Übungsblatt 5: Grundlagen von R Markdown

Wissenschaftliches Schreiben und Arbeiten, WiSe 2023/'24

Aufgabe A: Installation von R Markdown

1. Vorausgesetzt, R ist auf deinem PC schon installiert, reicht es, folgenden Befehl in der R-Konsole auszuführen (z. B. in RStudio):

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

2. Ein zweiter Schritt ist nötig, wenn du auf deinem PC keine lokale LaTeX-Installation hast (z. B. wenn du LaTeX bisher nur über Overleaf benutzt hast). Dann führe bitte noch folgende zwei Befehle in der R-Konsole aus – andernfalls wirst du mit R Markdown keine PDFs kompilieren können:

```r
install.packages("tinytex")
tinytex::install_tinytex()
```

Aufgabe B: „Hallo Welt!“ in drei Output-Formaten

1. Erstelle eine neue .Rmd-Datei. Dies geht in RStudio ganz einfach so: Klicke im Menü „File“, „New File“, „R Markdown…“, dann „OK“.
2. Die Präambel sollte nun als Default bereits die Felder `title`, `author`, `date` und `output` enthalten. Ändere nun den Wert des `title`-Felds zu `"Mein Dokument"` und den Wert des `author`-Felds zu deinem Vor- und Nachnamen (ebenfalls in `""`-Anführungszeichen).
   Die Felder `date` und `output` kannst du erst einmal unverändert lassen. Sie sollten standardmäßig die Werte `"`r Sys.Date()`"` bzw. `html_document` haben.
3. Lösche komplett den Default-Inhalt des Hauptteils und ersetze ihn durch nur Folgendes:
   `Hallo Welt!`
4. Speichere die Datei als `aufgabe_B_<Vorname>_<Nachname>.Rmd` ab, wobei du die Platzhalter `<Vorname>` und `<Nachname>` jeweils durch deinen Vor- und Nachnamen ersetzen solltest.
5. Kompiliere die Datei (in RStudio: klicken auf den „Knit-Button“ oder Strg+Shift+K).
   Es sollte sich nach einem kurzen Moment ein Pop-up-Fenster öffnen, in dem das Resultat als HTML-Dokument (d. h. im Format einer Webseite) angezeigt wird.
   Die generierte .html-Datei (`aufgabe_B_<Vorname>_<Nachname>.html`) müsste jetzt im selben Verzeichnis liegen wie die ursprüngliche .Rmd-Datei.
6. Ändere in der Präambel jetzt den Wert von `output` zu `pdf_document`.
   Kompiliere die .Rmd-Datei nun erneut. Es wird nun stattdessen ein PDF-Dokument erzeugt (`aufgabe_B_<Vorname>_<Nachname>.pdf`).
   Schaue dir das PDF-Dokument mit einem PDF-Darstellungsprogramm an.
7. Ändere schließlich in der Präambel den Wert von `output` zu `word_document`.
   Kompiliere die .Rmd-Datei abermals. Diesmal wird ein Word-Dokument erzeugt (`aufgabe_B_<Vorname>_<Nachname>.docx`).
   Du kannst dir das erzeugte Word-Dokument mit einem Office-Programm wie Microsoft Word oder LibreOffice Writer anschauen.

Aufgabe C: Textformatierung mit der Markdown-Syntax

1. Lade dir die .Rmd-Datei `aufgabe_C_unbearbeitet.Rmd` aus dem Kurs-Moodle herunter und ändere ihren Dateinamen zu `aufgabe_C_<Vorname>_<Nachname>.Rmd`, wobei die Platzhalter hier wieder deinen Vor- und Nachnamen meinen.
2. Öffne nun die Datei `aufgabe_C_<Vorname>_<Nachname>.Rmd` in RStudio.
3. Ändere in der Präambel den Wert des Felds `author` zu deinem Vor- und Nachnamen.
4. Formatiere die einzelnen Sätze/Elemente des Rohtextes im Hauptteil nun überall so, wie vom Textinhalt selbst suggeriert. Dabei wirst du die Markdown-Syntax benutzen müssen.
5. Wenn du mit der Textformatierung fertig bist, dann kompiliere die .Rmd-Datei zu einem PDF-Dokument.
   Das erzeugte Dokument müsste den Dateinamen `aufgabe_C_<Vorname>_<Nachname>.pdf` tragen.

