

# Package ‘soilhypfit’

July 24, 2025

**Version** 0.1-8

**Date** 2025-07-24

**Type** Package

**Title** Modelling of Soil Water Retention and Hydraulic Conductivity  
Data

**Author** Andreas Papritz [aut, cre]

**Maintainer** Andreas Papritz <papritz@retired.ethz.ch>

**Depends** nloptr(>= 1.2.1), snowfall

**Imports** graphics, mgcv, quadprog(>= 1.5-7), parallel, Rmpfr(>= 0.7-2),  
stats, SoilHyP(>= 0.1.3), utils

**Suggests** lattice

**SystemRequirements** gmp (>= 4.2.3), mpfr (>= 3.0.0)

**SystemRequirementsNote** 'MPFR' (MP Floating-Point Reliable Library,  
<http://mpfr.org/>) and 'GMP' (GNU Multiple Precision library,  
<http://gmplib.org/>) are required for Rmpfr

**Description** Provides functions for efficiently estimating properties of  
the Van Genuchten-Mualem model for soil hydraulic parameters from  
possibly sparse soil water retention and hydraulic conductivity data  
by multi-response parameter estimation methods  
(Stewart, W.E., Caracotsios, M. Soerensen, J.P. (1992)  
``Parameter estimation from multi-response data"  
<[doi:10.1002/aic.690380502](https://doi.org/10.1002/aic.690380502)>). Parameter estimation is simplified  
by exploiting the fact that residual and saturated water contents  
and saturated conductivity are conditionally linear parameters  
(Bates, D. M. and Watts, D. G. (1988)  
``Nonlinear Regression Analysis and Its Applications"  
<[doi:10.1002/9780470316757](https://doi.org/10.1002/9780470316757)>).  
Estimated parameters are optionally constrained by the evaporation  
characteristic length (Lehmann, P., Bickel, S., Wei, Z. and Or, D. (2020)  
``Physical Constraints for Improved Soil Hydraulic Parameter  
Estimation by Pedotransfer Functions" <[doi:10.1029/2019WR025963](https://doi.org/10.1029/2019WR025963)>)  
to ensure that the estimated parameters are physically valid.  
Common S3 methods and further utility functions allow to process,  
explore and visualise estimation results.

**License** GPL (>= 2) | LGPL-3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2025-07-24 10:00:03 UTC

## Contents

soilhypfit-package . . . . .	2
control_fit_wrc_hcc . . . . .	7
evaporative-length . . . . .	16
fit_wrc_hcc . . . . .	19
hc_model . . . . .	28
profile_loglikelihood . . . . .	30
sim_wrc_hcc . . . . .	36
soilhypfitS3methods . . . . .	38
swissforestsoils . . . . .	41
utility-functions . . . . .	43
vcov . . . . .	45
wc_model . . . . .	47
<b>Index</b>	<b>50</b>

---

soilhypfit-package	<i>The soilhypfit Package</i>
--------------------	-------------------------------

---

## Description

This is a summary of the features and functionality of **soilhypfit**, a package in R for parameteric modelling of soil water retention and hydraulic conductivity data.

## Details

### Estimation approach:

The main function, `fit_wrc_hcc`, estimates parameters of models for soil water retention and hydraulic conductivity by *maximum likelihood* (ml, default), *maximum posterior density* (mpd) *estimation* (Stewart et al., 1992) or nonlinear *weighted least squares* (wls) from data on volumetric *soil water content*,  $\theta^T = (\theta_1, \theta_2, \dots, \theta_{n_\theta})$ , and/or *soil hydraulic conductivity*,  $\mathbf{K}^T = (K_1, K_2, \dots, K_{n_K})$ , both measured at given capillary pressure head,  $\mathbf{h}^T = (h_1, h_2, \dots)$ .

For mpd and ml estimation, the models for the measurements are

$$\theta_i = \theta(h_i; \boldsymbol{\mu}, \boldsymbol{\nu}) + \varepsilon_{\theta,i}, \quad i = 1, 2, \dots, n_\theta,$$

$$\log(K_j) = \log(K(h_j; \boldsymbol{\mu}, \boldsymbol{\nu})) + \varepsilon_{K,j}, \quad j = 1, 2, \dots, n_K,$$

where  $\theta(h_i; \boldsymbol{\mu}, \boldsymbol{\nu})$  and  $K(h_j; \boldsymbol{\mu}, \boldsymbol{\nu})$  denote modelled water content and conductivity,  $\boldsymbol{\mu}$  and  $\boldsymbol{\nu}$  are the *conditionally linear* and *nonlinear* model parameters (see below and Bates and Watts, 1988,

sec. 3.3.5), and  $\varepsilon_{\theta,i}$  and  $\varepsilon_{K,j}$  are independent, normally distributed errors with zero means and variances  $\sigma_\theta^2$  and  $\sigma_K^2$ , respectively.

Let

$$Q_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{h}) = (\boldsymbol{\theta} - \boldsymbol{\theta}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu}))^\top \mathbf{W}_\theta (\boldsymbol{\theta} - \boldsymbol{\theta}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu})),$$

and

$$Q_K(\boldsymbol{\mu}, \boldsymbol{\nu}; \mathbf{K}, \mathbf{h}) = (\log(\mathbf{K}) - \log(\mathbf{K}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu})))^\top \mathbf{W}_K (\log(\mathbf{K}) - \log(\mathbf{K}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu}))),$$

denote the (possibly weighted) residual sums of squares between measurements and modelled values.  $\boldsymbol{\theta}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu})$  and  $\mathbf{K}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\nu})$  are vectors with modelled values of water content and hydraulic conductivity, and  $\mathbf{W}_\theta$  and  $\mathbf{W}_K$  are optional diagonal matrices with weights  $w_{\theta,i}$  and  $w_{K,j}$ , respectively. The weights are products of *case weights*  $w'_{\theta,i}$  and  $w'_{K,j}$  and *variable weights*  $w_\theta$ ,  $w_K$ , hence  $w_{\theta,i} = w_\theta w'_{\theta,i}$  and  $w_{K,j} = w_K w'_{K,j}$ .

The objective function for ml and mpd estimation is equal to (Stewart et al., 1992, eqs 14 and 15, respectively)

$$Q(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h}) = \frac{\kappa_\theta}{2} \log(Q_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{h})) + \frac{\kappa_K}{2} \log(Q_K(\boldsymbol{\mu}, \boldsymbol{\nu}; \mathbf{K}, \mathbf{h})),$$

where  $\kappa_v = n_v + 2$  for mpd and  $\kappa_v = n_v$  for ml estimation,  $v \in (\theta, K)$ , and weights  $w_{\theta,i} = w_{K,j} = 1$ . Note that  $Q(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})$  is equal to the negative logarithm of the concentrated loglikelihood or the concentrated posterior density, obtained by replacing  $\sigma_\theta^2$  and  $\sigma_K^2$  by their conditional maximum likelihood or maximum density estimates  $\hat{\sigma}_\theta^2(\boldsymbol{\mu}, \boldsymbol{\nu})$  and  $\hat{\sigma}_K^2(\boldsymbol{\mu})$  respectively (Stewart et al., 1992, p. 642).

For wls the objective function is equal to

$$Q(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h}) = Q_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{h}) + Q_K(\boldsymbol{\mu}, \boldsymbol{\nu}; \mathbf{K}, \mathbf{h}).$$

If either water content or conductivity data are not available, then the respective terms are omitted from  $Q(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})$ .

The function `fit_wrc_hcc` does not attempt to estimate the parameters by minimising  $Q(\boldsymbol{\mu}, \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})$  directly with respect to  $\boldsymbol{\mu}$  and  $\boldsymbol{\nu}$ . Rather, it exploits the fact that for *given nonlinear parameters*  $\boldsymbol{\nu}$ , the *conditionally linear parameters*  $\boldsymbol{\mu}^T = (\theta_r, \theta_s, \log(K_0))$  can be estimated straightforwardly by minimising the *conditional residual sums of squares*

$$Q_\theta^*(\theta_r, \theta_s; \boldsymbol{\theta}, \mathbf{h}, \boldsymbol{\nu}) = \left( \boldsymbol{\theta} - [\mathbf{1}, \mathbf{S}(\mathbf{h}; \boldsymbol{\nu})] \begin{bmatrix} \theta_r \\ \theta_s - \theta_r \end{bmatrix} \right)^\top \mathbf{W}_\theta \left( \boldsymbol{\theta} - [\mathbf{1}, \mathbf{S}(\mathbf{h}; \boldsymbol{\nu})] \begin{bmatrix} \theta_r \\ \theta_s - \theta_r \end{bmatrix} \right)$$

with respect to  $\theta_r$  and  $\theta_s - \theta_r$  and/or

$$Q_K^*(K_0; \mathbf{K}, \mathbf{h}, \boldsymbol{\nu}) = (\log(\mathbf{K}) - \log(K_0 \mathbf{k}(\mathbf{h}; \boldsymbol{\nu})))^\top \mathbf{W}_K (\log(\mathbf{K}) - \log(K_0 \mathbf{k}(\mathbf{h}; \boldsymbol{\nu}))),$$

with respect to  $\log(K_0)$ , where  $\mathbf{1}$  is a vector of ones,  $\mathbf{S}(\mathbf{h}; \boldsymbol{\nu})^\top = (S(h_1; \boldsymbol{\nu}), \dots, S(h_{n_\theta}; \boldsymbol{\nu}))$  and  $\mathbf{k}(\mathbf{h}; \boldsymbol{\nu})^\top = (k(h_1; \boldsymbol{\nu}), \dots, k(h_{n_K}; \boldsymbol{\nu}))$  are vectors of modelled *water saturation* and modelled *relative conductivity* values,  $\theta_r$  and  $\theta_s$  are the *residual* and *saturated water content*, and  $K_0$  is the *saturated hydraulic conductivity*.

Unconstrained conditional estimates, say  $\hat{\theta}_r(\boldsymbol{\nu})$ ,  $\hat{\theta}_s(\boldsymbol{\nu}) - \hat{\theta}_r(\boldsymbol{\nu})$  and  $\widehat{\log(K_0)}(\boldsymbol{\nu})$  can be easily obtained from the normal equations of the respective (weighted) least squares problems, and

quadratic programming yields conditional (weighted) least squares estimates that honour the inequality constraints  $0 \leq \theta_r \leq \theta_s \leq 1$ .

Let  $\hat{\boldsymbol{\mu}}(\boldsymbol{\nu})^T = (\hat{\theta}_r(\boldsymbol{\nu}), \hat{\theta}_s(\boldsymbol{\nu}), \log(\hat{K}_0)(\boldsymbol{\nu}))$  be the conditional estimates of the linear parameters obtained by minimising  $Q_\theta^*(\theta_r, \theta_s; \boldsymbol{\theta}, \mathbf{h}, \boldsymbol{\nu})$ , and  $Q_K^*(K_0; \mathbf{K}, \mathbf{h}, \boldsymbol{\nu})$ , respectively. `fit_wrc_hcc` then estimates the nonlinear parameters by minimising  $Q(\hat{\boldsymbol{\mu}}(\boldsymbol{\nu}), \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})$  with respect to  $\boldsymbol{\nu}$  by a nonlinear optimisation algorithm.

For mpd and ml estimation the variances of the model errors are estimated by (Stewart et al., 1992, eq. 16)

$$\hat{\sigma}_\theta^2 = \frac{Q_\theta(\hat{\boldsymbol{\mu}}(\hat{\boldsymbol{\nu}}), \hat{\boldsymbol{\nu}}; \boldsymbol{\theta}, \mathbf{h})}{\kappa_\theta},$$

and

$$\hat{\sigma}_K^2 = \frac{Q_K(\hat{\boldsymbol{\mu}}(\hat{\boldsymbol{\nu}}), \hat{\boldsymbol{\nu}}; \mathbf{K}, \mathbf{h})}{\kappa_K}.$$

Furthermore, for mpd and ml estimation, the covariance matrix of the estimated nonlinear parameters may be approximated by the inverse Hessian matrix of  $Q(\hat{\boldsymbol{\mu}}(\boldsymbol{\nu}), \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})$  at the solution  $\hat{\boldsymbol{\nu}}$  (Stewart and Sørensen, 1981), i.e.

$$\text{Cov}[\hat{\boldsymbol{\nu}}, \hat{\boldsymbol{\nu}}^T] \approx \mathbf{A}^{-1},$$

where

$$[\mathbf{A}]_{kl} = \frac{\partial^2}{\partial \nu_k \partial \nu_l} Q(\hat{\boldsymbol{\mu}}(\boldsymbol{\nu}), \boldsymbol{\nu}; \boldsymbol{\theta}, \mathbf{K}, \mathbf{h})|_{\boldsymbol{\nu}=\hat{\boldsymbol{\nu}}}.$$

### Details on parameter estimation:

*Models for water retention curves and hydraulic conductivity functions:*

Currently, `fit_wrc_hcc` allows to estimate the parameters of the simplified form of the Van Genuchten-Mualem (VGM) model (Van Genuchten, 1980) with the restriction  $m = 1 - \frac{1}{n}$ , see `wc_model` and `hc_model`. This model has the following parameters:

- $\boldsymbol{\mu}^T = (\theta_r, \theta_s, K_0)$  (see above) and
- $\boldsymbol{\nu}^T = (\alpha, n, \tau)$  where  $\alpha$  is the inverse air entry pressure,  $n$  the shape and  $\tau$  the tortuosity parameter.

Any of these parameters can be either estimated from data or kept fixed at the specified initial values (see arguments `param` and `fit_param` of `fit_wrc_hcc`).

*Imposing physical constraints on the estimated parameters:*

Parameters of models for the water retention curve and the hydraulic conductivity function may vary only within certain bounds (see `wc_model`, `hc_model` and `param_boundf` for allowed ranges). `fit_wrc_hcc` either estimates *transformed parameters* that vary over the whole real line and can therefore be estimated without constraints (see `param_transf`), or it uses algorithms (quadratic programming for estimating  $\boldsymbol{\mu}$ , nonlinear optimisation algorithms with box constraints for estimating  $\boldsymbol{\nu}$ ) that restrict estimates to permissible ranges, see *Details* section of `control_fit_wrc_hcc`.

In addition, for natural soils, the parameters of the VGM model cannot vary independently from each other over the allowed ranges. Sets of fitted parameters should always result in soil hydraulic quantities that are physically meaningful. One of these quantities is the *characteristic length*  $L_c$  of *stage-I* evaporation from a soil (Lehmann et al., 2008).  $L_c$  can be related to the parameters of the VGM model, see Lehmann et al. (2008, 2020) and `evaporative-length`.

Using several soil hydrological databases, Lehmann et al. (2020) analysed the mutual dependence of VGM parameters and proposed regression equations to relate the inverse air entry pressure  $\alpha$  and the saturated hydraulic  $K_0$  to the shape parameter  $n$ , which characterises the width of the pore size distribution of a soil. Using these relations, Lehmann et al. (2020) then computed the expected value (“target”)  $L_t$  of  $L_c$  for given  $n$  and tortuosity parameter  $\tau$ , see [evaporative-length.fit\\_wrc\\_hcc](#) allows to constrain estimates of the nonlinear parameters  $\nu$  by defining permissible lower and upper bounds for the ratio  $L_c/L_t$ , see arguments `ratio_lc_lt_bound` of [fit\\_wrc\\_hcc](#) and settings of [control\\_fit\\_wrc\\_hcc](#).

