Sustav za analizu i vizualizaciju sirovih podataka za praćenje aktivnosti D. melanogaster

Petrović, Milan

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike: Informacijski i komunikacijski sustavi

Milan Petrović

Sustav za analizu i vizualizaciju sirovih podataka za praćenje aktivnosti D. melanogaster

Diplomski rad

Rijeka, srpanj 2019.

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike: Informacijski i komunikacijski sustavi

Milan Petrović

Sustav za analizu i vizualizaciju sirovih podataka za praćenje aktivnosti D. melanogaster

Diplomski rad

Mentor: izv.prof.dr.sc. Ana Mešstrović
Komentor: dr.sc. Ana Filošević

Rijeka, srpanj 2019.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 3.6.2019.

Zadatak za diplomski rad

Pristupnik: Milan Petrović

Naziv diplomskog rada: Sustav za analizu i vizualizaciju sirovih podataka za praćenje aktivnosti D. melanogaster

Naziv diplomskog rada na eng. jeziku: Implementation of System for analysis and visualisation of row data for D. melanogaster Activity Monitoring

Sadržaj zadatka:

Zadatak diplomskog rada je modelirati i implementirati sustav za praćenje aktivnosti vinskih mušica u psrogramskom jeziku Python. Sustav treba korisniku nuditi grafičko sučelje u kojem je moguće odabrati različite oblike datoteka sa sirovim podacima, način na koji će se podaci analizirati (individualno ili na razini cijele populacije), te dodatne parametre kao što su vremensko razdoblje, raspon podataka i slično. Sustav kao rezultat treba vraćati statistički obrađene podatke i različite vizualizacije podataka. Nadalje, sustav treba omogućiti usporedbu rezultata za više različitih eksperimenata.

Mentor:

Izv. prof. dr. sc. Ana Meštrović

Voditelj za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Komentor:
Dr. sc. Ana Filošević

Zadatak preuzet: 3.6.2019.

(potpis pristupnika)

Sažetak

Drosophila melanogaster ili vinska mušica je životinjski model organizam koji se više od stoljeća koristi za biomedicinska istraživanja ovisnosti, spavanja, neurodegenerativnih bolesti i u razvoju lijekova. Jednostavan oblik ponašanja kao što je linearno kretanje moguće je kvantificirati uz pomoć komercijalno dostupnog sustava Drosophila Activity Monitoring System (DAMS). DAMS uz pomoć senzora bilježi broj prelazaka mušice na određenoj lokaciji unutar cjevčice. U ovisnosti o dužini vremenskog intervala, broju parametar i/ili broju jedinki u eksperimentu generiraju se velike matrice sirovih podataka koje nije moguće jednostavno analizirati upotrebom komercijalno dostupnih softverskih paketa. Potrebno je implementirati načela podatkovne znanosti (eng. data science) i bioinformatike te kreirati nova ili prilagoditi postojeća rješenja prema potrebama korisnika.

Cilj rada je kreirati korisničko sučlje (GUI od engl. graphical user interface) koje krajnjim korisnicima omogućava brz i jednostavan način učitavanja i obrade sirovih DAMS podataka. U radu je korišten programski jezik Python za obradu sirovih DAMS podataka tabličnog formata. Korištenjem Python biblioteke TkInter kreirano je sučelje za obradu i vizualizaciju podataka. Za obradu podataka stvoren je modul koji koristi biblioteku Pandas programskog jezika Python. Za vizualizaciju se koristi Python biblioteka Matplotlib. Korisniku će se omogućiti statistička obrada podataka i podjelu podataka na intervale. Osim toga moguće je pregledati rezultate u različitim vremenskim intervalima te ih grafički prikazati podataka u korisničkom sučelju.

