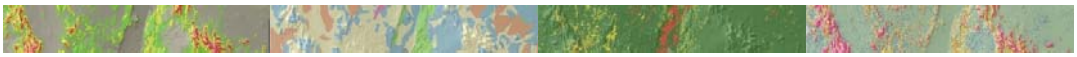
Makerere University College of Agricultural and Environmental Sciences

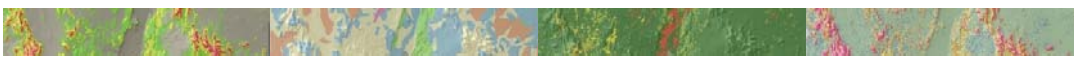


Short course

Soil Erosion Risk Modelling with R

21. / 22. February 2019

The aim of this course is to introduce the participants into environmental modelling of spatial data using the open source software R. The workshop will cover a basic introduction in R and RStudio and a brief introduction in working with spatial data in R. A practical example gives the participants the chance to apply the learned techniques. We will perform a simple soil erosion study implementing the Revised Universal Soil Loss Equation (RUSLE). Employing uncertainty analysis, the participants will learn how to interpret results and how to make informed decisions based on uncertain results. The workshop is intended to be hands on! Therefore, we expect that the participants of the workshop bring their own notebooks to follow the course. Data and Software will be provided during the workshop.



AUSTRIAN DEVELOPMENT COOPERATION appear oead

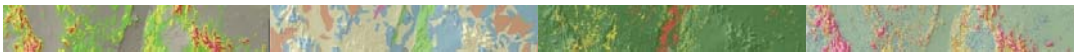
Preliminary schedule:

Day 1 (09:00 - 17:30)

- Opening
- Software and Data installation, Introduction to R
- Introduction to Spatial Data in R & Soil Erosion Risk Modelling using the RUSLE, including potential data sources

Day 2 (09:00 - 17:30)

- Application of RUSLE for a specific Case study (e.g. Sio-Malaba-Malakisi River Basin)
- Range of possible input layers & uncertainty assessment
- Influence of possible management practices on Soil erosion risk
- Calculation of Soil Loss for catchments / administrative boundaries (and comparison with observed data)
- Wrap-Up & Closing



This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex). The aim of this project is to strengthen the capacities of young researchers from Kenyan, Ugandan and Austrian Universities to cope with challenges associated to water quality and quantity, energy provision and food security in East Africa. The project is funded by the Austrian government through the APPEAR programme of the Austrian Development Cooperation. Project partners are TU Wien, Makerere University, TU Kenya, and BOKU.

Short course

Soil Erosion Risk Modelling with R

Overview | Organization

Mathew Herrnegger & Christoph Schürz

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

Overview - Soil Erosion Risk Modelling with R

- **Aim:** Introduce participants into environmental modelling of spatial data using the open source software R, also including an introduction to R and RStudio
- **Learning by doing:** Perform a soil erosion study implementing the Revised Universal Soil Loss Equation (RUSLE)
- **The world is uncertain:** Employ a parsimonious uncertainty analysis, to learn how to interpret results and how to make informed decisions based on uncertain results
- **Materials:** Data and Software will be provided
- **The short course will be hands on!**

Sessions

Day 1 (09:00 – 17:30)

- Overview | Organization
- Software and Data installation
- Our background: BOKU | HyWa | CapNex
- Introduction to R and RStudio
- 1 - Input data generation for the Revised Universal Soil Loss Equation (RUSLE)
- 2 - Estimation of soil erosion using the RUSLE

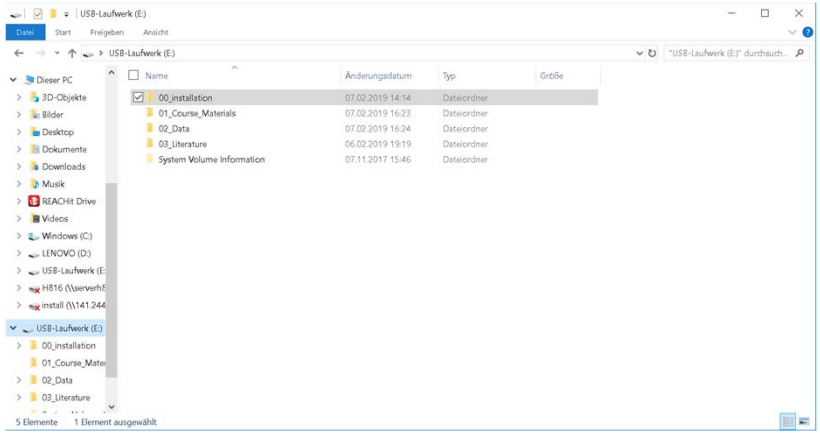
Day 2 (09:00 – 17:30)

- 2 - Estimation of soil erosion using the RUSLE
- 3 - Uncertainty assessment of soil erosion estimates
- **Closing ceremony**

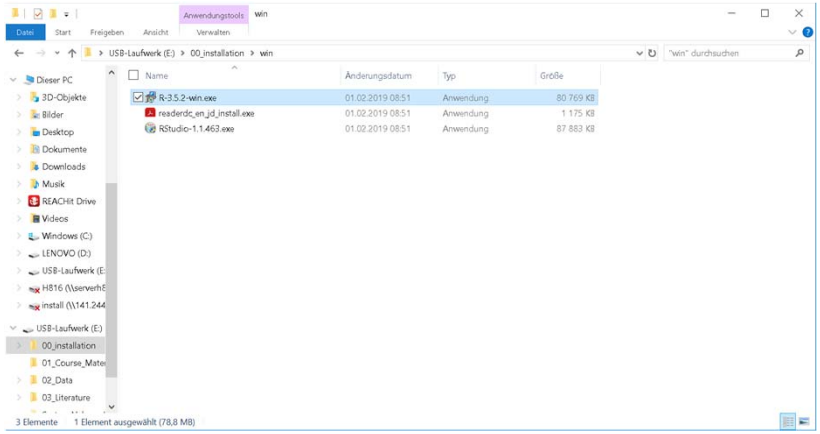
Organisation - Software and Data installation

- We have a number of USB sticks, where the course materials are stored
- Copy the folder from the USB stick to your Notebook – NOT on the desktop but directly to the root of your hard drive, e.g. c:\SoilErosionR or d:\SoilErosionR
- The folders contains:
  - PDF of presentations and documents showing the step-by-step procedure
  - Data for the course, e.g. Raster-Layers of Rainfall, NDVI etc., Shapefiles
  - Literature
- Software installation sequence:
  1. Base R: R-3.5.2-win.exe
  2. RStudio: RStudio-1.1.463.exe
  3. For Windows-User: Install Rtools (Rtools35.exe)
  4. Use the R-script provided (00_package_installation.R) in RStudio and install libraries

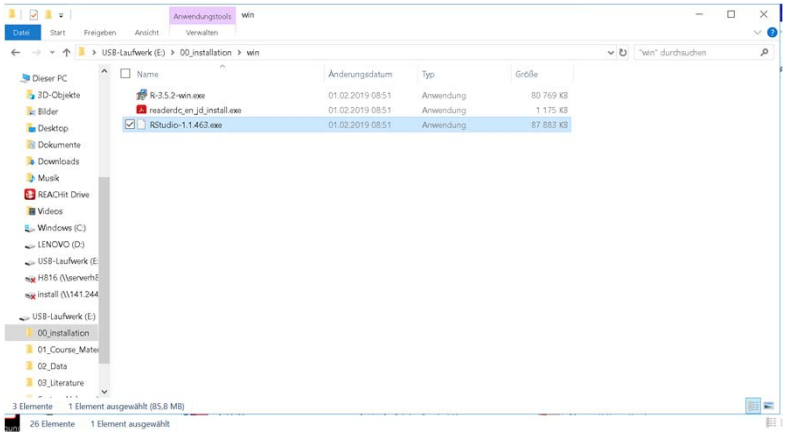
Software installation – Installation folder



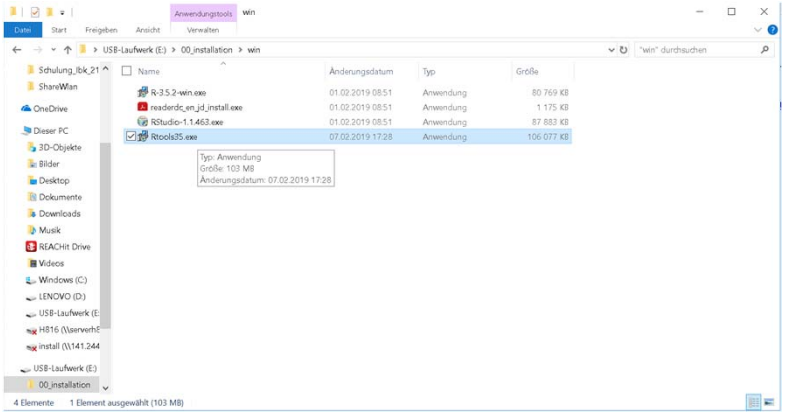
Software installation – Install Base R



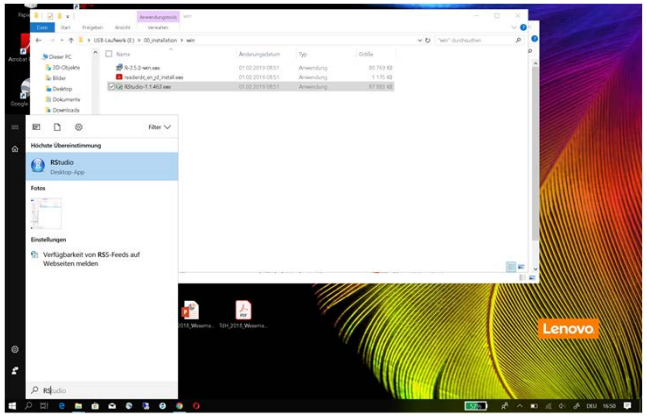
Software installation – Install RStudio



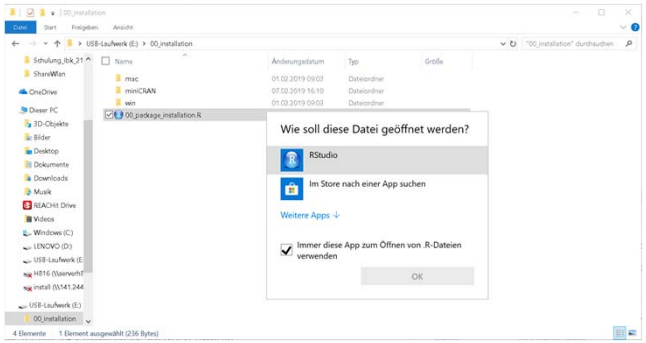
Software installation – Install Rtools if you use Windows



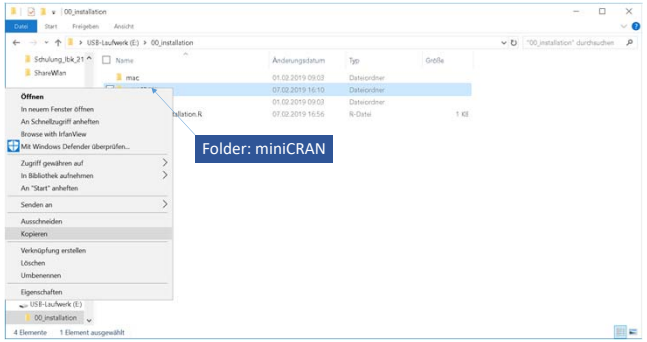
Software installation – Start RStudio



Software installation – open installation Rscript



Software installation – the file path to the Rpackage repository

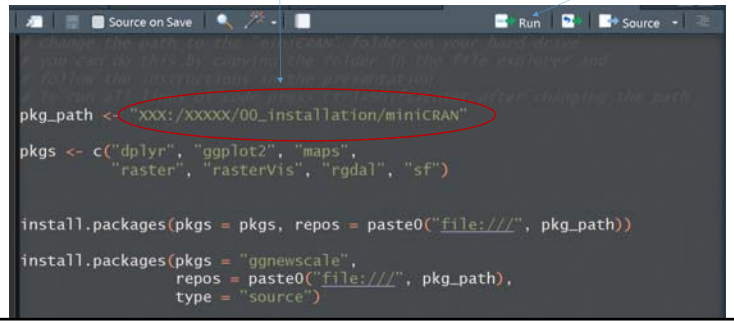


Software installation – R package installation

1. Replace this path with the path you copied

2. Select all code (CTRL+A or with mouse)

3. Run the entire code in the RScript




Thank you for your attention!

Short course

Soil Erosion Risk Modelling with R

Introduction to BOKU | HyWa | CapNex

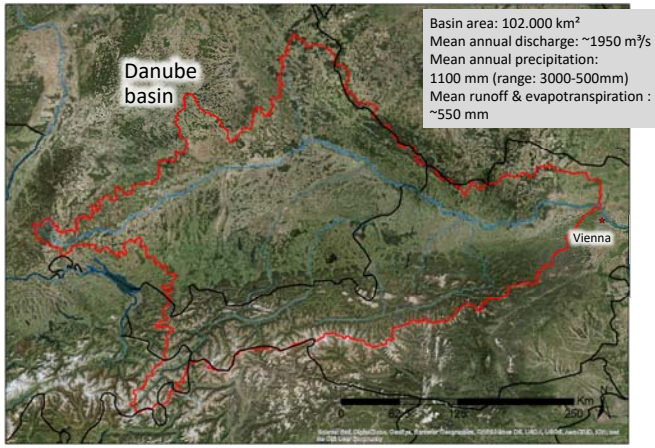
Mathew Herrnegger & Christoph Schürz

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

For the sake of orientation

For the sake of orientation

For the sake of orientation



University of Natural Resources and Life Sciences, Vienna (BOKU)

University of Natural Resources and Life Sciences, Vienna (BOKU)

- Founded in 1872
- ~ **13.000 students** in 8 Bachelor, 26 Master (+ several double degree programs; 11 Master programs in English) and several PhD programs (~ 800 students)
- **~1650 graduates** per year; students satisfaction: top ranked in Austria; 20% foreign students
- **Greenmetric World University Ranking**: no. **6** world wide (of 516 universities)
- ~ 1600 employees (full time equivalent), **2550 employees** (head count); ~700 scientists employed on a project basis; ~ **75 full professors** (1/3 non Austrians), ~ **130 Assoc. Profs**

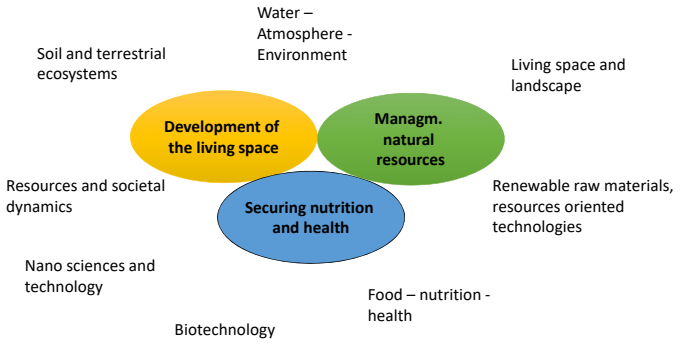
University of Natural Resources and Life Sciences, Vienna (BOKU)

Our Mission:

- **education and research centre for renewable resources**, which are a necessity for **human life**
- to contribute significantly to the **protection of life resources** for future generations
- connecting **natural sciences, engineering and economics**
- BOKU is trying to deepen the knowledge of an **ecologically and economically sustainable use of natural resources in a cultivated landscape**.

University of Natural Resources and Life Sciences, Vienna (BOKU)

Themes and competences



University of Natural Resources and Life Sciences, Vienna (BOKU)

Incoming students

University of Natural Resources and Life Sciences, Vienna (BOKU)

BOKU bachelor programs

- Forestry
- Wood and Fibre Technology
- Environment and Bio Resources Mangement
- Environmental Engineering
- Food Sciences and Biotechnology
- Agricultural Sciences
- Landscape Architecture and Planning
- Equine Sciences

Further information: www.boku4you.at : www.boku4you.at

University of Natural Resources and Life Sciences, Vienna (BOKU)

BOKU master programs (German)

- Agricultural and Food Economics (H 457)
- Alpine Natural Dangers/Watershed Regulation (477)
- Biotechnology (H 418)
- Crop Sciences (455)
- Environment and Bio Resources Management (H 427)
- Environmental Engineering and Water Management (H 431)
- Food Science and Technology (H 417)
- Forest Science (H 425)
- Landscape Architecture and Planning (H 419)
- Livestock Sciences (456)
- Organic Agricultural Systems and Agroecology (H500)
- Phytomedizin (H 422)
- Wildlife Ecology and Wildlife Management (H 423)
- Wood Technology and Management (H 426)

Further information: www.boku4you.at : www.boku4you.at

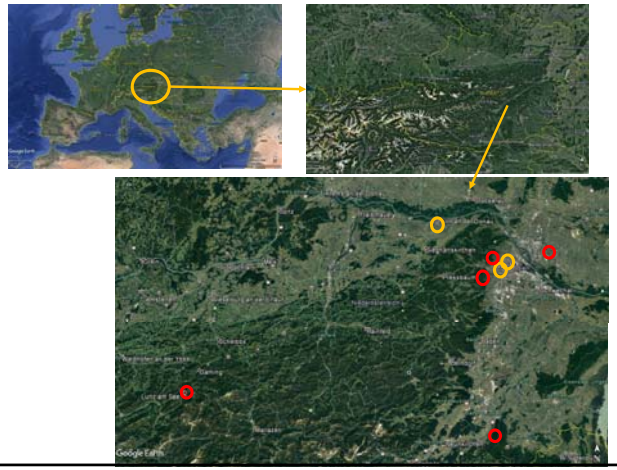
University of Natural Resources and Life Sciences, Vienna (BOKU)

BOKU master programs (International)

- Animal Breeding and Genetics
- Applied Limnology – Wetland Management
- Environmental Sciences – Soil, Water and Biodiversity (ENVEURO)
- European Forestry
- Horticultural Sciences
- Material and thermal utilization of renewable raw materials (in German)
- Mountain Forestry
- Natural Resources Management and Ecological Engineering (NARMEE)
- Mountain Risk Engineering
- Organic Agricultural Systems and Agroecology
- Safety in the Food Chain
- Sustainability in Agriculture, Food Production and Food Technology in the Danube Region
- Viticulture, Oenology and Wine Economy (in German)
- Water Management and Environmental Engineering

Further information: www.boku4you.at : www.boku4you.at

University of Natural Resources and Life Sciences, Vienna (BOKU)



University of Natural Resources and Life Sciences, Vienna (BOKU)

Site Türkenschanze

University of Natural Resources and Life Sciences, Vienna (BOKU)

Site Muthgasse

University of Natural Resources and Life Sciences, Vienna (BOKU)

Site Tulln

University of Natural Resources and Life Sciences, Vienna (BOKU)

Research farms and forests

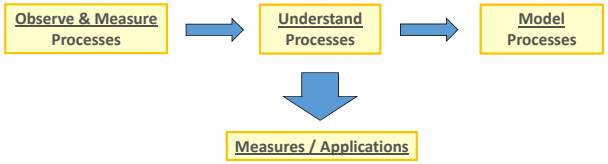
Institute for Hydrology and Water Management(HyWa)

HyWa

Institute for Hydrology and Water Management(HyWa)

Main research and teaching areas

- Hydrology
- Water Management / Integrative Catchment Management



Further information: http://www.wau.boku.ac.at/hywa/

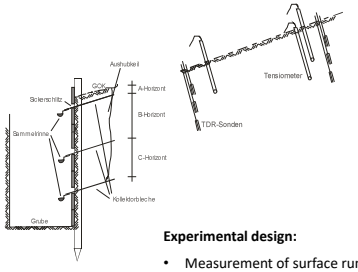
Institute for Hydrology and Water Management(HyWa)

Staff

- 1 Full Professor: Karsten Schulz (Head)
- 1 Emer. Prof.
- 3 Associate/Assistant Professors
- ~15 Project Assistants / PhD-Canditates / Senior Scientists
- ~5 Technicians
- 1 Secretary

Further information: http://www.wau.boku.ac.at/hywa/

Hydrology: Runoff Formation in Alpine Ecosystems



Experimental design:

- Measurement of surface runoff and interflow
- TDR measurements in different depths
- Complementary tensiometer measurements

Irrigation experiments in different forest stands

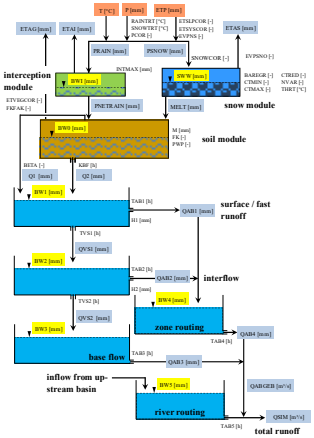
Hydrology: Rainfall-Runoff modelling

COSERO (COntinous SEmidistributed RunOff Model)

- Conceptual, HBV-type model
- Developed at the IWHW, ongoing changes and improvements since 1993
- Runoff process is modelled with a series of linear and non-linear reservoirs

Considered processes include:

- Snow accumulation and -melt
- Interception
- Infiltration, Soil storage
- Evapotranspiration
- Runoff separation (Surface-runoff, Inter-flow, Groundwater-flow)
- Lakes
- Local routing to catchment outlet
- Channel routing



Hydrology: Rainfall-Runoff modelling – COSERO (cont.)

Applications

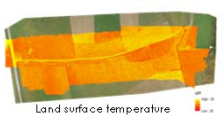
- Research
- Flood- and Inflow-Forecasting-Systems
- Climate change impact studies
- Vulnerability of water resources
- LULCC
- Water balance studies
- Digital Hydrological Atlas of Austria
- ….

Table 1 Examples of previous COSERO applications. Only those studies are listed where the first author of this paper had direct access to the executable models (to compute the data for this table). P: mean annual precipitation in calibration period. PET: mean annual potential evapotranspiration in calibration period. Q: mean annual runoff-depth in calibration period. All performance statistics (KGE', r, β, γ, NSE) are dimensionless.

Country[(a)]	River	Area (km²)	Snow	P (mm)	PET/P (-)	Q/P (-)	Calibration period	Objective	Calibration performance (-) KGE'	r	β	γ	NSE	Application	Reference
Monthly time step															
Austria	Ybbs[(b)]	507	Yes	1776	0.34	0.68	1961-1990	Manual[(c)]	0.84	0.87	0.98	0.90	0.76	Hydrological atlas	Kling and Nachtnebel (2009a)
Austria	Gail	1305	Yes	1588	0.33	0.69	1961-1990	Manual[(c)]	0.87	0.93	1.00	0.89	0.87	Research	Kling and Nachtnebel (2009b)
Germany	Danube	101 810	Yes	1130	0.52	0.53	1961-1990	KGE'	0.94	0.94	1.00	1.00	0.87	Research	Kling *et al.* (2012)
Zambia	Zambezi	519 399	No	942	1.71	0.08	1961-1990	KGE'	0.92	0.94	1.00	0.96	0.88	Decision support	Kling *et al.* (2014)
Daily time step															
Austria	Enns	2116	Yes	1215	0.42	0.74	1971-1983	Manual	0.94	0.95	1.03	0.99	0.89	Research	Nachtnebel and Fuchs (2004)
Austria	Gail	1305	Yes	1577	0.36	0.66	1971-1995	Manual	0.86	0.92	0.98	0.89	0.84	Research	Eder *et al.* (2005)
Austria	Glan[(c)]	432	Yes	902	0.67	0.35	1973-1988	NSE	0.84	0.89	1.03	0.89	0.79	Research	Gupta *et al.* (2009)
Portugal	Paiva	647	(Yes)	1676	0.58	0.64	1960-1990	KGE	0.95	0.96	0.98	0.99	0.91	Hydro power	Unpublished[(f)] (2011)
Turkey	Seyhan	13 264	Yes	514	1.81	0.45	2001-2006	KGE'	0.84	0.84	0.99	1.00	0.69	Hydro power	Unpublished[(f)] (2013)
Zambia	Zambezi	519 399	No	1012	1.62	0.07	1998-2009	KGE'	0.91	0.91	1.00	1.00	0.83	Flood forecasting	Unpublished[(f)] (2013)
Mozambique	Revubue	16 263	No	967	1.51	0.12	1998-2012	KGE'	0.83	0.83	1.01	1.00	0.65	Flood forecasting	Unpublished[(f)] (2013)
Laos	Kading	8668	No	2428	0.42	0.62	2000-2007	Manual	0.72	0.82	1.04	0.79	0.67	Hydro power	Unpublished[(f)] (2014)
Mali	Niger	1.6×10^6	No	488	4.63	0.05	1998-2005	KGE'	0.96	0.96	1.00	1.00	0.92	Flood forecasting	Unpublished[(f)] (2014)
Hourly time step															
California, US	Carson	922	Yes	1150	0.87	0.44	1990-1997	NSE	0.88	0.96	1.06	0.91	0.92	Research	Kling *et al.* (2008)
Arizona, US	Marshall	1.5	(Yes)	696	0.72	0.43	2007-2010[(d)]	KGE	0.86	0.87	1.04	0.97	0.74	Research	Unpublished[(d)] (2011)
Papua Guinea	Frieda	1036	No	7678	0.20	0.81	1995-2009	KGE'	0.87	0.87	1.00	0.99	0.73	Flood assessment	Unpublished[(f)] (2011)
15-minutes time step															
Austria	Traisen	729	Yes	1243	0.50	0.52	2003-2011	Manual	0.90	0.94	1.07	1.03	0.84	Flood forecasting	Stanzel *et al.* (2008)

(Kling et al. 2015)

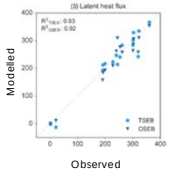
Hydrology: Remote sensing hydrology (e.g. evapotranspiration)


Land surface temperature

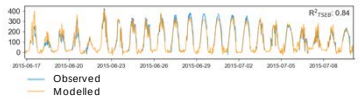

Energy balance model

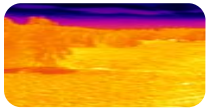

EC station for comparison

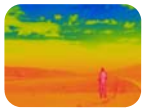
Instantaneous evapotranspiration from UAS measurements