*Choice of optimisation algorithm for estimating the nonlinear parameters:*

To estimate  $\nu$ , [fit\\_wrc\\_hcc](#) minimises  $Q(\hat{\mu}(\nu), \nu; \theta, K, h)$  either by a nonlinear optimisation algorithm available in the library *NLopt* (Johnson, see [nloptr](#)) or by the Shuffled Complex Evolution (SCE) optimisation algorithm (Duan et al., 1994, see [SCEoptim](#)). The choice of the algorithm is controlled by the argument settings of the function [control\\_fit\\_wrc\\_hcc](#):

1. global optimisation without constraints for the ratio  $L_c/L_t$   
(settings = "uglobal" or settings = "sce"),
2. global optimisation with inequality constraints for the ratio  $L_c/L_t$   
(settings = "cglobal"),
3. local optimisation without constraints for the ratio  $L_c/L_t$   
(settings = "ulocal"),
4. local optimisation with inequality constraints for the ratio  $L_c/L_t$   
(settings = "clocal").

The settings argument also sets reasonable default values for the termination (= convergence) criteria for the various algorithms, see [NLopt documentation, section Termination conditions](#). The *NLopt* documentation contains a very useful discussion of (constrained) optimisation problems in general, [global vs. local optimisation](#) and [gradient-based vs. derivative-free algorithms](#). Note that besides the settings argument of [control\\_fit\\_wrc\\_hcc](#), the arguments `nloptr` and `sce` along with the functions [control\\_nloptr](#) and [control\\_sce](#) allow to fully control the nonlinear optimisation algorithms, see [control\\_fit\\_wrc\\_hcc](#) for details.

*Computing initial values of parameters:*

For local optimisation algorithms “good” initial values of  $\nu$  are indispensable for successful estimation. [fit\\_wrc\\_hcc](#) allows to compute initial values of  $\alpha$  and  $n$  for the Van Genuchten model from water retention data by the following procedure:

1. Smooth the water retention data,  $(\theta_i, y_i = \log(h_i)), i = 1, 2, \dots, n_\theta$ , by an additive model.
2. Determine the saturation,  $S^*$ , and the logarithm of capillary pressure head,  $y^* = \log(h^*)$ , at the inflection point of the additive model fit.
3. Find the root, say  $\hat{m}$ , of  $S^* = (1 + 1/m)^{-m}$ . One obtains the right-hand side of this equation by solving  $\frac{\partial^2}{\partial y^2} [S_{VG}(\exp(y); \nu)] = 0$  for  $y$  and plugging the result into the expression for  $S_{VG}(\exp(y); \nu)$ , see [wc\\_model](#).
4. Compute  $\hat{n} = 1/(1 - \hat{m})$  and  $\hat{\alpha} = 1/\exp(y^*) (1/\hat{m})^{1-\hat{m}}$ . The second expression is again a result of solving  $\frac{\partial^2}{\partial y^2} [S_{VG}(\exp(y); \nu)] = 0$ .

Initial values for local optimisation algorithms can of course also be obtained by first estimating the parameters by a global algorithm. These estimates can be “refined” in a second step by a local unconstrained algorithm, possibly followed by a third call of [fit\\_wrc\\_hcc](#) to constrain the estimated parameters by the ratio  $L_c/L_t$ . The method [coef.fit\\_wrc\\_hcc](#) can be used to extract the estimated parameters from an object of class `fit_wrc_hcc` and to pass them as initial values to [fit\\_wrc\\_hcc](#), see [fit\\_wrc\\_hcc](#) for examples.

**Author(s)**

Andreas Papritz <papritz@retired.ethz.ch>.

**References**

- Bates, D. M., and Watts, D. G. (1988) Nonlinear Regression Analysis and its Applications. John Wiley & Sons, New York [doi:10.1002/9780470316757](https://doi.org/10.1002/9780470316757).
- Duan, Q., Sorooshian, S., and Gupta, V. K. (1994) Optimal use of the SCE-UA global optimisation method for calibrating watershed models, *Journal of Hydrology* **158**, 265–284, [doi:10.1016/0022-1694\(94\)900574](https://doi.org/10.1016/0022-1694(94)900574).
- Johnson, S.G. The NLOpt nonlinear-optimisation package. <https://github.com/stevengj/nlopt>.
- Lehmann, P., Assouline, S., Or, D. (2008) Characteristic lengths affecting evaporative drying of porous media. *Physical Review E*, **77**, 056309, [doi:10.1103/PhysRevE.77.056309](https://doi.org/10.1103/PhysRevE.77.056309).
- Lehmann, P., Bickel, S., Wei, Z., Or, D. (2020) Physical Constraints for Improved Soil Hydraulic Parameter Estimation by Pedotransfer Functions. *Water Resources Research* **56**, e2019WR025963, [doi:10.1029/2019WR025963](https://doi.org/10.1029/2019WR025963).
- Stewart, W.E., Caracotsios, M. Sørensen, J.P. 1992. Parameter estimation from multiresponse data. *AIChE Journal*, **38**, 641–650, [doi:10.1002/aic.690380502](https://doi.org/10.1002/aic.690380502).
- Stewart, W.E. and Sørensen, J.P. (1981) Bayesian estimation of common parameters from multiresponse data with missing observations. *Technometrics*, **23**, 131–141, [doi:10.1080/00401706.1981.10486255](https://doi.org/10.1080/00401706.1981.10486255).
- Van Genuchten, M. Th. (1980) A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, **44**, 892–898, [doi:10.2136/sssaj1980.03615995004400050002x](https://doi.org/10.2136/sssaj1980.03615995004400050002x).

**See Also**

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control [fit\\_wrc\\_hcc](#);

[soilhypfitmethods](#) for common S3 methods for class [fit\\_wrc\\_hcc](#);

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

---

control_fit_wrc_hcc	<i>Controlling fit_wrc_hcc</i>
---------------------	--------------------------------

---

## Description

This page documents options to control [fit\\_wrc\\_hcc](#). It describes the arguments of the functions `control_fit_wrc_hcc`, `param_boundf`, `param_transf`, `fwd_transf`, `dfwd_transf`, `bwd_transf`, `control_nloptr`, `control_sce` and `control_pcmp`, which all serve to steer [fit\\_wrc\\_hcc](#).

## Usage

```
control_fit_wrc_hcc(
  settings = c("uglobal", "ulocal", "clocal", "cglobal", "sce"),
  method = c("ml", "mpd", "wls"), hessian,
  nloptr = control_nloptr(), sce = control_sce(),
  wrc_model = "vg", hcc_model = "vgm",
  initial_param = c(alpha = 2., n = 1.5, tau = 0.5),
  approximation_alpha_k0 =
    c(c0 = 1, c1 = 5.55, c2 = 1.204, c3 = 2.11, c4 = 1.71),
  variable_weight = c(wrc = 1, hcc = 1),
  gam_k = 6, gam_n_newdata = 101, precBits = 256,
  min_nobs_wc = 5, min_nobs_hc = 5,
  keep_empty_fits = FALSE,
  param_bound = param_boundf(), param_tf = param_transf(),
  fwd_tf = fwd_transf(), deriv_fwd_tfd = dfwd_transf(), bwd_tf = bwd_transf(),
  pcmp = control_pcmp())

param_boundf(alpha = c(0 + 10 * sqrt(.Machine$double.eps), 500),
  n = c(1 + 10 * sqrt(.Machine$double.eps), 20), tau = c(-2, 20),
  thetar = c(0, 1), thetas = c(0, 1), k0 = c(0, Inf))

param_transf(alpha = c("log", "identity"), n = c("log1", "identity"),
  tau = c("logitlu", "identity"), k0 = c("log", "identity"))

fwd_transf(...)

dfwd_transf(...)

bwd_transf(...)

control_nloptr(...)

control_sce(reltol = 1.e-8, maxtime = 20, ...)

control_pcmp(ncores = detectCores() - 1L,
  fork = !identical(.Platform[["OS.type"]], "windows"))
```

## Arguments

settings	a keyword with possible values "uglobal" (default), "ulocal", "clocal", "cglobal" or "sce" to choose the approach for the nonlinear optimisation, see <i>Details</i> and <a href="#">soilhyppfitIntro</a> .
method	a keyword with possible values "ml" (maximum likelihood, default), "mpd" (maximum posterior density) or "wls" (weighted least squares) to choose the method for estimating the nonlinear parameters $\nu$ , see <a href="#">soilhyppfitIntro</a> .
hessian	a logical scalar controlling whether the Hessian matrix of the objective function should be computed with respect to the possibly transformed nonlinear parameters $\nu$ at the solution (default: TRUE if settings %in% c("uglobal", "ulocal", "sce") && method %in% c("mpd", "ml") and FALSE otherwise).
nloptr	a list of arguments passed to the optimiser <a href="#">nloptr</a> by its argument opts, or a function such as control_nloptr that generates such a list. Note that control_fit_wrc_hcc chooses sensible default values for the various components of the list in dependence of the value chosen for settings, but these defaults can be overridden by the arguments of control_nloptr, see <i>Details</i> .
sce	a list of arguments passed to the optimiser <a href="#">SCEoptim</a> by its argument control, or a function such as control_sce that generates such a list. Note that control_fit_wrc_hcc chooses sensible default values for the various components of the list, but these defaults can be overridden by the arguments of control_sce, see <i>Details</i> .
wrc_model	a keyword choosing the parametrical model for the water retention curve. Currently, only the <i>Van Genuchten</i> model ("vg") is implemented, see <a href="#">wc_model</a> and <a href="#">soilhyppfitIntro</a> .
hcc_model	a keyword choosing the parametrical model for the hydraulic conductivity function. Currently, only the <i>Van Genuchten-Mualem</i> model ("vgm") is implemented, see <a href="#">hc_model</a> and <a href="#">soilhyppfitIntro</a> .
initial_param	a named numeric vector with default initial values for the <i>nonlinear</i> parameters $\nu$ of the models for the water retention curve and/or hydraulic conductivity function. Currently, initial values must be defined for the parameters $\alpha$ (default 1.5 [ $\text{m}^{-1}$ ]), $n$ (default 2 [-]) and $\tau$ (default 0.5 [-]), hence the elements of initial_param must be named "alpha", "n" and "tau".
approximation_alpha_k0	a named numeric vector with constants to approximate the parameter $\alpha$ and the saturated hydraulic conductivity $K_0$ when constraining the estimated <i>nonlinear</i> parameters $\nu$ of the Van Genuchten-Mualem model by the <i>characteristic evaporative length</i> (Lehmann et al., 2020), see <a href="#">evaporative-length</a> and <a href="#">soilhyppfitIntro</a> . For consistency with other quantities, the following units should be used for the constants: <ul style="list-style-type: none"> <li>• c1: <math>\text{m}^{-1}</math>,</li> <li>• c3: <math>\text{m d}^{-1}</math>.</li> </ul> The remaining constants are dimensionless.
variable_weight	a named numeric vector of length 2 that defines the weights of water content and hydraulic conductivity measurements in the objective function for method



	= "wls". If equal to 1 (default) the weights of the variables are equal to the inverse variances of the water content and (possibly log-transformed) hydraulic conductivity measurements. If different from 1, then the inverse variances are multiplied by variable_weight to get the variable weights $w_\theta$ and $w_K$ , see <a href="#">fit_wrc_hcc</a> and <a href="#">soilhypfitIntro</a> . Note that for estimation methods mpd and ml the variable weights are equal to 1 but the case weights $w'_{\theta,i}$ and $w'_{K,j}$ may differ from 1.
gam_k	the dimension of the basis of the additive model for the water retention data when computing initial values of the parameters $\alpha$ and $n$ , see <a href="#">s</a> and <a href="#">soilhypfitIntro</a> .
gam_n_newdata	the number of evaluation points of the additive model for the water retention data when computing initial values of the parameters $\alpha$ and $n$ , see <a href="#">soilhypfitIntro</a> .
precBits	an integer scalar defining the default precision (in bits) to be used in high-precision computations by <a href="#">mpfr</a> , see <i>Details</i> .
min_nobs_wc	an integer scalar defining the minimum number of water content measurements per sample required for fitting a model to the water content data.
min_nobs_hc	an integer scalar defining the minimum number of hydraulic conductivity measurements per sample required for fitting a model to hydraulic conductivity data.
keep_empty_fits	a logical scalar controlling whether missing fitting results should be dropped for samples without any data or failed fits (FALSE, default) or kept (TRUE).
param_bound	a named list of numeric vectors of length 2 that define the allowed lower and upper bounds (box constraints) for the parameters of the models or a function such as <a href="#">param_boundf</a> which generates this list, see <i>Details</i> .
param_tf	a named vector of keywords that define the transformations to be applied to the model parameters before estimation or a function such as <a href="#">param_transf</a> , which generates this vector, see <i>Details</i> .
fwd_tf	a named list of invertible functions to be used to transform model parameters or a function such as <a href="#">fwd_transf</a> , which generates this list, see <i>Details</i> .
deriv_fwd_tfd	a named list of functions corresponding to the first derivatives of the functions defined in <a href="#">fwd_tf</a> or a function such as <a href="#">dfwd_transf</a> , which generates this list, see <i>Details</i> .
bwd_tf	a named list of inverse functions corresponding the functions defined in <a href="#">fwd_tf</a> or a function such as <a href="#">bwd_transf</a> , which generates this list, see <i>Details</i> .
pcmp	a list to control parallel computations or a function such as <a href="#">control_pcmp</a> that generates this list, see <a href="#">control_pcmp</a> for allowed arguments.
alpha	either a numeric vector of length 2 defining the allowed lower and upper bounds for the nonlinear parameter $\alpha$ ( <a href="#">param_boundf</a> ), or a keyword defining the transformation to be applied to the parameter $\alpha$ before estimation ( <a href="#">param_transf</a> ), see <i>Details</i> .
n	either a numeric vector of length 2 defining the allowed lower and upper bounds for the nonlinear parameter $n$ ( <a href="#">param_boundf</a> ), or a keyword defining the transformation to be applied to the parameter $n$ before estimation ( <a href="#">param_transf</a> ), see <i>Details</i> .

tau	either a numeric vector of length 2 defining the allowed lower and upper bounds for the nonlinear parameter $\tau$ (param_boundf), or a keyword defining the transformation to be applied to the parameter $\tau$ before estimation (param_transf), see <i>Details</i> .
thetar	a numeric vector of length 2 defining the allowed lower and upper bounds for the linear parameter $\theta_r$ (param_boundf), see <i>Details</i> .
thetas	a numeric vector of length 2 defining the allowed lower and upper bounds for the linear parameter $\theta_s$ (param_boundf), see <i>Details</i> .
k0	either a numeric vector of length 2 defining the allowed lower and upper bounds for the linear parameter $K_0$ (param_boundf), or a keyword defining the transformation to be applied to the parameter $K_0$ before estimation (param_transf), see <i>Details</i> .
reltol	a numeric scalar defining (one possible) convergence criterion for the optimiser SCEoptim, see argument control of SCEoptim for details.
maxtime	a numeric scalar defining the maximum duration of optimisation in seconds by the optimiser SCEoptim, see see argument control of SCEoptim for details.
ncores	an integer defining the number of cores for parallel computations. Defaults to the number of available cores minus one. ncores = 1 suppresses parallel computations.
fork	a logical scalar controlling whether forking should be used for parallel computations (default: TRUE on Unix and MacOS and FALSE on Windows operating systems). Note that setting fork = TRUE on Windows suppresses parallel computations.
...	further arguments, such as details on parameter transformations (fwd_transf, dfwd_transf, bwd_transf) or control options passed to the optimisers nloptr and SCEoptim, see <i>Details</i> .

## Details

### Enforcing bounds on the estimated parameters:

Parameters of models for the water retention curve and the hydraulic conductivity function may vary only within certain bounds (see [param\\_boundf](#) for allowed ranges). `fit_wrc_hcc` uses two mechanisms to constrain parameter estimates to permissible ranges:

#### 1. Parameter transformations

If a local algorithm is used for nonlinear optimisation (settings = "ulocal" or settings = "clocal") and a transformation not equal to "identity" is specified in param\_tf for any of the *nonlinear* parameters  $\nu$ , then the elements of  $\nu$  are transformed by the functions given in param\_tf. The values of the transformed parameters vary then over the whole real line, and an unconstrained algorithm can be used for nonlinear optimisation.