Ključne riječi — **Drosophila melanogaster, DAMS sustav, bioinformatika, GUI, Python**

Sadržaj

Poglavlje 1

Uvod

U posljednja dva desetljeća bioinformatika doživljava ubrzan razvoj baziran na novim tehnologijama koje omogućuju prikupljanje i pohranu velikog skupa bioloških podataka. Fokus je na razvoju novih tehnologija i sustava koji ubrazvaju i olakšavaju obradu sve većeg broja podataka koji se stvaraju u istraživanjima [15]. Obrada podataka stvorenih praćenjem eksperimenata najčešće uzima više vremena nego sama provedba istih. Iz toga razloga potrebno je implementirati sustave koji ubrzavaju procese obrade i vizualizacije podataka, a jedan takav sustav opisan je u ovom radu. U ovom radu opisana je implementacija aplikacije koja omogućava krajnjim korisnicima bez prethodnog iskustva u programiranju da koriste i na jednostavan način interpretiraju sirove tablične podatke praćenja aktivnosti vinskih mušica prije i nakon administracije psihostimulansa. U aplikaciji je omogućena selekcija tabličnih podataka, transformiranje i pohranu u nove željene oblike. Aplikacija nudi vizualizaciju podataka unutar različitih vremenskih intervala i usporedbu rezultata više eksperimenata. Nad odabranim podatcima provode se statistički izračuni (prosjek, standardna devijacija) i vizualizacija. Rezultati obrade korisniku se prikazuju u sučelju te mu se nudi izbor prikaza podataka ovisno o korisnikovim potrebama. Ovime se znatno smanjuje vrijeme obrade podataka čime korisnici mogu više vremena posvetiti provodenju eksperimenata i analizi rezultata što je glavni cilj ovog diplomskog rada.
U idućem poglavlju prikazan je povijesni razvoj programskog jezika Python, objektno orijentirano programiranje u Pythonu i način na koji se koristi zatim biblioteke i njihov način korištenja. U trećem poglavlju prikazan je način stvaranja podataka koji

se obrađuju. Objašnjen je format stvorenih podataka i način interpretacije, zahtjevi za analizu nad podatcima koje je potrebno izvršiti na individualnoj i populacijskoj razini i stvaranja biblioteke koja to omogućava. U četvrtom poglavlju definira se arhitektura programa, objašnjen je način na koji program radi i prikaz rada u korisničkom sučelju. U petom poglavlju prikazana je implementacija sustava nad podatcima. Objašnjeno je na koji način se provodi i izvršava individualna i populacijska analiza aktivnosti vinskih mušica prilikom administracije psihostimulansa, zatim populacijska analiza mučica aktivnosti po satu. U zadnjem djelu poglavlja objašnjen je način usporedbe individualnih rezultata analize iz više eksperimenata. U zaključnom poglavlju opisani su zaključci i navedene moguće smjernice daljnjeg razvoja i istraživanja.

Poglavlje 2

Programski jezik Python

2.1 O Pythonu

Python je programski jezik visoke razine razvijen za opće namjene. Autor programskog jezika Python je Guido van Rossum. Programski jezik Python koncipiran je krajem 1980-ih godina kao nasljednik ABC programskog jezika sposobnog za rukovanje iznimkama i povezivanje s operativnim sustavom Amoeba. Jezici koji su utjecali na razvoj programskoga jezika Python, a iz kojih su posuđeni neki principi su: Modula-3, Lisp, Perl, Haskell, Java, C, C++. Stilovi pisanja unutar programskog jezika Python su objektno orijentirano, strukturno i aspektno orijentirano. Zbog velike fleksibilnosti treći je programski jezik po zastupljenosti [1].
Godine 1991 izlazi Python 0.9.0, prva javno dostupna verzija [2], dok u siječnju 1994. izlazi verzija 1.0, uz nove značajke kao što su funkcionalni alati za programiranje lambda, map, filter i reduce. Python 2.0 izlazi 16. listopada 2000. s mnogim novim značajkama koje uključuju automatsko upravljanje memorijom i podršku za Unicode. Nova značajka bila je i upravljanje listama (engl. list comprehention) posuđena iz funkcionalnih programskih jezika SETL i Haskell. Glavna promjena je bila prelazak na transparentan proces razvoja i razvoj podržan od strane korisničke razvojne zajednice. Python 3.0 izlazi 3. prosinca 2008. Ova verzija osmišljena je da ukloni osnovne nedostatke u dizajnu jezika. Potrebne promijene nije bilo moguće implementirati i pritom zadržatiu potpunu kompatibilnost s 2.x verzijama jezik stoga

se prelazi na veziju 3.0.
Struktura kôda zasniva se na korištenju metoda uvlačenja(tabova) za razlikovanje programskih blokova. Programiranje u Pythonu vodi se jednostavnošću i čitljivosti kôda. Ovim stilom pisanja cilj je omogućiti programerima pisanje čistog i jednostavnog kôda neovisno o veličini projekta. Cijela filozofija pisanja programskog kôda u Pythonu najbolje se može vidjeti iz 19 smjernica koje se mogu vidjeti ako u program unesemo naredbu `import this` [3]. Nakon pokretanja programa na ekran se ispisuju smjernice. Neke koje najbolje opisuju stil pisanja programskog kôda su:

- beautiful is better than ugly,
- explicit is better than implicit,
- simple is better than complex,
- flat is better than nested,
- readability counts,
- if the implementation is hard to explain, it's a bad idea,
- if the implementation is easy to explain, it may be a good idea.

Inspiraciju za naziv Guido van Rossum dobio je čitajući skripte britanskoga serijala komedije Monty Python Flying Circus. Van Rossum je smatrao da sam jezik mora imati naziv koji je kratak, jednostavan i pomalo tajanstven, iz toga razloga nazvao ga je Python [4].

2.2 Objektno orijentirano programiranje u Pythonu

Prilikom pisanja programa koji su u svojoj strukturi složeni, sadrže više cjelina koje se pokreću u program, javlja se problem organizacije kôda. Ovaj problem može se pojednostaviti da se kôd strukturira u objekte koji međusobno komuniciraju i izvršavaju naredbe. Ovakav stil strukturiranja kôda naziva se paradigma objektno orijentiranog programiranja. Klase i objekti su dva glavna elementa objektno orijentiranog programiranja. Klasa stvara novi tip u kojem su objekti instance te klase [5].

Pohrana podataka u objektu ostvaruje se pomoću običnih varijabli koje pripadaju tom objektu. Objekti mogu koristiti i funkcije koje pripadaju klasi. Takve funkcije nazivamo metodama klase. U klasama razlikujemo dva tipa varijabli, varijable instance i varijable klase. Klasu stvaramo korištenjem ključne riječi `class`, u nastavku je prikazan jednostavan primjer stvaranja klase u Pythonu.

```
class Student:
def __init__(self, name, age):
self.name = name
self.age = age
p1 = Student("Mark", 20)
```

Ispis kôda 2.1 Primjer klase u programskom jeziku Python

U izrazu `p1 = Student("Mark", 20)` kreira se instanca klase Student tako da se prvo upisuje naziv klase, u ovom slučaju riječju `Student`, te zatim klasi prosljeđujemo vrijednosti imena i godine. Unutar same klase vrijednosti se pohranjuju u tu instancu pomoću ključne riječi `self`. Primijetimo u kôdu unutar klase nalazi se metoda `__initi__`. Ova metoda pokreće se čim se objekt klase instancira (stvori). Ova metoda korisna je za bilo koju inicijalizaciju, tj. prosljeđivanje početnih vrijednosti klase objektu. Kao što je navedeno klase mogu sadržavati i metode, metode su zapravo funkcije koje sadrže ključnu riječ `self`. U nastavku je prikazan primjer kôda metode unutar klase.

```
class SomeClass:
def greetings(self):
print('Hello stranger, Good to see you!')
p = SomeClass().greetings()

Output:
'Hello stranger, Good to see you!
```

Ispis kôda 2.2 Primjer metode unutar klase u programskom jeziku Python

2.3 Biblioteke i njihovo korištenje

Biblioteke u Pythonu sadrže skup funkcija i metoda koje omogućuju obavljanje složenih radnji. Radnje se ostvaruju pomoću poziva funkcija čiji je kôd već prethodno napisan . Biblioteke sadrže velik broj korisnih modula koji se mogu pozivati u samom programu. Python u svom osnovnom obliku sadrži osnovnu biblioteku koja je sama po sebi opsežna zbirka dobro dokumentiranih funkcija za svakodnevno programiranje. Osim osnovne biblioteke korisnici su razvili i druge koje se odnose na izvršenje posebnih zahtjeva u programu. Moduli u Pythonu mogu se podijeliti u tri vrste: stvoreni od strane korisnika, stvoreni od vanjskih izvora i prethodno instalirani koji dolaze s osnovnom verzijom Pythona. Python Package Index ili skraćeno PyPI, poznat i po nadimku Cheese Shop (iz Monty Pythonovog skeča Cheese Shop) službeni je repozitorij za pakete stvorene od treće strane za Python. Trenutno postoji preko 180 000 Python biblioteka kojima se može pristupiti putem PyPI-a [6].

