Fachbereich AIHI
Universität Salzburg
Univ.-Prof. Dr. Wolfgang Trutschnig

Salzburg, März 2024
SS 23/24

Übungsblatt 03 zu 'Applied AI Using R'

Ziel der dritten Übung ist es, (nested) lists und maps weiter zu vertiefen.

Wie in der UV angemerkt, sind Listen und Funktionen essentiell für alle weiteren Schritte im Kurs. Die Übungen dienen auch dazu, etwaige noch bestehende Unsicherheiten oder Verständnisprobleme zu beheben. Ab der vierten Einheit werden verschachtelte Listen, die grundlegenden bisher gelernten `purrr` Verben und `Closures` als bekannt vorausgesetzt.

Aufgabe 11.

1. Erstellen Sie eine benannte Liste (named list) mit den folgenden Infos:
   - Red: Apple, Cherry
   - Yellow: Banana
   - Orange: Orange
   - Green: Kiwi, Grapes
   - Purple: Plum
2. Schreiben Sie Code um die folgende Information zu extrahieren:
   - ... alle Früchte der Kategorie `red` auszulesen
   - ... die zweite Frucht der Kategorie `green`
3. Ergänzen Sie die Frucht `Strawberry` in der Kategorie `red`
4. Zählen Sie die Anzahl von Früchten jeder Farb-Kategorie

Aufgabe 12.
Verwenden Sie die Zeilen 3-11 in Snippets Sheet 03 um eine Pirate Bounty map zu erstellen.
Beantworten Sie dann die folgenden Fragen (using Code, vorzugsweise `purrr` Verben), wobei Ihr Code die `legend` Information verwenden soll, um die Bedeutung der Symbole herauszulesen.

1. Wie viele sand areas hat die Insel?
2. Was sind die Koordinaten des `X` (wobei wir die Karte als 7×6 Koordinatensystem interpretieren)?
3. Wie viele Zeilen haben weder forest noch mountain?
4. Wie viele Spalten haben nur water oder sand als Eintrag?

Aufgabe 13.
Lösen Sie die folgenden zwei Tasks.

1. Schreiben sie eine higher-order Funktion names `eliminate_element(n)`, die eine Funktion retourniert, die mit einer beliebigen Liste der Länge n als Input als Output die selbe Liste ohne das letzte Element liefert (siehe Snippets Sheet 03).

2. Verwenden Sie Ihre `eliminate_element(n)` Funktion um eine Funktionen namens `eliminate_last(n)` zu erstellen, die einen Integer n als Input bekommt und eine Funktion retourniert, die eine nested list als Input nimmt, von jeder inneren Liste das letzte Element eliminiert, und die so entstehenden verbleibende nested list als Output liefert.

Aufgabe 14.
Führen Sie die Zeilen 22-46 in Snippets Sheet 03 aus, betrachten Sie die damit produzierte Grafik (und ja, auch das Staunen gehört im Studium dazu), gehen Sie den Code Zeile für Zeile durch um zu verstehen, was hier gemacht wird. Ändert sich die entstehende Struktur, wenn ein anderer Startpunkt genommen wird? Ändert sich die Struktur, wenn andere Auswahlwahrscheinlichkeiten verwendet werden?

Aufgabe 15. [i]
We consider the so-called *eight queens problem* of distributing eight queens in the chess game on a 8×8 board in a way such that no queen conflicts with any other queen. We model any arrangement of queens as an atomic vector x of length 8 with entries in $\{1, \ldots, 8\}$ such that the first queen is located in column 1 and row $x[1]$, the second queen is in column 2 and row $x[2]$, and so on.

1. Write a function `check_horizontal(config)` that takes such a configuration as input and tests if no queen conflicts with any other queen on the horizontal axis

2. Write a function `build_diagonals(row, col)` that takes a square and returns a list of tuples, each being on the two diagonals going through `(row, col)` on the board.

3. Write a function `check_diagonals(row, col)` that checks if a queen on position `(row, col)` conflics via diagonals with any other queen on the board.

4. Write a function `check_board(config)` that checks if any queen conflicts with any other queen on the board.

5. Find a brute-force solution to the eight queens problem by either enumerating all possible configurations or randomly sampling configurations and then checking them.

[i] Auf Englisch formuliert, da unter dem Namen 'eight queens problems' weltweit bekannt