Note that the *linear* parameters  $\theta_r$  (residual) and  $\theta_s$  (saturated water content) are never transformed and for the saturated hydraulic conductivity,  $K_0$ , only "log" (default) or "identity" can be chosen. Quadratic programming (see [solve.QP](#)) is employed to enforce the box constraints specified in the argument param\_bound for  $\theta_r$  and  $\theta_s$ . Quadratic programming is also used to enforce the positivity constraint on  $K_0$  if  $K_0$  is not log-transformed ("identity"). Otherwise, the logarithm of  $K_0$  is estimated unconstrainedly, see [soilhyppfitIntro](#) for further details.

## 2. Box constraints

If a global algorithm is used for the optimisation (settings equal to "uglobal" "cglobal" or "sce") or if "identity" transformations are specified for all elements of  $\nu$ , then an optimisation algorithm is deployed that respects the box constraints given in param\_bound. If parameters are estimated for several soil samples in a single call of `fit_wrc_hcc` and if sample-specific box constraints should be used then the lower and upper bounds of the box-constraints must be specified by the arguments lower\_param and upper\_param of the function `fit_wrc_hcc`, see explanations there.

Further note that the transformations specified by param\_tf for the nonlinear parameters  $\nu$  are ignored when a global optimisation algorithm is used.

### Parameter transformations:

The arguments param\_tf, fwd\_tf, deriv\_fwd\_tfd, bwd\_tf define how the model parameters are transformed for estimation by local optimisation algorithms (see above and `soilhypfitIntro`). The following transformations are currently available:

"log":  $\log(x)$ ,  
 "log1":  $\log(x - 1)$ ,  
 "logitlu":  $\log((x-l)/(u-x))$  with  $l$  and  $u$  the allowed lower and upper bounds for a parameter, see param\_boundf,  
 "identity": no transformation.

These are the possible values that the various arguments of the function param\_transf accept (as quoted character strings), and these are the names of the list components returned by fwd\_transf, dfwd\_transf and bwd\_transf.

Additional transformations can be implemented by:

1. Extending the function definitions by arguments like  
`fwd_tf = fwd_transf(my_fun = function(x) your transformation),`  
`deriv_fwd_tfd = dfwd_transf(my_fun = function(x) your derivative),`  
`bwd_tf = bwd_transf(my_fun = function(x) your back-transformation),`
2. Assigning to a given argument of param\_transf the name of the new function, e.g.  
`alpha = "my_fun".`

Note that the values given to the arguments of param\_transf must match the names of the functions returned by fwd\_transf, dfwd\_transf and bwd\_transf.

### High-precision numerical computations:

Estimation of  $\log(K_0)$  is somewhat delicate for large values of the shape parameter  $n$  and/or small values of  $\alpha$ . The water saturation and the relative conductivity are then close to zero for capillary pressure head exceeding  $1/\alpha$ . To avoid numerical problems caused by limited accuracy of double precision numbers, `fit_wrc_hcc` uses the function `mpfr` of the package **Rmpfr** for high-accuracy computations. The argument precBits of `control_fit_wrc_hcc` controls the accuracy. Increase its value if computation fails with a respective diagnostic message.

### Options to choose the approach for nonlinear optimisation:

The argument settings defines sets of default options to control the optimisers. The following settings are currently defined:

"uglobal": unconstrained optimisation by any of the global algorithms (named "NLOPT\_G...") of the NLOpt library.

"cglobal": constrained optimisation by the global algorithm "NLOPT\_GN\_ISRES" of NLOpt that allows for inequality constraints.

"ulocal": unconstrained optimisation by any of the local algorithms (named "NLOPT\_L..." ) of NLOpt.

"clocal": constrained optimisation by any of the local algorithms ("NLOPT\_LN\_COBYLA", "NLOPT\_LN\_AUGLAG", "NLOPT\_LD\_AUGLAG", "NLOPT\_LD\_SLSQP", "NLOPT\_LD\_MMA"), "NLOPT\_LD\_CCSAQ") of NLOpt that allow for inequality constraints.

"sce": unconstrained optimisation by the global algorithm implemented in [SCEoptim](#).

The functions `control_nloptr` and `control_sce` allow finer control of the optimisers. `control_nloptr` and `control_sce` take any argument available to pass controlling options to the optimisers [nloptr](#) (by its argument `opts`) and [SCEoptim](#) (by its argument `control`), respectively.

#### *Controlling nloptr:*

The function `nloptr.print.options` prints all options available to control [nloptr](#) by its argument `opts`. Detailed information on the options can be found in the [NLOpt documentation](#).

The function `control_fit_wrc_hcc` sets meaningful defaults for `opts` in dependence of the chosen optimisation approach as specified by the argument `settings`, and it checks the consistency of the arguments of `control_nloptr` if they are explicitly passed to `fit_wrc_hcc`.

The following defaults are set by `control_fit_wrc_hcc` for the argument `opts` of [nloptr](#) (:

1. Unconstrained, global optimisation (`settings = "uglobal"`):

```
nloptr = control_nloptr(
  algorithm = "NLOPT_GN_MLSL_LDS",
  local_opts = list(
    algorithm = "NLOPT_LN_BOBYQA",
    xtol_rel = -1.,
    ftol_rel = 1.e-6
  ),
  xtol_rel = -1,
  ftol_rel = -1,
  maxeval = 125,
  maxtime = -1)
```

In addition, any parameter transformations specified by `param_tf` are overridden and the untransformed parameters ("identity") are estimated when `settings = "uglobal"` is chosen.

2. Constrained, global optimisation (`settings = "cglobal"`):

```
nloptr = control_nloptr(
  algorithm = "NLOPT_GN_ISRES",
  xtol_rel = -1,
  ftol_rel = -1,
  maxeval = 1000,
  maxtime = -1)
```

In addition, any parameter transformations specified by `param_tf` are overridden and the untransformed parameters ("identity") are estimated when `settings = "cglobal"` is chosen.

## 3. Unconstrained, local optimisation (settings = "ulocal"):

```
nloptr = control_nloptr(
  algorithm = "NLOPT_LN_BOBYQA",
  xtol_rel = -1,
  ftol_rel = 1.e-8,
  maxeval = 250,
  maxtime = -1)
```

## 4. Constrained, local optimisation (settings = "clocal"):

```
nloptr = control_nloptr(
  algorithm = "NLOPT_LD_CCSAQ",
  xtol_rel = -1,
  ftol_rel = 1.e-8,
  maxeval = 1000,
  maxtime = -1)
```

If the algorithm "NLOPT\_LD\_AUGLAG" is used for constrained, local optimisation then

```
nloptr = control_nloptr(
  algorithm = "NLOPT_LD_AUGLAG",
  local_opts = list(
    algorithm = "NLOPT_LD_LBFGS",
    xtol_rel = -1.,
    ftol_rel = 1.e-6
  ),
  xtol_rel = -1,
  ftol_rel = 1.e-8,
  maxeval = 1000,
  maxtime = -1)
```

For other, unspecified elements of opts default values as listed by `nloptr.print.options` are used.

*Controlling SCEoptim:*

The function `control_sce` sets meaningful defaults for the argument `control` of `SCEoptim`. Currently, the following defaults are defined:

```
sce = control_sce(
  reltol = 1e-08,
  maxtime = 20)
```

In addition, any parameter transformations specified by `param_tf` are overridden and the untransformed parameters ("identity") are estimated when settings = "sce" is chosen.

**Value**

`control_fit_wrc_hcc` creates a list with components `settings`, `hessian`, `method`, `nloptr`, `sce`, `wrc_model`, `hcc_model`, `initial_param`, `approximation_alpha_k0`, `variable_weight`, `gam_k`, `gam_n_newdata`, `precBits`, `min_nobs_wc`, `min_nobs_hc`, `keep_empty_fits`, `param_bound`, `param_tf`,

fwd\_tf, deriv\_fwd\_tfd, bwd\_tf, pcmp corresponding to its arguments and some further components (delta\_sat\_0, grad\_eps, use\_derivative) that cannot be changed by the user.

control\_nloptr and control\_sce create lists with control parameters passed to [nloptr](#) and [SCEoptim](#), respectively, see *Details*.

param\_boundf generates a list with allowed lower and upper bounds of the model parameters.

param\_transf generates a list with keywords that define what transformations are used for estimating the model parameters, and fwd\_transf, bwd\_transf and dfwd\_transf return lists of functions with forward and backward transformations and the first derivatives of the forward transformations, see *Details*.

control\_pcmp generates a list with control parameters for parallel computations.

### Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

### References

- Johnson, S.G. The NLOpt nonlinear-optimisation package. <https://github.com/stevengj/nlopt>.
- Lehmann, P., Assouline, S., Or, D. (2008) Characteristic lengths affecting evaporative drying of porous media. *Physical Review E*, **77**, 056309, [doi:10.1103/PhysRevE.77.056309](https://doi.org/10.1103/PhysRevE.77.056309).
- Lehmann, P., Bickel, S., Wei, Z., Or, D. (2020) Physical Constraints for Improved Soil Hydraulic Parameter Estimation by Pedotransfer Functions. *Water Resources Research* **56**, e2019WR025963, [doi:10.1029/2019WR025963](https://doi.org/10.1029/2019WR025963).

### See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

### Examples

```
# use of \donttest{} because execution time exceeds 5 seconds
data(sim_wrc_hcc)

# estimate parameters for a single soil sample by maximizing loglikelihood ...

# ... with unconstrained, global optimisation algorithm NLOPT_GN_MLSL
```

```

coef(
  fit1 <- fit_wrc_hcc(
    wrc_formula = wc ~ head, hcc_formula = hc ~ head,
    data = sim_wrc_hcc, subset = id == 2
  ), gof = TRUE)

# ... as fit1 but fitting parameter tau as well
coef(
  fit2 <- update(fit1,
    fit_param = default_fit_param(tau = TRUE)
  ), gof = TRUE)

plot(fit1, y = fit2)

# ... with unconstrained, local optimisation algorithm NLOPT_LN_BOBYQA,
#   initial values for alpha and n are computed from data and
#   transformed nonlinear parameters are estimated without box-constraints
coef(
  fit3 <- update(
    fit2,
    control = control_fit_wrc_hcc(settings = "ulocal"),
    verbose = 2), gof = TRUE)

# estimate parameters by unconstrained, weighted least squares minimisation with
#   algorithm NLOPT_LD_LBFGS, giving larger weight to conductivity data,
#   using specified initial values for alpha and n and
#   fitting untransformed nonlinear parameters with default box constraints
#   defined by param_boundf()
#   diagnostic output directly from nloptr
coef(
  fit4 <- update(
    fit2,
    param = c(alpha = 1.7, n = 2),
    control = control_fit_wrc_hcc(
      settings = "ulocal", method = "wls",
      variable_weight = c(wrc = 1, hcc = 2),
      nloptr = control_nloptr(algorithm = "NLOPT_LD_LBFGS", print_level = 3),
      param_tf = param_transf(alpha = "identity", n = "identity", tau = "identity")
    ), verbose = 0), gof = TRUE)

# ... as fit4 but giving even larger weight to conductivity data
coef(
  fit5 <- update(
    fit4,
    control = control_fit_wrc_hcc(
      settings = "ulocal", method = "wls",
      variable_weight = c(wrc = 1, hcc = 5),
      nloptr = control_nloptr(algorithm = "NLOPT_LD_LBFGS", print_level = 3),
      param_tf = param_transf(alpha = "identity", n = "identity", tau = "identity")
    ), verbose = 0), gof = TRUE)

plot(fit4, y = fit5)

```

---

evaporative-length	<i>Evaporative Characteristic Length</i>
--------------------	--

---

## Description

The functions `lc` and `lt` compute the *characteristic length*  $L_c$  of *stage-I* evaporation from a soil and its “target” (expected) value  $L_t$ , used to constrain estimates of nonlinear parameters of the Van Genuchten-Mualem (VGM) model for water retention curves and hydraulic conductivity functions.

## Usage

```
lc(alpha, n, tau, k0, e0, c0 = NULL, c3 = NULL, c4 = NULL)
```

```
lt(n, tau, e0, c0, c1, c2, c3, c4)
```

## Arguments

- |                    |  |
|--------------------|--|
| alpha              | parameter $\alpha$ (inverse air entry pressure) of the VGM model, see <a href="#">wc_model</a> and <a href="#">hc_model</a> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [ $\text{m}^{-1}$ ].  |
| n                  | parameter $n$ (shape parameter) of the VGM model, see <a href="#">wc_model</a> and <a href="#">hc_model</a> .  |
| tau                | parameter $\tau$ (tortuosity parameter) of the VGM model, see <a href="#">hc_model</a> .   |
| k0                 | saturated hydraulic conductivity $K_0$ , see <a href="#">hc_model</a> . If <code>k0</code> is missing or equal to NA in calls of <code>lc</code> then <code>k0</code> is approximated by the same relation as used for <code>lt</code> , see <i>Details</i> . For consistency with other quantities, the unit of $K_0$ should be <b>meter/day</b> [ $\text{m d}^{-1}$ ].   |
| e0                 | a numeric scalar with the <i>stage-I</i> rate of evaporation $E_0$ from a soil, see <i>Details</i> and <a href="#">soilhypfitIntro</a> . For consistency with other quantities, the unit of $E_0$ should be <b>meter/day</b> [ $\text{m d}^{-1}$ ].  |
| c0, c1, c2, c3, c4 | numeric constants to approximate the parameter $\alpha$ and the saturated hydraulic conductivity $K_0$ when computing $L_t$ , see <i>Details</i> and <a href="#">control_fit_wrc_hcc</a> . For consistency with other quantities, the following units should be used for the constants: <ul style="list-style-type: none"> <li>• c1: <math>\text{m}^{-1}</math>,</li> <li>• c3: <math>\text{m d}^{-1}</math>.</li> </ul> |

The remaining constants are dimensionless.

## Details

The *characteristic length* of *stage-I* evaporation  $L_c$  (Lehmann et al., 2008, 2020) is defined by

$$L_c(\nu, K_0, E_0) = \frac{\frac{1}{\alpha n} \left( \frac{2n-1}{n-1} \right)^{\frac{2n-1}{n}}}{1 + \frac{E_0}{K_{\text{eff}}}}$$



where  $\boldsymbol{\nu}^T = (\alpha, n, \tau)$  are the nonlinear parameters of the VGM model,  $K_0$  is the saturated hydraulic conductivity,  $E_0$  the stage-I evaporation rate and  $K_{\text{eff}} = 4 K_{\text{VGM}}(h_{\text{crit}}; K_0, \boldsymbol{\nu})$  is the effective hydraulic conductivity at the critical pressure

$$h_{\text{crit}} = \frac{1}{\alpha} \left( \frac{n-1}{n} \right)^{\frac{1-2n}{n}},$$

see `hc_model` for the definition of  $K_{\text{VGM}}(h; K_0, \boldsymbol{\nu})$ .

The quantity  $L_t$  is the expected value (“target”) of  $L_c$  for given shape ( $n$ ) and tortuosity ( $\tau$ ) parameters. To evaluate  $L_t$ , the parameters  $\alpha$  and  $K_0$  are approximated by the following relations

$$\hat{\alpha} = g_{\alpha}(n; c_0, c_1, c_2) = c_1 \frac{n - c_0}{1 + c_2(n - c_0)},$$

$$\hat{K}_0 = g_{K_0}(n; c_0, c_3, c_4) = c_3 (n - c_0)^{c_4}.$$

The default values for  $c_0$  to  $c_4$  (see argument `approximation_alpha_k0` of `control_fit_wrc_hcc`) were estimated with data on African desert regions of the database *ROSETTA3* (Zhang and Schaap, 2017), see Lehmann et al. (2020) for details.

