Integration von Geodaten

CAS FAB: Räumliche Daten in R

Nils Ratnaweera

Forschungsgruppe Geoinformatik

2021-11-30

Integration von Geodaten

- Bisher haben wir alle Datensätze einzeln betrachtet

Integration von Geodaten

- Bisher haben wir alle Datensätze einzeln betrachtet
- Wenn wir alle Datensätze ins gleiche Bezugssystem bringen, können wir diese *integrieren* bzw. überlagern

Integration von Geodaten

- Bisher haben wir alle Datensätze einzeln betrachtet
- Wenn wir alle Datensätze ins gleiche Bezugssystem bringen, können wir diese *integrieren* bzw. überlagern
- Überlagern kann heissen:

Integration von Geodaten

- Bisher haben wir alle Datensätze einzeln betrachtet
- Wenn wir alle Datensätze ins gleiche Bezugssystem bringen, können wir diese *integrieren* bzw. überlagern
- Überlagern kann heissen:
  - gemeinsam Visualisieren

Integration von Geodaten

- Bisher haben wir alle Datensätze einzeln betrachtet
- Wenn wir alle Datensätze ins gleiche Bezugssystem bringen, können wir diese *integrieren* bzw. überlagern
- Überlagern kann heissen:
  - gemeinsam Visualisieren
  - Information übertragen

Übung 5.1

- Starte ein neues Script `Uebung_5.R`
- Importiere darin alle räumlichen libraries
- Importiere die Datensätze `ausserberg.gpkg` (aus Übung 2) sowie `dhm200_2056.tif` (aus Übung 3)
  - im Zip File: `_data/processed/ausserberg.gpkg` bzw. `_data/processed/dhm200_2056.tif`

Übung 5.1

- Starte ein neues Script `Uebung_5.R`
- Importiere darin alle räumlichen libraries
- Importiere die Datensätze `ausserberg.gpkg` (aus Übung 2) sowie `dhm200_2056.tif` (aus Übung 3)
  - im Zip File: `_data/processed/ausserberg.gpkg` bzw. `_data/processed/dhm200_2056.tif`

Lösung

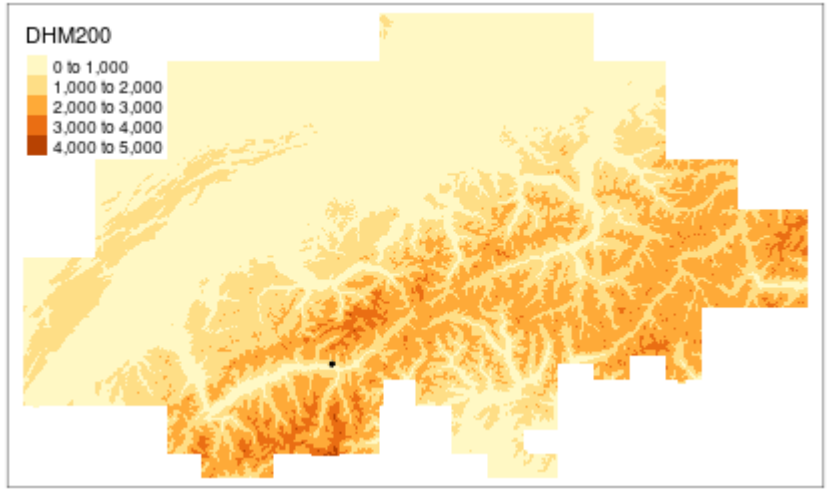
```
library(sf)
library(terra)
library(tmap)

ausserberg <- read_sf("_data/processed/ausserberg.gpkg")
dhm200 <- rast("_data/processed/dhm200_2056.tif")
```

Übung 5.2

Überlagere die beiden Datensätze in einem `tmap`-Plot, indem du diese mit `+` verkettest.

