UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

DIPLOMOVÁ PRÁCE

Zpracování eye-tracking dat v R

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **Mgr. Kamila Fačevicová, Ph.D.**
Vypracoval(a): **Veronika Kalabusová**
Studijní program: N0541A170026 Aplikovaná matematika
Studijní obor Aplikovaná matematika
Forma studia: prezenční
Rok odevzdání: 2022

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Veronika Kalabusová

Název práce: Zpracování eye-tracking dat v R

Typ práce: Diplomová práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: Mgr. Kamila Fačevicová, Ph.D.

Rok obhajoby práce: 2022

Abstrakt: eyetRack je nový R balíček a Shiny aplikace umožňující jednoduše analyzovat eye-tracking data, která pochází z eye-trackerů SMI nebo Tobii. Co nabízí? Základní analýzu pro úvodní představu o počtu a délkách fixací. Barploty obsahující počty fixací v jednotlivých oblastech zájmu a Dwell Time. Vizualizaci fixací se sakádami pomocí nejen metod rekurentní kvantifikační analýzy. Další možnost jak popsat eye-tracking data, přináší rekurence a rekurentní kvantifikační analýza. Rekurentní graf jako nástroj k vizualizaci rekurentních fixací však může být často při vyhodnocování výsledků příliš subjektivní. Z tohoto důvodu využíváme míry rekurentní kvantifikační analýzy, které umožňují kvantifikovat informace obsažené v rekurentním grafu. Pomocí RQA můžeme srovnávat rozdílné úkoly, či porovnávat více respondentů mezi sebou. Pro analýzu chování respondentů při provádění úkolů, můžeme využít koeficient K, který nám pomůže s rozhodnutím o ohniskové a okolní vzdálenosti.

Klíčová slova: software R, R balíček, Shiny aplikace, RQA, koeficient K

Počet stran: 86

Počet příloh: 0

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Veronika Kalabusová

Title: Eye-tracking data processing in R

Type of thesis: Master's

Department: Department of Mathematical Analysis and Applications of Mathematics

Supervisor: Mgr. Kamila Fačevicová, Ph.D.

The year of presentation: 2022

Abstract: eyetRack is a new R package and a Shiny application which facilitates the accessible analysis of eye-tracking data from SMI or Tobii eye-trackers. It offers a basic analysis for the initial conception of the number and duration of fixations. Barplot visualization allows to show a number of fixations in each Area of Interest and Dwell Time. The tool also offers visualization of the scanpath above the stimulus. The essential functionality of the application is the analysis through recurrence and recurrence quantification analysis. The recurrence plot can be displayed. However, visualization of recurrent fixations can often predispose to subjective bias when evaluating a set of results. For that reason, we used recurrence quantification analysis measures, which allow us to quantify data displayed in the recurrence plot. Using RQA, we can compare different tasks or compare multiple participants. The last functionality of the application is the calculation of coefficient K, which helps distinguish focal and ambient attention.

Key words: software R, R package, R Shiny, RQA, coefficient K

Number of pages: 86

Number of appendices: 0

Language: Czech

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracovala samostatně pod vedením paní Mgr. Kamily Fačevicové, Ph.D. a všechny použité zdroje jsem uvedla v seznamu literatury.

V Olomouci dne ....

podpis

Obsah

Seznam obrázků

Seznam tabulek

Poděkování

Na tomto místě bych ráda poděkovala vedoucí své diplomové práce Mgr. Kamile Fačevicové, Ph.D. za odborné vedení, pomoc a rady, které mi věnovala. Ráda bych také poděkovala panu RNDr. Stanislavu Popelkovi, Ph.D. za rady, které mi poskytl při tvorbě Shiny aplikace a především své rodině a přátelům, kteří mě v průběhu celého studia velmi podporovali.

Úvod

Cílem této diplomové práce je vytvoření a publikování R balíčku a Shiny aplikace pro eye-tracking data, která pochází z eye-trackerů značky SMI nebo Tobii.

Celou práci lze rozdělit do pěti kapitol. V první kapitole se nejprve seznámíme s tím, co je to programovací jazyk R, jaké jsou jeho výhody a nevýhody, co je to R balíček a jak jej můžeme jednoduše vytvořit a publikovat.

Ve druhé kapitole představíme rozšíření, které R nabízí a to Shiny aplikaci. Ve stručnosti ukážeme, jak vytvořit a publikovat jednoduchou Shiny aplikaci.

Třetí kapitola je věnovaná poznatkům z části převzatým z bakalářské práce, která byla věnována rekurentní kvantifikační analýze a na niž tato práce navazuje. V kapitole bude zaveden pojem rekurence, jaké metody rekurentní kvantifikační analýza používá a proč je vhodné míry RQA využívat.

Ve čtvrté kapitole představíme koeficient K a ukážeme, jak jej můžeme využít při rozhodování mezi ohniskovou či okolní pozorností.

Závěrečná kapitola je věnována praktické části, ve které si ukážeme všechny funkce, které R balíček pro eye-tracking data obsahuje. Pro ty, kdo chtějí využívat funkce z balíčku a vyhnout se programování v R, představíme Shiny aplikaci, se kterou si každý jednoduše poradí.

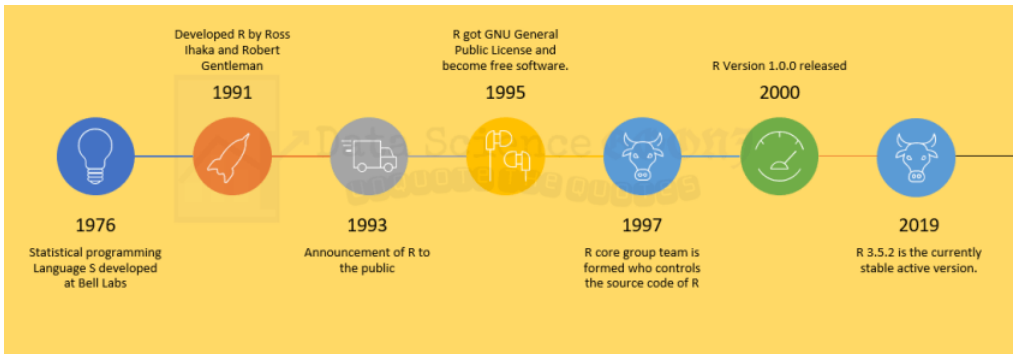
Kapitola 1

R balíček

1.1. Co je to R?

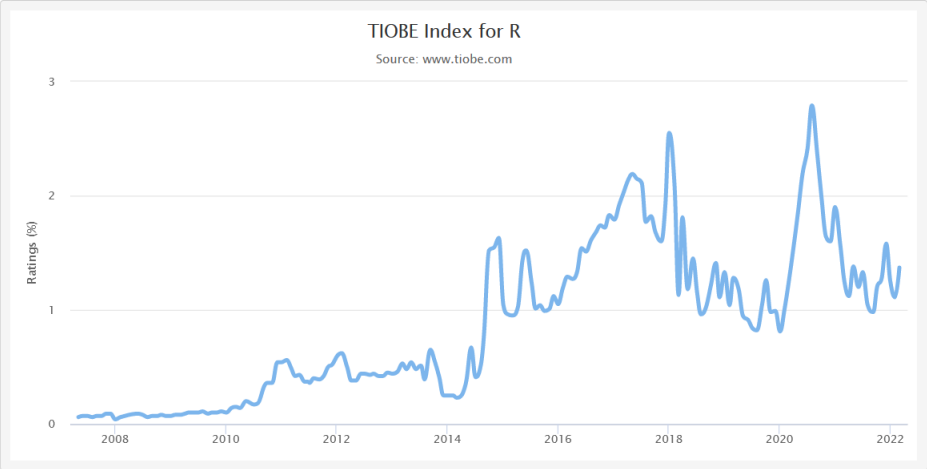
R je volně dostupný programovací jazyk, resp. softwarové prostředí, které je využíváno zejména pro analýzu dat a vizualizaci. Jazyk R je široce rozšířen v oblasti statistiky a strojového učení. R je implementací programovacího jazyku S [13]. Největším omezením jazyku S bylo, že byl dostupný pouze v komerčním balíčku S-PLUS. To vedlo k vytvoření nového bezplatného programovacího jazyku R.

Za vznikem nového prostředí stáli v roce 1991 profesoři katedry statistiky novozélandské univerzity v Aucklandu Ross Ihaka a Robert Gentleman. Písmeno R reprezentuje právě první písmena křestních jmen obou autorů [25]. V roce 1993 byl jazyk poprvé představen veřejnosti. První stabilní beta verze vznikla v roce 2000. Na obrázku 1.1 můžeme vidět další důležité milníky tohoto prostředí. V současné době se jedná o jeden z nejvíce využívaných programovacích jazyků ve statistickém prostředí.



Obrázek 1.1: Schéma nejdůležitějších milníků historie R.[9]

Během posledních let došlo u tohoto softwaru k velkému zvýšení popularity, což naznačuje i TIOBE index [1] na obrázku 1.2. Jaké je využití programování v R? R nenajdeme pouze v oblasti statistiky, má mnoho uplatnění v oblastech každodenního života. Jednou z možností, kde se setkáme s programováním v R, je bankovnictví. Banky používají jazyk R k odhalování podvodů, hodnocení klientů, vytváření modelů úvěrového rizika a provádění dalších typů analýzy rizik. Další z možností, kde se můžeme setkat s prostředím R je ve zdravotnictví. V současné (covidové) době, se R využívá například k modelování předpovědí, jak se bude nemoc v pandemii šířit [3].



Obrázek 1.2: Vývoj TIOBE indexu pro programovací jazyk R.[18]

Jako každý software, má i R své klady a zápory. Jeden z hlavních důvodů, proč se lidé z celého světa přiklání k využití R je, že se jedná o bezplatné prostředí, tudíž není nutné kupovat žádnou licenci ani platit extra poplatky za používání.

Výhodou programování v R je i otevřený zdrojový kód, tudíž kdokoliv s motivací či schopnostmi může rychle provádět úpravy a odhalovat chyby. Tím se značně zkracuje doba oproti komerčnímu provedení, kdy by bylo třeba čekat na opravu chyby prodejcem a následné vydání aktualizované verze.

Důvodem, proč využít R, je také možnost vytvořit vizuálně přitažlivé grafy (především díky balíčku `ggplot2`), které odlišují R od ostatních programovacích jazyků. Tyto výstupy poté můžeme využít při různých reportech či prezentacích.

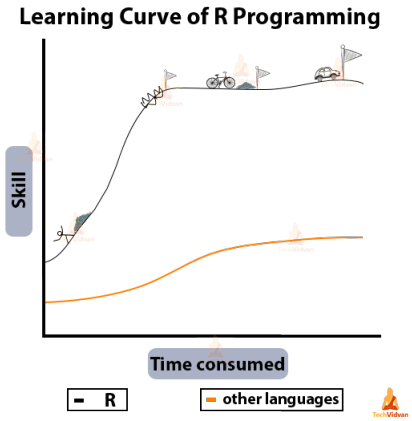
[1] TIOBE index je index popularity programovacích jazyků, který se počítá jednou za měsíc z počtu výsledků vyhledávání, které obsahují název jazyka. Index pokrývá vyhledávače jako jsou například Google, Yahoo!, Wikipedia, YouTube atd.[19]

Velkou výhodou R je, že se jedná o programovací jazyk, který je kompatibilní s jinými programovacími jazyky (jako jsou C, C++, Java, Python a další) a navíc se jedná o prostředí, které je nezávislé na platformě, proto kód napsaný na notebooku, který využívá Windows, půjde spustit bez jakýchkoliv omezení i na počítači využívajícím jiný operační systém jako je například Linux či Mac.

Slabší stránkou R je, že syntaxe jazyku R se od ostatních programovacích jazyků poměrně hodně liší. Ačkoliv se může zdát R pro začátečníka trochu obtížné (jak dokumentuje obrázek 1.3), nadšenci analýzy dat se jej čím dál tím raději učí kvůli úžasným funkcím, které R nabízí.

Dalším nedostatkem programování v R je, že při práci s daty se veškeré objekty ukládají do fyzické paměti počítače (na rozdíl například od Pythonu). Při zpracování rozsáhlých datových sad tak může dojít k zaplnění veškeré dostupné paměti.

Co je na jedné straně výhodou, může být i nevýhodou. Jelikož se na tvorbě funkcí a balíčků podílí tisíce nadšenců, může existovat několik různých možností, jak se dopracovat ke stejnému výsledku (přes rozdílné funkce a balíčky). Toto může vést k nadbytečnosti některých balíčků či jejich podprůměrné kvalitě.

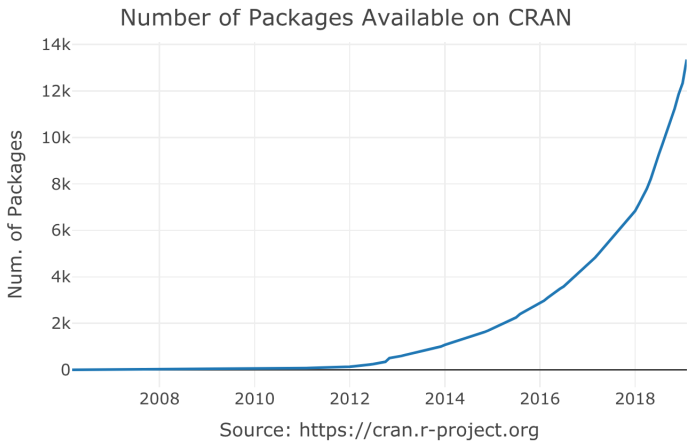


Obrázek 1.3: Jedna z nevýhod programování v R je, že na počátku je třeba mít více znalostí než u jiných programovacích jazyků [17].

1.2. R balíček

R balíček (R package) představuje rozšíření programovacího jazyka R. Jedná se o jednoduchou možnost, jak strukturovat naše kódy a funkce [25]. Obvykle se skládají z kódu, dat a dokumentace k funkcím ve standardizovaném formátu. Uživatelé si je mohou nainstalovat především prostřednictvím centralizovaného úložiště či mohou využít GitHub.

Jak ukazuje obrázek 1.4 došlo za poslední roky k velkému boomu ve zveřejňování nových balíčků. V roce 2019 bylo v centrálním depozitáři CRAN k dispozici více než 14 tisíc balíčků a rostoucí trend pokračuje i nadále. To je také jeden z důvodů, proč je R tak používaným prostředím. Je totiž velmi pravděpodobné, že se někdo jiný již zabýval problémem, který momentálně řešíme, a jeho balíček nám tak může usnadnit naši práci.



Obrázek 1.4: Jak se měnil počet R balíčků publikovaných v CRAN [8].

Proč bychom měli vytvářet R balíčky? Jedním z důvodů může být, že chceme sdílet naše funkce s ostatními. Pokud balíček zveřejníme, pak si jej může každý uživatel R nainstalovat a začít používat.

Publikování vytvořeného balíčku však není nezbytné. Balíček můžeme využívat jen pro naše účely. Vlastní funkce je vhodné zabalit do balíčku, především kvůli strukturované podobě. V případě, kdy po delší době budeme chtít použít některou z námi již dříve vytvořených funkcí, nebudeme muset zdlouhavě přemýšlet nad tím, kam jsme ji uložili a k čemu slouží. Stačí nahlédnout do popisu funkce, který je součástí balíčku a ihned víme, jaké jsou vstupní argumenty funkce, jaký výstup můžeme očekávat a k čemu je funkce určená.

Několikrát bylo zmíněno, že R balíčky mají specifickou strukturu. Co si pod tím představit? Každý balíček musí obsahovat:

- `DESCRIPTION` - soubor, který obsahuje základní informace o balíčku - název, číslo verze, o kterou se jedná, kontaktní informace na autora, kontakt na správce, na kterého je možné se obrátit v případě dotazů a v neposlední řadě také informace o tom, které balíčky jsou s ním provázané.

- `NAMESPACE` - tento soubor určuje rozhraní balíčku, které je následně prezentováno uživateli. Pomocí příkazu `export()` určíme, které funkce může uživatel využívat a v kódu volat přímo. Kromě exportů taky určuje, jaké funkce nebo celé balíčky jsou do daného balíčku importovány. Pokud váš balíček závisí na funkcích z jiného balíčku, musíme je importovat prostřednictvím souboru `NAMESPACE`.

- podadresář `man` - v tomto podadresáři najdeme soubory dokumentace pro všechny exportované objekty balíčku. U starších verzí R bylo nutné zadat dokumentace objektů přímo pomocí notace v LaTeXu, ale toto je nyní uživatelům usnadněno a to díky balíčku `roxygen2`, který umožňuje psát dokumentaci přímo do souborů s R kódem a následně automaticky generuje všechny soubory uvnitř tohoto adresáře.

- podadresář `R` - obsahuje veškerý náš R kód, který může být vložen buď v jednom souboru, či ve více souborech. Pro větší balíčky je vhodnější využít rozdělení do více souborů. Pro jednotlivé soubory platí, že nezáleží na jejich názvech, ale je lepší, pokud se v nich nevyskytují mezery.

Dále může balíček obsahovat například podadresář `data` s datovými sadami (pokud je vkládáme do balíčku) nebo delší popis balíčku známý jako `vignettes`. Tyto soubory však nejsou při tvorbě základního R balíčku nezbytné.

1.3. Jak vytvořit R balíček

Nyní si ukážeme, jak můžeme jednoduše krok za krokem nový balíček vytvořit. V této kapitole jsme vycházeli z poznatků [4] a [20].

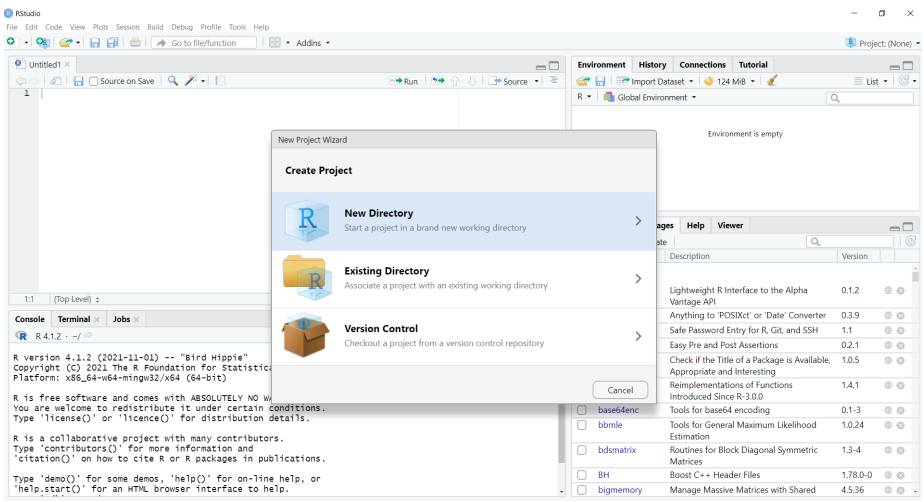
Pro ilustraci vytvoříme balíček na převod jednotek vzdálenosti, který bude obsahovat 2 funkce. První z nich bude `km2mil`, která převede vzdálenost v kilometrech na vzdálenost v mílích a funkci `mil2km`, která bude převádět vzdálenost v mílích na kilometry. Pokud do balíčku budeme vkládat funkce, které máme již připravené, je vhodné si nejprve vyzkoušet jejich funkčnost.

1. Nainstalujeme a načteme balíčky `devtools` a `roxygen2`. Balíček `devtools` umožňuje vytvoření nového balíčku a balíček `roxygen2` nám usnadní práci při vytváření nezbytné dokumentace. Pokud budeme chtít balíček publikovat, je nutné nainstalovat program `Git`[2]).

```
install.packages("devtools")
install.packages("roxygen2")

library(devtools)
library(roxygen2)
```

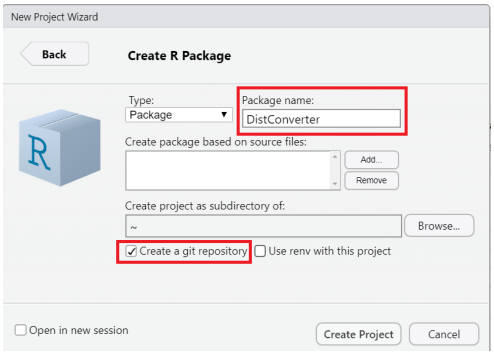
2. Vytvoříme nový projekt. File → New project. Otevře se nabídka (obrázek 1.5), kde si můžeme vybrat buď z již existujícího adresáře (pokud funkce již máme uložené), nebo vytvoříme nový adresář.



Obrázek 1.5: Printscreen obrazovky nabídky. Můžeme vytvořit nový adresář, či využít již stávající.

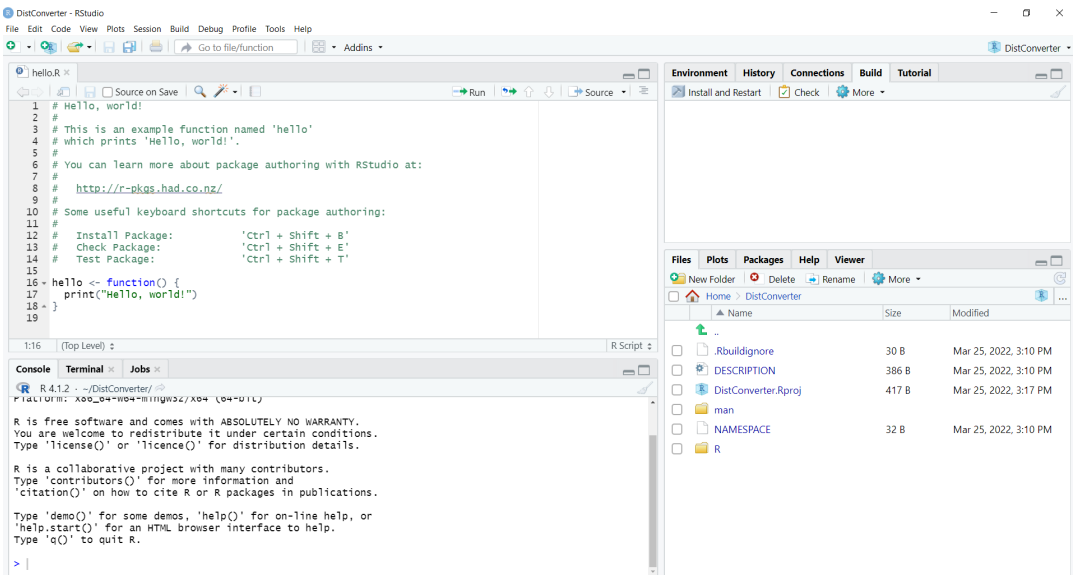
[2]Možnost stažení na odkaze: http://git-scm.com/downloads

3. Vybereme typ projektu – R package. Zvolíme název našeho nového balíčku. Jak můžeme vidět na obrázku 1.6, tak jsem náš ilustrativní balíček pojmenovala `DistConverter`. Zaškrtneme políčko `Create a git repository` (pokud budeme balíček publikovat).



