TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Analýza tréningových dat pomocí jazyka R

Bakalářská práce

Studijní program: B2646 – Elektrotechnika a informatika
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Vladimír Nevyhoštěný**
Vedoucí práce: doc. Ing. Otto Severýn, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Analysis of training data in language R

Bachelor thesis

Study programme: B2646 – Electrotechnology and informatics
Study branch: 1802R007 – Information technology

Author: **Vladimír Nevyhoštěný**
Supervisor: doc. Ing. Otto Severýn, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vladimír Nevyhoštěný**
Osobní číslo:	**M14000057**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Analýza tréningových dat pomocí jazyka R**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s funkčností a s API webových aplikací určených pro zpracování a analýzu dat sportovního tréningu, např. Strava či Garmin Connect.
2. Navrhněte SW řešení, které bude pomocí API získávat data z výše uvedených webových aplikací a transformovat je do formátů datových struktur jazyka R.
3. V jazyce R navrhněte, implementujte a otestujte takové metody analýzy získaných dat, které zmíněné aplikace neumožňují, ale které je vhodné provádět v rámci vyhodnocování sportovního tréningu.
4. Navržené řešení popište a zdokumentujte ve formě technické zprávy.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Adler, J.: R in a Nutshell. O'Reilly Media, Inc., 2012. ISBN 978-0596801700**

[2] **Braun, W.J., Murdoch, D.J.: A first course in statistical programming with R. Cambridge University Press, 2008 Cambridge, UK. ISBN 978-0521694247.**

[3] **Suchý, J.: Počítačová evidence, zpracování a analýza tréninkového zatížení pro potřeby řízení sportovního tréninku (na příkladu ledního hokeje a vytrvalostních vícebojů). FTVS UK, Praha. 2004.**

Vedoucí bakalářské práce: **doc. Ing. Otto Severýn, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Mgr. Petr Jeřábek, Ph.D.**
Katedra tělesné výchovy

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

Technická univerzita v Liberci
L.S. 14

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

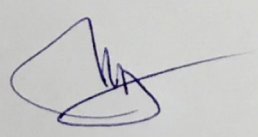
Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2017

Podpis: 

Poděkování

Děkuji vedoucímu práce doc. Ing. Ottovi Severýnovi, Ph.D. za metodické vedení a jeho cenné rady během tvorby této práce. Rovněž děkuji Mgr. Davidovi Kmochovi za pomoc při konfiguraci virtuální serveru a všem dalším, kteří přidali cenné rady k této práci.

Především bych chtěl poděkovat své rodině, která mě podporovala a dělala mi klidné zázemí ke studiu.

Abstrakt

Tato bakalářská práce se zabývá analýzou tréninkových dat pomocí jazyka R. Prvním cílem bylo seznámit se stávajícími aplikacemi, které trh nabízí a s jejich možnostmi. Dále bylo nutné navrhnout vlastní řešení způsobu získávání dat z těchto aplikací. Dalším cílem byla implementace vlastní analýz dat, které doplní chýbějící funkce stávajících aplikací. Výslednou aplikaci bylo zapotřebí zdokumentovat.

Výsledkem je webová aplikace, který je tvořen především jazykem R s balíčkem Shiny, ale i PHP. Aplikace je dostupná na internetu.

Klíčová slova

Jazyk R, Shiny, PHP, webová aplikace, analýza tréninkových dat, Strava.com, OAuth2

Abstract

This bachelor thesis is about using language R for training data analysis. The first aim was to do research on existing applications offered by the market a their potential. It was also necessary to design own solution for getting information from these application. Another aim was to implement own data analysis to complement the missing functionality of existing applications. Documentation of final application was required.

Keywords

Language R, Shiny, PHP, web application, training data analysis, Strava.com, OAuth2

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

API	Application Programming Interface
CRAN	The Comprehensive R Archive Network
CSV	Comma-separated values
FIT	Flexible and Interoperable Data Transfer
GC	Garmin Connect
GPS	Global Positioning System
GPX	GPS Exchange Format
GUI	Graphical User Interface
HR	Heart Rate
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over Transport Layer Security
JSON	JavaScript Object Notation
MATLAB	Matrix Laboratory
PDF	Portable Document Format
Permon	Performance Monitor
PHP	Hypertext Preprocessor
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCX	Training Center XML
TF	Tepová frekvence
TRIMP	Training Impulse
XML	Extensible Markup Language

Úvod

Sportovní výkony jsou neustále posouvány vpřed napříč všemi sportovními odvětvími. Jednou z hlavních rolí v tom hraje zefektivňování tréninkových metod. K tomu, zejména v posledních letech, velice napomáhá rozmach různých sportovních zařízeních, která monitorují tréninkové aktivity. Tato zařízení často disponují mnoha senzory, a tak jsou zdrojem mnoha druhů dat, avšak problém nastává s analýzou těchto dat. Mnohdy je problém z velkého množství dat, posbíraného z desítek tréninkových jednotek, získat ucelené informace, vypovídající o kondici sportovce.

Existují aplikace, které se touto problematikou zabývají. Ať už se jedná o profesionální aplikace, které jsou schopny monitorovat a plánovat tréninkové jednotky celým týmům, či aplikace nabízené běžným úživatelům neboli amatérským sportovcům při koupi sportovního zařízení. I přestože profesionální aplikace zdaleka převyšují rozsahem funkcí aplikace nabízené zdarma, tak vždy vznikne situace, kdy uživateli v aplikaci nějaká funkce chybí. Aplikace však často tvoří uzavřený systém a neumožňuje rozšíření cizími moduly. Uživateli tak často nezbývá nic jiného, než používat více aplikací zároveň a data mezi nimi synchronizovat. To může být komplikované, a také časově náročné.

Bakalářská práce se v první své části věnuje mapování dostupných možností analyzování sportovních dat. Jsou popsány základní vlastnosti a nedostatky aplikací dostupných na trhu. Zkoumány byly aplikace, které jsou dostupné zdarma. Shrnuté jsou také možnosti monitorování tréninku a zářízení, která umožňují monitorování tréninku.

Jelikož dle zadání práce mají být analýzy v rámci práce implementované v jazyce R, je tomuto jazyku věnovaná celá druhá kapitola. V kapitole jsou popsány základní principy jazyka, ale také nástoje a knihovny, které byly při použity v práci.

Cílem této práce je navrhnout a vytvořit aplikaci, která efektivně získá sportovní data a bude je analyzovat. Aplikace nemá za cíl konkurovat stávajícím řešením, které trh nabízí, ale měla by fungovat, jako jejich doplněk. Nutností je nalézt vhodný zdroj dat, který umožní celý proces pokud možno co nejvíce automatizovat. Dalším bodem je převedení do získaných dat do struktur jazyka R. Přidruženým cílem je vytvořit grafické uživatelské rozhraní, které bude multiplatformní a vyřešit přístup uživatelům k aplikaci.

I v průběhu vývoje aplikace docházelo k doplnění stávajících aplikací o funkce, které jim v době rešerše chyběly. K jedné takové změně došlo například u modelování výkonnosti aplikace Strava, což je funkce, která je podstatnou částí i výsledné aplikace. A tak i v průběhu vývoje byl na toto brán zřetel.

1 Rešerše

Pro kvalitní sportovní trénink je zcela bez pochyby nutné vést tréninkový deník. Napomáhá sportovci plánovat své budoucí tréninky tím, že mu umožní analyzovat tréninky předešlé. Deník poskytuje určitým způsobem motivaci pro další tréninky, jelikož umožňuje porovnávat výkony s minulou sezónou či minulým tréninkovým cyklem.

Přestože pro jednoduché sledování tréninků postačí záznam o délce trvání a typu tréninku, tak je v současnosti i pro rekreačního sportovce dostupné velké množství pomůcek, které mu umožní kvalitně monitorovat jeho trénink. Například dnes již poměrně finančně dostupné sporttestery[1] je vhodné použít pro sledování tepové frekvence. I mezi amaterskými sportovci se začínají více používat wattmetry pro sledování výkonu na jízdním kole. Tyto sporttestery a cyklopočítače bývají také často doplněny o GPS modul, díky kterému mohou sledovat rychlost pohybu či celkovou vzdálenost tréninkové jednotky. Vše, jako celek poskytuje už poměrně velkou škálu dat, tvořící podrobné informace o tréninkové jednotce.

1.1 Typy zařízení

Naprosto základním a běžným prostředkem, jak získat informace o tréninkové jednotce jsou stopky. Ty poskytují časový záznam, který je možné zpětně využít na analýzy typu: celkový počet odtrénovaných hodin během časového období či sledování trendu zlepšení na dané trati (úseku). Zatímco stopky se dají využít na celou řadu sportovních aktivit, existují jednoduché cyklopočítače, které krom stopek plní i funkci tachometru a měří tak i ujetou vzálenost na bicyklu. Zařízení tohoto typu mohou obsahovat i další senzory, například snímač tepové frekvence. Taková zařízení sice neumožní uživateli sledovat průběh tepové frekvence, ale umí zhodnotit maximální či průměrnou hodnotu tepové frekvence za celou tréninkovou jednotku.

Výše zmíněná zařízení z pravidla obsahují funcki mezičas (kolo). Jak plyne z jejího názvu, funkce umožňuje oddělit měřenou tréninkovou jednotku na menší úseky, a mnohdy také uložit do interní paměti. Rázem je možné získat data, která se nebudou týkat pouze tréninku jako celku, ale i jednotlivých úsecích zvolených uživatelem.

Poslední skupinou zařízení je skupina, do které spadají různé sporttestery, jež vytváří časové řady, často s GPS modulem (ale pravidlem to není). Nabízí data z mnoha senzorů. Mezi nejběžnější senzory současnosti patří: **tepová frekvence**,

[1] Počítač, nejčastěji v podobě hodinek, který umožňuje sledovat tepovou frekvenci.

GPS záznam či **hodnoty z barometru**. Z těchto dat je možné poměrně důkladně analyzovat tréninkovou jednotku.

Obrázek 1.1: Sporttester Garmin Fenix s GPS modulem. Velmi často mají sporttestery podobu náramkových hodinek.

1.1.1 Rozdělení sportovních zařízení

Sportovní zařízení na monitorování tréninků by se dala rozčlenit do těchto dvou skupin:

- Zařízení, která neumožňují vytvářet časové řady a disponují pouze skalárními naměřenými daty.
- Zařízení, která umí vytvářet datové řady. Velmi často tato zařízení disponují mnoha senzory, a tak dostáváme časové řady několika veličin (typicky: tepová frekvence, rychlost, výškové metry, aj.).

1.2 Typy dat

Tak, jak jsou výše popsaná zařízení rozřazená do pomyslných skupin, je možné rozdělit i data o trénincích do podobných skupin.

1.2.1 Skalární

Skupina dat, která charakterizují trénink jako celek. Nejčastěji jsou to tyto bývají veličiny: **celkový čas tréninku**, ušlá či ujetá **vzdálenost**, **průměrná tepová frekvence** a další.

Tato data velmi často slouží k k vytváření různých statistických přehledů za uplynulé období. Sportovci dávají důležitou informaci o objemu tréninků během určitého časového úseku, například měsíčního cyklu. Naproti tomu není ale dost dobře možné analyzovat konkrétní části tréninku. Není také možné získat informaci o tom, jakých hodnot dosáhl sportovec na vybraném časovém úseku.

Pokud se skalární data nevztahují pouze pro celou tréninkovou jednotku, ale i pro jednotlivé úseky tréninku, je možné analyzovat lépe jednotlivé úseky tréninku. Pro ilustraci podoby těchto dat může posloužit tabulka č. 1.1. Takový přehled daleko více vypovídá o náplni tréninku.

Úsek	Čas	Prům. TF	Max. TF
1	1:23	155	165
2	1:20	159	168
3	1:19	163	171

Tabulka 1.1: Názorná úkazka dat pro jednotlivé úseky vybrané z celé tréninkové jednotky.

1.2.2 Časové řady

Nejcennější data pro analýzu tréninku jsou data, která bychom mohli označit jako vektorová, neboli data, která mají podobu časové řady. Díky těmto časovým řadám je možné, komplexně analyzovat tréninkovou jednotku. Příkladem časové řady může být záznam cyklistického tréninku v aplikaci Strava (obrázek 1.2).

Data bývají zaznamenána většinou s přesným intervalem jedné sekundy, ale to se může lišit s každým zařízením či nastavením.

1.2.3 Nejčastěji používané veličiny

Měřené veličiny se mohou pro každý sport trochu měnit, nicméně je možné shrnout ty veličiny, které se nejčastěji měří a umí je měřit běžný sporttester. Mezi ně patří:

- Rychlost pohybu
- Tepová frekvence
- Vzdálenost
- Nadmořská výška

Některé sporttestery sledují například i veličiny jako je teplota prostředí či počet uplavaných délek bazénu v módu plavání. Zatím pouze na jízdním kole je možné smysluplně monitorovat výkon, který nejlépe vypovídá o výkonu sportovce.

1.2.4 Záznamy tréninkových jednotek

Běžně má záznam z tréninkové jednotky, při použití zařízení, které vytváří časové řady, podobu souboru. Nejčastejšími typy souborů jsou GPX [9] či FIT [7]. Tyto dva typy souborů jsou jakýmsi standardem v uchovávání záznamů ze sporttesterů a dalších sportovních zařízeních.

Soubor typu FIT je proprietární a používán výhradně zařízeními značky Garmin. GPX má otevřený formát a vychází z XML struktury. Alternativou k GPX je

TCX soubor. TCX narozdíl od GPX lépe zachází s dalšími veličinami, jako tepová frekvence, kadence aj.

```
<trkpt lat="50.7952870" lon="15.1431300">
 <ele>730.0</ele>
 <time>2017-01-08T07:29:54Z</time>
 <extensions>
  <gpxtpx:TrackPointExtension>
   <gpxtpx:atemp>19</gpxtpx:atemp>
   <gpxtpx:hr>101</gpxtpx:hr>
   <gpxtpx:cad>0</gpxtpx:cad>
  </gpxtpx:TrackPointExtension>
 </extensions>
</trkpt>
```

Ukázka kódu 1.1: Ukázka jednoho časového bodu vybraného z GPX souboru aktivity obsahující základní veličiny: pozici, nadmořskou výšku, čas, tepovou frekvenci. Soubor GPX má podobu schéma XML.

1.3 Zpracování dat

1.3.1 Ruční

Nejstarším způsobem, jak evidovat tréninkové jednotky je zapisováním si jich ručně. Zapisování ručně umožňuje si podobu deníku velmi přizpůsobit. Pokud je využit k záznamu tabulkový procesor, tak je možné na data vytvářet i základní pohledy a statistiky. Zásadním problémem tohoto způsobu vedení deníku je, že není nijak automatizován. Stejně tomu je i s importem nových dat, který rovněž není automatizovaný. Také pro některé komplikovanější analýzy tréninku může být tabulkový procesor poněkud nedostačující.

