Übung 3: Einfache Rasterdaten

CAS FAB: Räumliche Daten in R

Nils Ratnaweera

Forschungsgruppe Geoinformatik

2021-11-30

Vorbereitung

Installiere zudem das R Package `terra`

Erstelle dann ein neues R Script mit dem Namen `Uebung_3.R` und lade darin die libraries `sf` sowie `tmap`.

```
library(sf)
library(tmap)
```

Übung 3.1

- Such das digitale Höhenmodell der Schweiz (200m Auflösung)
- Auch hier kannst du die folgenden Adressen nutzen:
  - opendata.swiss
  - map.geo.admin.ch
  - swisstopo.admin.ch
- Entzippe das File (sofern nötig) und schau dir den Inhalt an

Übung 3.1

- Such das digitale Höhenmodell der Schweiz (200m Auflösung)
- Auch hier kannst du die folgenden Adressen nutzen:
  - opendata.swiss
  - map.geo.admin.ch
  - swisstopo.admin.ch
- Entzippe das File (sofern nötig) und schau dir den Inhalt an

Lösung

https://www.swisstopo.admin.ch/de/geodata/height/dhm25200.html

Shortlink (für diesen Kurs): https://bit.ly/3Hj4X0K

Input: Raster Datenformate

Inhalt des heruntergeladenen zip-Files:

Eigentliche Daten:

- DHM200_polyface.dxf
- DHM200.asc
- DHM200.xyz

Metadaten und Lizenzbedigungen:

- license.txt
- Metadata_gm03.xml
- Metadata_PDF.pdf
- Metadata_xml_iso19139.xml

Input: Raster Datenformate

Der gleiche Datensatz (`DHM25 200`) in 3 unterschiedlichen Datenformaten:

- ~~DHM200_polyface.dxf~~
- DHM200.asc
- DHM200.xyz

Input: Raster Datenformate

ESRI ArcInfo ASCII Grid

- Dateierweiterung *.asc
- ein Datenformat von ESRI (siehe die Spezifikationen)
- beginnt mit mehreren Zeilen Metaadaten, darauf folgen die eigentlichen Werte
- kann in einem Texteditor geöffnet werden:

```
NCOLS 1926
NROWS 1201
XLLCORNER 479900.
YLLCORNER 61900.
CELLSIZE 200.
NODATA_VALUE -9999.
-9999. -9999. -9999. -9999. -9999. -9999. -9999. -9999. -9999. -9999. -9999. -9999.
...
...
...
835.415 863.55 887.424 869.213 855.539 845.878 829.714 815.258 807.458 799.816 799.2
```

Input: Raster Datenformate

ASCII Gridded XYZ

- Dateierweiterung *.xyz
- Ein offenes Format
- Beinhaltet 3 Spalten: x- und y- Koordinaten sowie Zellwert
- kann in einem Texteditor geöffnet werden:

```
655000.00 302000.00 835.01
655200.00 302000.00 833.11
655400.00 302000.00 831.20
```

Input: Raster Datenformate

Um Rasterdaten in R zu importieren verwenden wir das Package `terra`.

```
install.packages("terra")
```

```
library(terra)

dhm200 <- rast("_data/original/dhm25_200/DHM200.xyz")
```

- Aus `terra` benötigen wir die Funktion `rast`
- das Importieren funktioniert gleich, unabhängig von der Dateierweiterung
- eine summarische Zusammenfassung erhält man via Konsole:

```
dhm200
```