2.3.1 Biblioteka TkInter

TkInter je Pythonov standardni GUI (Graphical User Interface) paket [7]. TkInter je uključen u standardni instalacijski paket Pythona. Ime TkInter dolazi od izvedenice iz riječi Tk i Interface. Autori TkInter biblioteke i osobe zaslužne za razvoj su Fredrik Lundh, Steen Lumholt i Guido van Rossum [8]. Terminologije koja se najčešće spominju prilikom rada s TkInterom su:

- window (prozor) iako sam pojam ima više značenja ovisno o kontekstu, najčešće se odnosi na kvadratni prikaz na korisnikovom zaslonu,
- widget (grafički element) pojam za bilo koji od blokova koji čine aplikaciju u korisničkom sučelju. Widgete možemo podijeliti ovisno o njihovoj ulozi u programu (tekstualni, gumb, oznaka, poruka... i slično),
- frame (okvir) Osnovna jedinica prilikom organizacije složenijih sučelja. Okvir je pravokutni prostor koji sadrži ostale grafičke elemente (widgete).

2.3.2 Biblioteka Pandas

Pandas je biblioteka za manipulaciju i analizu podataka. Omogućava korisniku strukturiranje podataka u vremenskim serijama ako vremenska varijabla postoji u podatcima. Naziv ove biblioteke izveden je iz riječi 'panel' i 'data' [9]. Autor ove biblioteke je Wes McKinney [9]. Za vrijeme rada u AQR Capital Managementu javila se potreba za fleksibilnim alatom visokih performansi za kvantitativnu analizu financijskih podataka. Iz toga razloga Wes McKinney 2008. započinje razvijati biblioteku pandas koja bi to omogućavala. Prije samog napuštanja AQR-a McKinney uvjerava upravu da se pandas objavi kao biblioteka otvorenog kôda. Pandas nudi razne funkcije koje omogućavaju jednostavan rad s velikim skupovima podataka. Neke od tih funkcija su:

- pohrana podataka u tablične strukture koje se u pandas biblioteci nazivaju DataFrame. Ovi objekti omogućavaju daljnju manipulaciju nad istima,
- pukovanje sa zapisima koji nedostaju u podatcima,
- promjena indeksa u podatcima, dodjeljivanje indeksa nad određenim stupcem jedinstvenih podataka,
- umetanje i brisanje stupaca i redaka,
- vizualizacija podataka pomoću vremena,
- filtriranje podataka,
- pozivanje funkcija za izračun nad podatcima te pohrana vrijednosti u nove stupce.

2.3.3 Biblioteka Matplotlib

Matplotlib je biblioteka za vizualizaciju podataka [10]. Pomoću API-a omogućava ugrađivanje vizualizacija u aplikacije koje koriste korisničko sučelje (Graphical User Interface (GUI)) kao što su TkInter, Qt ili GTK+ [10]. Matplotlib izvorno je napisao John D. Hunter, Matplotlib ima aktivnu zajednicu koja održava i unaprjeđuje ovu biblioteku. Prije smrti Johna D. Huntera, Michael Droettboom nominiran je za vodećeg razvojnog programera Matplotliba. Matplotlib 1.2 je prva verzija koja

podržava Python 3.x te će u skorije vrijeme prebaciti se potpuno na Python 3.