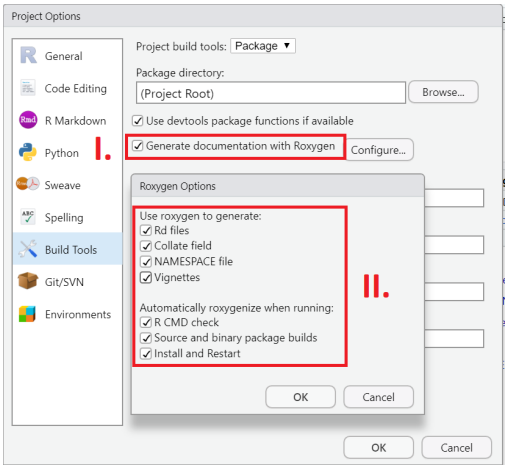
Obrázek 1.6: Printscreen okna, kde vkládáme název balíčku, popř. vybíráme podadresář.

4. Automaticky se otevře nový skript `hello.R` s ilustrativní funkcí. Na obrázku 1.7 vidíme v pravém dolním rohu strukturu balíčku, stejně jak jsme si ji popsali v kapitole 1.2, tzn. soubor `DESCRIPTION`, `NAMESPACE` a podadresáře `man` a `R`.



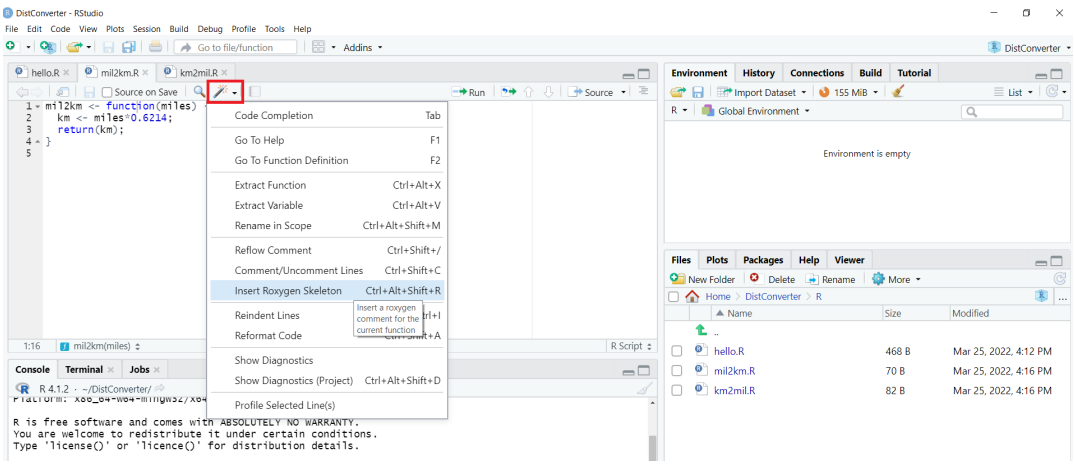
Obrázek 1.7: Rozložení RStudia po vytvoření nového projektu - v hlavní části ilustrativní funkce, v pravém dolním rohu struktura balíčku.

5. Pro automatické vytváření dokumentů, v záložce `Build` (na hlavním panelu) najdeme záložku `Configure Build Tools`, kde v záložce `Build Tools` zaškrtneme políčko Generate Documentation with Roxygen (obrázek 1.8), následně stiskneme `Configure` a zaškrtneme všechna políčka v nabídce.



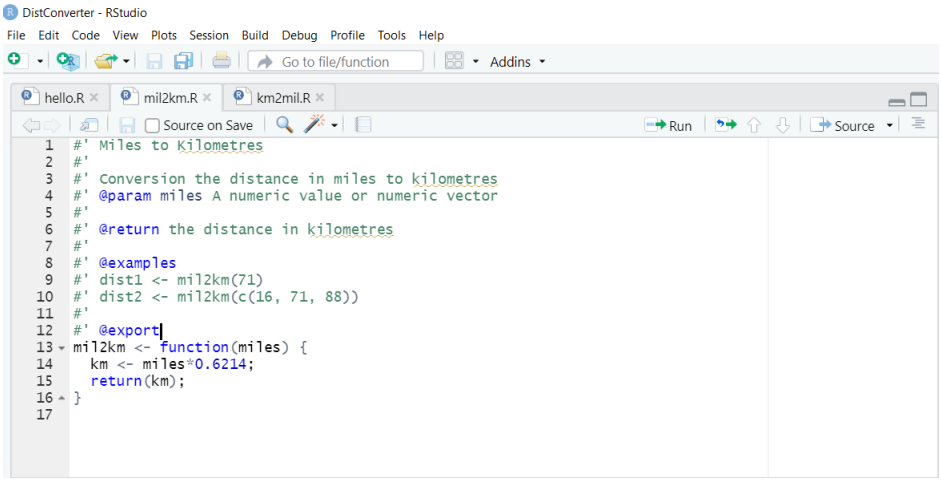
Obrázek 1.8: Printsreen záložky Configure Build Tools, kterou upravujeme.

6. Vytvoříme nové skripty s funkcemi, které bude balíček obsahovat a uložíme je do podadresáře `R`. V záložce `Code Tools` najdeme příkaz `Insert Roxygen Skeleton` a automaticky vytvoříme strukturu dokumentace k funkce (obrázek 1.9). Je nutné umístit kurzor dovnitř funkce, jinak RStudio není schopné strukturu vytvořit.



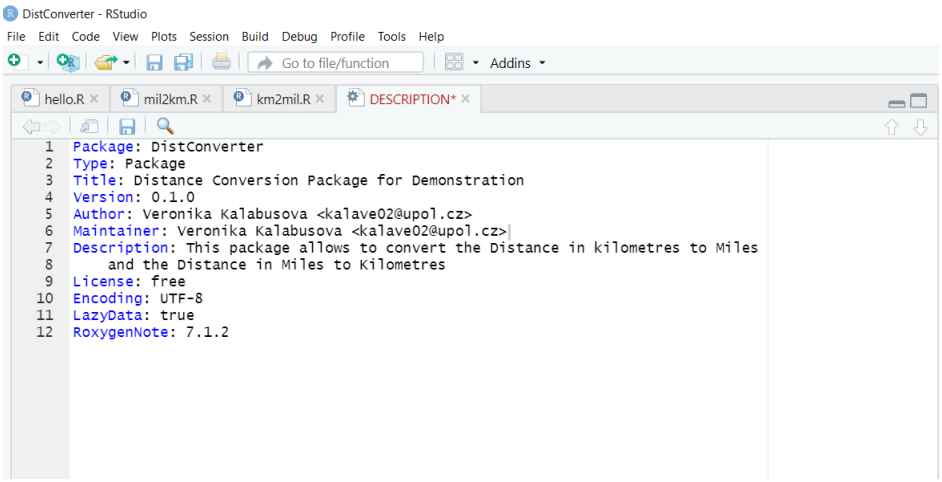
Obrázek 1.9: Příkaz k vytvoření struktury dokumentace k funkci - Insert Roxygen Skeleton

7. Upravíme dokumentaci funkcí - název funkce, její krátký popis, parametry, které funkce obsahuje a co je jejím výstupem. Pro uživatele můžeme vložit i příklady použití (obrázek 1.10). Důležité je funkci vložit u parametru `@export`, jinak by se funkce v balíčku neobjevila.



Obrázek 1.10: Printsreen úpravy dokumentace funkce mil2km.

8. Upravíme popis balíčku - soubor `DESCRIPTION`. Vložíme název balíčku, jméno autora a správce včetně kontaktu, aby bylo možné řešit případné dotazy, a detailněji popíšeme balíček. Příklad popisu balíčku vidíme na obrázku 1.10. Stejně jako v předchozích případech je nutné změny uložit.



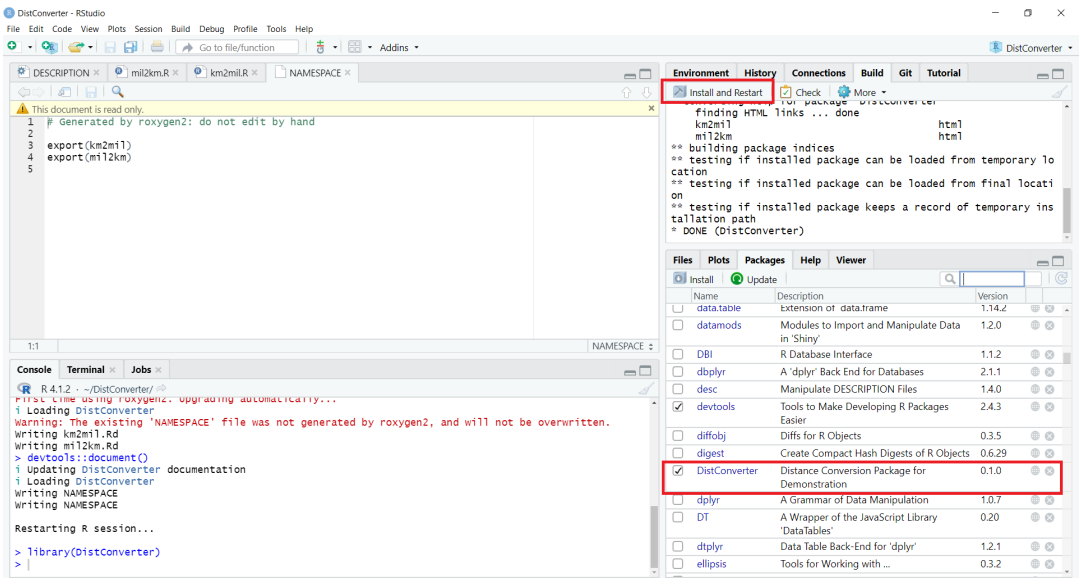
Obrázek 1.11: Detailnější popis balíčku jako součást souboru DESCRIPTION

9. Můžeme smazat vytvořený skript s funkcí `hello.R`, včetně manuálu (nechceme, aby byla součástí balíčku). Pomocí příkazu `devtools::document()` automaticky vytvoříme dokumentaci k funkcím, včetně aktualizovaného souboru `NAMESPACE`. Může se objevit varování (stejně jako na obrázku 1.12), že se soubor `NAMESPACE` nepodařilo aktualizovat. Řešením je smazání souboru ze složky a opětovné spuštění příkazu `devtools::document()`. Následně by měly být problémy odstraněny.

```
> devtools::document()
i Updating DistConverter documentation
First time using roxygen2. Upgrading automatically...
i Loading DistConverter
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be overwritten.
Writing km2mil.Rd
Writing mil2km.Rd
> devtools::document()
i Updating DistConverter documentation
i Loading DistConverter
Writing NAMESPACE
Writing NAMESPACE
>
```

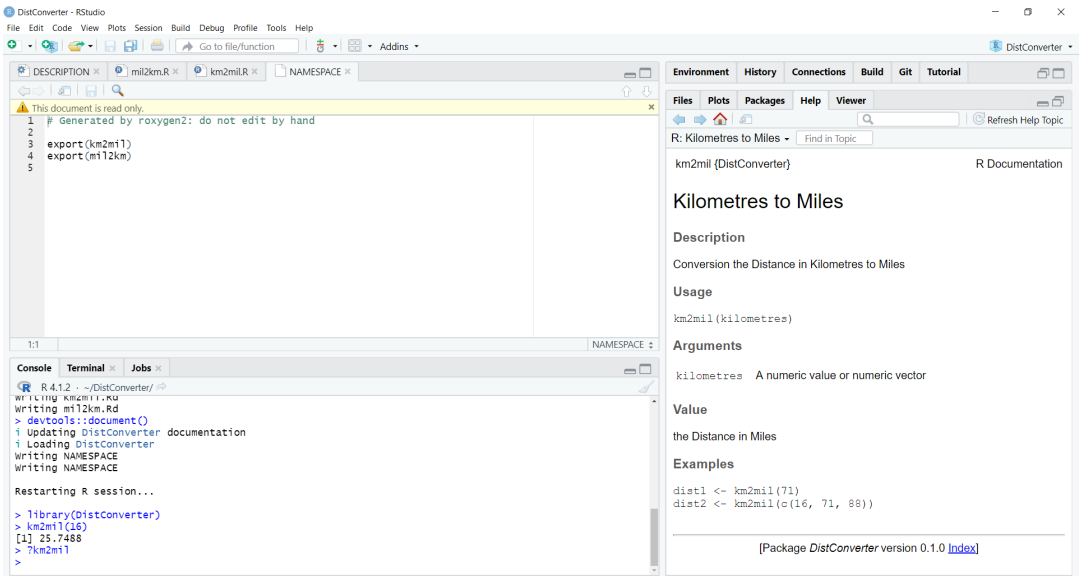
Obrázek 1.12: Warning message - soubor NAMESPACE se nepodařilo aktualizovat.

10. Pro vytvoření balíčku a umístění do seznamu balíčků stiskneme v pravém horním panelu záložku `Build` a tlačítko `Install and Restart`. Po restartu RStudia se balíček automaticky objeví v seznamu všech balíčků, které máme nainstalované (obrázek 1.13) a můžeme jej využívat.



Obrázek 1.13: Nově vytvořený balíček DistConverter se objevil v seznamu nainstalovaných balíčků.

11. Po nahrání balíčku již můžeme využívat všechny funkce, které balíček `DistConverter` obsahuje včetně dokumentace k funkcím. Dokumentaci k funkci `km2mil` vidíme na obrázku 1.14.

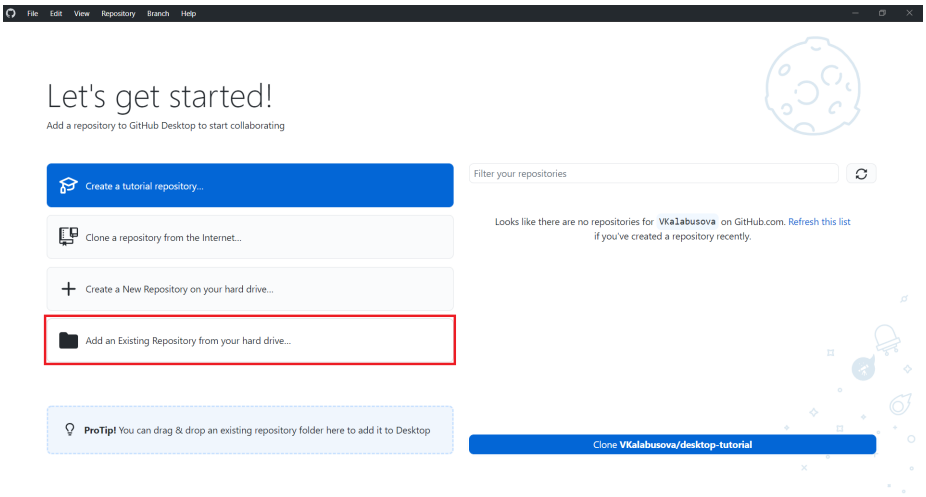


Obrázek 1.14: Po nahrání balíčku již můžeme využívat funkce, které balíček obsahuje.

1.4. Jak publikovat R balíček?

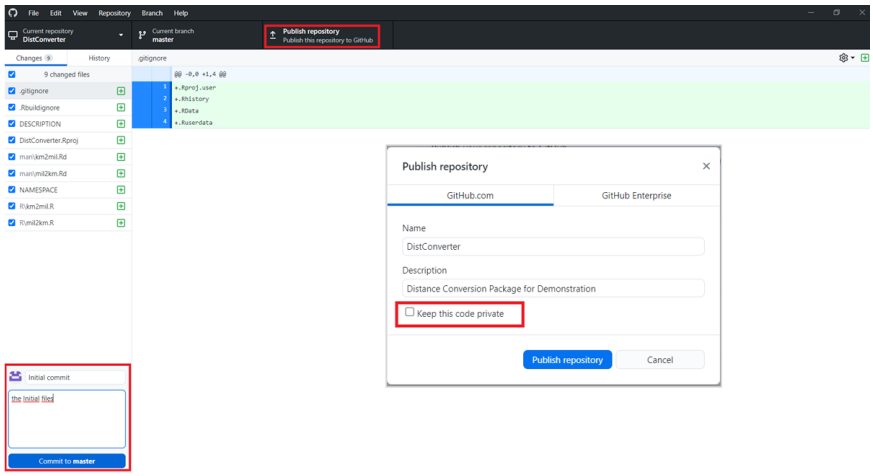
Pokud se rozhodneme náš balíček sdílet, pak máme několik možností, kam jej umístit. V této kapitole si ukážeme, jak balíček publikovat na serveru GitHub. Vycházeli jsme především z návodu [10]. Pro zjednodušení je vhodné nainstalovat aplikaci `GitHub Desktop`.

1. Vytvoříme si účet na GitHub.com. Otevřeme aplikaci `GitHub Desktop` a vybereme možnost vytvoření depozitáře z již existující složky v počítači, protože při vytváření balíčku jsme zaškrtli políčku `Create a git repository` (viz. obrázek 1.6).



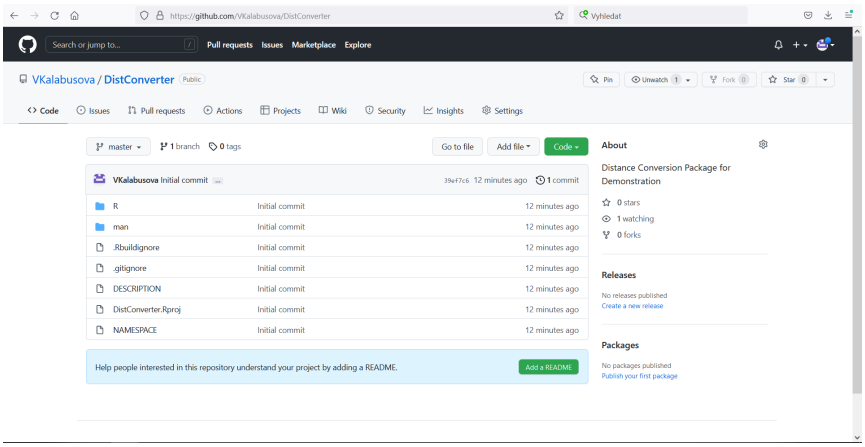
Obrázek 1.15: Úvodní stránka aplikace GitHub Desktop.

2. Na levé straně obrazovky se zobrazí všechny soubory, které jsou součástí balíčku. V levém dolním rohu lze volitelně označit všechny soubory za primární, abychom při následných úpravách věděli, ze kterých souborů jsme vycházeli (není nutné). Následně na hlavním panelu stiskneme tlačítko `Publish repository`. Otevře se nám vyskakovací okno, kam přidáme krátký popis balíčku. Pokud chceme balíček sdílet, pak nezapomeňme odškrtnout políčko políčko `Keep this code private`.



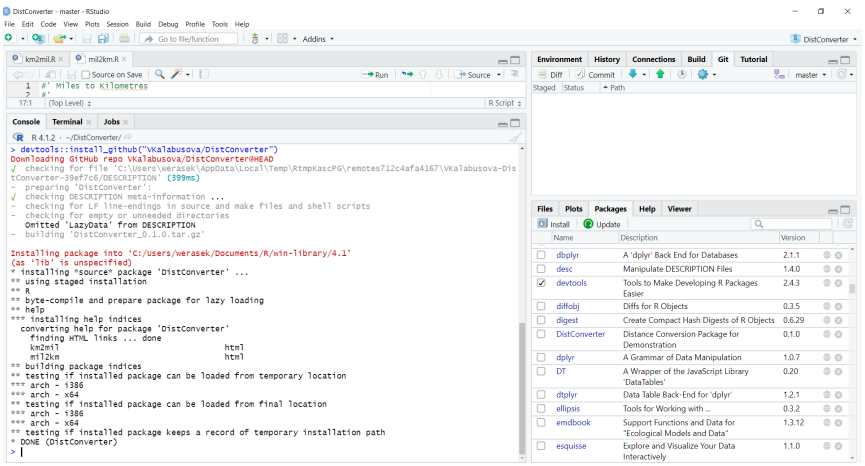
Obrázek 1.16: Publikování balíčku na server GitHub.

3. Po přihlášení na GitHub.com v záložce `Your repositories` již vidíme depozitář `DistConverter`, po jeho rozkliknutí se objeví soubory, které jsou součástí depozitáře, resp. našeho balíčku - kódy funkcí, `NAMESPACE`, `DESCRIPTION`, ...)



Obrázek 1.17: Soubory, které jsou součástí depozitáře (balíčku) DistConverter.

4. Nyní je již balíček k dispozici ke stažení pomocí příkazu `devtools::install_github("<autor>/<název balíčku>")`. v našem případě bychom použili příkaz:
`devtools::install_github("VKalabusova/DistConverter")`, tím se balíček nainstaluje a můžeme všechny funkce, které obsahuje, začít používat.



Obrázek 1.18: Stažení publikovaného balíčku DistConverter.

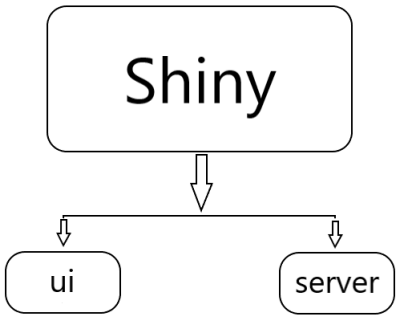
Kapitola 2

Shiny

2.1. Co je to Shiny?

Shiny je R balíček, který představuje rozšíření softwaru R pro tvorbu interaktivních webových aplikací. Takto vytvořené aplikace přináší pro uživatele příjemné prostředí, se kterým může pracovat i neznalec programovacího jazyku R. Není třeba nic programovat. V kapitole o Shiny budeme vycházet především z publikací [24], [15] a poznámek z předmětu PREZR, který vedla paní Mgr. Kamila Fačevicová, Ph.D. Každá Shiny aplikace, jak ukazuje obrázek 2.1, se skládá ze 2 základních částí:

- `ui` (user interface) - uživatelské rozhraní, které řídí rozvržení a vzhled naší aplikace,
- `server` - server obsahuje pokyny, které probíhají na pozadí počítače.



Obrázek 2.1: Shiny aplikace je tvořena ze 2 částí - ui a serveru.

2.2. Uživatelské rozhraní - ui

Uživatelské rozhraní určuje, jak bude naše aplikace vypadat. Vybíráme si zde, jaké rozložení bude mít Shiny aplikace, jaký bude její název, jaké máme volitelné parametry a v neposlední řadě také to, jaký obsah se nám zobrazí.

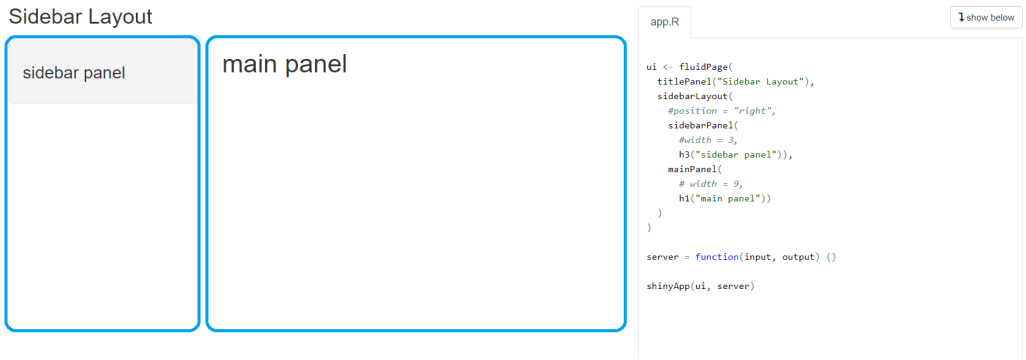
2.2.1. Rozvržení Shiny aplikace

Prvním z několika zásadních kroků při tvorbě uživatelského rozhraní, je volba rozložení stránky aplikace. Můžeme volit z několika možností, které mezi sebou můžeme různě kombinovat. Je však důležité, aby každá z vnořených funkcí byla oddělena čárkou.