1.3.2 Webová aplikace Strava

Webová aplikace [13], která je určená zejména pro cyklisty a běžce. Nenabízí zařízení, kterým je možné sledovat tréninkové aktivity, ale spolupracuje s většinou výrobců sporttesterů a nabízí import dat z jejich zařízení. Dále také nabízí aplikaci pro mobilní telefony, která umožňuje zaznamenávat trasu, rychlost, ale i jiné údaje vztahující se k aktivitě. Další možností, jak přidat nové aktivity do služby je nahrát je ze souborů či je zadat ručně.

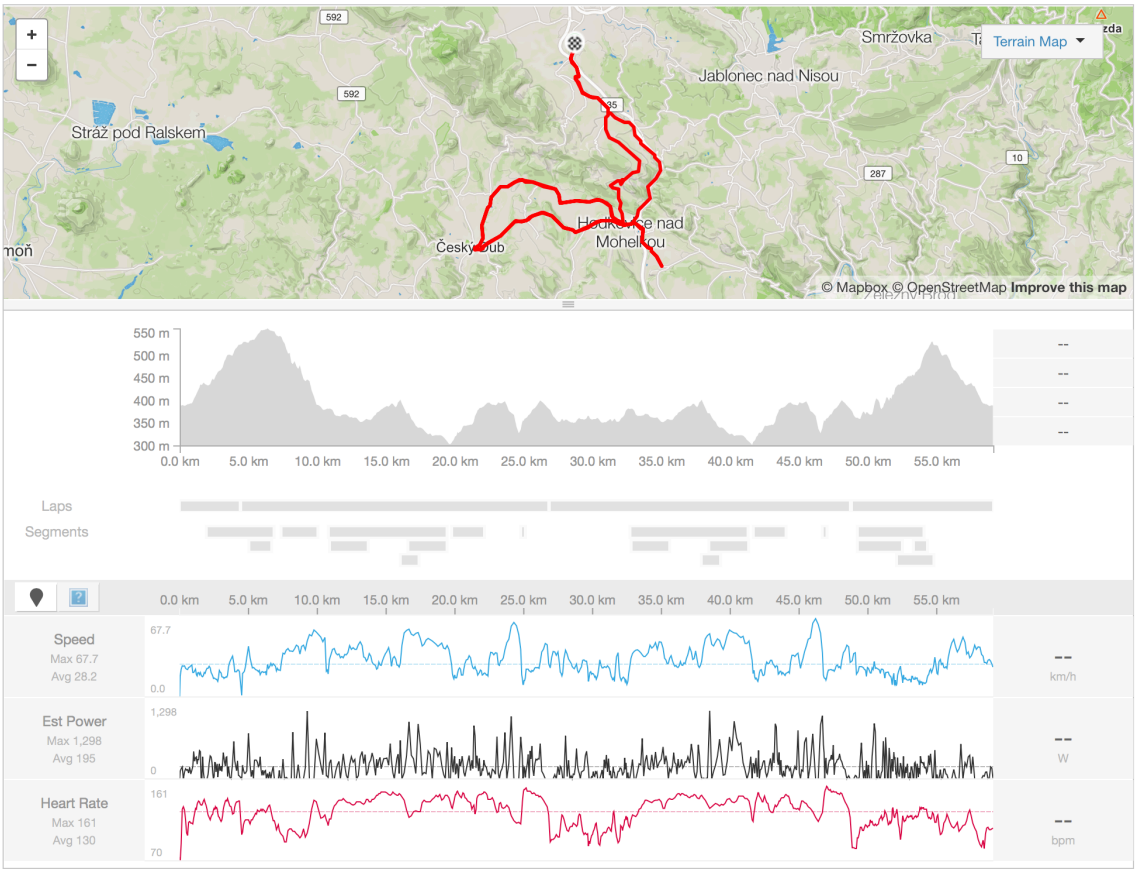
Strava nabízí skrz své webové rozhraní mnoho zajímavých informací o tréninku. Velkou předností Stravy je také možnost vytvářet na naměřených trasách segmenty, na kterých lze analyzovat svůj výkon a porovnat s ostatními sportovci. Aplikace tak působí do jisté míry jako sociální síť.

Výhody webové aplikace

- Aplikace nabízí poměrně detailní analýzu tréninkové jednotky. Nabízí porovnání výkonů proti minulým pokusům na celou trať či pouze vybrané úseky (Lap, segmenty).
- Aplikace poskytuje poměrně užitečné dodatečně vypočtené údaje, jako například GAP[2].

Nevýhody aplikace

- Aplikace sice částečně podporuje náhled dalších osob do tréninkového deníku sportovce. Chybí ovšem možnost exportování reportů vybraného sportovce například ve formátu tabulkového procesoru.
- V aplikaci chybí možnost kategorizace tréninků a následná analýza kategorie jako celku.



Obrázek 1.2: Webová aplikace Strava a její detail na tréninkovou jednotku. [13]

[2]GAP - (Grade Adjusted Pace) je odhad tempa na rovinnaté trati. [3]

1.3.3 Garmin Connect

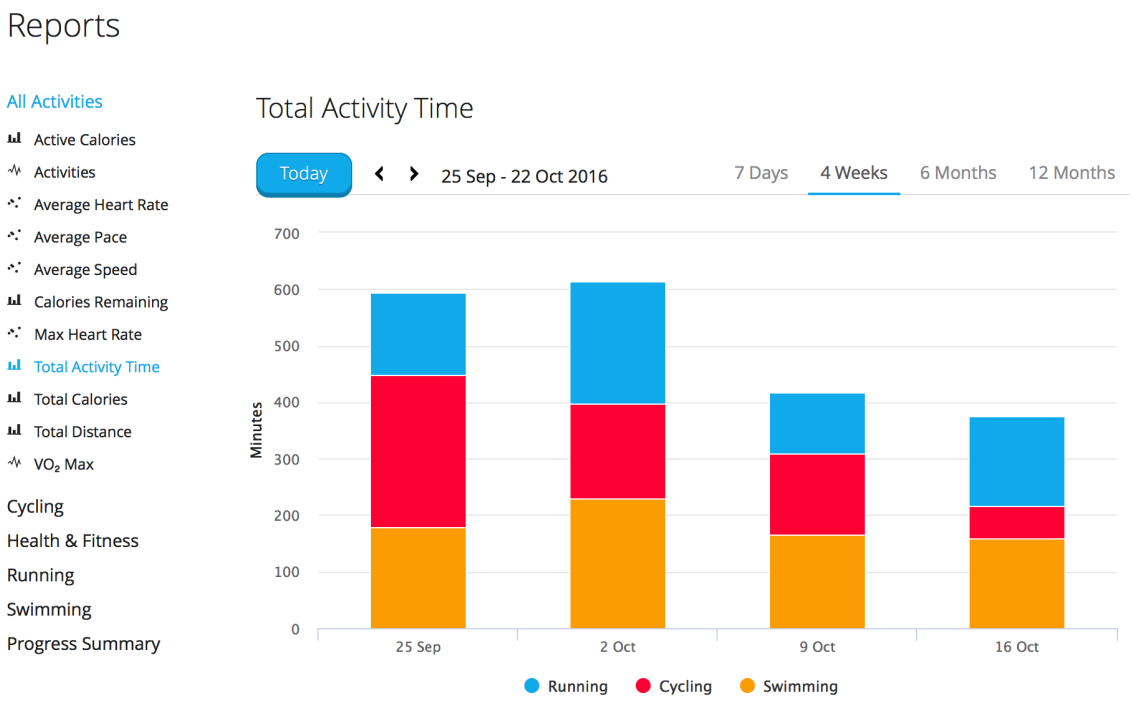
Další velmi rozšířenou službou je Garmin Connect [8] (dále jen GC). GC je rozšířený hlavně díky sportovním přístrojům Garmin. Velkou předností služby je možnost zaznamenávat velké množství typů dat. GC je jako sportovní aplikace poměrně rozsáhlá. Slouží jako sporttracker, do kterého je možné zaznamenávat hodnoty od ušlých kroků, až po přijmuté kalorie. Aplikace umožňuje dělat základní reporty nad filtrovanými aktivitami, a tak celkem obstojně sloužit jako tréninkový deník.

Výhody aplikace

- GC podporuje velké množství měřených veličin i ty méně rozšířené (vertikální oscilace při běhu apod.).
- Aplikace dobře spolupracuje se zařízeními Garmin. Skrze aplikace je možné vytvářet tréninkové plány či trasy a nahrát je do zařízení.

Nevýhody aplikace

- Aplikace nabízí poměrně detailní reporty tréninkových jednotek v podobě grafů. Chybí ovšem možnost konfigurace těchto reportů. Uživatel si sice může vybrat časové období, ze kterého má být tvořen report, ale to je vše, jak uživatel může podobu reportu ovlivnit. Chybí také možnost export těchto reportů například do PDF formátu. Jediná možnost je exportovat data pro graf ve formátu CSV, nikoliv graf samotný.
- Podobně jako u Strava chybí možnost kategorizace tréninků.



Obrázek 1.3: Webová aplikace GC a její funkce reportu aktivit vybraného časového výseku. [8]

1.3.4 Další webové aplikace

Garmin není jediný výrobce na trhu se zařízeními pro záznam sportovních aktivit. Mezi významné výrobce patří i Polar, Suunto či TomTom. I tito výrobci nabízí webové aplikace pro práci s naměřenými daty. Přestože tyto aplikace často nabízí funkce velmi podobné těm, které nabízí Strava či GC, mají alespoň v ČR menší popularitu. Výše zmíněné aplikace mají menší komunitu a absenci profesionálních sportovců.

- **Suunto** nabízí aplikaci **Movescount** [14], která slouží zároveň jako sportovní sociální síť, ale také i jako tréninkový deník.
- **Polar** nabízí dvě aplikace. Jednou je **Polarpersonaltrainer** [11], která více zaměřená na evidenci a analýzu tréninku. Druhou je **Polar Flow** [10], která funguje víc jako sociální síť v podobě sporttrackeru.
- **TomTom** nabízí aplikaci **TomTom MySports** [15], která funguje jako tréninkový deník.

1.3.5 Profesionální

Existují také služby, které svými funkcemi převyšují potřeby a většinou i možnosti běžného ameterského sportovce a jsou tak určeny profesionálům. Tyto služby umí

často vytvářet tréninkové skupiny a spojovat tak sportovce a trénéry. Mnohdy moniturují kromě tréninkových aktivit i zdravotní stav sportovce, úrazy či lékařské zprávy. Tyto služby jsou zejména díky jejich vysoké ceně určeny pro velké sportovní organizace a týmy. To je také důvod, proč jsou málo rozšířené mezi sportovci amatéry. Za profesionální službu je možné považovat **TrainingPeaks** [16] či **AthleteMonitoring** [6].

1.4 Analýza dat

1.4.1 Tréninkový impulz

Existuje mnoho způsobů jak měřit dopad trénikového usilí na sportovce. Toto usilí je často nazýváno tréninkový impulz, neboli TRIMP [18] (z anglického **Tr**aining **Imp**ulse). Tréninkový impulz může být definován časem či vzdáleností, ale lepší je využít tepové frekvence (v případě cyklistických tréninků ideálně výkon).

Hodnota TRIMP je použita při modelování výkonnosti pomocí Banistrova modelu (viz. 1.4.2) a jiných modelů. Výpočtem TRIMP získáme bezrozměrnou hodnotu, která vypovídá o úsilí vynaloženém při tréninku a je vstupem do Banistrova modelu.

Exponenciální měření tepové frekvence

Poměrně přesnou a sofistikovanou metodou, jak určit velikost tréninkového impulzu pomocí tepové frekvence je Exponenciální měření tepové frekvence (podle anglického Exponential Heart Rate Scaling). Pro určení velikosti impulzu je zapotřebí zjistit pro každou hodnotu tepové frekvence dosažené při tréninku, jak dlouho se na této hodnotě sportovec pohyboval. Z názvu metody je patrné, že s rostoucí tepovou frekvencí velikost tréninkového impulzu exponenciálně narůstá.

Výpočet intenzity tréninkového impulzu metodou exponenciálním měřením tepové frekvence je definován vzorcem 1.1.

$$TRIMP^{exp} = \sum_{i} (T_i \cdot HR_{ri} \cdot ce^{\delta \cdot HR_r}) \tag{1.1}$$

Kde:

- $\mathbf{T_i}$ je čas strávený na dané tepové frekvenci v minutách. Vzorové hodnoty jsou v tabulce 1.2.
- $\mathbf{HR_{ri}}$ je tepové frekvence vážená maximální a klidovou tepovou frekvencí sportovce (vzorec 1.2).
- δ a **c** jsou konstanty určující závislost mezi tepovou frekvencí a hladinou laktátu. Pro muže byla tato konstanta δ empiricky určena na hodnotu **1,92**, pro ženy **1,67**. Konstanta **c** pak pro muže na hodnotu **0,64** a ženy **0,86** [18].

Pro výpočet je nutné převést naměřenou tepovou frekvenci na HR_r (Heart Rate Reserve), kdy je tepová frekvence vážená v závislosti na klidové a maximální tepové

Tabulka 1.2: Příklad vstupních hodnot pro výpočet Exponenciální měření tepové frekvence.

i	$\mathbf{HR_{ri}}$ **[-]**	$\mathbf{T_i}$ **[min]**
1	128	0,23
	...	
23	151	1,34
24	152	1,45
25	153	1,53
	...	
40	168	0,12

frekvenci. Pro přesný výpočet je vhodné, aby hodnoty maximální a klidové frekvence byly naměřeny například při sportovním zátěžovém testu.

$$HR_r = \frac{HR - HR_{min}}{HR_{max} - HR_{min}} \tag{1.2}$$

Kde:

- $\mathbf{HR_r}$ je tepová frekvence
- $\mathbf{HR_{min}}$ je minimální tepová frekvence
- $\mathbf{HR_{max}}$ je maximální tepová frekvence

1.4.2 Sledování výkonnosti

Nejjednoduší metodou, jak sledovat výkonnost sportovce je pozorování jeho výsledků, respektive časů na měřených úsecích či tratích. Podmínky nejsou vždy stejné a časté testování sportovce může mít negativní dopad na jeho tréninkový plán. Existují modely, které se dají použít na sledování výkonnosti sportovce a také trend výkonnosti. Tyto modely napomáhají v ladění formy na klíčovou soutěž či odhalovat tréninkovou monotónost. Tedy stav, kdy se tréninkové impulzy sportovce neustále opakují a tréninky se stávají neefektivními.