Evapotranspiration timeseries from field radiometer measurements



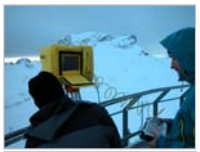


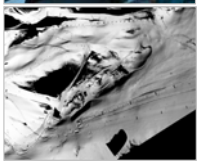


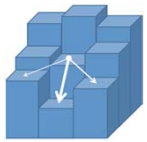
Institute for Hydrology and Water Management(HyWa)

Hydrology: Snow Hydrology-Monitoring & Modeling

UFS Schneefernerhaus







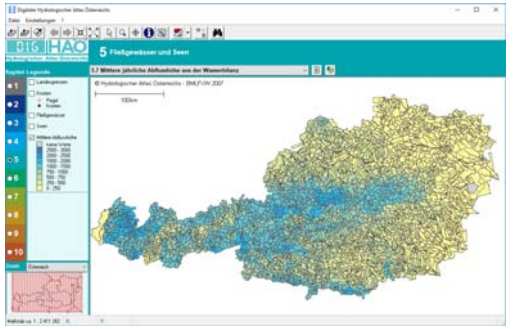
SnowSlide

Institute for Hydrology and Water Management(HyWa)

Water Management: Hydrological Atlas of Austria

A nation wide set of digital and printed maps of the main hydrological features of Austria

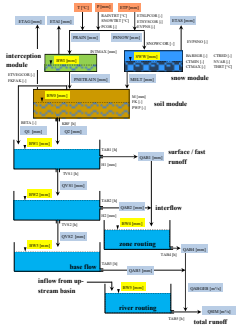
Hydrologischer Atlas Österreichs

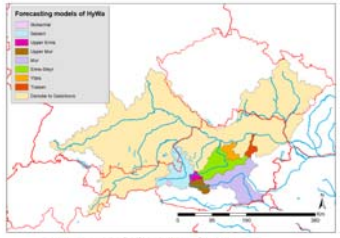


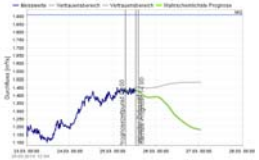
Institute for Hydrology and Water Management(HyWa)

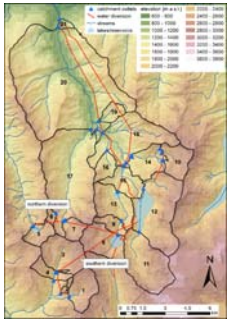
Water Management: Operational Runoff Forecasting

Rainfall-Runoff Model COSERO





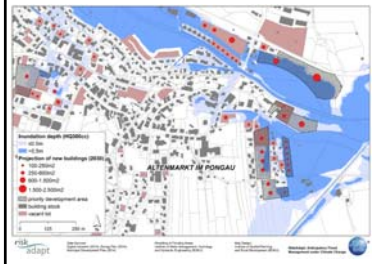


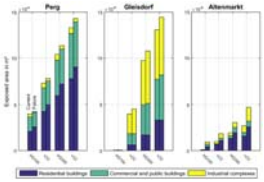


Institute for Hydrology and Water Management(HyWa)

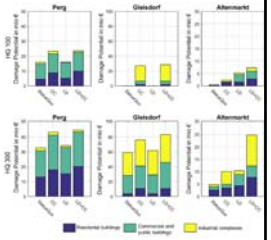
Water Management: Flood risk assessment under changing environments

Hydraulic Modelling & Analysis of flood hazards





Assessment of damage potentials and flood risk



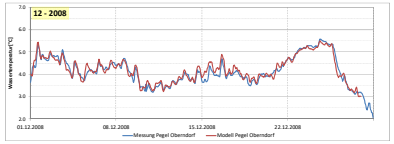
Institute for Hydrology and Water Management(HyWa)

Water Management: Influence of thermal loads in river temperature

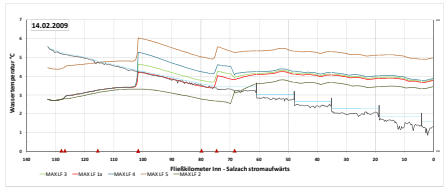
Anthropogenic thermal loads

Field measurements

Modeling



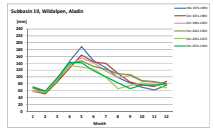
Projections for decision makers

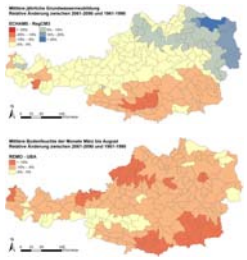


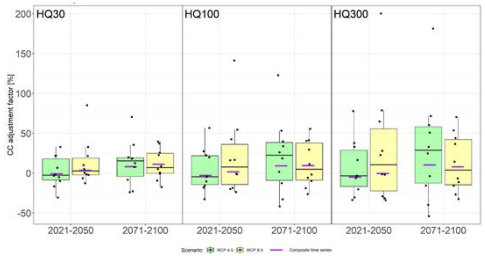
Institute for Hydrology and Water Management(HyWa)

Water Management: Potential impacts of CC on water resources

- Water balances
- Snow cover
- Floods / flood extremes
- Drinking water resources
- Groundwater recharge
- Water quality





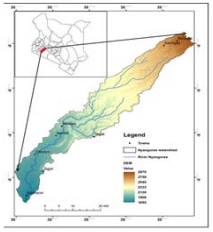


Institute for Hydrology and Water Management(HyWa)

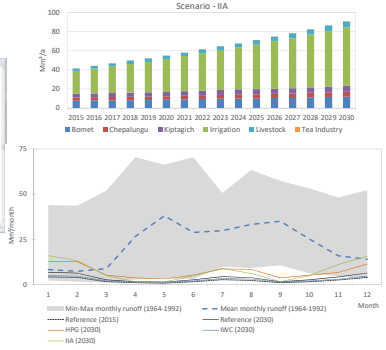
Water Management: Water use and allocation

„MaMa-Hydro: Exploring water resources planning and management options - Masai Mara River Basin in Kenya"

Nyangores catchment

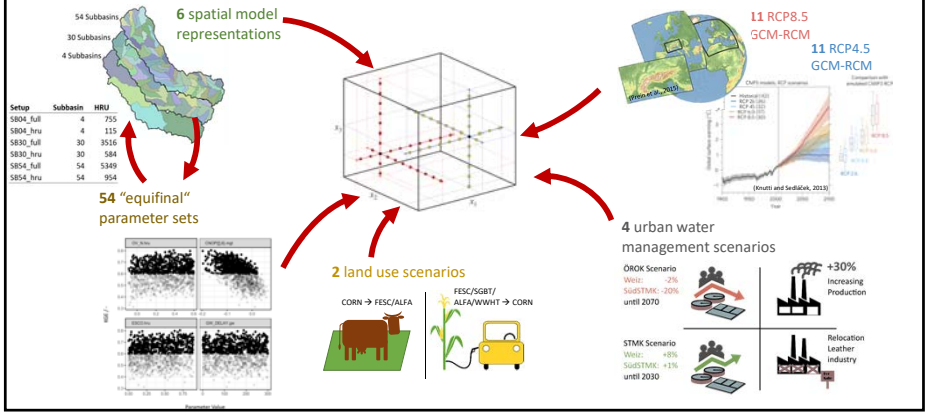


"Water Evaluation And Planning" system WEAP



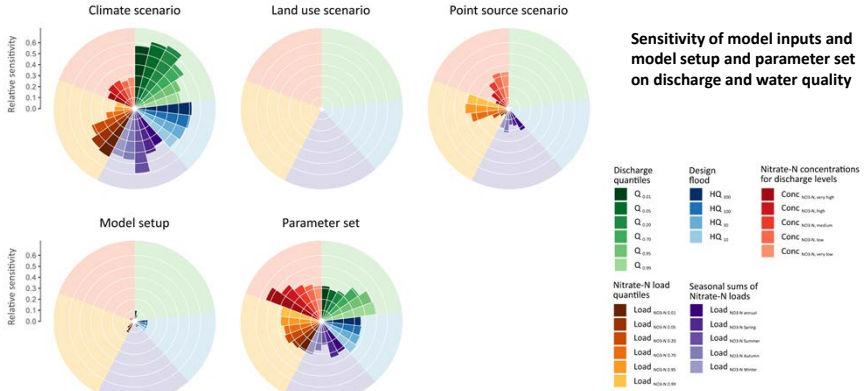
Institute for Hydrology and Water Management(HyWa)

Water Management: Water quality under climate and land use change



Institute for Hydrology and Water Management(HyWa)

Water Management: Water quality under climate and land use change



"Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" | CapNex

CapNex - Project

"Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" | CapNex

- Within Austrian Partnership Programme in Higher Education and Research for Development (APPEAR)
  - funded by the Austrian Development Agency (ADA), the operational unit of the Austrian Development Cooperation (ADC)
  - implemented by "Österreichischer Austauschdienst" (OeAD)

CapNex - Background

Competition for natural resources

Natural resources

- Water bodies (rivers, lakes, sea, ground water, …)
- Soils and biomass (wood, food crops, energy crops, …)
- Land (agriculture, roads, houses, …)
- Metals and minerals

Problem: competition for natural resources by different uses

- River: hydro power plant (energy) or irrigation (water/food) or fisheries (food)
- Groundwater: for irrigation (food) or drinking (water)
- Soils: jatropha plantation (energy) or maize plantation (food)
- Land: tree plantation (energy) or pasture (food)

- Sollution: sustainable management of natural resources
- Problem: different competences, lack of cooperation between sectors

https://twitter.com/kenyapics/status/822026455974449152

CapNex - Background

The Water-Energy-Food security (WEF) Nexus approach

(Theoretical) Framework, which

- "highlights the interdependencies between achieving water, energy and food security for human well-being"
- "is a fundamental shift, from a pure sectoral approach towards a cross-sectoral, coherent and integrated perspective"
- "the […] resources land, water and energy are part of [an] ecosystem and must be used and protected in a balanced manner"

Results of the conference „The Water, Energy and Food Security Nexus – Solutions for the Green Economy" (Bonn 2011 conference) https://www.water-energy-food.org/start/

CapNex – Project partners

TECHNISCHE UNIVERSITÄT WIEN Vienna | Austria — Institute for Water Quality, Resource and Waste Management (IWR) (Coordinator)

Makerere University College of Agricultural and Environmental Sciences — College of Agricultural and Environmental Sciences (CAES)

TECHNICAL UNIVERSITY OF KENYA — Department of Biosystems and Environmental Engineering (BEE)

Universität für Bodenkultur Wien, University of Natural Resources and Applied Life Sciences, Vienna — Institute of Water Management, Hydrology and Hydraulic Engineering (IWHW)

CapNex – Project partners

TECHNISCHE UNIVERSITÄT WIEN

Institute for Water Quality, Resource and Waste Management (IWR) (Coordinator)

Universität für Bodenkultur Wien, University of Natural Resources and Applied Life Sciences, Vienna

Institute of Water Management, Hydrology and Hydraulic Engineering (IWHW)

CapNex – Project objectives

i. Establishing the scientific foundations (approaches, methods, datasets, information, knowledge) for capacity building on the WEF nexus in Kenya and Uganda

ii. Building up capacities for WEF nexus investigation, as well as interpretation and communication (stakeholder dialogue) of research results at university levels in Kenya and Uganda

iii. Building up capacities among stakeholders on various geographical, institutional, and political-administrative levels for WEF nexus guided decision making and mitigation measures

CapNex – Case study area

Sio-Malaba-Malakisi River Basin in Kenya and Uganda

CapNex – Case study area

Sio-

CapNex – Case studies

- **Case study A: Water use and allocation, Water demand of crops, Satellite precipitation products (hydrology, water quantity)**
  - Formulated by BOKU
  - Carried out by BOKU and TU Kenya
- **Case study B: Erosion (erosion models, nutrient flows, water quality)**
  - Formulated by TU Kenya
  - Carried out by TU Kenya and BOKU (support from TU Wien and Makerere)
- **Case study C: Soil and water conservation measures in agriculture**
  - Formulated by Makerere
  - Carried out by Makerere (support from TU Kenya and TU Wien)
- **Case study D: Waste biomass (manure) for food or energy production**
  - Formulated by Makerere
  - Carried out by Makerere and TU Wien
- **Framework and integration of case study results (TU Wien + all)**

CapNex – Capacity building at University level

- **Case study A: Water use and allocation, Water demand of crops, Satellite precipitation products (hydrology, water quantity)**
  - 1 Master and 1 Bachelor student (BOKU)
  - 1 PhD student (BOKU / TU Kenya)
- **Case study B: Erosion (erosion models, nutrient flows, water quality)**
  - 2 PhD students (TU Kenya)
  - 1 PhD student (TU Kenya / BOKU)
- **Case study C: Soil and water conservation measures in agriculture**
  - 2 Master students (Makerere)
- **Case study D: Waste biomass (manure) for food or energy production**
  - 2 Master students (Makerere)
- **Framework and integration of case study results (TU Wien + all)**
  - 1 Master and 2 PhD students (TU Wien)


Thank you for your attention!

Short course

Soil Erosion Risk Modelling with R

Introduction to R and RStudio

Mathew Herrnegger & Christoph Schürz

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

What is R?

Software-Package, programming language and -environment

- Open Source implementation of the language S
- Available for different platforms: UNIX, Windows, MacOS
- Initially (1994) developed by *Ross Ihaka* and *Robert Gentleman* at the University of Auckland
- First non-Beta-Version R 1.0.0 published in 2000. Every year new versions available - current version: 3.5.2.
- Maintenance and development through a worldwide group of volunteers from the research community, but also from the economical private and industrial sector
- Primary distribution via websites (www.r-project.org) und archives (cran.rproject.org)

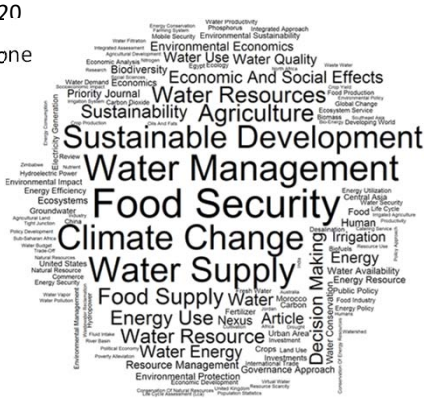
What is R?

Software-Package, programming language and -environment

- Mathematical analyses and calculations
- Visualisation of data
- Functionality easily extendable through external packages
- Due to large data amounts, processing and analyses of (scientific) results is frequently not possible in classical spreadsheet-programmes (e.g. Excel)

Why R?

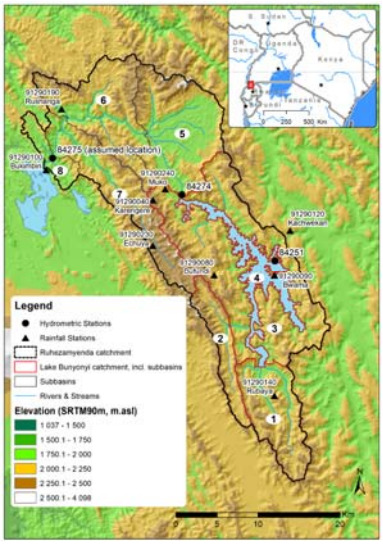
- **It is cool ☺**
- **Very powerful, open-source, many users worldwide**
- **Significant help online (your problem has probably been solved before)**
- **Functionality can be massively enhanced by external Libraries and Packages**
  - For Windows, on CRAN (Comprehensive R Archive Network), libraries 13120 (October 2018) are available, up from 11720 in November 2017 (+12% in one year!)



Example: Word cloud of keywords associated with the Water-Energy-Food Security Nexus in scientific literature generated with the *package „wordcloud“*

Why R?

✓ **Example: Climate change impacts on hydrology study**

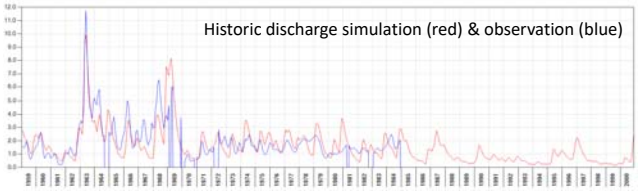


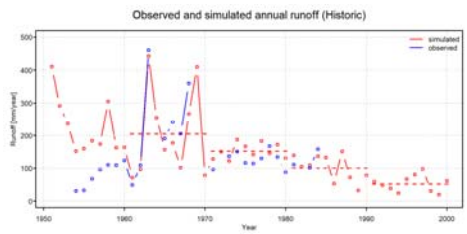
- Ruhezamyenda catchment in the south-west of Uganda
- Total area 722 km², of which 381 km² belong to the Lake Bunyonyi catchment
- Mean annual Rainfall: 1074 mm/a

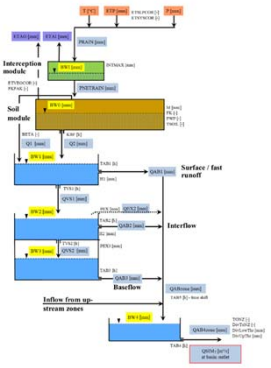
Why R?

✓ **Example: Climate change impacts on hydrology study**

Given a hydrological model calibrated with the historic (observed) climate data as input, the model must be run with an ensemble of precipitation and temperature inputs from different climate models for the future







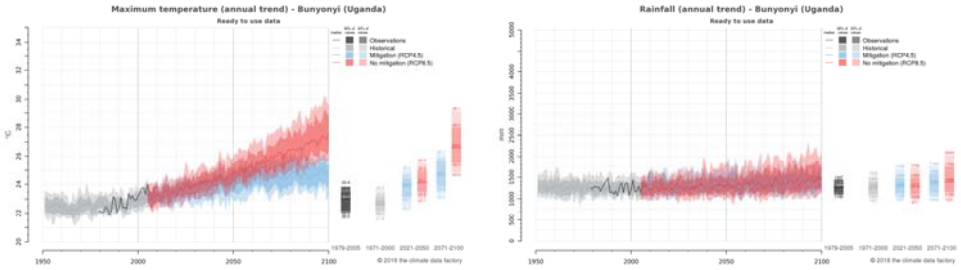
COSERO model structure

Why R?

✓ **Example: Climate change impacts on hydrology study**

Given a hydrological model calibrated with the historic (observed) climate data as input, the model must be run with an ensemble of precipitation and temperature inputs from different climate models for the future

- 22 times series of P & T for RCP4.5 and 30 for RCP8.5 covering 1950-2100 (**total 52 input data sets**)



Why R?

✓ **Example: Climate change impacts on hydrology study**

Given a hydrological model calibrated with the historic (observed) climate data as input, the model must be run with an ensemble of precipitation and temperature inputs from different climate models for the future

- 22 times series of P & T for RCP4.5 and 30 for RCP8.5 covering 1950-2100 (**total 52 input data sets**)

- Manual manipulation would be time-intensive (e.g. preparing inputs to a format, which are accepted by the model, calculate potential ET, analyse and visualize results etc.)
  - Use R to make things easier and more efficient and have time for something else!

Why R?

✓ **Example: Climate change impacts on hydrology study**

Use R to run the hydrological model 52 times, for every data set from the climate models

1. Read climate data
2. Calculate potential ET, here on the basis of Tmin & Tmax (52 times in total)
3. Prepare folder structure for single model run and prepare and save necessary data files (ET, P, T) (52 in total)
4. Run hydrological model for ensemble member i
5. Save outputs (i of 52 in total)
6. Analyse and visualize results

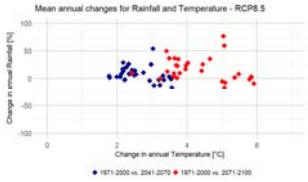
„do this 52 times“

Why R?

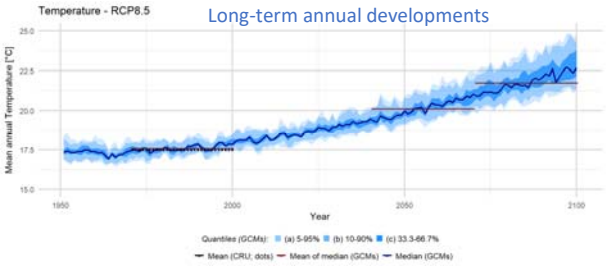
✓ **Example: Climate change impacts on hydrology study**

Visualize results

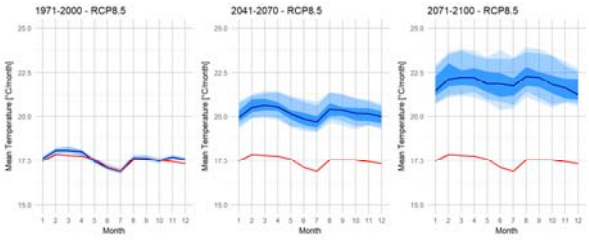
Long-term mean annual changes



Long-term annual developments



Long-term mean seasonal changes

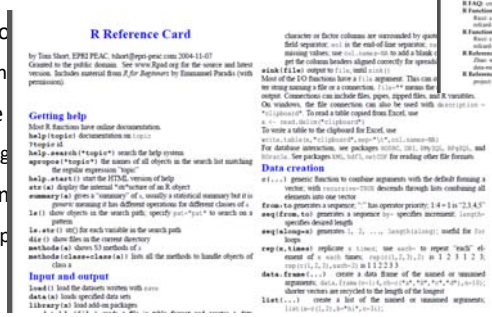


Help & Documentation

- Usually an appropriate help system is available
- Help functions in R via menu or by typing help()
- In RStudio: Pressing “F1” opens the help page, depending on the location of the curser
- Substantial online-documentation available (**google!**)
- Good starting point, e.g. R-Introductions, Short Documents and Reference Cards:

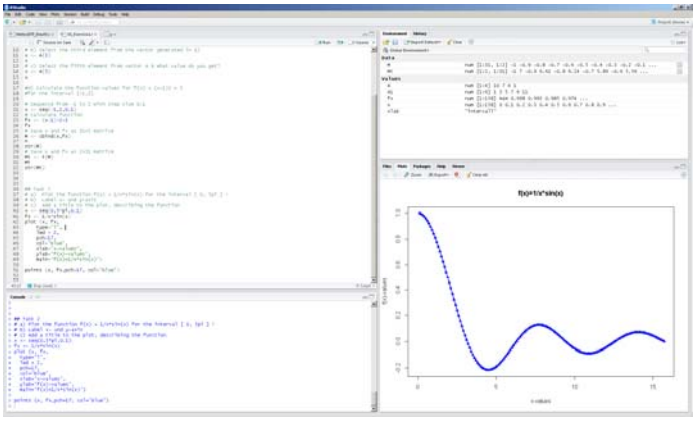
  http://cran.r-project.org/other-docs.html
- Literature provided:
  - „A (very) short introduction to R“ by P. Torfs & C. Brauer
    - Covers (among other things) installation of RStudio, data types, functions, examples, plotting etc.
  - “YaRrr! The Pirate’s Guide to R” by Nathaniel D. Phillips
    - Longer, but entertaining and amusing
  - “R Reference Card” by Tom Short + Other Cheatsheets
    - Good Cheat sheet and probably the most popular reference card

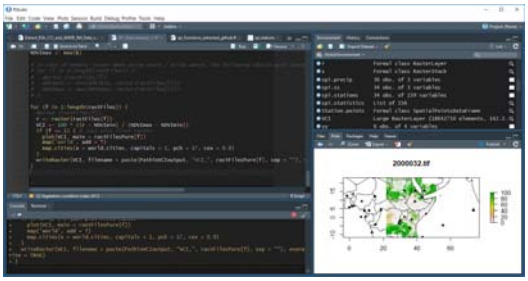
Help & Documentation



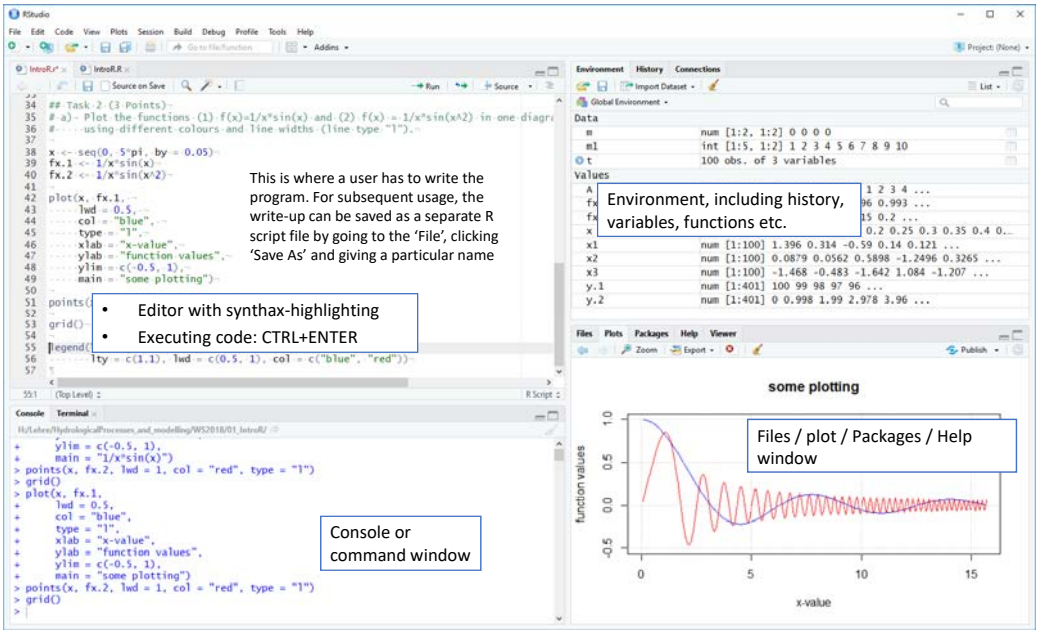
RStudio

- Integrated development environment for R, which must initially be installed on your system
- Graphical user interface with syntax highlighting
- Free and Open source, supporting Windows, MacOS, Linux
- Installation-package under http://www.rstudio.com/ or provided with the materials



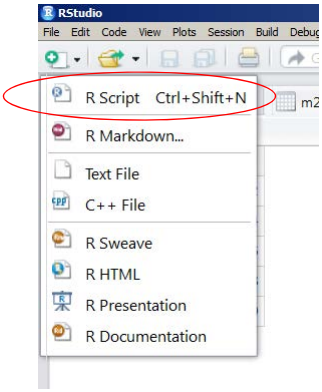


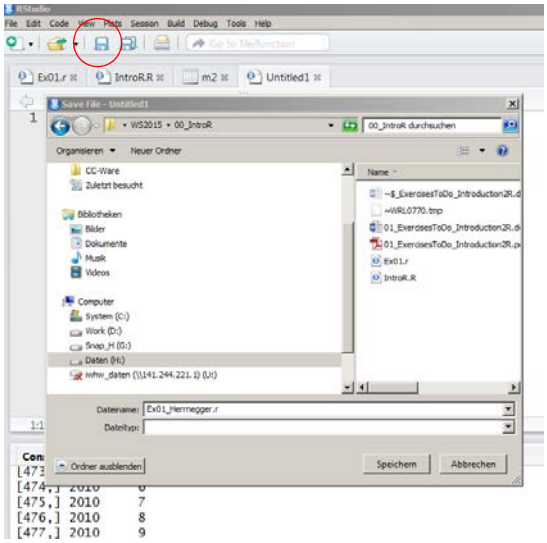
RStudio



Using the script window / Editor

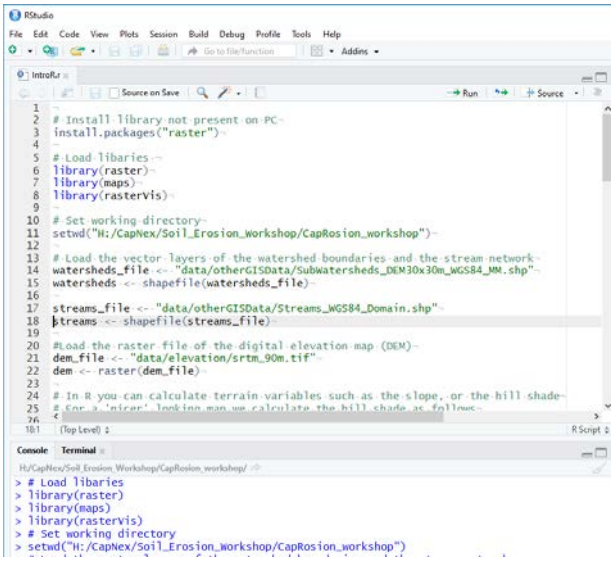
Start by opening a new R Script and saving it to an appropriate location





Using the script window / Editor

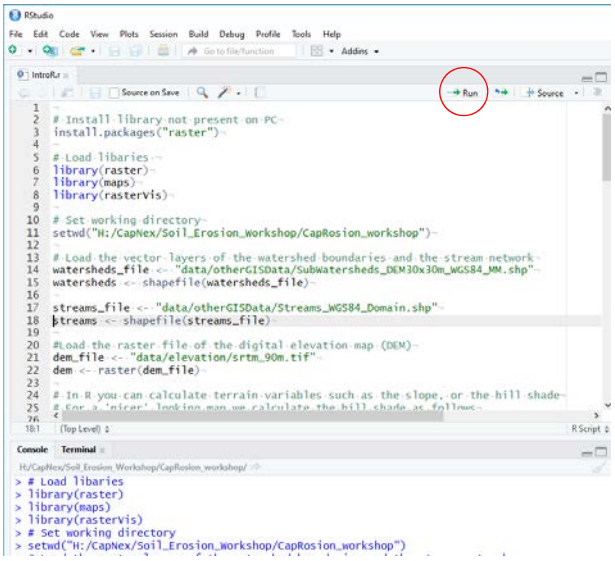
Structure the script, e.g. using hashtags (#) for commenting the code as shown



Be consequent with commenting all of your scripts. Take the time - it really helps YOU and others to understand, what you are doing!