Nejprve se musíme rozhodnout, zda chceme, aby se velikost strany aplikace přizpůsobovala velikosti okna, ve kterém je otevřena. K tomu slouží funkce `FluidPage()`. Pokud bychom chtěli mít velikost fixně danou, pak použijeme funkci `FixedPage()`. Nyní si představíme nejzákladnější rozvržení aplikací, které Shiny nabízí.

Sidebar Layout

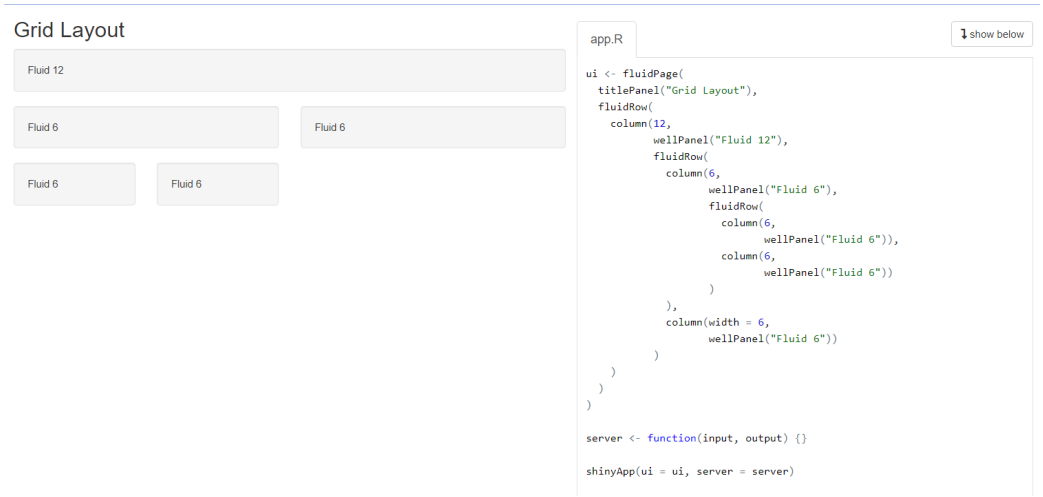
Sidebar Layout bývá často základním nastavením většiny aplikací. Toto rozvržení rozděluje stránku na 2 části. První část, tzv. sidebarPanel, tvoří 1/3 strany a většinou obsahuje vstupní proměnné. Druhá část (mainPanel) tvoří 2/3 strany a je určena především pro výstupy analýzy. Výchozím nastavením je umístění postranního panelu vlevo, avšak pomocí parametru `position` můžeme jeho umístění jednoduše změnit. Také šířka částí není striktně daná a můžeme ji měnit pomocí parametru `width`, celková šířka obou panelů však musí dát dohromady hodnotu 12. Rozvržení stránky aplikace pomocí sidebar Layoutu můžeme vidět na obrázku 2.2.



Obrázek 2.2: Rozložení stránky aplikace pomocí sidebar Layoutu.

Grid Layout

Větší flexibilitu při rozdělování jednotlivých oblastí stránky dává uživatelům rozvržení pomocí "mřížky". Nejprve pomocí funkce `FluidRow()` přidáme do naší aplikace řádek a následně daný řádek dělíme na sloupce s požadovanou šířkou pomocí příkazu `column()`. Šířku sloupců upravujeme pomocí parametru `width`. Celková šířka sloupců uvnitř každé funkce `FluidPage()` musí dát hodnotu 12. Toto rozložení nám umožňuje vytvořit libovolnou strukturu stránky. Na obrázku 2.3 můžeme vidět rozložení stránky na 3 řádky, kde každý řádek je tvořen sloupci s poloviční šířkou, než měl na předchozím řádku.



Obrázek 2.3: Rozložení stránky aplikace pomocí FluidRow.

Tabset Layout

Uživatelé často potřebují rozdělit svoji aplikaci do samostatných sekcí. K tomu je ideální funkce `tabsetPanel()`, kterou můžeme kombinovat s rozdělením `sidebarLayout()`. Tato funkce nám umožňuje rozdělit mainPanel pomocí záložek do několika samostatných částí, kam každou záložku přidáváme pomocí funkce `tabPanel()`. Do funkce `tabPanel()` píšeme název, který se zobrazí uživateli společně s obsahem dané záložky. Na obrázku 2.4 vidíme příklad rozdělení aplikace na 3 části - Tab1, Tab2 a Tab3.

Obrázek 2.4: Rozložení stránky aplikace pomocí tabsets.

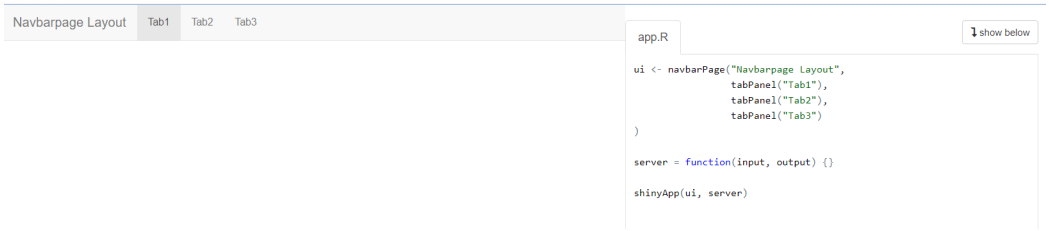
Navlist Layout

V případě, že máme delší názvy jednotlivých záložek, či jich máme více, je vhodné použít funkci `navlist()`. Navlist umožňuje uspořádat jednotlivé záložky pod sebe, jak je ukázáno na obrázku 2.5. Jednotlivé záložky vkládáme pomocí funkce `tabPanel()`. Mezi záložky můžeme vkládat různé nadpisy či další text.

Obrázek 2.5: Rozložení stránky aplikace pomocí Navlists.

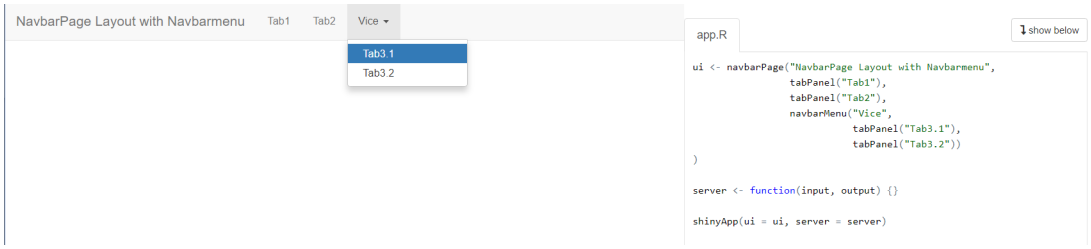
NavbarPage

Oblíbené rozložení aplikace je pomocí funkce `navbarPage()`, kde jsou záložky umístěny v horním panelu. Velkou výhodou je, že jednotlivé záložky mohou mít své vlastní rozložení stránky. Na obrázku 2.6 můžeme vidět rozložení aplikace pomocí funkce `navbarPage()`.



Obrázek 2.6: Rozložení stránky aplikace pomocí Navbarpage.

Pomocí funkce `navbarMenu()` můžeme přidat druhou úroveň záložek. Do horního panelu tím přidáme nabídku, která může odkazovat na další záložky (obrázek 2.7.



Obrázek 2.7: Rozložení stránky aplikace pomocí Navbarpage s více úrovněmi záložek.

2.2.2. Vstupní parametry

Důležitou součástí vytváření uživatelského rozhraní naší aplikace je nastavení vstupních parametrů a možností, z nichž následně může uživatel tyto vstupy volit. Hodnoty vstupních parametrů se ukládají do proměnné `input`, se kterou budeme pracovat v části `server` naší aplikace. Každý vstupní prvek obsahuje argumenty:

1. inputID - název, pod kterým se náš výběr uloží a pod kterým s ním budeme pracovat na serveru. Každý input musí mít jiné ID. Pokud jako název proměnné zadáme například `"number"`, potom se na hodnotu této proměnné budeme na serveru odkazovat pomocí příkazu `input$number`,

2. label - popis inputu, který uživatel v aplikaci vidí. Příkladem popisku u číselné proměnné může být "Vyberte své oblíbené číslo".

Většina vstupů má i další parametry, které můžeme měnit, nejsou však již pro všechny shodné. Pro zjištění všech parametrů, které daný input obsahuje, můžeme využít `help` funkce - například `?selectInput`.

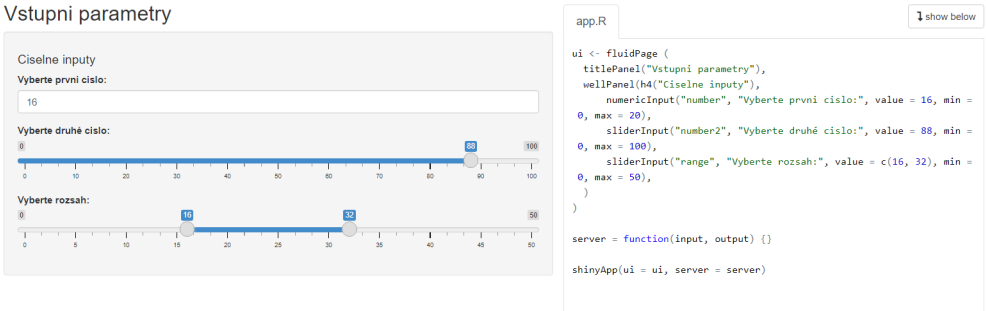
Nyní si představíme některé ze základních vstupních parametrů, se kterými můžeme pracovat.

Číselné vstupy

- `numericInput` - výběr číselné hodnoty. Můžeme volit rozpětí vstupu, ze kterého bude uživatel vybírat a lze také nastavit výchozí hodnotu, která se zobrazí po spuštění aplikace,

- `sliderInput` - posuvník pro výběr číselné hodnoty. Můžeme volit rozpětí, ze kterého uživatel vybírá. Zadáváme výchozí hodnotu, nebo vektor 2 čísel (pokud chceme, aby tato proměnná představovala rozpětí).

Na obrázku 2.8 můžeme vidět možnosti výběru číselných vstupů.



Obrázek 2.8: Možnosti číselných vstupů v aplikaci Shiny.

Výběr 1 možnosti ze seznamu

- `selectInput` - výběr z rozbalovací záložky. Vhodné, pokud máme více variant, či varianty nechceme zobrazovat.
- `radioButtons` - výběr ze seznamu, který je pro uživatele viditelný. Je ideální možností pokud variant nemáme mnoho. Pokud nechceme mít seznam variant zobrazen vertikálně, pak přidáním parametru `inline = TRUE` jej umístíme do řádku.

Na obrázku 2.9 vidíme příklad vstupních parametrů Shiny aplikace, ve kterých vybíráme 1 možnost ze seznamu.

Obrázek 2.9: Výběr 1 možnosti ze seznamu

Výběr více možností ze seznamu

- `selectInput` - výběr variant z rozbalovací záložky. Nutné přidat parametr `multiple = TRUE`.
- `pickerInput` - výběr variant z rozbalovací záložky. Můžeme přidat tlačítko pro výběr všech možností, což ulehčí uživateli práci,
- `checkboxGroupInput` - výběr variant ze seznamu, který je celý zobrazen. Vhodné pokud chceme vidět všechny možnosti výběru. Ukázku zaškrtnutí více variant můžeme vidět na obrázku 2.10.

Obrázek 2.10: Výběr více možností ze seznamu.

Vložení souboru

Velmi důležitým typem vstupu je možnost vložení externího souboru pro následnou analýzu dat uživatelem. Soubory můžeme vkládat pomocí funkce `fileInput`. Na obrázku 2.11 je znázorněna základní struktura pro vkládání .csv souboru. Pokud bychom například chtěli vkládat data ve formátu .txt, pak bychom nastavili parametr `accept = ".txt"`.

Obrázek 2.11: Vložení souboru.

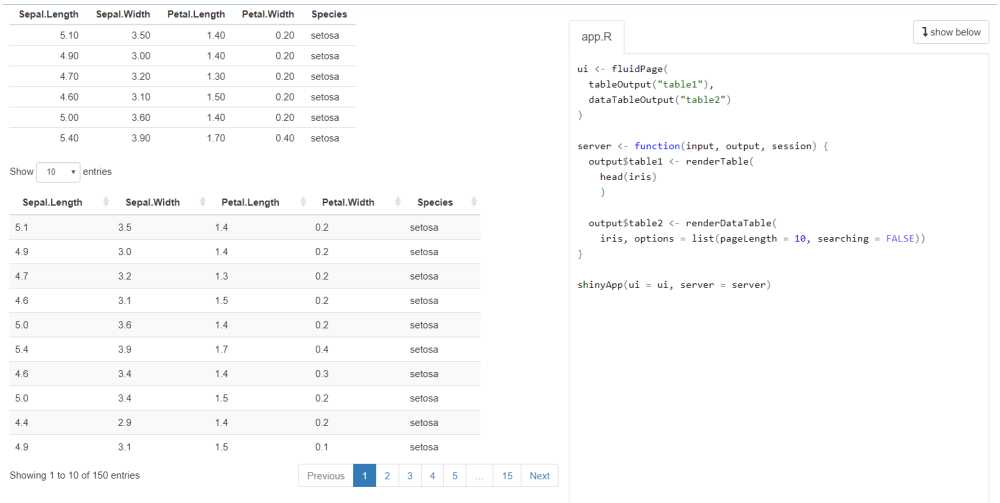
2.2.3. Výstupní parametry

Výstupy v uživatelském rozhraní představují pouze typy objektů, avšak jejich obsah definujeme až na serveru. Výstupní parametry a server jsou vzájemně provázány, proto si ukážeme ihned v této kapitole, jaké funkce na serveru musíme použít pro vytvoření výstupů v uživatelském rozhraní. Stejně jako vstupním parametrům přiřazujeme každému výstupnímu parametru jedinečné ID, =na které se budeme na serveru odkazovat. Například budeme-li chtít v aplikaci zobrazit tabulku, které dáme ID `"table"`. Na serveru pak k ní budeme přistupovat pomocí příkazu `output$table`. Možnosti výstupů v Shiny jsou:

Tabulka

- `tableOutput` - výstupem je tabulka, která je vhodná především pro zobrazení malých matic či menších datových tabulek. Na serveru tuto tabulku vytvoříme pomocí příkazu `renderTable`,
- `dataTableOutput` - výstupem je tabulka, která je ideální variantou pokud chceme zobrazit rozsáhlé datové soubory. Zobrazuje fixní počet řádků, který je volitelný. Na serveru ji tvoříme pomocí funkce `renderDataTable`.

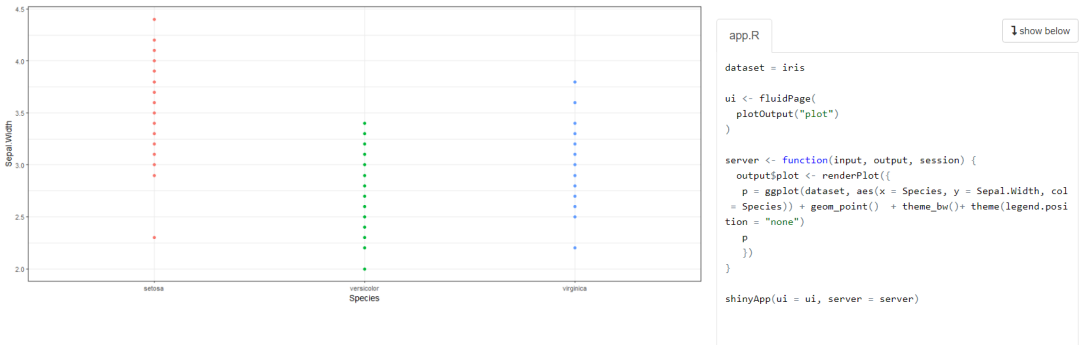
Na obrázku 2.12 vidíme výstup v aplikaci pomocí funkce `tableOutput` a `dataTableOutput` pro data `iris`.



Obrázek 2.12: Výstup formou tabulky.

Graf

Grafické výstupy uvnitř aplikace vytváříme pomocí funkce `plotOutput`. Na serveru daný výstup vytvoříme pomocí funkce `renderPlot`. Ukázku použití funkce `plotOutput`, resp. `renderPlot` vidíme na obrázku 2.13.



Obrázek 2.13: Grafický výstup.

Text

- `textOutput` - výstupem je nestrukturovaný text. Na serveru jej vytvoříme pomocí funkce `renderText`.
- `verbatimTextOutput` - výstupem je text se strukturou, kterou bychom obdrželi jako výstup kódu v R. Je vhodné tento výstup použít pokud chceme v aplikaci zobrazit například výstup funkce `summary`. Na serveru tento výstup vytvoříme pomocí funkce `renderPrint`.

Na obrázku 2.14 můžeme vidět rozdíl mezi výstupy funkce `textOutput` a `verbatimTextOutput`.

Obrázek 2.14: Textový výstup.

Ukládání výstupů

Další z možností výstupu, které Shiny nabízí, je uložení daného výstupu ve zvoleném formátu. Do uživatelského rozhraní pomocí funkce `downloadButton` umístíme tlačítko, které bude ukládat obsah. Na serveru pomocí funkce `downloadHandler` připravíme data na stažení. V prvním argumentu funkce (`file`) nejprve vytvoříme název souboru, pod kterým budeme data ukládat a určíme jeho formát. Druhým argumentem funkce je `content`. Zde určujeme obsah, který se do souboru uloží. Na obrázku 2.15 můžeme vidět jako název pro ukládání označení `"data"`, ke kterému přidáme datum, kdy ke stažení souboru došlo (například `data-2022-04-16.csv`).

Obrázek 2.15: Stahování outputů - downloadHandler

2.3. Server

Server je nejdůležitější částí celé aplikace. Právě tady probíhají všechny příkazy, které tvoří naši aplikaci. Umístění výstupní funkce do uživatelského rozhraní pouze říká, kde se má daný objekt v aplikaci zobrazit, ale rozhodující je funkce na serveru, která vytvoří veškeré výstupy analýzy (například tabulka, text, graf), které uživatel uvidí. Pokud jsme v uživatelském rozhraní označili výstup například `"table"`, musíme jej pod stejným ID volat i na serveru (`output$table`. Server představuje funkci, která je složena ze 3 základních částí:

- `input` - představuje seznam všech hodnot vstupních parametrů v naší aplikaci. Tyto hodnoty jsou v tomto seznamu uloženy pod stejným názvem, jako jsou označeny inputID jednotlivých vstupních parametrů,
- `output` - představuje seznam všech výstupů. Jsou zde uloženy všechny výstupy, které jsou odeslány do uživatelského prostředí, a uživatel je vidí. Jsou uloženy pod stejným ID, které jsme výstupním parametrům dali v uživatelském rozhraní.
- `session` - představuje prostředí, které využíváme pro přístup k informacím a funkcím. Jedná se o volitelný parametr.

Na serveru tvoříme obsah výstupních objektů pomocí funkcí `render`, které jsme si představili v kapitole 2.2.3.

2.4. Reaktivita

Základní vlastností Shiny aplikací je reaktivita. Výstup se stává reaktivním, jestliže jeho obsah závisí na hodnotě vstupního parametru, který uživatel zvolil. Pokud nestanovíme jinak, pak se výstup aktualizuje při každé změně vstupního parametru.

Na obrázku 2.16 vidíme aplikaci, která počítá obsah čtverce a obdélníku. Aplikace obsahuje 2 číselné vstupní parametry - délku strany a (inputID = `"a"`) a délku strany b (inputID = `"b"`). Výstupem je text, který informuje uživatele o obsahu čtverce a obsahu obdélníku. V daném případě se budou výstupní hodnoty měnit po jakékoliv změně vstupních proměnných. Na serverovou část přidáme proměnnou `obsah1`, která říká, že se jedná o reaktivní text, který spojí větu "Obsah čtverce je"a vypočtenou celkovou hodnotu obsahu jako `input$a*input$a`. Následně vytvoříme výstup, který zobrazíme uživateli. Jelikož se jedná o textový výstup, pak použijeme funkci `renderText`, do které umístíme proměnnou `obsah1()`. Závorky u proměnné označují, že se jedná o reaktivní hodnotu.

Obrázek 2.16: Reaktivní prostředí Shiny aplikace.

Pokud bychom nechtěli při každé změně vstupních parametrů znovu vytvářet výstup, pak je možností vložit do uživatelského rozhraní tlačítko, po jehož stisknutí teprve dojde k novému vytvoření výstupu. Tuto možnost použití vidíme na obrázku 2.17, kde jsme do uživatelského prostředí přidali reaktivní tlačítko `zobrazit`. Teprve po stisknutí tohoto tlačítka dojde ke změně výstupu.

Obrázek 2.17: Reaktivní prostředí Shiny aplikace. Zobrazování výstupů až po stisknutí tlačítka.

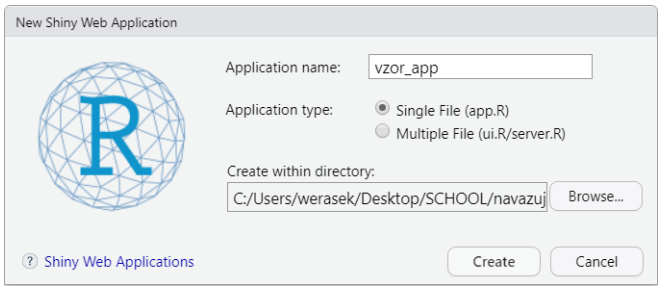
Představili jsme si základy pro vytvoření Shiny aplikace. Nyní si jednoduchou aplikaci vytvoříme.

2.5. Jak vytvořit Shiny aplikaci

V této části si ukážeme, jak můžeme jednoduše vytvořit novou Shiny aplikaci. Před tím než se pustíme do vytváření Shiny aplikace, je nutné si nejprve nainstalovat knihovnu `shiny`.

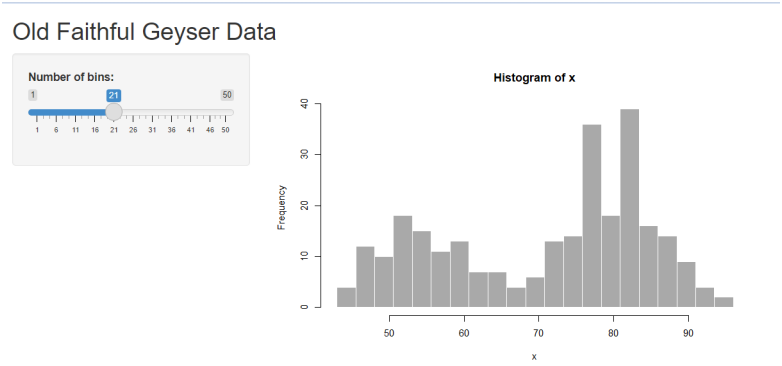
Pro ilustraci vytvoříme Shiny aplikaci, která bude mít rozložení `sidebarLayout`. SidebarPanel bude obsahovat 2 vstupní parametry. Hlavní část bude tvořena 2 záložkami - `Summary` a `Plot`. V záložce `Summary` zobrazíme shrnutí dat a v záložce `Plot` vykreslíme graf, kde na osách grafu budou námi zvolené proměnné. Pro zjednodušení bude naše nová aplikace využívat již implementovaná data v R - `mtcars`.