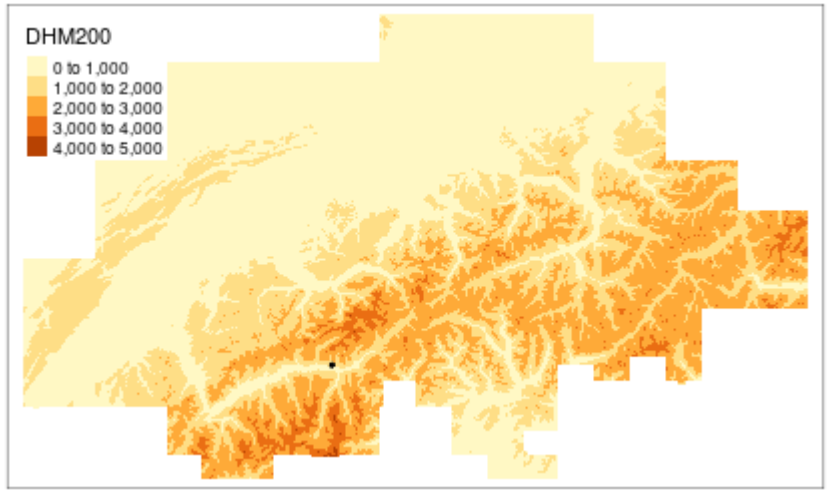
Lösung

```
tm_shape(dhm200) +
  tm_raster() +
  tm_shape(ausserberg) +
  tm_dots()
```



Lösung

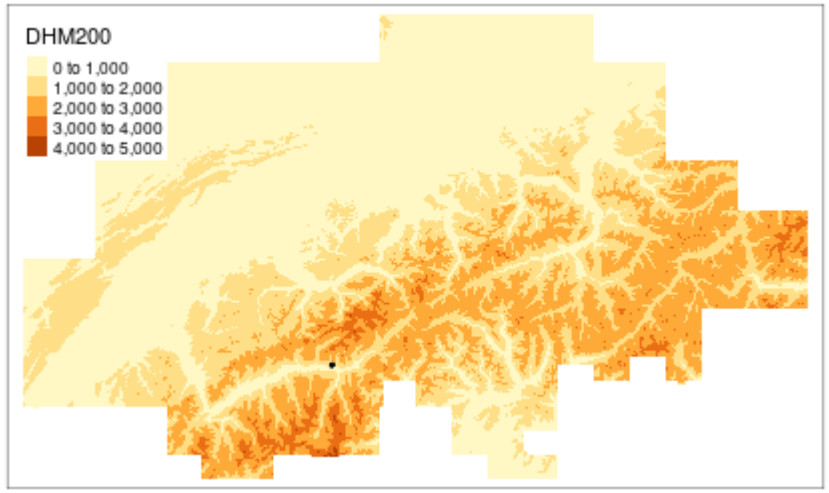
```
tm_shape(dhm200) +
  tm_raster() +
  tm_shape(ausserberg) +
  tm_dots()
```



- da das `dhm200` die ganze Schweiz abdeckt, sind unsere Punkte kaum erkennbar.

Lösung

```
tm_shape(dhm200) +
  tm_raster() +
  tm_shape(ausserberg) +
  tm_dots()
```



- da das `dhm200` die ganze Schweiz abdeckt, sind unsere Punkte kaum erkennbar.
- Lösung: raster mittels unseren Punktdaten "zuschneiden" (`crop`)

Übung 5.3

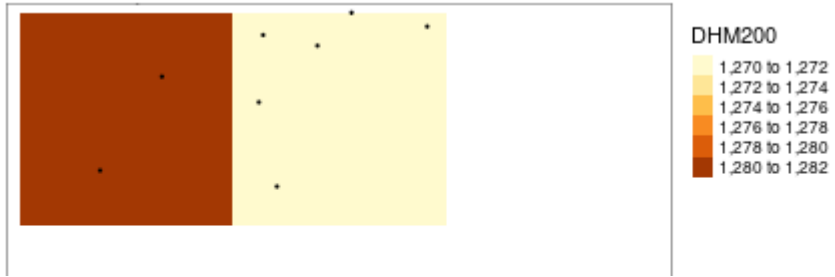
- Mit `crop()` können wir ein Raster auf den "extent" von einem Vektor Datensatz zuschneiden
- Schneide `dhm200` auf den extent von `ausserberg` zu
- Visualisiere das resultierende Raster mit `tmap` (wieder gemeinsam mit `ausserberg`)