Using the script window / Editor

Executing the code



- Select a line with the cursor (here Line 18) and press CTRL+Enter or press the Run-button
- If you select several lines and press CTRL+Enter, all this lines will be executed
- If you execute a code it will pop-up in the console

Console / Command window

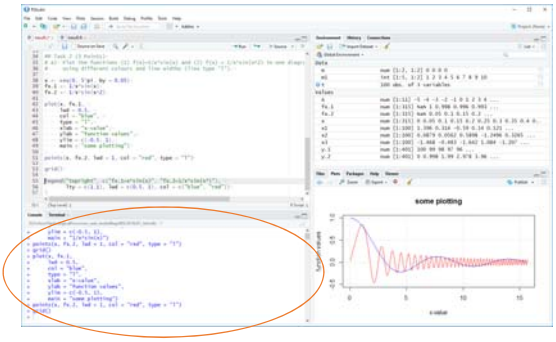
- Can also be used as a calculator, e.g.

```
> 10^2+36
[1] 136
>
```

- Arithmetic operations

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponent
()	Brackets

- Other functions
  - sin(), cos(), tan(),…
  - sqrt(), exp(), log(),…
  - plot(), mean(), sum() etc.



Exercise: Calculate the percentage of your life you have spent at university, i.e. the ratio between years spent at university and age multiplied by 100….

Variables / Data types & structures

- ***Variables*** in a computer program are analogous to "Buckets" , "Envelopes“ or “Containers”, where information can be maintained and referenced
- Variables can consist of different ***data types***:

Type	Example
character	"a", "BW0"
numeric	2, 15.5
integer	2
logical	TRUE, FALSE
complex	1+4i

- Different ***data structures*** exist in base R to store variables, e.g. *vectors, matrices, data frames or lists*.
- Data structure of *raster* needs external library

Vectors

Variables in R are generally defined via the assignment operator „<-“

```
> A <- 1 #scalar
> A
[1] 1
```

(Scalar, 1×1)

```
> A <- c(1, 3, 5, 7, 9, 11)
> A
[1]  1  3  5  7  9 11
```

c… concatenate

(Vector, 1×6)

```
> A <- 1:10
> A
 [1]  1  2  3  4  5  6  7  8  9 10
```

(Vector, 1×10)

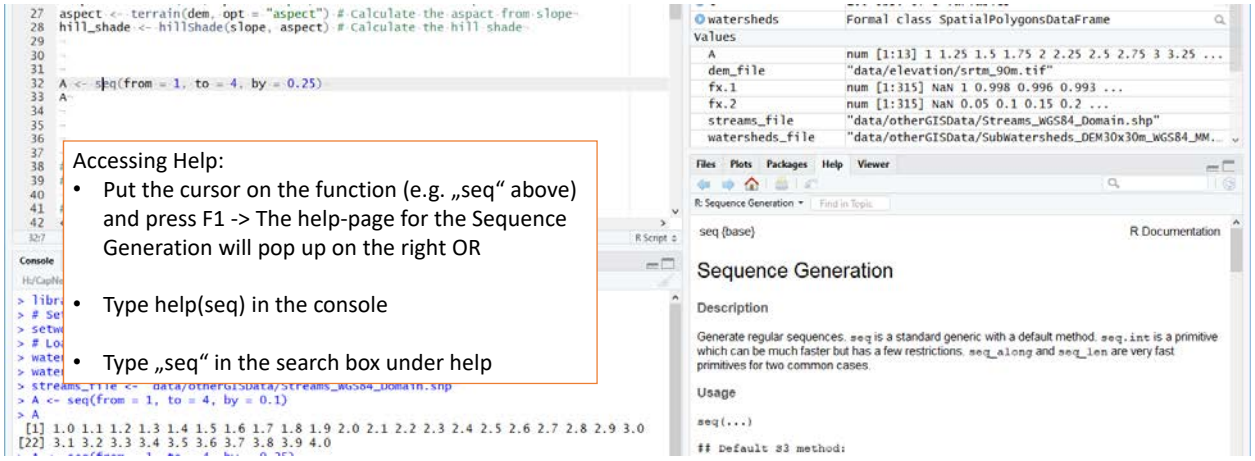
```
> A <- c("one", "two", "three", "4")
> A
[1] "one"   "two"   "three" "4"
> str(A)
 chr [1:4] "one" "two" "three" "4"
> class(A)
[1] "character"
```

(Vector, but this time consisting of characters – use str() and class() to check this)

Vectors

- Vector with a regular sequence: *seq*(from = …, to = …, by = …)

```
> A <- seq(from = -5, to = 5, by = 1)
> A
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```



Accessing Help:

- Put the cursor on the function (e.g. „seq“ above) and press F1 -> The help-page for the Sequence Generation will pop up on the right OR
- Type help(seq) in the console
- Type „seq“ in the search box under help

Vectors

- Vector with a regular sequence: *seq*(from = …, to = …, by = …)

```
> A <- seq(from = -5, to = 5, by = 1)
> A
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

- Vector with a defined repitition: *rep*(pattern, repitition)

```
> A <- rep(1:4, each=3)
> A
 [1] 1 1 1 2 2 2 3 3 3 4 4 4
>
> A <- rep(1:4, times=3)
> A
 [1] 1 2 3 4 1 2 3 4 1 2 3 4
```

- Retrieve length of vector: *length()*

```
> length(A)
[1] 12
```

Matrices

✓ **Matrices** (2-dimensional vector)

```
> m <- matrix(data=c(1,2,3,4), nrow=2, ncol=2)
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

	col_1	col_2
row_1	1	3
row_2	2	4

Using colnames() and rownames() for naming, e.g. colnames(m) <- c("col_1", "col_2")

```
> m <- matrix(0, nrow=2, ncol=2)
> m
     [,1] [,2]
[1,]    0    0
[2,]    0    0
```

(2x2 Matrix filled with „0“)

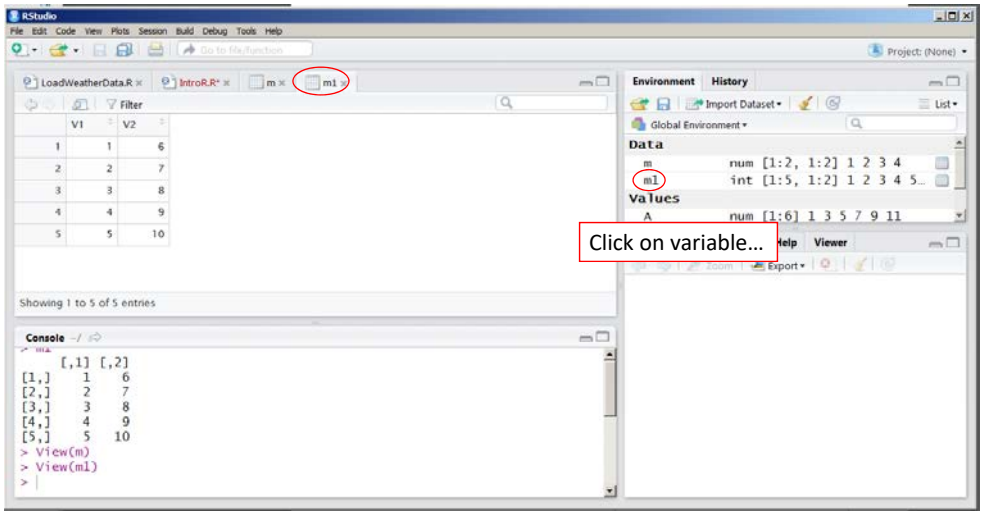
```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> m1 <- matrix(data=x, nrow=5, ncol=2)
> m1
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
```

(Vector, 1×10)

(Matrix, 5×2)

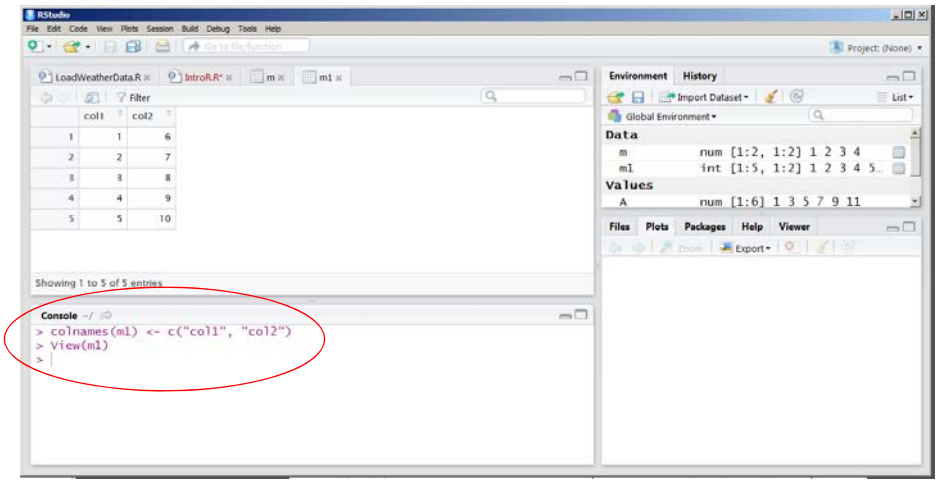
Matrices

✓ Displaying variables in RStudio (also applicable for other data structures)



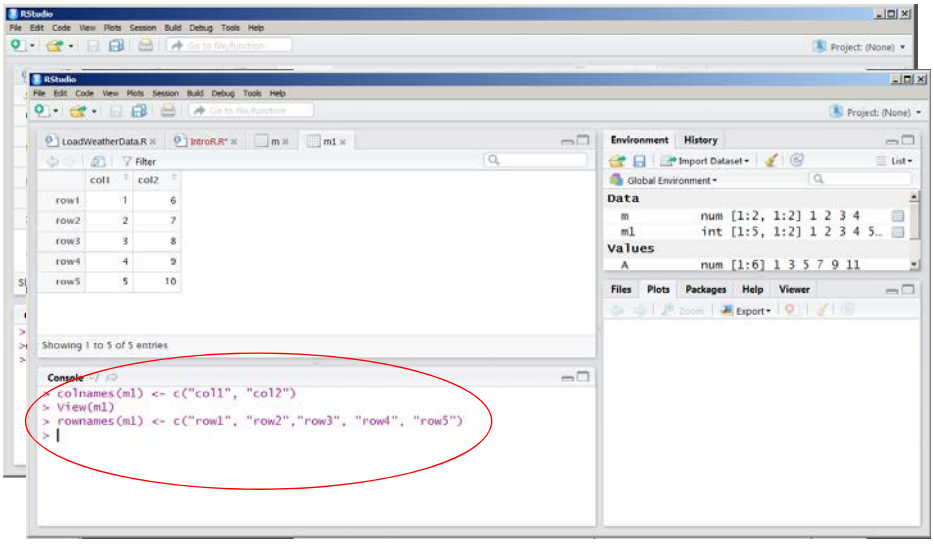
Matrices

✓ Definition of column and row names: *colnames*()



Matrices

✓ Definition of column and row names: *colnames*(); *rownames*()



Matrices

✓ Merging via *cbind*(); *rbind*()

```
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
>
> B
     [,1] [,2]
[1,]    5    7
[2,]    6    8
>
```

```
>
> cbind(A,B)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
>
> rbind(A,B)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    5    7
[4,]    6    8
>
```

Merge via column

Merge via rows

Matrices

✓ Arithmetic operations (e.g. +, -, *, /, ^) are performed „*element by element*“ (this is also true for other data structures, e.g. vectors!)

```
>
> Z
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
>
>
> Y
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
>
```

```
> Y + Z
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    6   10   14   18
[2,]    4    8   12   16   20
>
> Y - Z
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
>
> Y * Z
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    9   25   49   81
[2,]    4   16   36   64  100
>
> Y / Z
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    1    1    1    1    1
>
> Y ^ Z
     [,1] [,2]  [,3]     [,4]        [,5]
[1,]    1   27  3125   823543   387420489
[2,]    4  256 46656 16777216 10000000000
> .
```

Matrices

✓ **Matrices: some useful functions**

- *sum(), colSums(), rowSums()* — sums
- *mean(), colMeans(), rowMeans()* — mean values
- *nrow(), ncol()* — get number fo rows / columns
- *t()* — transpose

```
> z
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> colSums(z)
[1] 10 26 42 58
> colSums(z)[2]
[1] 26
> colMeans(z)
[1]  2.5  6.5 10.5 14.5
```

```
> y
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> t(y)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
> a <- 1:99
> a
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
[29] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
[57] 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
[85] 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
> sum(a)
[1] 4950
> mean(a)
[1] 50
> a[1:2] <- NA
> a
 [1] NA NA  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
[29] 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
[57] 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
[85] 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
> sum(a)
[1] NA
> sum(a, na.rm = TRUE)
[1] 4947
```

„na.rm = TRUE“ makes R ignore missing values (NAs)

Matrices

✓ **Extracting elements (also applicable to other data structures)**

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> m1 <- matrix(data=x, nrow=5, ncol=2)
> m1
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> m1[3,]
[1] 3 8
```

3. column

```
> m1[,2]
[1]  6  7  8  9 10
```

2. row

```
> m1[4,2]
[1] 9
```

Element in 4. row and 2. column

```
> m1[3:5,]
     [,1] [,2]
[1,]    3    8
[2,]    4    9
[3,]    5   10
```

All elements in 3. to 5. row

```
> m1[3:5,2]
[1]  8  9 10
```

All elements in 3. to 5. row, but 2. column

matrix[row, column]

Define via „row“ and „column“ within the square brackets, which elements are to be selected. Single values (e.g. 3, 4) or vectors (e.g. 1:4 or c(1, 5, 6, 9) can be used.

Matrices

✓ **Extracting elements (also applicable to other data structures)**

```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> m1 <- matrix(data=x, nrow=5, ncol=2)
> m1
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> m1[3,]
[1] 3 8
> m1[,2]
[1]  6  7  8  9 10
> m1[4,2]
[1] 9
> m1[3:5,]
     [,1] [,2]
[1,]    3    8
[2,]    4    9
[3,]    5   10
> m1[3:5,2]
[1]  8  9 10
```

3. column

2. row

Element in 4. row and 2. column

All elements in 3. to 5. row

All elements in 3. to 5. row, but 2. column

matrix[row, column]

Define via „row“ and „column“ within the square brackets, which elements are to be selected. Single values (e.g. 3, 4) or vectors (e.g. 1:4 or c(1, 5, 6, 9) can be used.

```
> m1[3:5,2] <- 999
> m1
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3  999
[4,]    4  999
[5,]    5  999
```

This selection method can also be used to replace elements within a data structure, using the assignment operator „<-“

Matrices

✓ **Extracting elements with conditions**

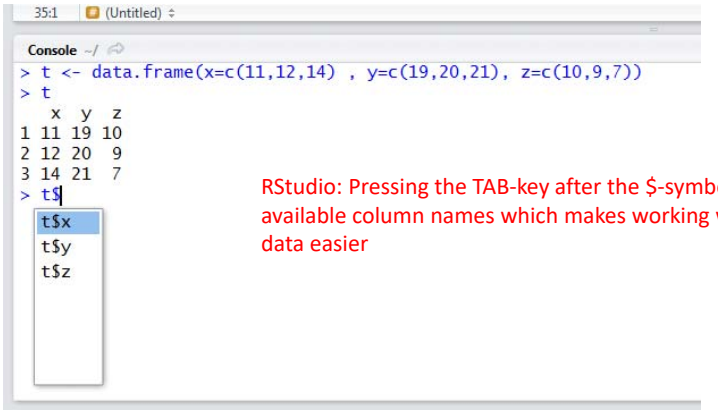
Console ~/

```
> t4
     [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    1    3
[4,]    2    4
[5,]    1    5
[6,]    2    6
> t4[t4[,1] == 1, ]
     [,1] [,2]
[1,]    1    1
[2,]    1    3
[3,]    1    5
```

Extract all elements of matrix t4, where the values in the first column *(i.e. t4[, 1])* equals the value 1

Data frames

✓ **Data frames: Matrices with column names, which can be called in RStudio**



RStudio: Pressing the TAB-key after the $-symbol shows the available column names which makes working with (large) data easier

Lists

✓ **Lists: Data are organized in columns, but must not be of the same length or data type**

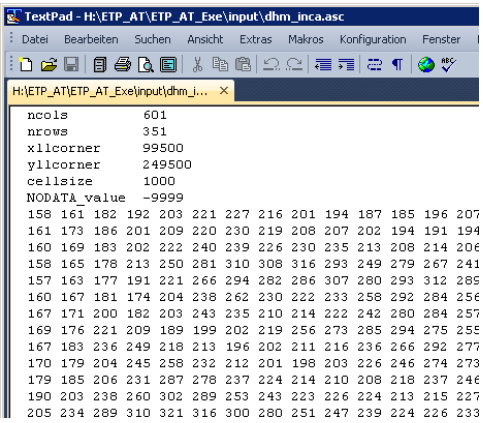
```
> b <- list(one = "one", sequence = 1:10, uniform = runif(10, min = 0, max = 1))
> b
$`one`
[1] "one"

$sequence
 [1]  1  2  3  4  5  6  7  8  9 10

$uniform
 [1] 0.56134205 0.30138145 0.21622966 0.04329151 0.35659288 0.28423329 0.63885751
 [8] 0.44049347 0.51252242 0.86052632

> b$sequence
 [1]  1  2  3  4  5  6  7  8  9 10
> b$uniform
 [1] 0.56134205 0.30138145 0.21622966 0.04329151 0.35659288 0.28423329 0.63885751
 [8] 0.44049347 0.51252242 0.86052632
> b[[3]]
 [1] 0.56134205 0.30138145 0.21622966 0.04329151 0.35659288 0.28423329 0.63885751
 [8] 0.44049347 0.51252242 0.86052632
```

Similar to data frames: Pressing the TAB-key after the $-symbol shows the names of the columns

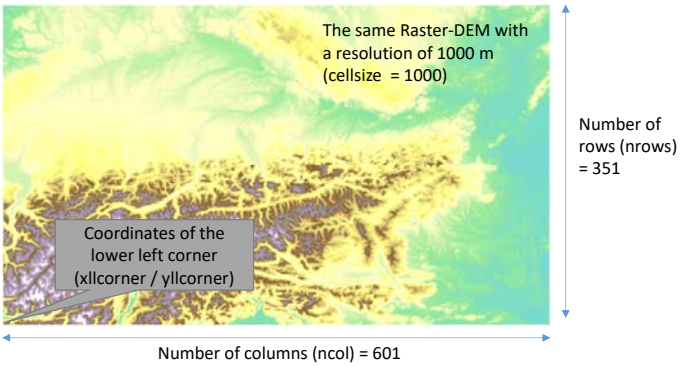
List elements can also be accessed via a double square bracket [[“position in list”]]

Rasters

- ✓ **Rasters are not a data structure, which are included in base R, but depend on external libraries**
  - In case the library is not available on the PC, it must initially be installed: install.packages("raster")
  - Then load the package “Raster” to read and manipulate rasters: library(raster)



Example of a Raster-DEM stored as an „ESRI ASCII GRID“ viewed in a standard text editor



Rasters

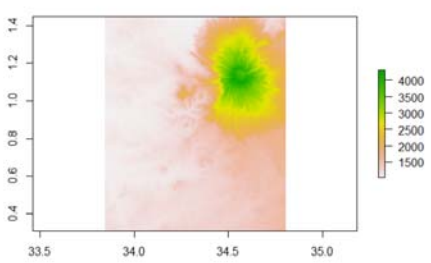
- ✓ **Rasters are not a data structure, which are included in base R, but depend on external libraries**
  - In case the library is not available on the PC, it must initially be installed: install.packages("raster")
  - Then load the package “Raster” to read and manipulate rasters: library(raster)
  - Loading a raster file into a variable: dem <- raster(path_to_file)

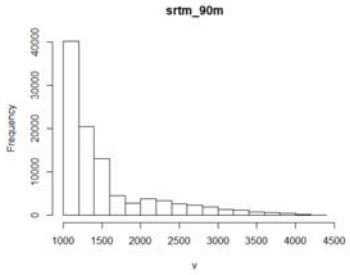
```
> # Install library not present on PC
> #install.packages("raster")
>
> # Load libary
> library(raster)
>
> #Load the raster file of the digital elevation map (DEM)
> dem_file <- "H:/CapNex/Soil_Erosion_Workshop/CapRosion_workshop/data/elevation/srtm_90m.tif"
> dem <- raster(dem_file)
> dem
class       : RasterLayer
dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
resolution  : 0.0008333333, 0.0008333333  (x, y)
extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : H:\CapNex\Soil_Erosion_Workshop\CapRosion_workshop\data\elevation\srtm_90m.tif
names       : srtm_90m
values      : 1045, 4314  (min, max)
```

Information to raster

Rasters

- ✓ **Rasters are not a data structure, which are included in base R, but depend on external libraries**
  - In case the library is not available on the PC, it must initially be installed: install.packages("raster")
  - Then load the package “Raster” to read and manipulate rasters: library(raster)
  - Loading a raster file into a variable: dem <- raster(path_to_file)
  - plot(variable_name) plots the raster…
  - hist(variable_name_of_raster) shows you the distribution of the data in the raster





User-defined functions in R

- Corresponds to the existing functions in R (e.g. plot(), mean(), etc.) or subroutines in other programming languages
- Makes sense for frequently used calculations or routines
- After the function is defined, it is globally available within the session

Syntax of user-defined functions

```
function.name <- function (arg1, arg2,…) {
        Purpose of function, i.e. computations
        involving arg1, arg2, etc.
        return(object)
}
```

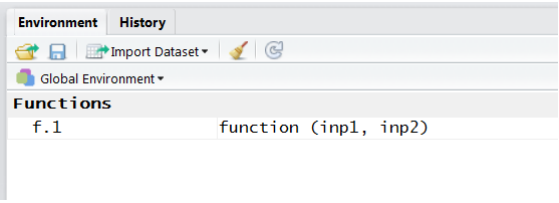
- The assignment operator „<-“ is needed to tell R, in what object (in the above case “function.name” to put the function
- Key word „function“ tells R, that what comes next is a function
- The parentheses after „function“ form the „argument list“ of your function (=input into function (arg1, arg2 in the above case))
- Everything between the braces, {}, is the body of your function and includes the computations involving the argument list

Example of user-defined functions

Function with multiple tasks and returning multiple results in a

list with named elements

```
> f.1 <- function(inp1, inp2) {
+   z1 <- inp1 + inp2
+   z2 <- inp1 + 2*inp2
+   return(list(result1 = z1, result2 = z2))
+ }
```



➢ Running the code of the function (select and press CTRL+Enter) -> function f.1 becomes available in the RStudio Environment

Example of user-defined functions

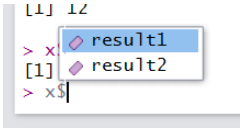
Function with multiple tasks and returning multiple results in a

list with named elements

- Calling the function with input arguments inp1 = 2 and inp2 = 5

```
> f.1(2,5)
$result1
[1] 7

$result2
[1] 12

>
> x <- f.1(2,5)
> x
$result1
[1] 7

$result2
[1] 12

> x$result1
[1] 7
```

```
> f.1 <- function(inp1, inp2) {
+   z1 <- inp1 + inp2
+   z2 <- inp1 + 2*inp2
+   return(list(result1 = z1, result2 = z2))
+ }
```



- Since results are stored in a list, pressing the *TAB button* gives a preview of the single list elements

Programming tools

✓ **For-loop**

- For calculations with a similar pattern, which are to be performed more often

```
for (loop.object in loop.vector) {
      LOOP.CODE
}
```

- Loop.object: The object that will change for each iteration of the loop, e.g. variable "i"
- Loop.vector: A vector specifying all values that the loop object will take over the loop, e.g. a:b (e.g. 1:10), seq(), or c(…)
- LOOP.CODE: The code between braces will be executed for all values in the loop.vector.