Banistrův model

Jedním z těchto modelů je Banistrův model [18]. Model je vhodný pro všechny typy sportů, u kterých je možné určit intenzitu tréninkového impulzu. Model má dvě složky. Složku trénovanosti a složku únavy. Na základě těchto dvou složek je určena výkonnost sportovce. Model je počítán pro jednotlivé dny. Model je definován následujícím vzorcem:

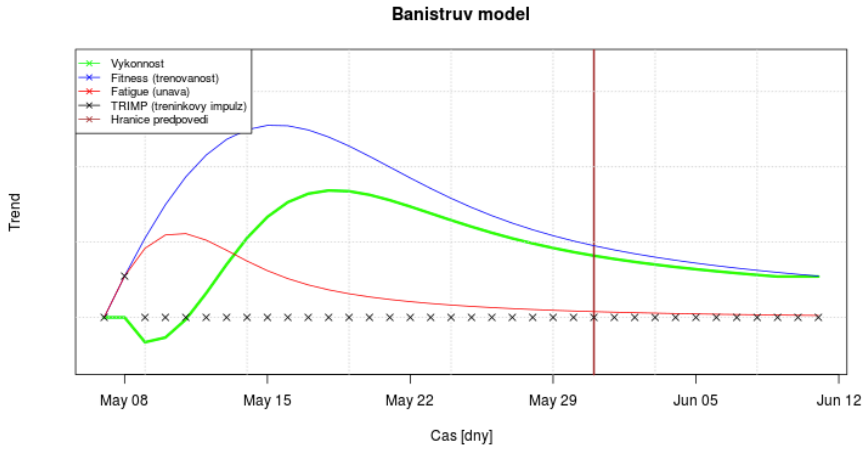
$$p_t = p_0 + k_1 \cdot \sum_{i=1}^{t-1} w_i \cdot e^{\frac{-(t-i)}{\tau_1}} - k_2 \cdot \sum_{i=1}^{t-1} w_i \cdot e^{\frac{-(t-i)}{\tau_2}} \tag{1.3}$$

Kde:

- $\mathbf{t}$ je index časové řady pro jednotlivé dny.
- $\mathbf{p_0}$ je počáteční hodnota výkonnosti. Spolu s dalšími je to vstupní parametr modelu, který si uživatel může zvolit.
- $\mathbf{k_1}$ je konstanta trénovanosti
- $\mathbf{w_i}$ je časová řada, kde je pro každý den vypočtena velikost tréninkového impulzu. V případě vícefázového tréninku jsou hodnoty všech fází sečteny pro celý den. Často je velikost určena metodou Exponenciální měření tepové frekvence, která je zmíněná výše.
- τ_1 je doba navrácení složky trénovanosti do původních hodnot ve dnech
- $\mathbf{k_2}$ je konstanta únavy
- τ_2 je doba navrácení složky únavy do původních hodnot ve dnech

Konstanty, které definují model, ho umožňují co nejvíce přizpůsobit sportovci. Například pokud je $\mathbf{k_1}$ větší než $\mathbf{k_2}$, znamená to delší dobu zotavení z tréninkové jednotky a naopak.

Složky Banistrova modelu (únava, trénovanost a výkonnost) jsou bezrozměrné a nemají tedy žádné jednotky. Hodnota složek představuje pomyslnou úroveň pro každý den. V případě složky výkonnosti je možné tvrdit, že čím vyšší hodnoty dosahuje, tak tím je vyšší skutečná výkonnost sportovce. Jednotlivé hodnoty nicméně nejsou až tak důležité, jako spíše trend jednotlivých složek modelu. Například sledování složky únavy může napomoct sportovci se vhodně připravovat v době bezprostředně před jeho soutěží.



Obrázek 1.4: Reakce Banistrova modelu na jeden tréninkový impulz. Vypočteno pro třicet dní.

2 O jazyku R

Programovací jazyk R můžeme považovat za volně dostupnou odnož komerčního jazyka S. Jazyk S, a stejně tak jazyk R, je určený pro statistické výpočty a data mining.

Jazyk S byl vyvinut Johnem Chembersem v Bell laboratories. Představení první verze jazyku S proběhlo v roce 1976. Zpočátku to byl jazyk bez názvu, ovšem brzy dostal jedno písmenný název S po vzoru programovacího jazyku C. Písmeno S vycházelo z častého použití tohoto písmena ve statistických výpočtech. V roce 1979 byl vytvořen port pro UNIX, který se brzy stal hlavní platformou S. Jazyk R byl vytvořen Rossem Ihakem a Robertem Gentlemanem na Aucklandské univerzitě v roce 1993 jako GNU projekt. Nyní, na vývoji pracuje tým R Development Core Team. Jméno R vychází částečně z prvních jmen autorů, a také souvisí s názvem jazyka S.

Publikací k jazyku R není tolik, zvláště v českém jazyce, jako u jiných a rozšířenějších programovacích jazyků. Velmi užitečná, avšak v anglickém jazyce, je samotná oficiální webová prezentace projektu R[1], kde se nachází instalace R a dokumentace k jazyku.

Pro programování R lze použít i obecného textového editoru, avšak existují různá vývojová prostředí s GUI, např.: R Commander, JGR či RKWard.

Hojně používaným prostředkem pro vývoj R je RStudio [4], které je dostupné zdarma ve verzi OpenSourceEdition. RStudio má funkce známe z ostatních vývojových prostředí, jako třeba integrovanou dokumentaci nebo kontrolu syntaxe. Pro řešení mé práce bylo využito právě zmiňované RStudio.

2.1 Charakteristika R

Převážná část prostředí R a jeho knihoven je napsána v jazyce C, zejména pak části, které jsou výpočetně náročné. Některé funkce jsou napsány samotným R. Toto řešení umožňuje připojit kód v C či C++ v reálném čase přímo v R.

Syntaxe R se nevymyká zaběhnutým zvyklostem. Snad jen přiřazení. Jelikož R podporuje levé a pravé přiřazení, je operátorem přiřazení dvojice znaků `<-` (`->`), i přesto R podporuje `=`. Pro vytvoření globální proměnné je zapotřebí použít operátor `<<-`.

```
x <- 4
4 -> x # je totéž
```

2.1.1 Operátory

R obsahuje standartní operátory, tedy aritmetické `(+ - * \ ^)`, logické operátory `(&& ||)` a porovnávací operátory `(== != > < >= =<)`.

```
?print # zobrazí nápovědu pro funkci print
help("print") # udělá totéž
```

2.1.2 Podmínky

Jsou v R, jako v jiných programovacích jazycích. Podmínka if-else má syntaxi `if(podmínka) { v případě true } else { v případě false }`, nebo zkráceně `ifelse(podmínka, příkaz v true, příkaz v false)`. Datové struktury a typy, které jsou použity v následujících příkladech jsou popsány v kapitole 2.3.

```
x <- 5:15 # inicializace vektoru od 5 do 15
ifelse(x > 10, x, 0) # podmínka, která zohlední každý prvek vektoru
[1] 0 0 0 0 0 0 11 12 13 14 15
ifelse(x[6] / 2 == 5, "true", "false") # prvky vektoru jsou přístupné
pod indexy v hranatých závorkách
[1] "true")
```

2.1.3 Smyčky

Smyčky má R mnohé. Včetně známých smyček `while(podmínka) příkaz` a `for(sekvence) příkaz`. Součástí R je však skupina smyček Apply, jenž se používá na uplatnění funkcí na prvky matic, dvojrozměrných polí apod. Do této skupiny patřejí smyčky `apply, by, eapply, lapply, mapply, rapply, tapply` . Smyčky apply se od sebe liší převážně tím, na jaké datové kontejnery se používají. Funkce, které se mají aplikovat smyčkou mohou být různé, například matematické abs (absolutní hodnota), log2 (logiritmus o základu dva) či max a min. V některých případech při práci s daty v maticích a jiných datových kontejnerech je výhodnejší použít vhodné funkce. Například funkce `rowMeans()`, s parametrem maticí, dokáže vypočítat aritmetický průměr řádků až stokrát ryhleji, než-li smyčka apply, nemluvě o řešení for smyčkou.

```
x = matrix(c(-5:4, -2:7), nrow = 10, ncol = 2) # vytvori matici 10
x 2 naplnenou posloupnostmi -5 až 4 a -2 až 7
apply(x, 2, mean) # vypočet průměr sloupců
[1] -0.5 2.5
# pro řádky musí mít druhý parametr funkce apply hodnotu 1 (sloupce
hodnotu 2), pro oboje 1:2
```

2.1.4 Funkce

Funkce se definují slovem `function`, spolu s argumenty funkce a tělem funkce. Návratový datový typ se neuvádí. Hodnota, kterou má funkce vrátit se vkládá jako argument do příkazu `return()`. Pro pozdější volání funkce je nutné definovat též jméno funkce, kterým může být v podstatě cokoliv, až na jména stávajících funkcí. Proměnné vytvořené uvnitř funkce existují pouze po dobu použití funkce a nejsou dále dostupné mimo funkci. Z funkcí je možné přistupovat ke globálním proměnným přes operátor `<<-`.

```
square <- function(x) { # definice funkce square
sq <- x^2
return(sq)
}
square(5)
[1] 25
```

Jazyk R se dá považovat za programovací jazyk, který podporuje více než jedno programovací paradigma. Jazyk mísí prvky objektového programování, ale ze značné části se chová jako jazyk funkcionální.

Objektové programování v R připomínají mnohé generické funkce. Je tomu například u funkce `print()`, která umí vypsat do konzole snad všechny druhy datových struktur R, stačí je pouze předat jako parametr funkce. Oproti tomu, struktury v R, nemají vlastnosti a metody, které se volají složením jmen struktury a metody. Pro získání délky seznamu X je nutné volat funkci `length(X)`, nikoliv `X.length()`. Programy v R nejčastěji vznikají tak, že hlavní funkce je tvořena více menšími funkcemi.

Jazyk R je možné používat jako programovací jazyk interaktivní, ale i neinteraktivní. Pracovat s R je možné interaktivně přes konzoli. Je možné, vypsat na konzoli nápovědu k balíčku či funkci, stejně tak jako volat jiné funkce. Uživatel, ale také může vytvářet vlastní R skripty či balíčky, které pak pouze spouští a používá R kód neinteraktivně. Skripty je možné i debugovat a pracovat s nimy podobně, jako se soubory jiných programovacích jazyků.

2.2 Zabudované funkce

R obsahuje velké množství vestavěných funkcí, které jsou velmi užitečné a snad i nezbytné pro práci s R. Tyto funkce napomáhají k různým výpočtům nad daty či vizualizaci dat. Zde je malý výčet těchto funkcí.

2.2.1 Obecné funkce

- `help()` – zobrazí stránku s nápovědou pro funkci zadanou parametrem. Nápovědu je možné i získat vsazením `?` před funkci.
- `library()` – načte daný balíček, což je nutné pro použití balíčku.

- `c()` – spojí zadané atributy do sekvence (vektoru).
- `length()` – vrací délku (velikost) objektu.
- `range()` – najde maximum a minimum pro numerické hodnoty.
- `rep()` – vytvoří sekvenci zadaného prvku o zadané délce.
- `sort()` – třídí vektor nebo faktor sestupně nebo vzestupně. Umožňuje také vybrat algoritmus, jaký bude použit na třídění.
- `unique()` – odstraní duplikáty vektoru.
- `round(), signif()` – zaokrouhlení čísla.
- `trunc()` – z desetinného čísla vrátí integer.
- `setwd()` – nastaví pracovní adresář. To je velmi užitečné při práci se soubory - není třeba uvádět absolutní cestu k souborům.

2.2.2 Vykreslovací funkce

Tvoří nedílnou součást jazyka R. Díky nim je možné vizualizovat data. Jsou to funkce následující:

- `plot()` – vykresluje data do grafů. Je to naprosto základní a velmi důležitá funkce pro vizualizaci dat. Umožňuje vykreslovat do X-Y grafů. Díky velkému množství atributů je možné kompletně modifikovat výsledný graf.
- `par()` – seskupuje více grafů do jednoho pohledu.
- `curve()` – vykresluje křivku zadanou vzorcem.
- `points()` – přidá sadu bodů do existujícího grafu.
- `lines()` – přidá sadu bodů, znázorněných jako přímka, do existujícího grafu.
- `hist()` – vykreslí histogram.
- `png(), pdf(), jpeg()` – vykreslují graf do specifických obrazových formátů.

2.2.3 Statistické funkce

R jich obsahuje opravdu velmi mnoho. Všechny funkce balíku `stats` jsou popsány příkazem `help(package = "stats")`. Mezi ty základní patří:

- `lm()` – vytvoří lineární model. Velmi stěžejní funkce co se do statistických výpočtů v R týče. Pojme mnoho patametrů, které definují lineární model.
- `glm()` – vytváří zobecněný lineární model.
- `mean()` – poskytuje aritmetický průměr.
- `summary()` – vypíše přehled o vstupních (max, min, arit. průměr, meadián).

2.3 Datové struktury

R zahrnuje pět základních datových typů.

Character je datový typ, který nese jeden či více znaků. Je to obdoba řetězce. Inicializace se provádí uvozením řetězce do dvojtých uvozovek.

Numeric je číselný datový typ, jak je z názvu patrné. Uchovává jak celá čísla, tak s plovoucí desetinnou čárkou.

Integer je oproti numeric striktně celočíselný. Při jeho inicializaci je nutné, buď za číslo vložit L `x <- 2L`, nebo použít funkci `as.integer()`.

Logical , neboli boolean datový typ zprostředkovává logické hodnoty TRUE nebo FALSE. Pro přímé přiřazení je nutné zadat hodnoty velkými písmeny.

Complex je datový typ přímo určený pro komplexní čísla. Přiřazení se provádí zcela intuitivně `x <- 2 + 5i`.

R má také velkou škálu datových struktur.

