Worksheet 01 · Ein kleines R-Projekt

Verzeichnisse, Skriptstruktur und kontrollierte Änderungen

Ziel

Sie richten ein kleines R-Projekt ein, trennen Eingangsdaten, Skripte und Ergebnisse und führen einen überschaubaren Analyseablauf aus. Anschließend verändern Sie einen Parameter und prüfen die Wirkung. Sie nutzen dabei relative Pfade, einen aussagekräftigen Skriptkopf und kurze Kommentare.

Beispiel: Sie laden die Gebietsgrenze des Burgwalds, erzeugen einen Puffer und speichern eine Karte sowie die berechnete Geometrie. Die Daten stammen aus dem Kursrepository LV-19-d19-006-25-R. Die Pufferberechnung dient hier als einfache Übung zur Skriptnutzung.

1 · Das Projekt anlegen

Öffnen Sie in RStudio **File → New Project → New Directory → New Project**. Nennen Sie das Projekt `burgwald-mini` und wählen Sie einen Speicherort. Öffnen Sie es künftig über `burgwald-mini.Rproj`.

Legen Sie über den Reiter **Files → New Folder** drei Unterordner an:

Ort im Projekt	Inhalt
`burgwald-mini.Rproj`	Öffnet das Projekt mit dem Projektordner als Arbeitsverzeichnis.
`data/`	Heruntergeladene Eingangsdaten; diese werden nicht überschrieben.
`scripts/`	Ihre R-Skripte.
`output/`	Berechnete Daten und Abbildungen; diese lassen sich erneut erzeugen.

Prüfen Sie in der **Konsole**:

```r
getwd()
list.files()
```

Das Arbeitsverzeichnis muss Ihr Ordner `burgwald-mini` sein. Die drei Unterordner und die Projektdatei sollten angezeigt werden. Alle folgenden Pfade beziehen sich auf diesen Projektordner, auch wenn das Skript in `scripts/` liegt. Ein zusätzliches `setwd()` ist nicht erforderlich.

Installieren Sie das benötigte Paket einmalig in der Konsole, falls es noch fehlt:

```r
install.packages("sf")
```

2 · Ein übersichtliches Skript anlegen

Erstellen Sie über **File → New File → R Script** ein Skript und speichern Sie es als `scripts/01_burgwald_puffer.R`. Übernehmen Sie den folgenden Code und ergänzen Sie Name und Datum im Kopf.

Der Aufbau bleibt einfach: **Header → Paket → Einstellungen → Daten → Verarbeitung → Ausgabe → Kontrolle.** Änderungen an Pfaden oder Parametern erfolgen gesammelt im Abschnitt „Einstellungen“.

```r
# ------------------------------------------------------------
# Titel:   Burgwald - Gebietsgrenze puffern
# Autor:   Ihr Name
# Datum:   JJJJ-MM-TT
# Zweck:   Gebietsgrenze laden, puffern und Ergebnisse speichern
# Quelle:  https://github.com/gisma/LV-19-d19-006-25-R
# Eingabe: data/burgwald_hull.GPKG
# Ausgabe: Puffer als GeoPackage und Karte unter output/
# Start:   RStudio-Projekt burgwald-mini.Rproj öffnen
# ------------------------------------------------------------

# 1. Paket ---------------------------------------------------
library(sf)

# 2. Einstellungen -------------------------------------------
puffer_m <- 100

daten_url <- paste0(
  "https://raw.githubusercontent.com/",
  "gisma/LV-19-d19-006-25-R/main/burgwald_hull.GPKG"
```

```
)
eingabe <- file.path("data", "burgwald_hull.GPKG")
ausgabe <- file.path("output", paste0("puffer_", puffer_m, "m"))

# 3. Daten laden ----------------------------------------------
# Der Download erfolgt nur, wenn die Eingabedatei noch fehlt.
if (!file.exists(eingabe)) {
  download.file(daten_url, destfile = eingabe, mode = "wb")
}
gebiet <- st_read(eingabe, quiet = TRUE)

# 4. Verarbeiten ----------------------------------------------
# UTM 32N verwendet Meter: Der Pufferabstand ist damit eindeutig.
gebiet <- st_transform(gebiet, 25832)
puffer <- st_buffer(gebiet, dist = puffer_m)

# 5. Ergebnisse speichern -------------------------------------
# Nur die gleichnamige Ergebnisdatei wird beim Wiederholen ersetzt.
st_write(puffer, paste0(ausgabe, ".gpkg"),
         delete_dsn = TRUE, quiet = TRUE)

png(paste0(ausgabe, ".png"), width = 1000, height = 800)
plot(st_geometry(puffer), col = "lightblue",
     main = paste("Burgwald - Puffer", puffer_m, "m"))
plot(st_geometry(gebiet), col = "darkgreen", add = TRUE)
legend("topright", legend = c("Gebiet", "Puffer"),
       fill = c("darkgreen", "lightblue"), bty = "n")
dev.off()

# 6. Kontrollieren --------------------------------------------
print(st_crs(gebiet)$epsg)
print(data.frame(
  gebiet_km2 = as.numeric(st_area(gebiet)) / 1e6,
  puffer_km2 = as.numeric(st_area(puffer)) / 1e6
))
print(list.files("output"))
```