1. Vytvoříme nový soubor. File → New File → Shiny Web App. Otevře se nám okno (obrázek 2.18), kde si zvolíme název aplikace - v našem případě jsme zvolili `vzor_app`. Určíme, zda chceme mít aplikaci uloženou v 1 souboru `app.R`, či chceme mít každou část `ui` a `server` zvlášť (vhodné pro rozsáhlé aplikace).



Obrázek 2.18: Vytvoření nového skriptu k Shiny aplikaci.

2. Otevře se nám automaticky skript s ilustrativní aplikací, kterou můžeme ihned spustit. Jedná se o aplikaci, která vykresluje histogram rozdělení délek mezi erupcemi (v minutách) pro gejzír Old Faithful v národním parku Yellowstone. V levé části na obrázku 2.19 vidíme číselný parametr ve formě posuvníčku, kterým můžeme měnit, na kolik částí se bude dělit osa x.



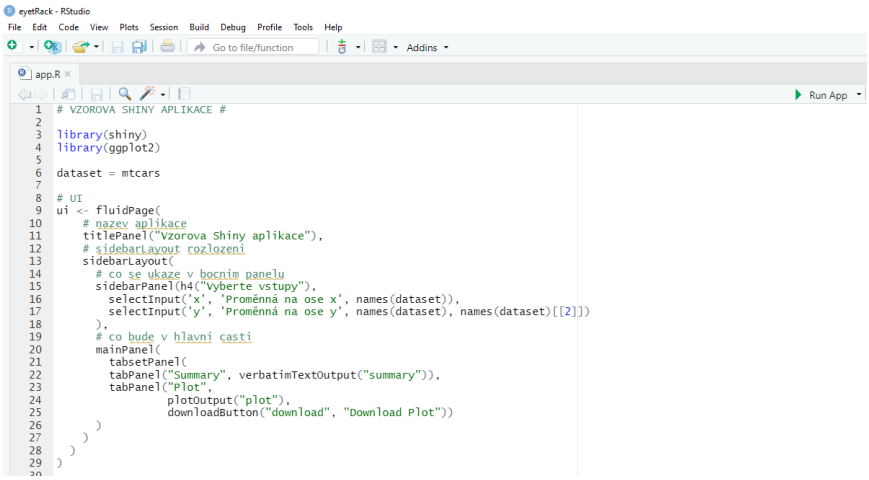
Obrázek 2.19: Ilustrativní automaticky vytvořená apliakce.

3. Vytvoříme vlastní uživatelské prostředí. Můžeme využít strukturu ilustrativní funkce, nebo vytvořit novou. Pomocí funkce `FluidPage` zajistíme, že se velikost naší aplikace přizpůsobí velikosti okna. Zvolíme název aplikace pomocí příkazu `titlePanel("Vzorova Shiny aplikace")`. Využijeme rozložení `sidebarLayout`, kde do bočního panelu umístíme vstupní parametry:

   - selectInput - proměnná na ose x, kterou budeme v části server volat pomocí příkazu `input$x`. Na výběr budeme mít ze všech proměnných, které data `mtcars` obsahují,
   - selectInput - proměnná na ose y. Na serveru ji budeme volat jako `input$y`. Na výběr budeme mít ze stejných proměnných jako u proměnné `x`, při spuštění aplikace bude přednastavenou hodnotou zvolena 2.proměnná z dat `mtcars`.

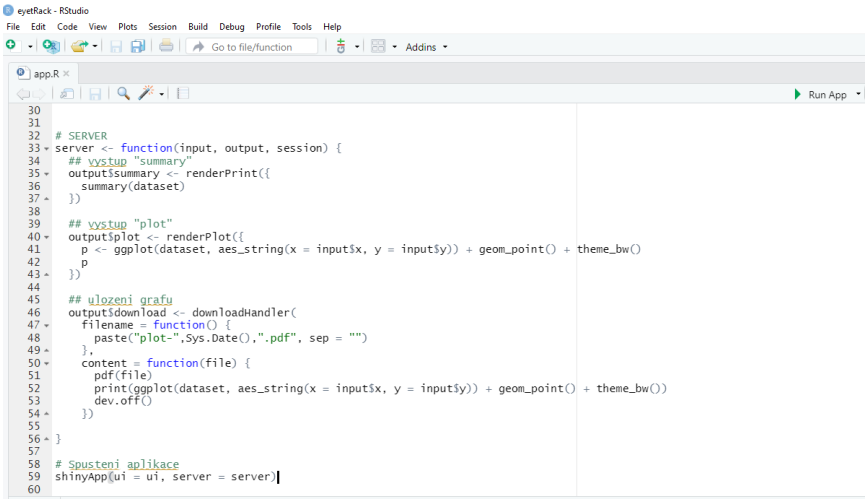
   Do hlavní části umístíme záložky s výstupními parametry:

   - `Summary` - výstupním parametrem je funkce `verbatimTextOutput`, jelikož chceme zobrazit summary dat. Na serveru budeme s touto funkcí pracovat pomocí příkazu `output$summary`.
   - `Plot` - obsahuje 2 výstupy. Nejprve zobrazujeme graf pomocí funkce `plotOutput`, pod který umístíme tlačítko pro jeho stažení pomocí příkazu `downloadButton`.



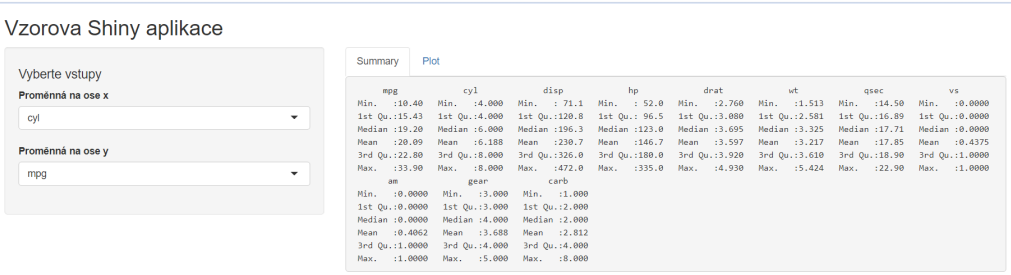
Obrázek 2.20: Vytvoření uživatelského rozhraní nové aplikace.

4. Vytvoříme serverovou část. Tabulku, která bude obsahovat výstup z funkce `summary` (`output$summary`), vytvoříme pomocí funkce `renderPrint`. Dále chceme vytvořit graf, u kterého se mění proměnné na osách dle naší volby. Graf vytvoříme pomocí funkce `renderPlot`. Uvnitř funkce definujeme prměnné grafu `x = input$x`, resp. `y = input$y`. To zajistí, že při změně vstupu dojde k vytvoření nového grafu. Posledním výstupem, který nám zbývá, je tlačítko na uložení. K tomu využijeme funkci `downloadHandler`. Už nám chybí pouze zadat název souboru, pod kterým se náš graf bude ukládat a jeho formát. U naší vzorové aplikace, jejíž část `server` můžeme vidět na obrázku 2.21, jsme zvolili, že chceme ukládat ve formátu .pdf.



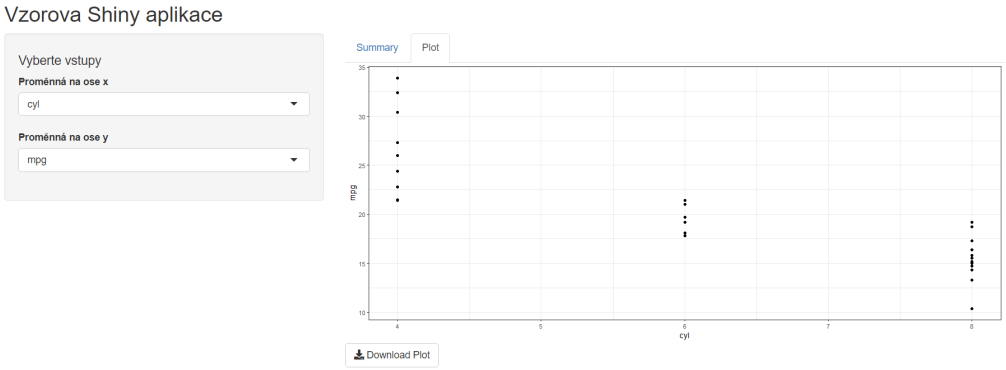
Obrázek 2.21: Vytvoření funkcí na serveru.

5. Spustíme aplikaci pomocí příkazu `shinyApp(ui, server)`, nebo pomocí tlačítka `Run app` vpravo na panelu nástrojů dokumentu. Aplikace se otevře v novém okně a můžeme ji začít používat. Na obrázku 2.22 vidíme záložku `Summary` naší nově vytvořené aplikace.



Obrázek 2.22: Vytvořená Shiny aplikace - záložka Summary.

V záložce `Plot` na obrázku 2.23 vidíme graf, ve kterém můžeme měnit proměnné, které se budou vykreslovat na osách x a y.



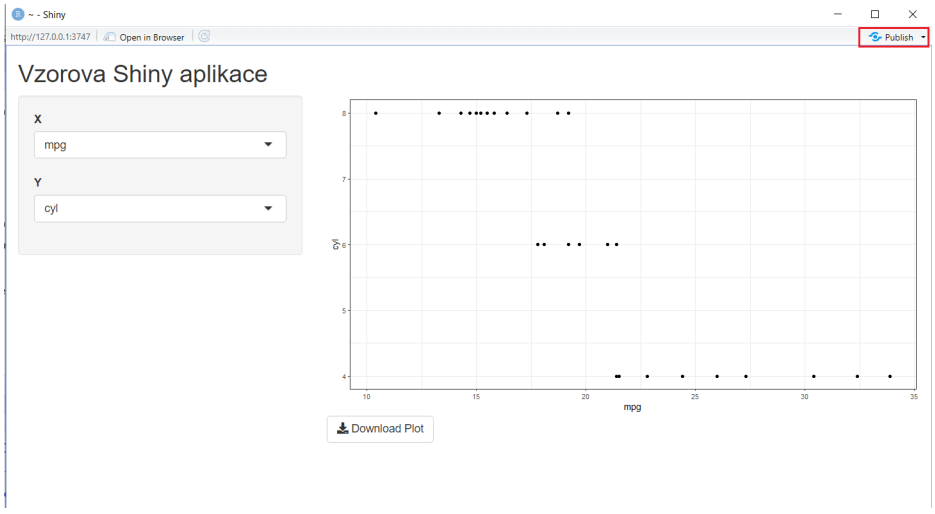
Obrázek 2.23: Vytvořená Shiny aplikace - záložka Plot.

2.6. Jak publikovat Shiny aplikaci?

V této části budeme vycházet z návodu [16]. Nejjednodušší způsob, jak změnit naši Shiny aplikaci na webovou stránku, kterou můžeme sdílet s ostatními, je použít `shinyapps.io` (hostingovou službu RStudio pro aplikace Shiny). Tato hostingová služba nám umožňuje nahrát naši aplikaci přímo z R Studia. Máme na výběr z několika verzí. Bezplatná verze nám umožňuje publikování 5 aplikací s celkovým počtem 25 aktivních hodin za měsíc. Pokud bychom chtěli zvýšit počet aktivních hodin, či počet publikovaných aplikací, je nutné si zaplatit rozšiřovací verzi.

Publikování aplikace si ukážeme na vzorové aplikaci, kterou jsme vytvořili v předešlé kapitole.

1. Vytvoříme si účet na serveru shinyapps.io.
2. V RStudiu otevřeme skript s aplikací, kterou se chystáme publikovat a spustíme ji.
3. Po otevření okna s aplikací v pravém horním rohu stiskneme tlačítko `Publish` (obrázek 2.24).



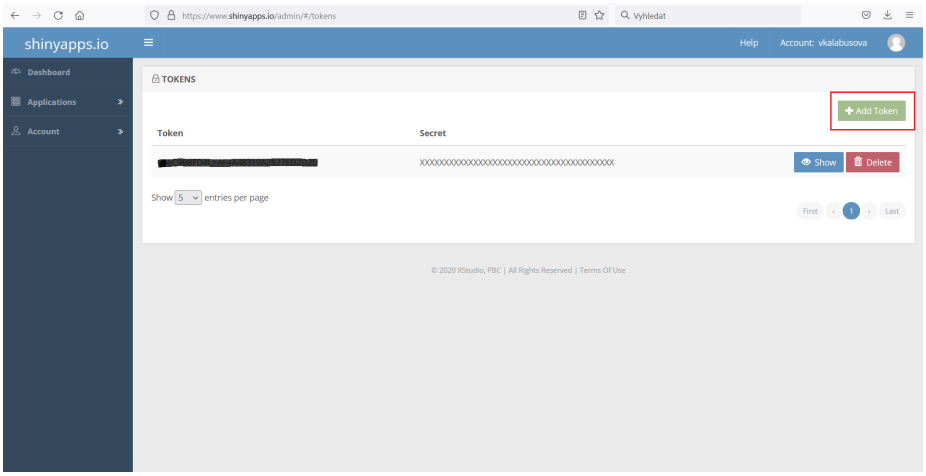
Obrázek 2.24: Spustíme aplikaci a stiskneme Publish.

4. Pokud je to naše první aplikace Shiny, kterou chceme publikovat, pak by pole `Publish From Account` mělo být prázdné. Kliknutím na `Add New Account` propojíme RStudio s účtem na `shinyapps.io`, který jsme si vytvořili.

5. Klikneme na link odkazující na server `shinyapps.io` a přihlásíme se.

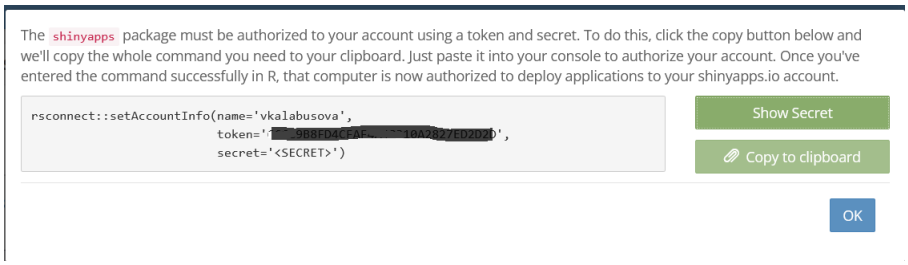
6. Klikneme na naše jméno a následně na `Tokens`. Jelikož se jedná o první publikovanou aplikaci, pak by zde žádný "token" neměl být.

7. Stiskneme tlačítko `Add Token` (obrázek 2.25), následně na `Show`



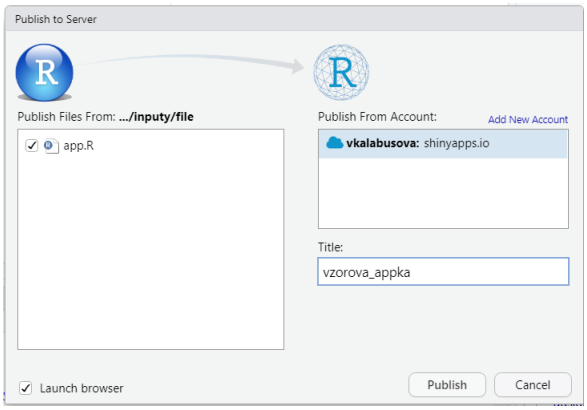
Obrázek 2.25: Přihlásíme se na server a následně přidáme nový "token".

8. Stiskneme `Show secret`, zkopírujeme kód z okna, které můžeme vidět na obrázku 2.26, vložíme jej do konzole RStudia a spustíme. Nyní je náš počítač oprávněn k umístění aplikací na server `shinyapp.io`.



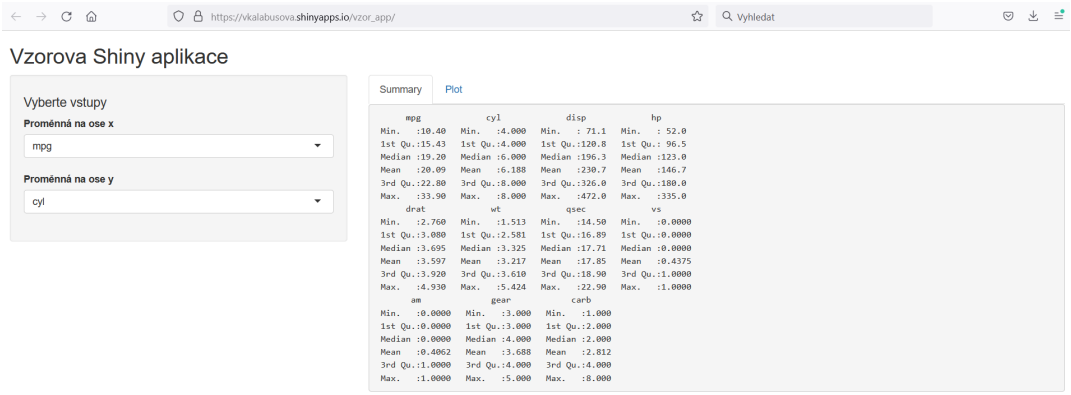
Obrázek 2.26: Kopírujeme kód atd

9. Vrátíme se zpět do okna, kde můžeme aplikaci publikovat, vybereme název (bez mezery) a klikneme na tlačítko `Publish`.



Obrázek 2.27: Publikování aplikace

10. Po několika vteřinách (v závislosti na velikosti aplikace) se spustí internetový prohlížeč s naší Shiny aplikací.



Obrázek 2.28: Nová aplikace je připravená k využívání.

11. Nyní může aplikaci používat každý, kdo má daný odkaz. Pokud bychom chtěli cokoliv v aplikaci upravit, pak po úpravách a spuštění aplikace stiskneme tlačítko `Republish` a publikování zopakujeme.

Kapitola 3

Rekurentní kvantifikační analýza

Rekurentní kvantifikační analýza (Recurrence Quantification Analysis - RQA) je poměrně mladá metoda analýzy dat. Základní myšlenkou této metody je najít, popsat a kvantifikovat určité opakující se vzorce v chování časové řady. Například se může jednat o sledování stejného místa obrázku při zkoumání pohybů očí respondenta, analýzu stejné hodnoty kurzů na trhu a mnohé další. [5]

S možností použít rekurenci pro charakterizování dynamických systémů přišel Henri Poincaré již na konci 19. století, avšak jeho objev musel na praktické vyžití, z důvodu méně výkonných počítačů, počkat více než 70 let [12].

Důležitým milníkem byl rok 1937, kdy představil J. P. Eckmann s kolegy graf rekurence jako nástroj pro vizualizaci rekurentních bodů.

Interpretace rekurentního grafu je však často příliš subjektivní [6]. Z tohoto důvodu představili na konci 90. let 20. století Joseph P. Zbilut a Charles L. Webber rekurentní kvantifikační analýzu, která umožňuje kvantifikovat a jednoduše popsat informace, které rekurentní graf obsahuje.

Hlavní výhodou rekurentní kvantifikační analýzy je její jednoduchost a využitelnost v mnoha různých disciplínách, kterými jsou například fyzika, ekonomie a inženýrství. V publikaci [1] Nicola C. Anderson popisuje a ukazuje, jak můžeme použít rekurentní kvantifikační analýzu a její míry pro charakteristiku chování pohybu očí. RQA v tomto případě určuje, jak často se respondenti vrací do již dříve prohlédnutých oblastí obrázku. Především z Andersonovy práce v této kapitole vycházíme, dále z publikací [11], [21], [22] a [23].

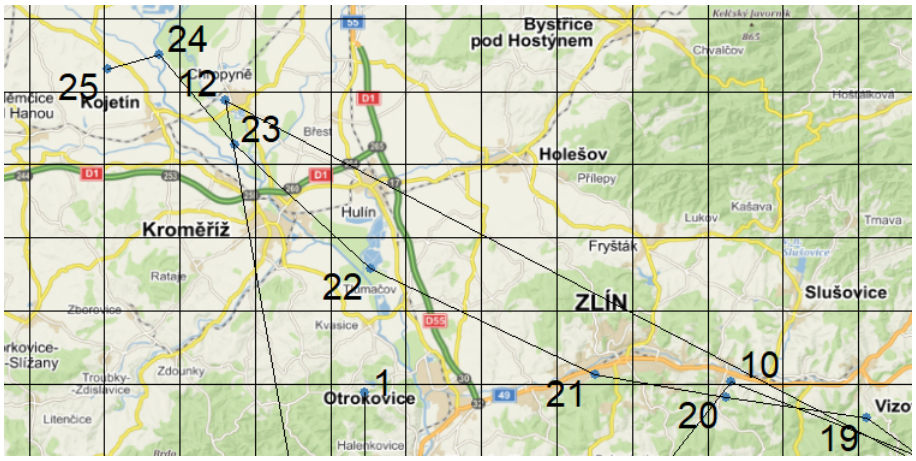
3.1. Rekurence

Uvažujme posloupnost fixací f_i, $i = 1, \ldots, N$, kde každá z fixací má danou x-ovou a y-ovou souřadnici, tj. $f_i = \langle x_i, y_i \rangle$ a dobu trvání t_i, $i = 1, \ldots, N$. Řekneme, že fixace f_i a f_j se opakují, resp. jsou rekurentní, jestliže jsou obě fixace blízko sebe. V oblasti eye-trackingu můžeme přistupovat k rekurenci pomocí různých metod, které si následně představíme.

3.1.1. Metoda fixní mřížky

První z možností, jak určit, zda jsou fixace rekurentní, je pomocí metody fixní mřížky. Jak napovídá název metody, obrázek, na který se respondent díval, rozdělíme pomocí mřížky na stejně velké čtverce. Následně považujeme dvě fixace za rekurentní, pokud obě fixace leží ve stejném čtverci. Pokud se podíváme na obrázek 3.1, pak bychom uvažovali fixace 24 a 25 jako rekurentní.

Nevýhodou u této metody je, že rozdělení pomocí mřížky provádíme bez ohledu na obsah obrázku. V některých místech se tak může jevit rozdělení příliš hrubé, či naopak příliš jemné. Další z nevýhod je také to, že dvě fixace nemusí být považovány za rekurentní, i když leží velmi blízko sebe - například fixace 10 a 20 na obrázku 3.1.