## Value

A numeric scalar with the characteristic evaporative length (`lc`) or its expected value (`lt`).

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## References

- Lehmann, P., Assouline, S., Or, D. (2008) Characteristic lengths affecting evaporative drying of porous media. *Physical Review E*, **77**, 056309, doi:10.1103/PhysRevE.77.056309.
- Lehmann, P., Bickel, S., Wei, Z., Or, D. (2020) Physical Constraints for Improved Soil Hydraulic Parameter Estimation by Pedotransfer Functions. *Water Resources Research* **56**, e2019WR025963, doi:10.1029/2019WR025963.
- Zhang, Y. , Schaap, M. G. 2017. Weighted recalibration of the Rosetta pedotransfer model with improved estimates of hydraulic parameter distributions and summary statistics (Rosetta3). *Journal of Hydrology*, **547**, 39-53, doi:10.1016/j.jhydrol.2017.01.004.

## See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

`prfloglik_sample` for profile loglikelihood computations;  
`wc_model` and `hc_model` for currently implemented models for soil water retention curves and hydraulic conductivity functions;

## Examples

```
# use of \donttest{} because execution time exceeds 5 seconds

# estimate parameters of 4 samples of the Swiss forest soil dataset
# that have water retention (theta, all samples), saturated hydraulic conductivity
# (ksat) and optionally unsaturated hydraulic conductivity data
# (ku, samples "CH2_4" and "CH3_1")

data(swissforestsoils)

# select subset of data
sfs_subset <- droplevels(
  subset(
    swissforestsoils,
    layer_id %in% c("CH2_3", "CH2_4", "CH2_6", "CH3_1")
  ))

# extract ksat measurements
ksat <- sfs_subset[!duplicated(sfs_subset$layer_id), "ksat", drop = FALSE]
rownames(ksat) <- levels(sfs_subset$layer_id)
colnames(ksat) <- "k0"

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L

# unconstrained estimation (global optimisation algorithm NLOPT_GN_MLSL)
# k0 fixed at measured ksat values
rsfs_uglob <- fit_wrc_hcc(
  wrf_formula = theta ~ head | layer_id,
  hcc_formula = ku ~ head | layer_id,
  data = sfs_subset,
  param = ksat,
  fit_param = default_fit_param(k0 = FALSE),
  control = control_fit_wrc_hcc(
    settings = "uglobal", pcmp = control_pcmp(ncores = ncpu)))
summary(rsfs_uglob)
coef(rsfs_uglob, lc = TRUE, gof = TRUE)

# constrained estimation by restricting ratio Lc/Lt to [0.5, 2]
# (global constrained optimisation algorithm NLOPT_GN_MLSL)
# k0 fixed at measured ksat values
rsfs_cglob <- update(
  rsfs_uglob,
  control = control_fit_wrc_hcc(
    settings = "cglobal", nloptr = control_nloptr(ranseed = 1),
```

```

    pcmp = control_pcmp(ncores = ncpu)))
summary(rsfs_cglob)
coef(rsfs_cglob, lc = TRUE, gof = TRUE)

# get initial parameter values from rsfs_cglob
ini_param <- cbind(
  coef(rsfs_cglob)[, c("alpha", "n")],
  ksat
)
# constrained estimation by restricting ratio Lc/Lt to [0.5, 2]
# (local constrained optimisation algorithm NLOPT_LD_CCSAQ)
# k0 fixed at measured ksat values
rsfs_cloc <- update(
  rsfs_uglob,
  param = ini_param,
  control = control_fit_wrc_hcc(
    settings = "clocal", nloptr = control_nloptr(ranseed = 1),
    pcmp = control_pcmp(ncores = ncpu)))
summary(rsfs_cloc)
coef(rsfs_cloc, lc = TRUE, gof = TRUE)

op <- par(mfrow = c(4, 2))
plot(rsfs_uglob, y = rsfs_cglob)
on.exit(par(op))

op <- par(mfrow = c(4, 2))
plot(rsfs_uglob, y = rsfs_cloc)
on.exit(par(op))

```

---

fit\_wrc\_hcc

---

*Parametric Modelling of Soil Hydraulic Properties*


---

## Description

The function `fit_wrc_hcc` estimates parameters of models for the soil water retention curve and/or soil hydraulic conductivity function from respective measurements by nonlinear regression methods, optionally subject to physical constraints on the estimated parameters. `fit_wrc_hcc` uses optimisation algorithms of the `NLOpt` library (Johnson, see [nloptr-package](#)) and the Shuffled Complex Evolution (SCE) algorithm (Duan et al., 1994) implemented in the function [SCEoptim](#).

## Usage

```

fit_wrc_hcc(
  wrc_formula, hcc_formula, data,
  subset = NULL, wrc_subset = subset, hcc_subset = subset,
  weights = NULL, wrc_weights = weights, hcc_weights = weights,
  na.action, param = NULL, lower_param = NULL, upper_param = NULL,
  fit_param = default_fit_param(),

```

```
e0 = 2.5e-3, ratio_lc_lt_bound = c(lower = 0.5, upper = 2),
control = control_fit_wrc_hcc(), verbose = 0)

default_fit_param(
  alpha = TRUE, n = TRUE, tau = FALSE,
  thetar = TRUE, thetas = TRUE, k0 = TRUE)
```

## Arguments

wrc_formula	an optional two-sided formula such as $wc \sim head$ or $wc \sim head \mid id$ , specifying the variables for the water content (wc), the capillary pressure head and, optionally, for sample ids when model parameters are estimated for several soil samples at the same time, see <a href="#">formula</a> and <i>Details</i> .
hcc_formula	an optional two-sided formula such as $hc \sim head$ or $hc \sim head \mid id$ , specifying the variables for the hydraulic conductivity (hc), the capillary pressure head and, optionally, for sample ids when model parameters are estimated for several soil samples at the same time. See <a href="#">formula</a> and <i>Details</i> .
data	a mandatory data frame containing the variables specified in the formula, the subset and weights arguments.
subset	an optional expression generating a vector to choose a subset of water content and hydraulic conductivity data. The expression is evaluated in the environment generated by <code>model.frame(wrc_formula, data)</code> and <code>model.frame(hcc_formula, data)</code> , respectively.
wrc_subset	an optional expression generating a vector to choose a subset of water content data. The expression is evaluated in the environment generated by <code>model.frame(wrc_formula, data)</code> . Defaults to subset.
hcc_subset	an optional expression generating a vector to choose a subset of hydraulic conductivity data. The expression is evaluated in the environment generated by <code>model.frame(hcc_formula, data)</code> . Defaults to subset.
weights	an optional expression generating a numeric vector of case weights $w'_{\theta,i}$ and $w'_{K,i}$ (default: 1, see <a href="#">soilhypfitIntro</a> ) for water content and hydraulic conductivity data. The expression is evaluated in the environment generated by <code>model.frame(wrc_formula, data)</code> and <code>model.frame(hcc_formula, data)</code> , respectively.
wrc_weights	an optional expression generating a numeric vector of case weights $w'_{\theta,i}$ (see <a href="#">soilhypfitIntro</a> ) for water content data. The expression is evaluated in the environment generated by <code>model.frame(wrc_formula, data)</code> . Defaults to weights.
hcc_weights	an optional expression generating a numeric vector of case weights $w'_{K,i}$ (see <a href="#">soilhypfitIntro</a> ) for hydraulic conductivity data. The expression is evaluated in the environment generated by <code>model.frame(hcc_formula, data)</code> . Defaults to weights.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action argument of <a href="#">options</a> , and is <code>na.fail</code> if that is unset. The “factory-fresh” default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.

param	an optional named numeric vector (or a numeric matrix or a dataframe with specified row and column names, see <i>Details</i> ) with initial values for the model parameters. Currently, param may have elements (or columns) named "alpha", "n", "tau", "thetar", "thetas", "k0", see <a href="#">wc_model</a> and <a href="#">hc_model</a> and <i>Details</i> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [ $\text{m}^{-1}$ ] and the unit of $K_0$ <b>meter/day</b> [ $\text{m d}^{-1}$ ].
lower_param	an optional named numeric vector (or a numeric matrix or a dataframe with specified row and column names, see <i>Details</i> ) with lower bounds for the parameters of the models. Currently, lower_param may have elements (or columns) named "alpha", "n", "tau", "thetar", "thetas", "k0", see <a href="#">wc_model</a> , <a href="#">hc_model</a> and <a href="#">param_boundf</a> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [ $\text{m}^{-1}$ ] and the unit of $K_0$ <b>meter/day</b> [ $\text{m d}^{-1}$ ]. If lower bounds are specified for $\theta_r$ but not for $\theta_s$ then the lower bounds specified for $\theta_r$ will also be used for $\theta_s$ .
upper_param	an optional named numeric vector (or a numeric matrix or a dataframe with specified row and column names, see <i>Details</i> ) with upper bounds for the parameters of the models. Currently, upper_param may have elements (or columns) named "alpha", "n", "tau", "thetar", "thetas", "k0", see <a href="#">wc_model</a> , <a href="#">hc_model</a> and <a href="#">param_boundf</a> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [ $\text{m}^{-1}$ ] and the unit of $K_0$ <b>meter/day</b> [ $\text{m d}^{-1}$ ]. If upper bounds are specified for $\theta_s$ but not for $\theta_r$ then the upper bounds specified for $\theta_s$ will also be used for $\theta_r$ .
fit_param	a named logical vector (or a logical matrix or a dataframe with specified row and column names, see <i>Details</i> ) containing flags that control whether model parameters are estimated (TRUE) or kept fixed at the initial values (FALSE) as specified in param. This vector can be generated easily by the function <code>default_fit_param</code> . Currently, fit_param may have elements (or columns) named "alpha", "n", "tau", "thetar", "thetas", "k0", see <i>Details</i> , <a href="#">wc_model</a> and <a href="#">hc_model</a> .
e0	a numeric scalar (or named vector, see <i>Details</i> ) with the stage-I rate of evaporation $E_0$ from a soil (default $2.5 \cdot 10^{-3} \text{ m d}^{-1}$ ) required to evaluate the <i>characteristic evaporative length</i> , see <a href="#">evaporative-length</a> and <a href="#">soilhyfitIntro</a> . For consistency with other quantities, the unit of $E_0$ should <b>meter/day</b> [ $\text{m d}^{-1}$ ]. Note that e0 is ignored when an unconstrained nonlinear optimisation algorithm is chosen (argument settings of <a href="#">control_fit_wrc_hcc</a> equal to "uglobal", "sce" or "ulocal").
ratio_lc_lt_bound	a named numeric vector of length 2 (or a matrix with specified rownames and two columns, see <i>Details</i> ) defining the default lower and upper bounds of the ratio $L_c/L_t$ ( <a href="#">Lehmann et al., 2008, 2020</a> ) for constrained estimation of the <i>non-linear</i> parameters $\nu^T = (\alpha, n, \tau)$ (default values 0.5 (lower) and 2 (upper)), see <a href="#">evaporative-length</a> and <a href="#">soilhyfitIntro</a> . Note that ratio_lc_lt_bound is ignored when an unconstrained nonlinear optimisation algorithm is chosen (argument settings of <a href="#">control_fit_wrc_hcc</a> equal to "uglobal", "sce" or "ulocal").
control	a list with options to control fit_wrc_hcc or a function such as <a href="#">control_fit_wrc_hcc</a> that generates such a list. The main argument settings

	of <code>control_fit_wrc_hcc</code> selects a nonlinear optimisation approach, see <a href="#">soilhyfitIntro</a> and <a href="#">control_fit_wrc_hcc</a> for full details.
verbose	<p>positive integer controlling logging of diagnostic messages to the console and plotting of data and model curves during fitting.</p> <p>verbose &lt; 0 suppresses all output,</p> <p>verbose &gt;= 0 suppresses all output except warning messages,</p> <p>verbose &gt;= 1 displays at the end the measurements along with the fitted model curves,</p> <p>verbose &gt;= 2 prints for each iteration the parameters values, the value of the objective function and the ratio of <math>L_c/L_t</math> (see <a href="#">evaporative-length</a>),</p> <p>verbose &gt;= 3 plots for each iteration the measurements along with the fitted model curves.</p> <p>Logging of further diagnostics during fitting is controlled in addition by the arguments <code>print_level</code>, <code>check_derivatives</code>, <code>check_derivatives_print</code> when <code>nloptr</code> is used (see <a href="#">nloptr.print.options</a> and <a href="#">control_nloptr</a>) and by the argument <code>trace</code> of <code>SCEoptim</code> (see <a href="#">control_sce</a>).</p>
alpha	a logical scalar controlling whether the inverse air entry pressure $\alpha$ should be estimated (TRUE, default) or kept fixed at the initial value (FALSE) specified by <code>param</code> , see <a href="#">wc_model</a> and <a href="#">hc_model</a> .
n	a logical scalar controlling whether the shape parameter $n$ should be estimated (TRUE, default) or kept fixed at the initial value (FALSE) specified by <code>param</code> , see <a href="#">wc_model</a> and <a href="#">hc_model</a> .
tau	a logical scalar controlling whether the tortuosity parameter $\tau$ should be estimated (TRUE) or kept fixed at the initial value (FALSE, default) specified by <code>param</code> , see <a href="#">hc_model</a> .
thetar	a logical scalar controlling whether the residual water content $\theta_r$ should be estimated (TRUE, default) or kept fixed at the initial value (FALSE) specified by <code>param</code> , see <a href="#">wc_model</a> .
thetas	a logical scalar controlling whether the saturated water content $\theta_s$ should be estimated (TRUE, default) or kept fixed at the initial value (FALSE) specified by <code>param</code> , see <a href="#">wc_model</a> .
k0	a logical scalar controlling whether the saturated hydraulic conductivity $K_0$ should be estimated (TRUE, default) or kept fixed at the initial value (FALSE) specified by <code>param</code> , see <a href="#">hc_model</a> .

## Details

### Estimating model parameters of water retention curves and/or hydraulic conductivity functions:

The function `fit_wrc_hcc` estimates model parameters of water retention curves and/or hydraulic conductivity functions by *maximum likelihood* (ml, default) *maximum posterior density* (mpd) or *nonlinear least squares* (wls), see argument `method` of [control\\_fit\\_wrc\\_hcc](#). Measurements must be available for at least one of the two material functions. If one type of data is missing then the respective formula, subset and weights arguments should be omitted.

If both types of data are available then measurements are weighed when computing the residual sum of squares for wls, but unit weights are used by default for mpd and ml estimation, see [soilhypfitIntro](#). The wls weights are the product of the *case weights*  $w'_{\theta,i}$  and  $w'_{K,i}$  given by `weights`, `wrc_weights` and `hcc_weights`, respectively, and the *variable weights* as specified by the argument `variable_weight` of `control_fit_wrc_hcc`. Note that the `variable_weights` are not used “as is”, but they are multiplied by the inverse variances of the water content or the (log-transformed) conductivity data per sample to obtain the variable weights  $w_{\theta}$ ,  $w_K$  mentioned in [soilhypfitIntro](#).