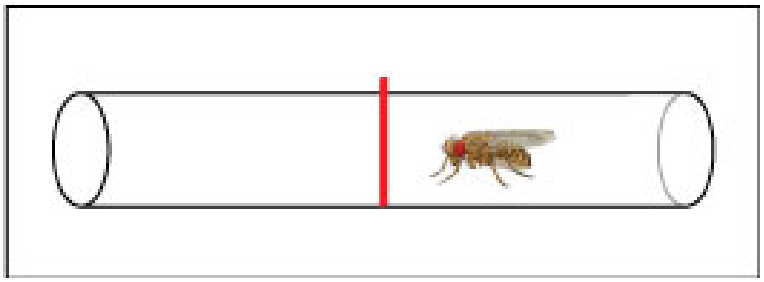
Poglavlje 3

Podatci

3.1 Praćenje aktivnosti D. melanogaster

Vinska mušica ili latinski Drosophila melanogaster je kukac dvokrilac najčešće korišten za biomedicinska istraživanja [11]. Drosophila se koristi u istraživanjima zbog lakog uzgoja, brzog razmnožavanja i stvaranja velikog broja potomaka u kratkom vremenu. Vrijeme nastanka nove generacije potomaka je 14 dana što čini prednost prilikom istraživanja gena i nasljednih svojstava nad drugim model organizmima kod kojih je razvojni ciklus do nekoliko mjeseci [12]. Glavni razlog popularnosti vinske mušice u biomediciskim istraživanjima je jeftin uzgoj, kratak razvojni ciklus i visoka homologija s ljudskim genomom. Radi toga koristi se za istraživanja u području genetike, fiziologije i evolucije. Na temelju lineranog gibanja mušica moguće je pratiti utjecaj vrste testirane supstance, njezine doze i vrste administracije na aktivnost mušice. Praćenjenje aktivnosti vrši se pomoću Drosophila Activity Monitor (DAM) sustava. DAM sustav bilježi broj prelazaka sredine staklene cjevčice u kojoj se nalazi po jedna jedinka putem senzora koji bilježi prekide infracrvene zrake iz izvora. DAM sustav ima 32 kanala za praćenje lokomotornih aktivnosti 32 jedinke pomoću softvera za prikupljanje podataka [13]. Infracrvena zraka postavljena u središte svake cijevčice mjeri broj prijelaza koje je napravila pojedinačna mušica u intervalu od jedne minute, te se bilježi kao broj prekida zrake ili prelazaka po minuti (Pfeiffenberger et al. 2010). Na Slici 3.1 može se vidjeti kako izgleda mušica u DAM1 sustavu i način praćenja njene

aktivnosti pomoću infracrvene zrake. Brojevi lokomotornih aktivnosti se prikupljaju



Slika 3.1 D.melanogaster za vrijeme boravka u DAM1 sustavu

(Tablica) na računalu pomoću PSIU9 jedinice napajanja (TriKinetics). Koristeći DAMFileScan softver, pohranjeni podaci se izvlače na temelju zadanih parametara (Tablica 3.1). Sirovi podaci u obliku .txt datoteka koje je prikupio DAM sustav obrađuju se i analiziraju na populacijskoj ili individualnoj razini. Sustav prikuplja podatke 30 minuta prije i 30 minuta nakon administracije volatilih psihotimulansa populaciji od 32 mušice. Obzirom na dozu, vrijeme izlaganja i vrstu psihostimulansa postoji razlika u lokomociji mušica prije i nakon admistracije psihostimulansa.

Tablica 3.1 Primjer zapisivanja i praćenja vremenskih perioda.

vrsta postupka	period prikupljanja podataka	proteklo vrijeme (min)
stanje mirovanja	08:22-08:51	30
nakon prvog unosa	09:01-09:30	30
stanje mirovanja	14:22-14:51	30
nakon drugog unosa	15:01-15:30	30

3.2 Analiza podataka

3.2.1 Populacijska analiza podataka

Za analizu podataka temeljenih na populaciji, sirovi podaci analizirani su kao prosječna količina aktivnosti 32 mušice u jednoj minuti [14]. Aktivnost je potrebno prikazati linijskim grafikonom prosječne količine aktivnosti u minuti, iz kojeg se mogu izvesti zaključci o kinetici perturbacije koju inducira primjena psihostimulansa. Ovi grafikoni daju informacije o snazi i trajanju utjecaja psihostimulansa na aktivnost kretanja populacije.

3.2.2 Individualna analiza podataka

Individualni odgovor mušice na psihostimulans dobiva se usporedbom prosječne aktivnosti u minuti pojedinačne mušice, prije i nakon primjene psihostimulansa (prvi i / ili drugi) [14]. Iz tog skupa podataka može se objektivno izračunati nekoliko svojstava ponašanja na individualnoj razini (Tablica 3.2).