Input: Raster Datenformate

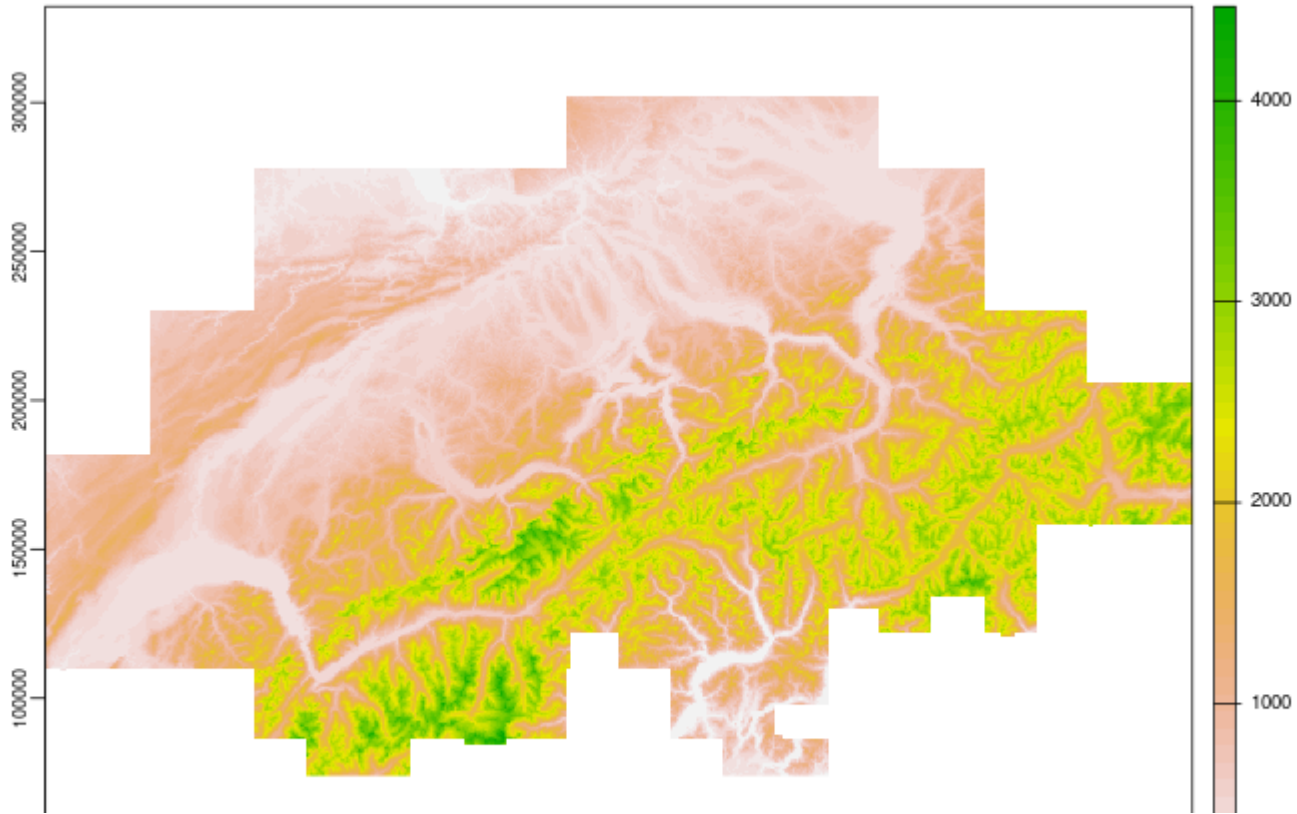
```
dhm200
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200, 200  (x, y)
## extent      : 479900, 865100, 73900, 302100  (xmin, xmax, ymin, ymax)
## coord. ref. :
## source      : DHM200.xyz
## name        :  DHM200
## min value   :     193
## max value   : 4556.63
```

Input: Raster Datenformate

Eine einfache Visualisierung erhält man mit dem `plot()` Befehl:

```
plot(dhm200)
```



Übung 3.2

In welchem Koordinatensystem befindet sich dieses Höhenmodell?

Tipp: Konsultiere die Metadaten!

Übung 3.2

In welchem Koordinatensystem befindet sich dieses Höhenmodell?

Tipp: Konsultiere die Metadaten!

