_hours	timestep in hours
forcing	data frame with with columns for forcing inputs
pars	sac parameters

Value

data.frame (1 column per zone) of unrouted channel inflow (tci), sac states uztwc, uzfwc, lztwc, lzfsc, lzfpc, adimc, and snow water equivalent (swe), and adjusted forcing data.

Examples

```
# Simple: full model chain via the bundled example, tci and states returned together
run <- load_example("NRKW1")
head(run$sacsnow_tci)

# Low-level: populate etd_mm / pet_mm via fa_nwrfc first, then
# get per-zone tci and states directly.
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
states <- sac_snow_states(dt_hours, forcing_adj, nrkw1_pars)
```

sac_snow_uh *Execute SAC-SMA, SNOW17 and UH with given parameters*

Description

Execute SAC-SMA, SNOW17 and UH with given parameters

Usage

```
sac_snow_uh(dt_hours, forcing, pars)
```

Arguments

dt_hours	timestep in hours
forcing	data frame with columns for forcing inputs
pars	sac parameters

Value

Vector of routed flow in cfs

Examples

```
# Simple: full model chain via the bundled example
run <- load_example("NRKW1")
head(run$sim)

# Low-level: populate etd_mm / pet_mm via fa_nwrfc first,
# then call the combined SAC-SMA / SNOW17 / UH wrapper.
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
flow_cfs <- sac_snow_uh(dt_hours, forcing_adj, nrkw1_pars)
```

---

sac_snow_uh_lagk *Execute SAC-SMA, SNOW17, UH and LAG-K with given parameters*

Description

Execute SAC-SMA, SNOW17, UH and LAG-K with given parameters

Usage

```
sac_snow_uh_lagk(dt_hours, forcing, uptribs, pars)
```

Arguments

dt_hours	timestep in hours
forcing	data frame with columns for forcing inputs
uptribs	data frame with columns for upstream flow data
pars	sac parameters

Value

Vector of routed flow in cfs

Examples

```
# Simple: full model chain via the bundled example (NRKW1 has upstream tribs)
run <- load_example("NRKW1")
head(run$sim)

# Low-level: populate etd_mm / pet_mm via fa_nwrfc first, then chain
# SAC-SMA / SNOW17 / UH with Lag-K routing of upstream tributaries.
data(nrkw1_forcing)
data(nrkw1_pars)
data(nrkw1_upflow)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
flow_cfs <- sac_snow_uh_lagk(dt_hours, forcing_adj, nrkw1_upflow, nrkw1_pars)
```

---

scale_uplimit *Find a reasonable scale upper limit for optimization*

Description

Find a reasonable scale upper limit for optimization

Usage

```
scale_uplimit(shape, dt_hours)
```

Arguments

shape	shape parameter
dt_hours	timestep

Value

numeric value indicating the upper limit for the scale

---

sfc_pressure *Conputes surface pressure in hPa from a given elevation*

Description

Conputes surface pressure in hPa from a given elevation

Usage

```
sfc_pressure(elev)
```

Arguments

elev	surface elevation in meters

Value

Surface pressure in hPa

Examples

```
sp <- sfc_pressure(0)
```

---

sfln2_daily_flow *SFLN2 observed daily flow*

Description

Daily average observed streamflow at SFLN2.

Usage

```
sfln2_daily_flow
```

Format

A data.frame with columns:

year,month,day Date columns

flow_cfs Daily average streamflow (cfs)

Source Data source identifier

---

sfln2_forcing *SFLN2 forcing data*

Description

6-hour forcing data for Salmon Falls Creek (SFLN2), 2 zones plus a consumptive use zone, period of record.

Usage

```
sfln2_forcing
```

Format

A named list of 3 data.frames (SFLN2-1, SFLN2-2, SFLN2-CU), each with columns:

year,month,day,hour Datetime columns

map_mm Mean areal precipitation (mm)

mat_degc Mean areal temperature (deg C)

ptps Fraction of precipitation as snow

---

sfln2_inst_flow *SFLN2 observed instantaneous flow*

Description

Instantaneous (6-hour) observed streamflow at SFLN2.

Usage

```
sfln2_inst_flow
```

Format

A data.frame with columns:

year,month,day,hour Datetime columns

flow_cfs Instantaneous streamflow (cfs)

Source Data source identifier

---

sfln2_pars *SFLN2 optimal parameters*

Description

Optimal parameters from NWRFC autocalibration for Salmon Falls Creek NR San Jacinto NV (USGS-13105000). 2-zone SAC-SMA/SNOW17/UH model with chanloss and consuse (6 hour timestep).