2.3.1 Vector

Asi nejběžnější datová struktura jazyka R. Je možné `vector` přirovnat k polím jiných programovacích jazyků. Prvky vectoru mohou být datové typy character, logical, integer, numeric a complex. Způsobů jak inicializovat vector je několik. Je to možné například funkcí `vector()`, nebo přímo funkcí se specifickým datovým typem, např.: `numeric()` `character()`. Důležité je, že vector i ostatní datový struktury jsou indexované v R s počátkem v 1 stejně, jako třeba v MATLABu.

```
x <- vector(mode = "logical", length = 5)
x
[1] FALSE FALSE FALSE FALSE FALSE
x[3] <- TRUE
x
[1] FALSE FALSE TRUE FALSE FALSE
x[1:4] # Vybrání elementů 1 až 4
[1] FALSE FALSE TRUE FALSE
numeric(5)
[1] 0 0 0 0 0
x <- c(x, TRUE) # Přidání elementu na konec vektoru
```

2.3.2 Matrix

Jinými slovy také matice. Jedná se ve své podstatě o vícerozměrný vektor (pole). Všechny elementy struktury matrix musí být stejného datového typu. S typem matrix se manipuluje velmi podobně jako s typem vector.

```
x <- matrix(1:10, nrow = 2, ncol = 5)
#Vytvoření matice o 2 řádcích a 5 sloupcích a zaplnění sekvencí 1
až 10
x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8    10
x[1, 2:4] # Vypsání prvního řádku a sloupců 2 až 4
[1] 3 5 7
```

2.3.3 List

Narozdíl od vector-u umožňuje, aby každý prvek seznamu byl jiného datového typu. Seznam v R není tudíž homogenní. Prvkem seznamu může být dokonce i jiný seznam. Jedná se tedy o speciální typ vektoru. Právě možnost uchovat různé typy prvků a následně je získat všechny stejným způsobem, činí seznam velmi praktickou datovou strukturou. Přístup k prvkům je lehce rozdílný, než je tomu u vektoru. Pro přístup k prvkům seznamu je nutné přistupovat pomocí dvojité hranaté závorky. (`list[[index]]`).

```
x <- as.list(1:10) # List s posloupností integerů
typeof(x)
[1] "list"
typeof(x[1])
[1] "list"
typeof(x[[1]])
[1] "integer"
```

2.3.4 Factor

Datová struktura, která umožňuje uchovávat data a rozřazovat je do kategorií. Faktor je vhodné použít na data, která nabývají specifickou skupinou hodnot. Příkladem mohou být dny v týdnu. Dny v týdnu tvoří skupinu hodnot, které se opakují. Naproti tomu datum, není vhodné uchovávat ve faktoru. Výhodou faktoru je, že paměťově velmi úsporný, protože data jsou rozdělena do kategorií a hodnoty jsou, pak dále uloženy jako index kategorie. I přes svoji podobnost, faktor není totéž co vektor. Pokud faktor obsahuje čistě numerická data, tak ve výsledku se k nim nechová jako k numerickým datům a nelze například na faktor aplikovat funkci pro výpočet aritmetického průměru `mean`. Výčet všech kategorií, tudíž seznam všech použitých hodnot, je obsažen v `Levels`.

```
x <- factor(c(1, 1, 1, 0, 0, 1), labels = c("FALSE", "TRUE"))
# Jako vstupní data je posloupnost 1 a 0
# Parametr labels určuje pojmenování nominálních hodnot - faktor
tyto hodnoty vnitřně seřadí, proto nejdříve FALSE (0), TRUE (1)
x
[1] TRUE TRUE TRUE FALSE FALSE TRUE
Levels: FALSE TRUE
summary(x) # Volání přehledu nad proměnnou x (factorem)
FALSE TRUE
    2    4
```

2.3.5 Data frame

Jedná se o jednu z nejdůležitějších datových struktur R. Představuje tabulku hodnot. Hodnoty mohou nabývat různých datových typů. Data se vně dají upravovat, například pojmenováním sloupcu (`colnames()`) apod. Data frame je často produktem funkcí typu `read.csv()` či `read.table()`. Může být ovšem zkonstruován implicitně funkcí `data.frame()`. K hodnotám datovému rámci se dá přistupovat přes funkce (např.: `head()` - nabídne prvních šest řádků), nebo přes indexování. Indexování se provádí přes hranaté závorky, kam se zadává klíč řádku a klíč sloupce (`frame[radek, sloupec]`). Indexování datového rámce lze rozšířit o různá rozmezí apod., například výběr třetího řádku a prvního a třetího slouce `frame[3,c(1,3)]`, nebo výběr třetího až šestého řádku a všech sloupců `x[3:6,]`. Přístup ke sloupcům je také možný pomocí znaku `$` a jména sloupce (`x$jmenoSloupce`). Velmi zajímavý pohled na data frame nabízí funkce `summary()`, která vypočítá hodnoty jako aritmetický průměr, medián, minimální a maximální hodnotu pro každý sloupec. Na první pohled se může `data frame` jevit stejný jako `matrix`, ale není tomu tak. Hlavním rozdílem je, že matice musí mít všechny elementy stejného datového typu, což v případě `data frame` není nutné a datové typy elementů je možné kombinovat.

```
x <- data.frame(id = 1:10, letter = letters[11:20], x = runif(10))
# vytvoření data framu s id čísla 1 až 10, sloupce letter písmen
abecedy 21 až 30 a sloupce x o deseti náhodných číslech
summary(x)
       id    letter        x
Min.   : 1.00   k     :1   Min.   :0.02418
1st Qu.: 3.25   l     :1   1st Qu.:0.14467
Median : 5.50   m     :1   Median :0.27772
Mean   : 5.50   n     :1   Mean   :0.38080
3rd Qu.: 7.75   o     :1   3rd Qu.:0.49161
Max.   :10.00   p     :1   Max.   :0.99680
```

2.4 R balíčky

Balíčky jsou nedílnou součástí jazyka R. Balíček je ve své podstatě soubor funkcí v R. Tyto funkce jsou doplňkem k implicitním funkcím, zabudovaných v R.

Mnoho balíčku, které jsou svým použitím běžné, jsou přibaleny v základní instalaci R. Takový balíček stačí pouze načíst pomocí funkce `library()` s jménem balíčku jako parametr funkce. Může se stát, že je potřeba pracovat s balíčkem, který není stažen do knihovny, aneb složky, kde jsou všechny balíčky lokálně uloženy. V tom případě je nutné použít funkci `install.packages()`, která stáhne balíček z repozitáře CRAN a umístí jí do knihovny balíčků. Pokud je balíček, který má být naistalován funckí `install.packages()`, již stažen do knihovny, dojde k jeho případné aktualizaci nejnovější verzí z CRAN. Je velmi časté, že jednotlivé balíčky spolu spolupracují a vyžadují pro práci s nimi i načtení všech závislostí. V případě, že všechny závislé balíčky jsou staženy do knihovny, tak funkce `library()` načte i tyto balíčky. Může se stát, že požadovaný balíček není stažen a v takovém případě je nutné, nejprve balíček stáhnout. Nutno podotknout, že balíčky jsou vývojáři velmi často aktualizované, je tedy nutné mít pro použití těchto balíčku stále aktuální verzi R. Verzi R je velmi jednoduché zjistit pomocí příkazu `version`. V neposlední řadě je také velmi užitečná funkce `search()`, která vypíše všechny načtené balíčky.

Velmi nápomocné jsou nástroje pro práci s balíčky, které poskytuje, již zmíněné, vývojové prostředí R-Studio. Pomocí R-Studia je možné vyhledávat balíčky na CRAN, a pak je jednoduše instalovat pomocí grafického rozhraní. R-Studio umí i například najít všechny balíčky, které je možné aktualizovat a aktualizovat je hromadně.

2.5 Shiny

Shiny [17] je balíček R od tvůrce R-Studia. Slouží k vývoji webových aplikací v jazyce R. Tyto aplikace se nazývají **Shiny apps**. Vývoj těchto aplikací se provádí pouze v jazyce R a není tak nutné mít znalosti HTML či JavaScriptu. Balíček nabízí velké množství doplňků a komponent uživatelského prostředí pro tvorbu uživatelsky přívětivých aplikací.

2.5.1 ui.R

Shiny aplikace je tvořena dvěma komponentami. Jednou z nich je UI.R, které vytváří grafické rozhraní aplikace. Takové grafické rozhraní umožní uživateli zadávat vstup a dynamicky generovat výstup. Výstup může mít podobu grafů, textu či souboru. Shiny vytváří responzivní GUI, které se přizpůsobí rozlišení obrazovky (vhodné pro mobilní zařízení).

Shiny nabízí většinu grafických prvků, na které jsme zvyklí z jiných vývojových prostředí. Mezi vybrané patří:

- Textové pole

- Posuvníky
- Tlačítka
- Pole se seznamem (combo box)
- Tabulky
- Grafy
- Dialog pro manipulaci se soubory

GUI Shiny aplikace je tvořeno voláním funkcí pro jednotlivé grafické prvky. Například pro vytvoření nadpisu je volána funkce `titlePanel("Napis aplikace")`. Pro vytvoření vstupu textové pole je volána funkce `textInput("nazevPrvku")`. Konfigurace grafických prvků se provádí skrze argumenty funkcí.

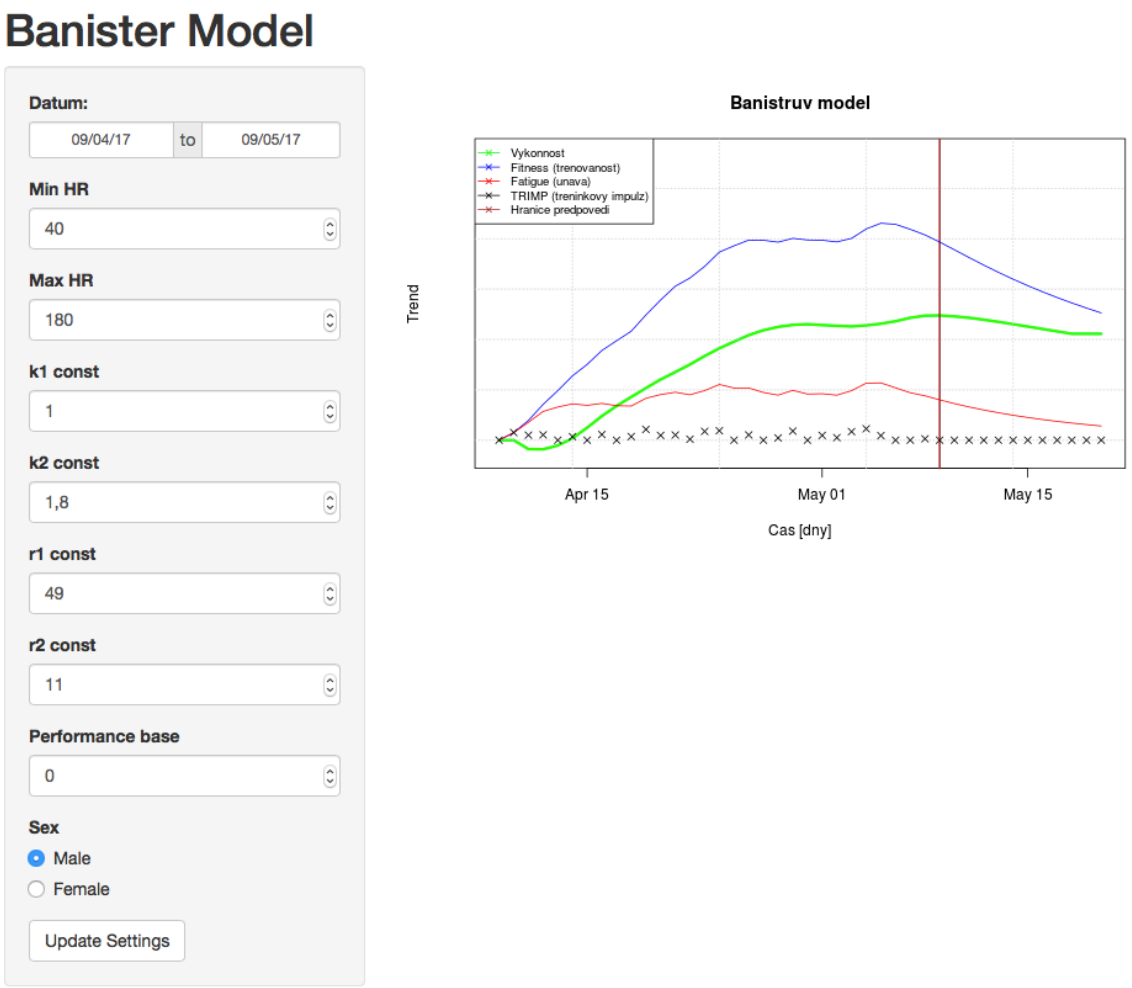
2.5.2 server.R

Druhou komponentou Shiny aplikace je server.R. Tato část aplikace obsahuje procedury, které převedou vstup uživatele na výstup, který má být zobrazen v grafickém rozhraní. Zde se odehrává veškerá logika aplikace. K vstupům od uživatele je přistupováno pomocí proměnné `input` (např.: `input$mazevPrvku`). Podobně se přistupuje i k výstupu proměnnou `output`.

2.5.3 Spouštění Shiny aplikace

Shiny aplikaci je možné spustit funkcí `runApp("slozkaAplikace/")`. Pro odladění aplikace je vhodné použít RStudio, které umožňuje využít breakpointy a chod aplikace krokovat. Předešlím způsobem se aplikace spustí pouze lokálně.

Pro sdílení aplikací je možné využít služby shinyapps.io [12]. Tato cloudová služba nabízí přístup k vlastním aplikacím přes internet bez potřeby vlastního serveru. Základní funkce služby jsou dostupné zdarma. Další možností, jak šířit vlastní aplikace je zprovoznění vlastního Shiny serveru [5]. Tento server je rovněž v základní verzi nabízen zdarma. Shiny serveru je věnována sekce 3.6.5.



Obrázek 2.1: Ukázka aplikace vytvořené balíčkem Shiny.

3 Vývoj aplikace

Aplikace, která je výsledkem práce se jmenuje `Permon`. Jedna se o zkratku z anglického slovního spojení **Per**formance **Mon**itor , jelikož jednou z hlavních funkcí aplikace je monitorování výkonnosti sportovce.

3.1 Motivace pro vznik aplikace

Smyslem aplikace rozhodně není konkurovat stávajícím aplikacím 1.3, které trh se záznamem sportovních aktivit nabízí. Za úkol má tyto aplikace rozšířit o funkce, které buď neobsahují či jsou dostupné za určitý poplatek. Aplikace by také měla dát uživateli větší možnost si prostředí a analýzu dat konfigurovat. Bohužel tyto aplikace není možné doplnit o moduly, je nutné vytvořit vlastní aplikaci.

Jednou z možností jak analyzovat tréninkové jednotky je sledování jejich efektivity a jejich dopad na sportovcovu výkonnost. Tuto funkci aplikace GC a Strava nabízí, nicméně v nepříliš vhodné formě.

Například ve sledování výkonnosti v podání aplikace Strava je problém, že aplikace pracuje pouze s cyklistickými tréninky. Dalším nedostatkem je, že aplikace nenabízí možnost si výpočet výkonnosti nijak přizpůsobit. Výpočet Stravy je založen na Banistrově modelu (viz. 1.4.2), k jehož výpočtu se využívá několik konstant, které by měl mít možnost uživatel upravit. Dalším podstatným nedostatkem sledování výkonnosti v podobě Stravy je, že se jedná o funkci, která je přístupná pouze po zaplacení prémiového předplatného na rok.