Osjetljivost (SENS) izračunata je kao broj ili postotak mušica, koje povećavaju svoju aktivnost kada je bazna lokomotorna aktivnost (B) uspoređena s aktivnošću nakon prve primjene psihostimulansa (A1). Pojedinačni podaci o mušici izračunati su kao prosjek tijekom 5 minuta prije i nakon primjene psihostimulansa. Aktivnost je zatim kategorizirana kao "ista", "smanjena" ili "povećana". Mušice koje povećavaju svoju aktivnost predstavljaju podskupinu populacije koja odgovara na psihostimulans. Ista usporedba može se izvesti u razlici ponašanja nakon prve i druge administracije psihostimulansa, s tim da oni koji odgovaraju na tu drugu dozu ne moraju uključivati one koji su odgovorili povećanom aktivnošću na prvu dozu. Stoga, usporedba između prve i druge izloženosti samo ukazuje na osjetljivost na drugu dozu [14]. Kako bi se izračunao broj ili postotak mušica koje razvijaju lokomotornu senzitizaciju (LS) pri ponovljenoj primjeni psihostimulansa, mušice koje pokazuju uzlazni porast lokomotorne aktivnosti: prije (B) $<$ nakon 1. (A1) $<$ nakon 2. (A2). Ti su se kriteriji koristili za izračunavanje Lokomotorna senzitizacija (LS) u pojedinim mušicama koje su primile više od dvije doze psihostimulansa.

Tablica 3.2 Kriteriji izračuna SENS i LS mušica u individualnoj analizi.

vremenski interval		5 min prosjek
broj mušica		1
SENS	kriterij	B <A1
	način računanja	$SENS = \frac{\text{broj SENS mušica}}{\text{ukupni broj mušica}}$
LS	kriterij	B <A1 <A2
	način računanja	$LS = \frac{\text{broj LS mušica}}{\text{ukupni broj mušica}}$

3.3 Obrada podataka

Podatci se prikupljaju i pohranjuju na računalo pomoću DAMFileScan softver i pohranjuju se u .txt formatu. U prvom stupcu nalazi se jedinstveni broj retka. Vrijeme je pohranjeno u drugom i trećem stupcu, datum u formatu "DD MM YY" i vrijeme "HH:MM:SS". Aktivnosti mušica slijedi nakon vremenskih stupaca i proteže se do kraja unosa. Iako se prati maksimalno aktivnost 32 mušice program ponekad stvara stupce koji ne sadržavaju informacije, tako ukupan broj rezultirajućih stupaca zna biti oko 40. U nastavku je prikazan izgled sirovih podataka koji se pohranjuju na računalo.

```
1 80348 26 Apr 16 09:00:00  1 0 0 0 0 0 1 1 0 0 0 5 0 0 0...
2 80349 26 Apr 16 09:01:00  1 0 0 0 0 0 1 0 0 0 0 5 0 0 0...
3 80350 26 Apr 16 09:02:00  1 0 0 0 0 0 1 0 0 0 0 0 0 0 0...
4 80351 26 Apr 16 09:03:00  1 0 0 0 0 0 1 1 0 0 0 0 0 0 0...
5 80352 26 Apr 16 09:04:00  1 0 0 0 0 0 1 0 0 1 0 0 0 0 0...
```

Ispis kôda 3.1 Prikaz sirovih podataka učitanih iz .txt datoteke

Transformacije koje je potrebno izvesti nad podatcima:

- brisanje prvoga stupca (jedinstvena oznaka retka),
- spajanje stupaca datuma i vremena, pritom promjena formata datuma u "YYYY-DD-MM". Rezultirajući stupac je formata "YYYY-DD-MM HH:MM:SS",

- mogućnost selekcije stupaca koji sadržavaju aktivnosti mušica,
- stvaranje stupaca s vrijednostima populacije (sum, mean, std).