Programming tools

- ✓ **For-loop**
  - ➢ For calculations with a similar pattern, which are to be performed more often

```
for (loop.object in loop.vector) {
    LOOP.CODE
}
```

```
> for(i in 1:10) {
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

```
> for(i in c(1,5,7,99)) {
+   print(paste("Number is", i))
+ }
[1] "Number is 1"
[1] "Number is 5"
[1] "Number is 7"
[1] "Number is 99"
```

```
> # Loop to add integers from 1 to 100
> current.sum <- 0 # The starting value of current.sum
> for(i in 1:100) {
+   current.sum <- current.sum + i # Add i to current.sum
+ }
> current.sum # Print the result!
[1] 5050
```

Programming tools

- ✓ **For-loop**
  - ➢ For calculations with a similar pattern, which are to be performed more often

```
for (loop.object in loop.vector) {
    LOOP.CODE
}
```

```
> h <- 6:9
> h
[1] 6 7 8 9
> r <- rep(NA, length(h))
> r
[1] NA NA NA NA
> for (i in c(2,3)) {
+   r[i] <- h[i]*100
+ }
> r
[1]  NA 700 800  NA
```


Thank you for your attention!

1 - Input data generation for the RUSLE

Christoph Schuerz and Mathew Herrnegger
Institute for Hydrology and Water Management (HyWa), University of Natural Resources and Life Sciences (BOKU), Vienna, Austria

---

This part of the course covers the generation of the input layers for the Revised Universal Soil Loss Equation (RUSLE). We will look at their calculation and potential data sources for generating the layers. While learning different methods to calculate the RUSLE inputs, the participants will also, step by step, learn new R features and functions to perform the required GIS operations in R.

Contents

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

Introduction

The Revised Universal Soil Loss Equation RUSLE

The *RUSLE* is widely used by soil conservationists in the world to estimate the erosion by water. The the inital Universal Soil Loss Equation (USLE) was developed by W. H. Wischmeier, D. D. Smith, and others with the U.S. Department of Agriculture (USDA), Agricultural Research Service (ARS), Soil Conservation Service (SCS), and Purdue University in the late 1950s. Its field use began in the Midwest of the U.S. in the 1960s. In 1987, ARS, SCS, and several cooperators began a project to revise the USLE to form the *RUSLE*. This empirically based equation, derived from a large mass of field data, computes sheet and rill erosion using values representing the major factors affecting erosion [Renard et al., 1991]. The *RUSLE* estimates the long-term average annual soil loss (A in $t \cdot ha^{-1} \cdot yr^{-1}$) by multiplying these spatially distributed factors that represent:

- the erosivity of the rainfall R in $MJ \cdot mm \cdot ha^{-1} \cdot h^{-1} \cdot yr^{-1}$
- the erodibility of the soil K in $t \cdot ha \cdot h \cdot ha^{-1} \cdot MJ^{-1} \cdot mm^{-1}$
- the topography expressed by the slope length and steepness LS as unitless factor
- the coverage of the soil by plants C as unitless factor, and
- the support practices P as unitless factor

Wise use of prediction technology like the USLE/RUSLE requires that the user be aware of a procedure's limitations. The USLE/RUSLE is an equation that estimates average annual soil loss by sheet and rill erosion on those portions of landscape profiles where erosion, but not deposition, is occurring. It does **not estimate deposition** like that at the toe of concave slopes, and it does **not estimate sediment yield** at a downstream location. Also, it does **not include ephemeral gully erosion**. Furthermore, the USLE/RUSLE does not provide information on sediment characteristics, such as those needed in many water quality initiatives.

An important scientific limitation of the USLE/RUSLE as an empirically based equation is that it does not represent fundamental hydrologic and erosion processes explicitly. For example, the effect of runoff, as might be reflected in a hydrologic model, is not represented directly in the USLE/RUSLE. Fundamental erosion processes and their interactions are not represented explicitly. An example where the USLE does not give the proper result is the deposition of sediment in furrows on flat grades. Analysis of any single data set may show significant differences between estimates with the USLE/RUSLE and observed data. Such limited comparisons are not at all an indication of the overall performance of the USLE/RUSLE. As an empirical equation derived from experimental data, the USLE/RUSLE adequately represents the first-order effects of the factors that affect sheet and rill erosion [Renard et al., 1991, 1997].

The soil loss equation is written as follows:

$$A = R \cdot K \cdot LS \cdot C \cdot P$$

One of the first tasks in any erosion study, where the *RUSLE* is implemented, is to calculate the five spatially distributed model inputs for the study area of interest. From the literature many methods are available to calculate these inputs, often implementing freely available data products. Thus, this is a major advantage for a general assessment of the erosion risk in data scarce regions.

Goal of this course section

In this part of the course we present and/or mention different methods to calculate two different realizations for each of the RULSE inputs. Three realizations will be calculated to demonstrate how to perform simple raster operations in R. For other RUSLE input realizations, we provide sources that provide further information on how to calculate these inputs.

The study area for which we demonstrate the input data generation (and erosion risk assessment in general) covers the river basins of the Malaba and the Malakisi rivers. These basins are located in the south of Mount Elgon at the border between Kenya and Uganda. Several erosion studies exist for the region, mainly however for the Ugandan part of Mt. Elgon and not covering larger areas [see e.g. Bamutaze, 2010, Jiang et al., 2014, Semalulu et al., 2014].

The key outcomes of this course section are in summary:

- Learn different methods to generate the inputs of the *RUSLE*
- Get familiar with basic *R* commands in a project context
- Learn basic GIS raster (and vector) operations in *R*

R Packages

A key benefit of R is the large number of R packages that are available and support the R user in a wide range of tasks. If the computer has an Internet connection, R packages are easily installed using the following commands:

```
# Install packages from CRAN
install.packages("Name_of_the_package")

# Load packages. After that, R functions from the package can be used.
library(Name_of_the_package)
```

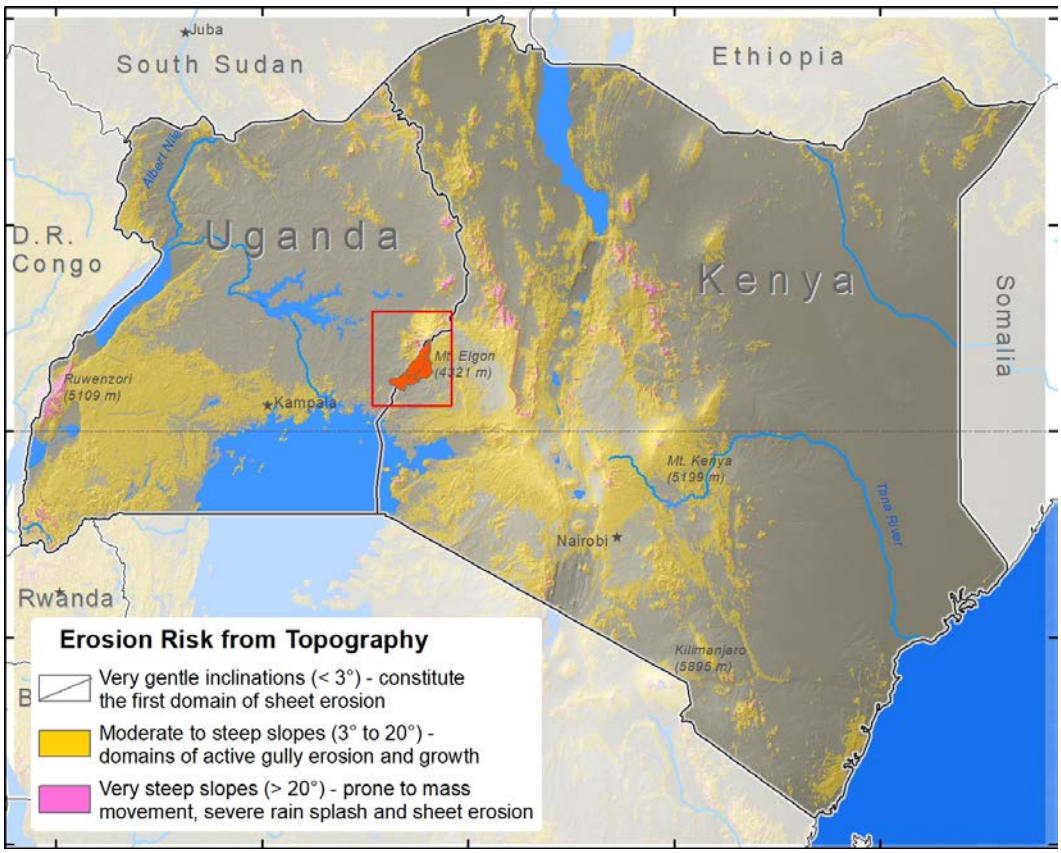
Apart from installing packages using the above commands, it is also possible to install them via the "Packages"-tab in Rstudio with a GUI.

For this course we installed the R packages needed for our analyses in a different way, as it may have caused troubles when a whole class installs R packages over a slow Internet connection. R packages have to be installed once and can then be called for any further analysis. Thus, we will skip the package installation here. It is however good practice to load all the R packages that are needed in the analyses performed in the R script at the beginning of the script. Therefore we will load all R packages we are going to use in this part here:

```
library(raster)
library(maps)
library(rasterVis)
```

Study area

The analysis will be performed for the river basins of the Malaba and the Malakisi, which are located on the southern side of Mount Elgon in the border region of Kenya and Uganda. The basins cover a total area of 1524 km^2. The location of the study area and the domain we will analyse (red box) is shown in the following figure.



The map above was generated in a classical GIS software with big effort. In contrast, getting a fast (but admittedly rather rough) overview of the study area can be easily done in R. Therefore and in a first step, we will plot (i) the location of the study area in an overview map and (ii) a more detailed illustration of the river basins together with the elevation of the region. This gives you a first idea how simple vector and raster operations in R work.

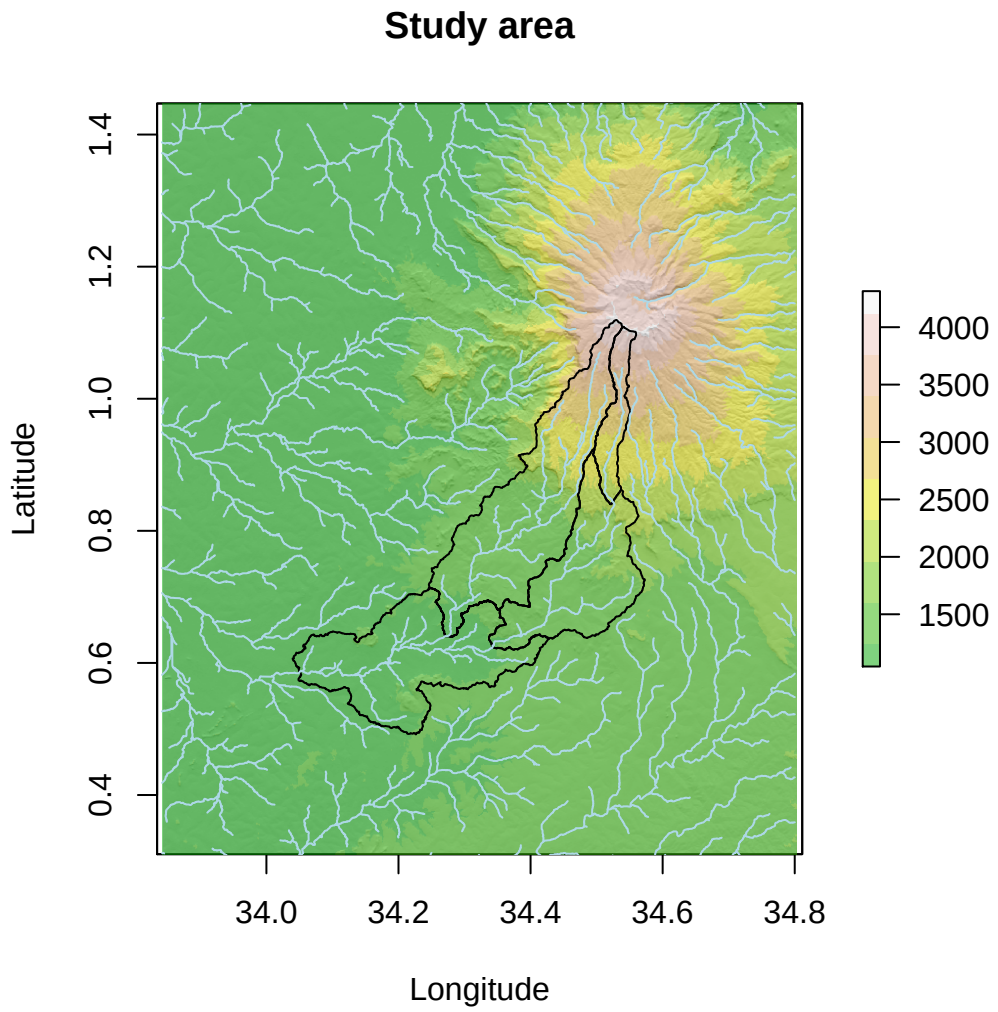
```
# To use relative paths for loading data, one can set the working directory
# with 'setwd()'. Generally we do not recommend to set the working directory
# but use the absolute file paths. Here however it improves the readability.
# You should replace the path below with the path you saved your data.
setwd("Define:/the/path/to/the/folder/on/your/computer")

# Load the vector layers of the watershed boundaries and the stream network
watersheds_file <- "data/otherGISData/SubWatersheds_DEM30x30m_WGS84_MM.shp"
watersheds <- shapefile(watersheds_file)

streams_file <- "data/otherGISData/Streams_WGS84_Domain.shp"
```

```
streams <- shapefile(streams_file)

#Load the raster file of the digital elevation map (DEM)
dem_file <- "data/elevation/srtm_90m.tif"
dem <- raster(dem_file)

# In R you can calculate terrain variables such as the slope, or the hill shade
# For a 'nicer' looking map we calculate the hill shade as follows
slope <- terrain(dem, opt = "slope")   # Calculate the slop from the DEM
aspect <- terrain(dem, opt = "aspect") # Calculate the aspact from slope
hill_shade <- hillShade(slope, aspect) # Calculate the hill shade

# To have the hillshade available for other plots we save the raster as a .tif
writeRaster(hill_shade, "data/otherGISData/hillshade.tif")
```

```
# (i) Plotting an overview map
# We will plot country boundaries using the function map from the 'maps'
# package, thereby defining the bounding box with xlim and ylim of the area to
# be mapped. To highlight the countries of Kenya and Uganda, we will then fill
# them with a slightly darker grey colour ("grey80").
e_africa <- map("world", xlim = c(27.5,42), ylim = c(-4.8,5.5), fill = TRUE,
                col = "grey95")

map(e_africa, c("Kenya", "Uganda"), fill = TRUE, col = "grey80", add = TRUE)

# The 'maps' package also includes the lakes of the world.
# We add them to the plot by saying add = TRUE.
# We fill them with the color 'lightblue'
map("lakes", add = TRUE, fill = TRUE, col = "lightblue")

# add our watershed boundaries and color them 'red'
plot(watersheds, add = TRUE, col = "red")

# Finally we add a title to the plot
title("Overview of study area")
```

Overview of study area

```
# Plotting the detailed map of the study region
# As a base layer we plot the calculated hill shade in gray colors.
# We do not want to plot the legend (legend = FALSE)
plot(hill_shade, col = grey(0:100/100), legend = FALSE,
     xlab = "Longitude", ylab = "Latitude", main = "Study area")

# We add the DEM raster we loaded above, again using 'plot'.
# As a color scheme 10 terrain.colors were used. Other available color
# schemes would be e.g. heat.colors(), rainbow() etc... "alpha = 0.5" is used
# to define some transparency of the DEM so that the hill shade remains visible.
plot(dem, col = terrain.colors(10, alpha = 0.5), add = TRUE)

# We add watershed boundaries. Lines is equal to plot and adds automatically
lines(watersheds, lwd = 1)
# Finally we add the stream network again using lines.
lines(streams, col = "lightblue")
```


Study area
Latitude
1.4
1.2
1.0
0.8
0.6
0.4
34.0
34.2
34.4
34.6
34.8
Longitude
4000
3500
3000
2500
2000
1500

The RUSLE input layers

We will calculate realizations for the RUSLE inputs R, K, and C (one each). The other realizations of the model inputs that will be used in later examples are explained briefly in this section. We will also provide links to acquire data that is useful for the calculation of RUSLE inputs.

Rainfall erosivity R

A common method to determine the rainfall erosivity is to infer R from the long-term mean annual precipitation. Many different relationships between R and the annual precipitation sums have been established over the years [e.g. Moore, 1979, Renard and Freimund, 1994, Yu and Rosewell, 1996]. The rainfall erosivity is mostly affected by high intensity short term precipitation events. These relationships between R and long-term annual precipitation therefore assume, that the annual precipitation is correlated to the strong rainfall events. Also, the relationships were usually established for specific regions in the world and should be used with care when transferred to other regions. Recent work by Panagos et al. [2017] derived global rainfall erosivities (GloREDa data set) from over 6000 global high temporal rainfall station records. This product is also freely available online from ESDAC-JRC. This data set was prepared for the study domain of the course and will be used later. In this session we will calculate R applying the method of Moore [1979]. For the long-term annual average precipitation the WorldClim2.0 data product [Fick and Hijmans, 2017] was used that is available on a 30 arcsecond resolution and freely available online from worldclim.org.

Calculation of R using the method of Moore [1979]

We implement this method here as it was derived from station data in East Africa. Moore [1979] suggests a differentiation of Coastal zones, low lands and the plateaus. For simplicity we here however use the relationship of Moore that does not differentiate between these regions:

```
# The equation of Moore
# We implement the equation of Moore as a function in R
calculate_r_moore <- function(p) {
  ke <- 11.46*p - 2226
  r <- 0.029*ke - 26
  r_si <- 17.02*r # Conversion from imperial to SI units
  return(r_si)
}
```

Short introduction to user-defined functions in R

We implemented the Moore-equation using a user-defined function. This function basically corresponds to other existing functions in R (e.g. `plot(...)` shown above) or subroutines in other programming languages. Using functions makes sense for frequently needed calculations or routines. In our case, for example, we may want to use different precipitation data sets. In this case we could simply apply the function to calculte the rainfall erosivity after Moore more often, but using different inputs into the function. Also, using functions help to keep a code clean and tidy. The general syntax of functions in R is:

```
function.name <- function(input1, input2, ...) {
  purpose of function, i.e. computations involving input1 and input2 etc.
  return(object)
}
```

The assignment operator <- is needed to tell R, in what object (in the above case "function.name") to save the function. The key word `function` tells R, that what comes next is a function. The parentheses after `function` form the "argument list" of your function (=inputs into function). Everything between the braces, {}, is the body of your function and includes the computations involving the argument list. Finally, the `return(...)` statement tells R, what result to give back from the function.

The precipation data derived from WorldClim 2.0 was already prepared for this course and is available in the data folder

```
# The precipitation file path on the hard drive
precip_file <- "data/climate/wc2_0_precip_sum.tif"
#Loading the precipitation data as a raster in R
precip <- raster(precip_file)
```

The *R* factor can now be calculated from the precipitation using the R function `calc()`:

```
r_moore <- calc(x = precip, fun = calculate_r_moore)
```

In our case we need to call our user-defined function `calculate_r_moore()` by using the function `calc()` from the raster package. This is a special case, since the input data we use is a raster.

Exploring the calculated R layer

It is very important and crucial to evaluate the performed analyses and calculation steps. **Many things can go wrong!** Here is a small summary of operations that can be done to explore our spatial data.

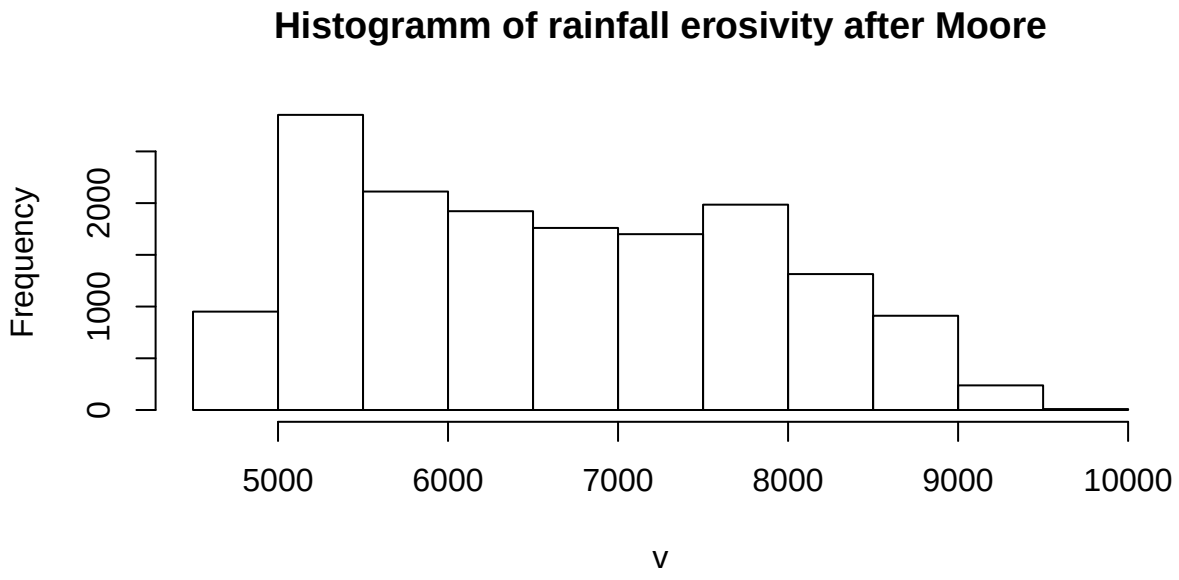
A first check of the data can be done with `summary()`. Here we can see if unusually large values, too many NA values, or not plausible quantile values are visible that indicate errors in the calculation:

```
summary(r_moore)
```

```
##            layer
## Min.    4533.773
## 1st Qu. 5526.476
## Median  6507.866
## 3rd Qu. 7644.808
## Max.    9658.496
## NA's       0.000
```

A histogram shows the distribution of the calculated values and provides a good overall picture of the data:

```
hist(r_moore, main = "Histogramm of rainfall erosivity after Moore")
```



Writing a raster to the hard drive

The calculated raster files can be stored at the hard drive in many different ways. To easily use it with other software as well (e.g. load the raster in any GIS (QGIS, ArcGIS)), saving the file as a GeoTiff is recommended. We save our new layer `r_moore` as a '.tif' using the function `writeRaster()`:

```
writeRaster(r_moore, filename = "data/rusle_input/r_factor/r_moore.tif",
            overwrite = TRUE)
```

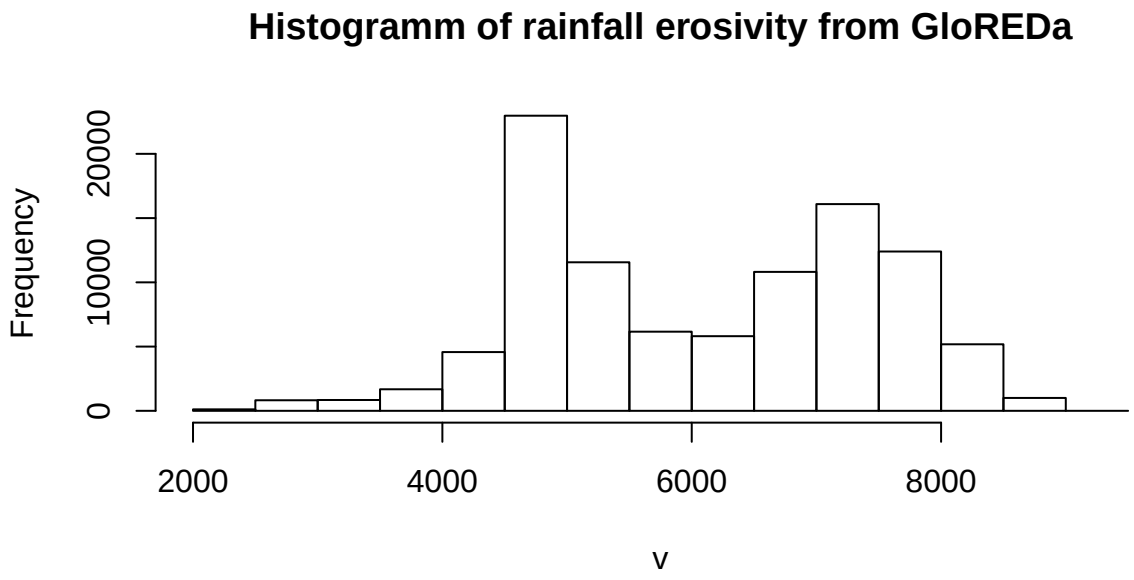
Comparison of Moore and GloREDa

To evaluate the plausibility of the calculated layer a comparison of other data sets is good practice. For the course we also prepared the GloREDa data set [Panagos et al., 2017] that is derived from high temporal precipitation measurements.