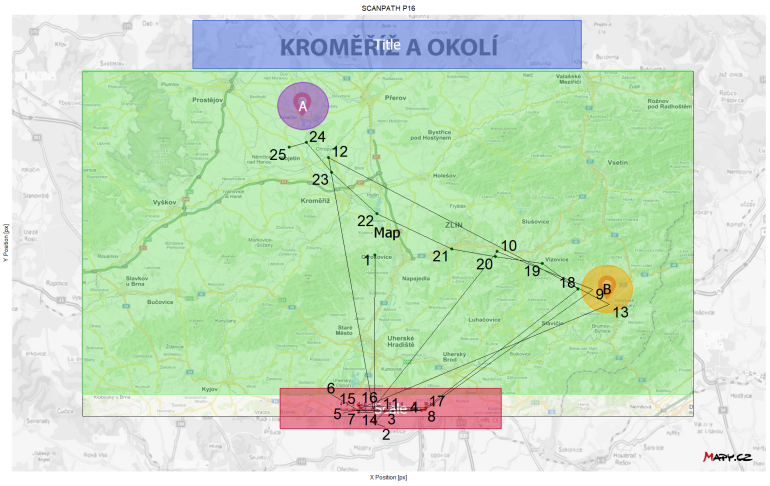


Obrázek 3.1: Rozdělení obrázku pomocí metody fixní mřížky. Rozměry čtverců jsou 64x64 pixelů.

3.1.2. Metoda oblastí zájmu - AOI

Problém s rozdělením obrázku bez ohledu na obsah řeší metoda oblastí zájmu. V této metodě na začátku definujeme oblasti zájmu - významné části obrázku (areas of interest). Poté postupujeme obdobně jako v předchozím případě, tj. řekne-
me, že dvě fixace jsou rekurentní, jestliže obě fixace leží ve stejné oblasti zájmu. Možné rozdělení obrázku na oblasti zájmu můžeme vidět na obrázku 3.2, kdy jsme rozdělili mapu na titulek (modrá barva), bod A (fialová), bod B (žlutá), mapu (zelená), měřítko (červená) a prázdné místo (bílá).

Také v tomto případě nastává problém, že dvě fixace nejsou považovány za rekurentní i když leží blízko sebe, jak ukazují fixace 9 a 18 na obrázku 3.2. Na druhou stranu i fixace, které jsou od sebe více vzdáleny, mohou být rekurentní, jak vidíme u fixací 18 a 25 téhož obrázku.



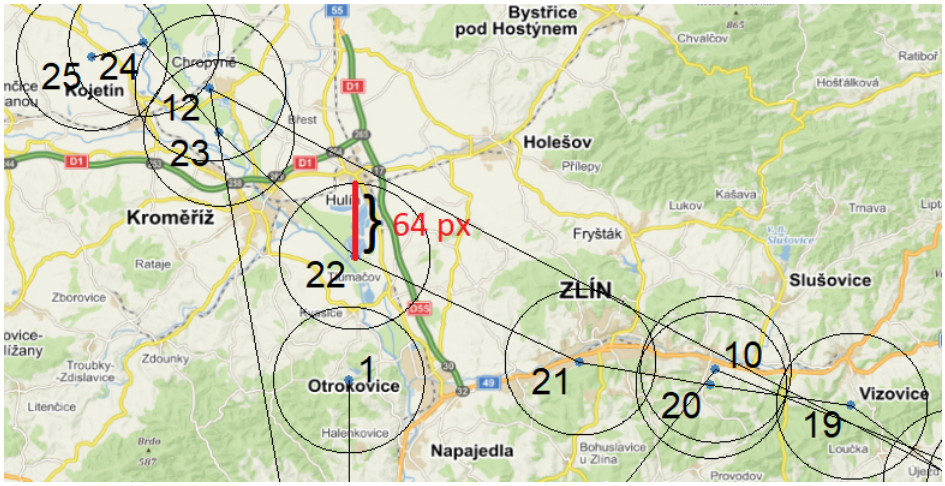
Obrázek 3.2: Rozdělení obrázku pomocí metody oblastí zájmu.

3.1.3. Metoda fixní vzdálenosti

Nedostatky obou předchozích metod, kdy nemusí být odhaleny rekurentní fixace, i když leží velmi blízko sebe, eliminuje metoda fixní vzdálenosti. Řekneme, že dvě fixace f_i a f_j jsou rekurentní, jestliže $d(f_i, f_j) \leq \rho$, kde d je zvolená metrika vzdálenosti (např. listonošská, eukleidovská, ...) a ρ je zvolený poloměr. Pokud zvolíme eukleidovskou metriku, tak je vzdálenost d mezi dvěma fixacemi dána vztahem

$$d(f_i, f_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \text{kde } i, j = 1, ..., N.$$

V tomto případě na obrázku 3.3 odhalíme, že fixace 10 a 20 jsou rekurentní, jelikož vzdálenost mezi nimi je menší než zvolený poloměr 64 pixelů, u metody fixní mřížky tomu tak nebylo.



Obrázek 3.3: Rozdělení obrázku pomocí metody fixní vzdálenosti.

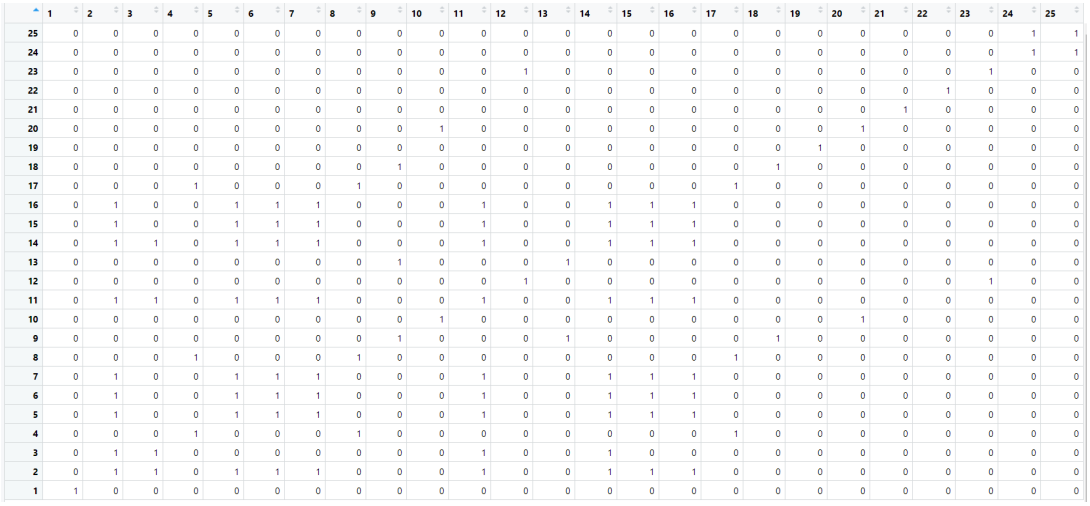
3.2. Matice rekurence

Matice rekurence je čtvercová matice typu NxN, která zaznamenává rekurentní fixace. V případě, kdy neuvažujeme dobu trvání, obsahuje pouze hodnoty 0 nebo 1. Pokud jsou fixace f_i a f_j rekurentní, pak je prvek matice r_{ij} roven 1, jestliže fixace f_i a f_j naopak nejsou rekurentní, pak je prvek matice r_{ij} roven 0. Prvky rekurentní matice r_{ij} pro metodu fixní vzdálenosti definujeme pomocí vztahu

$$r_{ij} = \begin{cases} 1, & d\left(f_i, f_j\right) \leq \rho \qquad i,j = 1,\ldots,N \\ 0, & jinak \end{cases} \tag{3.1}$$

kde d je zvolená metrika vzdálenosti (např. listonošská, eukleidovská, ...) a ρ je zvolený poloměr.

Ukázku matice rekurence pro metodu fixní vzdálenosti se zvoleným poloměrem 64 pixelů a eukleidovskou metrikou, kdy neuvažujeme dobu trvání fixací můžeme vidět na obrázku 3.4.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
17	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
15	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
14	0	1	1	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	1	1	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
6	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
5	0	1	0	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Obrázek 3.4: Matice rekurence pro metodu fixní vzdálenosti s poloměrem 64 pixelů bez uvažování doby trvání.

Pokud budeme uvažovat metodu zohledňující dobu trvání fixací, pak jsou-li fixace f_i a f_j rekurentní, je prvek rekurentní matice na pozici r_{ij} dán hodnotou, která odpovídá součtu dob trvání fixací f_i a f_j.

Prvky rekurentní matice pro metodu fixní vzdálenosti definujeme pomocí vztahu

$$r_{ij}^{t} = \begin{cases} t_i + t_j, & d\left(f_i, f_j\right) \leq \rho \qquad i, j = 1, \ldots, N \\ 0, & jinak \end{cases} \tag{3.2}$$

kde d je zvolená metrika vzdálenosti (např. listonošská, eukleidovská, ...), ρ je zvolený poloměr a t_i, t_j označují dobu trvání i-té a j-té fixace.

Příklad matice rekurence pro metodu fixní vzdálenosti s poloměrem 64 pixelů, s využitím eukliedovské metriky a s uvažováním doby trvání fixací můžeme vidět na obrázku 3.5.

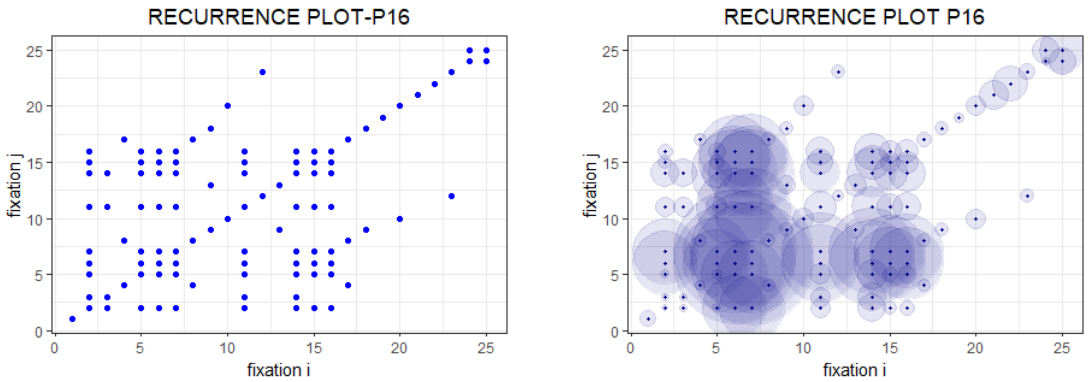
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1120	1952
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	288	1120
23	0	0	0	0	0	0	0	0	0	0	0	568	0	0	0	0	0	0	0	0	0	0	688	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1488	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1200	0	0	0	0
20	0	0	0	0	0	0	0	0	0	836	0	0	0	0	0	0	0	0	0	816	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	400	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	564	0	0	0	0	0	0	0	0	552	0	0	0	0	0	0	0
17	0	0	0	508	0	0	0	580	0	0	0	0	0	0	0	0	592	0	0	0	0	0	0	0	0
16	0	632	0	0	700	3072	3096	0	0	0	1168	0	0	1532	912	992	0	0	0	0	0	0	0	0	0
15	0	552	0	0	620	2992	3016	0	0	0	1088	0	0	1452	832	912	0	0	0	0	0	0	0	0	0
14	0	1172	1184	0	1240	3612	3636	0	0	0	1708	0	0	2072	1452	1532	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	752	0	0	0	928	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	448	0	0	0	0	0	0	0	0	0	0	568	0	0
11	0	808	820	0	876	3248	3272	0	0	0	1344	0	0	1708	1088	1168	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	856	0	0	0	0	0	0	0	0	0	836	0	0	0	0	0
9	0	0	0	0	0	0	0	0	576	0	0	0	752	0	0	0	0	564	0	0	0	0	0	0	0
8	0	0	0	496	0	0	0	568	0	0	0	0	0	0	0	0	580	0	0	0	0	0	0	0	0
7	0	2736	0	0	2804	5176	5200	0	0	0	3272	0	0	3636	3016	3096	0	0	0	0	0	0	0	0	0
6	0	2712	0	0	2780	5152	5176	0	0	0	3248	0	0	3612	2992	3072	0	0	0	0	0	0	0	0	0
5	0	340	0	0	408	2780	2804	0	0	0	876	0	0	1240	620	700	0	0	0	0	0	0	0	0	0
4	0	0	0	424	0	0	0	496	0	0	0	0	0	0	0	0	508	0	0	0	0	0	0	0	0
3	0	284	296	0	0	0	0	0	0	0	820	0	0	1184	0	0	0	0	0	0	0	0	0	0	0
2	0	272	284	0	340	2712	2736	0	0	0	808	0	0	1172	552	632	0	0	0	0	0	0	0	0	0
1	640	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Obrázek 3.5: Matice rekurence pro metodu fixní vzdálenosti s poloměrem 64 pixelů a uvažováním doby trvání.

3.3. Graf rekurence

Rekurentní fixace můžeme vizualizovat pomocí grafu rekurence. Jedná se o grafické znázornění matice rekurence. Pokud jsou fixace f_i a f_j rekurentní, pak je na pozici i, j v grafu vykreslena tečka. V případě, že se fixace f_i a f_j neopakují, pak na pozici i, j není vykresleno nic, což ukazuje obrázek 3.6 vlevo.

Na obrázku 3.6 vpravo vidíme situaci, kdy uvažujeme graf rekurence s dobou trvání fixací. Velikost tečky v grafu je nyní úměrná součtu dob trvání fixací f_i a f_j.



Obrázek 3.6: Graf rekurence bez uvažování doby trvání fixací (vlevo) a při uvažování doby trvání (vpravo)

Rekurentní graf má vždy zvýrazněnou hlavní diagonálu (incidenční linii), protože porovnáváme fixace se sebou samými ($i = j$). Z důvodu symetrie metriky vzdálenosti, jsou symetrické i rekurentní grafy dle hlavní diagonály. Pro každého respondenta je generován samostatný rekurentní graf. Fixace jsou vykreslovány postupně, čím větší je vzdálenost mezi rekurentní fixací a incidenční linii, tím delší je doba (určená počtem fixací), mezi prvotní a fixací opětovnou. Například pokud uvažujeme, že respondent provedl 25 fixací a fixace 3 a 25 jsou rekurentní, pak to značí, že respondent si oblast prohlédl téměř ihned na začátku (3. fixace) a do stejného místa se vrátil 25. fixací, tj. na úplném konci analýzy.

Nevýhodou grafu rekurence je, že interpretace může být hodně subjektivní. Každý z nás v něm může najít odlišné vzorce, proto často přecházíme ke kvantifikování informací z rekurentního grafu pomocí měr rekurence.

3.4. Míry rekurence

Graf rekurence nám poskytuje pouze grafické znázornění rekurentních fixací. Abychom mohli mezi sebou porovnávat různé respondenty, či různá zadání, je nutné graf doplnit o kvantitativní míry rekurence. Tyto míry nám následně umožňují charakterizovat chování systému.

Z důvodu symetrie rekurentního grafu, zjišťujeme kvantitativní míry pouze z hodnot v horním trojúhelníku nad diagonálou, kterou taktéž vynecháme, neboť nám nedává žádnou novou informaci, pouze říká, že každá z fixací je sama se sebou rekurentní.

V oblasti eye-trackingu využíváme čtyři základní míry, které si za okamžik představíme, ovšem existuje jich mnohem více. Kompletní seznam měr, které se používají, můžeme najít v publikaci [14]. Míry opakování a hodnota CORM zachycují celkovou strukturu posloupnosti fixací. Měří, kolikrát byl pohled opakovaně zaměřen na dané části obrázku (opakování) a jestli tyto opětovné fixace nastávají dříve nebo později v průběhu testu (CORM). Determinismus a laminarita jsou míry, které zkoumají strukturu podrobněji. Vyjadřují posloupnosti fixací, které se opakují (determinismus) a body, ve kterých se vyskytuje podrobnější (detailní) prohlížení oblasti obrázku (laminarita).

V případě, kdy budeme uvažovat i dobu trvání času, dávají nám míry rekurentní kvantifikační analýzy ještě podrobnější informace o průběhu testu.

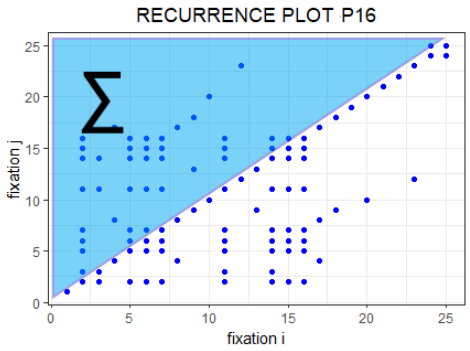
3.4.1. Míra rekurence

První charakteristikou, kterou si představíme, je míra rekurence. Dává nám informaci o tom, jak často se respondent vracel do již dříve prohlédnuté oblasti.

Míru rekurence definujeme jako

$$REC = 100\frac{2R}{N\,(N-1)}, \tag{3.3}$$

kde R představuje součet opakujících se bodů v horním trojúhelníku rekurentního grafu, resp. součet hodnot v horním trojúhelníku rekurentní matice, $R = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} r_{ij}$ (obrázek 3.7) a N značí počet fixací, které v testu provedeme.



Obrázek 3.7: R představuje součet bodů v horním trojúhelníku rekurentního grafu.

V případě uvažování doby trvání míra rekurence značí, jaké procento času respondent věnoval rekurentním fixacím a původní vzorec můžeme předefinovat jako

$$REC^t = 100\frac{R^t}{(N-1)\,T}, \tag{3.4}$$

kde R^t představuje součet dob trvání opakujících se fixací v horním trojúhelníku rekurentního grafu, resp. součet hodnot v horním trojúhelníku rekurentní matice, $R^t = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} r_{ij}^t$ a T označuje celkovou dobu všech fixací, $T = \sum_{i=1}^{N} t_i$.

3.4.2. Míra determinismu

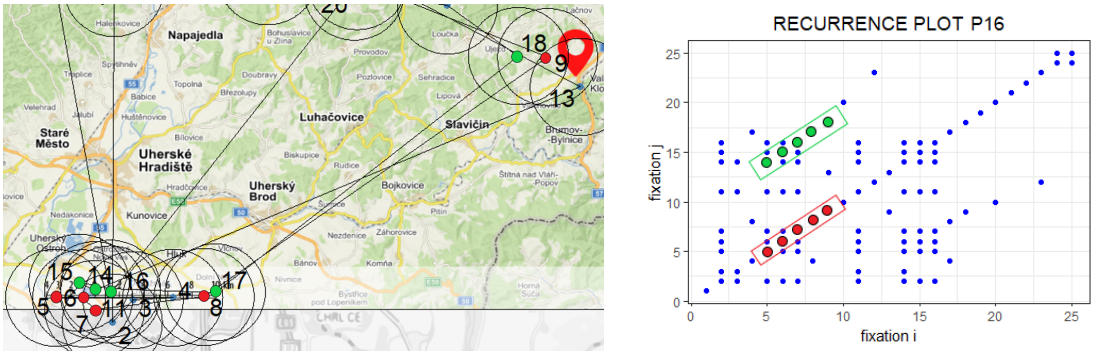
Determinismus nám poskytuje údaje o předvídatelnosti systému. Míra determinismu vyjadřuje, jak často lidé při sledování obrázku opakují určité vzory, představuje podíl opakujících se bodů, které tvoří diagonální linie.

Míru determinismu definujeme jako

$$DET = 100\frac{|\ D_L\ |}{R} \tag{3.5}$$

kde $|D_L|$ označuje počet bodů v horním trojúhelníku v rekurentním grafu, resp. rekurentní matici, které tvoří diagonální linii s minimální délkou L. Minimální délka diagonální linie je obecně stanovena jako $L = 2$, lze ji ale dle potřeby měnit. R představuje součet opakujících se bodů v horním trojúhelníku v rekurentním grafu (viz. míra rekurence).

Délka diagonální linie vyjadřuje počet fixací, které tvoří opakující se vzory pohybu očí. Na obrázku 3.8 můžeme vidět, že fixace 14, 15, 16, 17 a 18 (zelená barva) vytvořily v rekurentním grafu diagonální linii délky 5, jelikož opakovaly stejný vzorec pohybu očí, jako fixace 5, 6, 7, 8 a 9 (červená barva).



Obrázek 3.8: Ukázka pohybu očí, které v grafu rekurence (vpravo) vytvoří diagonální linii délky 5.

Pokud budeme uvažovat dobu trvání fixací, pak míra determinismu označuje procento času rekurentních fixací, které opakovaly nějaký vzor. Míru determinismu můžeme předefinovat jako

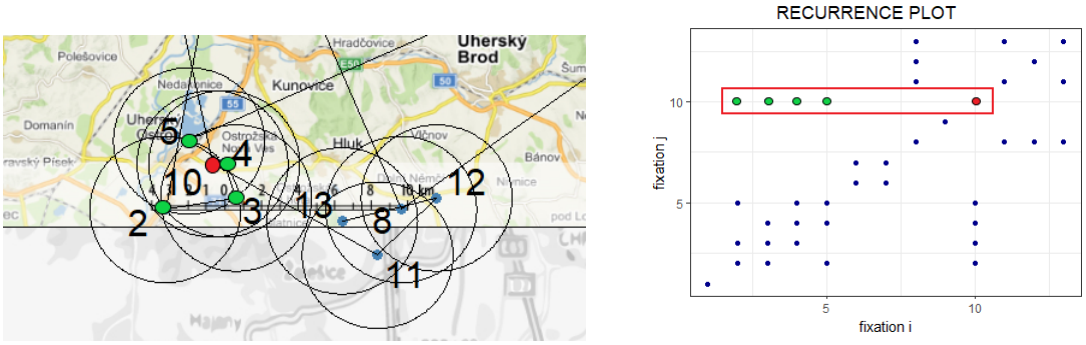
$$DET^t = \frac{100}{R^t}\sum_{(i,j)\in D_L} r_{ij}^t \tag{3.6}$$

kde R^t je součet dob trvání opakujících se fixací v horním trojúhelníku, a suma označuje součet prvků matice rekurence, které tvoří diagonální linie.

3.4.3. Míra laminarity

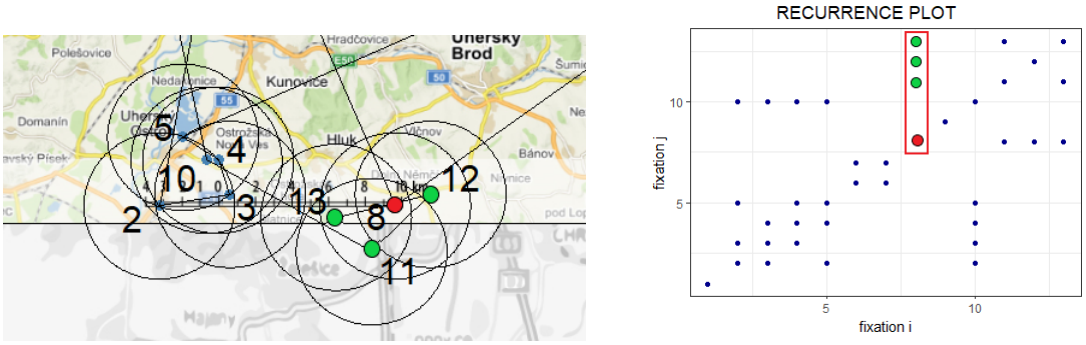
Laminarita obecně ukazuje detailnější prohlížení určité oblasti obrázku. Míra laminarity je ovlivňována horizontálními a vertikálními liniemi.