Übung 5.3

- Mit `crop()` können wir ein Raster auf den "extent" von einem Vektor Datensatz zuschneiden
- Schneide `dhm200` auf den extent von `ausserberg` zu
- Visualisiere das resultierende Raster mit `tmap` (wieder gemeinsam mit `ausserberg`)

Lösung

```
dhm200_cropped <- terra::crop(dhm200, ausserberg)

tm_shape(dhm200_cropped) +
  tm_raster() +
  tm_shape(ausserberg) +
  tm_dots() +
  tm_layout(legend.outside = TRUE)
```



Übung 5.4

Übung 5.4

- die Auflösung des Raster Datensatzes ist zu grob!!

Übung 5.4

- die Auflösung des Raster Datensatzes ist zu grob!!
- Lösung: Hoch aufgelöster Datensatz `dhm25` (aus Übung 4) und einlesen (zip-File: `processed/dhm25_2056.tif`)

Übung 5.4

- die Auflösung des Raster Datensatzes ist zu grob!!
- Lösung: Hoch aufgelöster Datensatz `dhm25` (aus Übung 4) und einlesen (zip-File: `processed/dhm25_2056.tif`)
- wiederhole das Zuschneiden mittels `crop` sowie das Visualisieren mittels `tmap`

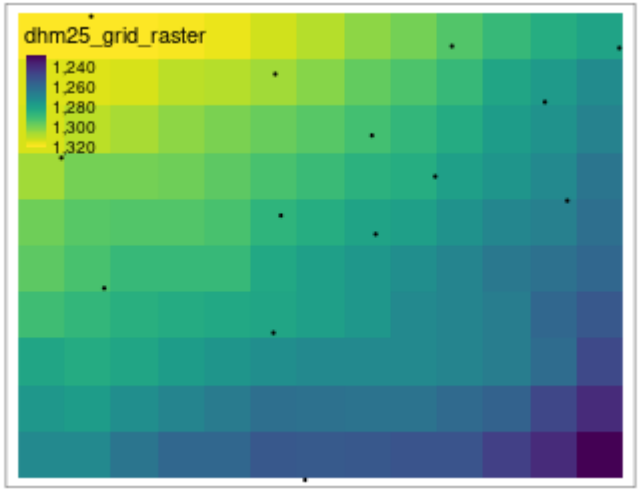
Übung 5.4

- die Auflösung des Raster Datensatzes ist zu grob!!
- Lösung: Hoch aufgelöster Datensatz `dhm25` (aus Übung 4) und einlesen (zip-File: `processed/dhm25_2056.tif`)
- wiederhole das Zuschneiden mittels `crop` sowie das Visualisieren mittels `tmap`

Lösung

```
dhm25 <- rast("_data/processed/dhm25_2056.tif")
dhm25_crop <- crop(dhm25, ausserberg)
```

```
tm_shape(dhm25_crop) +
  tm_raster(style = "cont", palette = "viridis") +
  tm_shape(ausserberg) +
  tm_dots()
```



Rasterwerte extrahieren

- bisher haben wir zwei Datensätze (Raster und Vektor) visuell überlagert

Rasterwerte extrahieren

- bisher haben wir zwei Datensätze (Raster und Vektor) visuell überlagert
- nächster Schritt: **Information** von Raster → Punkt Datensatz übertragen