```
# File path of the GloREDa data set
r_gloreda_file <- "data/rusle_input/r_factor/r_gloreda.tif"

# Loading the GloREDa data set with raster
r_gloreda <- raster(r_gloreda_file)
```

To compare the two resulting histograms, we also plot the distribution of the GloREDa data set:

```
hist(r_gloreda, main = "Histogramm of rainfall erosivity from GloREDa")
```

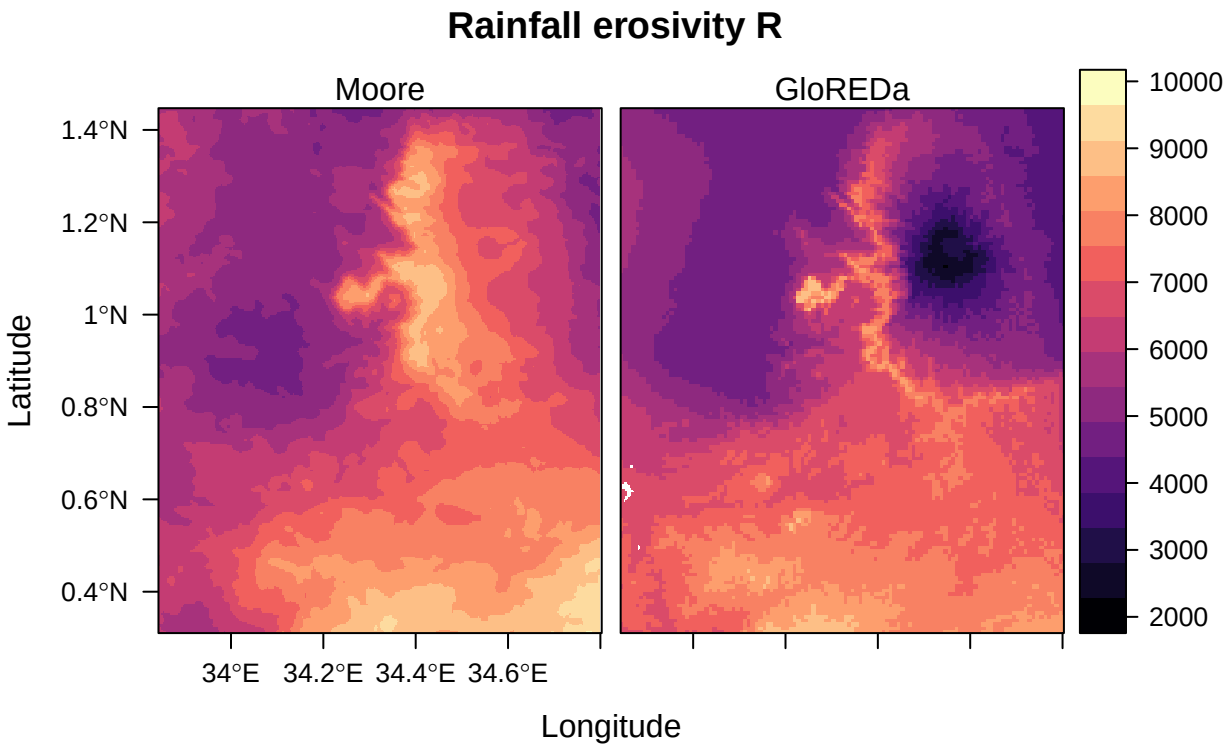


To visualize both layers in a combined plot we add them together to a "raster stack" using the function `stack()`. The problem with these two layers is that, although they have the same coordinate system and grid cell size, their raster cells do not match. Therefore it is necessary to project one raster based on the other raster before continuing. The 'raster' package provides the function `projectRaster()` to perform this operation:

```
# Project the raster r_moore
r_moore <- projectRaster(from = r_moore, to = r_gloreda)

# Create a combined raster stack for the visualization below.
r_layer <- stack(list(Moore = r_moore, GloREDa = r_gloreda))
```

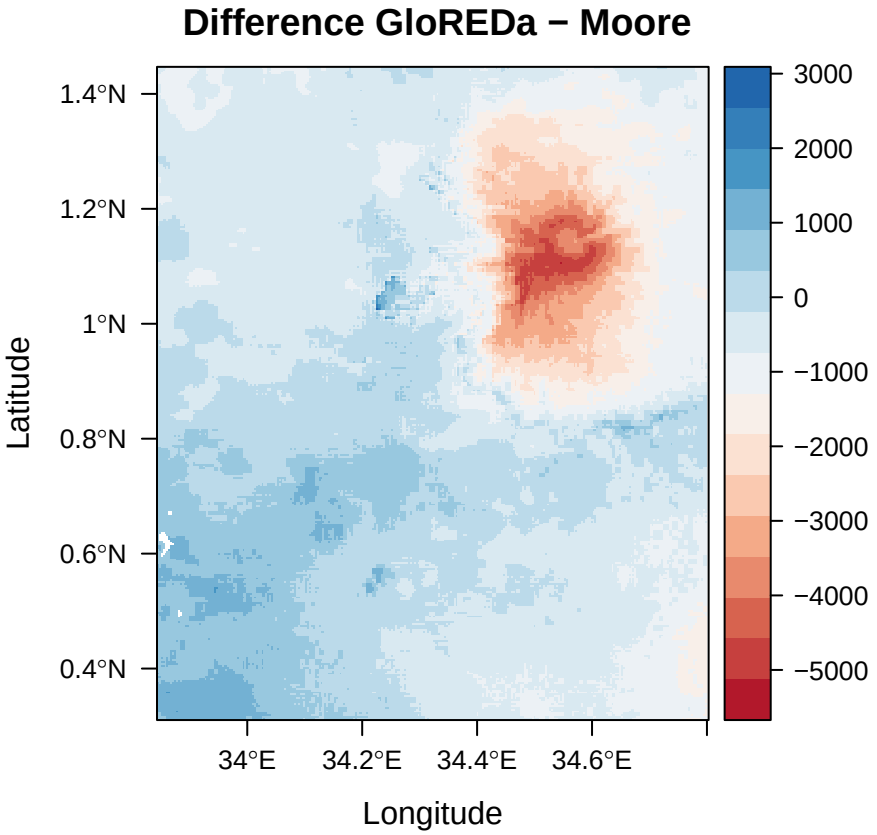
The raster stack can now be plotted using the function `plot()` again. The visualization is however not very 'nice' (each plot has a legend, etc.). A much better visualization is provided by the 'rasterVis' package using the function `levelplot()`:

```
levelplot(r_layer, main = "Rainfall erosivity R")
```



Another good way to compare the two raster layers is to plot the differences between the layers. Therefore we calculate a new difference layer from the raster stack and plot it:

```
# Calculate the differences between the two R layers
r_diff <- calc(r_layer, fun = diff)

# Plot the difference layer
levelplot(r_diff, margin = FALSE, par.settings = RdBuTheme,
          main = "Difference GloREDa - Moore")
```



The difference plot of the two layers clearly shows strong differences in the rainfall erosivities at Mount Elgon. Close to the top of the Mountain, GloREDa shows a very low R, while Moore shows an increase in R (as it is solely driven by the long-term precipitation and the precipitation sums increase with altitude at Mount Elgon). In all other regions (blue in the difference plot), GloREDa shows larger R values compared to Moore.

Soil erodibility K

The majority of methods to calculate the soil erodibility K use the physical soil parameters Sand, Silt, and Clay fraction and the organic carbon (or matter) fraction in the top soil layer. Common methods for the calculation of K that are found in literature are the method of Torri [see e.g. Torri et al., 1997, Yang et al., 2003, Naipal et al., 2015], the Method of Williams [see e.g. Williams, 1995, Karamage et al., 2017], and the Method of Wischmeier [see e.g. Borrelli et al., 2017, Panagos et al., 2014, 2015b, Wischmeier and Smith [1987]].

Data from soil analyses are rarely at hand for any erosion study. In most cases the soil physical properties are derived from large global soil data bases, that provide physical and chemical soil properties for different depths in a grid format. Two comprehensive data sets that are freely available online are the SoilGrids data base [Hengl et al., 2017] that can be accessed via soilgrids.org or the Global Soil DataSet for use in Earth System Models GSDE [Shangguan et al., 2014].

In this course we will calculate the K factor using the Method of Williams together with top soil layers derived from SoilGrids that were already prepared for our study region and can be found in the 'data/soil' folder. The data provided with the course material also provides a realization of K using the Method of Torri with SoilGrids data. This layer will be of use in a later part of the course. We will however compare again these two layers here.

Calculation of K using the method of Williams [1995]

This method is of interest as it is implemented in models or recommended by models that are frequently used in agricultural/hydrological applications, such as the EPIC model [Williams, 1995] and the SWAT model [Arnold et al., 1998]. The equation of Williams can be defined in R as follows:

```
calculate_k_williams <- function(sndprc, sltprc, clyprc, orcprc){
  a <- (0.2 + 0.3*exp(-0.0256*sndprc*(1 - sltprc/100)))
  b <- (sltprc/(clyprc + sltprc))^0.3
  c <- 1 - (0.25*orcprc)/(orcprc + exp(3.72 - 2.95*orcprc))
  sn1 <- 1 - sndprc/100
  d <- 1 - (0.7*sn1)/(sn1 + exp(-5.51 + 22.9*sn1))
  k <- 0.1317*a*b*c*d
  return(k)
}
```

Before calculating *K* we have to load the soil layers into R. As these are raster layers, we again use the command `raster()`:

```
# The input file paths
sand_file <- "data/soil/SNDPPT_M_sl1_250m.tif"
silt_file <- "data/soil/SLTPPT_M_sl1_250m.tif"
clay_file <- "data/soil/CLYPPT_M_sl1_250m.tif"
orgc_file <- "data/soil/ORCDRC_M_sl1_250m.tif"

#Loading the raster files
sand <- raster(sand_file)
silt <- raster(silt_file)
clay <- raster(clay_file)
orgc <- raster(orgc_file)
```

Checking the data is again highly recommended. As examples we will calculate the summary statistics for the Sand layer and the Organic Carbon layer.

```
summary(sand)
```

```
##         SNDPPT_M_sl1_250m
## Min.                   27
## 1st Qu.                41
## Median                 46
## 3rd Qu.                55
## Max.                  255
## NA's                    0
```

```
summary(orgc)
```

```
##         ORCDRC_M_sl1_250m
## Min.                -32768
## 1st Qu.                 60
## Median                  74
## 3rd Qu.                111
## Max.                   318
## NA's                     0
```

The summary shows that the sand layer most likely provides the percentages of sand. The maximum value however shows the value `255`. In this case it is the `NA` value of the raster layer when (as in this case) the values are saved as INT8 (8-bit integer) values. In contrast, the organic carbon values seem very large (larger than 100). A look at the SoilGrids documentation shows that the unit is $g \cdot kg^{-1}$. Therefore we have to divide that layer by `10` to get the percentages of the organic carbon. The minimum value that is shown is `-32768`. This is a typical `NA` value for 16-bit signed integer layers. Apparently `raster()` did not 'recognize' the `NA` values. We can however define the `NA` values 'manually' by using the function `NAvalue()`.

```
# Re-define the NA values of the raster layers
NAvalue(sand) <- 255
NAvalue(silt) <- 255
NAvalue(clay) <- 255
NAvalue(orgc) <- -32768

# Correct the orgc layer
orgc <- orgc/10

#Again check if the statistics of the layers are now right
summary(orgc)
```

```
##         ORCDRC_M_sl1_250m
## Min.                  0.4
## 1st Qu.               6.0
## Median                7.4
## 3rd Qu.              11.1
## Max.                 31.8
## NA's                138.0
```

Now the statistics for organic carbon seam reasonable and we can continue with the calculation of K. Unfortunately we cannot simply use the raster function `calc()` as this function only works with one raster object (Although it might handle a raster stack with multiple layers). To work with multiple rasters we can now use the function `overlay()`:

```
k_williams <- overlay(sand, silt, clay, orgc, fun = calculate_k_williams)
```

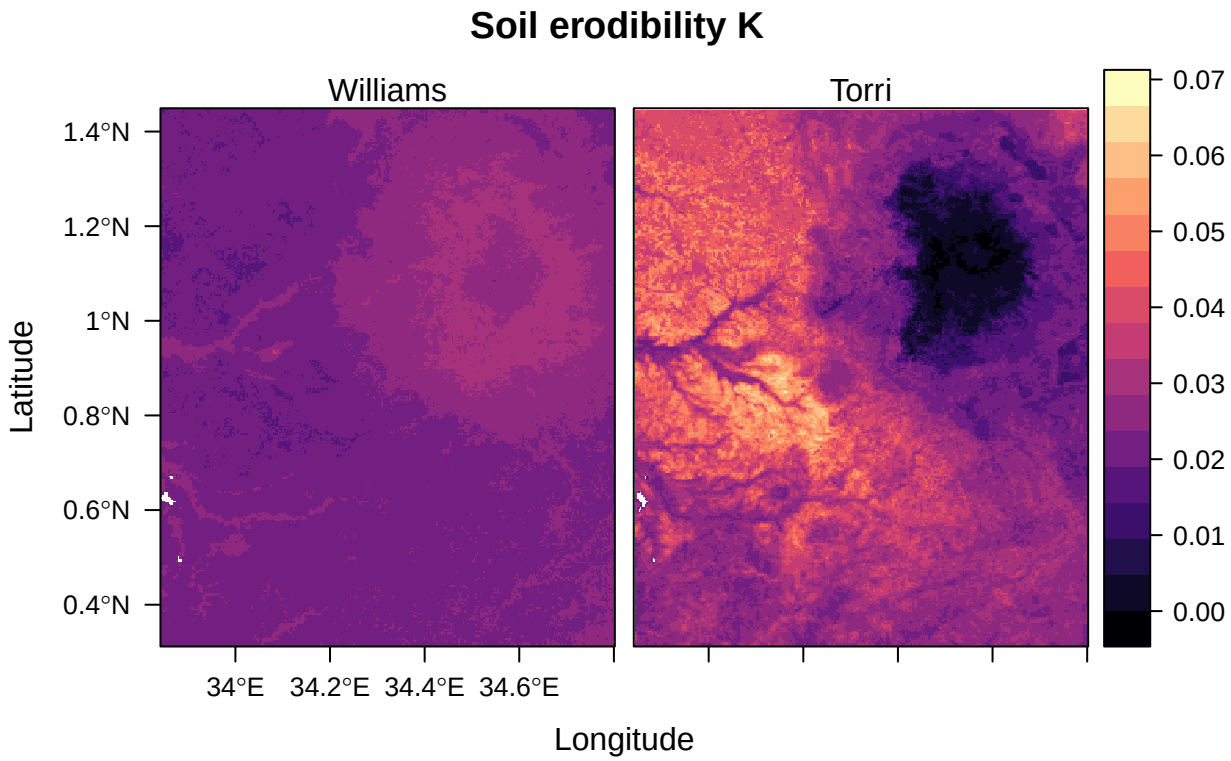
Comparison to the method of Torri et al. [1997]

We will compare the results of our analysis above to the layer that was prepared with the method of Torri. This check should provide some plausibility to see whether the two methods result in similar ranges for *K*.

```
# The file path of the .tif layer for k_torri
k_torri_path <- "data/rusle_input/k_factor/k_torri_soilgrids.tif"

#Load the raster layer for k_torri
k_torri <- raster(k_torri_path)
```

As both layers `k_williams` and `k_torri` were generated from SoilGrids, both layers fully overlap. Therefore, we can easily create a raster stack from them to again make a comparison plot:

```
# Stack the two raster layers
k_layer <- stack(list(Williams = k_williams, Torri = k_torri))

# Plot the two layers with levelplot
levelplot(k_layer, main = "Soil erodibility K")
```



The *K* values of the two layers are in a similar range. Generally, k_torri shows a much larger heterogeneity compared to k_williams. Also the zones of high and low *K* values differ for the two layers.

Write the raster output

Finally when we are convinced that the calculations are plausible, we write the raster to our hard drive as a .tif file.

```
writeRaster(k_williams, filename = "data/rusle_input/k_factor/k_williams.tif",
            overwrite = TRUE)
```

Slope length and steepness LS

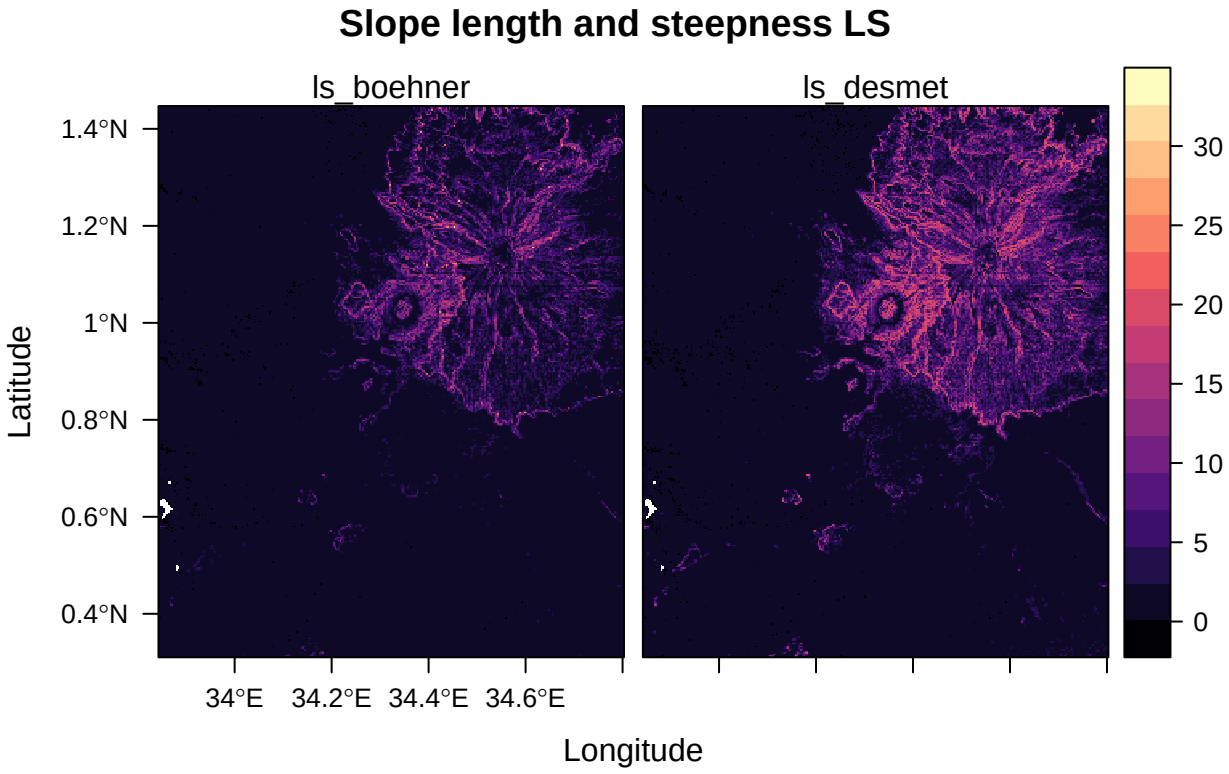
The preparation of the *LS* factor will not be performed in the course. The course material however provides two realizations for the *LS* factor that were calculated with the metod of Desmet and Govers [1996] and the method of Boehner and Selige [2006]. Especially the method of Desmet and Govers [1996] is often applied in literature to derive the *LS* factor. The basis for the calculation is a DEM. In the case of the course data sets we used the SRTM data set [Jarvis et al., 2008]. The methods to calculate the *LS* factor are implemented in some GIS (e.g. SAGA-GIS).

Here we just load the two realizations of *LS* provided in the course and compare them in a combined plot. It is possible to load multiple raster files and directly organizing them in a raster stack by using the function `stack()` rather than loading all raster files with the command `raster()`. Therefore we have to have a vector of all file names we want to load (this can be done with the function `list.files()`) and load them with `stack()`.

```
# List all LS .tif files
ls_files <- list.files(path = "data/rusle_input/ls_factor",
                       pattern = ".tif$",
                       full.names = TRUE)
# Load all LS layers with stack
ls_layer <- stack(ls_files)
```

We can now directly plot the *LS* layers:

```
levelplot(ls_layer, main = "Slope length and steepness LS")
```



The patterns are again very similar, though clear differences are also visible. What both layers have in common, is that large *LS* values are visible in the Mount Elgon region (obviously where steep slopes are present).

Cover management factor C

The *C*-factor relates the soil loss from land with a specific vegetation cover to the soil loss that would result from clean-tilled, continuous fallow land. Two main approaches can be found in the literature to calculate *C*. The first approach relates land uses classified in land cover products and agronomic statistics to C-factor values derived from field experiments [see e.g. Borrelli et al., 2017, Yang et al., 2003, Panagos et al., 2015a, Bosco et al., 2015]. Agronomic statistics are either available from governemntal offices (e.g. UBOS in Uganda or the KNBS in Kenya), or from global data products [see e.g. Monfreda et al., 2008]. For the land use classification products, such as MODIS Landcover [Channan et al., 2014] or ESA-CCI-LC [Bontemps et al., 2013] are used frequently.

The second approach infers the C-factor from vegetation indices that are readily available from satellite based remote sensing data. The standard vegetation index used here is the NDVI (Normalized Difference Vegetation Index) that is available globally as a MODIS product [Didan, 2015]. Several mathematical relationships between NDVI and *C* are available from the literature, where however only the method proposed by Van der Knijff et al. [2000] was implemented in recent literature [see e.g. the soil erosion study for Uganda by Karamage et al., 2017].

In the course we will calculate the *C* factor from a time series of NDVI maps derived for the rainy season (March-June, October-November) for the years 2008-2012 from the MODIS data set. We will calculate the mean of the NDVI data and then apply the method of Van der Knijff et al. [2000] to the mean raster values. Note, that it is also an option to calculate the *C* factor for every

NDVI image and to calculate the mean afterwards.

Here we provide the necessary NDVI raster images due to time constraints. The R package *MODIStsp* [Busetto and Ranghetti, 2016], however, allows for a very convenient download of MODIS data, not only of Ecosystem Variables such as NDVI, but also other MODIS products.

A second realization of *C* is provided with the course materials that was calculated from Agronomic statistics derived from Monfreda et al. [2008] and overlayed with the agricultural areas in the MODIS LC map.

The method of Van der Knijff et al. [2000] can be defined as an R function as follows:

```
calculate_c_knijff <- function(ndvi) {
  alpha <- 2 # as suggested by Knijff 2000
  beta  <- 1 # as suggested by Knijff 2000
  c <- min(exp(-alpha * (ndvi/(beta - ndvi))), 1)
  return(c)
}
```

To load all NDVI layers simply list all NDVI layer files stored in the 'data/ndvi' folder and load them with `stack()`:

```
# Listing all files of the time series for NDVI
ndvi_files <- list.files(path = "data/ndvi",
                         pattern = ".tif$",
                         full.names = TRUE)
# Load all 45 NDVI layers into a raster stack
ndvi_timeseries <- stack(ndvi_files)
```

Again it is recommended to explore the layers. As we now are working with a raster stack, we actually should check every layer in the stack. Exemplary we will look at the summary statistics of the first layer. To access the first layer we can select that layer similar as a `list` entry using the double squared brackets `list[[1]]`;

```
summary(ndvi_timeseries[[1]])
```

```
##           mean_ndvi
## Min.    -0.05394775
## 1st Qu.  0.56365776
## Median   0.60044152
## 3rd Qu.  0.63995790
## Max.     0.85479319
## NA's     0.00000000
```

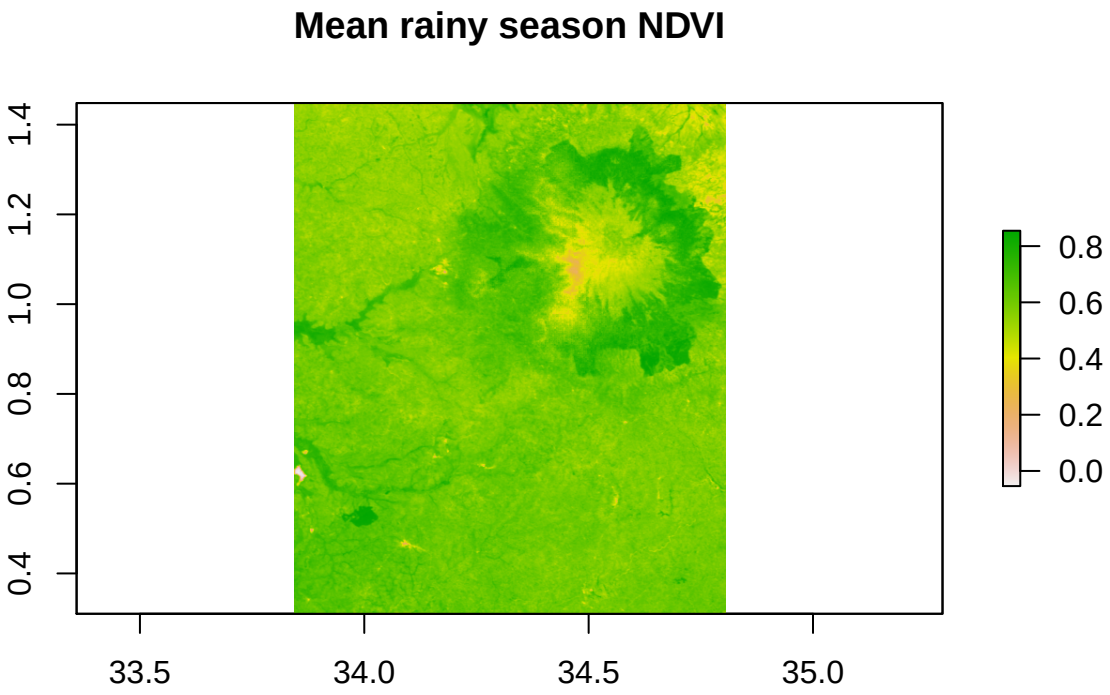
We can see from the summary statistics that the range of `ndvi_timeseries[[1]]` is between `-1974` and `9660`. The NDVI ranges however between -0.2 and 1. The manual for MODIS NDVI indicates that a scale factor of 0.0001 must be applied.

```
ndvi_timeseries <- 0.0001*ndvi_timeseries
```

To get the mean value for the NDVI from all the MODIS images we simply apply the function `mean()` with `calc()`. Here we have to be careful that no mean is calculated for pixels where at least

one layer has an `NA` value. To avoid this, we have to use the option `na.rm = TRUE` (which means 'removing NA values = TRUE') from the function `mean()`.