Aufgabe D: Ein dynamisches Dokument mit R Markdown

1. Lade dir die Tabellendatei `L1vsL2SprecherLesezeiten.csv` aus dem Kurs-Moodle herunter.
   (Sie enthält die Ergebnisse eines **fiktiven** Lesezeit-Experiments mit 300 Teilnehmenden.)
   Bemerkung: Es ist die gleiche Datei, die auch schon im letzten Übungsblatt zu R verwendet wurde.
2. Erstelle eine neue .Rmd-Datei mit `"Dynamischer Bericht"` im `title`-Feld und deinem Vor- und Nachnamen im `author`-Feld. Lösche aus der Präambel die Zeile, in der ein `date`-Feld spezifiziert wird, sofern eine solche vorhanden ist.
   Lege in der Präambel nun außerdem den Dateityp PDF als gewünschtes Output-Format fest.
   Der Hauptteil der .Rmd-Datei soll zunächst komplett leer sein.
   Speichere die so erstellte .Rmd-Datei als `aufgabe_D_<Vorname>_<Nachname>.Rmd` ab.
3. Lege gleich zu Beginn des Hauptteils fest, dass alle R-Code-Blocks, die in deiner .Rmd-Datei folgen werden, im zu kompilierenden PDF-Dokument verborgen bleiben sollen.
   Nur der von R-Code-Blocks erzeugte R-Output (bspw. Wertausgaben, Grafiken) soll also im zu kompilierenden PDF-Dokument angezeigt werden, nicht jedoch der R-Code selbst.
4. Erstelle einen R-Code-Block, in welchem die Pakete `dplyr` und `ggplot2` geladen werden und außerdem der Inhalt von `L1vsL2SprecherLesezeiten.csv` mit dem `read.csv(...)`-Befehl importiert wird, so dass das Ergebnis des Imports in einem neuen Datenrahmen `lesezeiten` gespeichert wird.
5. Schreibe darunter einen Paragraphen (entweder stichpunktartig oder in kurzen Sätzen), in welchem die statistischen Maße Minimum, Maximum, Median, Durchschnitt und Varianz der Datenspalten `lesezeiten$AlterJahre` sowie `lesezeiten$SatzLesezeitMs` berichtet werden.
   Wichtig: Schreibe die Werte der Maße im Bericht nicht manuell auf (so wie noch in Übungsblatt 4), sondern benutze die neu kennengelernte Funktionalität von R Markdown, mit der R-Ausdrücke direkt innerhalb von Fließtext eingebettet und beim Kompilieren des Dokuments ausgewertet werden können.
6. Erstelle darunter einen neuen R-Code-Block, in welchem unter Verwendung des Pakets `ggplot2` ein Box-Plot ausgegeben wird, dessen x-Achse die zwei Kategorien der Spalte `Muttersprachler` unterscheidet (d. h. `"ja"` vs. `"nein"`) und dessen y-Achse die Zahlenwerte aus der Spalte `SatzLesezeitMs` repräsentiert. Zusätzlich soll die Kategorieunterscheidung `Muttersprachler` `"ja"`/`"nein"` auch durch unterschiedliche Füllfarben hervorgehoben werden.
   Der Plot soll den Titel „Satz-Lesezeiten von Mutter- vs. Nichtmuttersprachlern“ tragen; außerdem soll die y-Achse die Beschriftung „Satz-Lesezeit in ms“ haben.
   (Hinweis: Du kannst einen solchen Box-Plot ganz nach Blaupause des **Codes für den auf Folie 78 der Vorlesung vom 1.12. gezeigten Box-Plot** erstellen.)
7. Kompiliere nun die Datei `aufgabe_D_<Vorname>_<Nachname>.Rmd` zu einem PDF-Dokument und prüfe, ob das erzeugte Dokument auch tatsächlich so aussieht, wie gewünscht.
   Ändere anschließend den Dateinamen des erzeugten PDFs vom automatisch festgelegten `aufgabe_D_<Vorname>_<Nachname>.pdf` zu `dynBericht1_<Vorname>_<Nachname>.pdf`.

8. Modifiziere die .Rmd-Datei `aufgabe_D_<Vorname>_<Nachname>.Rmd` so, dass in demjenigen R-Code-Block, in welchem der Datensatz `lesezeiten` importiert wird, direkt nach dem CSV-Import noch ein Befehl hinzukommt, der alle Beobachtungen aus dem Datensatz entfernt, für die der Wert von `AlterJahre` kleiner als 18 oder größer als 27 ist. Der so reduzierte Datenrahmen soll wieder der Variablen `lesezeiten` zugewiesen werden.

9. Kompiliere nun die Datei `aufgabe_D_<Vorname>_<Nachname>.Rmd` erneut zu einem PDF-Dokument und vergleiche den Inhalt des nun neu generierten PDFs mit der zuvor in Schritt 7 erzeugten Version. Ändere zuletzt den Dateinamen des soeben neu generierten PDFs vom wieder automatisch festgelegten `aufgabe_D_<Vorname>_<Nachname>.pdf` zu `dynBericht2_<Vorname>_<Nachname>.pdf`.

Aufgabe E: Bedingungen und Schleifen (Programmieren in R)

1. Erstelle eine neue .Rmd-Datei mit Default-Output-Format HTML. Setze in der Präambel deinen Vor- und Nachnamen als `author`, lege „Programmieraufgabe“ als `title` fest und lösche die Zeile mit dem Feld `date`, sofern vorhanden.
   Speichere die .Rmd-Datei als `aufgabe_E_<Vorname>_<Nachname>.Rmd` ab.

2. Stelle sicher, dass die .Rmd-Datei so konfiguriert ist, dass R-Code-Blocks im zu kompilierenden Dokument *sichtbar* sind (in der Regel ist dies per Default ohnehin der Fall).