Garmin nabízí ve svých zařízení funcki `Training effect`, která rovněž určuje dopad tréninku na sportovcovu výkonnost. Tréninkové jednotce je přidělen stupeň, který určuje, zdali měl trénink udržovací efekt či se jedná o trénink, který zlepší sportovcovu výkonnost. Nevýhodou je, že funkce nepropojuje více tréninků dohromady a neumožňuje vizualizaci hodnot v rámci daného časového úseku.

3.2 Zdroj dat

Zásadní požadavek na funkčnost aplikace je snadný přístup k datům tréninkových jednotek. Jedním ze způsobů, jak získat data pro aplikaci je exportovat aktivity sportovce ze sporttesterů do souborů (např.: `.gpx` či `.fit`) a tyto soubory následně číst a importovat aplikací. Export těchto souborů je zdlouhavý, pracný a celkově takový proces sběru dat není efektivní, zvlášť při hromadném exportu více aktivit.

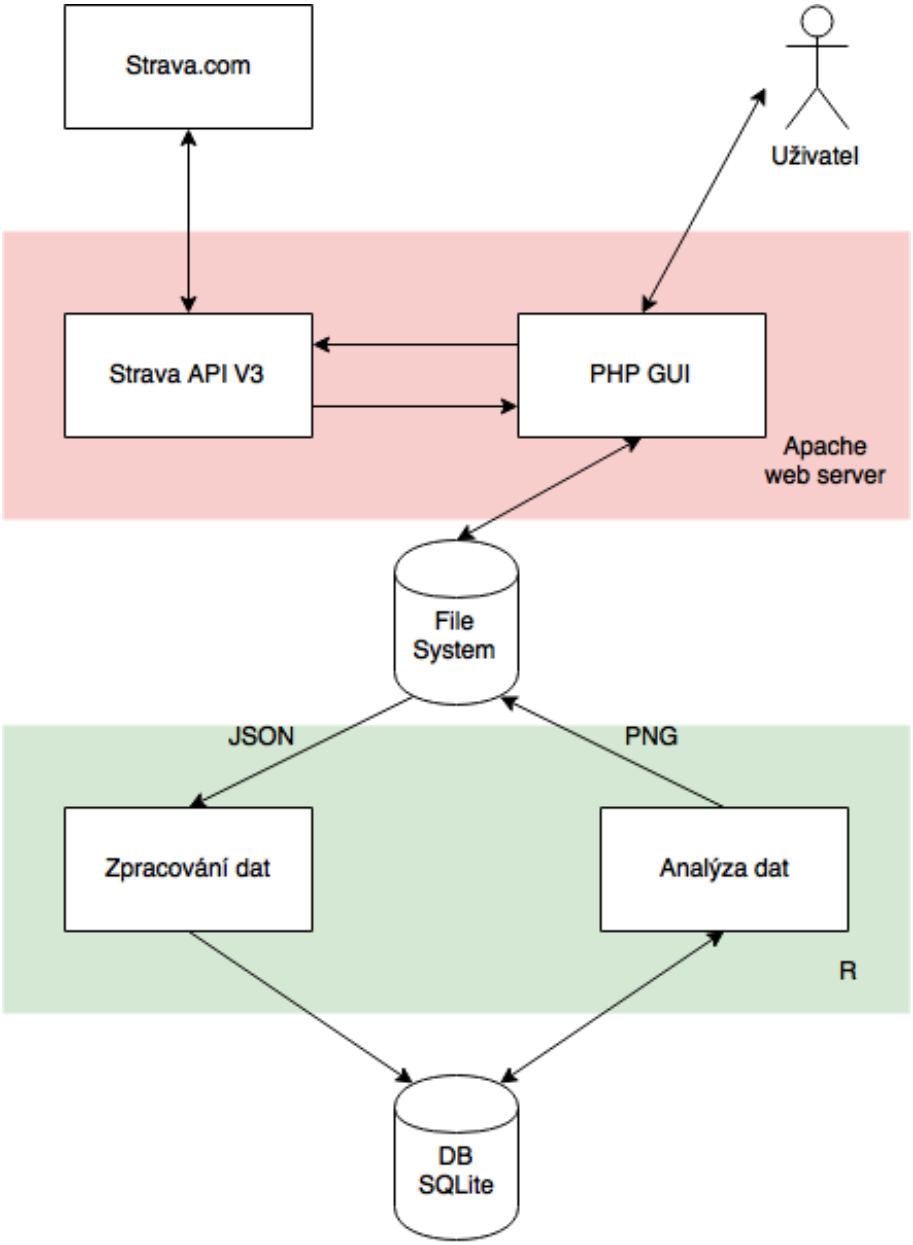
Toto řešení aplikace má pouze jako krajní a nouzové, například pro situace, kdy uživatel má data o sporotvní aktivitě zaznamenaná pouze v podobě souboru.

Daleko efektivnějším způsobem, jak se aplikace může dostat k datům je využití dříve popsaných aplikací třetích stran, které fungují jako různé tréninkové deníky či sportrackery. Tyto aplikaci mají již vyřešený přístup k datům ze sporttesteru a to i po hardwarové stránce. Odpadá nutnost řešit komunikaci s různými druhy zařízení, přičemž způsob komunikace může být u každého zařízení odlišný. Některé z těchto aplikací byly popsána v kapitole 1.3.

3.2.1 Strava API V3 jako zdroj dat

Primárním zdrojem dat pro vyvíjenou aplikaci byla webová vybrána aplikace strava.com (více 1.3.2). A to především díky velmi propracovaném webovému API [22], které aplikace nabízí, které je navíc zdarma. Vyřešen je také import dat z různých zařízení, jelikož strava.com spolupracuje s většinou výrobců těchto zařízení.

Přihlašování je řešeno skrze OAuth2 [2], a tak aplikace pro import dat nepotřebuje získat od uživatele přístupové údaje. Uživatel je přesměrován na autorizační portál Strava API, kde se přihlásí. Po úspěšném přihlášení je přesměrován zpět do aplikace Performance Monitor spolu s přístupovým tokenem. Aplikace se následně prokazuje tímto tokenem, a proto aplikace nemusí znát přístupové údaje uživatele. Protokol OAuth2 zaručuje velkou bezpečnost a proto je pro použití Strava API vyžadován. Samotná komunikace s API probíhá pomocí požadavků a odpovědí HTTP protokolu.



Obrázek 3.1: Diagram toku dat v aplikaci Performance Monitor.

3.3 Převod dat do R struktur

Většina požadavků směřující na Strava API ze strany aplikace jsou typu `GET`. Tato metoda je použita pro získání dat. Odpovědi na požadavky mají strukturu `JSON`.

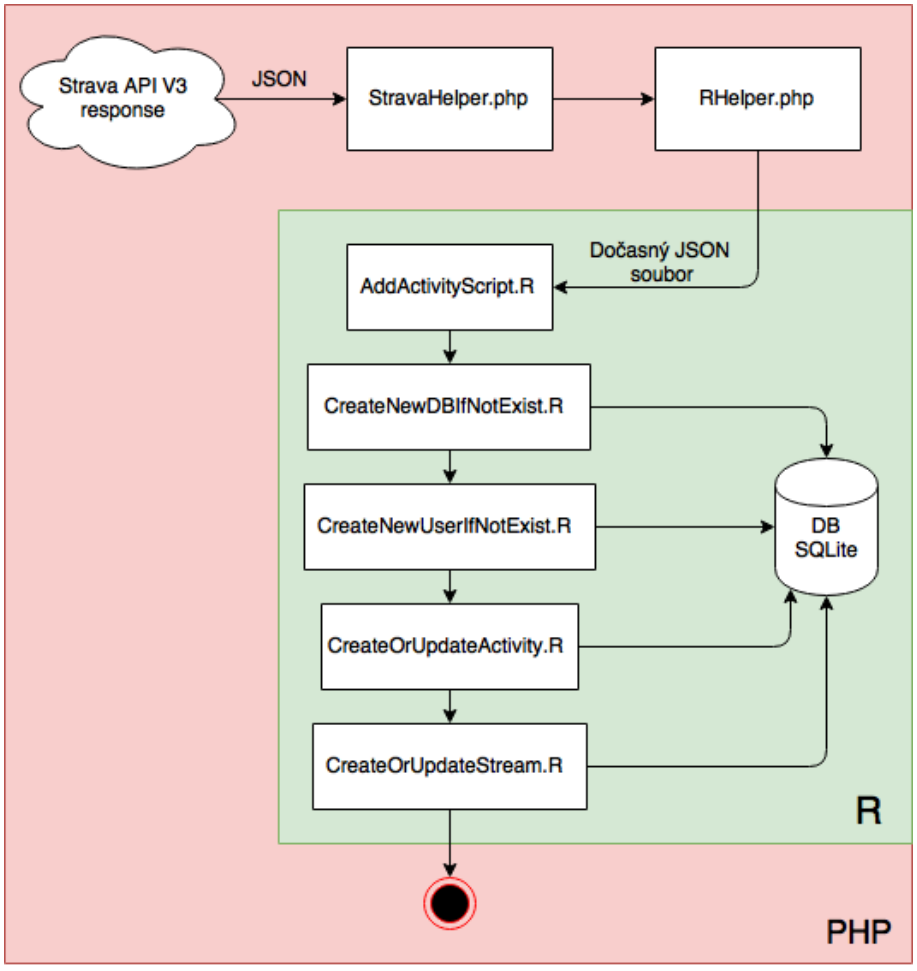
Jelikož je komunikace se Strava API realizována jazykem PHP, tak jsou odpovědi přijmuty pomocí PHP třídy `StravaHelper`. Data z odpovědí jsou uloženy do dočasných souborů ve složce `temp` (viz. 4.1) souborového systému.

Následně je volán skrze PHP třídu `RHelper` R script `AddActivityScript.R`, který načte data z dočasných souborů. Data jsou následně kontrolována či upravena

pro práci v R. Na závěr se data z jednotlivých aktivit uloží do lokální databáze aplikace `Data.db`. Script `AddActivityScript.R` volá další scripty, které realizují další dílčí úkony. Jsou to tyto scripty:

- `CreateNewDBIfNotExist.R` je script, který zkontroluje, zdali je dostupná lokální databáze případně ji vytvoří.
- `CreateNewUserIfNotExist.R` je script, který přidá nového uživatele do databáze v případě, že uživatel není již založen.
- `CreateOrUpdateActivity.R` je script, který přidá či upraví stávající aktivitu, načtenou aktivitou ze souborů JSON.
- `CreateOrUpdateStream.R` je script, který přidá nebo upraví stávající časové řady, pro načtenou aktivitu.

R scripty, které pracují s tréninkovými daty, dále komunikují s lokální databází a dočasné soubory mohou být smazány.



Obrázek 3.2: Diagram procesů, které realizují import dat a převod do struktur jazyka R.

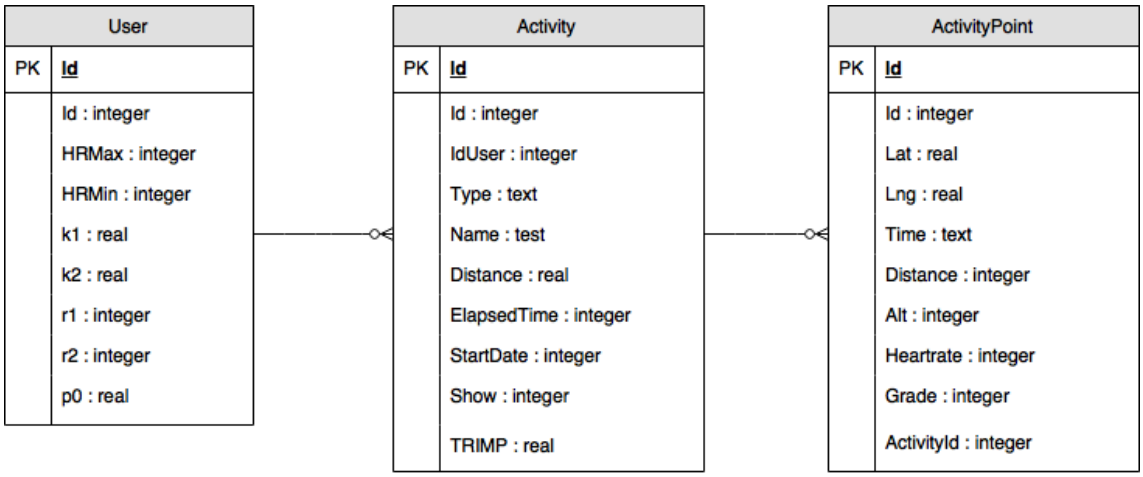
3.4 Úložiště dat a datový model

Následující obrázek 3.3 představuje datový diagram. Je z něho patrný způsob uchování dat o aktivitách přenesených ze `strava.com`. Struktura dat není složitá a bylo nutné udělat rozhodnutí, zdali využit pro uchování dat databázový systém. S ohledem na postupně příbývající objem dat se tato volba jeví jako správná a to především z hlediska výkonu. V případě uchování dat v podobě souborového systému by výkon s přibývajícími daty výrazně klesal. Další výhodou použití databáze je dotazovací jazyk (např.: dotazovací jazyk SQL).

Využit je relační databázový systém `SQLite` [19]. A to zejména z důvodů toho, že se jedná o systém, který funguje s absencí serveru a data jsou ukládána pouze v jednom souboru, který není závislý na platformě. To přináší výhodu snadné přenositelnosti databáze. `SQLite` také postrádá konfigurační soubory, a tak je jediným možným způsobem, jak konfigurovat databázový systém, využít příkazů `PRAGMA`.

Krom původních importovaných dat jsou do databáze ukládáné i později vypočtené hodnoty tykajících se tréninkových záznamů či hodnoty týkající se uživatele (např.: minimální a maximální tepová frekvcence).

Komunikace s databází probíhá pouze skrze R scripty. Použit je balíček `sqldf` popsaný v 3.8.2.



Obrázek 3.3: Datový diagram databáze sportovních dat.

SQLite implementuje téměř celý standart `SQL-92`, až na některé typy příkazů (např.: `OUTER RIGHT JOIN`).

3.5 Dokumentace R scriptů

V následující sekci jsou zdokumentované vyznamné R scripty a jejich funkce.

3.5.1 Script TrainingImpulsePerformanceCalc.R

Script, který je zaměřen na modelaci výkonnosti pomocí Banistrova modelu. Dále obsahuje funkce spojené s kalibrací tohoto modelu. Script pracuje s balíčky `sqldf` a `getopt`.