```
# Let us calculate the mean NDVI, save it as a .tif and plot it.
mean_ndvi <- calc(ndvi_timeseries, mean, na.rm = TRUE)
writeRaster(mean_ndvi, "data/ndvi/mean_ndvi.tif", overwrite = TRUE)
plot(mean_ndvi, main = "Mean rainy season NDVI")
```



Now we can calculate *C* based on the mean NDVI:

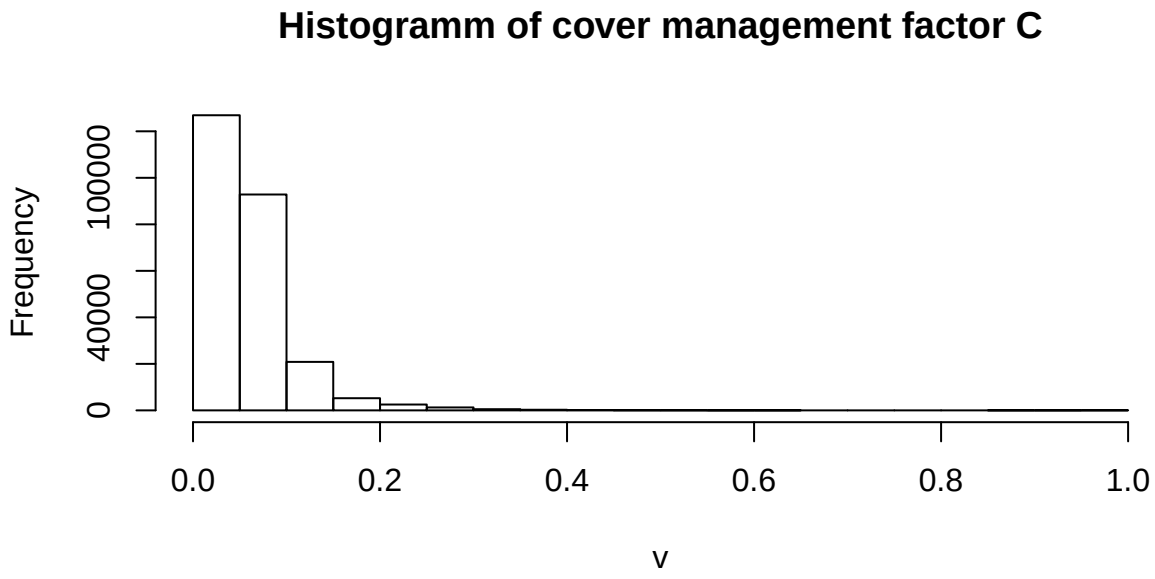
```
c_ndvi <- calc(mean_ndvi, fun = calculate_c_knijff)
```

Note, that we could however also calculate the *C* factor for all NDVI layers separately and then average the results. This we however do not do in the course.

```
c_timeseries <- calc(ndvi_timeseries, fun = calculate_c_knijff)
c_ndvi_ts <- calc(c_timeseries, mean, na.rm = TRUE)
```

We can then plot the histogramm for *C* to check plausiblity and finally save the layer we just generated:

```
hist(c_ndvi, main = "Histogramm of cover management factor C")
```

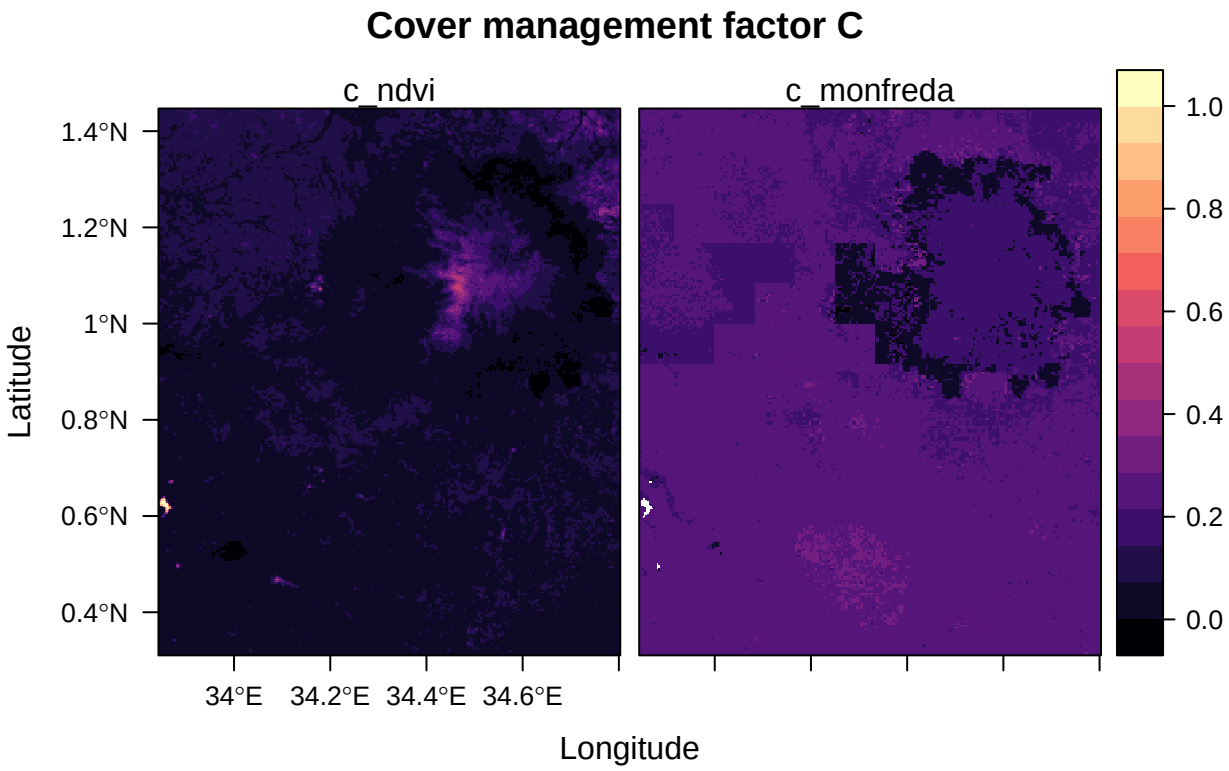



```
writeRaster(c_ndvi, filename = "data/rusle_input/c_factor/c_ndvi.tif",
            overwrite = TRUE)
```

Comparison to the method of Monfreda et al. [2008]

As a final step, we can compare the results of our calculations for *C* to the layer that was prepared with the method after Monfreda et al. [2008]:

```
# The file path of the .tif layer for c_monfreda
c_monfreda_path <- "data/rusle_input/c_factor/c_modis_monfreda.tif"
#Load the raster layer for c_monfreda
c_monfreda <- raster(c_monfreda_path)
# Project the raster c_ndvi, since they have different extents
c_ndvi <- projectRaster(from = c_ndvi, to = c_monfreda)
# Create a combined raster stack for the visualization below
c_layer <- stack(list(c_ndvi = c_ndvi, c_monfreda = c_monfreda))

levelplot(c_layer, main = "Cover management factor C")
```



Summary

The *RUSLE* is widely used by soil conservationists in the world to estimate the erosion by water. This empirically based equation, derived from a large mass of field data, estimates the long-term average annual soil loss (A in $t \cdot ha^{-1} \cdot yr^{-1}$) by multiplying different spatially distributed input layers influencing erosion.

After this session we have to our disposition the following input layers for further calculations:

- Erosivity of rainfall R
  - R after Moore [1979] *(calculated)*
  - R after *GloREDa (provided)*
- Erodibility of soil K
  - K after Williams [1995] *(calculated)*
  - K after Torri et al. [1997] *(provided)*
- Topography represented by slope length and steepness LS
  - LS after Desmet and Govers [1996] *(provided)*
  - LS after Boehner and Selige [2006] *(provided)*
- Cover management factor C
  - C after Van der Knijff et al. [2000] *(calculated)*
  - C after Monfreda et al. [2008] *(provided)*

For the time being we will ignore the support practices P, but may look at it at a later point. With this data we are ready to apply the RUSLE model in the next session.

References

J. G. Arnold, R. Srinivasan, R. S. Muttiah, and J. R. Williams. Large area hydrologic modeling and assessment part I: model development. *Journal of the American Water Resources Association*, 34(1): 73–89, feb 1998. doi: 10.1111/j.1752-1688.1998.tb05961.x.

Y Bamutaze. *Patterns of water erosion and sediment loading in Manafwa in catchment on Mt. Elgon, Eastern Uganda*. PhD thesis, Department of Geography, Geo-Information and Climatic Sciences, Makerere University, 2010.

J. Boehner and T. Selige. Spatial Prediction of Soil Attributes Using Terrain Analysis and Climate Regionalisation. *Goettinger Geographische Abhandlungen*, 115:13–27, 2006.

S. Bontemps, P. Defourny, J. Radoux, E. Van Bogaert, C. Lamarche, F. Achard, P. Mayaux, M. Boettcher, C. Brockmann, G. Kirches, M. Zühlke, V. Kalogirou, F.M. Seifert, and O. Arino. Consistent global land cover maps for climate modelling communities: current achievements of the ESA's land cover CCI. In *Proceedings of the ESA Living Planet Symposium*, pages 9–13. 2013.

Pasquale Borrelli, David A. Robinson, Larissa R. Fleischer, Emanuele Lugato, Cristiano Ballabio, Christine Alewell, Katrin Meusburger, Sirio Modugno, Brigitta Schütt, Vito Ferro, Vincenzo Bagarello, Kristof Van Oost, Luca Montanarella, and Panos Panagos. An assessment of the global impact of 21st century land use change on soil erosion. *Nature Communications*, 8(1), 2017. doi: 10.1038/s41467-017-02142-7.

C. Bosco, D. De Rigo, O. Dewitte, J. Poesen, and P. Panagos. Modelling soil erosion at European scale: Towards harmonization and reproducibility. *Natural Hazards and Earth System Sciences*, 15 (2):225–245, 2015. doi: 10.5194/nhess-15-225-2015.

Lorenzo Busetto and Luigi Ranghetti. Modistsp: an r package for preprocessing of modis land products time series. *Computers and Geosciences*, 97:40–48, 2016. ISSN 0098-3004. doi: 10.1016/j.cageo.2016.08.020. URL https://github.com/ropensci/MODIStsp.

S Channan, K Collins, and WR Emanuel. Global mosaics of the standard modis land cover type data. *University of Maryland and the Pacific Northwest National Laboratory, College Park, Maryland, USA*, 30, 2014.

PJJ Desmet and Gerard Govers. A GIS procedure for automatically calculating the USLE LS factor on topographically complex landscape units. *Journal of soil and water conservation*, 51(5):427–433, 1996.

K. Didan. MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V006., 2015.

Stephen E Fick and Robert J Hijmans. WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, 37(12):4302–4315, 2017. doi: 10.1002/joc.5086.

Tomislav Hengl, Jorge Mendes De Jesus, Gerard B.M. Heuvelink, Maria Ruiperez Gonzalez, Milan Kilibarda, Aleksandar Blagotić, Wei Shangguan, Marvin N. Wright, Xiaoyuan Geng, Bernhard Bauer-Marschallinger, Mario Antonio Guevara, Rodrigo Vargas, Robert A. MacMillan, Niels H. Batjes, Johan G.B. Leenaars, Eloi Ribeiro, Ichsani Wheeler, Stephan Mantel, and Bas Kempen. *SoilGrids250m: Global gridded soil information based on machine learning*, volume 12. Public Library of Science, 2017. ISBN 1111111111. doi: 10.1371/journal.pone.0169748.

A. Jarvis, E. Guevara, H. I. Reuter, and A. D. Nelson. Hole-filled SRTM for the globe. Version 4. CGIAR-CSI SRTM 90m Database. <http://srtm.csi.cgiar.org>, 2008.

Boyi Jiang, Yazidhi Bamutaze, and Petter Pilesjö. Climate change and land degradation in africa: a case study in the mount elgon region, uganda. *Geo-spatial Information Science*, 17(1):39–53, 2014. doi: 10.1080/10095020.2014.889271.

Fidele Karamage, Chi Zhang, Tong Liu, Andrew Maganda, and Alain Isabwe. Soil erosion risk assessment in Uganda. *Forests*, 8(2):52, feb 2017. doi: 10.3390/f8020052.

Chad Monfreda, Navin Ramankutty, and Jonathan A. Foley. Farming the planet: 2. Geographic distribution of crop areas, yields, physiological types, and net primary production in the year 2000. *Global Biogeochemical Cycles*, 22(1):1–19, 2008. doi: 10.1029/2007GB002947.

T R Moore. Rainfall Erosivity in East Africa. *Geografiska Annaler. Series A, Physical Geography*, 61 (3/4):147–156, 1979. doi: 10.2307/520909.

V. Naipal, C. Reick, J. Pongratz, and K. Van Oost. Improving the global applicability of the RUSLE model - Adjustment of the topographical and rainfall erosivity factors. *Geoscientific Model Development*, 8(9):2893–2913, 2015. doi: 10.5194/gmd-8-2893-2015.

Panos Panagos, Katrin Meusburger, Cristiano Ballabio, Pasqualle Borrelli, and Christine Alewell. Soil erodibility in Europe: A high-resolution dataset based on LUCAS. *Science of the Total Environment*, 479-480(1):189–200, 2014. doi: 10.1016/j.scitotenv.2014.02.010.

Panos Panagos, Pasquale Borrelli, Katrin Meusburger, Christine Alewell, Emanuele Lugato, and Luca Montanarella. Estimating the soil erosion cover-management factor at the European scale. *Land Use Policy*, 48:38–50, 2015a. doi: 10.1016/j.landusepol.2015.05.021.

Panos Panagos, Pasquale Borrelli, Jean Poesen, Cristiano Ballabio, Emanuele Lugato, Katrin Meusburger, Luca Montanarella, and Christine Alewell. The new assessment of soil loss by water erosion in Europe. *Environmental Science and Policy*, 54:438–447, 2015b. doi: 10.1016/j.envsci.2015.08.012.

Panos Panagos, Pasquale Borrelli, Katrin Meusburger, Bofu Yu, Andreas Klik, Kyoung Jae Lim, Jae E. Yang, Jinren Ni, Chiyuan Miao, Nabansu Chattopadhyay, Seyed Hamidreza Sadeghi, Zeinab Hazbavi, Mohsen Zabihi, Gennady A. Larionov, Sergey F. Krasnov, Andrey V. Gorobets, Yoav Levi, Gunay Erpul, Christian Birkel, Natalia Hoyos, Victoria Naipal, Paulo Tarso S. Oliveira, Carlos A. Bonilla, Mohamed Meddi, Werner Nel, Hassan Al Dashti, Martino Boni, Nazzareno Diodato, Kristof Van Oost, Mark Nearing, and Cristiano Ballabio. Global rainfall erosivity assessment based on high-temporal resolution rainfall records. *Scientific Reports*, 7(1): 4175, dec 2017. doi: 10.1038/s41598-017-04282-8.

Kenneth G. Renard and Jeremy R. Freimund. Using monthly precipitation data to estimate the R-factor in the revised USLE. *Journal of Hydrology*, 157(1-4):287–306, 1994. doi: 10.1016/0022-1694(94)90110-4.

Kenneth G Renard, George R Foster, Glenn A Weesies, and Jeffrey P Porter. Rusle: Revised universal soil loss equation. *Journal of soil and Water Conservation*, 46(1):30–33, 1991.

Kenneth G Renard, George R Foster, GA Weesies, DK McCool, DC Yoder, et al. *Predicting soil erosion by water: a guide to conservation planning with the Revised Universal Soil Loss Equation (RUSLE)*, volume 703. United States Department of Agriculture Washington, DC, 1997.

O Semalulu, V Kasenge, A Nakanwagi, W Wagoire, and J Tukahirwa. Financial loss due to soil erosion in the mt. elgon hillsides, uganda: A need for action. *Sky Journal of Soil Science and Environmental Management*, 3(3):29–35, 2014.

Wei Shangguan, Yongjiu Dai, Qingyun Duan, Baoyuan Liu, and Hua Yuan. A global soil data set for earth system modeling. *Journal of Advances in Modeling Earth Systems*, 6:249–263, 2014. doi: 10.1002/2013MS000293.

D. Torri, J. W. A. Poesen, and L. Borselli. Predictability and uncertainty of the soil erodibility factor using a global dataset. *Catena*, 31:1—-22, 1997. doi: https://doi.org/10.1016/S0341-8162(97)00036-2.

J.M. Van der Knijff, R.J.A. Jones, and L. Montanarella. Soil Erosion Risk Assessment in Europe, EUR 19044 EN. Technical report, European Soil Bureau, European Comission, 2000.

J. R. Williams. The EPIC model - Soil Erosion. In Vijay P. Singh, editor, *Computer Models of Watershed Hydrology*, pages 909–1000. Water Resources Publications, Highlands Ranch, CO, USA, 1995. ISBN 0918334918, 9780918334916.

W. H. Wischmeier and D. D. Smith. Predicting rainfall erosion losses - a guide to conservation planning. Technical report, Hyattsville, Maryland, 1987.

Dawen Yang, Shinjiro Kanae, Taikan Oki, Toshio Koike, and Katumi Musiake. Global potential soil erosion with reference to land use and climate changes. *Hydrological Processes*, 17(14):2913–2928, 2003. doi: 10.1002/hyp.1441.

B. Yu and C.J. Rosewell. A robust estimators of the R-factor for the Universal Soil Loss Equation. *American Society of Agricultural Engineers*, 39(2):559–561, 1996. doi: https://doi.org/10.13031/2013.27535.

2 - Estimation of soil erosion using the RUSLE

Christoph Schuerz and Mathew Herrnegger
Institute for Hydrology and Water Management (HyWa), University of Natural Resources and Life Sciences (BOKU), Vienna, Austria

---

In this part of the course the RUSLE model will be applied to estimate the annual long-term average soil erosion for the study area of the Mount Elgon region. Additionally we will assess how the Malaba and Malakisi watersheds as well as the administrative units in the Mount Elgon region are affected by soil erosion. Such information is highly valuable for the management of water and land resources.

Contents

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

R packages

At the start of our R script we again define all R packages that we will use in the analysis.

```
library(raster)
library(rasterVis)
```

The RUSLE input layers

In the previous section of the course we learned what inputs are required to calculate the annual long-term average soil erosion with the RUSLE model and how to generate the required inputs R, K, LS, C (and P). Here we will combine the generated (and provided) input layers to a RUSLE model. To recapitulate the soil loss equation is written as follows:

$$A = R \cdot K \cdot LS \cdot C \cdot P$$

where R is the the rainfall erosivity in $MJ \cdot mm \cdot ha^{-1} \cdot h^{-1} \cdot yr^{-1}$, K is the soil erodibility in $t \cdot ha \cdot h \cdot ha^{-1} \cdot MJ^{-1} \cdot mm^{-1}$, LS is the slope length and steepness factor, C is the cover management factor, and P is the support practice factor.

For our case study we choose one (arbitary) realization of every RUSLE input. This does not necessarily mean that the selected realization of an input is better than the other and both realizations are equally valid. The choice is however somewhat arbitary.

First, we load the input layers that we generated and had written to the hard drive again into R with `raster()`:

```
# Again we set the working directory to use the shorter relative
# file ptaths
setwd("Define:/the/path/to/the/folder/on/your/computer")
# open arbitary input layers
r  <- raster("data/rusle_input/r_factor/r_gloreda.tif")
k  <- raster("data/rusle_input/k_factor/k_williams.tif")
ls <- raster("data/rusle_input/ls_factor/ls_desmet.tif")
c  <- raster("data/rusle_input/c_factor/c_ndvi.tif")
p  <- 1
```

We now have to keep in mind that we generated the input layers from different data sources. Therefore, the input layers have different pixelsizes and do not exactly overlap. In order to multiply the raster layers pixel-by-pixel, we have to project and adopt the pixel sizes of the raster layers to one (reference) raster. As the LS raster has the finest resolution (90m) we project all other inputs to that spatial resolution. We can project a raster to any other raster using the function `projectRaster()`. We have to be careful, however, what method we apply for the interpolation of values. As we want to keep the calculated values and do not intend to interpolate values to the finer raster resolution, we use the nearest neighbor method (`method = "ngb"`):

```
r <- projectRaster(from = r, to = ls, method = "ngb")
k <- projectRaster(from = k, to = ls, method = "ngb")
c <- projectRaster(from = c, to = ls, method = "ngb")
```

Assessment of the soil erosion in the study area

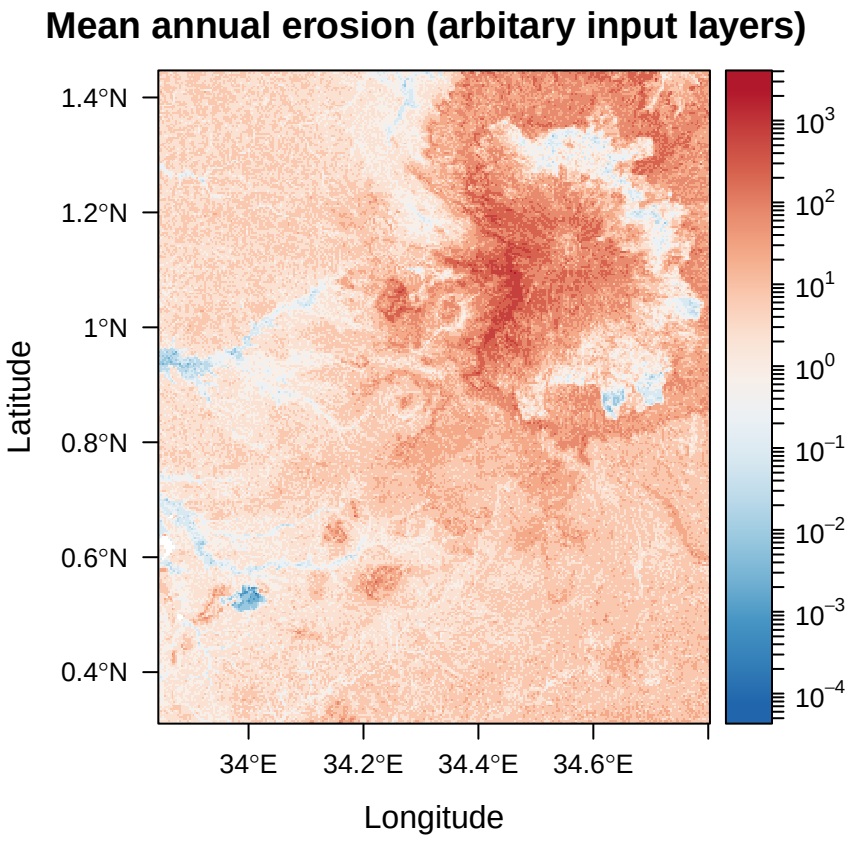
Calculation of the soil erosion using the RUSLE

With the prepared input layers we can now calculate the soil erosion using the RUSLE. We can simply multiply all input layers to calculate the annual soil loss A:

```
a <- r*k*ls*c*p
```

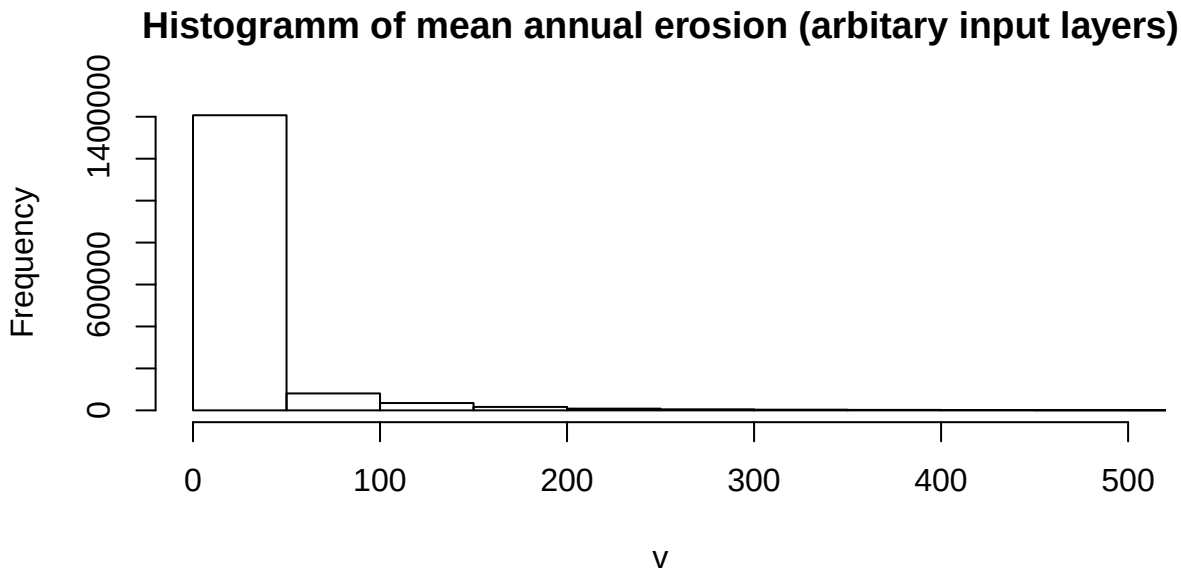
We can now plot the results of the RUSLE. Again we could simply use `plot()`. A better (and maybe more beautiful) visualization is however possible when using the `levelplot()` function from the `rasterVis` package. As the erosion will be very low in many regions, but local hotspots for erosion will be present around Mount Elgon, plotting the erosion on a logarithmic scale might be advantageous. Therefore we have to set the option `zscaleLog = TRUE`. For the intuitive interpretation of the results it is a good idea to plot high erosion rates in red and low erosion in blue. This can be done by using the Blue-Red color theme (by setting `par.setting = BuRdTheme`):

```
levelplot(a, zscaleLog = TRUE, par.settings = BuRdTheme, margin = FALSE, main = "Mean annual ero
```



A histogram shows the distribution of the calculated values and provides a good overall picture of the data. Note, that we added `xlim = c()`, to better visualize the results.