### Estimating model parameters for a single or multiple soil samples:

When parameters are estimated for a single soil sample only, then model formulae are of the form `wc ~ head` and/or `hc ~ head`, and there is no need to specify a sample id. In this case `param`, `lower_param`, `upper_param`, `fit_param` and `ratio_lc_lt_bound` are vectors and `e0` is a scalar. `fit_wrc_hcc` allows to fit models to datasets containing data for multiple soil samples. The model formula must then be of the form `wc ~ head|id` and/or `hc ~ head|id` where the factor `id` identifies the soil samples. If `param`, `lower_param`, `upper_param`, `fit_param` and `ratio_lc_lt_bound` are vectors and `e0` is a scalar then this information is used for processing all the soil samples. If specific information per sample should be used then `param`, `lower_param`, `upper_param`, `fit_param` and `ratio_lc_lt_bound` must be matrices (or data frames) with as many rows as there are soil samples. The matrices (or data.frames) must have rownames matching the levels of the factor `id` defining the soil samples. `e0` must be a named vector with as many elements as there are soil samples and the names must again match the levels of `id`.

By default, `fit_wrc_hcc` processes data of multiple soil samples in parallel, see [control\\_pcmp](#) for options to control parallel computing. Note that diagnostic output (except warning messages) is suppressed in parallel computations. If computations fail for a particular soil sample, the id of the sample with the failed fit can be extracted by the utility function [select\\_failed\\_fits](#) and the respective error messages by [extract\\_error\\_messages](#).

### Controlling fit\_wrc\_hcc:

The argument `control` is used to pass a list of options to `fit_wrc_hcc` that steer the function, see [soilhypfit-package](#) and [control\\_fit\\_wrc\\_hcc](#) for full details.

## Value

`fit_wrc_hcc` returns an object of [class](#) `fit_wrc_hcc`, which is a list with the following components:

- |                  |  |
|------------------|--|
| <code>fit</code> | <p>a list with as many components as there are soils samples. Each component is in turn a list that contains the estimated parameters and accompanying information:</p> <ul style="list-style-type: none"> <li>• <code>converged</code>: an integer code issued by <a href="#">SCEoptim</a> or <a href="#">nloptr</a> on (non-) convergence, see <a href="#">convergence_message</a> and <a href="#">NLOpt return values</a>.</li> <li>• <code>message</code>: a character string issued by <a href="#">SCEoptim</a> or <a href="#">nloptr</a> on (non-)convergence.</li> <li>• <code>evaluation</code>: the number of evaluations of the objective function and of the gradient.</li> <li>• <code>objective</code>: the value of the objective function evaluated at the solution. The contributions of the water retention curve and the hydraulic conductivity function to objective are reported as attributes <code>obj_wc</code> and <code>obj_hc</code>. The attributes <code>ssq_wc</code> and <code>ssq_hc</code> are the respective residual sums of squares <math>Q_{\theta}</math> and <math>Q_K</math>, see <a href="#">soilhypfitIntro</a>.</li> </ul> |
|------------------|--|

- **gradient**: the gradient of the objective function at the solution with respect to the (possibly transformed) nonlinear parameters.
- **lp**: the estimated values for the linear parameters  $\boldsymbol{\mu}^T = (\theta_r, \theta_s, K_0)$ , see [wc\\_model](#) and [hc\\_model](#).
- **nlp**: the estimated values for the nonlinear parameters  $\boldsymbol{\nu}^T = (\alpha, n, \tau)$ , see [wc\\_model](#) and [hc\\_model](#).
- **inequality\_constraint**: a list with the values of the inequality constraints evaluated at the solutions. Currently, values are reported for the expressions.

$$-(\frac{L_c}{L_t} - l)$$

$$\frac{L_c}{L_t} - u$$

in the only component `lc`, where  $l$  and  $u$  are the lower and upper bounds of the ratio, see argument `ratio_lc_lt_bound` and [evaporative-length](#). The values of  $L_c$ ,  $L_t$ , the imposed bounds on their ratio and `e0` are returned as attributes of `lc`.

- **hessian**: optionally the Hessian matrix of the objective function with respect to the possibly transformed nonlinear parameters at the solution.
- **variable\_weight**: a named vector with the variable weights  $w_\theta$  and  $w_K$ , see *Details*.
- **weight**: a list with one or two components named `weights_wc` and/or `weights_hc` with the case weights  $w_{\theta,i}$  and  $w_{K,i}$  used in the objective function, see *Details*.
- **fitted**: a list with one or two components named `wrc` and/or `hcc` with the fitted values for the water retention and the (possibly log-transformed) hydraulic conductivity data.
- **residuals**: a list with one or two components named `wrc` and/or `hcc` with residuals for the water retention and the (possibly log-transformed) hydraulic conductivity data.
- **model**: a list with one or two components named `wrc` and/or `hcc` with the [model.frames](#) for the water retention and hydraulic conductivity data.
- **initial\_objects**: a list with the values for `param`, `fit_param`, `variable_weight`, `param_bound`, `e0` and `ratio_lc_lt_bound` as taken from the (processed) arguments of the call of `fit_wrc_hcc`.

`sample_id_variable`

the name of the variable defining the samples.

`wrc` logical scalar signalling whether water retention data were used to estimate the parameters.

`wrc_formula` formula defining the variables for water content and hydraulic head of the water retention data (NULL if not `wrc` equal to FALSE).

`wrc_mf` unevaluated call of [model.frame](#) to build the modelframe for the water retention data (NULL if `wrc` equal to FALSE).

`wrc` logical scalar signalling whether water retention data were used to estimate the parameters.



hcc_formula	formula defining the variables for hydraulic conductivity and hydraulic head of the hydraulic conductivity data (NULL if not hcc equal to FALSE).
hcc_mf	unevaluated call of <code>model.frame</code> to build the model frame for the water retention data (NULL if hcc equal to FALSE).
control	a list with the options used to control <code>fit_wrc_hcc</code> , see <code>control_fit_wrc_hcc</code> .
call	the matched call.
na.action	information returned by <code>model.frame</code> on the special handling of NAs.

### Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

### References

- Duan, Q., Sorooshian, S., and Gupta, V. K. (1994) Optimal use of the SCE-UA global optimisation method for calibrating watershed models, *Journal of Hydrology* **158**, 265–284, doi:10.1016/0022-1694(94)900574.
- Johnson, S.G. The NLOpt nonlinear-optimisation package. <https://github.com/stevengj/nlopt>.
- Lehmann, P., Assouline, S., Or, D. (2008) Characteristic lengths affecting evaporative drying of porous media. *Physical Review E*, **77**, 056309, doi:10.1103/PhysRevE.77.056309.
- Lehmann, P., Bickel, S., Wei, Z., Or, D. (2020) Physical Constraints for Improved Soil Hydraulic Parameter Estimation by Pedotransfer Functions. *Water Resources Research* **56**, e2019WR025963, doi:10.1029/2019WR025963.

### See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

`vcov` for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

### Examples

```
# use of \donttest{} because execution time exceeds 5 seconds

data(sim_wrc_hcc)

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L
```

```

# estimate parameters for single sample ...

# ... from wrc and hcc data
plot(rfit_wrc_hcc <- fit_wrc_hcc(
  wrc_formula = wc ~ head, hcc_formula = hc ~ head,
  data = sim_wrc_hcc, subset = id == 1))
coef(rfit_wrc_hcc, gof = TRUE)

# ... from wrc data
plot(rfit_wrc <- fit_wrc_hcc(
  wrc_formula = wc ~ head,
  data = sim_wrc_hcc, subset = id == 1))
coef(rfit_wrc, gof = TRUE)

# ... from hcc data
plot(rfit_hcc <- fit_wrc_hcc(
  hcc_formula = hc ~ head,
  data = sim_wrc_hcc, subset = id == 1))
coef(rfit_hcc, gof = TRUE)

# ... from wrc and hcc data
# keeping some parameters fixed
plot(rfit_wrc_hcc_fixed <- fit_wrc_hcc(
  wrc_formula = wc ~ head, hcc_formula = hc ~ head,
  data = sim_wrc_hcc, subset = id == 1,
  param = c(alpha = 2.1, thetar = 0.1),
  fit_param = default_fit_param(alpha = FALSE, thetar = FALSE)),
  y = rfit_wrc_hcc)
coef(rfit_wrc_hcc, gof = TRUE)
coef(rfit_wrc_hcc_fixed, gof = TRUE)

# estimate parameters for 3 samples simultaneously by ...

# ... unconstrained, global optimisation algorithm NLOPT_GN_MLSL (default)
rfit_uglob <- fit_wrc_hcc(
  wrc_formula = wc ~ head | id, hcc_formula = hc ~ head | id,
  data = sim_wrc_hcc,
  control = control_fit_wrc_hcc(pcmp = control_pcmp(ncores = ncpu)))
summary(rfit_uglob)
op <- par(mfrow = c(3, 2))
plot(rfit_uglob)
on.exit(par(op))

# ... unconstrained, global optimisation algorithm SCEoptim
rfit_sce <- update(
  rfit_uglob,
  control = control_fit_wrc_hcc(
    settings = "sce", pcmp = control_pcmp(ncores = ncpu)))
coef(rfit_sce, gof = TRUE, lc = TRUE)
convergence_message(2, sce = TRUE)
op <- par(mfrow = c(3, 2))

```

```

plot(rfit_sce, y = rfit_uglob)
on.exit(par(op))

# ... unconstrained, local optimisation algorithm NLOPT_LN_BOBYQA,
#   logging iteration results to console
rfit_uloc <- update(
  rfit_uglob,
  param = as.matrix(coef(rfit_uglob, what = "nonlinear")),
  control = control_fit_wrc_hcc(
    settings = "ulocal", pcmp = control_pcmp(ncores = 1L)),
  verbose = 2)
coef(rfit_uloc, gof = TRUE, lc = TRUE)

# ... constrained, global optimisation algorithm NLOPT_GN_ISRES
rfit_cglob <- update(
  rfit_uglob,
  ratio_lc_lt_bound = c(lower = 0.8, upper = 1.2),
  control = control_fit_wrc_hcc(
    settings = "cglobal", nloptr = control_nloptr(ranseed = 1),
    pcmp = control_pcmp(ncores = ncpu)))
coef(rfit_cglob, gof = TRUE, lc = TRUE)

# ... constrained, local optimisation algorithm NLOPT_LD_CCSAQ
#   starting from unconstrained, locally fitted initial values
rfit_cloc_1 <- update(
  rfit_uglob,
  param = coef(rfit_uloc, what = "nonlinear"),
  ratio_lc_lt_bound = c(lower = 0.8, upper = 1.2),
  control = control_fit_wrc_hcc(
    settings = "clocal", pcmp = control_pcmp(ncores = ncpu)))
coef(rfit_cloc_1, gof = TRUE, lc = TRUE)
op <- par(mfrow = c(3, 2))
plot(x = rfit_uloc, y = rfit_cloc_1)
on.exit(par(op))

# ... constrained, local optimisation algorithm NLOPT_LD_CCSAQ
#   starting from constrained, globally fitted initial values,
#   using sample-specific evaporation rates and limits for ratio lc/lt
rfit_cloc_2 <- update(
  rfit_uglob,
  param = as.matrix(coef(rfit_cglob, what = "nonlinear")),
  e0 = c("1" = 0.002, "2" = 0.0025, "3" = 0.003),
  ratio_lc_lt_bound = rbind(
    "1" = c(lower = 0.7, upper = 2),
    "2" = c(lower = 0.8, upper = 1.4),
    "3" = c(lower = 0.8, upper = 2)
  ),
  control = control_fit_wrc_hcc(
    settings = "clocal", pcmp = control_pcmp(ncores = ncpu)))
coef(rfit_cloc_2, gof = TRUE, lc = TRUE, e0 = TRUE)

# ... global optimisation algorithm NLOPT_GN_MLSL
#   with sample-specific box-constraints

```

```

rfit_uglob_bc <- update(
  rfit_uglob,
  lower_param = rbind(
    "1" = c(alpha = 2.4, n = 1.11, thetar = 0.2, thetas = 0.6),
    "2" = c(alpha = 1.2, n = 1.12, thetar = 0.2, thetas = 0.6),
    "3" = c(alpha = 1.2, n = 1.13, thetar = 0.2, thetas = 0.6)
  ),
  upper_param = rbind(
    "1" = c(alpha = 20.1, n = 2.51, thetar = 0.6, thetas = 0.6),
    "2" = c(alpha = 20.2, n = 1.5, thetar = 0.6, thetas = 0.6),
    "3" = c(alpha = 1.3, n = 2.53, thetar = 0.6, thetas = 0.6)
  )
)
coef(rfit_uglob, gof = TRUE)
coef(rfit_uglob_bc, gof = TRUE)

```

---

 hc\_model

---

*Models for Soil Hydraulic Conductivity Functions*


---

## Description

The functions `hc_model` and `hcrel_model` compute, for given capillary pressure head  $h$ , the *hydraulic conductivity*  $K(h)$  and the *relative hydraulic conductivity*  $k(h)$  respectively, of a soil by parametrical models.

## Usage

```
hcrel_model(h, nlp, precBits = NULL, hcc_model = "vgm")
```

```
hc_model(h, nlp, lp, precBits = NULL, hcc_model = "vgm")
```

## Arguments

- |     |  |
|-----|--|
| h   | a mandatory numeric vector with values of capillary pressure head for which to compute the hydraulic conductivity. For consistency with other quantities, the unit of head should be <b>meter</b> [m].   |
| nlp | a mandatory named numeric vector, currently with elements named "alpha", "n" and "tau", which are the <i>nonlinear</i> parameters $\nu^T = (\alpha, n, \tau)$ , where $\alpha$ , $n$ and $\tau$ are the inverse air entry pressure, the shape and the tortuosity parameters, see <i>Details</i> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [m <sup>-1</sup> ]. |
| lp  | a mandatory named numeric vector, currently with a single element named "k0", which is the saturated hydraulic conductivity $K_0$ , the only <i>linear</i> parameter of the model, see <i>Details</i> . For consistency with other quantities, the unit of $K_0$ should be <b>meter/day</b> [m d <sup>-1</sup> ].  |

precBits	an optional integer scalar defining the maximal precision (in bits) to be used in high-precision computations by <a href="#">mpfr</a> . If equal to NULL (default) then <a href="#">mpfr</a> is not used and the result of the function call is of storage mode double, see <a href="#">soilhypfitIntro</a> .
hcc_model	a keyword denoting the parametrical model for the hydraulic conductivity function. Currently, only the <i>Van Genuchten-Mualem</i> model ( <code>wrc_model = "vgm"</code> ) is implemented, see <i>Details</i> .

## Details

The functions `hccrel_model` and `hc_model` currently model soil hydraulic conductivity functions only by the simplified form of the Van Genuchten-Mualem model (*Van Genuchten, 1980*) with the restriction  $m = 1 - \frac{1}{n}$ , i.e. by

$$k_{\text{VGM}}(h; \boldsymbol{\nu}) = S_{\text{VG}}(h; \boldsymbol{\nu})^\tau \left[ 1 - \left( 1 - S_{\text{VG}}(h; \boldsymbol{\nu})^{\frac{n}{n-1}} \right)^{\frac{n-1}{n}} \right]^2,$$

$$K_{\text{VGM}}(h; \mu, \boldsymbol{\nu}) = K_0 k_{\text{VGM}}(h; \boldsymbol{\nu}),$$

where  $\mu = K_0$  is the saturated hydraulic conductivity ( $K_0 > 0$ ),  $\boldsymbol{\nu}^T = (\alpha, n, \tau)$  are the inverse air entry pressure ( $\alpha > 0$ ), the shape ( $n > 1$ ) and tortuosity parameter ( $\tau > -2$ ), and  $S_{\text{VG}}(h; \boldsymbol{\nu})$  is the the volumetric water saturation, see `sat_model` for an expression.

Note that  $\mu$  and  $\boldsymbol{\nu}$  are passed to the functions by the arguments `lp` and `nlp`, respectively.

## Value

A numeric vector with values of (relative) hydraulic conductivity.