Pro větší efektivitu při práci s daty byly vytvořené dvě struktury, se kterými script pracuje. První z nich je struktura `Activity`, která má následující podobu:

- `Activity$UserId`: id uživatele
- `Activity$Activity`: základní informace o aktivity v podobě objektu `data frame`.
- `Activity$Stream`: stream aktivity

Další strukturou je `Stream`. Jedná se o strukturu, které uchovává informace průběhu aktivity v podobě časových řad. Je tvořena objektem `data frame`. `Stream` tvoří entity ActivityPoint (viz. 3.4) převedené do objektu `data frame`.

Funkce HRReserve

Funkce pro výpočet vážené tepové frekvence na základě klidové a maximální tepové frekvence.

`@param HR`: Hodnota tepové frekvence, která má být převedena.

`@param HRMin`: Hodnota klidové tepové frekvence..

`@param HRMax`: Hodnota maximální tepové frekvence.

`@return`: Vrací hodnotu vážené tepové frekvence.

```
HRReserve <- function(HR, HRMin, HRMax)
```

Funkce TRIMPExponentialHRScalingForActivity

Funkce výpočet hodnotu tréninkové impulzu z tréninkové jednotky.

`@param Stream`: Stream aktivity.

`@param HRMax`: Hodnota maximální tepové frekvence.

`@param HRMin`: Hodnota klidové tepové frekvence.

`@param Male`: Boolean hodnota TRUE v případě, že uživatel je můž.

`@param TimeUnitInSecond`: Počet záznamů ve streamu za jednu sekundu.

`@param TimeUnitsInMinute`: Počet záznamů ve streamu za jednu minutu.

`@return`: Vrací hodnotu reálného čísla reprezentující hodnotu TRIMP, NULL v případě nemožnosti výpočtu.

```
TRIMPExponentialHRScalingForActivity <- function(Stream, HRMax,
HRMin, Male = TRUE, TimeUnitInSecond = 1, TimeUnitsInMinute = 60)
```

Funkce CalculateOrGetTRIMPExponentialHRScalingForActivity

Funkce, která získá již vypočtenou hodnotu TRIMP z databáze. Případně ji vypočte a uloží do databáze.

`@param Activity`: Struktura tréninkové jednotky.

`@param Stream`: Stream aktivity.

`@param HRMax`: Hodnota maximální tepové frekvence.

`@param HRMin`: Hodnota klidové tepové frekvence.

`@param Male`: Boolean hodnota TRUE v případě, že uživatel je muž.

`@param TimeUnitInSecond`: Počet záznamů ve streamu za jednu sekundu.

`@param TimeUnitsInMinute`: Počet záznamů ve streamu za jednu minutu.

`@param Recalculate`: Boolean TRUE v případě vyžádání přepočtení hodnoty.

`@return`: Vrací hodnotu reálného čísla reprezentující hodnotu TRIMP, NULL v případě nemožnosti výpočtu.

```
CalculateOrGetTRIMPExponentialHRScalingForActivity <-
function(Activity, Stream, HRMax, HRMin, Male = TRUE,
TimeUnitInSecond = 1, TimeUnitsInMinute = 60, Recalculate =
FALSE)
```

Funkce TRIMPExponentialHRScalingForDateRange

Funkce, která získá hodnoty TRIMP pro každý den v daném rozmezí.

`@param From`: Datum Od ve formátu %d/%m/%Y.

`@param To`: Datum Do ve formátu %d/%m/%Y.

`@param UserId`: Id uživatele.

`@param Male`: Boolean hodnota TRUE v případě, že uživatel je muž.

`@param TimeUnitInSecond`: Počet záznamů ve streamu za jednu sekundu.

`@param TimeUnitsInMinute`: Počet záznamů ve streamu za jednu minutu.

`@return`: Vrací pojmenovaný seznam hodnot TRIMP. Jména hodnot jsou jednotlivé datumy v rámci zvoleného časového období.

```
TRIMPExponentialHRScalingForDateRange <- function(From, To,
UserId, Male = TRUE, TimeUnitInSecond = 1, TimeUnitsInMinute =
60)
```

Funkce PerformanceBanisterModel

Funkce, která vypočítává Banistrův model.

`@param From`: Datum Od ve formátu %d/%m/%Y.

`@param To`: Datum Do ve formátu %d/%m/%Y.

`@param UserId`: Id uživatele.

`@param Male`: Boolean hodnota TRUE v případě, že uživatel je muž.

`@param k1`: Hodnota konstanty k1.

`@param k2`: Hodnota konstanty k2.

`@param r1`: Počet dní za jak dlouho se vrátí složka Fitness do původní hodnoty.

`@param r2`: Počet dní za jak dlouho se vrátí složka Fatigue do původní hodnoty.

`@param p0`: Počáteční hodnota složky výkonnosti.

`@param CalculateNextFewEmptyDays`: Boolean hodnota, TRUE v případě doplnění předpovědi výkonnosti.

`@return`: Vrací seznam 4 složek Fitness, Fatigue, Performance a TRIMP pro každé datum.

```
PerformanceBanisterModel <- function(From, To, UserId, Male =
TRUE, k1 = 1.0, k2 = 1.8, r1 = 49, r2 = 11, p0 = 0,
CalculateNextFewEmptyDays = TRUE)
```

Funkce Plot

Funkce, která vykresluje složky Banistrova modelu proložené klouzavím průměrem.

`@param Model`: Struktura reprezentující Banistrův model.

`@param Save`: Boolean hodnota, v případě TRUE dojde k uložení výsledného grafu.

`@param r1`: Konstanta r1 Banistrova modelu pro doplnění předpovědi.

`@return`: N.A

```
Plot <- function(Model, Save = FALSE, r2 = 11)
```

Funkce FindSameIntervalsInRun

Funkce, která hledá rovinné úseky ve vložené sportovní aktivitě.

`@param Run`: Struktura běžecké aktivity.

`@param AltTolerance`: Hodnota tolerance rovinatého úseku.

`@return`: Seznam rovinatých úseků.

```
FindSameIntervalsInRun <- function(Run, AltTolerance = 15)
```

Funkce CalculateSpeedForRowInStream

Funkce, která počítá rychlost mezi dva záznamy streamu aktivity.

`@param Row`: Aktuální zázname streamu.

`@param PreviousRow`: Předchozí záznam streamu.

`@return`: Vrací číselnou hodnotu rychlosti mezi záznamy.

```
CalculateSpeedForRowInStream <- function(Row, PreviousRow)
```

Funkce GetStraightsInGivenDistance

Funkce, která vrací rovinaté úseky aktivity o dané délce.

`@param Run`: Struktura běžecké aktivity.

`@param MetersLong`: Délke úseků v metrech.

`@param AltTolerance`: Hodnota tolerance rovinatého úseku.

`@return`: Číselná hodnota nejrychlejšího času v sekundách.

```
GetStraightsInGivenDistance <- function(Run, MetersLong = 300,
AltTolerance = 15)
```

Funkce GetFastestStraightsInGivenDistanceOfActivity

Funkce, která vrací nejrychlejší čas na úseku o dané velikosti aktivity.

`@param Run`: Struktura běžecké aktivity.

`@param MetersLong`: Délka úseků v metrech.

`@param AltTolerance`: Hodnota tolerance rovinatého úseku.

`@return`: Seznam rovinatých úseků o dané délce.

```
GetFastestStraightsInGivenDistanceOfActivity <- function(Run,
MetersLong = 300, AltTolerance = 15)
```

Funkce GetBestTimeForEachDayForStraight

Funkce, která z daných aktivit vytvoří časové řady s nejlepšími dosaženými časy na daný rovinatý úsek pro každý den.

`@param Activities`: Seznam struktur aktivit.

`@param MetersLong`: Délka úseků v metrech.

`@param AltTolerance`: Hodnota tolerance rovinatého úseku.

`@return`: Číselná hodnota nejrychlejšího času v sekundách pro každý den.

```
GetBestTimeForEachDayForStraight <- function(Activities,
MetersLong = 300, AltTolerance = 15)
```

3.5.2 Script GetActivity.R

V tomto scriptu jsou obsaženy funkce, které zprostředkovávají komunikaci s databází. Script pracuje s balíčky `sqldf` a `getopt`.

Funkce GetActivity

Funkce, která vyhledá v databázi aktivitu podle zadaného id.

`@param Id`: Id hledané aktivity.

`@return`: Vrací hledanou aktivitu v její struktuře.

```
GetActivity <- function(Id)
```

Funkce GetActivities

Funkce, která vyhledá v databázi aaktivity podle zadaných id.

`@param Ids`: Vektor id hledaných aktivit.

`@return`: Vrací seznam vyhledaných aktivity.

```
GetActivities <- function(Ids)
```

Funkce GetActivitiesInDateRange

Funkce, která vyhledá v databázi aktivity uživatele v zadaném časovém období.

`@param From`: Datum Od ve formátu %d/%m/%Y.

`@param To`: Datum Do ve formátu %d/%m/%Y.

`@param UserId`: Id uživatele.

`@return`: Vrací seznam vyhledaných aktivity.

```
GetActivitiesInDateRange <- function(From, To, UserId)
```

3.6 Použité technologie

Z předchozích kapitol je patrné, že jádro aplikace je vyvíjeno v jazyce R (viz 2). Pomocí R jsou počítány veškeré analýzy nad daty. Ačkoliv jsou R scripty koncipovány tak, aby je bylo možné spouštět v příkazové řádce a výpočty konfigurovat pomocí parametrů, je součástí aplikace grafické uživatelské rozhraní vyvíjené v jazyce PHP. Jak bylo zmíněno, manipulace s daty probíhá pomocí dotazovacího jazyka SQL.

3.6.1 Použití jazyka R

Jazyk R je primárně zaměřený na statistické výpočty a právě proto byl využit i pro tuto práci. Analyzování velkých dat je v R velmi rychlé a efektivní. R například disponuje velmi efektivními nástroji pro práci s časovými řadami. Dalším důvodem pro použití R jsou mé předchozí zkušenosti s tímto prostředím z bakalářského projektu.

3.6.2 Použití jazyka PHP

Úmysl použití jazyka PHP v souvistlosti s grafickým uživatelským rozhraním spočívá především v tom, aby aplikace, respektive `GUI` bylo multiplatformní. Komunikace se Strava API V3 je zprostředkována rozvněž na úrovni PHP s využitím knihovny StravaApi (citace). Pro analýzu a výpočty jsou pak volány R scripty s různými parametry. Výstup těchto scriptů je následně zobrazen uživateli zpětně v GUI. Celý proces znázorňuje obrázek 3.1.

3.6.3 Virtuální server permon.mti.tul.cz

Pro chod aplikace by obyčejný webový server s podporou PHP byl nedostačující. To zejména z důvodu použití jazyka R a Shiny serveru, jelikož běžná distribuce OS těmito technologiemi nedisponuje. Proto byl použit virtuální počítač ze síte Virtul. Virtuální počítač byl vytvořen v následující konfiguraci:

- **Procesor:** 1
- **Paměť:** 2 GB
- **Prostor:** 16 GB
- **Doména:** permon.mti.tul.cz

Jako operační systém je použit Ubuntu 16.04 v 64-bitové desktopové verzi. Tento OS byl vybrán především díky ověřené funkčnosti Shiny serveru na této plarformě. Kromě samotného Shiny serveru byl rovněž nainstalován Apache server, SSH server a jazyk R.

Firewall serveru je konfigurován tak, že jsou přístupné zvenční následující porty:

- **22** pro administraci serveru pomocí SSH.
- **80** pro webový server na protokolu HTTP (přesměrování na port 443 na úrovni webového serveru Apache).
- **443** protokol HTTPS webového serveru
- **3838** pro Shiny server

3.6.4 Webový server Apache

Spolu s použitím jazyka PHP je zapotřebí použít webový server. Byl vybran webový server Apache HTTP Server. Apache je rozšířen na většinu dnes používaných platforem. Na server byl nainstalován PHP modul ve verzi 7.0. Veškerá komunikace webového serveru je přesměrována na port 443, a tudíž je komunikace realizována na protokolem HTTPS. Je použit certifikát podepsaný sám sebou, nebo Self-signed certificate a uživatel tak musí přidat do svého webového prohlížeče bezpečnostní výjimku.

3.6.5 Shiny server

Pro běh Shiny aplikací, které jsou podstatnou součástí celé aplikace Permon, je zapotřebí vlastního Shiny serveru. Služba shinnyapps.io nemohla být použita, jelikož Shiny aplikace musí mít přístup k databázi s daty aplikace. RStudio nabízí nyní předkompilovanou verzi Shiny serveru pro operační systém Ubuntu 12.04 a vyšší ve 64-bitové verzi. Základní a neplacená verze serveru je plně vyhovující pro aplikaci

Permon. Neplacená verze umožňuje chodu více aplikací zároveň a přístup k aplikacím přes internet. Chybí funkce přihlášení uživatelů, ale to je v aplikaci vyřešeno pomocí Strava API v PHP části aplikace. Dalším významným nedostatkem neplacené verze je, že nemožňuje zabezpečení aplikací protokolem SSL, a tak jednotlivé Shiny aplikace nejsou takto zabezpečené narozdíl od zbytku aplikace.

3.7 Vývojové prostředí

3.7.1 RStudio

Jako vývojové prostředí při vytváření R scriptů aplikace bylo použito RStudio [4]. RStudio je v základní verzi volně dostupné pro Windows, macOS a Linux. RStudio má mnoho funkcí i nad rámec této práce a zde jsou vypsány jen vybrané funkce, které byly využity při tvorbě aplikace:

- Možnost debugování a krokování R scriptů
- Jednoduchá správa R balíčků včetně jejich aktualizací
- Generování reportů k R scriptům

3.8 Použité knihovny a balíčky

3.8.1 R balíček jsonlite

Pro zpracování konfiguračních souborů ve formátu `jsonlite` byl použit balíček `jsonlite` [21]. Tento balíček není obsažen v základní instalaci R a je tedy nutné ho stáhnout z repozitáře CRAN spolu s balíčky na něm závuslými. Mezi základní funkce balíčku patří funkce `toJSON()`, která převádí R objekty do JSON a funkce `fromJSON()`, která naopak převádí JSON na objekty v R.

3.8.2 R balíček sqldf

Balíček, který umožňuje, jazyku R, komunikovat s vybranými databázovými systémy pomocí SQL příkazů. Mezi podporované databázové systémy patří: SQLite [20], H2, PostgreSQL a MySQL. Výsledek SQL příkazů je uchován v datové struktuře `data frame`, tedy struktuře, která má velmi blízko databázovým tabulkám. Spojení s databází se vytvoří pomocí funkce `dbConnect()` a dotazy na databázy se provádí voláním funkce `dbGetQuery()`, která jako parametry přijímá spojení s databází a SQL příkaz.