Rasterwerte extrahieren

- bisher haben wir zwei Datensätze (Raster und Vektor) visuell überlagert
- nächster Schritt: **Information** von Raster → Punkt Datensatz übertragen
- dazu müssen wir `ausserberg` von einem `sp`- in ein `SpatVector` Objekt konvertieren

Rasterwerte extrahieren

- bisher haben wir zwei Datensätze (Raster und Vektor) visuell überlagert
- nächster Schritt: **Information** von Raster → Punkt Datensatz übertragen
- dazu müssen wir `ausserberg` von einem `sp`- in ein `SpatVector` Objekt konvertieren
- danach können wir das `SpatVector` Objekt gemeinsam mit `extract` verwenden

Übung 5.5

- Wandle `ausserberg` mit der Funktion `vect()` in ein `SpatVector` Objekt und speichere es als `ausserberg_vect`
- Schau dir `ausserberg_vect` an, was hat sich verändert?
- Verwende die Funktion `extract` mit `ausserberg_vect` um die Höhenwerte aus `dhm25` zu extrahieren
- Speichere den output in einer Variabel und beguteachte diese

Übung 5.5

- Wandle `ausserberg` mit der Funktion `vect()` in ein `SpatVector` Objekt und speichere es als `ausserberg_vect`
- Schau dir `ausserberg_vect` an, was hat sich verändert?
- Verwende die Funktion `extract` mit `ausserberg_vect` um die Höhenwerte aus `dhm25` zu extrahieren
- Speichere den output in einer Variabel und beguteachte diese

Lösung

```
ausserberg_vect <- vect(ausserberg)
elev <- extract(dhm25, ausserberg_vect) # <- die Funktion extract() extrahiert die Information
```

Lösung

```
elev                                    # <- der output ist eine data.frame mit 2 Spalten
```

```
##    ID dhm25_grid_raster
## 1   1           1307.219
## 2   2           1269.035
## 3   3           1284.346
## 4   4           1292.309
## 5   5           1282.125
## 6   6           1281.335
## 7   7           1320.213
## 8   8           1292.845
## 9   9           1267.855
## 10 10           1307.206
## 11 11           1277.758
## 12 12           1281.398
## 13 13           1287.539
## 14 14           1244.080
## 15 15           1294.878
```

Übung 5.6

Spiele die Höheninformation aus `extract` zurück in `ausserberg`.

Übung 5.6

Spiele die Höheninformation aus `extract` zurück in `ausserberg`.

Lösung

```
ausserberg$elevation <- elev[,2]          # <- die 2. Spalte aus elev auf ausserberg übertragen
```