Vodorovné linie představují oblasti, které jsme nejdříve pozorovali do detailu a později jsme se k nim letmo vrátili - na obrázku 3.9 jsme nejdříve detailněji pozorovali místo při fixacích 2, 3, 4, a 5 (zelená barva) a poté jsme se do stejné oblasti vrátili fixací 10 (červená barva), to způsobilo vytvoření horizontální linie délky 4 v rekurentním grafu.



Obrázek 3.9: Ukázka pohybu očí, které v grafu rekurence (vpravo) vytvoří horizontální linii délky 4.

Naopak svislé linie představují oblasti, které jsme si nejdříve letmo prohlédli, později se k nim vrátili a zkoumali je více podrobně. To můžeme vidět na obrázku 3.10, kdy jsme si místo rychle prohlédli - fixace 8 (červená barva), později jsme se do oblasti vrátili - fixace 11, 12, 13 a pozorovali místo detailněji, což způsobilo vytvoření vertikální linie s délkou 3 v rekurentním grafu.



Obrázek 3.10: Ukázka pohybu očí, které v grafu rekurence (vpravo) vytvoří vertikální linii délky 3.

Větší shluky opakujících se bodů v grafu rekurence (ať už jsou tvořeny horizontálami, či vertikálami) značí detailnější prohlížení oblasti.

Míru laminarity definujeme jako

$$LAM = 100\frac{\mid H_L \mid + \mid V_L \mid}{2R} \tag{3.7}$$

kde $|H_L|$ označuje počet bodů v horním trojúhelníku v rekurentním grafu, resp. rekurentní matici, které tvoří horizontální linii s minimální délkou L. $|V_L|$ označuje počet bodů v horním trojúhelníku rekurentního grafu, resp. rekurentní matici, které tvoří vertikální linii s minimální délkou L. Minimální délka horizontálních a vertikálních linii je opět obecně stanovena jako $L = 2$, lze ji ale dle potřeby měnit. R představuje součet opakujících se bodů v horním trojúhelníku v rekurentním grafu (viz. míra rekurence).

Pokud budeme uvažovat dobu trvání fixací, pak míra laminarity vyjadřuje procento celkového času rekurentních fixací, který byl věnován opětovnému detailnímu prohlížení vybraných oblastí či jejich důkladnému prohlížení s pozdějším krátkým návratem. Míru laminarity můžeme předefinovat jako

$$LAM^t = \frac{100}{2R^t}\left(\sum_{(i,j)\in H_L} r_{ij}^t + \sum_{(i,j)\in V_L} r_{ij}^t\right) \tag{3.8}$$

kde $R^t = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} r_{ij}^t$ představuje součet prvků r_{ij}^t v horním trojúhelníku matice rekurence a uvnitř sum sčítáme prvky z horního trojúhelníku matice rekurence, které tvoří horizontální, resp. vertikální linie alespoň délky L.

3.4.4. Center of recurrent mass - CORM

Hodnota míry CORM vyjadřuje, kde přibližně (v čase) se vyskytuje většina rekurentních bodů.

CORM definujeme jako vzdálenost středu většiny opakujících se bodů od incidenční linie (hlavní diagonály).

$$CORM = 100\frac{\sum_{i=1}^{N-1}\sum_{j=i}^{N}(j-i)\,r_{ij}}{(N-1)\,R} \tag{3.9}$$

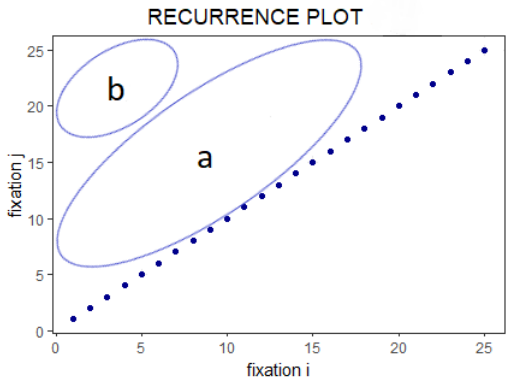
kde r_{ij} představuje prvek rekurentní matice na pozici i, j a R je součet opakujících se bodů v horním trojúhelníku v rekurentním grafu, resp. v rekurentní matici (viz míra rekurence).

V případě, že budeme uvažovat dobu trvání fixací, pak CORM označuje, u kterých rekurentních fixací trávíme nejvíce času. CORM můžeme následně předefinovat jako

$$CORM^t = 100 \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (j-i)\, r_{ij}^t}{(N-1)^2\, T} \tag{3.10}$$

kde r_{ij}^t představuje prvek rekurentní matice na pozici i, j, a T je součet doby trvání všech fixací, tj. $T = \sum_{i=1}^{N} t_i$.

Jedná se o znormovanou veličinu, jejíž maximální hodnota je rovna 100. Malé hodnoty CORM značí, že opětovné fixace mají tendenci nastávat brzo po prvním prohlédnutí oblasti, zatímco velké hodnoty značí, že opětovné fixace nastávají později po prvním prohlédnutí. Na obrázku 3.11 je vidět, že hodnota CORM je malá, jestliže jsou rekurentní body blízko incidenční linie (a). Pokud k opakování dochází později v čase, rekurentní body jsou od incidenční linie dál (b) a hodnota CORM bude tudíž vysoká.



Obrázek 3.11: Rozložení rekurentních bodů v grafu rekurence.

Kapitola 4

Okolní a ohnisková pozornost

Vzhledem k vzájemnému spojení pozornosti a sakád, se má za to, že trvání fixací a amplituda sakády odrážejí výběr pozornosti. K charakterizování dvou druhů pozornosti byl v roce 2016 představen koeficient K jako nástroj pro rozlišení okolní a ohniskové pozornosti při provádění kartografických úkolů. [7]. Okolní pozornost je typicky charakterizována relativně krátkými fixacemi následovanými dlouhými sakádami. Naopak ohnisková pozornost je popsána dlouhými fixacemi následovanými krátkými sakádami.

Koeficient K definujeme jako

$$K_i = \frac{d_i - \mu_d}{\sigma_d} - \frac{a_{i+1} - \mu_a}{\sigma_a} \qquad K = \frac{1}{N-1} \sum_{i=1}^{N-1} K_i \tag{4.1}$$

kde
d_i ... doba trvání i-té fixace
μ_d ... průměr doby trvání fixací
σ_d ... směrodatná odchylka doby trvání fixací
a_{i+1} ... amplituda sakády fixace f_i a f_{i+1} (ve stupních)
μ_a ... průměr amplitud sakád
σ_d ... směrodatná odchylka amplitud sakád
N ... počet fixací

Například $K_i = 1$ značí, že trvání aktuální fixace je o více než jednu směrodatnou odchylku delší než následující amplituda sakády, zatímco $K_i = -1$ značí, že amplituda sakády je o jednu standardní odchylku delší než trvání předchozí fixace. Pokud $K_i = 0$, pak délka fixace a následná amplituda sakády jsou statisticky ekvivalentní. [2]

Koeficient K zachycuje časový vztah mezi standardizovanou délkou fixace (z-skóre) a následnou amplitudou sakády. $K > 0$ označuje ohniskové sledování, zatímco $K < 0$ naznačuje okolní sledování. Při kolísání mezi ohniskovou a okolní pozorností by koeficient K mohl indikovat změny v kognitivní zátěži odpovídající stimulu nebo složitosti úkolu. Pokud se pozornost postupem času stává více ohniskovou, pak koeficient K může indikovat závěr vizuálního pátrání, nebo například nudu, či vyvrcholení rozhodnutí daného zadání.

Amplitudu sakády a_{i+1} pro fixace f_i a f_{i+1} určíme následovně:

- spočítáme vzdálenost mezi fixacemi f_i a f_{i+1}

$$d\left(f_i, f_{i+1}\right) = \sqrt{\left(x_i - x_{i+1}\right)^2 + \left(y_i - y_{i+1}\right)^2}, \quad \text{kde } i = 1, ..., N-1.$$

- získanou vzdálenost v pixelech převedeme na centimetry

$$1\ px = 0.0264\ cm$$

- následně pro vzdálenost x cm respondenta od monitoru, dostaneme hodnotu amplitudy sakády

$$a_{i+1} = arctan\Big(\frac{d\left(f_i, f_{i+1}\right)\ \ v\ cm}{x}\Big)$$

Kapitola 5

Praktická část

Cílem praktické části bylo vytvoření R balíčku a Shiny aplikace pro eye-tracking data. V následující kapitole si představíme všechny funkce, které vytvořený balíček `eyetRack` obsahuje a ukážeme, jaké možnosti přináší vytvořená Shiny aplikace. Funkčnost balíčku a aplikace budeme prezentovat na datových sadách, které poskytla Katedra Geoinformatiky Univerzity Palackého v Olomouci.

5.1. R balíček eyetRack

V nově vytvořeném balíčku `eyetRack` najdeme 7 základních funkcí pro zpracování eye-tracking dat. Dále zde najdeme vzorová data z eye-trackeru SMI, která může uživatel využít pro názorné použití funkcí. Balíček `eyetRack` je volně dostupný na serveru Github.com. Pro nainstalování balíčku provedeme příkaz

```
devtools::install_github("VKalabusova/eyetRack")
```

Všechny datové sady, které chceme analyzovat pomocí funkcí z balíčku `eyetRack`, musí obsahovat následující proměnné:

- označení obrázku, který respondent sledoval - Stimulus (Presented Media name)
- označení respondenta - Participant (Participant name)
- pořadí fixací - Index (Eye movement type index)
- začátek fixací - Event Start Trial Time [ms]
- konec fixací - Event End Trial Time [ms]
- délku jednotlivých fixací - Event Duration [ms] (Gaze event duration)
- x-ovou pozici fixace - Fixation Position X [px] (Fixation point X)

- y-ovou pozici fixace - Fixation Position Y [px] (Fixation point Y)
- označení oblastí zájmu, do kterých se respondent díval - AOI Name (AOI hit - například AOI hit [10 - left])
- označení o jaký pohyb oka se jednalo - Eye movement type (pouze pro eye-trackery Tobii)

V závorkách najdeme označení požadovaných proměnných pro eye-tracker Tobii. Ukázková data z eye-trackeru SMI, která `eyetRack` obsahuje, najdeme v balíčku pod názvem `data_SMI`. Jejich ukázku můžeme vidět na obrázku 5.1.

Trial	Stimulus	Participant	Jmeno	Pohlavi	Index	Event Start Trial Time [ms]	Event End Trial Time [ms]	Event Duration [ms]	Fixation Position X [px]	Fixation Position Y [px]	AOI Name
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	1	1.9	321.9	320.0	924.7	631.3	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	2	385.9	521.9	136.0	922.6	1072.6	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	3	557.9	705.8	147.9	949.2	1043.1	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	4	729.8	941.8	212.0	1043.7	1040.8	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	5	973.8	1177.8	204.0	869.8	1039.9	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	6	1213.8	3789.4	2575.6	884.6	1040.7	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	7	3973.4	6573.1	2599.7	883.0	1046.4	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	8	6612.9	6896.9	284.0	1051.3	1039.0	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	9	6964.9	7252.8	287.9	1479.5	722.3	B
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	10	7332.9	7760.8	427.9	1235.9	622.2	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	11	7852.7	8524.6	671.9	920.1	1035.6	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	12	8696.6	8920.6	224.0	807.0	375.4	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	13	9004.5	9468.5	463.9	1521.3	761.8	B
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	14	9552.6	10588.3	1035.7	888.6	1034.6	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	15	10632.3	11048.3	416.0	882.5	1026.0	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	16	11212.3	11708.2	495.9	883.2	1028.3	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	17	11744.2	12040.2	296.0	1054.4	1032.9	Scale
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	18	12124.1	12400.1	276.0	1442.1	720.9	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	19	12440.1	12640.0	199.9	1351.4	654.3	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	20	12676.0	13084.0	408.0	1232.1	636.0	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	21	13116.0	13715.9	599.9	1121.0	615.7	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	22	13779.8	14523.8	743.9	929.6	523.0	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	23	14579.7	14923.7	344.0	814.3	414.8	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	24	14983.7	15127.7	143.9	750.7	335.8	Map
Trial034	09-M1-CX-SI-VE.jpg	P16	Trnová	Žena	25	15151.7	16127.5	975.8	706.6	348.4	Map

Obrázek 5.1: Ukázková data z eye-trackeru SMI, která najdeme v balíčku.

5.1.1. basic analysis

První funkcí, kterou v balíčku najdeme, je funkce `basic_analysis`. Pomocí této funkce získá úvodní představu o počtu a délkách fixací. Vstupními parametry jsou:

- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název stimulu, na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta či respondentů, které budeme zkoumat.

Syntaxe volání funkce je ve tvaru

```
basic_analysis(data = data, eye_tracker = "SMI",
    object = "09-M1-CX-SI-VE.jpg",
    participant = c("P14", "P16", "P01", "P05"))
```

Analyzujeme datovou sadu, kterou jsme označili `data`, získanou z eye-trackeru SMI, subjekt pozoroval mapu označenou "09-MI-CX-SI-VE.jpg"a zajímají nás respondenti P14, P16, P01 a P05.

Výstupem je tabulka 5.1, ve které najdeme proměnné

- `Participant` - označení respondenta/respondentů,
- `NoF` - počet fixací, které daný respondent udělal (Number of Fixations),
- `TTFF` - dobu do první fixace v ms (Time to First Fixation) - pokud využíváme eye-trackery Tobii, pak tato informace není v datech obsažena, proto výsledkem bude NA,
- `FFD` - dobu první fixace v ms (First Fixation Duration),
- `TD` - celkovou dobu fixací v ms (Total Duration),
- `AFD` - průměrnou dobu fixace v ms (Average Fixation Duration)

Participant	**NoF**	**TTFF**	**FFD**	**TD**	**AFD**
P14	81	1.6	263.9	36395.8	449.33
P16	25	1.9	320.0	14489.8	579.59
P01	25	2.6	339.9	6907.4	276.30
P05	56	1.4	216.0	11903.3	212.56

Tabulka 5.1: Výstup funkce basic_analysis pro eye-tracker SMI.

Pro eye-tracker Tobii a respondenty 6 a 19, kteří pozorovali obrázek označený 10.jpg, bychom jako výstup dostali tabulku 5.2.

Participant	**NoF**	**TTFF**	**FFD**	**TD**	**AFD**
Participant6	13	NA	240	3483	267.92
Participant19	15	NA	327	3050	203.33

Tabulka 5.2: Výstup funkce basic_analysis pro eye-tracker Tobii.

5.1.2. barplot_AOI

Další funkci, která je součástí balíčku `eyetRack`, je funkce `barplot_AOI`. Umožňuje nám vykreslit počty a délky fixací v jednotlivých oblastech zájmu. Vytvoří 2 sloupcové grafy, kde první udává informaci o počtu fixací v jednotlivých oblastech zájmu a druhý poskytuje informaci o délce času stráveného v jednotlivých oblastech zájmu.

Vstupními parametry funkce jsou

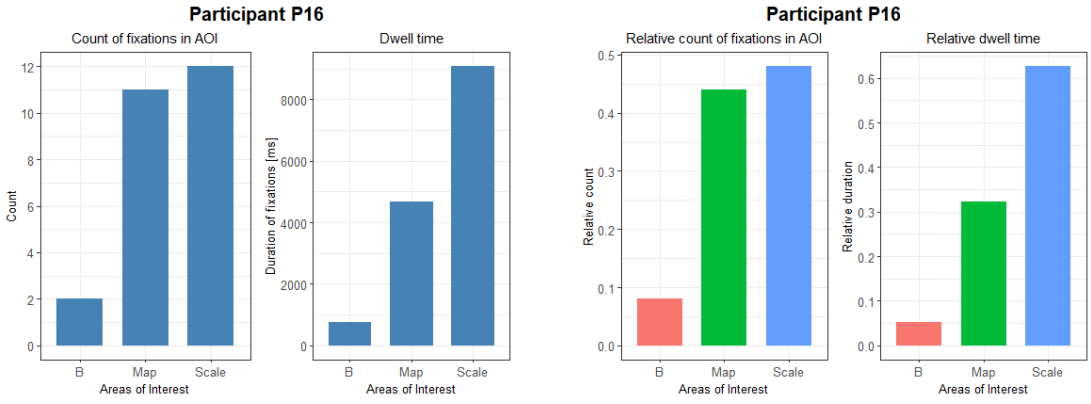
- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název stimulu, na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta, kterého budeme zkoumat,
- `count` - informace, zda chceme výsledky v absolutních, či relativních hodnotách (`"absolute"`/`"relative"`). Výchozím nastavením jsou absolutní počty,
- `same_col` - informace, zda chceme, aby všechny sloupce měly stejnou barvu (TRUE/FALSE). Výchozím nastavením je stejná barva.
- `bar_col` - specifikace barvy, která bude ve všech sloupcích použita. Výchozím nastavením je světle modrá,
- `col_man` - proměnná, která označuje, zda si chceme sami volit barvy, které budou použity (pokud same_col = FALSE). Hodnoty jsou TRUE (manuální volba) a FALSE (necháme R, aby samo vygenerovalo vektor barev dle počtu oblastí zájmů),
- `col_set` - pokud col_man nastavíme na TRUE a budeme si chtít zvolit barvy dle sebe, pak do proměnné col_set zadáme vektor barev, které chceme použít, počet barev odpovídá počtu oblastí zájmů,
- `bar_width` - šířka sloupců v grafu,
- `top_size` - velikost společného nadpisu,
- `title_size` - velikost nadpisu,
- `x_size` - velikost popisků na ose x,
- `x_labsize` - velikost textu na ose x,

- `x_angle` úhel popisků na ose x,
- `y_size` - velikost popisků na ose y,
- `y_labsize` - velikost textu na ose y,
- `y_angle` - úhel popisků na ose y.

Syntaxe volání funkce je ve tvaru

```
barplots_AOI(data = data, eye_tracker = "SMI",
             object = "09-M1-CX-SI-VE.jpg",
             participant = "P16", count = "absolute",
             same_col = T, bar_col = "steelblue",
             bar_width = 0.7, top_size = 15,
             title_size = 11, col_man = F,
             col_set = NULL, x_size = 10, x_labsize = 10,
             x_angle = 0, y_size = 10, y_labsize = 10,
             y_angle = 0
             )
```

Výstup funkce pro respondenta P16, který sledoval obrázek 09-MI-CX-SI-VE.jpg můžeme vidět na obrázku 5.2. Vidíme, že ačkoliv počty fixací v oblastech mapy a měřítka byly téměř shodné, tak výrazně větší čas strávil respondent pozorováním měřítka. V pravé části obrázku 5.2 máme pro ukázku vykresleny barploty s relativními počty a automaticky vygenerovanými barvami.



Obrázek 5.2: Výstup funkce barplots_AOI. Vlevo grafy s absolutními hodnotami a stejnou barvou sloupců. Vpravo relativní počty a automaticky vygenerované barvy.

5.1.3. visualization

Pro vizualizaci fixací a sakád využijeme funkci `visualization`. Fixace můžeme znázornit samostatně, nebo si můžeme vybrat některou z metod, které využívá rekurentní kvantifikační analýza.

Vstupními parametry jsou:

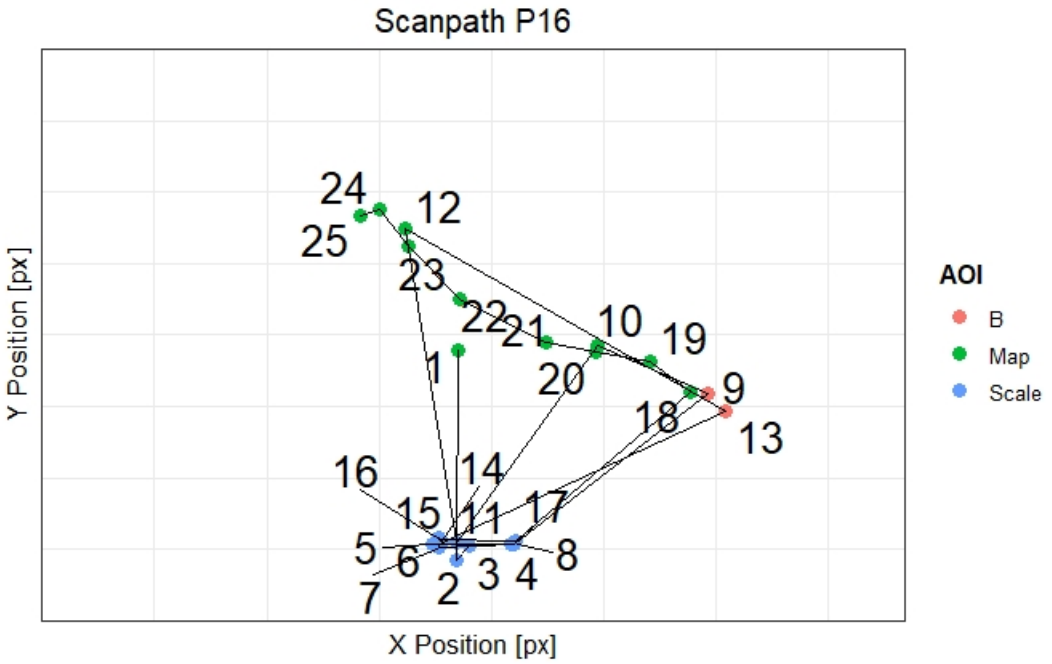
- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název stimulu, na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta, kterého budeme zkoumat,
- `method` - zvolená metoda vizualizace (`"Simple"` - pouze fixace a sakády, `"Simple with time"` - fixace s délkami fixací, `"Fixed distance"` - metoda fixní vzdálenosti, kolem fixací se vykreslí kružnice o volitelném poloměru, `"Fixed grid"` - metoda fixní mřížky, zobrazí se mřížka s volitelnou stranou čtverce, `"AOI"` - metoda oblastí zájmů, barvy bodů jednotlivých fixací jsou zbarveny dle oblasti zájmu, ve které leží, `"AOI with time"` - metoda oblastí zájmu, kde uvažujeme dobu trvání fixací,
- `fig` - možnost, zda chceme vykreslit obrázek Stimulu pod graf (TRUE/-FALSE),
- `img` - obrázek stimulu,
- `img_AOI` - obrázek stimulu rozdělený na oblasti zájmu,
- `scaling` - relativní velikost bubliny odpovídající délce fixace. Základní hodnota je 100,
- `r` - poloměr kružnice v pixelech pro metodu fixní vzdálenosti, velikost čtverce v mřížce pro metodu fixní mřížky. Základní hodnota je nastavena na hodnotu 60 pixelů,
- `time` - zda uvažujeme dobu trvání fixací (TRUE/FALSE),
- `size` - vektorově zadané rozměry obrázku, na který se respondent díval, první složka označuje šířku, druhá složka značí výšku. Výchozí nastavení c(1920,1200),

- `col_man` - proměnná, která označuje, zda si chceme sami volit barvy u metody AOI. Hodnoty jsou TRUE (manuální volba) a FALSE (necháme R, aby samo vygenerovalo vektor barev dle počtu oblastí zájmů),
- `col_set` - pokud col_man nastavíme na TRUE a budeme si chtít zvolit barvy dle sebe, pak do proměnné col_set zadáme vektor barev, které chceme použít, počet musí odpovídat počtu oblastí zájmu,
- `point_size` - velikost bodů v grafu,
- `point_col` - barva bodů pro metodu fixní vzdálenosti a fixní mřížky,
- `bubble_col` - barva bubliny (pokud uvažujeme dobu trvání fixací),
- `title_size` - velikost nadpisu,
- `x_labsize` - velikost textu na ose x,
- `y_labsize` - velikost textu na ose y,
- `legend_size` - velikost textu v legendě (metody s oblastmi zájmu),
- `text_size` - velikost textu v grafu,
- `sac_size` - šířka linie sakád,
- `circle_size` - šířka linie kružnic (metoda fixní vzdálenosti),
- `grid_size` - šířka linie mřížky (metoda fixní mřížky),
- `lightness` - průhlednost bubliny (pokud uvažujeme dobu trvání fixací).