3.8.3 R balíček getopt

Balíček `getopt` [23] usnadňuje práci s volbami a vstupními parametry uživatele v případě, že je R script spouštěn z terminálu. Balíček je ekvivalentem funkce `getopt()`

v jazyce C. Roztřídí všechny parametry na vstupu a u nich kontroluje předepsaný datový typ. Součástí balíčku je tisk nápovědy ke skriptu. Všechny volby a parametry se předem specifikují a případně doplňují i o text nápovědy. Základní funkcí balíku je `getopt()`, která vrací pojmenovanou strukturu `list` obsahující vstupní parametry.

4 Adresářová struktura aplikace

Použité programovací jazyky R a PHP nenutí vývojáře dodržet specifickou adresářovou strukturu aplikace, její návrh je tedy zcela na samotném vývojáři. Struktura Performance Manageru je navržena tak, aby byla oddělena výpočetní část psaná v jazyce R od PHP části, která definuje grafické rozhraní.

Kořenový adresář obsahuje inicializační soubor aplikace `index.php` a soubor s popisem aplikace `README.md`. Dále obsahuje následující podadresáře:

4.1 Adresář r/

Adresář, který obsahuje veškeré R scripty a další soubory:

- Složku se scripty pro manipulaci s daty (`/r/DataManipulation`).
- Složku se scripty pro analýzu dat (`/r/Calc`).
- Soubor databáze `Data.db`, ve které jsou ukládány veškerá data a pomocné výpočty
- Script `CreateDataDB.sql`, který výtváří zmiňovanou databázi.
- Podadresář `/r/temp`, kde jsou ukládány různé dočasné soubory, jako vygenerované grafy či JSON soubory s daty.

4.2 Adresář php/

Tento adresář uchovává PHP část aplikace. Obsaženy jsou zdrojové kódy pro grafické uživatelské rozhraní:

- Model (`/php/model`).
- View (`/php/view`).
- Controller (`/php/controllers`).
- PHP script `routes.php`, který reaguje na uživatelské požadavky a zobrazuje příslušné komponenty GUI.
- Adresář `/php/r` jsou třídy pro spouštění R scriptů.

- V adresáři `/php/strava` uchovány třídy pro komunikaci se Strava API.
- Adresář `/php/logger` obsahuje třídu pro logování událostí v aplikaci.

4.3 Adresář log/

Adresář, který obsahuje `.log` soubory, do kterých je logována činost aplikace.

4.4 Adresář install/

Adresář s R scriptem `R_packages_install.R`, který aktualizuje balíčky a instaluje chybějící R balíčky nutné pro chod aplikace.

5 Aplikace z pohledu uživatele

Aplikace `Permon` je dostupná na adrese `permon.mti.tul.cz`. Pro použití aplikace je nezbytné vlastnit účet webové aplikace Strava a jím se do aplikace přihlásit (viz. 3.2.1).

5.1 GUI aplikace

Grafické uživatelské rozhraní webové aplikace Permon je možné rozdělit na dvě dílčí části. V první části má uživatel možnost se přihlásit a manipulovat s novými daty. V této části je možné spouštět Shiny aplikace. Shiny aplikace pak tvoří druhou část GUI. Každá Shiny aplikace je spouštěna automaticky ve vyskakovacím okně (pop-up). V případě, že webový prohlížeč tato okna blokuje, je aplikace otevřena ve nové záložce prohlížeče.

5.1.1 Přihlášení uživatele

Jak již bylo zmíněno, aplikace je uživateli přístupná uživateli pouze po přihlášení účtem webové aplikace Strava. Pro přihlášení je uživatel přesměrová na aplikaci Strava, kde má možnost se přihlásit stávajícím účtem či si vytvořit nový.

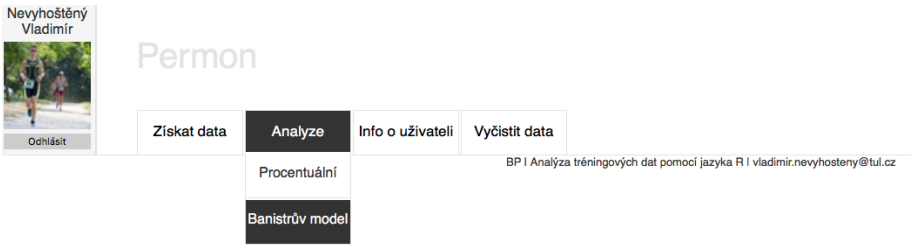
Obrázek 5.1: Přihlašovací dialog aplikace Permon.

5.1.2 Po přihlášení uživatele

Po úspěšném přihlášení uživatele se mu zpřístupní menu aplikace. Rovněž aplikace získá základní informace o uživateli (např.: jméno a avatar uživatele).

Strukturu menu aplikace je následující:

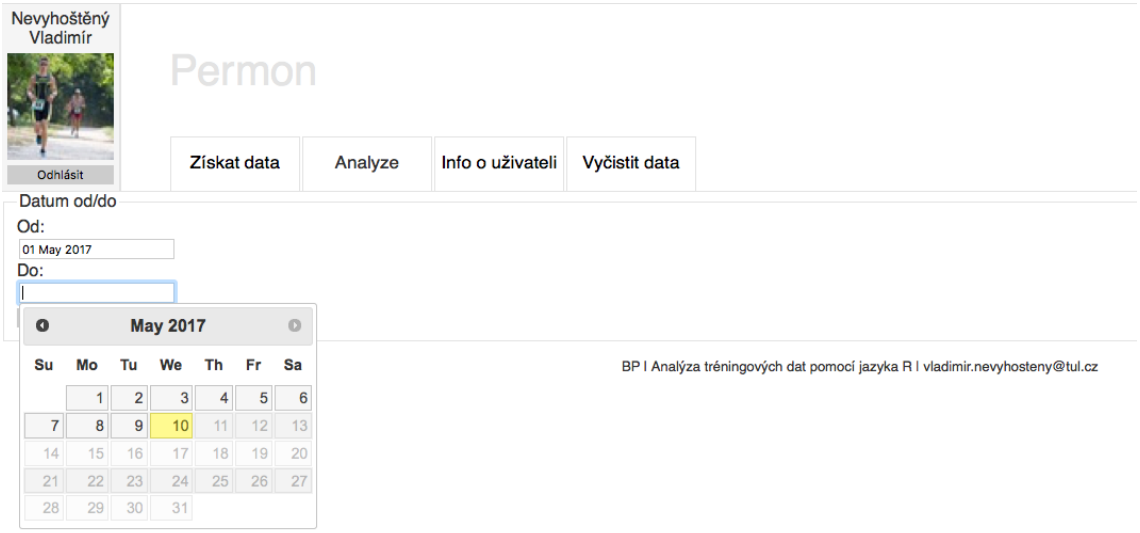
- **Získat data:** import nový dat z aplikace Strava.
- **Analyze:**
  - **Procentuální:** Shiny aplikace pro vizualizaci poměru tréninků pro vybrané období.
  - **Banistrův model:** Shiny aplikace pro modelování výkonnosti
- **Info o uživateli:** základní informace o nahraných datech uživatele.
- **Vyčistit data:** možnost smazání dočasných souborů uživatele.



Obrázek 5.2: Aplikace po přihlášení uživatele a její menu.

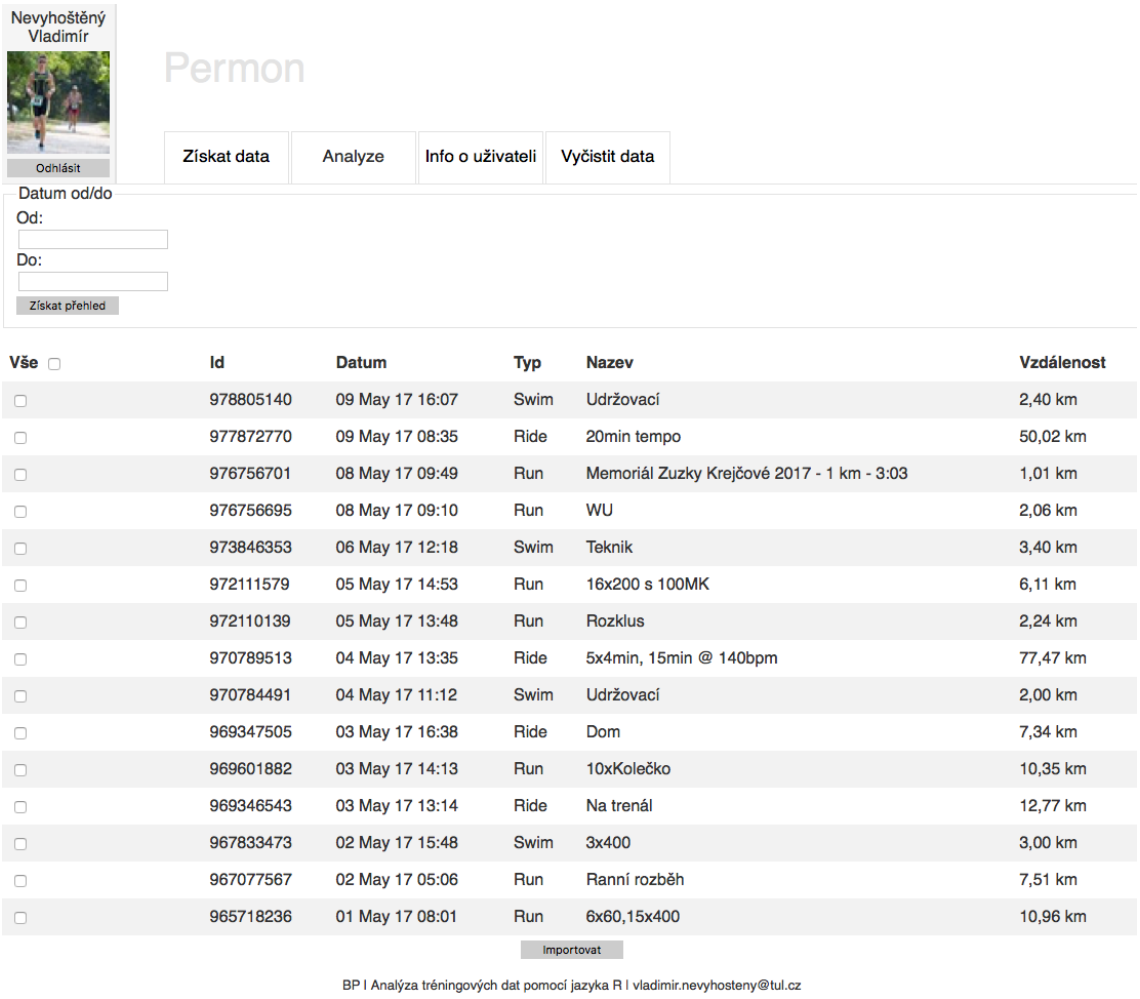
5.1.3 Import nových dat do aplikace

Je nezbytné, aby uživatel importoval data jeho sportovních aktivitách do aplikace Permon. Data se importují z webové aplikace Strava. Uživatel nejprve zvolí pomocí prvku GUI `date picker` datumy od a do kdy mají být aktivity importovány. Výchozí nastavení je poslední týden v případě, že uživatel nezvolí žádný datum.



Obrázek 5.3: Výběr časového rozmezí pro import nových dat z aplikace Strava.

Následně se uživateli zobrazí tabulka s přehledem aktivit v daném časovém období. Aktivity jsou zobrazené se základními informacemi o aktivitě a uživatel si může vybrat konkrétní aktivity, kterou budou importovány, nebo importovat všechny. V případě, že byla zvolena aktivita, která byla již v minulosti importována, dojde k jejímu přepsání, a tak se aktivity nebudou duplikovat.

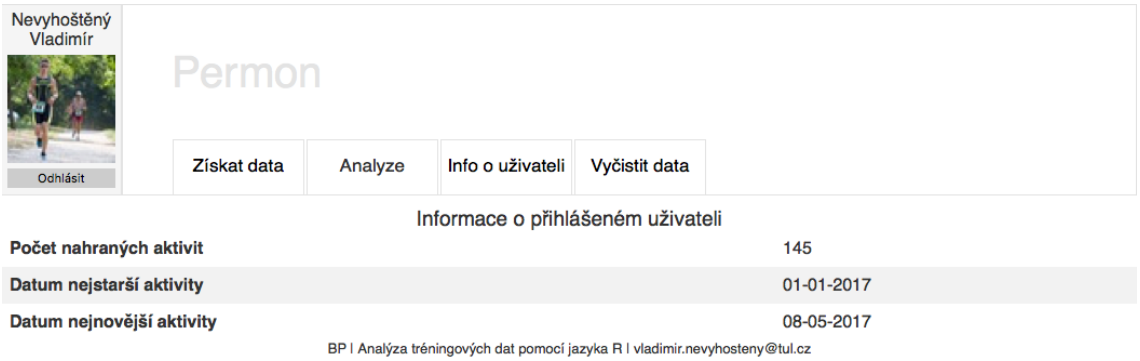


Vše	Id	Datum	Typ	Nazev	Vzdálenost
☐	978805140	09 May 17 16:07	Swim	Udržovací	2,40 km
☐	977872770	09 May 17 08:35	Ride	20min tempo	50,02 km
☐	976756701	08 May 17 09:49	Run	Memoriál Zuzky Krejčové 2017 - 1 km - 3:03	1,01 km
☐	976756695	08 May 17 09:10	Run	WU	2,06 km
☐	973846353	06 May 17 12:18	Swim	Teknik	3,40 km
☐	972111579	05 May 17 14:53	Run	16x200 s 100MK	6,11 km
☐	972110139	05 May 17 13:48	Run	Rozklus	2,24 km
☐	970789513	04 May 17 13:35	Ride	5x4min, 15min @ 140bpm	77,47 km
☐	970784491	04 May 17 11:12	Swim	Udržovací	2,00 km
☐	969347505	03 May 17 16:38	Ride	Dom	7,34 km
☐	969601882	03 May 17 14:13	Run	10xKolečko	10,35 km
☐	969346543	03 May 17 13:14	Ride	Na trenál	12,77 km
☐	967833473	02 May 17 15:48	Swim	3x400	3,00 km
☐	967077567	02 May 17 05:06	Run	Ranní rozběh	7,51 km
☐	965718236	01 May 17 08:01	Run	6x60,15x400	10,96 km

Obrázek 5.4: Přehled aktivit určených pro import. Pomocí zašktávacího pole může uživatel určit, které aktivity budou importovány.

5.1.4 Informace o datech uživatele

Pro větší přehled uživatele o jeho importovaných datech je zde stránka **Info o uživateli**. Stránka zobrazuje počet nahraných aktivit, a také datum nejstarší a nejnovější importované aktivity.



Obrázek 5.5: Přehled importovaných aktivit uživatele.

