Übung: Datenanalyseumgebung einrichten

Positron, R, Python und Quarto

Prof. Dr. Nicolas Meseth

Bevor die eigentliche Datenanalyse und Schreibarbeit beginnen kann, braucht ihr eine funktionsfähige Arbeitsumgebung. Diese Übung führt euch Schritt für Schritt durch die Installation und Einrichtung aller benötigten Werkzeuge: von R und Python über die Entwicklungsumgebung Positron bis hin zu Quarto für reproduzierbare Berichte. Am Ende überprüft ihr mit einem kleinen Testszenario, dass alles reibungslos zusammenarbeitet, und denkt darüber nach, warum ihr diese Werkzeuge statt verbreiteter Alternativen wie Excel oder Word einsetzt.

Schritt 1: R installieren

R ist die primäre Sprache für diesen Kurs. Sie deckt den gesamten Analyseprozess ab: Daten einlesen, bereinigen, analysieren und visualisieren.

1. Öffnet die Website `cran.r-project.org` und ladet die aktuelle R-Version für euer Betriebssystem herunter. Führt das Installationsprogramm mit den Standardeinstellungen aus.

2. Überprüft die Installation: Öffnet ein Terminal (unter Windows z. B. PowerShell) und gebt `R.exe --version` ein. Welche Version wird angezeigt?

Schritt 2: Python installieren

Python verwenden wir als Ergänzung zu R, vor allem für den Zugriff auf KI-Modelle und Python-Bibliotheken, die keine direkte R-Entsprechung haben.

3. Ladet die neueste Version von Python von `python.org` herunter. Achtet bei der Installation unter Windows darauf, die Option *Add Python to PATH* zu aktivieren. Führt das Installationsprogramm mit den Standardeinstellungen aus.

4. Öffnet ein Terminal und überprüft die Python-Installation mit `python --version` (unter macOS und Linux ggf. `python3 --version`). Stellt sicher, dass die angezeigte Version $\geq$ 3.12 ist.

Schritt 3: Positron installieren

Positron ist eine auf Data Science ausgerichtete Entwicklungsumgebung, die auf dem VS-Code-Unterbau aufbaut und R und Python gleichwertig unterstützt. Im Vergleich zu RStudio bietet Positron modernere Editor-Features und eine bessere Python-Integration.

5. Ladet Positron von `positron.posit.co` herunter und installiert es. Startet Positron nach der Installation und prüft in der Statusleiste unten im Fenster, ob eure R-Version automatisch erkannt wurde.

6. Macht euch mit der Benutzeroberfläche vertraut. Öffnet ein neues R-Skript (`File → New File → R File`) und führt `1 + 1` in der Konsole aus. Findet heraus, wo die Bereiche *Console*, *Variables* (Umgebungsanzeige) und *Explorer* (Dateimanager) zu finden sind. Wozu werden sie jeweils verwendet?

Schritt 4: Ein Projekt einrichten

Gut organisierte Projekte sind die Grundlage reproduzierbarer Analysen. Jedes Projekt bekommt seine eigene Ordnerstruktur und eine isolierte Paketumgebung über `renv`.

7. Erstellt einen neuen Ordner `setup-test` an einem sinnvollen Ort auf eurem Computer. Öffnet diesen Ordner in Positron über `File → Open Folder`. Überprüft in der R-Konsole mit `getwd()`, dass das Arbeitsverzeichnis auf den geöffneten Ordner zeigt.

8. Initialisiert eine isolierte Paketumgebung mit `renv`. Öffnet die R-Konsole im Projekt und führt `renv::init()` aus. Überprüft anschließend mit `renv::status()`, dass die Umgebung korrekt eingerichtet wurde.

9. Installiert die folgenden Grundpakete für alle Kurse: `tidyverse`, `janitor` und `skimr`. Verwendet das Metapaket `pacman`, das ihr zuerst installieren müsst, falls es noch nicht vorhanden ist. Überprüft nach der Installation, dass alle drei Pakete fehlerfrei geladen werden können.

Schritt 5: Quarto einrichten und testen

Quarto ist das System, mit dem ihr in diesem Kurs Analysen als reproduzierbare Berichte dokumentiert. In diesem Schritt erstellt ihr ein kleines Testdokument, das gleichzeitig sicherstellt, dass R, Pakete und Quarto korrekt zusammenarbeiten.