Usage

```
sfln2_pars
```

Format

A data.frame with 5 columns:

p_name Unique parameter identifier

name Parameter name

type Model type (sac, snow, uh, fa, chanloss, consuse)

zone Zone name

value Parameter value

---

uh *Two parameter unit hydrograph routing for one or more basin zones*

Description

Two parameter unit hydrograph routing for one or more basin zones

Usage

```
uh(
  dt_hours,
  tci,
  pars,
  sum_zones = TRUE,
  start_of_timestep = TRUE,
  backfill = TRUE
)
```

Arguments

dt_hours	timestep in hours
tci	channel inflow matrix, one column per zone
pars	parameters
sum_zones	should routed flows from multiple zones be added and returned as a vector, or kept separate and returned as a matrix
start_of_timestep	should the output flow data be shifted by one timestep to account for forcing data that uses beginning of timestep labeling
backfill	when start_of_timestep is TRUE, should the first value be duplicated

Value

Vector of routed flow in cfs

Examples

```
# Simple: full model chain via the bundled example; routed flow per zone
run <- load_example("NRKW1")
head(run$sacsnow_sf)

# Low-level: run FA -> SAC-SMA/SNOW17 to get TCI, then route it with UH.
data(nrkw1_forcing)
data(nrkw1_pars)
dt_hours <- 6
forcing_adj <- fa_nwrfc(dt_hours, nrkw1_forcing, nrkw1_pars)
tci <- sac_snow(dt_hours, forcing_adj, nrkw1_pars)
flow_cfs <- uh(dt_hours, tci, nrkw1_pars)
```

---

uh2p *R port of the nwsrfs UH fortran code*

Description

Returns ordinates for a 2 parameter (shape,scale) gamma unit hydrograph. The ordinates are based on the given timestep (in hours). To match the Fortran code, a max length is used. A warning is issued if the UH does not terminate before the given max length.

Usage

```
uh2p(shape, scale, timestep, max_len = 1000)
```

Arguments

shape	gamma shape parameter
scale	gamma scale parameter
timestep	timestep in hours
max_len	max length of the uh

Value

vector of ordinates for a 2 parameter (shape,scale) gamma unit hydrograph

Examples

```
dt <- 6
shape <- 2
scale <- 1
y <- uh2p(2, 1, 6)
x <- seq(dt, dt * length(y), by = dt)
plot(x, y, t = "l")
```

uh2p_cfs_in *Create a 2 parameter gamma unit hydrograph with units cfs/in*

Description

Create a 2 parameter gamma unit hydrograph with units cfs/in

Usage

```
uh2p_cfs_in(shape, scale, timestep, area)
```

Arguments

shape	gamma shape parameter
scale	gamma scale parameter
timestep	timestep in hours
area	basin area in square miles

Value

stuff

Examples

```
dt <- 6
shape <- 2
scale <- 1
y <- uh2p_cfs_in(2, 1, 6, 1000)
x <- seq(dt, dt * length(y), by = dt)
plot(x, y, t = "l")
```

uh2p_get_scale *Get the scale parameter from a 2 parameter gamma unit hydrograph given the shape parameter and time of concentration.*

Description

Get the scale parameter from a 2 parameter gamma unit hydrograph given the shape parameter and time of concentration.

Usage

```
uh2p_get_scale(shape, toc, dt_hours)
```

Arguments

shape	gamma shape parameter
toc	time of concentration (hours)
dt_hours	UH timestep (hours)

Value

scalar scale parameter

Examples

```
uh2p_get_scale(2, 50, 1)
```

---

uh2p_get_scale_r *Get the scale parameter from a 2 parameter gamma unit hydrograph given the shape parameter and time of concentration.*

Description

Get the scale parameter from a 2 parameter gamma unit hydrograph given the shape parameter and time of concentration.

Usage

```
uh2p_get_scale_r(shape, toc, dt_hours)
```

Arguments

shape	gamma shape parameter
toc	time of concentration (hours)
dt_hours	UH timestep (hours)

Value

stuff

Examples

```
uh2p_get_scale_r(2, 50, 1)
```

uh2p_root — *root finding function for finding a scale parameter given shape and toc*

Description

root finding function for finding a scale parameter given shape and toc

Usage

```
uh2p_root(scale, shape, dt_hours, toc)
```

Arguments

scale	scale parameter
shape	shape parameter
dt_hours	timestep
toc	time of concentration

Value

function value for root finding

uh2p_seek — *Objective function for finding a scale parameter given shape and toc*

Description

Objective function for finding a scale parameter given shape and toc

Usage

```
uh2p_seek(scale, shape, dt_hours, toc)
```

Arguments

scale	blah
shape	blah
dt_hours	blah
toc	blah

Value

objective function value

uh2p_seek2 *Objective function for finding a scale parameter given shape and toc*

Description

Objective function for finding a scale parameter given shape and toc

Usage

```
uh2p_seek2(scale, shape, dt_hours, toc)
```

Arguments

scale	blah
shape	blah
dt_hours	blah
toc	blah

Value

function value for root finding

update_pars *Update parameters and re-run model chain*

Description

Update parameters and re-run model chain

Usage

```
update_pars(run, new_pars)
```

Arguments

run	An "nwsrfs_run" object from nwsrfs_run() or load_example()
new_pars	A data.frame with columns "p_name" and "value" containing parameters to update. All p_name values must exist in the current pars.

Value

A new "nwsrfs_run" object with updated parameters

Index