5.1.5 Shiny aplikace Banistrův model

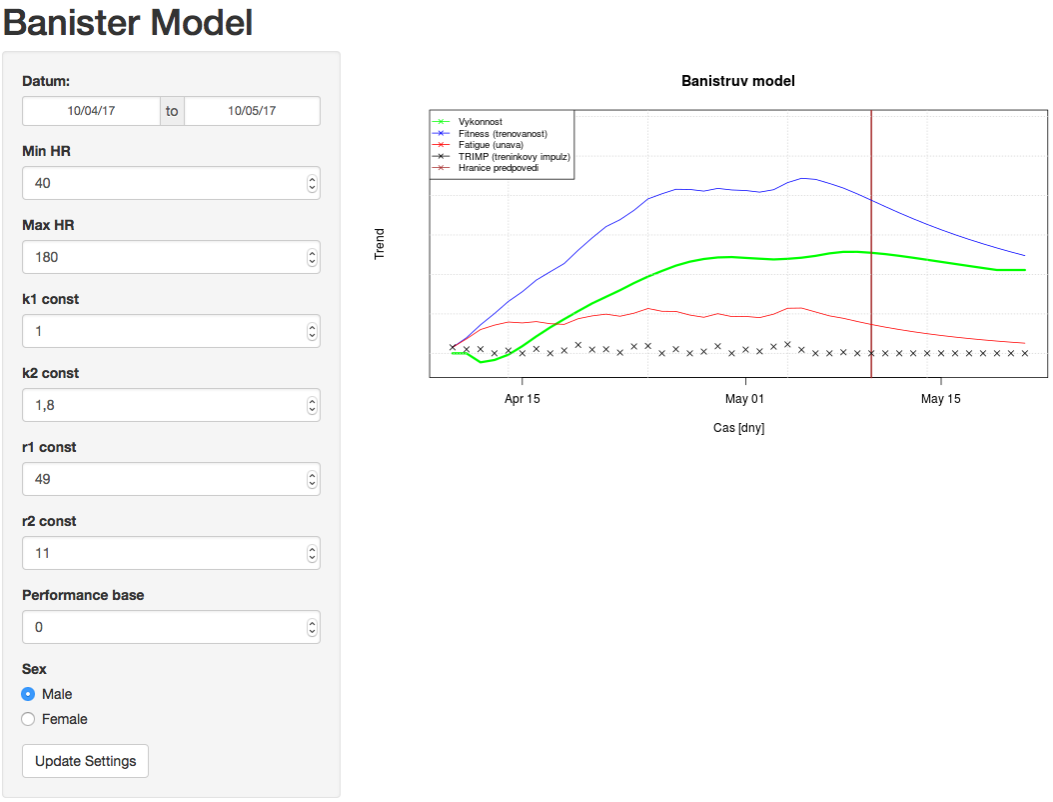
Jak bylo již popsáno Shiny aplikace se otvírají v odděleném okně či záložce webového prohlížeče (záleží na nastavení prohlížeče). Je tomu tak i u Shiny aplikace Banistrův model, která je podstatným prvkem aplikace Permon. Podrobně je Banistrův model popsán v sekci 1.4.2.

Grafické uživatelské rozhraní této aplikace je rozděleno na dvě funkční části. První částí je panel s uživatelskými vstupy. Pomocí těchto vstupů je uživatel schopen konfigurovat výsledný model. Hodnoty vstupů je možné uložit tlačítkem `Update Settings`, tak aby je uživatel nemusel při opětovném použití aplikace zadávat znova. Model je možné konfigurovat následujícími vstupy:

- **Datum:** pomocí prvku `date picker` se zvolí časové období, pro které má být model vypočten.
- **Min HR:** minimální neboli klidová tepová frekvence uživatele.
- **Max HR:** maximální tepová frekvence. Spolu s hodnotou Min HR tyto hodnoty ovlivňují `TRIMP` (viz. 1.4.1).
- **k1 const, k2 const, r1 const, r2 const, Performance base:** jsou vstupy pro konstanty, které definují model. Více v sekci 1.4.2.
- **Sex:** tímto vstupem se určuje pohlaví uživatele, které rovněž ovlivní výpočet modelu.

Druhou částí je graf, který je výstupem aplikace. Jednotlivé složky modelu jsou zobrazeny křivkami a popsány legendou, která se nachází v levé horní části grafu. Nejdůležitější složka `Výkonnost` je popsána křivkou zelené barvy. Trénovanost modrou barvou a únava červenou. Velikost tréninkových impulzů pro každý den je vyznačena černými křížky. Vertikální hnědá přímka určuje hranici, od které model začíná jednotlivé složky předpovídat. Délku předpovědi určuje hodnota konstanty `r1`. Osa **X** je popsána jako `Cas` a jednotlivé hodnoty iterují po dnech. Vzhledem

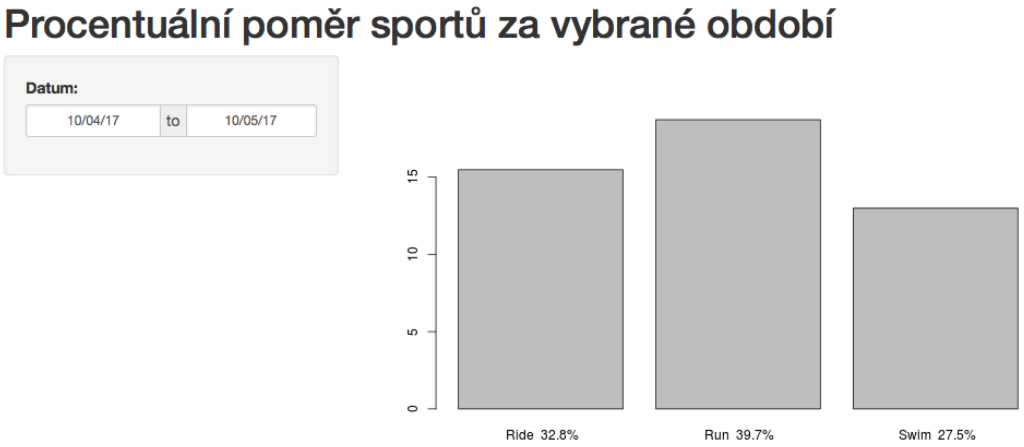
k tomu, že hodnoty modelu jsou bezrozměrné a nepříliš důležité a je spíše podstatné sledovat průběh křivek, nemá osa **Y** uvedené hodnoty a je popsána jako `Trend`.



Obrázek 5.6: Shiny aplikace pro výpočet Banistrova modelu.

5.1.6 Shiny aplikace Procentuální poměr sportů za vybrané období

Další Shiny aplikací, která je součástí aplikace Permon je **Procentuální poměr sportů za vybrané období**. Jak vypovídá její název umožňuje uživateli získat přehled kolik procent celkového času zabral daný sport na vybraném časovém období. Jediným vstupem aplikace jsou datumy určující zmiňované časové období.



Obrázek 5.7: Grafické uživatelské rozhraní Shiny aplikace Procentuální poměr sportů za vybrané období.

5.2 Přímé ovládání R scriptů

Vybrané R scripty jsou vyvíjeny tak, aby je bylo možné spoustit i bez použití GUI. Tyto scripty jsou spouštěny pomocí terminálu (příkazové řádky). Uživatel může takto spouštět scripty v případě, že nemá k dispozici webový server či chce nějakým způsobem činnost scriptů modifikovat. Tyto scripty vyžadují pro jejich fungování různé vstupní parametry, jako například: cesta k lokální databázi s daty či ID uživatele. Lokální databázi si uživatel může vygenerovat s pomocí SQL scriptu `CreateDataDB.sql`. Vstupní parametry se zadávají obdobným způsobem, jako u příkazů v terminálu. Scripty jsou spouštěny příkazem `Rscript`.

Pro znázornění, například tisk nápovědy pro script se provede následujícím způsobem `Rscript TrainingImpulsePerformanceCalc.R -h`.

Závěr

Výsledkem této bakalářské práce je funkční webová aplikace Permon, která umožňuje analyzovat tréninkové jednotky. Jelikož je takových aplikací mnoho, bylo zapotřebí se důkladně seznámit co v současnosti trh nabízí a jaké má naopak nedostatky. Aplikace Permon funguje jako doplněk k těmto aplikacím a nemá tak za cíl konkurovat ostatním aplikacím. Bylo také nutné vyřešit otázku, jak bude aplikace získavat sportovní data. I tato problematika je rozebrána v teoretické části spolu s nejčastějšími formáty a také zařízeními, které tréninkovou jednotku monitorují.

Téměř veškerá logická část aplikace je vyvíjena v jazyce R. Práce s jazykem R v takovém rozsahu byla pro mě nová, a tak je jazyku R věnována celá kapitola, která se zaobírá základním principům a balíčkům, které byly použity při vývoji aplikace. Při vývoji byl z velké části použit balíček Shiny, který umožňuje vytvářet webové aplikace kompletně v jazyce R.

Zdrojem dat pro aplikaci Permon se stalo API webové aplikace Strava. Strava API V3 bylo použito, protože je velmi propracované a především je volně dostupné, což jiné aplikace nenabízí. Pro přístup do aplikace Permon jsou využité uživatelské účty aplikace Strava. Přihlášení probíhá pomocí protokolu OAuth2. Importovaná data jsou převedena do vlastní struktury a uchována v lokální SQLite databázi.

Grafické uživatelské rozhraní je vyvíjeno v jazyce PHP. Jazyk PHP byl použit, aby byla vytvořena webová aplikace, která bude mutliplatformní. V PHP je rovněž implementovaná veškerá komunikace se Strava API spolu s přihlašováním uživatelů pomocí protokolu OAuth2. Aplikace je dostupná na adrese **permon.mti.tul.cz**. Komunikace je automaticky přesměrovaná na protokol HTTPS a webová aplikace podepsána vlastnoručně podepsaným certifikátem. Aplikaci byl přidělen virtuální server ze univerzitní sítě Virtul. Bylo nutné provést konfiguraci tohoto serveru a nainstalování potřebných technologiích. Jako webový server byl použit server Apache. Pro chod R části aplikace byl nainstalován Shiny server.

Aplikace Permon je určena především amatérským sportovcům, pro které jsou pokročilé funkce profesionálních aplikací nedostupné. Podstatnou část aplikace tvoří modelování výkonnosti sportovce pomocí Banistrova modelu. Tato analýze může napomoc sportovci vyvarovat se například chronické únavě v důsledku přetrénování. Uživatel má také možnost pozměnit si vstupní parametry tohoto modelu, a tak si více přizpůsobit model svým potřebám.

Výsledná aplikace je nyní ve stavu, kdy má vyřešený zdroj dat a také přihlašování uživatelů. Aplikace je tak v hodná pro rozsíření o další analýzy dat. Doplněn by mohl být například Bussoův model pro sledování výkonnosti a zohledněním tréninkové monotónosti. Dalším rozšířením by mohla být kalibrace vstupních parametrů modelů

pro sportovce na základě starších dat.

Literatura

[1] R-Project [online]. 1997, [cit. 11-05-2017].
Dostupné z: <https://www.r-project.org>

[2] OAuth 2.0 [online]. 2007, [cit. 23-03-2017].
Dostupné z: <https://oauth.net/2/>

[3] Strava Training Glossary for Running [online]. 2013, [cit. 11-05-2017].
Dostupné z: <https://support.strava.com/hc/en-us/articles/216917157-Strava-Training-Glossary-for-Running>

[4] RStudio [online]. 2016, [cit. 17-02-2017].
Dostupné z: <https://www.rstudio.com>

[5] Shiny Server [online]. 2016, [cit. 11-05-2017].
Dostupné z:
<https://www.rstudio.com/products/shiny/shiny-server/>

[6] AthleteMonitoring [online]. 2017, [cit. 11-05-2017].
Dostupné z: <http://www.athletemonitoring.com>

[7] FIT [online]. 2017, [cit. 23-03-2017].
Dostupné z: <http://wiki.openstreetmap.org/wiki/FIT>

[8] Garmin Connect [online]. 2017, [cit. 11-05-2017].
Dostupné z: <https://connect.garmin.com>

[9] GPX [online]. 2017, [cit. 23-03-2017].
Dostupné z: <http://www.topografix.com/gpx.asp>

[10] Polar Flow [online]. 2017, [cit. 17-02-2017].
Dostupné z: <https://flow.polar.com>

[11] Polar Personal Trainer [online]. 2017, [cit. 17-02-2017].
Dostupné z: <https://www.polarpersonaltrainer.com>

[12] shinyapps.io [online]. 2017, [cit. 11-05-2017].
Dostupné z: <http://www.shinyapps.io>

[13] Strava [online]. 2017, [cit. 11-05-2017].
Dostupné z: <https://www.strava.com/>

[14] Suunto Movescount [online]. 2017, [cit. 17-02-2017].
Dostupné z: <http://www.movescount.com/>

[15] TomTom MySport [online]. 2017, [cit. 17-02-2017].
Dostupné z: <https://mysports.tomtom.com>

[16] TrainingPeaks [online]. 2017, [cit. 11-05-2017].
Dostupné z: <https://www.trainingpeaks.com>

[17] Chang, W.; Cheng, J.; Allaire, J.; aj.: *shiny: Web Application Framework for R [software]*. 2017, [cit. 25-04-2017], r package version 1.0.2.
Dostupné z: <https://CRAN.R-project.org/package=shiny>

[18] Clarke, D. C.; Skiba, P. F.: Rationale and resources for teaching the mathematical modeling of athletic training and performance. *Advances in Physiology Education*, ročník 37, č. 2, 2013: s. 134–152, ISSN 1043-4046, doi:10.1152/advan.00078.2011,
<http://advan.physiology.org/content/37/2/134.full.pdf>.
Dostupné z: <http://advan.physiology.org/content/37/2/134>

[19] D. Richard Hipp, J. M., Dan Kennedy: *SQLite [software]*. 2000, [cit. 25-04-2017], relační databázový systém verze 3.17.0.
Dostupné z: <http://sqlite.org/download.html>

[20] Grothendieck, G.: *sqldf: Perform SQL Selects on R Data Frames [software]*. 2014, [cit. 25-04-2017], r package version 0.4-10.
Dostupné z: <https://CRAN.R-project.org/package=sqldf>

[21] Ooms, J.; Lang, D.; Hilaiel, L.: Package 'jsonlite' [online]. 2015.
Dostupné z: <https:
//stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html>

[22] Strava.com: *Strava's V3 API [software]*. V3 vydání, 2017, [cit. 23-03-2017].
Dostupné z: <https://strava.github.io/api/>

[23] from Trevor L Davis., A. D. C.: *getopt: C-like getopt behavior. [software]*. 2013, [cit. 25-04-2017], r package version 1.20.0.
Dostupné z: <https://CRAN.R-project.org/package=getopt>