Lösung

→ im alten Schweizer Koordinatensystem CH1903 LV03

Übung 3.2

Wie hoch ist die Auflösung?

Übung 3.2

Wie hoch ist die Auflösung?

Lösung

Räumliche Auflösung

Auflösung

Distanz	200

→ 200 Meter

Übung 3.3

Importiere `DHM200` in R und schau dir das Objekt in der Konsole sowie mit `plot()` an.

Übung 3.3

Importiere `DHM200` in R und schau dir das Objekt in der Konsole sowie mit `plot()` an.

Lösung

```
dhm200 <- rast("_data/original/dhm25_200/DHM200.xyz")
dhm200
plot(dhm200)
```

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:
- `st_crs(meinvektordatensatz) <- 21781` (← für das alte Schweizer Koordinatenbezugssystem)

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:
- `st_crs(meinvektordatensatz) <- 21781` (← für das alte Schweizer Koordinatenbezugssystem)
- für Rasterdaten funktioniert es leicht anders:

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:
- `st_crs(meinvektordatensatz) <- 21781` (← für das alte Schweizer Koordinatenbezugssystem)
- für Rasterdaten funktioniert es leicht anders:

```
crs(dhm200) <- "epsg: 21781"
```

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:
- `st_crs(meinvektordatensatz) <- 21781` (← für das alte Schweizer Koordinatenbezugssystem)
- für Rasterdaten funktioniert es leicht anders:

```
crs(dhm200) <- "epsg: 21781"
```

- `crs()` statt `st_crs`

Input: Koordinatenbezugssystem *festlegen*

- Das Koordinatenbezugssystem haben wir bereits für Vektordaten festgelegt
- dabei haben wir folgenden Befehl verwendet:
- `st_crs(meinvektordatensatz) <- 21781` (← für das alte Schweizer Koordinatenbezugssystem)
- für Rasterdaten funktioniert es leicht anders:

```
crs(dhm200) <- "epsg: 21781"
```

- `crs()` statt `st_crs`
- `"epsg: 21781"` (mit Anführungs- und Schlusszeichen) statt `21781`

Input: Koordinatenbezugssystem *festlegen*

```
dhm200
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200, 200  (x, y)
## extent      : 479900, 865100, 73900, 302100  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=somerc +lat_0=46.9524055555556 +lon_0=7.43958333333333 +k_0=1 +x_0=600000 +y_0=2
## source      : DHM200.xyz
## name        :  DHM200
## min value   :     193
## max value   : 4556.63
```

Input: Koordinatenbezugssystem *transformieren*

Input: Koordinatenbezugssystem *transformieren*

- Koordinatenbezugssystem von `dhm200`: `CH1903 LV03` bzw. `EPSG: 21781`

Input: Koordinatenbezugssystem *transformieren*

- Koordinatenbezugssystem von `dhm200`: `CH1903 LV03` bzw. `EPSG: 21781`
- Analog Vektordaten: in das *neue* Schweizer Koordinatenbezugssytem transformieren

Input: Koordinatenbezugssystem *transformieren*

- Koordinatenbezugssystem von `dhm200`: `CH1903 LV03` bzw. `EPSG: 21781`
- Analog Vektordaten: in das *neue* Schweizer Koordinatenbezugssytem transformieren
- Vektordaten: Funktion `st_transform`

Input: Koordinatenbezugssystem *transformieren*

- Koordinatenbezugssystem von `dhm200`: `CH1903 LV03` bzw. `EPSG: 21781`
- Analog Vektordaten: in das *neue* Schweizer Koordinatenbezugssytem transformieren
- Vektordaten: Funktion `st_transform`
- Rasterdaten: Funktion `project`

Input: Koordinatenbezugssystem *transformieren*

- Koordinatenbezugssystem von `dhm200`: `CH1903 LV03` bzw. `EPSG: 21781`
- Analog Vektordaten: in das *neue* Schweizer Koordinatenbezugssytem transformieren
- Vektordaten: Funktion `st_transform`
- Rasterdaten: Funktion `project`

```
dhm200_2056 <- project(dhm200, "epsg: 2056")
```