Syntaxe volání funkce je ve tvaru

```
visualization(data = data, eye_tracker = "SMI",
              object = "09-M1-CX-SI-VE.jpg",
              participant = "P16", method = "AOI",
              fig = FALSE, img = NULL, img_AOI = NULL,
              scaling = 100, r = 60 , time = FALSE,
              size = c(1920, 1200), col_man = FALSE,
              col_set = c(), point_size = 3,
              point_col = "steelblue",
              bubble_col = "steelblue", title_size = 15,
              x_labsize = 13, y_labsize = 13,
              legend_size = 10,  text_size = 7,
              sac_size = 0.2, circle_size = 0.1,
              grid_size = 0.3, lightness = 0.4)
```

Výstup pro vizualizaci fixací respondenta P16 metodou oblastí zájmu s automaticky generovanými barvami bez vykreslení obrázku pod graf můžeme vidět na obrázku 5.3.



Obrázek 5.3: Vizualizace fixací pomocí metody AOI bez uvažování doby trvání fixací.

5.1.4. recurrence

Pro vytvoření matice rekurence a vizualizaci rekurentních bodů pomocí grafu rekurence, použijeme funkci `recurrence`.

Vstupními parametry funkce jsou:

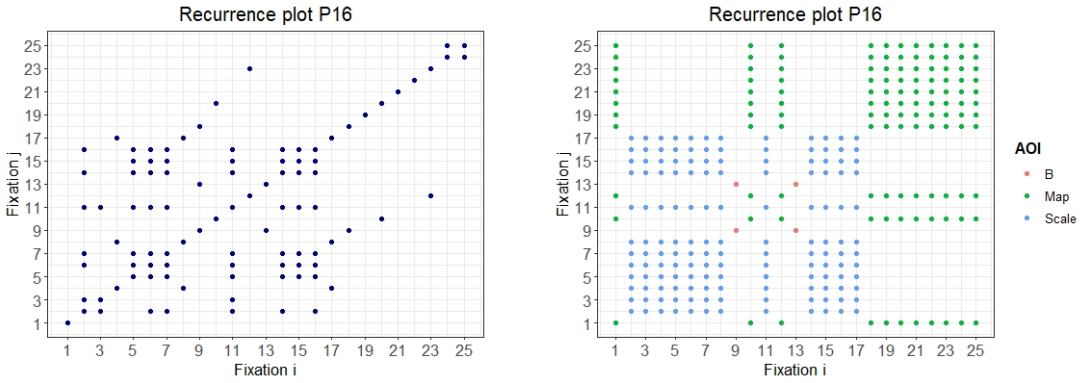
- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název objektu, na který se respondent díval a který chceme analyzovat,
- `participant` - název respondenta, kterého chceme analyzovat

- `method` - metoda rekurentní analýzy (`"Fixed distance"` - fixní vzdálenost, `"Fixed grid"` - fixní mřížka, `"AOI"` - oblasti zájmu),
- `r` - poloměr kružnice v pixelech pro metodu fixní vzdálenosti, velikost čtverce v mřížce pro metodu fixní mřížky. Základní hodnota je nastavena na hodnotu 60 pixelů,
- `scaling` - relativní velikost bubliny odpovídající délce fixace. Základní hodnota je 100,
- `size` - vektorově zadané rozměry obrázku, na který se respondent díval, první složka označuje šířku, druhá složka značí výšku. Výchozí nastavení c(1920,1200),
- `time` - zda uvažujeme dobu trvání fixací (TRUE/FALSE),
- `col_man` - proměnná, která označuje, zda si chceme sami volit barvy u metody AOI. Hodnoty jsou TRUE (manuální volba) a FALSE (necháme R, aby samo vygenerovalo vektor barev dle počtu oblastí zájmů),
- `col_set` - pokud col_man nastavíme na TRUE a budeme si chtít zvolit barvy dle sebe, pak do proměnné col_set zadáme vektor barev, které chceme použít, počet musí odpovídat počtu oblastí zájmu,
- `point_size` - velikost bodů v grafu,
- `point_col` - barva bodů pro metodu fixní vzdálenosti a metodu fixní mřížky,
- `bubble_col` - barva bublin představující dobu trvání fixací (pokud uvažujeme dobu trvání fixací),
- `title_size` - velikost nadpisu,
- `legend_size` - velikost legendy (pro metodu AOI),
- `x_labsize` - velikost textu na ose x,
- `y_labsize` - velikost textu na ose y,
- `x_size` - velikost popisků na ose x,
- `y_size` - velikost popisků na ose y,
- `break` - rozdělení na osách,
- `lightness` - průhlednost bubliny (pokud uvažujeme dobu trvání fixací).

Syntaxe volání funkce je ve tvaru

```
recurrence(data = data, eye_tracker = "SMI",
           object = "09-M1-CX-SI-VE.jpg",
           participant = "P16", method = "Fixed distance",
           r = 60, scaling = 100,  size = c(1920, 1200),
           time = FALSE, col_man = FALSE, col_set = c(),
           point_size = 1.5, point_col = "darkblue",
           bubble_col = "steelblue", title_size = 15,
           legend_size = 10, x_labsize = 13,
           y_labsize = 13, x_size = 13, y_size = 13,
           breaks = 2, lightness = 0.2)
```

Výstupem funkce je tabulka obsahující matici rekurence a graf rekurence, který můžeme vidět na obrázku 5.4 vlevo. Graf rekurence pro metodu oblastí zájmu bez uvažování doby času a s automaticky generovanými barvami najdeme na obrázku 5.4 vpravo.



Obrázek 5.4: Výstup funkce recurrence. Vlevo graf rekurence pro metodu fixní vzdálenosti bez uvažování doby času. Vpravo graf rekurence pro metodu oblastí zájmu bez uvažování doby času s automaticky vygenerovanými barvami.

5.1.5. measures

Pokud bychom chtěli spočítat základní míry, které rekurentní kvantifikační analýza využívá, pak použijeme funkci `measures`.

Vstupními parametry funkce jsou:

- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název objektu, na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta/respondentů, které budeme zkoumat,
- `method` - metoda rekurentní analýzy (`"Fixed distance"` - fixní vzdálenost, `"Fixed grid"` - fixní mřížka, `"AOI"` - oblasti zájmu),
- `r` - poloměr kružnice v pixelech pro metodu fixní vzdálenosti, velikost čtverce v mřížce pro metodu fixní mřížky. Základní hodnota je nastavena na hodnotu 60 pixelů,
- `size` - vektorově zadané rozměry obrázku, na který se respondent díval, první složka označuje šířku, druhá složka značí výšku. Výchozí nastavení c(1920, 1200),
- `L` - minimální délka diagonálních, horizontálních a vertikálních linií. Základní hodnota je nastavena na hodnotu 2,
- `time` - informace, zda uvažujeme dobu trvání fixací (TRUE/FALSE). Výchozím nastavením je FALSE.

Syntaxe volání funkce je ve tvaru

```
measures(data = data, eye_tracker = "SMI",
         object = "09-M1-CX-SI-VE.jpg",
         participant = c("P14", "P16", "P01", "P05"),
         method = "Fixed distance", r = 60,
         size = c(1920, 1200),  L = 2, time = FALSE)
```

Výstupem funkce je tabulka, ve které najdeme následující proměnné:

- `Participant` - označení respondenta,
- `R` - počet, resp. součet doby trvání, rekurentních bodů v rekurentním grafu,

- `REC` - míra rekurence [%]
- `DET` - míra determinismu [%]
- `LAM` - míra laminarity [%]
- `CORM` - CORM

Pro respondenty P14, P16, P01 a P05 při použití metody fixní vzdálenosti bez uvažování doby trvání fixací obdržíme jako výstup tabulku 5.3

Participant	**R**	**REC**	**DET**	**LAM**	**CORM**
P14	239	7.377	36.820	46.234	29.304
P16	36	12.000	36.111	50.000	26.620
P01	17	5.667	17.647	17.647	26.716
P05	179	11.623	49.721	52.235	23.840

Tabulka 5.3: Výstup funkce measures pro metodu fixní vzdálenosti bez uvažování doby trvání fixací.

Pokud bychom uvažovali stejné vstupní hodnoty a uvažovali bychom dobu trvání fixací, pak výstup funkce `measures` vidíme v tabulce 5.4.

Participant	**R**	**REC**	**DET**	**LAM**	**CORM**
P14	195798	6.725	39.644	47.016	1.924
P16	62588	17.998	46.514	65.025	4.279
P01	9490	5.725	26.048	15.195	1.659
P05	80908	12.358	53.036	52.123	3.013

Tabulka 5.4: Výstup funkce measures pro metodu fixní vzdálenosti s uvažováním doby trvání fixací.

5.1.6. coeffK_single

Pomocí funkce `coeffK_single` můžeme zjistit, jaké prvky K_i tvoří výsledný koeficient K.

Vstupními parametry funkce jsou:

- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,

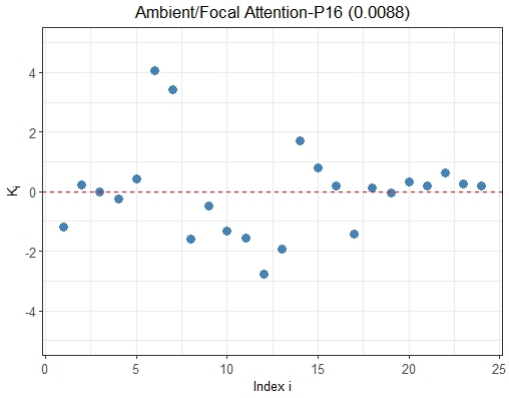
- `object` - název stimulu (objektu), na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta, kterého budeme zkoumat,
- `distance` - vzdálenost respondenta od monitoru (v cm). Základní hodnota je nastavena na 60 cm,
- `lim` - limity na y-ové ose v grafu. Výchozí nastavení je c(-10, 10),
- `point_col` - barva bodů v grafu,
- `point_size` - velikost bodů v grafu,
- `title_size` - velikost nadpisu,
- `x_size` - velikost popisků na ose x,
- `x_labsize` - velikost textu na ose x,
- `x_angle` - úhel popisků na ose x,
- `y_size` - velikost popisků na ose y,
- `y_labsize` - velikost textu na ose y,
- `y_angle` - úhel popisků na ose y.

Syntaxe volání funkce je ve tvaru

```
coeffK_single(data = data, eye_tracker = "SMI",
              object = "09-M1-CX-SI-VE.jpg",
              participant = "P16",
              distance = 60, lim = c(-10,10),
              point_col = "steelblue", point_size = 3,
              title_size = 13, x_size = 10,
              x_angle = 0, x_labsize = 10, y_size = 10,
              y_angle = 0, y_labsize = 10)
```

Výstup funkce můžeme vidět na obrázku 5.5. Jedná se o tabulku s prvky K_i, které tvoří koeficient K. Druhou složkou výstupu je graf, ve kterém jsou vykresleny prvky K_i. V nadpisu grafu je zobrazena i výsledná hodnota koeficientu K.

	K_i
1	-1.174
2	0.213
3	0.001
4	-0.235
5	0.421
6	4.057
7	3.411
8	-1.595
9	-0.492
10	-1.328



Obrázek 5.5: Výstup funkce coeffK_single. Vlevo ukázka tabulky jednotlivých K, ze kterých se celkový koeficient K skládá. Vpravo jejich vykreslení.

5.1.7. coeff_K

Pro rozhodnutí o ohniskové nebo okolní pozornosti a zjištění koeficientu K využijeme funkci `coeff_K`.

Vstupní parametry funkce jsou:

- `data` - datová sada,
- `eye_tracker` - označení eye-trackeru, ze kterého datová sada pochází - `"SMI"` nebo `"Tobii"`,
- `object` - název stimulu (objektu), na který se respondent díval a který chceme analyzovat,
- `participant` - označení respondenta/respondentů, které budeme zkoumat,
- `distance` - vzdálenost respondenta od monitoru (v cm). Základní hodnota je nastavena na 60 cm,
- `lim` - limity na y-ové ose v grafu. Výchozí nastavení je c(-1, 1),
- `point_col` - barva bodů v grafu,
- `point_size` - velikost bodů v grafu,
- `title_size` - velikost nadpisu,
- `x_size` - velikost popisků na ose x,
- `x_labsize` - velikost textu na ose x,

- `x_angle` úhel popisků na ose x,
- `y_size` - velikost popisků na ose y,
- `y_labsize` - velikost textu na ose y,
- `y_angle` - úhel popisků na ose y.

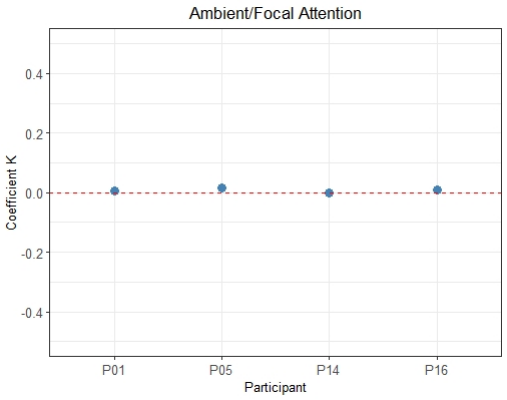
Syntaxe volání funkce je ve tvaru

```
coeff_K(data = data, eye_tracker = "SMI",
        object = "09-M1-CX-SI-VE.jpg",
        participant = c("P14", "P16", "P01", "P05"),
        distance = 60, lim = c(-10,10),
        point_col = "steelblue", point_size = 3,
        title_size = 13, x_size = 10,
        x_angle = 0, x_labsize = 10,
        y_size = 10, y_angle = 0, y_labsize = 10)
```

Výstupem je tabulka s koeficienty K pro zvolené respondenty. Druhou složkou výstupu je graf, ve kterém jsou koeficienty K pro jednotlivé respondenty vykresleny.

Pro respondenty P14, P15, P01 a P05 můžeme vidět výstup funkce na obrázku 5.6.

	Participant	Coeff_K
1	P14	-0.0015
2	P16	0.0088
3	P01	0.0056
4	P05	0.0144



Obrázek 5.6: Výstup funkce coeff_K. Vlevo tabulka obsahující hodnoty koeficientu K. Vpravo vykreslení hodnot koeficientu K.

5.2. Shiny aplikace eyetRack

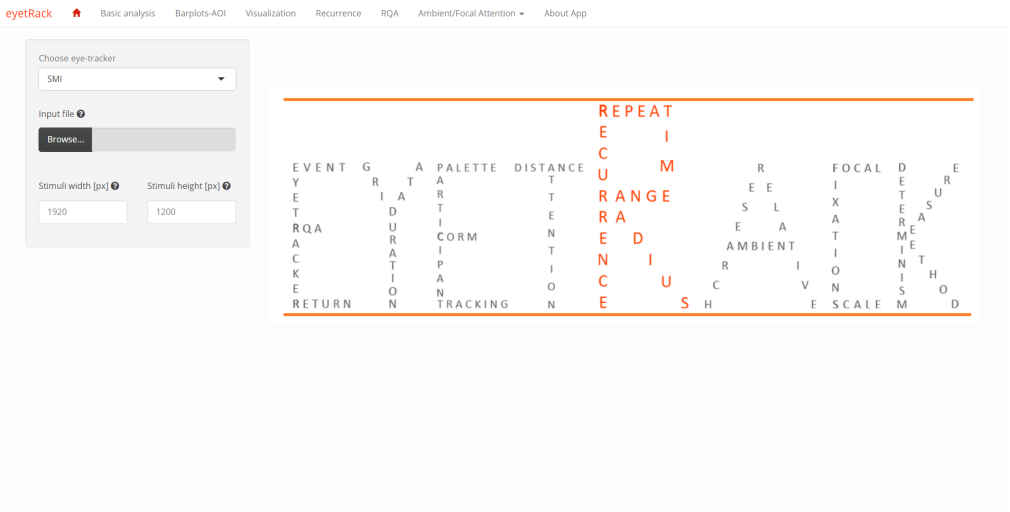
V této kapitole si představíme novou aplikaci, která využívá funkce z balíčku `eyetRack`. Aplikaci `eyetRack` můžeme najít na odkaze https://vkalabusova.shinyapps.io/shiny/ nebo na stránkách Eye-tracking Group Katedry Geoinformatiky Univerzity Palackého v Olomouci v záložce Tools. (http://eyetracking.upol.cz/tools/).

Na úvodní stránce aplikace, kterou můžeme vidět na obrázku 5.7, nahráváme data ve formátu .txt nebo .tsv. Data z eye-trackeru SMI nebo Tobii musí obsahovat proměnné:

- označení obrázku, který respondent sledoval - Stimulus (Presented Media name)
- označení respondenta - Participant (Participant name)
- pořadí fixací - Index (Eye movement type index)
- začátek fixací - Event Start Trial Time [ms]
- konec fixací - Event End Trial Time [ms]
- délku jednotlivých fixací - Event Duration [ms] (Gaze event duration)
- x-ovou pozici fixace - Fixation Position X [px] (Fixation point X)
- y-ovou pozici fixace - Fixation Position Y [px] (Fixation point Y)
- označení oblastí zájmu, do kterých se respondent díval - AOI Name (AOI hit - například AOI hit [10 - left])
- označení o jaký pohyb oka se jednalo - Eye movement type (pouze pro eye-trackery Tobii)

V závorkách najdeme označení požadovaných proměnných pro eye-tracker Tobii.

Nejdříve vybereme, ze kterého eye-trackeru data pochází. Vybírat můžeme z eye-trackerů značky SMI a Tobii. Volíme rozměry stimulu, na který se respondent díval. Výchozí nastavení je šířka 1920 pixelů a výška 1200 pixelů. Pro ukázku jsem použila stejná data z eye-trackerů SMI jako v případě R balíčku.



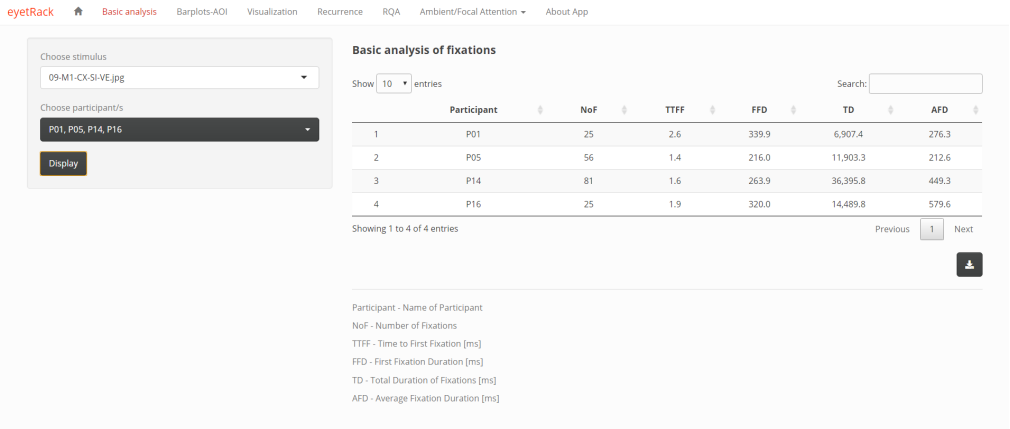
Obrázek 5.7: Úvodní stránka aplikace eyetRack.

V první záložce `Basic analysis`, můžeme získat úvodní představu o počtu a délkách fixací. Vybereme označení obrázku, který chceme analyzovat a vybereme jednoho či více respondentů. Na obrázku 5.8 jsme vybrali stimulus označený jako 09-M1-CX-SI-VE.jpg a respondenty P01, P05, P14 a P16. Jako výstup se nám objeví tabulka obsahující proměnné:

- `Participant` - označení respondenta/respondentů,
- `NoF` - počet fixací, které daný respondent udělal (Number of Fixations),
- `TTFF` - dobu do první fixace v ms (Time to First Fixation) - pokud využíváme eye-trackery Tobii, pak tato informace není v datech obsažena, proto výsledkem bude NA,
- `FFD` - dobu první fixace v ms (First Fixation Duration),
- `TD` - celkovou dobu fixací v ms (Total Duration),
- `AFD` - průměrnou dobu fixace v ms (Average Fixation Duration)

Následně můžeme tabulku uložit ve formátu .csv.

V základní analýze bude pro Tobii data u proměnné TTFF prázdné místo, protože tato data neobsahují informace o začátcích a koncích fixací.

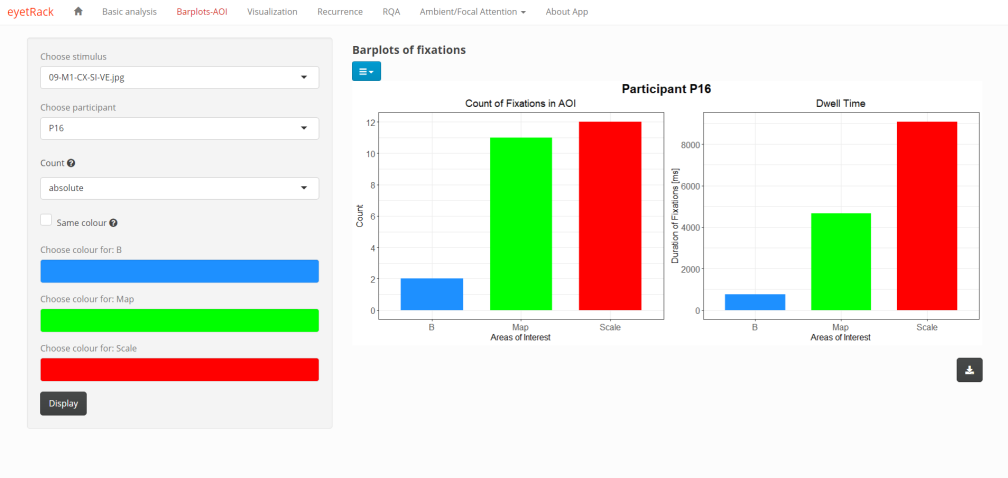


Obrázek 5.8: Základní analýza v Shiny aplikaci pro respondenty P01, P05, P14 a P16.