Führen Sie das Skript zunächst abschnittsweise mit **Ctrl+Enter** aus. Prüfen Sie anschließend die PNG-Datei im Ordner `output`. Öffnen Sie das erzeugte GeoPackage in QGIS und vergleichen Sie es mit der Eingabedatei.

Kontrolle: Vergleichen Sie Gebiets- und Pufferfläche in der Konsole. Prüfen Sie in QGIS, dass

der Puffer die Gebietsgrenze vollständig umschließt und die Ausgabedateien unter `output/` liegen.

3 · Einen Parameter gezielt ändern

Ändern Sie ausschließlich `puffer_m <- 100` zu `puffer_m <- 500`. Schreiben Sie davor einen kurzen Kommentar, warum Sie den Parameter verändern. Führen Sie das gesamte Skript mit **Source** erneut aus.

Vergleichen Sie die beiden Karten und die ausgegebenen Flächen. Prüfen Sie außerdem, ob beide Ergebnisvarianten unter `output/` liegen und die Eingabedatei unter `data/` unverändert geblieben ist.

Dokumentieren Sie: Pufferabstand in Metern, beide berechneten Flächen und die Dateinamen der 100-m- und 500-m-Variante.

4 · Einen Pfadfehler verstehen

Ändern Sie vorübergehend die Definition von `eingabe` zu:

```
eingabe <- file.path("daten", "burgwald_hull.GPKG")
```

Führen Sie das Skript erneut aus. Lesen Sie die Fehlermeldung und prüfen Sie mit `getwd()` und `dir.exists("daten")`, worauf der Pfad verweist. Stellen Sie anschließend den richtigen Pfad wieder her.

Nutzen Sie bei Bedarf die Dokumentation oder KI. Ein passender Auftrag lautet:

> Mein RStudio-Projekt enthält die Ordner data, scripts und output. Erkläre diese Fehlermeldung und nenne die kleinste notwendige Korrektur. Ändere keine weiteren Teile des Skripts: [Fehlermeldung und betroffene Codezeilen einfügen].

Halten Sie in einem Kommentar fest, was falsch war und wie Sie die Korrektur geprüft haben. Kommentare sollen den Zweck oder eine Entscheidung erklären, beispielsweise die Wahl eines metrischen Koordinatensystems.

5 · Ohne gespeicherte Objekte erneut ausführen

Starten Sie R über **Session → Restart R** neu. Führen Sie das Skript anschließend vollständig mit **Source** aus. Es muss seine Daten selbst einlesen und alle benötigten Objekte erneut erzeugen können.

Ergebnis

Geben Sie Ihren Projektordner mit dem kommentierten Skript und den beiden Ergebnisvarianten weiter. Ergänzen Sie eine kurze Datei `README.md` im Projektordner mit vier Angaben:

- Projektdatei öffnen und anschließend `scripts/01_burgwald_puffer.R` ausführen.
- Benötigtes Paket: `sf`; Internetzugang für den ersten Datenabruf.
- Datenquelle und eigenes Abrufdatum.
- Ihre Änderung, deren beobachtete Wirkung und gegebenenfalls verwendete KI-Hilfe.

Erfolgskriterium: Eine andere Person erkennt, welche Dateien Eingangsdaten, Skripte und Ergebnisse sind, und kann den Ablauf aus einer frischen R-Sitzung erneut ausführen.