```
hist(a, xlim = c(0, 500) ,
     main = "Histogramm of mean annual erosion (arbitary input layers)")
```



Mean soil erosion in the Malaba and Malakisi catchments

In a next step we want to assess the mean soil erosion for the areas of the Malaba and the Malakisi watersheds. We already used the shapefile of the sub-watersheds of these river basins to plot the location of the study area. Now we will use the shape-polygons to calculate the mean values from our raster layer `a` that are located within the boundaries of each shape-polygon. The raster package provides the function `extract` to do this operation. We want to calculate the mean values and therefore define `fun = mean`. As in previous examples, we want to avoid that the returned result is `NA` just because single pixels were `NA`. Therefore we add the option `na.rm = TRUE`. `extract` has the option to return the result simply as a vector, but also as a spatial object. This can be defined when `sp = TRUE` is set.

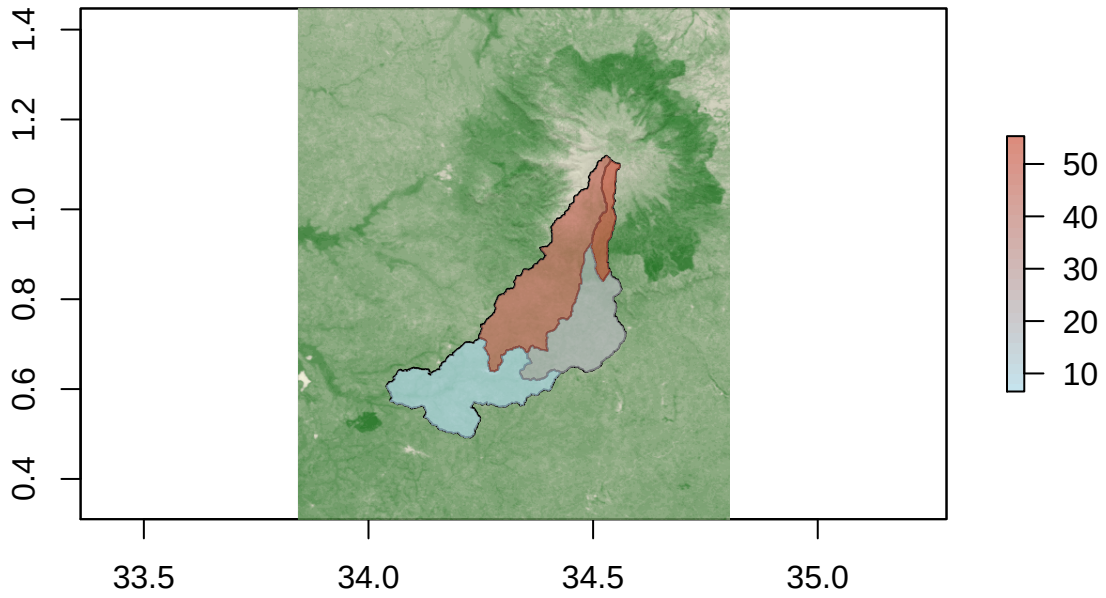
```
# Load the watershed boundaries
watersheds_file <- "data/otherGISData/SubWatersheds_DEM30x30m_WGS84_MM.shp"
watersheds <- shapefile(watersheds_file)

# Calculate the mean erosion values
a_watersheds <- extract(a, watersheds, fun = mean, na.rm = TRUE, sp = TRUE)
```

Using the R base `plot` function for displaying the results has limits. Altough more sophisticated plotting functions exist in R, we want to keep the examples here as simple as possible. To get a reasonable plot for the mean soil erosion, we have to implement workarounds and also convert the shapefile (a_watersheds) to a raster object. This raster object is easier to plot, also including a legend. We will also add the hillshade to have an impression of the terrain. As a background layer we will additionally show the mean NDVI, which we generated and used in part 1 of the course. Hence, we will also have to load these layers into R.

With the results and the additional loaded layers, we can then generate a map by overlaying the rasters and shapefiles step by step. We will now also use custom color ramps, as the provided color palettes (e.g. `terrain.colors()`, or `topo.colors()`) do not result in appropriate

visualisations. Therefore, we will create a custom `colorRampPalette` for the erosion results and the vegetation. For the definition of the `colorRampPalette` we have to provide the colors to be plotted and also define the order:

```
# To use the base plot function of R, it is more convenient to plot the mean
# erosions as raster. Therefore we convert the result ("a_watersheds") into a raster
a_watersheds_raster <- rasterize(a_watersheds, a, field = "layer")

# Load the hillshade and the NDVI layers into R
hill_shade <- raster("data/otherGISData/hillshade.tif")
mean_ndvi  <- raster("data/ndvi/mean_ndvi.tif")

# Define custom color ramps
erosion_palette <- colorRampPalette(c("lightblue","coral3"))
veget_palette   <- colorRampPalette(c("lightyellow","lightyellow3","darkgreen"))

# Plotting the hill_shade as the base layer
plot(hill_shade, col = grey(1:100/100), legend = FALSE)
# Adding the NDVI as proxy for vegetation in brown to green colors and with
# an alpha (transparency) value so that the hillshade is still visible
plot(mean_ndvi, col = veget_palette(25), alpha = 0.75, add = TRUE, legend = FALSE)
# The add the basin boundaries
lines(watersheds)
# Finally ad the results of our assessment in blue (low erosion) to red (high erosion)
plot(a_watersheds_raster, add = TRUE , col = erosion_palette(25), alpha = 0.7)
```



Short excursus to plotting and data science

Until now we tried to use the base R `plot` function for most of our visualization tasks. In the previous example however we saw, that for more complex visualizations base `plot` reaches its limitations. Over the years, many new ways (and R packages) were developed for data visualization. Some of them can also handle the plotting of maps and shape-polygons or shape-files in general.

The most prominent and popular R package for data visualization at the moment (that is also our preferred way to plot) is the `ggplot2` package. It uses a slightly different "grammar" to write the code for plotting. It is however (when you understood the concept) a very intuitive way for plotting, as the concept basically works for every plot in the same way (in contrast with base plot, differrent visualization tasks require different ways to write the code).

The way how `ggplot2` requires the data to look like is also a little different to what we have shown so far in this course (as we specifically handled spatial data until now). The preferred data format to do data analysis in R is the `data.frame` format. For those who have done some data analysis in Excel may see a `data.frame` as "the R version" of an Excel table.

Unfortunately, this course cannot cover all fields and R packages that are useful to perform data analysis efficiently. As the R community is very productive and provides many free resources online to learn R, we will provide some links for those who want to "dive" deeper into R:

- A comprehensive online book to learn a modern way to do data science in R can be found here: R for Data Science an excellent book by Hadley Wickham and Garrett Grolemund.

- RStudio provides compact but informative study materials ("cheat sheets") for their R packages. Packages that are useful for any data analysis are `dplyr` (see the Data Transformation Cheat Sheet), or `ggplot2` (see the Data Visualization Cheat Sheet)
- The Cookbook for R is a nice summary of code that might provide solutions to basic questions for e.g. plotting with `ggplot2`.

We will demonstrate some useful functions and routines you can use for many analyses in the following examples. Many of the concepts shown below can be transefed to other tasks you might encounter when working with R. In the next section we will repeat the plot we created above with `base` plotting. Now we will however try to do it a more `tidy` way.

Additionaly R packages we require

```
# We require some additional R packages

# dplyr
# Very useful for data transformation (selcting, filtering, mutating variables).
# Similar to working with pivot tables in Excel
library(dplyr)

# ggplot2
# A comprehensive R package for data visualization
library(ggplot2)

# ggnewscale
# This package was created by a user from the R community and has to be
# installed in a different way. It is already installed for the course.
# The package allows to use multiple colorramps in a ggplot
# The installation would work like the following, but requires an internet connection:
# install.packages("devtools") # devtools is required to install from github
# devtools::install_github("eliocamp/ggnewscale")
library(ggnewscale)

# sf (Simple Features)
# Simple Features are shape-polygons (similar to shapefile objects).
# They can be displayed and handled as a data.frame, which is great!
library(sf)
```

Loading and modifying the spatial data (the "data.frame" way)

The "native" R data structure to work with is the `data.frame`. Therefore we will convert the raster layers `hill_shade` and `mean_ndvi` to `data.frames`:

```
# Convert the raster layers to data.frames
hillshade_df <- as.data.frame(hill_shade, xy = TRUE)
ndvi_df <- as.data.frame(mean_ndvi, xy = TRUE)
```

A `data.frame` is a table where each column is a variable and each row gives the values of all variables for one observation. As example `ndvi_df` is now a `data.frame` with the 3 variables x, y, and mean_ndvi. Each row therefore gives the x/y coordinates and their corresponding NDVI value:

```
head(ndvi_df)
```

```
##          x        y mean_ndvi
## 1 33.84451 1.447066 0.5159924
## 2 33.84660 1.447066 0.5187424
## 3 33.84869 1.447066 0.5291729
## 4 33.85078 1.447066 0.5247185
## 5 33.85287 1.447066 0.5263207
## 6 33.85495 1.447066 0.5402882
```

Similar to treating the raster objects in tables, the `sf` package provides a way to handle polygon-shapefiles the same way. We can load the watershed shapefile as a simple feature using the function `read_sf`:

```
watersheds_sf <- read_sf(watersheds_file)
```

The shapefile we read with the function `shapefile` and the simple feature that we just read with `read_sf` are the same collection of polygons of the subbasins in our study area. The two objects are however organized in different ways in R. The `sf` object also provides all the required spatial information. The central information is however the data (as a `data.frame`!), similar to the attribute table in a traditional GIS.

Calculate the mean erosion (the "data.frame" way)

We can now work in the same way with the `sf` object as we did before with the `shapefile`. However, we can additionally use the object to do data analysis, as we would do with a `data.frame`. A basic symbol you will find in many R scripts is the `$` operator. It is used to call individual variables of a `data.frame` or a `list`. If we only want to display the names of the watersheds, we can do that the following way:

```
# Display the column in the simple feature table that holds the names
watersheds_sf$Name
```

```
## [1] "Malakisi Sub-Watershed 1"       "Malakisi Sub-Watershed 2"
## [3] "Lwakhakha-Malaba Sub-Watershed" "Malaba-Malakisi Sub Watershed"
```

We can also add new columns with the `$` operator and the name of the new columns (or the name of the old column if we want to overwrite the values of a variable). In our example we now add the information of the mean erosion for the subbasins to our simple feature by assigning the values we calculate with `extract` to the new column `.$mean_erosion`:

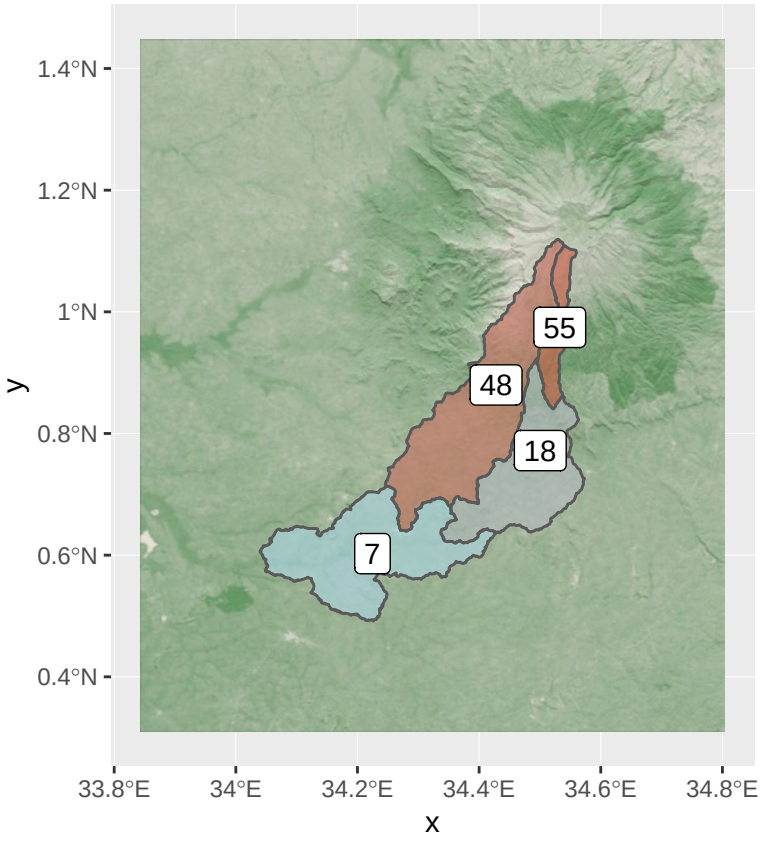
```
watersheds_sf$mean_erosion <- extract(a, watersheds_sf,
                                      fun = mean,
                                      na.rm = TRUE)
```

Plotting the mean erosion using ggplot2

`ggplot2` is a a very, very comprehensive R package for plotting. Thus, if you want to learn and really master `ggplot2` (what we strongly recommend), we refer to the online resources listed above. In brief, `ggplot2` works with the concept of grammar of graphics [Wilkinson et al., 2005]. `ggplot` builds a plot based on different layers that are added together with `"+"`. Each ggplot starts with the command `ggplot()`. Here we can allready define the data in the brackets with `data =`, but also the parameters that control the plot in ggplot called aesthetics (`aes()`). The next layer usually is a geometry (`geom_xxxx()`), followed by other layers that modify the plot. A not so basic example is our map of the mean erosion values for the subbasins. It however contains (as it is the case for every ggplot) the basic setup of a ggplot:

```
# Starting a ggplot
ggplot() +
# We add the hillshade layer with geom_raster and define the x and y coord.
# and define that the variable "hillshade" in "hillshade_df"" defines the color of the pixels.
  geom_raster(data = hillshade_df, aes(x = x, y = y, fill =  hillshade)) +
# We define that the hillshade should be plotted with a grey color gradient
  scale_fill_gradientn(colours = grey((1:100)/100)) +
# As we need another coler gradient for the next layer we tell the plot
# that the next layer requires new fill color settings
  new_scale_fill() +
# We add as further raster layer the NDVI but set alphe = 0.5 for transparancy
  geom_raster(data = ndvi_df, aes(x = x, y = y, fill =  mean_ndvi), alpha = 0.5) +
# We use the same color gradient we defined for the base plot for NDVI
  scale_fill_gradientn(colours = veget_palette(25)) +
# The next layer also needs the fill gradient to define the color.
# Therefore tell the plot again to use a new scale for filling
  new_scale_fill() +
# We now add the polygons using the geometry "sf"
# As aesthetics we only have to define what controls the fill color of the
# polygons. In our case we want to color them accordint to their mean erosion.
  geom_sf(data = watersheds_sf, aes(fill = mean_erosion), alpha = 0.6) +
# We use the same color gradient we defined for the base plot for the erosion
  scale_fill_gradientn(colours = erosion_palette(25)) +
# We add the values of the mean erosion as labels to the plot. We
# round the values not to show all digits.
  geom_sf_label(data = watersheds_sf, aes(label = round(mean_erosion, digits = 0))) +
# To avoid a legend to be plotted, we say in the theme of the plot that the
# position of the legend should be "none"
  theme(legend.position = "none")
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may
## not give correct results for longitude/latitude data
```



Mean erosion on an administrative level

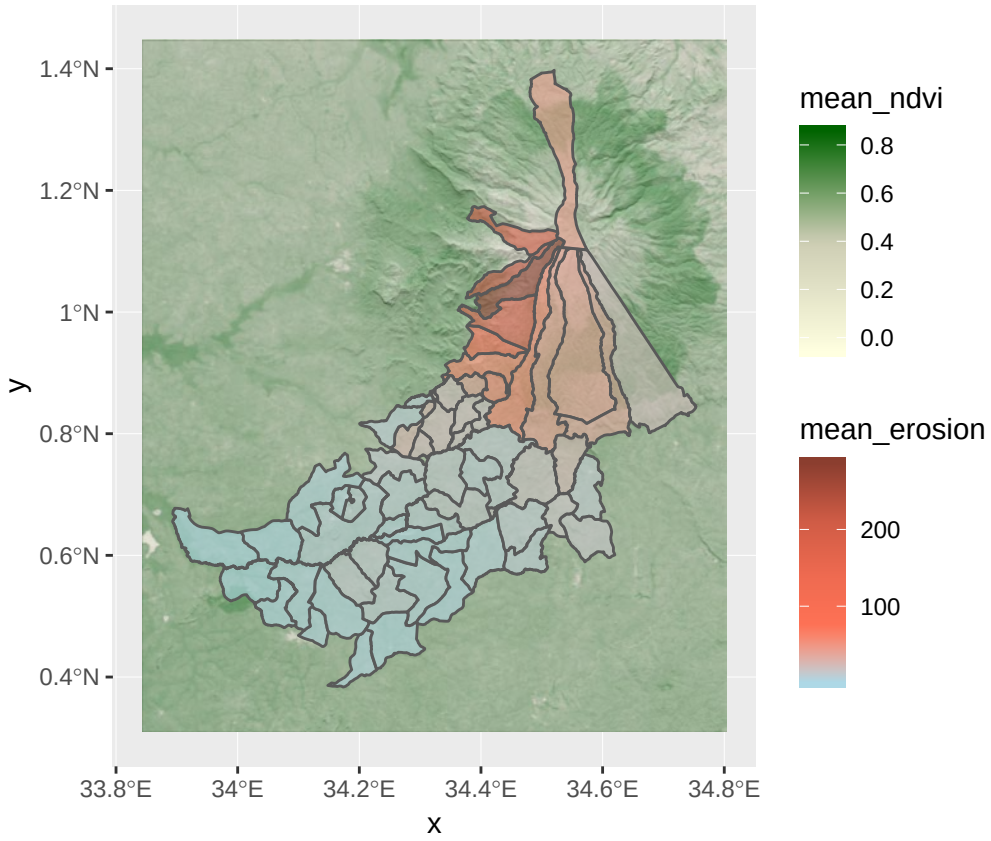
For agricultural and watershed managers it is relevant to have information provided on an administrative level. In a next step we calculate the mean erosion rates on the ward level in Kenya and the sub-county level in Uganda.

```
# Loading the administrative boundaries as a simple feature
admin_bound <- read_sf("data/otherGISData/KenyaWards_UgandaSubcounties_MM.shp")

# Calculate the mean erosion on administrative level
admin_bound$mean_erosion <- extract(a, admin_bound,
                                    fun = mean,
                                    na.rm = TRUE)
```

Visualization of the mean erosion on adminstrative levels

The visualization uses the same code as we have written for the ggplot above. Now we want to keep the legends of the plot, as labels with erosion values in the plot make the entire plot illegible. Further, we modify the color ramp for the erosion, as we now have to visualize a much wider range of erosion sums. We plot "lower" erosion rates in blue and high erosion rates in red. To shift the color ramp, we simply add further shades of red to the colors we use:

```
ggplot() +
  geom_raster(data = hillshade_df, aes(x = x, y = y, fill =  hillshade)) +
  scale_fill_gradientn(colours = grey((1:100)/100), breaks = 1000) +
  new_scale_fill() +
  geom_raster(data = ndvi_df, aes(x = x, y = y, fill =  mean_ndvi), alpha = 0.5) +
  scale_fill_gradientn(colours = veget_palette(25)) +
  new_scale_fill() +
  geom_sf(data = admin_bound, aes(fill = mean_erosion), alpha = 0.6) +
  scale_fill_gradientn(colours = c("lightblue","coral1", "coral2",
                                   "coral3", "coral4"))
```

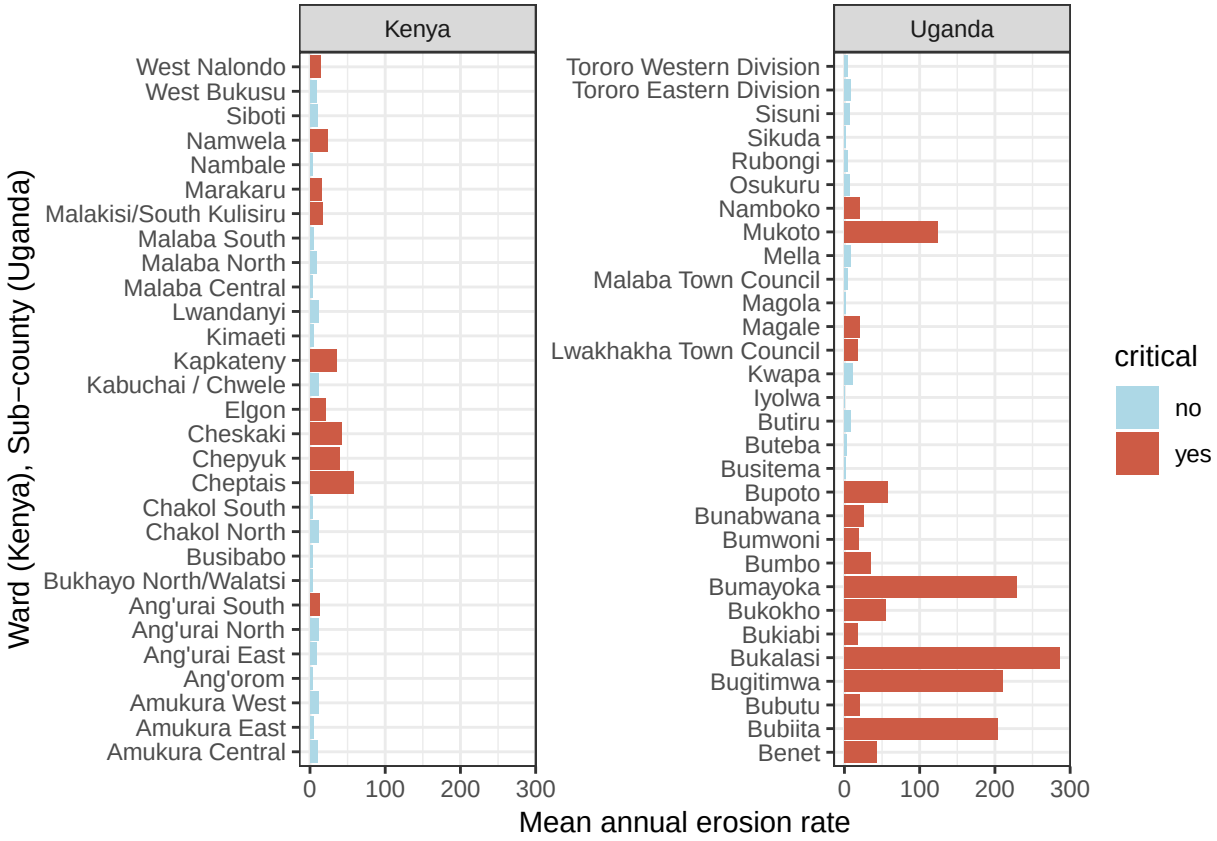


It is obvious that the administrative units around Mount Elgon are affected most by soil erosion. In a watershed management context however one should also consider areas that are (although not as severe) affected by soil erosion above a critical threshold. Therefore, ranking the adminstrative units in a plot would strongly suppport any management decisions.

Identifying the severely affected administrative units

Now we can use the big advantage of simple feature objects and perform simple data analysis for our administrative units. We want to plot the mean erosion for all administrative units and would like to emphasize the units where the mean soil erosion is above 12 $tons\,\mathrm{cot}\,ha^{-1} \cdot yr^{-1}$ that can be considered as tolerable [Montgomery, 2007].

```
# We add a variable named "critical" that says whether the erosion is critcal or not
admin_bound <- mutate(admin_bound,
                      critical = ifelse(mean_erosion > 12, "yes", "no"))

# We now plot the erosion rates of the administrative units in a bar plot
# We add the data and the aesthetics in the ggplot command
ggplot(data = admin_bound, aes(x = KeWar_UgSu, y = mean_erosion)) +
# For a barplot we use the geometry "column"
  geom_col(aes(fill = critical)) +
# Not critical erosion rates should be plotted in blue, critical in red
  scale_fill_manual(values = c("lightblue", "coral3")) +
# Facetting is useful when multiple plots for different groups (in our case
# one for Kenya and one for Uganda) should be plotted
  facet_wrap(.~Country, scales = "free_y") +
# Sometimes plots are better to read when we flip the coordinate system
  coord_flip() +
# Changing the label on the x axis (now y axis)
  xlab("Ward (Kenya), Sub-county (Uganda)") +
# Changing the label on the y axis (now x axis)
  ylab("Mean annual erosion rate") +
# ggplot provides some template themes. The black/white theme is good
  theme_bw()
```



We can see from the plot, that the average erosion rates for the administrative units are fre-

quently very high. Out of a total of 59 administrative units analysed, 26 or 44% show critical erosion rates. Generally we should critically evaluate the results, also analysing the input layers generated. Nevertheless, such a visualization emphasizes the regions that require more attention for mitigation measures and support decisions of a watershed manager.

In this modelling exercise we chose an arbitary set of input layers for the application of the *RUSLE* model. The questions remains, how large the spread or uncertainty in the soil erosion estimates is, if different input layer combinations are used. This is what we will look at in the third and final session.

References

David R Montgomery. Soil erosion and agricultural sustainability. *Proceedings of the National Academy of Sciences*, 104(33):13268–13272, 2007.

Leland Wilkinson, D. Wills, D. Rope, A. Norton, and R. Dubbs. *The Grammar of Graphics*. Springer-Verlag GmbH, 2005. ISBN 0387245448.

3 - Uncertainty assessment of soil erosion estimates

Christoph Schuerz and Mathew Herrnegger
Institute for Hydrology and Water Management (HyWa), University of Natural Resources and Life Sciences (BOKU), Vienna, Austria

---

In this part of the course we analyze the uncertainties in the soil erosion assessment that result from different realizations of the model inputs that can be used in the RUSLE model. Simulation results are always uncertain. Here we show, how these uncertainties can be evaluated and analysed.

Contents

This short course is conducted in the framework of the academic partnership project "Capacity building on the water-energy-food security Nexus through research and training in Kenya and Uganda" (CapNex); funded by the Austrian government through the APPEAR program of the Austrian Development Cooperation.

R packages

At the start of our R script we again define all R packages that we will use in the analysis.