Input: Koordinatenbezugssystem *transformieren*

Input: Koordinatenbezugssystem *transformieren*

```
dhm200
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200, 200  (x, y)
## extent      : 479900, 865100, 73900, 302100  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=somerc +lat_0=46.9524055555556 +lon_0=7.43958333333333 +k_0=1 +x_0=600000 +y_0=2
## source      : DHM200.xyz
## name        :  DHM200
## min value   :     193
## max value   : 4556.63
```

Input: Koordinatenbezugssystem *transformieren*

```
dhm200
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200, 200  (x, y)
## extent      : 479900, 865100, 73900, 302100  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=somerc +lat_0=46.9524055555556 +lon_0=7.43958333333333 +k_0=1 +x_0=600000 +y_0=2
## source      : DHM200.xyz
## name        :  DHM200
## min value   :     193
## max value   : 4556.63
```

```
dhm200_2056
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200.0001, 200.0001  (x, y)
## extent      : 2479899, 2865099, 1073900, 1302100  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=somerc +lat_0=46.9524055555556 +lon_0=7.43958333333333 +k_0=1 +x_0=2600000 +y_0=
## source      : memory
## name        :   DHM200
## min value   :      193
```

Übung 3.4

- Transformiere `dhm200` in das Koordinatenbezugssystem `CH1903+ LV95`
- Speichere den Output als `dhm200_2056`

Übung 3.4

- Transformiere `dhm200` in das Koordinatenbezugssystem `CH1903+ LV95`
- Speichere den Output als `dhm200_2056`

Lösung

```
dhm200_2056 <- project(dhm200, "epsg: 2056")
```

Übung 3.5

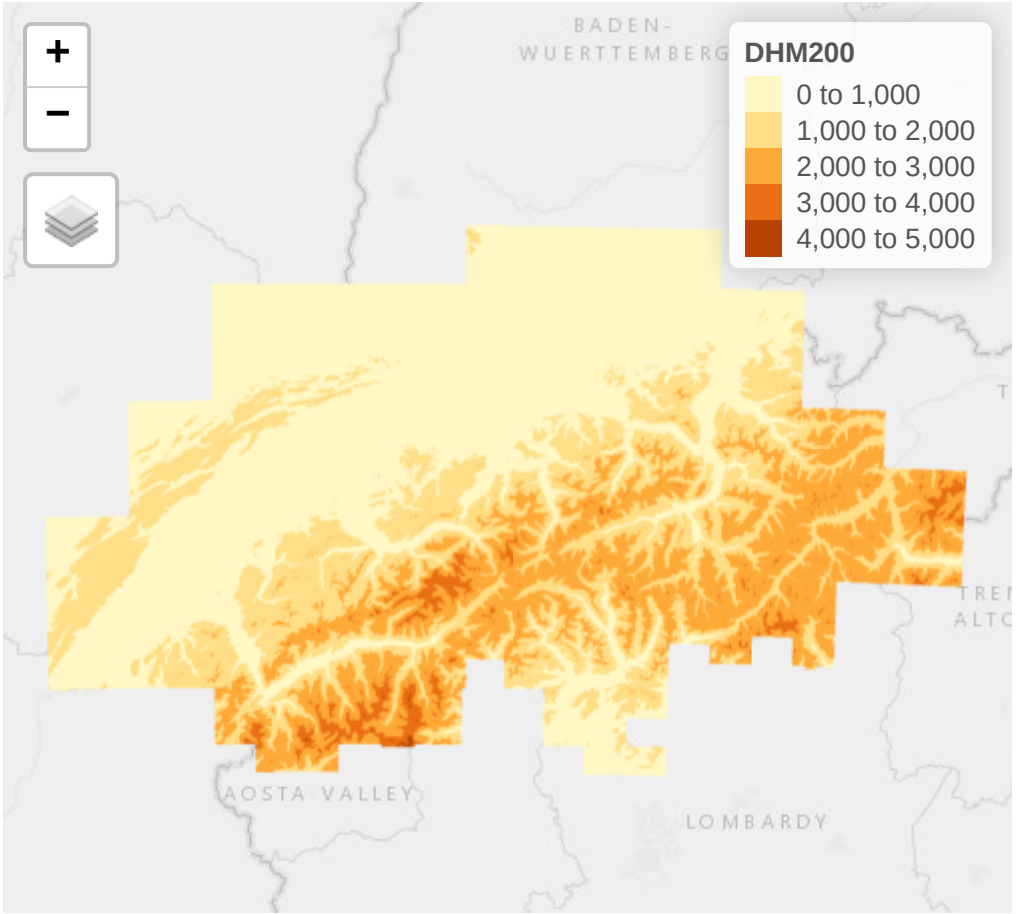
Visualisiere `dhm200_2056` mit `tmap`.