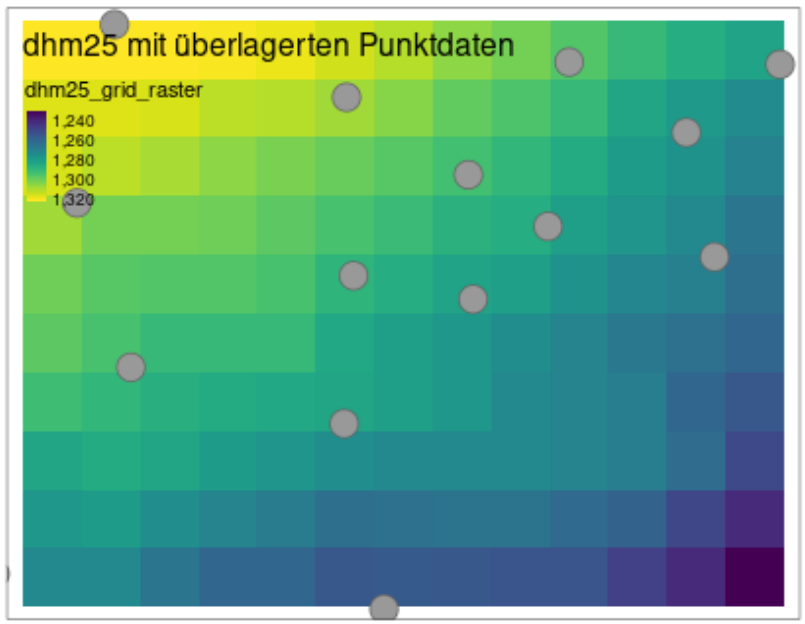
Übung 5.7

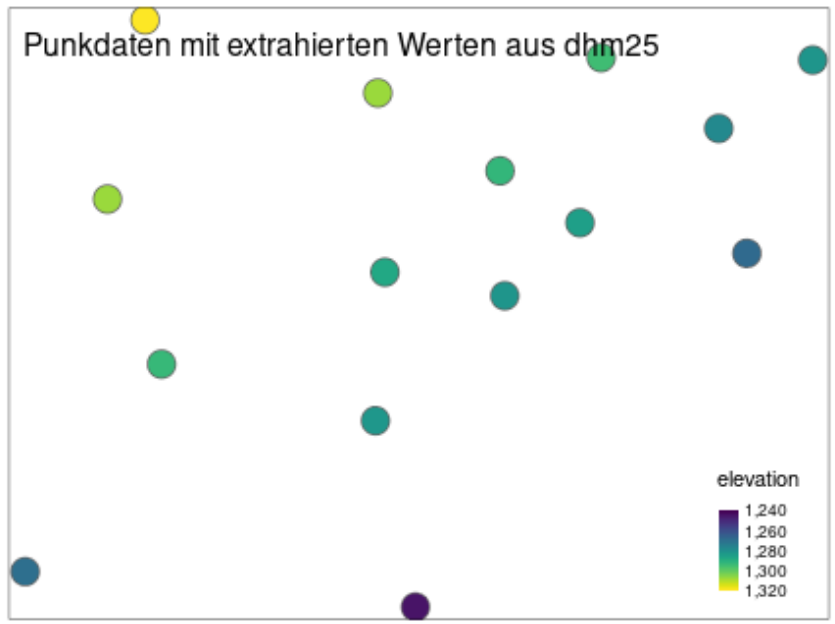
Visualisiere nun `ausserberg` und Färbe die Punkte nach ihrer Höheninformation ein.

Übung 5.7

Visualisiere nun `ausserberg` und Färbe die Punkte nach ihrer Höheninformation ein.

Lösung





Vektordaten zuschneiden

- nun wollen wir zwei Vektordatensätze miteinander verschneiden
- Ausgangslage:
  - wir verfügen über einen TWW Datensatz der Schweiz (https://bit.ly/3CqNRKT)
  - wir verfügen über den Gemeindelayer der Schweiz (https://bit.ly/3CaAj5W)
  - wir wollen alle TWW Flächen innerhalb der Gemeinde Landquart erhalten

Übung 5.8

- Lade diese beiden Datensätze herunter und importiere sie in `R` (swissboundaries *Hoheitsgebiet*)
- Transformiere sie in `ESPG 2056`

Übung 5.8

- Lade diese beiden Datensätze herunter und importiere sie in `R` (swissboundaries *Hoheitsgebiet*)
- Transformiere sie in `ESPG 2056`

Lösung

```
# Datensätze einlesen:
tww <- read_sf("_data/original/TWW/TWW_LV95/trockenwiesenweiden.shp")
hoheitsgebiet <- read_sf("_data/original/ch.swisstopo.swissboundaries3d-gemeinde-flaeche.fill/s

# Gemeindegrenzenen in EPSG 2056 transformieren und nur Landquart selektieren
hoheitsgebiet <- st_transform(hoheitsgebiet, 2056)
```