```
library(raster)
library(rasterVis)
library(dplyr)
library(ggplot2)
library(sf)
```

Soil erosion estimates from different RUSLE setups

In the previous section we worked with one realization of the RUSLE model to estimate the mean annual soil erosion for the Mount Elgon region. This RUSLE model setup involved one specific (arbitary) realization for each one of the model inputs. At the beginning of the course we saw however, that there usually are different ways to calculate the model inputs (we considered two different realizations per model input).

We can now analyze the changes in the soil erosion estimates when we also consider other realizations for the RUSLE model inputs in the simulation. As we provide (or generated) two realizations for each of the four RUSLE model inputs, there are 16 different combinations of the RUSLE model possible to calculate the soil erosion.

In the following you will learn how you can use R and standard elements of programming, such as `for` loops to automatically calculate the soil erosion with all 16 model combinations. 16 model setups are theoretically managable to calculate manually. A few more realizations of the inputs would however result in hundreds or thousands of different model setups, that are infeasible to calculate without any automated procedure. In the following you will see how you can automatize such a process using R.

In a first step we want to list all the realizations that we have available for the RUSLE model inputs and will derive all possible RUSLE model combinations from them:

```
# Again we set the working directory to use the shorter relative
# file ptaths
setwd("Define:/the/path/to/the/folder/on/your/computer")
```

```
# We list the names of the layers for all model inputs
r_layer  <- list.files("data/rusle_input/r_factor")
k_layer  <- list.files("data/rusle_input/k_factor")
ls_layer <- list.files("data/rusle_input/ls_factor")
c_layer  <- list.files("data/rusle_input/c_factor")
```

R provides an easy way to find all the combinations of variables using the function `expand.grid()`. When we apply that function to our four input layers we get as a result a `data.frame` with four columns and 16 rows that show all 16 possible model combinations:

```
rusle_comb <- expand.grid(r = r_layer,
                          k = k_layer,
                          ls = ls_layer,
```

```
                          c = c_layer,
# CAUTION here! A problem of R is that factors are preferred. If you do not
# know what to do with factors PLEASE use "stringsAsFactors = FALSE"!
                          stringsAsFactors = FALSE)


rusle_comb
```

```
##                r                     k             ls                    c
## 1  r_gloreda.tif k_torri_soilgrids.tif ls_boehner.tif c_modis_monfreda.tif
## 2    r_moore.tif k_torri_soilgrids.tif ls_boehner.tif c_modis_monfreda.tif
## 3  r_gloreda.tif        k_williams.tif ls_boehner.tif c_modis_monfreda.tif
## 4    r_moore.tif        k_williams.tif ls_boehner.tif c_modis_monfreda.tif
## 5  r_gloreda.tif k_torri_soilgrids.tif  ls_desmet.tif c_modis_monfreda.tif
## 6    r_moore.tif k_torri_soilgrids.tif  ls_desmet.tif c_modis_monfreda.tif
## 7  r_gloreda.tif        k_williams.tif  ls_desmet.tif c_modis_monfreda.tif
## 8    r_moore.tif        k_williams.tif  ls_desmet.tif c_modis_monfreda.tif
## 9  r_gloreda.tif k_torri_soilgrids.tif ls_boehner.tif           c_ndvi.tif
## 10   r_moore.tif k_torri_soilgrids.tif ls_boehner.tif           c_ndvi.tif
## 11 r_gloreda.tif        k_williams.tif ls_boehner.tif           c_ndvi.tif
## 12   r_moore.tif        k_williams.tif ls_boehner.tif           c_ndvi.tif
## 13 r_gloreda.tif k_torri_soilgrids.tif  ls_desmet.tif           c_ndvi.tif
## 14   r_moore.tif k_torri_soilgrids.tif  ls_desmet.tif           c_ndvi.tif
## 15 r_gloreda.tif        k_williams.tif  ls_desmet.tif           c_ndvi.tif
## 16   r_moore.tif        k_williams.tif  ls_desmet.tif           c_ndvi.tif
```

In the previous example, where we calculated the soil erosion based on one realization of the RUSLE, we first loaded all required layers of *R*, *K*, *LS*, and *C* and projected the input layers to the raster of the *LS* layer. In a final step we then calculated the soil erosion raster by multiplying all layers.

We will use the same procedure here, but will repeat it for all 16 model combinations. To repeat the procedure automatically, we loop over all model combinations using a `for` loop. A for loop has the following structure:

```
for(i in 1:5) {
  cat("Hello I am loop number", i, "!", "\n")
}
```

```
## Hello I am loop number 1 !
## Hello I am loop number 2 !
## Hello I am loop number 3 !
## Hello I am loop number 4 !
## Hello I am loop number 5 !
```

As you can see the for loop needs a "index variable" that we call `i` here. With `in` we tell the variable which values it should take. In our case we tell the index variable that it should take the values 1 to 5 and execute the code between the `{}` brackets for each index variable. In our case it simply prints the text, which loop iteration the loop is currently doing. We will now use the `for` loop for many steps to eventually calculate the erosion with all 16 model setups.

Projection of the RUSLE model inputs

We know from the calculation of the soil erosion that we had to project the input layers to one raster (in our case the grid of the LS layers) in order to multiply all layers. This step requires some computation time. Therefore it is not advisable to do the projection step for every realization of the RUSLE model. To project all the raster input layers in one step and save the raster layers before performing the RUSLE model calculations will therefore save a lot of computation time.

```
# We will now list all the paths to all RUSLE input layers
# The "recursive" setting means, to also show the content of folders in folders
rusle_input_path <- list.files(path = "data/rusle_input",
                               recursive = TRUE,
                               full.names = TRUE)

# We load a "reference" layer that we will use to project all other inputs
ref_raster <- raster("data/rusle_input/ls_factor/ls_desmet.tif")

# We will save the inputs in a list that we have to initialize before
rusle_input <- list()

# With a for loop we iterate over all input paths and load the layers
# We use "length" here to get the number of inputs. Be careful with fixed
# values, since they might change when you change something in your code!
for(i in 1:length(rusle_input_path)) {
  input_i <- raster(rusle_input_path[i])
  input_i <- projectRaster(from = input_i, to = ref_raster, method = "ngb")
  rusle_input <- c(rusle_input, input_i)
}

# "basename" extracts the file name from a long path
input_names <- basename(rusle_input_path)

# We now give the entries in our input list the correct names:
names(rusle_input) <- input_names

# Lets look at the variable
rusle_input
```

```
## $c_modis_monfreda.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : c_modis_monfreda
## values      : 0, 0.3188018  (min, max)
##
```

```
##
## $c_ndvi.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : c_ndvi
## values      : 7.706405e-06, 1  (min, max)
##
##
## $k_torri_soilgrids.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : k_torri_soilgrids
## values      : 9.418945e-12, 0.06764529  (min, max)
##
##
## $k_williams.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : k_williams
## values      : 0.01733038, 0.03467216  (min, max)
##
##
## $ls_boehner.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : ls_boehner
## values      : 0.065, 39.4908  (min, max)
##
##
## $ls_desmet.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
```

```
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : ls_desmet
## values      : 0.03, 19.30461  (min, max)
##
##
## $r_gloreda.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : r_gloreda
## values      : 2274.247, 9063.919  (min, max)
##
##
## $r_moore.tif
## class       : RasterLayer
## dimensions  : 1364, 1152, 1571328  (nrow, ncol, ncell)
## resolution  : 0.0008333333, 0.0008333333  (x, y)
## extent      : 33.84292, 34.80292, 0.3104172, 1.447084  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : r_moore
## values      : 4533.773, 9658.496  (min, max)
```

Calculate different RUSLE setups

We can now use the projected inputs and are ready to calculate all combinations of the RUSLE model. Again we will use a `for` loop to automatize that process:

```
# This time our variable "a" will not only store the result of one simulation
# but all simulation results. We will save all simulations in a "list", which
# initialize before the for-loop.
a <- list()

# We use nrow() here to get the number of combinations we have to execute
# Be careful with fixed values since they might change when you change
# something in your code!
for (i in 1:nrow(rusle_comb)) {
# We take the "i-th" value from the rusle combinations for r
# You can see we again use the "$" operator to select the columns "r"
  r_file  <- rusle_comb$r[i]
# We do the same for all the other inputs too
  k_file  <- rusle_comb$k[i]
```

```
  ls_file <- rusle_comb$ls[i]
  c_file  <- rusle_comb$c[i]

# We load the respective model inputs now from our input list.
# Take care here. To access the content of a list we need "[[ ]]"
  r  <- rusle_input[[r_file]]
  k  <- rusle_input[[k_file]]
  ls <- rusle_input[[ls_file]]
  c  <- rusle_input[[c_file]]
  p  <- 1

# We save the i_th simulation temporarily
  a_i <-  r*k*ls*c*p

# Add the new calculated "a_i" to the list "a"
  a <- c(a, a_i)
}
```

Identify differences between realizations

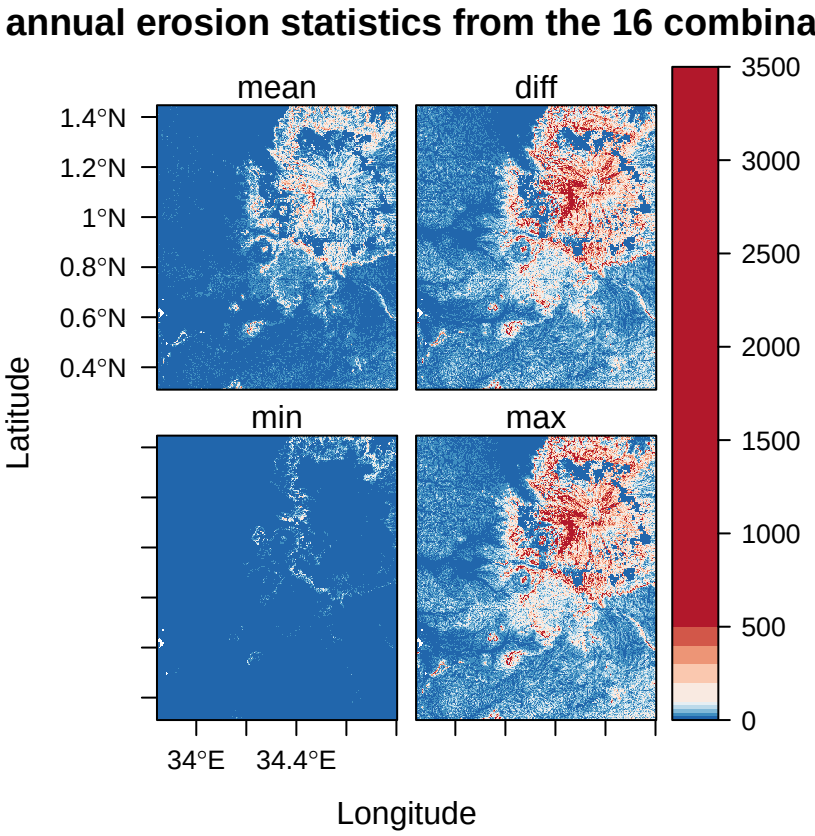
The list `a` contains 16 raster layers which represent the different possible combinations for evaluating the RUSLE. We can now analyze all realizations individually. A good overview could be by plotting the ranges of the soil erosion estimates in a spatial map.

To calculate the pixelwise differences between the 16 realizations we will create a raster stack, as the `raster` package works well with raster stacks. For the raster stack we calculate the pixelwise mean-, minimum-, and maximum- values. Then we can calculate the differences between the minimum and maximum layers:

```
a_stack <- stack(a)
a_min   <- calc(a_stack, min, na.rm = TRUE)
a_max   <- calc(a_stack, max, na.rm = TRUE)
a_mean  <- calc(a_stack, mean, na.rm = TRUE)

a_diff <- a_max - a_min

a_stat <- stack(list(mean = a_mean, diff = a_diff,
                     min  = a_min,  max  = a_max))
```

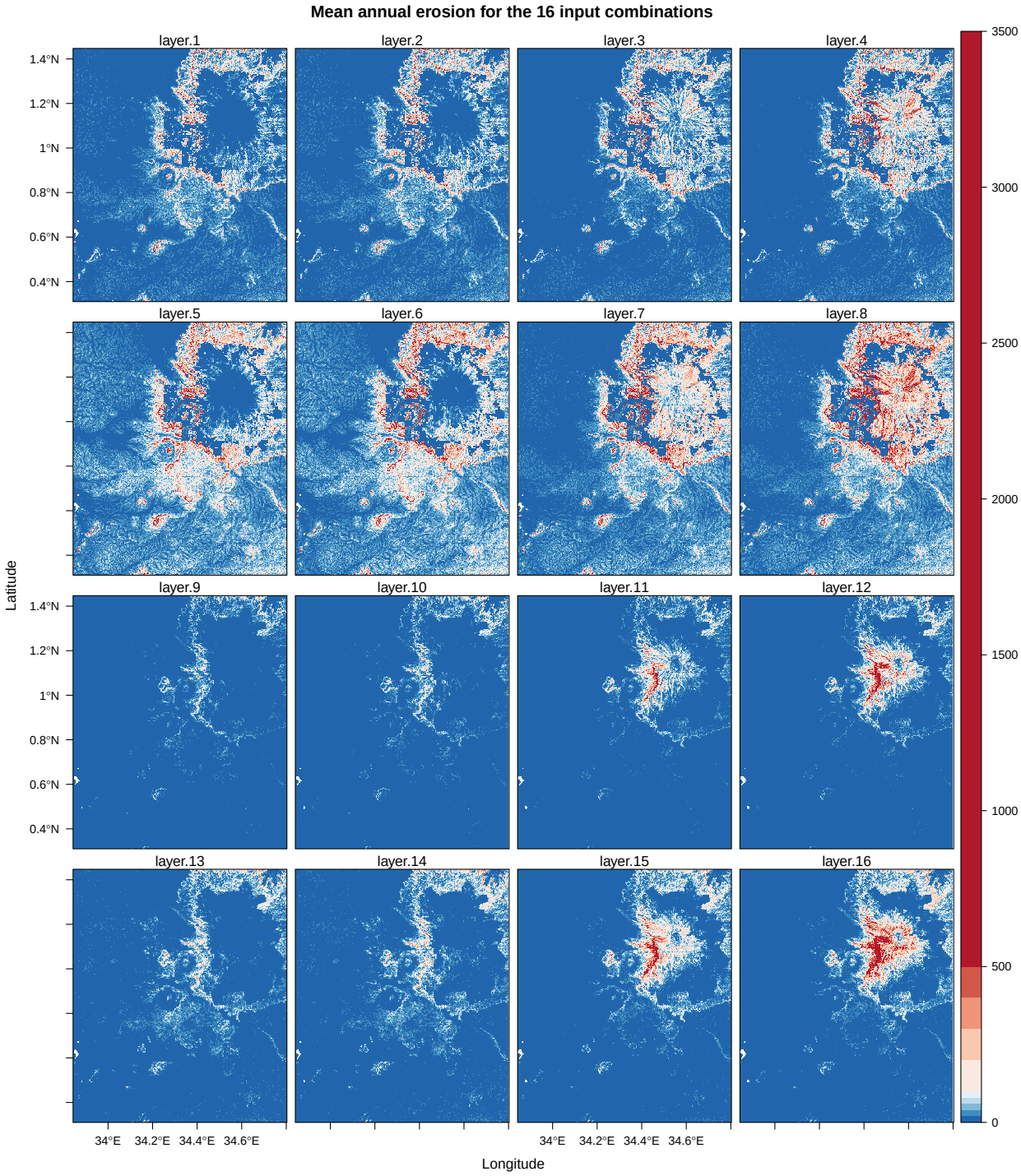
Analysing the above results using functions like `summary` or `hist` shows that the ranges in the difference layer are very wide as for some zones the realizations of the RUSLE model strongly disagree. Therefore we again plot the results on a logarithmic scale:

```
levelplot(a_stat, par.settings = BuRdTheme,
          at = c(seq(0,100, 20), seq(200,500, 100), 3500),
          main = "Mean annual erosion statistics from the 16 combinations")
```



Additionaly we will also show all 16 realizations in a combined plot, with the aim to show the differences between the individual realizations (Attention - this takes quite some time):

```
levelplot(a_stack, par.settings = BuRdTheme,
          at = c(seq(0,100, 20), seq(200,500, 100), 3500),
          layout = c(4,4),
          main = "Mean annual erosion for the 16 input combinations")
```


Mean annual erosion for the 16 input combinations
layer.1
layer.2
layer.3
layer.4
layer.5
layer.6
layer.7
layer.8
layer.9
layer.10
layer.11
layer.12
layer.13
layer.14
layer.15
layer.16
1.4°N
1.2°N
1°N
0.8°N
0.6°N
0.4°N
34°E
34.2°E
34.4°E
34.6°E
Latitude
Longitude
3500
3000
2500
2000
1500
1000
500
0

Soil erosion on the administrative level

In the previous session we calculated the mean annual erosion for the administrative units in Kenya and Uganda based on one realization of the RUSLE model. We created a plot that should support the decision making process whether soil erosion mitigation measures may be necessary in a region or not.

Now we realized however, that many different RUSLE model setups are possible and a strong variability in the resulting soil erosion is the case. We therefore repeat the assessment of soil erosion on the administrative level and take into account the uncertainties in the soil erosion assessment that results from different model inputs.

Aggregation of RUSLE realizations

We will apply the same concept as we used before to calculate the mean erosion for the administrative units, but for all realizations of the RUSLE.

To perform the `extract` for all RUSLE realizations we load the administrative boundaries as a simple feature object:

```
# Loading the administrative boundaries as a simple feature
admin_bound <- read_sf("data/otherGISData/KenyaWards_UgandaSubcounties_MM.shp")
```

Again, we use a `for` loop to iterate over all combinations and save the results in a similar way as we did above (this takes quite some time, since the extract function is slow and we have to do it 16 times):

```
# The variable where we will save our results is again an initially empty list
a_admin <- list()

# We will again iterate over all combinations
for (i in 1:nrow(rusle_comb)) {
# Save the results on a temporary variable
  a_admin_i <- extract(a[[i]], admin_bound, fun = mean, na.rm = TRUE)
# For each extract we create a result data.frame that has the inforamtion of
# the administrative units the mean erosions and which RUSLE model setup used
  a_table <- data_frame(admin_level = admin_bound$KeWar_UgSu,
                        country = admin_bound$Country,
                        a = a_admin_i[,1],
                        setup = i)
# We save all simulation results in a list
  a_admin[[i]] <- a_table
}

# With "bind_rows()" we can combine the list to a table
a_admin_table <- bind_rows(a_admin)
```

We can now do again some transforamtions to our data to get further information that we can use in the plot. Again we want to know wether the calculated erosion rates are "critical" or not. This time we decide to define the mean erosion as critical if the mean of the 16 model realizations

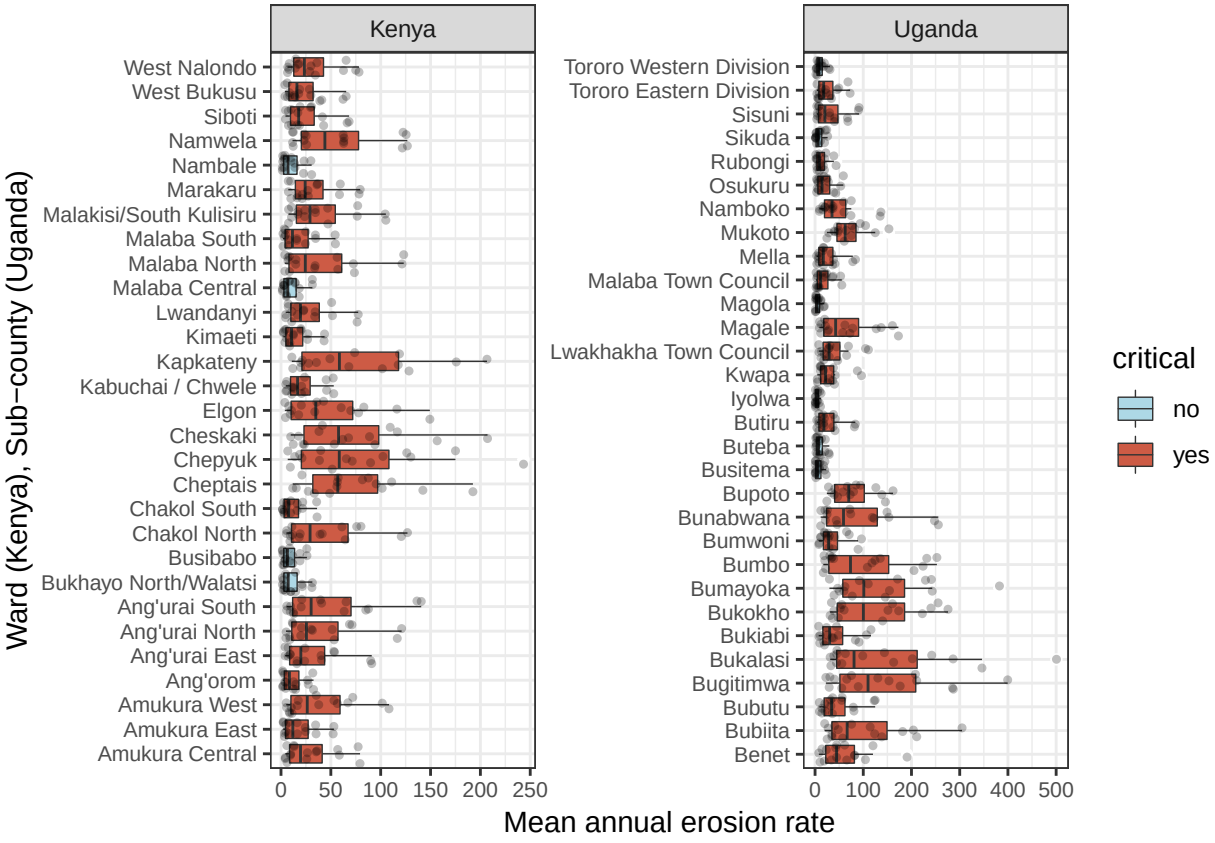
for an administrative unit was above our defined threshold of 12 $tons\,\mathrm{cot}\,ha^{-1} \cdot yr^{-1}$ [Montgomery, 2007] or not. The following lines show how "easy" it is to do the calculation in R:

```
# We group our erosion results table by the adminstrative units
a_admin_table <- group_by(a_admin_table, admin_level)
# After grouping we calculate the mean erosion for each group
a_admin_table <- mutate(a_admin_table, a_mean = mean(a))
# Finally we decide if the mean erosion is critical or not
a_admin_table <- mutate(a_admin_table,
                        critical = ifelse(a_mean > 12, "yes", "no"))
```

Visualization of the RUSLE realizations

Again, we want to make informed decisions for any measures in the administrative units; this time based on the different realizations of the soil erosion simulations. We have to re-think the idea of the bar plots we used the last time and find ways to visualize the additional information. It is always recommended to have a look at the "raw" data (e.g. in simple point plots, if the data set is not too large) before plotting it in any aggregated form (e.g. box-plots). In the following we try a visualization that combines both, the aggregated information as well as the "raw" data points:

```
ggplot(data = a_admin_table, aes(x = admin_level, y = a)) +
# As a first layer we create a boxplot for all adminsitrative units
# We color the boxes depending on whether the mean eriosion is critical or not
  geom_boxplot(aes(fill = critical), outlier.color = "white", lwd = 0.25) +
# We plot the 16 realizations over the boxes to "see the data behind the boxes"
  geom_jitter(alpha = 0.25, shape = 16, size = 1.25) +
# We use again the color schen for critical or not critical as previously
  scale_fill_manual(values = c("lightblue", "coral3")) +
# We again facet the plot for Kenya and Uganda
    facet_wrap(.~country, scales = "free") +
  coord_flip() +
    xlab("Ward (Kenya), Sub-county (Uganda)") +
# Changing the label on the y axis (now x axis)
  ylab("Mean annual erosion rate") +
# ggplot provides some template themes. The black/white theme is good
  theme_bw() +
  theme(axis.text = element_text(size = 8))
```



The red/blue boxes show the range in which 50% of the results are located (25%-75% quantiles). The vertical line in every box shows the mean result from the 16 possible combinations. The points show the erosion rate for the single results, to highlight the distribution of the raw results.

The plot shows a large variance and in consequence large uncertainty in the soil erosion estimates in most administrative units. The plot however also shows that the mean of the estimates in the administrative units is mostly above the critical level of 12 tones per hectare and year, indicating a tendency to high soil erosion risk. It can therefore be concluded that, altough large uncertainties exist, the soil erosion risk is potentially high in most of the areas.

Summary and Conclusions

In this short course we presented the procedures and methods to assess the soil erosion risk applying the RUSLE model. We highlighted the power and capacity of R to perform these analysis and environmental modelling with spatially distributed data in general. We presented a simple procedure to assess the uncertainty of the results when different input factors to the RUSLE model are used. Altough uncertainties exist, the analysis showed obvious trends towards significant soil erosion risk in most of the administrative units sharing an area with the Malaba and Malakisi River Basins.

The results were not compared to observed sediment load data in the rivers. The USLE/RUSLE is an equation that estimates average annual soil loss by sheet and rill erosion on those portions of landscape profiles where erosion, but not deposition, is occurring. It does *not estimate deposition* like that at the toe of concave slopes, and it does *not estimate sediment yield* at a downstream location. Also, it does *not include ephemeral gully erosion*. Furthermore, the USLE/RUSLE does

not provide information on sediment characteristics, such as those needed in many water quality initiatives. However, using sediment load observations, which exist for the two River Basins analysed, it would be possible to evaluate the general trends we found. This comparison must however be performed with explicit care, especially when considering the limitations of the RUSLE mentioned above. Since deposition is not considered, the modelling results will be significantly higher compared to the observations. Nevertheless the observations of sediment loads are helpful to put our results into a real-world context, highlighting the importance of maintaining a monitoring network and carrying out regular measurements. This does not only concern water quality data, but also other hydro-meteorological variables, such as river discharge, rainfall, temperature or evapotranspiration.

In the course we did not "touch" the conservation support or management practice factor P, but used a value of 1. The application and maintenance of best agricultural practises could substantially reduce the soil erosion risk. To account for these agricultural practices (e.g. contouring, strip-cropping or terracing) in the RUSLE, P factor values could be implemented to evaluate potential (large-scale) effects of improved agricultural practices. Karamage et al. [2017], for example, give values for P ranging from 0.10 to 0.20 for terracing, from 0.27 to 0.50 for strip cropping and 0.55 to 1.00 for contouring. Applying these (multiplicative) numbers to our results would lead to a substantial reduction of the soil erosion risk, probably reducing the current high mean value to a value below the critical level in most areas. This analysis shows that it is in our hands to potentially make a difference and that mitigation measures exist.

In the context of the Water-Energy-Food Nexus erosion plays a significant role. Continuous erosion leads to soil loss and in consequence reduction of agricultural output. On the other hand erosion and inadequate management practices lead to very high sediment loads in rivers and streams. Using this surface water for drinking water supply therefore leads to noteworthy higher effort and costs for purification. In the Malaba River basin, drinking water is extracted from surface waters for several drinking water schemes (Tororo / Lwakhakha / Lirima). The Lirima gravity flow scheme was constructed between 2013 and 2016 and is located in headwater area of the catchment in the proximity of Mt. Elgon National Park, with its more or less undisturbed landscape. The effort for water purification here is significantly lower, compared to the downstream schemes of Lwakhakha and Tororo. These locations are very much more affected by high sediment loads. This practical example highlights the existing interconnections and shows that improvements in agricultural management practices would have several benefits.

References

Fidele Karamage, Chi Zhang, Tong Liu, Andrew Maganda, and Alain Isabwe. Soil erosion risk assessment in Uganda. *Forests*, 8(2):52, feb 2017. doi: 10.3390/f8020052.

David R Montgomery. Soil erosion and agricultural sustainability. *Proceedings of the National Academy of Sciences*, 104(33):13268–13272, 2007.