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## References

Van Genuchten, M. Th. (1980) A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, **44**, 892–898, doi:[10.2136/sssaj1980.03615995004400050002x](https://doi.org/10.2136/sssaj1980.03615995004400050002x).

## See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

## Examples

```
## define capillary pressure head (unit meters)
h <- c(0.01, 0.1, 0.2, 0.3, 0.5, 1., 2., 5., 10.)

## compute (relative) hydraulic conductivity
hcrel <- hcrel_model(h, nlp = c(alpha = 1.5, n = 2, tau = 0.5))
hc <- hc_model(h, nlp = c(alpha = 1.5, n = 2, tau = 0.5), lp = c(k0 = 5))

## display hydraulic conductivity function
op <- par(mfrow = c(1, 2))
plot(hcrel ~ h, log = "xy", type = "l")
plot(hc ~ h, log = "xy", type = "l")
on.exit(par(op))
```

---

profile\_loglikelihood *Profile Loglikelihoods and Confidence Intervals for Parametric Modelling of Soil Hydraulic Properties*

---

## Description

The function `prfloglik_sample` computes for a single soil sample a loglikelihood profile as a function of the specified values for subsets of model parameters of soil water retention and/or soil hydraulic conductivity functions. The function `confint_prfloglik_sample` computes a confidence interval of one model parameter based on the likelihood ratio test for a single soil sample, and the S3 method `confint` computes confidence intervals of *nonlinear* model parameters for multiple soil samples.

## Usage

```
prfloglik_sample(object, values, soil_sample,
  ncores = min(detectCores() - 1L, NROW(values)), verbose = 0)

confint_prfloglik_sample(object, parm = names(default_fit_param()),
  soil_sample, level = 0.95, test = c("F", "Chisq"),
  denominator_df = c("nonlinear", "all"), param_bound = NULL,
  root_tol = .Machine$double.eps^0.25, froot_tol = sqrt(root_tol),
  verbose = 0)

## S3 method for class 'fit_wrc_hcc'
confint(object,
  parm = names(object[["control"]][["initial_param"]]), level = 0.95,
  subset = NULL, type = c("loglik", "normal"), test = c("F", "Chisq"),
  denominator_df = c("nonlinear", "all"),
  root_tol = .Machine$double.eps^0.25, froot_tol = sqrt(root_tol),
  ncores = detectCores() - 1L, verbose = 0, ...)
```

**Arguments**

object	an object of class <code>fit_wrc_hcc</code> , see <a href="#">fit_wrc_hcc</a> .
values	a <code>data.frame</code> or a matrix with the values of the conditionally linear ( $\mu$ ) and nonlinear parameters ( $\nu$ ) that should be kept fixed to compute the likelihood profile (mandatory argument, see <a href="#">soilhypfitIntro</a> , <a href="#">wc_model</a> and <a href="#">hc_model</a> for information about the parametrization of models for soil water retention curves and/or soil hydraulic conductivity functions.). The names of the columns of values must match the names of model parameters.
soil_sample	a character scalar with the label of the soil sample for which the loglikelihood profile or the confidence interval should be computed. If object contains parameter estimates for a single soil sample then <code>soil_sample</code> is ignored, otherwise <code>soil_sample</code> is a mandatory argument.
ncores	an integer defining the number of cores for parallel computations. <code>ncores = 1</code> suppresses parallel computations.
verbose	positive integer controlling logging of diagnostic messages to the console and plotting of data and model curves during fitting, see <a href="#">fit_wrc_hcc</a> .
parm	character scalar ( <code>confint_prfloglik_sample</code> ) or vector ( <code>confint</code> ) with name(s) of parameter(s) for which to compute the confidence interval(s). Note that <code>confint_prfloglik_sample</code> allows to compute a confidence interval for <i>all</i> parameters (including the linear ones), whereas the <code>confint</code> method computes confidence intervals for the <i>nonlinear</i> parameters ( $\nu$ ) only, see <a href="#">soilhypfitIntro</a> for information about the parametrization of models for soil water retention curves and/or soil hydraulic conductivity functions.
level	numeric scalar with the confidence level required to compute the confidence interval.
test	character keyword specifying whether to use the asymptotic $\chi^2$ -distribution or the finite sample approximate F-distribution for the likelihood ratio test statistic when computing the confidence interval, see <i>Details</i> .
denominator_df	character keyword specifying whether the denominator degrees of freedom for the F-distribution of the test statistic is equal to the number of estimated <i>nonlinear</i> parameters ("nonlinear", default) or equal to the total number of estimated parameters ("all").
param_bound	a numeric vector of length 2 with the allowed range of the parameter for which the confidence interval is computed. The limits of the confidence interval are searched within this range, see <i>Details</i> . When equal to <code>NULL</code> (default) <code>param_bound</code> is taken from to component <code>initial_objects</code> of the selected component fit of object, see <i>Details</i> .
root_tol	a numeric scalar defining the desired accuracy (convergence tolerance) for root finding by <a href="#">uniroot</a> when computing the confidence interval.
froot_tol	a numeric scalar defining the desired accuracy (function value tolerance) for deciding whether a root has been found.
subset	an integer, character or logical vector to choose the soil samples for which confidence intervals should be computed. Defaults to <code>NULL</code> which computes the intervals for all samples contained in object.

type	character keyword specifying whether to compute confidence intervals based on the likelihood ratio test ("logik", default) by <code>confint_prfloglik_sample</code> or based on the asymptotic normal distribution of maximum likelihood estimates ("normal"), see <i>Details</i> .
...	additional arguments passed to methods, currently not used.

## Details

### Computing likelihood profiles:

The function `prfloglik_sample` computes the loglikelihood profile for a subset of the nonlinear ( $\nu$ ) (and linear  $\mu$ ) model parameters. We denote the profiling parameters of interest by  $\phi$  and the nuisance parameters that are estimated when computing the profile by  $\psi$ , so that  $(\phi^T, \psi^T)^T$  is a rearranged version of the parameter vector  $(\mu^T, \nu^T)^T$ , see [soilhyfitIntro](#). `prfloglik_sample` computes the estimates  $\hat{\psi}(\phi)$  of  $\psi$  and the profile loglikelihood  $Q(\phi, \hat{\psi}(\phi); \theta, K, h)$  as a function of  $\phi$ .

To compute the estimates,  $\psi$  is partitioned into the nonlinear and conditionally linear parameters, see [soilhyfitIntro](#) for details. `prfloglik_sample` uses the packages **parallel** and **snowfall** for parallelized computation of loglikelihood profiles.

### Computing likelihood based confidence intervals:

The function `confint_prfloglik_sample` computes the confidence interval of a single model parameter  $\phi \in (\mu^T, \nu^T)^T$  based on the likelihood ratio test. The interval is computed by assuming either

- that the test statistic  $T(\phi) = \Delta Q(\phi) = 2(Q(\hat{\phi}, \hat{\psi}; \theta, K, h) - Q(\phi, \hat{\psi}(\phi); \theta, K, h))$  follows a  $\chi_1^2$ -distribution with 1 degrees of freedom (possible choice when both water retention and/or hydraulic conductivity data were used to estimate the parameters), or
- that the transformed test statistic  $T(\phi) = (\exp(\Delta Q(\phi)/n) - 1)(n - p)$  follows an  $F(1, n - p)$ -distribution where  $n$  is the number of water content or hydraulic conductivity measurements, and  $p$  is the number of estimated parameters (see Uusipaikka, 2008, p. 115, for details). `denominator_df` allows to control how  $p$  is chosen. Note that this test distribution can only be chosen (and is then the default) when only water retention or only hydraulic conductivity data were used to estimate the parameters.

`confint_prfloglik_sample` computes profile loglikelihoods  $Q(\phi, \hat{\psi}(\phi); \theta, K, h)$  by `prfloglik_sample` and then uses [uniroot](#) to search the roots of the equation

$$f(\phi) = q_T(\gamma) - T(\phi)$$

in the interval defined by `param_bound`.  $q_T(\gamma)$  is the  $\gamma$ -quantile of the chosen test distribution for  $T$ .

### Computing confidence intervals for several soil samples:

The `confint` method computes 2 types of confidence intervals (in dependence of `type`) of only the nonlinear parameters  $\nu$ , possibly for multiple soil samples at the same time:

- intervals based on the asymptotic normal distribution of maximum likelihood estimates with standard errors computed by the `vcov` method for class `fit_wrc_hcc` (see [vcov.fit\\_wrc\\_hcc](#),
- intervals based on the likelihood ratio test by `confint_prfloglik_sample`. The intervals for several soil samples are computed in parallel.



**Requirements for computing loglikelihood profiles:**

The parameters contained in object must be estimated by maximum likelihood (method = "ml", see [soilhypfitIntro](#) and [control\\_fit\\_wrc\\_hcc](#)). Use of other estimation methods results in an error.

Preferably an unconstrained local algorithm (settings = "ulocal", see [soilhypfitIntro](#) and [control\\_fit\\_wrc\\_hcc](#)) is used for minimizing the negative loglikelihood when generating object. Use of other algorithms generates a warning message.

**Value**

The function `prfloglik_sample` returns a `data.frame` with the columns of values (parameters of interest  $\phi$ ), a column `loglik` with the maximized profile loglikelihood  $Q(\phi, \hat{\psi}(\phi); \theta, \mathbf{K}, \mathbf{h})$ , columns with the estimated parameters  $\hat{\psi}(\phi)$ , columns with the gradient of the loglikelihood with respect to the estimated nonlinear parameters (missing if all nonlinear parameters were fixed) and a column `converged`, indicating whether convergence has occurred (see [convergence\\_message](#)) when estimating the nonlinear parameters (equal to NA when all nonlinear parameters are fixed).

The function `confint_prfloglik_sample` returns a numeric vector of length 2 with the lower and upper limits of the confidence interval. If no roots were found then NA is returned. The returned result further contains as attribute list `prfloglik` the parameter estimate  $\hat{\phi}$  (`param_estimate`), the maximized loglikelihood  $Q(\hat{\phi}, \hat{\psi}; \theta, \mathbf{K}, \mathbf{h})$  (`loglik`), the quantile of the test distribution  $q_{\text{test}}(\gamma)$  (`qtest`), the type of test distribution used (`test`), the significance level, the number of water content (`nobs_wrc`) and conductivity measurements (`nobs_hcc`) and the function values of  $f(\phi)$  evaluated at the roots returned by [uniroot](#).

The method `confint` returns a `dataframe` with the lower and upper limits of the confidence intervals for the estimated nonlinear parameters.

**Author(s)**

Andreas Papritz <[papritz@retired.ethz.ch](mailto:papritz@retired.ethz.ch)>.

**References**

Uusipaikka, E. (2008) Confidence Intervals in Generalized Regression Models. Chapman & Hall/CRC Press, Boca Raton [doi:10.1201/9781420060386](#).

**See Also**

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

`evaporative-length` for physically constraining parameter estimates of soil hydraulic material functions.

### Examples

```
# use of \donttest{} because execution time exceeds 5 seconds

if(!requireNamespace("lattice", quietly = TRUE)){
  install.packages("lattice", repos = "https://cloud.r-project.org/")
  requireNamespace("lattice", quietly = TRUE)
}

data(sim_wrc_hcc)

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L

# estimate parameters for 3 samples simultaneously by ...

# ... unconstrained, global optimisation algorithm NLOPT_GN_MLSL (default)
rfit_uglob <- fit_wrc_hcc(
  wrc_formula = wc ~ head | id, hcc_formula = hc ~ head | id,
  data = sim_wrc_hcc,
  control = control_fit_wrc_hcc(param_bound = param_boundf(
    alpha = c(0.00001, 50), n = c(1.0001, 7), tau = c(-1, 5)
  ), pcmp = control_pcmp(ncores = ncpu))

# ... unconstrained, local optimisation algorithm NLOPT_LN_BOBYQA,
rfit_uloc <- update(
  rfit_uglob,
  param = as.matrix(coef(rfit_uglob, what = "nonlinear")),
  control = control_fit_wrc_hcc(
    settings = "ulocal", param_tf = param_transf(
      alpha = "identity", n = "identity", tau = "identity"
    ), param_bound = param_boundf(
      alpha = c(0.00001, 50), n = c(1.0001, 7), tau = c(-1, 5)
    ), pcmp = control_pcmp(ncores = ncpu))

# extract estimated parameters for sample id == "1"
coef_id_1 <- unlist(coef(rfit_uloc, gof = TRUE, se = TRUE, subset = "1"))

# compute loglikelihood profile of parameter alpha for sample id == "1"
rprfloglik_alpha <- prfloglik_sample(
  rfit_uloc, values = data.frame(alpha = seq(1.5, 3.0, length = 40L)),
  soil_sample = "1", ncores = ncpu)
# plot loglikelihood profile along with 95% confidence intervals
plot(loglik ~ alpha, rprfloglik_alpha, type = "l")
abline(v = coef_id_1["alpha"])
# 95% confidence intervall based on likelihood ratio test
abline(h = -coef_id_1["obj"] - 0.5 * qchisq(0.95, 1), lty = "dashed")
# 95% confidence intervall based on asymptotic normal distribution
segments(
  x0 = coef_id_1["alpha"] + qnorm(0.025) * coef_id_1["se.alpha"],
```

```

    x1 = coef_id_1["alpha"] + qnorm(0.975) * coef_id_1["se.alpha"],
    y0 = min(rprfloglik_alpha$loglik)
  )

  # compute loglikelihood profile of parameter n for sample id == "1"
  rprfloglik_n <- prfloglik_sample(
    rfit_uiloc, values = data.frame(n = seq(1.7, 2.25, length = 40L)),
    soil_sample = "1", ncores = ncpu
  )

  # plot loglikelihood profile along with 95% confidence intervals
  plot(loglik ~ n, rprfloglik_n, type = "l")
  abline(v = coef_id_1["n"])
  # 95% confidence intervall based on likelihood ratio test
  abline(h = -coef_id_1["obj"] - 0.5 * qchisq(0.95, 1), lty = "dashed")
  # 95% confidence intervall based on asymptotic normal distribution
  segments(
    x0 = coef_id_1["n"] + qnorm(0.025) * coef_id_1["se.n"],
    x1 = coef_id_1["n"] + qnorm(0.975) * coef_id_1["se.n"],
    y0 = min(rprfloglik_n$loglik)
  )

  # compute loglikelihood profile of parameters alpha and n for sample id == "1"
  rprfloglik_alpha_n <- prfloglik_sample(
    rfit_uiloc, values = expand.grid(
      alpha = seq(1.5, 3.0, length = 40L), n = seq(1.7, 2.25, length = 40L)),
    soil_sample = "1", ncores = ncpu
  )

  # joint confidence region of alpha and n based on likelihood ratio test
  lattice::levelplot(loglik ~ alpha + n, rprfloglik_alpha_n,
    panel = function(x, y, z, subscripts, ...){
      lattice::panel.levelplot(x, y, z, subscripts, ...)
      lattice::panel.levelplot(x, y, z, subscripts, region = FALSE, contour = TRUE,
        at = -coef_id_1["obj"] - 0.5 * qchisq(0.95, 2),
        lty = "solid"
      )
      lattice::panel.levelplot(x, y, z, subscripts, region = FALSE, contour = TRUE,
        at = -coef_id_1["obj"] - 0.5 * qchisq(0.9, 2),
        lty = "dashed"
      )
      lattice::panel.levelplot(x, y, z, subscripts, region = FALSE, contour = TRUE,
        at = -coef_id_1["obj"] - 0.5 * qchisq(0.8, 2),
        lty = "dotted"
      )
      lattice::panel.points(coef_id_1["alpha"], coef_id_1["n"], pch = 16)
      lattice::panel.lines(
        x = rprfloglik_alpha[, c("alpha", "n")], col = "blue"
      )
      lattice::panel.lines(
        x = rprfloglik_n[, c("alpha", "n")], col = "magenta"
      )
    }, key = list(
      corner = c(1, 0.97),

```

```

lines = list(
  lty = c(rep("solid", 3), "dashed", "dotted"),
  col = c("blue", "magenta", rep("black", 3))
),
text = list(c(
  "estimate of n as a function of fixed alpha",
  "estimate of alpha as a function of fixed n",
  "95% joint confidence region",
  "90% joint confidence region",
  "80% joint confidence region"
))
))

# compute 95%-confidence interval
(ci.alpha <- confint_prfloglik_sample(
  rfit_uloc, parm = "alpha", soil_sample = "1"
))
# use limits to draw loglikelihood profile for alpha
rprfloglik_alpha <- prfloglik_sample(
  rfit_uloc, values = data.frame(
    alpha = seq(0.9 * ci.alpha[1], 1.1 * ci.alpha[2], length = 40L)),
    soil_sample = "1", ncores = ncpu)
plot(loglik ~ alpha, rprfloglik_alpha, type = "l")
lines(
  ci.alpha,
  rep(diff(unlist(attr(ci.alpha, "prfloglik")[c("qtest", "loglik")])) , 2)
)
abline(v = attr(ci.alpha, "prfloglik")["estimate"], lty = "dashed")

# 95%-confidence intervals of all nonlinear parameters based for all
# soil samples asymptotic normal distribution of maximum likelihood estimates
confint(rfit_uloc, type = "normal")

# 95%-confidence intervals of all nonlinear parameters based for all
# soil samples based on likelihood ratio test
confint(rfit_uloc, ncores = ncpu)

```

---

sim\_wrc\_hcc

---

*Simulated Soil Water Retention and Hydraulic Conductivity Data*


---

## Description

The data give simulated values of water content and hydraulic conductivity at given capillary pressure head for 3 soil samples.