Tipp: Um ein Polygon zu visualisieren sind wir wie folgt vorgegangen

```
tm_shape(gemeindegrenzen) + tm_polygons()
```

Lösung

```
# tmap_mode("view") # optional
tm_shape(dhm200_2056) + tm_raster()
```



Übung 3.6

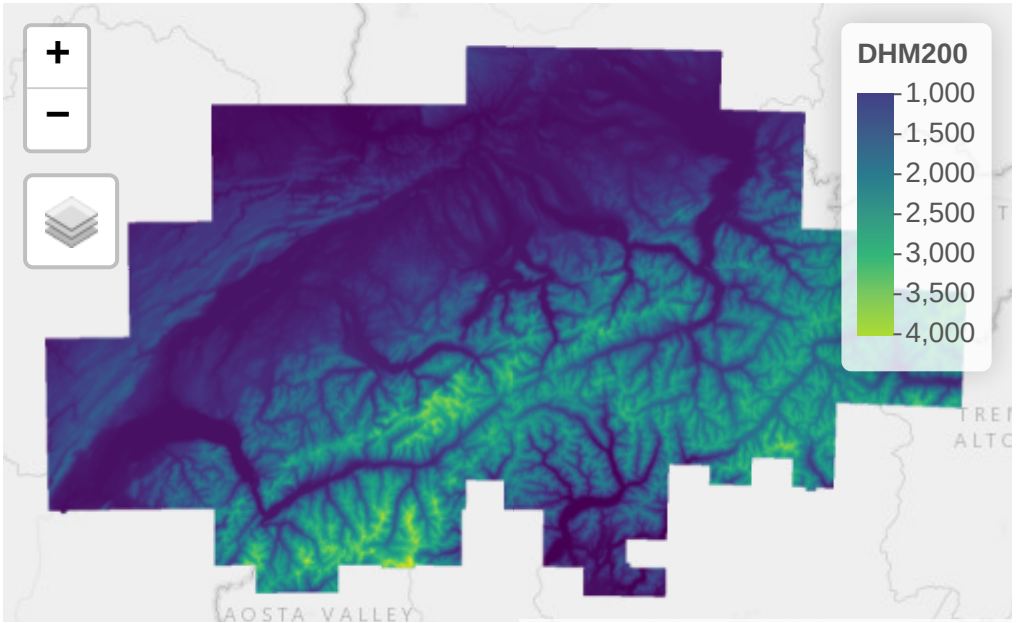
Verändere die Darstellungsweise des Rasters mithilfe von `style` und `palette`. Tipp, schau dir die Hilfe von `?tm_raster` an.

Übung 3.6

Verändere die Darstellungsweise des Rasters mithilfe von `style` und `palette`. Tipp, schau dir die Hilfe von `?tm_raster` an.