Übung 5.9

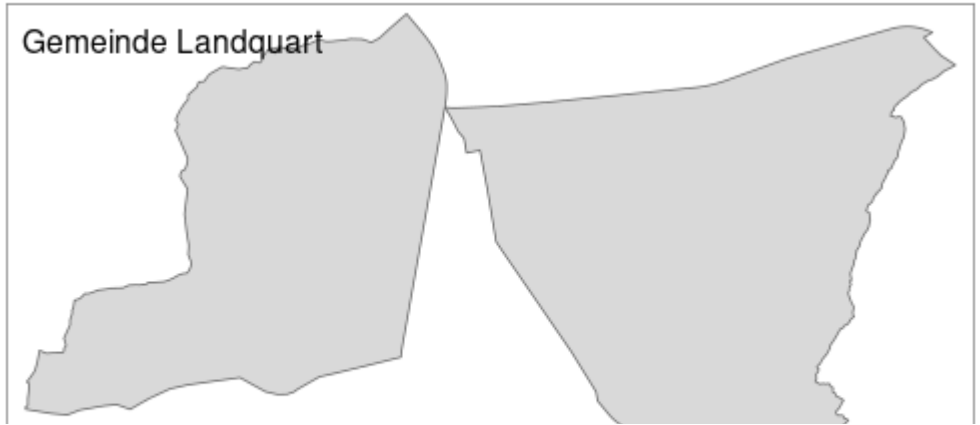
Erstelle ein neues Objekt `landquart`, welches nur die Gemeinde Landquart beinhaltet und visualisiere diese.

Übung 5.9

Erstelle ein neues Objekt `landquart`, welches nur die Gemeinde Landquart beinhaltet und visualisiere diese.

Lösung

```
landquart <- hoheitsgebiet[hoheitsgebiet$NAME == "Landquart", ]

tm_shape(landquart) +
  tm_polygons() +
  tm_layout(title = "Gemeinde Landquart")
```



Übung 5.10

Überlagere die `TWW` Flächen mit der Gemeindegrenze von Landqart.

Übung 5.10

Überlagere die `TWW` Flächen mit der Gemeindegrenze von Landqart.

Lösung

```
tm_shape(tww) +
  tm_polygons() +
  tm_layout(title = "TWW der Schweiz")  +
  tm_shape(landquart) +
  tm_polygons(col = "red",border.col = "red")
```

Übung 5.11

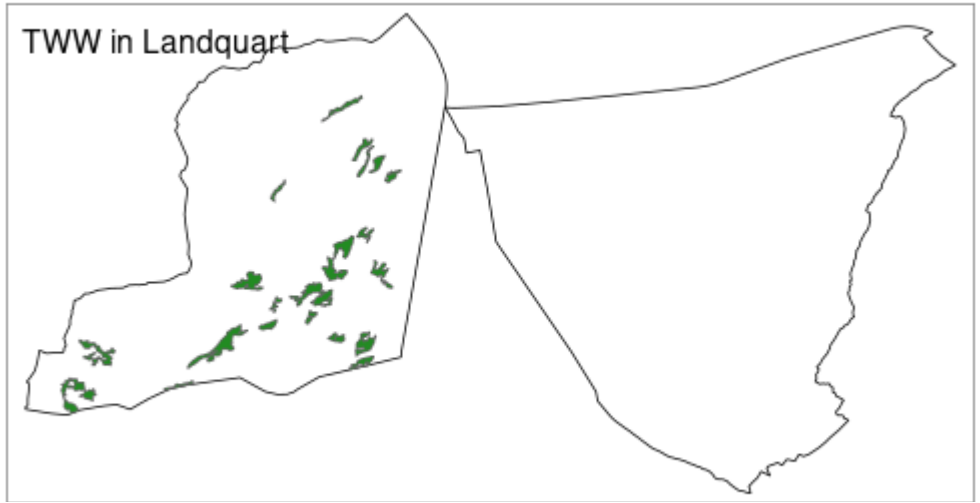
- Verwende die Funktion `st_intersection()` um die TWW-Flächen auf die Gemeindegrenze von Landquart zu zuschneiden.
- Visualisiere das Resultat

Übung 5.11

- Verwende die Funktion `st_intersection()` um die TWW-Flächen auf die Gemeindegrenze von Landquart zu zuschneiden.
- Visualisiere das Resultat