## Usage

```
data(sim_wrc_hcc)
```

## Format

A data frame with 28 observations on the following 4 variables.

id a factor with levels 1, 2, 3 coding the soil samples.

head a numeric vector with values of capillary pressure head (unit m).

wc a numeric vector with values of simulated volumetric water content (unit -)

hc a numeric vector with values of simulated hydraulic conductivity (unit  $\text{m d}^{-1}$ ).

## Details

The values of wc and hc were simulated by the model of *Van Genuchten Mualem* (Van Genuchten, 1980, see [wc\\_model](#) and [hc\\_model](#)) with the following parameters:

sample id	$\theta_r$	$\theta_s$	$K_0$ [ $\text{m d}^{-1}$ ]	$\alpha$ [ $\text{m}^{-1}$ ]	$n$	$\tau$
1	0.05	0.45	0.1	2	2	0.5
2	0.1	0.5	5	1.5	1.5	0.5
3	0.05	0.45	2	1.4	1.3	0.5

Normally distributed errors were added to the model values (wc: sd: 0.05; log(hc): sd 0.5).

## References

Van Genuchten, M. Th. (1980) A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, **44**, 892–898, [doi:10.2136/sssaj1980.03615995004400050002x](#).

## Examples

```
data(sim_wrc_hcc)

if(!requireNamespace("lattice", quietly = TRUE)){
  install.packages("lattice", repos = "https://cloud.r-project.org/")
  requireNamespace("lattice", quietly = TRUE)
}

lattice::xyplot(wc ~ head|id, type = "l", sim_wrc_hcc,
  scales = list( x = list(log=TRUE)))
lattice::xyplot(hc ~ head|id, type = "l", sim_wrc_hcc,
  scales = list( log = TRUE))
```

## Description

This page documents the methods `coef`, `summary`, `print`, `plot` and `lines` for the class `fit_wrc_hcc`.

## Usage

```
## S3 method for class 'fit_wrc_hcc'
coef(object, what = c("all", "nonlinear", "linear"),
      subset = NULL, residual_se = FALSE, se = FALSE, gof = FALSE, lc = FALSE,
      e0 = FALSE, bound = lc, ...)

## S3 method for class 'fit_wrc_hcc'
summary(object, what = c("all", "nonlinear", "linear"),
        subset = NULL, gof = TRUE, lc = TRUE, ...)

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

## S3 method for class 'fit_wrc_hcc'
plot(x, what = c("wrc", "hcc"), y = NULL,
     subset = NULL, ylim_wc = NULL, ylim_hc = NULL,
     head_saturation = 0.01,
     beside = identical(sum(par("mfrow")), 2L), pch = 1, col_points = "black",
     col_line_x = "blue", lty_x = "solid",
     col_line_y = "orange", lty_y = "dashed",
     xlab_wc = "head [m]", ylab_wc = "water content [-]",
     xlab_hc = "head [m]", ylab_hc = "hyd. conductivity [m/d]",
     draw_legend = TRUE, draw_parameter = FALSE, cex_legend = 0.7, ...)

## S3 method for class 'fit_wrc_hcc'
lines(x, what = c("wrc", "hcc"), id = 1,
      head_saturation = 0.01, ...)
```

## Arguments

<code>object, x, y</code>	an object of class <code>fit_wrc_hcc</code> , see <a href="#">fit_wrc_hcc</a> .
<code>what</code>	character keyword indicating the type of parameters to return ( <code>coef</code> ) or the type of data to plot.
<code>subset</code>	an integer, character or logical vector to choose the soil samples for which data and model curves are displayed or extracted. Defaults to <code>NULL</code> which displays results for all soil samples.

<code>residual_se</code>	a logical scalar to control whether residual standard errors (= standard deviations of residuals) should be returned, see <i>Details</i> .
<code>se</code>	a logical scalar to control whether standard errors of the nonlinear parameters $\nu$ should be returned, see <i>Details</i> and <code>vcov.fit_wrc_hcc</code> .
<code>gof</code>	a logical scalar to control whether goodness-of-fit statistic should be returned.
<code>lc</code>	a logical scalar to control whether the characteristic evaporative length should be returned, see <code>evaporative-length</code> .
<code>e0</code>	a logical scalar to control whether the evaporation rate should be returned. This is only effective for constrained estimation, see <code>evaporative-length</code> .
<code>bound</code>	a logical scalar to control whether the lower and upper bounds of the ratio $L_c/L_t$ should be returned. This is only effective for constrained estimation, see <code>evaporative-length</code> .
<code>ylim_wc</code>	optional numeric vector of length 2 to set the range of water content values displayed on the y-axis (default NULL for automatic axis scales).
<code>ylim_hc</code>	optional numeric vector of length 2 to set the range of hydraulic conductivity values displayed on the y-axis (default NULL for automatic axis scales).
<code>head_saturation</code>	head value (unit m) assigned to zero head values in plots with logarithmic head scale.
<code>beside</code>	a logical scalar controlling whether water retention curves and hydraulic conductivity functions of a sample should be plotted side by side.
<code>pch</code>	plotting 'character', i.e., symbol to use for the measurements, see <code>points</code> .
<code>col_points</code>	color code or name for symbol colors for the measurements, see <code>par</code> .
<code>col_line_x</code>	color code or name for the line representing the fitted model x, see <code>par</code> .
<code>lty_x</code>	type of line representing the fitted model x, see <code>par</code> .
<code>col_line_y</code>	color code or name for the line representing the fitted model y, see <code>par</code> .
<code>lty_y</code>	type of line representing the fitted model y, see <code>par</code> .
<code>xlab_wc</code>	a character string with the annotation for the x-axis of a water retention curve.
<code>ylab_wc</code>	a character string with the annotation for the y-axis of a water retention curve.
<code>xlab_hc</code>	a character string with the annotation for the x-axis of a hydraulic conductivity function.
<code>ylab_hc</code>	a character string with the annotation for the y-axis of a hydraulic conductivity function.
<code>draw_legend</code>	a logical scalar controlling whether a legend with the values of the arguments x and y and the residual sums of squares is drawn if y is non-NULL.
<code>draw_parameter</code>	a logical scalar controlling whether the parameters are drawn (default FALSE).
<code>cex_legend</code>	a character expansion factor for annotations by <code>legend</code> .
<code>id</code>	a character string or integer scalar to select the sample for which to plot the modelled water retention curve or hydraulic conductivity function.
<code>...</code>	additional arguments passed to methods.

## Details

Residual standard errors, standard errors of the nonlinear parameters and confidence intervals based on the asymptotic normal distribution are computed only for mpd and ml estimates, see [soilhypfitIntro](#), [control\\_fit\\_wrc\\_hcc](#) and [vcov](#).

The plot method for class `fit_wrc_hcc` displays for each sample the measurements of the water retention curve and/or the hydraulic conductivity function, along with the fitted model curve(s). Optionally, the curves of a second model fit (specified by the argument `y`) can be plotted for each sample.

The lines method adds the curve of a fitted model to an existing plot.

## Value

The method `coef` returns a dataframe with the estimated parameters (and optionally standard errors), optionally the value of the objective function along with convergence code and/or information on the characteristic evaporative length.

The method `summary` generates a list (of class `summary.fit_wrc_hcc`) with the following components:

`data` a named integer vector with the total number of samples (`nsamp`) and the number of samples with only water retention (`nwrc`), only hydraulic conductivity (`nhcc`) and both type of measurements (`nwrchcc`).

`control` a list with a subset (`settings`, `nloptr`, `sce`, `approximation_alpha_k0`, `param_bound`, `param_tf`) of the components of `object[["control"]]`, see [fit\\_wrc\\_hcc](#) and [control\\_fit\\_wrc\\_hcc](#).

`result` a dataframe with the estimated parameters and optionally the residual sum of squares along with convergence code and/or information on the characteristic evaporative length.

`call` the call component of object.

Note that only a print method is available for class `summary.fit_wrc_hcc`.

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.



## Examples

```
# use of \donttest{} because execution time exceeds 5 seconds

data(sim_wrc_hcc)

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L

# estimate parameters for 3 samples by unconstrained, global optimisation
# algorithm NLOPT_GN_MLSL
# sample 1: use only conductivity data
# sample 2: use only water content data
# sample 3: use both types of data
rfit_uglob <- fit_wrc_hcc(
  wrc_formula = wc ~ head | id, hcc_formula = hc ~ head | id,
  wrc_subset = id != 1, hcc_subset = id != 2,
  data = sim_wrc_hcc, fit_param = default_fit_param(tau = TRUE),
  control = control_fit_wrc_hcc(param_bound = param_boundf(
    alpha = c(0.00001, 50), n = c(1.0001, 7), tau = c(-1, 5)
  ), pcmp = control_pcmp(ncores = ncpu))
print(rfit_uglob)
summary(rfit_uglob)
coef(rfit_uglob, what = "nonlinear")
coef(rfit_uglob, what = "linear", gof = TRUE)
coef(vcov(rfit_uglob), status = TRUE, se = FALSE)
op <- par(mfrow = c(3, 2))
plot(rfit_uglob)
on.exit(par(op))
```

---

swissforestsoils

*Physical properties of Swiss forest soils*


---

## Description

The data give basic physical properties, water content and hydraulic conductivity measurements (at given capillary pressure head) for 128 soil layers (horizons) measured at 23 forest sites in Switzerland.

## Usage

```
data(swissforestsoils)
```

## Format

A data frame with 1373 observations on the following 21 variables.

`profile_id` a factor with short labels for the 23 sites.

profile a factor with with long labels for the 23 sites.  
 longitude a numeric vector with the latitude of the site in degree.  
 latitude a numeric vector with the latitude of the site in degree.  
 layer\_id a factor with labels for the 128 soil layer.  
 layer\_ub, layer\_lb numeric vectors with the upper and lower depth (unit cm) of the soil layer for the measurements of the basic physical properties (particle\_density, ..., ksat).  
 particle\_density a numeric vector with the density of the solid soil material (unit  $\text{g cm}^{-3}$ ).  
 bulk\_density a numeric vector with soil (bulk) density (unit  $\text{g cm}^{-3}$ ).  
 porosity a numeric vector with the soil porosity (unit volume percentage).  
 clay a numeric vector with the clay content (unit mass percentage).  
 silt a numeric vector with the silt content (unit mass percentage).  
 sand a numeric vector with the sand content (unit mass percentage).  
 ksat a numeric vector with the saturated hydraulic conductivity (unit  $\text{m d}^{-1}$ ).  
 head a numeric vector with capillary pressure head at which theta (water retention curve) and/or ku (hydraulic conductivity function) were measured (unit m).  
 layer\_ub\_theta, layer\_lb\_theta numeric vectors with the upper and lower depth (unit cm) of the soil layer for which the water retention curve was measured.  
 theta a numeric vector with volumetric water content measurements (dimensionless) of the water retention curve.  
 layer\_ub\_ku, layer\_lb\_ku a numeric vector with the upper and lower depth (unit cm) of the soil layer for which the water retention curve was measured.  
 ku a numeric vector with hydraulic conductivity measurements (unit  $\text{m d}^{-1}$ ) of the hydraulic conductivity function.

### Details

clay, silt and sand refer to soil particles with diameter less than 2, between 2 and 50 and larger than 50  $\mu\text{m}$ .

### Source

Richard, F. & Lüscher, P. 1978 – 1987. Physikalische Eigenschaften von Böden der Schweiz. Lokalformen Bände 1 – 4. Eidgenössische Anstalt für das forstliche Versuchswesen, Birmensdorf.

### Examples

```

# use of \donttest{} because execution time exceeds 5 seconds

# estimate parameters using all samples (samples with water retention,
# hydraulic conductivity, or with both type of measurements)

data(swissforestsoils)

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L

```

```
# unconstrained estimation (global optimisation algorithm NLOPT_GN_MLSL)
r_uglob <- fit_wrc_hcc(
  wrc_formula = theta ~ head | layer_id,
  hcc_formula = ku ~ head | layer_id,
  data = swissforestsoils,
  control = control_fit_wrc_hcc(
    settings = "uglobal", pcmp = control_pcmp(ncores = ncpu)))
summary(r_uglob)
coef(r_uglob)
```

utility-functions

*Utility functions for package soilhyppfit*

## Description

This page documents the functions `convergence_message`, `extract_error_messages`, `select_failed_fits` and `check_param_boundf`.

## Usage

```
convergence_message(x, sce = FALSE)
```

```
extract_error_messages(object, start = 1, stop = 80, prt = TRUE)
```

```
select_failed_fits(object)
```

```
check_param_boundf(y, compare_thetar_thetas = TRUE)
```

## Arguments

x	an integer scalar issued by the optimisers of <code>nloptr</code> or <code>SCEoptim</code> on (non-)convergence
sce	a logical scalar to select the optimiser <code>nloptr</code> (FALSE, default) or <code>SCEoptim</code> (TRUE).
object	an object of class <code>fit_wrc_hcc</code> , see <code>fit_wrc_hcc</code> .
prt	a logical scalar controlling whether the error messages should be printed.
start, stop	integer scalar with the first and last character to print.
y	a named list of numeric vectors of length 2 that define the allowed lower and upper bounds (box constraints) for the parameters of the models, see argument <code>param_bound</code> of <code>control_fit_wrc_hcc</code> .
compare_thetar_thetas	logical scalar to control cross-comparison of valid ranges of parameters <code>thetar</code> and <code>thetas</code> .

## Details

The function `convergence_message` prints a message that explains the convergence codes, for `nloptr`, see **NLopt return values**. The function `extract_error_messages` extract the error messages of estimations that failed and optionally prints sub-strings of them.

The function `select_failed_fits` returns the ids of the soil samples for which parameter estimation failed. The function `check_param_boundf` checks the validity and consistency of bounds of box constraints of model parameters.

## Value

The function `convergence_message` and `extract_error_messages` return invisibly the convergence code or the error messages.