3. Unterteile den Hauptteil der .Rmd-Datei in drei Abschnitte.
   Der erste soll den Titel „Beispiel Bedingungen“, der zweite den Titel „Beispiel `while`-Schleife“ und der dritte den Titel „Beispiel `for`-Schleife“ tragen.

4. Im ersten Abschnitt („Beispiel Bedingungen“):

   a. Beschreibe kurz und knapp in Worten, dass jetzt gleich ein R-Code-Block folgt, in dem eine Variable `zahl` definiert wird, der der Zahlenwert `-5` zugewiesen wird.
   b. Erstelle dann unter dieser wörtlichen Beschreibung einen einzeiligen R-Code-Block, in dem die beschriebene Zuweisung erfolgt.
   c. Beschreibe darunter in Worten, dass nun geprüft werden soll, ob die zugewiesene Zahl positiv, negativ oder null ist.
   d. Erstelle hierunter nun einen weiteren R-Code-Block, in welchem
      i. mit `if` geprüft wird, ob `zahl` positiv ist – wenn ja, dann soll `"Die Zahl ist positiv."` ausgegeben werden,
      ii. andernfalls (mit `else if`) geprüft wird, ob `zahl` negativ ist – wenn ja, dann wiederum soll `"Die Zahl ist negativ."` ausgegeben werden,
      iii. ansonsten (`else`) Folgendes ausgegeben wird: `"Die Zahl ist gleich null."`
   e. Kompiliere das Dokument einmal kurz in diesem Zwischenstand (als HTML) und prüfe, ob sich das korrekte Resultat ergibt.

5. Im zweiten Abschnitt („Beispiel `while`-Schleife“):

   a. Beschreibe in deinen eigenen Worten, dass gleich ein R-Code-Block folgt, in dem der Variablen `zahl` der Wert `1` zugewiesen wird. Außerdem werde einer weiteren Variablen `schwellenwert` der Wert `100` zugewiesen.
   b. Erstelle dann darunter, gemäß der obigen Beschreibung, einen zweizeiligen R-Code-Block, in welchem diese beiden Variablenzuweisungen vorgenommen werden.
   c. Erkläre nun in Worten, dass jetzt eine `while`-Schleife folgt, wobei in jeder Iteration (d. h. in jedem Schleifenschritt) der Wert von `zahl` erst verdoppelt und dann ausgegeben werden soll. Die Schleife solle außerdem nur solange laufen, wie `zahl` kleiner als `schwellenwert` ist.
   d. Erstelle hierunter dann also einen R-Code-Block mit einer `while`-Schleife, die solange läuft, wie `zahl` kleiner als `schwellenwert` ist.
      In jeder Iteration durch die Schleife soll der Wert von `zahl` erst einmal verdoppelt werden; anschließend soll der aktuelle Wert von `zahl` ausgegeben werden.
   e. Kompiliere das Dokument im aktuellen Zwischenstand (als HTML) und prüfe das Resultat.

6. Im dritten Abschnitt („Beispiel `for`-Schleife“):
    a. Mache zunächst mit Worten deutlich, dass gleich ein einzelner R-Code-Block folgt, der eine `for`-Schleife enthält.
    b. Erstelle gleiche darunter tatsächlich einen R-Code-Block, der eine `for`-Schleife umfasst.
    Die Schleife soll über einen Vektor der Zahlen von 1 bis 10 iterieren und während jeder Iteration die Quadratwurzel der aktuell eingelesenen Zahl ausgeben.
    Hinweis/Erinnerung: Einen Vektor der Zahlen von 1 bis 10 kann man mit `1:10` erstellen; die Quadratwurzel einer Zahl kann man mit `sqrt(...)` berechnen lassen.
    c. Kompiliere das vollständige Dokument schließlich in diesem finalen Bearbeitungsstand als HTML-Datei und prüfe, ob das Resultat deinen Erwartungen entspricht.
7. Finde das final kompilierte HTML-Dokument (`aufgabe_E_<Vorname>_<Nachname>.html`) in deinem Datei-Explorer. Öffne es dann einmal mit einem Webbrowser deiner Wahl.

Abgabe:

Lade bis zum 11.01.24 um 23:59 Uhr folgende elf Dateien in der Abgabemaske im Kurs-Moodle hoch:

- `aufgabe_B_<Vorname>_<Nachname>.Rmd`
  `aufgabe_B_<Vorname>_<Nachname>.html`
  `aufgabe_B_<Vorname>_<Nachname>.pdf`
  `aufgabe_B_<Vorname>_<Nachname>.docx`
- `aufgabe_C_<Vorname>_<Nachname>.Rmd`
  `aufgabe_C_<Vorname>_<Nachname>.pdf`
- `aufgabe_D_<Vorname>_<Nachname>.Rmd` (im letzten Bearbeitungsstand)
  `dynBericht1_<Vorname>_<Nachname>.pdf`
  `dynBericht2_<Vorname>_<Nachname>.pdf`
- `aufgabe_E_<Vorname>_<Nachname>.Rmd`
  `aufgabe_E_<Vorname>_<Nachname>.html`