Lösung

```
tm_shape(dhm200_2056) + tm_raster(style = "cont", palette = "viridis")
```



Input

- Wir haben das DHM auf unsere Bedürfnisse angepasst (CRS gesetzt und transformiert)

Input

- Wir haben das DHM auf unsere Bedürfnisse angepasst (CRS gesetzt und transformiert)
- Wir können unser verändertes Objekt (`dhm200_2056`) exportieren, so dass diese Änderungen abgespeichert werden

Input

- Wir haben das DHM auf unsere Bedürfnisse angepasst (CRS gesetzt und transformiert)
- Wir können unser verändertes Objekt (`dhm200_2056`) exportieren, so dass diese Änderungen abgespeichert werden

```
writeRaster(dhm200_2056,"_data/processed/dhm200_2056.tif", overwrite = TRUE)
```

Input

- Wir haben das DHM auf unsere Bedürfnisse angepasst (CRS gesetzt und transformiert)
- Wir können unser verändertes Objekt (`dhm200_2056`) exportieren, so dass diese Änderungen abgespeichert werden

```
writeRaster(dhm200_2056,"_data/processed/dhm200_2056.tif", overwrite = TRUE)
```

- beim Import ist die CRS Information bekannt (CRS setzen und transformieren ist nicht mehr nötig)

```
dhm200_2056 <- rast("_data/processed/dhm200_2056.tif")

dhm200_2056
```

```
## class       : SpatRaster
## dimensions  : 1141, 1926, 1  (nrow, ncol, nlyr)
## resolution  : 200.0001, 200.0001  (x, y)
## extent      : 2479899, 2865099, 1073900, 1302100  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=somerc +lat_0=46.9524055555556 +lon_0=7.43958333333333 +k_0=1 +x_0=2600000 +y_0=
## source      : dhm200_2056.tif
## name        :   DHM200
## min value   :      193
## max value   : 4556.412
```

Übung 3.7

Exportiere `dhm200_2056` als `tif` File

Übung 3.7

Exportiere `dhm200_2056` als `tif` File

Lösung

```
writeRaster(dhm200_2056,"_data/processed/dhm200_2056.tif")
```

Rückblick

Wir haben…

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert
- diesem Rasterdatensatz das korrekte Koordinatenbezugssystem zugewiesen (`crs`)

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert
- diesem Rasterdatensatz das korrekte Koordinatenbezugssystem zugewiesen (`crs`)
- diesen Rasterdatensatz in ein anderes Koordinatensystem transformiert (`project`)

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert
- diesem Rasterdatensatz das korrekte Koordinatenbezugssystem zugewiesen (`crs`)
- diesen Rasterdatensatz in ein anderes Koordinatensystem transformiert (`project`)
- diesen Rasterdatensatz mit `plot()` sowie `tmap` visualisert

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert
- diesem Rasterdatensatz das korrekte Koordinatenbezugssystem zugewiesen (`crs`)
- diesen Rasterdatensatz in ein anderes Koordinatensystem transformiert (`project`)
- diesen Rasterdatensatz mit `plot()` sowie `tmap` visualisert
- mit verschiedenen Darstellungformen in `tmap` gearbeitet (optionen `style` und `palette`)

Rückblick

Wir haben…

- ein Höhenmodell der Schweiz heruntergeladen
- 3 unterschiedliche Datenformaten von Rasterdaten kennengelernt
- ein Rasterdatensatz mithilfe von `rast` aus `terra` in R importiert
- diesem Rasterdatensatz das korrekte Koordinatenbezugssystem zugewiesen (`crs`)
- diesen Rasterdatensatz in ein anderes Koordinatensystem transformiert (`project`)
- diesen Rasterdatensatz mit `plot()` sowie `tmap` visualisert
- mit verschiedenen Darstellungformen in `tmap` gearbeitet (optionen `style` und `palette`)
- DHM: *ein* Wert pro Zelle. Es gibt aber Situationen, wo wir mehreren Werten pro Zelle benötigen