Lösung

```
tww_landquart <- st_intersection(tww, landquart)
```



Vektordaten selektieren

- Mit `st_intersection` haben wir TWW Flächen verschnitten, da `st_intersetion` die Schnittmenge beider Polygone nimmt

Vektordaten selektieren

- Mit `st_intersection` haben wir TWW Flächen verschnitten, da `st_intersetion` die Schnittmenge beider Polygone nimmt

Vektordaten selektieren

- Mit `st_intersection` haben wir TWW Flächen verschnitten, da `st_intersetion` die Schnittmenge beider Polygone nimmt

- Alternativ können wir alle TWW Flächen selektieren, die mindestens Teilweise innerhalb des Gemeindegebietes liegen

Vektordaten selektieren

- Mit `st_intersection` haben wir TWW Flächen verschnitten, da `st_intersetion` die Schnittmenge beider Polygone nimmt

- Alternativ können wir alle TWW Flächen selektieren, die mindestens Teilweise innerhalb des Gemeindegebietes liegen

```
tww_landquart2 <- tww[landquart, ]
```

Übung 5.12

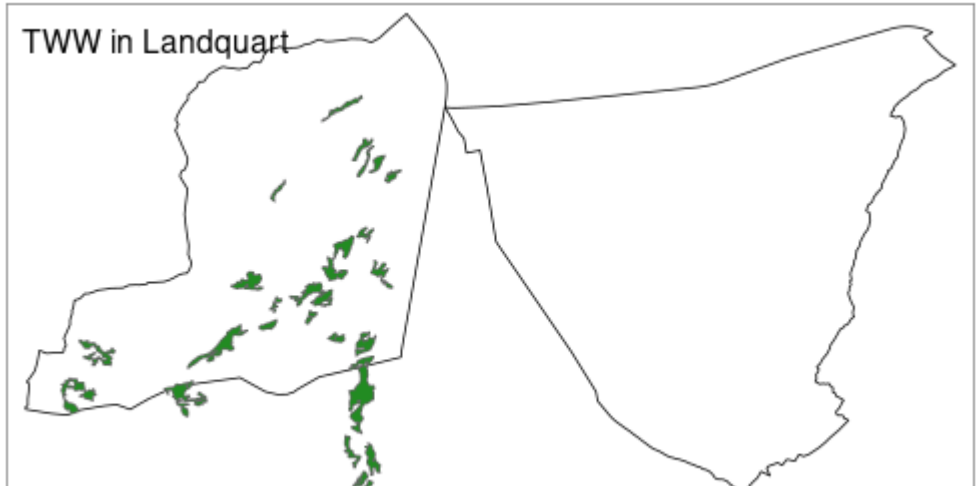
- Selektiere die TWW Flächen, welche sich zumindest Teilweise in der Gemeinde Landquart befinden und speichere den Output als `tww_landquart2`
- Visualisiere das Resultat mit `tmap`
- Vergleiche `tww_landquart2` mit `tww_landquart`. Wie unterscheiden sich diese?

Übung 5.12

- Selektiere die TWW Flächen, welche sich zumindest Teilweise in der Gemeinde Landquart befinden und speichere den Output als `tww_landquart2`
- Visualisiere das Resultat mit `tmap`
- Vergleiche `tww_landquart2` mit `tww_landquart`. Wie unterscheiden sich diese?

Lösung

```
tww_landquart2 <- tww[landquart, ]
```



Übung 5.13

Exportiere `tww_landquart2` als Geopackage

Übung 5.13

Exportiere `tww_landquart2` als Geopackage

Lösung

```
st_write(tww_landquart2, "_data/processed/tww_landquart.gpkg", delete_layer = TRUE)
```

```
## Deleting layer `tww_landquart' using driver `GPKG'
## Writing layer `tww_landquart' to data source
##   `_data/processed/tww_landquart.gpkg' using driver `GPKG'
## Writing 12 features with 12 fields and geometry type Multi Polygon.
```