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## References

Duan, Q., Sorooshian, S., and Gupta, V. K. (1994) Optimal use of the SCE-UA global optimisation method for calibrating watershed models, *Journal of Hydrology* **158**, 265–284, doi:10.1016/0022-1694(94)900574.

Johnson, S.G. The NLopt nonlinear-optimisation package. <https://github.com/stevengj/nlopt>.

## See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

## Examples

```
convergence_message(3)
convergence_message(2, sce = TRUE)
```

---

vcov	vcov <i>Method for Class fit_wrc_hcc</i>
------	--

---

## Description

This page documents the method `vcov` for the class `fit_wrc_hcc` and its `coef` method. `vcov` extracts the covariance matrices of the nonlinear parameters  $\hat{\boldsymbol{\nu}}$  estimated by maximum likelihood or maximum posterior density.

## Usage

```
## S3 method for class 'fit_wrc_hcc'
vcov(object, subset = NULL, grad_eps,
      bound_eps = sqrt(.Machine$double.eps), ...)

## S3 method for class 'vcov_fit_wrc_hcc'
coef(object, se = TRUE, correlation = se,
      status = FALSE, ...)
```

## Arguments

<code>object</code>	either an object of class <code>fit_wrc_hcc</code> for <code>vcov</code> or an object of class <code>vcov_fit_wrc_hcc</code> for <code>coef</code> .
<code>subset</code>	an integer, character or logical vector to choose the soil samples for which covariance matrices should be extracted. Defaults to <code>NULL</code> , which extracts the covariances for all soil samples.
<code>grad_eps</code>	a numeric scalar defining a critical magnitude of the moduli of scaled gradient components so that they are considered to be approximately equal to zero, see <i>Details</i> .
<code>bound_eps</code>	a numeric scalar defining the critical difference between parameter estimates and the boundaries of the parameter space so that the estimates are considered to be identical to the boundary values, see <i>Details</i> .
<code>se</code>	a logical scalar to control whether standard errors of the estimated nonlinear parameters $\hat{\boldsymbol{\nu}}$ should be returned ( <code>TRUE</code> , default) or variances ( <code>FALSE</code> ).
<code>correlation</code>	a logical scalar to control whether correlations ( <code>TRUE</code> , default) or covariances ( <code>FALSE</code> ) of the estimated nonlinear parameters $\hat{\boldsymbol{\nu}}$ should be returned.
<code>status</code>	a logical scalar to control whether diagnostics should be returned along with the results.
<code>...</code>	additional arguments passed to methods, currently not used.

## Details

The function `vcov` extracts (co-)variances of the nonlinear parameters from the inverse Hessian matrix of the objective function at the solution  $\hat{\nu}$  for `mpd` and `ml` estimates, see [soilhypfitIntro](#) and *Stewart and Sørensen (1981)*.

`vcov` checks whether the gradient at the solution is approximately equal to zero and issues a warning if this is not the case. This is controlled by the argument `grad_eps` which is the tolerable largest modulus of the scaled gradient (= gradient divided by the absolute value of objective function) at the solution. The function `control_fit_wrc_hcc` selects a default value for `grad_eps` in the dependence of the chosen optimisation approach (argument settings of `control_fit_wrc_hcc`).

`vcov` sets covariances equal to NA if the parameter estimates differ less than `bound_eps` from the boundaries of the parameter space as defined by `param_boundf`.

## Value

The method `vcov` returns an object of class `vcov_fit_wrc_hcc`, which is a list of covariance matrices of the estimated nonlinear parameters for the soil samples. The attribute `status` of the matrices qualifies the covariances.

The `coef` method for class `vcov_fit_wrc_hcc` extracts the entries of the covariances matrices, optionally computes standard errors and correlation coefficients and returns the results in a dataframe.

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## References

Stewart, W.E. and Sørensen, J.P. (1981) Bayesian estimation of common parameters from multire-sponse data with missing observations. *Technometrics*, **23**, 131–141,  
doi:10.1080/00401706.1981.10486255.

## See Also

[soilhypfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhypfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[wc\\_model](#) and [hc\\_model](#) for currently implemented models for soil water retention curves and hydraulic conductivity functions;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

## Examples

```
# use of \donttest{} because execution time exceeds 5 seconds

data(sim_wrc_hcc)

# define number of cores for parallel computations
if(interactive()) ncpu <- parallel::detectCores() - 1L else ncpu <- 1L

# estimate parameters for 3 samples by unconstrained, global optimisation
# algorithm NLOPT_GN_MLSL
# sample 1: use only conductivity data
# sample 2: use only water content data
# sample 3: use both types of data
rfit_uglob <- fit_wrc_hcc(
  wrc_formula = wc ~ head | id,
  hcc_formula = hc ~ head | id,
  wrc_subset = id != 1,
  hcc_subset = id != 2,
  data = sim_wrc_hcc,
  control = control_fit_wrc_hcc(pcmp = control_pcmp(ncores = ncpu)))
print(rfit_uglob)
summary(rfit_uglob)
coef(rfit_uglob, what = "nonlinear")
coef(rfit_uglob, what = "linear", gof = TRUE)
coef(vcov(rfit_uglob), status = TRUE, se = FALSE)
op <- par(mfrow = c(3, 2))
plot(rfit_uglob)
on.exit(par(op))
```

---

wc\_model

---

*Models for Soil Water Retention Curves*


---

## Description

The functions `sat_model` and `wc_model` compute, for given capillary pressure head  $h$ , the volumetric water saturation  $S(h)$  and the volumetric water content  $\theta(h)$ , respectively, of a soil by parametrical models.

## Usage

```
sat_model(h, nlp, precBits = NULL, wrc_model = "vg")

wc_model(h, nlp, lp, precBits = NULL, wrc_model = "vg")
```

## Arguments

h	a mandatory numeric vector with values of capillary pressure head for which to compute the volumetric water saturation or content. For consistency with other quantities, the unit of pressure head should be <b>meter</b> [m].
nlp	a mandatory named numeric vector, currently with elements named "alpha" and "n", which are the <i>nonlinear</i> parameters $\boldsymbol{\nu}^T = (\alpha, n)$ , where $\alpha$ and $n$ are the inverse air entry pressure and the shape parameter, see <i>Details</i> . For consistency with other quantities, the unit of $\alpha$ should be <b>1/meter</b> [m <sup>-1</sup> ].
lp	a mandatory named numeric vector, currently with elements named "thetar" and "thetas", which are the <i>linear</i> parameters $\boldsymbol{\mu}^T = (\theta_r, \theta_s)$ where $\theta_r$ and $\theta_s$ are the residual and saturated water content, respectively, see <i>Details</i> .
precBits	an optional integer scalar defining the maximal precision (in bits) to be used in high-precision computations by <code>mpfr</code> . If equal to NULL (default) then <code>mpfr</code> is not used and the result of the function call is of storage mode double, see <a href="#">soilhyfitIntro</a> .
wrc_model	a keyword denoting the parametrical model for the water retention curve. Currently, only the <i>Van Genuchten model</i> ( <code>wrc_model = "vg"</code> ) is implemented, see <i>Details</i> .

## Details

The functions `sat_model` and `wc_model` currently model soil water retention curves only by the simplified form of the model by *Van Genuchten (1980)* with the restriction  $m = 1 - \frac{1}{n}$ , i.e.

$$S_{VG}(h; \boldsymbol{\nu}) = (1 + (\alpha h)^n)^{\frac{1-n}{n}}$$

$$\theta_{VG}(h; \boldsymbol{\mu}, \boldsymbol{\nu}) = \theta_r + (\theta_s - \theta_r) S_{VG}(h; \boldsymbol{\nu})$$

where  $\boldsymbol{\mu}^T = (\theta_r, \theta_s)$  are the residual and saturated water content ( $0 \leq \theta_r \leq \theta_s \leq 1$ ), respectively, and  $\boldsymbol{\nu}^T = (\alpha, n)$  are the inverse air entry pressure ( $\alpha > 0$ ) and the shape parameter ( $n > 1$ ).

Note that  $\boldsymbol{\mu}$  and  $\boldsymbol{\nu}$  are passed to the functions by the arguments `lp` and `nlp`, respectively.

## Value

A numeric vector with values of volumetric water saturation (`sat_model`) or water content (`wc_model`).

## Author(s)

Andreas Papritz <papritz@retired.ethz.ch>.

## References

Van Genuchten, M. Th. (1980) A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, **44**, 892–898, doi:10.2136/sssaj1980.03615995004400050002x.



**See Also**

[soilhyppfitIntro](#) for a description of the models and a brief summary of the parameter estimation approach;

[fit\\_wrc\\_hcc](#) for (constrained) estimation of parameters of models for soil water retention and hydraulic conductivity data;

[control\\_fit\\_wrc\\_hcc](#) for options to control `fit_wrc_hcc`;

[soilhyppfitmethods](#) for common S3 methods for class `fit_wrc_hcc`;

[vcov](#) for computing (co-)variances of the estimated nonlinear parameters;

[prfloglik\\_sample](#) for profile loglikelihood computations;

[evaporative-length](#) for physically constraining parameter estimates of soil hydraulic material functions.

**Examples**

```
## define capillary pressure head (unit meters)
h <- c(0.01, 0.1, 0.2, 0.3, 0.5, 1., 2., 5., 10.)

## compute water saturation and water content
sat <- sat_model(h, nlp = c(alpha = 1.5, n = 2))
theta <- wc_model(
  h, nlp = c(alpha = 1.5, n = 2), lp = c(thetar = 0.1, thetas = 0.5))

## display water retention curve
op <- par(mfrow = c(1, 2))
plot(sat ~ h, log = "x", type = "l")
plot(theta ~ h, log = "x", type = "l")
on.exit(par(op))
```

# Index

- \* **datasets**
  - sim\_wrc\_hcc, 36
  - swissforestsoils, 41
- \* **hydraulic conductivity function**
  - control\_fit\_wrc\_hcc, 7
  - evaporative-length, 16
  - fit\_wrc\_hcc, 19
  - hc\_model, 28
  - profile\_loglikelihood, 30
  - soilhypfit-package, 2
  - soilhypfitS3methods, 38
  - utility-functions, 43
  - vcov, 45
- \* **methods**
  - soilhypfitS3methods, 38
  - vcov, 45
- \* **print**
  - soilhypfitS3methods, 38
  - vcov, 45
- \* **saturated hydraulic conductivity**
  - control\_fit\_wrc\_hcc, 7
  - evaporative-length, 16
  - fit\_wrc\_hcc, 19
  - hc\_model, 28
  - profile\_loglikelihood, 30
  - soilhypfit-package, 2
  - soilhypfitS3methods, 38
  - utility-functions, 43
  - vcov, 45
- \* **soil water characteristic**
  - control\_fit\_wrc\_hcc, 7
  - evaporative-length, 16
  - fit\_wrc\_hcc, 19
  - soilhypfit-package, 2
  - soilhypfitS3methods, 38
  - utility-functions, 43
  - vcov, 45
  - wc\_model, 47
- \* **soil water retention curve**
  - control\_fit\_wrc\_hcc, 7
  - evaporative-length, 16
  - fit\_wrc\_hcc, 19
  - soilhypfit-package, 2
  - soilhypfitS3methods, 38
  - utility-functions, 43
  - vcov, 45
  - wc\_model, 47
- \* **soil water**
  - control\_fit\_wrc\_hcc, 7
  - evaporative-length, 16
  - fit\_wrc\_hcc, 19
  - hc\_model, 28
  - profile\_loglikelihood, 30
  - soilhypfit-package, 2
  - soilhypfitS3methods, 38
  - utility-functions, 43
  - vcov, 45
  - wc\_model, 47
- bwd\_transf (control\_fit\_wrc\_hcc), 7
- check\_param\_boundf (utility-functions), 43
- class, 23
- coef.fit\_wrc\_hcc, 5
- coef.fit\_wrc\_hcc (soilhypfitS3methods), 38
- coef.vcov\_fit\_wrc\_hcc (vcov), 45
- confint.fit\_wrc\_hcc (profile\_loglikelihood), 30
- confint\_prfloglik\_sample (profile\_loglikelihood), 30
- control\_fit\_wrc\_hcc, 4–6, 7, 11, 16, 17, 21–23, 25, 29, 33, 40, 43, 44, 46, 49
- control\_nloptr, 5, 22
- control\_nloptr (control\_fit\_wrc\_hcc), 7
- control\_pcmp, 23
- control\_pcmp (control\_fit\_wrc\_hcc), 7
- control\_sce, 5, 22

- control\_sce (control\_fit\_wrc\_hcc), 7
- control\_soilhypfit
  - (control\_fit\_wrc\_hcc), 7
- convergence\_message, 23, 33
- convergence\_message
  - (utility-functions), 43
- default\_fit\_param (fit\_wrc\_hcc), 19
- dfwd\_transf (control\_fit\_wrc\_hcc), 7
- evaporative-length, 16
- extract\_error\_messages, 23
- extract\_error\_messages
  - (utility-functions), 43
- fit\_wrc\_hcc, 2–7, 9–11, 14, 17, 19, 29, 31, 33, 38, 40, 43, 44, 46, 49
- formula, 20
- fwd\_transf (control\_fit\_wrc\_hcc), 7
- hc\_model, 4, 6, 8, 14, 16, 18, 21, 22, 24, 25, 28, 31, 33, 37, 40, 44, 46
- hcrel\_model (hc\_model), 28
- lc (evaporative-length), 16
- legend, 39
- lines.fit\_wrc\_hcc
  - (soilhypfitS3methods), 38
- lt (evaporative-length), 16
- model.frame, 24, 25
- mpfr, 9, 11, 29, 48
- na.exclude, 20
- na.fail, 20
- na.omit, 20
- nloptr, 5, 8, 10, 12, 14, 22, 23, 43, 44
- nloptr.print.options, 12, 13, 22
- options, 20
- par, 39
- param\_boundf, 4, 10, 21, 46
- param\_boundf (control\_fit\_wrc\_hcc), 7
- param\_transf, 4
- param\_transf (control\_fit\_wrc\_hcc), 7
- plot.fit\_wrc\_hcc (soilhypfitS3methods), 38
- points, 39
- prfloglik\_sample, 6, 14, 18, 25, 40, 44, 46, 49
- prfloglik\_sample
  - (profile\_loglikelihood), 30
- print.fit\_wrc\_hcc
  - (soilhypfitS3methods), 38
- print.summary\_fit\_wrc\_hcc
  - (soilhypfitS3methods), 38
- profile\_loglikelihood, 30
- s, 9
- sat\_model (wc\_model), 47
- SCEoptim, 5, 8, 10, 12–14, 19, 22, 23, 43
- select\_failed\_fits, 23
- select\_failed\_fits (utility-functions), 43
- sim\_wrc\_hcc, 36
- soilhypfit (soilhypfit-package), 2
- soilhypfit-package, 2
- soilhypfitIntro, 8–11, 14, 16, 17, 20–23, 25, 29, 31–33, 40, 44, 46, 48, 49
- soilhypfitIntro (soilhypfit-package), 2
- soilhypfitmethods, 6, 14, 17, 25, 29, 33, 44, 46, 49
- soilhypfitmethods
  - (soilhypfitS3methods), 38
- soilhypfitS3methods, 38
- solve.QP, 10
- summary.fit\_wrc\_hcc
  - (soilhypfitS3methods), 38
- swissforestsoils, 41
- uniroot, 31–33
- utility-functions, 43
- vcov, 6, 14, 17, 25, 29, 33, 40, 44, 45, 49
- vcov.fit\_wrc\_hcc, 32, 39
- wc\_model, 4–6, 8, 14, 16, 18, 21, 22, 24, 25, 31, 33, 37, 40, 44, 46, 47