V záložce Barplot-AOI si můžeme vykreslit počty a délku fixací v jednotlivých oblastech zájmů. V levé části vybereme stimulus, který chceme analyzovat a jednoho z respondentů. Dále si můžeme vybrat, zda chceme zobrazit počty absolutního či relativního typu. Ve výchozím nastavení je zvolen absolutní počet. Barvy jednotlivých sloupců představující oblasti zájmu můžeme nastavit dle našich představ, nebo můžeme využít přednastavených barev. Také je možné zvolit, aby měly všechny sloupce stejnou barvu a to pomocí zaškrtnutí políčka `Same colour`,

Vykreslí se nám vedle sebe dva barploty. Na barplotu vlevo vidíme počet fixací v jednotlivých oblastech zájmu. Na barplotu vpravo můžeme vidět dobu trvání fixací v jednotlivých oblastech zájmu, resp. dwell time. Na obrázku 5.9 můžeme například vidět, že i když fixací bylo v oblastech mapy a měřítka téměř stejně, tak výrazně delší dobu respondent věnoval měřítku.

V záložce vedle grafu můžeme upravovat další vstupní parametry, jako jsou velikost nadpisu, velikost nadpisu jednotlivých barplotů, šířka sloupců, velikost popisků os, velikost textu na osách a úhly popisků na osách. Graf můžeme uložit ve formátu .pdf



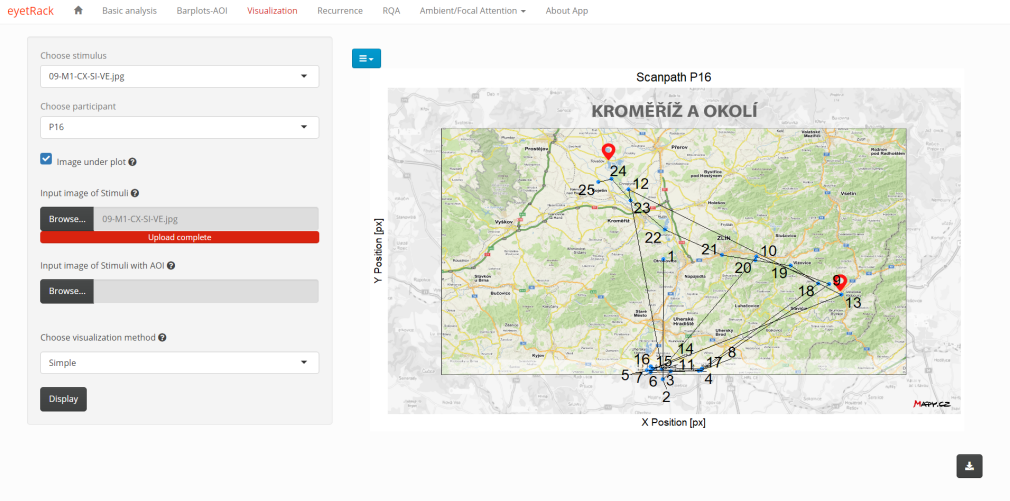
Obrázek 5.9: Shiny aplikace Barplots-AOI pro respondenta P16.

Fixace si můžeme zobrazit v záložce `Vizualization`. Vybereme označení obrázku, který chceme analyzovat a respondenta, který nás zajímá. Pokud chceme pod graf vložit obrázek stimulu, na který se respondent díval, pak po zaškrtnutí políčka `Image under plot`, máme možnost vložit obrázek stimulu, či obrázek stimulu rozděleného na oblasti zájmu. Vše vkládáme ve formátu .jpg. Následně zvolíme metodu pro vizualizaci. Na výběr máme z metod:

- `Simple` - vykreslení fixací a sakád,
- `Simple with time` - vykreslení fixací s délkami trvání a sakád,
- `Fixed distance` - kolem každé z fixací je vykreslena kružnice o poloměru, který si zvolíme,
- `Fixed grid` - vykreslení fixací a sakád s fixní mřížkou se zvolenou délkou strany čtverce,
- `AOI` - vykreslení fixací a sakád, kde barvy fixací se liší podle oblastí zájmu, ve kterých jsou umístěny (můžeme volit),
- `AOI with time` - vykreslení fixací s délkami fixací a sakád, kde barvy fixací se liší dle oblastí zájmu, ve kterých jsou umístěny,

V záložce vedle grafu můžeme měnit další vstupy grafu, jako jsou velikost nadpisu, velikost textu na osách, velikost textu v grafu, velikost bodů, barvy bodů, šířky linií (sakád, kružnic, mřížky), u metod, které obsahují délku fixací, můžeme volit velikost a průhlednost bublin. Vše můžeme uložit ve formátu .pdf.

Na obrázku 5.10 jsme vizualizovali fixace respondenta P16 pomocí metody `Simple` s umístěním mapy pod graf.



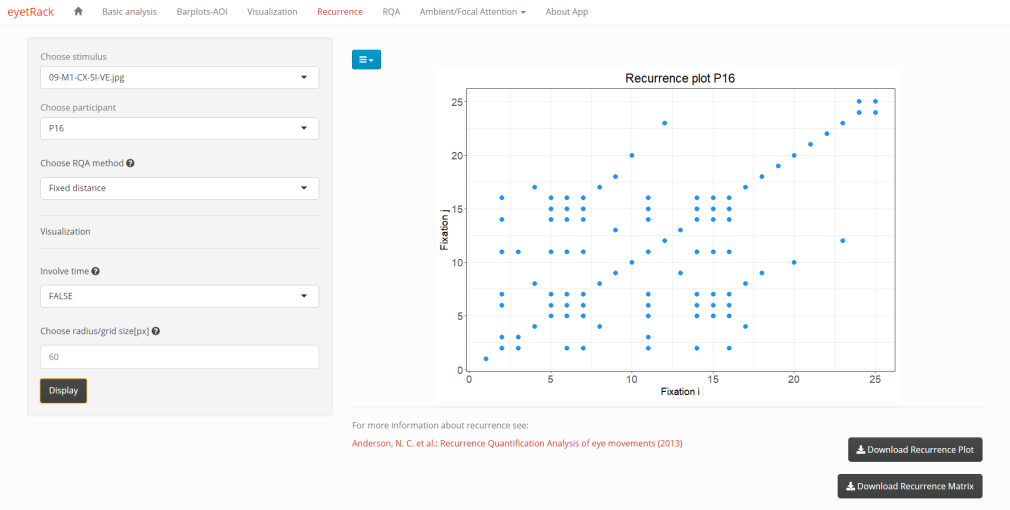
Obrázek 5.10: Shiny aplikace vizualizace fixací.

Pro zobrazení rekurentního grafu využijeme záložku `Recurrence`. Vybereme označení obrázku, který chceme analyzovat a respondenta, který nás zajímá. Zvolíme metodu rekurentní kvantifikační analýzy, kterou chceme použít (Fixed distance, Fixed grid, AOI) a zvolíme, zda uvažujeme dobu trvání fixací, či nikoliv. Ve výchozím nastavení máme přednastavenu metodu fixní vzdálenosti bez uvažování doby trvání fixací. U metody fixní vzdálenosti si můžeme zvolit poloměr pro rekurentní fixace, u metody fixní mřížky volíme velikost čtverců mřížky.

V záložce vedle grafu můžeme měnit další vstupy grafu, jako jsou velikost nadpisu, velikost textu na osách, velikost bodů, barvy bodů, u metod, které obsahují délku fixací můžeme volit velikost a průhlednost bublin, které představují dobu trvání fixací.

Graf rekurence můžeme uložit ve formátu .pdf. Můžeme také uložit matici rekurence ve formátu .csv, ta však není zobrazena.

Pro metodu fixní vzdálenosti a respondenta P16 můžeme vidět graf rekurence na obrázku 5.11.



Obrázek 5.11: Shiny aplikace graf rekurence.

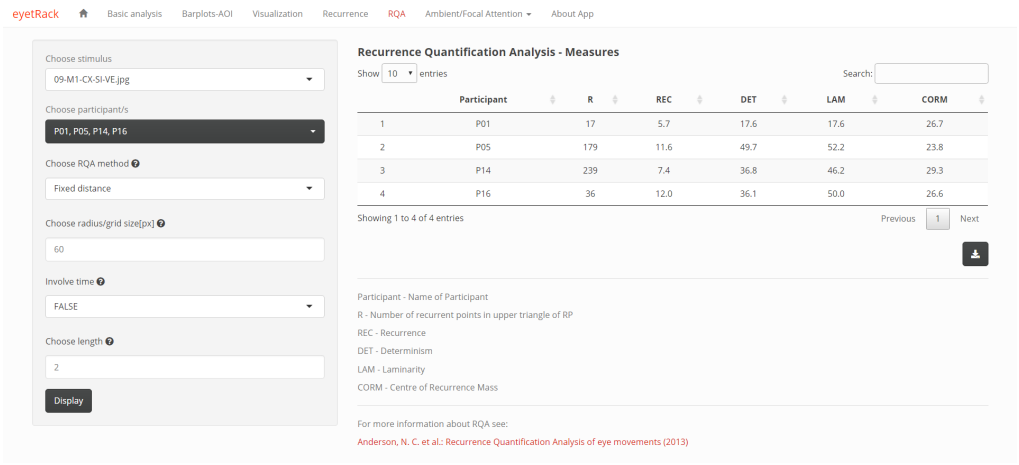
Míry rekurentní kvantifikační analýzy najdeme v záložce `RQA`. Vybereme označení obrázku, který chceme analyzovat a respondenta či respondenty, kteří nás zajímají. Zvolíme metodu rekurentní kvantifikační analýzy, kterou chceme použít (Fixed distance, Fixed grid, AOI) a zvolíme, zda uvažujeme dobu trvání fixací, či nikoliv. Ve výchozím nastavení máme přednastavenu metodu fixní vzdálenosti bez uvažování doby trvání fixací. U metody fixní vzdálenosti si můžeme zvolit poloměr pro rekurentní fixace, u metody fixní mřížky volíme velikost čtverců mřížky. Dále můžeme volit minimální délku diagonál, resp. horizontál a vertikál.

Jako výstup se nám objeví tabulka obsahující proměnné:

- `Participant` - označení respondenta,
- `R` - počet, resp. součet doby trvání, rekurentních bodů v rekurentním grafu,
- `REC` - míra rekurence [%]
- `DET` - míra determinismu [%]
- `LAM` - míra laminarity [%]
- `CORM` - CORM

Následně můžeme tabulku uložit ve formátu .csv.

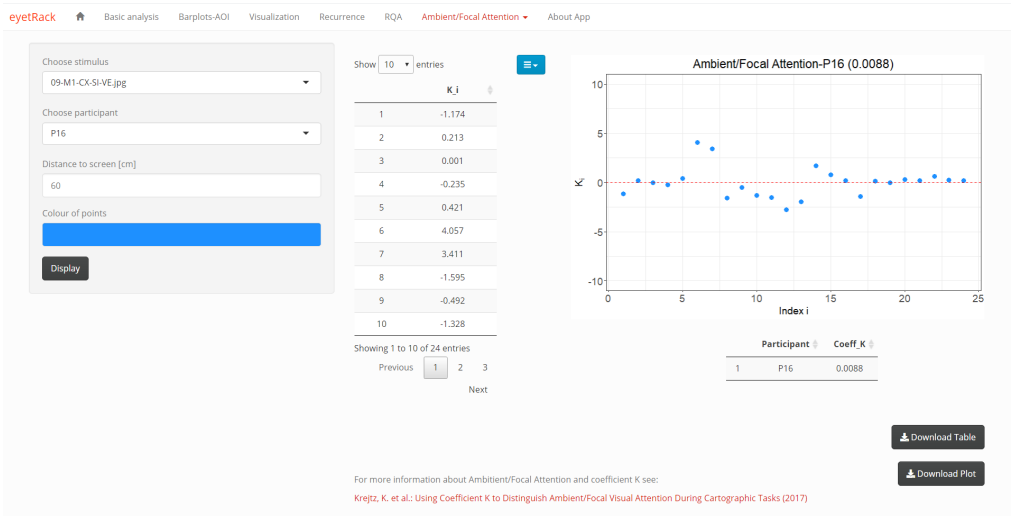
Pro respondenty P01, P05, P14 a P16 pro metodu fixní vzdálenosti s poloměrem 60 px dostaneme míry rekurentní kvantifikační analýzy, které vidíme na obrázku 5.12.



Obrázek 5.12: Shiny aplikace míry rekurentní kvantifikační analýzy

Pokud chceme zjistit u vybraného respondenta, z jakých hodnot se skládá koeficient K, pak využijeme záložku `Ambient/Focal Attention - Single`, kde po zvolení označení obrázku, který chceme analyzovat, vybrání respondenta a specifikaci vzdálenosti respondenta od monitoru dostaneme tabulku s hodnotami K_i, které tvoří koeficient K a graf s vykreslenými hodnotami K_i.

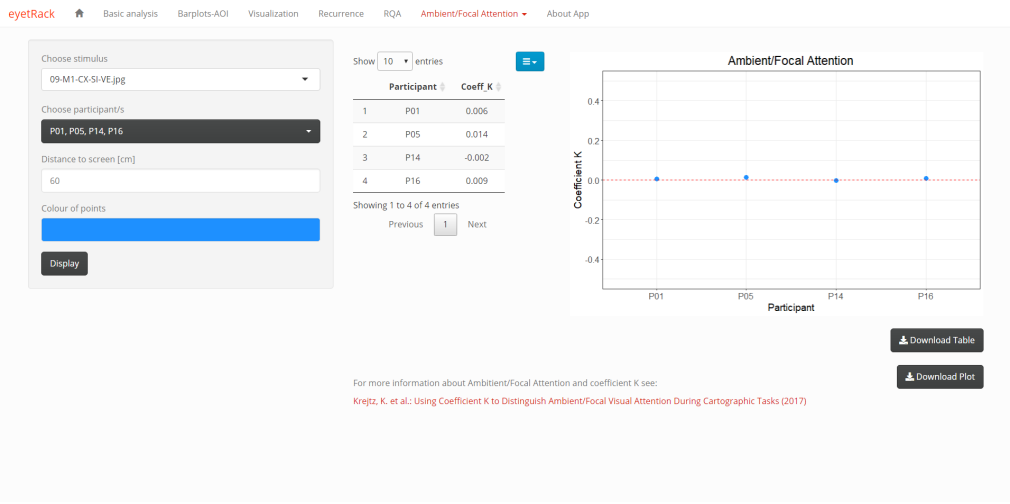
V záložce vedle grafu můžeme měnit další vstupy grafu, jako jsou velikost nadpisu, velikost textu na osách, velikost popisků na osách, velikost bodů, barvy bodů a limity na ose y. Tabulku si můžeme uložit ve formátu .csv a graf ve formátu .pdf. Na obrázku 5.13 můžeme vidět hodnoty K_i pro respondenta P16 včetně výsledného koeficientu K.



Obrázek 5.13: Shiny aplikace ohnisková a okolní pozornost pro 1 respondenta.

Pro porovnání koeficientu K různých respondentů využijeme záložku `Ambient/ Focal Attention - Multiple`, kde po zvolení označení obrázku, který chceme analyzovat a vybrání jednoho či více respondentů a specifikaci vzdálenosti respondentů od monitoru dostaneme tabulku s koeficienty K a graf s vykreslenými hodnotami koeficientu K dle vybraných respondentů jak můžeme vidět na obrázku 5.14.

V záložce vedle grafu můžeme měnit další vstupy grafu, jako jsou velikost nadpisu, velikost textu na osách, velikost popisků na osách, velikost bodů, barvy bodů a limity na ose y. Tabulku si můžeme uložit ve formátu .csv a graf ve formátu .pdf.



Obrázek 5.14: Shiny aplikace ohnisková a okolní pozornost.

Poslední záložka `About App` je věnována základním informacím o Shiny aplikaci a především návodu, jak s touto aplikací pracovat.

Závěr

Ve své práci jsem představila nově vytvořené nástroje určené pro analýzu eye-tracking dat, která pochází z eye-trackerů značky SMI nebo Tobii.

Prvním nástrojem, který pro analýzu eye-tracking dat můžeme využít, je R balíček `eyetRack`. Tento balíček obsahuje 7 základních funkcí, které nabízí základní analýzu počtu a délek fixací, vykreslení rozložení fixací v jednotlivých oblastech zájmu, vizualizaci fixací dle vybraných metod, výpočet rekurentní matice a vykreslení rekurentního grafu, výpočet měr rekurentní kvantifikační analýzy a v neposlední řadě výpočet a grafické zobrazení koeficientu K pro možné rozdělení mezi okolní a ohniskovou pozorností vybraných respondentů.

Další možností pro ty, kteří by se chtěli vyhnout programování v R, jsem představila nově vytvořenou aplikaci `eyetRack`, která je volně dostupná a využívá všechny funkce z výše zmíněného R balíčku.

Největším přínosem práce bylo právě vytvoření nové Shiny aplikace, která bude uživatelsky příjemná. Doufám, že se `eyetRack` stane rozšířeným nástrojem, který lidem usnadní analýzu dat v oblasti eye-trackingu.

Literatura

[1] Anderson N. C., Bischof W. F.,Laidlaw K. E. W., Risko E. F., Kingstone A.: *Recurrence quantification analysis of eye movements*. Behavior Research Methods, 2013.

[2] Carbonell I., de la Fuente J., Afriyie P.: *Using Mobile Eye Tracking and Coefficient K for Analysing Usability Trials* [online] Proceedings of 29th IAPRI Symposium on Packaging, Enschede, vol. 1, s. 345-357, 2019 [cit. 2022-04-12]. Dostupné z: https://www.researchgate.net/publication/339275027_Using_Mobile_Eye_Tracking_and_Coefficient_K_for_Analysing_Usability_Trials.

[3] Codecademy: *What is R used for?* [online] 2021 [cit. 2022-03-01]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-r-used-for/

[4] Kim I. S., Martin P., McMurry N., Halterman A.: *Instructions for Creating Your Own R Package* [online] 2018. [cit. 2022-02-12] Dostupné z: https://web.mit.edu/insong/www/pdf/rpackage_instructions.pdf.

[5] Klein Ch., Ettinger U.: *Eye Movement Research: An Introduction to its Scientific Foundations and Applications* [online] Springer Nature Switzerland AG, 2019 [cit. 2022-03-01]. Dostupné z: https://books.google.cz/books?id=biK3D wAAQBAJ&pg=PA633&dq=Christoph+Klein&hl=cs&sa=X&ved=0ahUKEwiD0b7I5brpAhWPnxQKHfJZAXgQ6AEIdDAI#v=onepage&q&f=false.

[6] Kodera J., Van Quang T.: *Vizuální nelineární rekurentní analýza a její aplikace na český akciový trh* [online] Politická ekonomie, vol. 3, s. 305-322, 2009 [cit. 2022-02-25]. Dostupné z: https://polek.vse.cz/pdfs/pol/2009/03/02.pdf.

[7] Krejtz K., Çöltekin A., Duchowski A., Niedzielska A.: *Using Coefficient K to Distinguish Ambient/Focal Visual Attention During Cartographic Tasks* [online] Journal of Eye Movement Research, 10(2), s. 1-13, 2017 [cit. 2022-04-14]. Dostupné z: http://coltekin.net/arzu/publications/krejtz_etal_2017.pdf.

[8] Krispin R.: *Hands-On Time Series Analysis with R* [online] Packt, 2019 [cit. 2022-03-27]. Dostupné z: https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788629157/1/ch01lvl1sec06/getting-started-with-r.

[9] Kumar M.: *R Overview and History* [online] 2020. [cit. 2022-03-17]. Dostupné z: https://medium.com/@ArtisOne/r-overview-and-history-75ecb036d0df.

[10] Malouche D.:*How to create an R package and publish it in Github?* [online], 2020. [cit. 2022-03-28]. Dostupné z: https://www.youtube.com/watch?v=kQ5QkN4Kx4Q

[11] Marwan N., Romano M. C., Thiel M., Kurths J.: *Recurrence plots for the analysis of comlpex systems* Physics Reports 438, s. 237-329, 2007.

[12] Marwan N.: *Historical Review of Recurrence Plots* [online] The European Physical Journal Special Topics, 164(1), s. 3–12, 2008 [cit. 2022-03-03]. Dostupné z: https://www.researchgate.net/publication/225110160_A_historical_review_of_recurrence_plots

[13] Nandi K.: *Overview of R* [online] [cit, 2022-01-19]. Dostupné z: https://makemeanalyst.com/r-programming/overview-of-r/

[14] Riley M. A., Van Orden G. C.: *Tutorials in contemporary nonlinear methods for the behavioral sciences* [online] Retrieved March 1, 2005. [cit. 2022-02-20] Dostupné z: https://www.nsf.gov/sbe/bcs/pac/nmbs.jsp.

[15] Shiny:*Learn Shiny.* [online] [cit. 2022-04-22] Dostupné z: https://shiny.rstudio.com/tutorial/

[16] Soetewey A.:*How to publish a Shiny app: example with shinyapps.io* [online] 2020. [cit. 2022-04-19] Dostupné z: https://towardsdatascience.com/how-to-publish-a-shiny-app-example-with-shinyapps-io-ec6c6604d8

[17] TechVidvan: *Should you start learning R? Weigh the Pros and Cons of R programming* [online] [cit. 2022-02-24]. Dostupné z: https://techvidvan.com/tutorials/pros-and-cons-of-r/

[18] TIOBE: *TIOBE index* [online] [cit. 2022-01-06]. Dostupné z: https://www.tiobe.com/tiobe-index/

[19] TIOBE: *The R Programming Language* [online] [cit. 2022-03-26]. Dostupné z: https://www.tiobe.com/tiobe-index/r/

[20] Trestletech:*Create an R Package in RStudio* [online], 2013. Dostupné z: https://www.youtube.com/watch?v=9PyQlbAEujY

[21] Vaidyanathan P., Pelz J., Alm C., Shi P., Haake A.: *Recurrence quantification analysis reveals eye-movement behavior differences between experts and novices* [online] Conference: Proceedings of the Symposium on Eye Tracking Research and Applications, 2014 [cit. 2022-03-05]. Dostupné z: https://www.researchgate.net/publication/262154876_Recurrence_quantification_analysis_reveals_eye-movement_behavior_differences_between_experts_and_novices

[22] Webber Ch. L., Ioana C., Marwan N.: *Recurrence Plots and Their Quantifications: Expanding Horizons* [online] Springer International Publishing, Switzerland, 2016 [cit. 2022-03-10]. Dostupné z: https://books.google.cz/books?id=4fY0DAAAQBAJ&printsec=frontcover&dq=recurrence+plot+and+their+quantification&hl=cs&sa=X&ved=0ahUKE wjQsIOYhbnpAhU3UBUIHXbEAwYQ6AEIMjAB#v=onepage&q&f=false.

[23] Webber Ch. L., Marwan N.: *Recurrence Quantification Analysis: Theory and Best Practices* [online] Springer International Publishing, Switzerland, 2015 [cit. 2022-03-08]. Dostupné z: https://books.google.cz/books?id=af4sBAAAQBAJ&printsec=frontcover&dq=recurrence+plot+and+their+quantification&hl=cs&sa=X&ved=0ahUKEwjQsIOYhbnpAhU3UBUIHXbEAwYQ6AEIKDAA#v=onepage&q&f=false

[24] Wickham H., *Mastering shiny* [online] O'Reilly Media, Inc., 2021 [cit. 2022-02-06]. Dostupné z: https://mastering-shiny.org/.

[25] Wickham H., Bryan J., *R Packages: Organize, Test, Document, and Share Your Code* [online] O'Reilly Media, Inc., 2015 [cit. 2022-02-03]. Dostupné z: https://r-pkgs.org/