10. Überprüft, ob Quarto verfügbar ist. Öffnet das integrierte Terminal in Positron (`Terminal → New Terminal`) und gebt `quarto --version` ein. Quarto wird normalerweise zusammen mit Positron mitgeliefert.

11. Erstellt im Projektverzeichnis eine neue Datei `setup-test.qmd` (`File → New File → Quarto Document`). Tragt im YAML-Header den Titel `"Setup-Test"` ein und setzt `format: html`. Fügt dann einen R-Code-Block hinzu, der `tidyverse` lädt und ein Streudiagramm aus dem eingebauten Datensatz `mtcars` erzeugt: Fahrzeuggewicht (`wt`) auf der x-Achse und Kraftstoffverbrauch (`mpg`) auf der y-Achse.

12. Rendert das Dokument zu HTML: Klickt auf den *Render*-Button in der Werkzeugleiste oder führt im Terminal `quarto render setup-test.qmd` aus. Öffnet die erzeugte HTML-Datei und prüft, ob der Plot korrekt angezeigt wird.

13. Rendert das Testdokument ein zweites Mal als PDF: Ändert im YAML-Header `format: html` auf `format: pdf` und rendert erneut. Quarto installiert dabei ggf. automatisch TinyTeX (eine kompakte LaTeX-Distribution). Vergleicht das PDF mit der HTML-Version: Sehen beide inhaltlich identisch aus?

Schritt 6: Typst installieren (optional)

14. Ein weiteres spannendes Schreibwerkzeug, das textbasiert arbeitet, ist Typst. Es bietet eine moderne, einfach zu erlernende Alternative zu LaTeX mit einer einfacheren Syntax und schnellerem Rendering. In diesem Buch werden wir hauptsächlich mit Quarto arbeiten, weil es wunderbar mit der Arbeit mit Daten harmoniert. Wer aber neugierig ist, kann Typst als Ergänzung installieren und ausprobieren.

Typst ist lediglich ein Komamndozeilentool, das ein in der Sprache Typst geschriebenen Dokument in ein PDF rendern kann. Ihr könnt es auf verschiedene Weise installieren. Hier geht es zum Download.

Schritt 7: Kleine Zusatzhelfer

Neben den Kernwerkzeugen gibt es einige Ergänzungen, die euch die tägliche Arbeit erheblich erleichtern.

15. Installiert Git von `git-scm.com` (Windows) oder über Homebrew (`brew install git`, macOS). Konfiguriert anschließend euren Namen und eure Hochschul-E-Mail und überprüft mit `git --version`, dass Git gefunden wird.

16. Aktualisiert den `renv`-Snapshot eures Projekts, um alle neu installierten Pakete in der Lock-Datei zu erfassen.

17. Informiert euch über KI-Assistenten für die Programmierung. Im Kurs werden wir gelegentlich *Claude Code* (Anthropic), *Codex* (OpenAI) oder *GitHub Copilot* (Microsoft) als VS-Code-Erweiterung verwenden. Recherchiert kurz, was die beiden Tools können, und notiert, wie sich KI-Assistenten eurer Meinung nach auf das Erlernen von Programmierung auswirken könnten, sowohl positiv als auch negativ.

Schritt 8: Reflexion

18. Vergleicht R mit Tabellenkalkulationsprogrammen wie Microsoft Excel. Welche Aufgaben lassen sich in Excel gut erledigen? Für welche Aufgaben ist R klar im Vorteil? Nennt mindestens zwei Stärken von R gegenüber Excel und überlegt, ob es auch Szenarien gibt, in denen Excel die bessere Wahl ist.

19. Vergleicht R (mit `ggplot2`) mit Visualisierungstools wie Tableau oder Power BI. Was können diese Tools, was R nur schwer kann? Und umgekehrt: Was spricht für den Einsatz von `ggplot2`?

20. Vergleicht Quarto mit Textverarbeitungsprogrammen wie Microsoft Word. Was sind die zentralen Unterschiede im Arbeitsablauf? Welche Vorteile bietet Quarto bei der Erstellung wissenschaftlicher oder technischer Berichte, welche Nachteile hat es?

21. Welche Bedeutung hat *Reproduzierbarkeit* für wissenschaftliche Analysen? Erläutert, wie die in diesem Experiment eingerichteten Werkzeuge (R, Quarto, renv, Git) gemeinsam dazu beitragen, dass Analysen reproduzierbar sind.