3.4 Kreiranje modula za obradu podataka

Zbog potrebe izvršavanja više vrsta obrada na podatcima kreirana je klasa `DataEdit_`. U ovoj klasi deklarirane su metode koje obrađuju podatke. Metode su podijeljene na dvije vrste ovisno o njihovoj namjeni u programu (pomoćne i glavne). Pomoćne metode vraćaju i prosljeđuju obrađene podatke drugim metodama. A glavne metode se samo pozivaju i prilikom poziva izvode transformaciju podataka i pohranjuju ih u nove DataFrame-ove koji se kasnije koriste za vizualizaciju ili pohranu u programu. Razlog ovoj strukturi metoda je potreba za čestim zahtjevima istog oblika transformacije podataka, primjerice odabir stupaca i redaka. U nastavku je prikazan primjer metode za odabir stupaca.

```
def select_col(df, start):
start_time = datetime.strptime(start, '%H:%M')
end_time = start_time + timedelta(minutes=30)

df['datetime'] = pd.to_datetime(df['datetime'])
df = df.loc[df['datetime'].dt.time.between(start_time.time(),
    end_time.time())]

df = df.reset_index(drop=True)

return df
```

Ispis kôda 3.2 Pomoćna metoda za odabir stupaca

Prikaz pozivanja metode iz klase `select_col()` u metodi `split_for_graph()`.

```
def split_for_graph(self, start_bsl_morning, start_1st_expo,
start_bsl_noon, start_2nd_expo):
df = self.controller.df.copy()
df_1 = DataEdit_.select_col(df, start_bsl_morning)
```

```
df_2 = DataEdit_.select_col(df, start_1st_expo)
df_3 = DataEdit_.select_col(df, start_bsl_noon)
df_4 = DataEdit_.select_col(df, start_2nd_expo)
```

Ispis kôda 3.3 Primjer korištenja metode select_col u metodi split_for_graph()

Spajanje stupaca datuma i vremena, pritom promjena formata datuma u "YYYY-DD-MM". Rezultirajući stupac je formata "YYYY-DD-MM HH:MM:SS".

```
df_info['datetime'] = df.date + ' ' + df.time
```

Ispis kôda 3.4 Spajanje stupaca "date" i "time" i pohrana u novi stupac "datetime"

Stvaranje stupaca s vrijednostima populacije (sum, mean, std).

```
df['mean'] = df.iloc[:,7:].mean(axis=1)
df['std'] = df.iloc[:,7:].std(axis=1)
df['sum'] = df.iloc[:,7:].sum(axis=1)
```

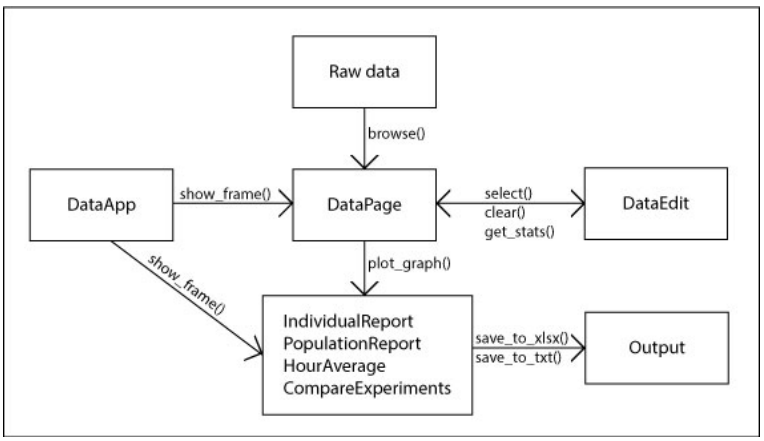
Ispis kôda 3.5 Računanje vrijednosti mean sum i std i pohrana u pripadajuće stupce.

Poglavlje 4

Definiranje sustava za obradu podataka

4.1 Arhitektura programa

Rad programa ostvaren je na principu da klasa `DataApp` prilikom pokretanja program na korisnikovom ekranu prikazuje `DataPage` klasu. Korisnik učitava podatke pritiskom na gumb `browse file` koji aktivira funkciju `browse()`. Ova klasa poziva određene metode iz klase `DataEdit` koje vrše obradu i vraćaju obrađene podatke. Uvid u rezultate obrade korisniku su omogućeni odabirom gumba `Individual Report`, `Population Report` i `Hour Average` koji se prikazuju pozivom funkcije `show_frame()`. Korisnik pohranjuje podatke pritiskom na gumbe `Save to XLSX` i `Save to txt` koji pozivaju za to pripadajuće funkcije `save_to_xlsx()` i `save_to_txt()` ovisno o formatu u kojem se podatci žele pohraniti.



Slika 4.1 Skica organizacije programa

4.2 Osnovni dio

Pokretanje rada programa ostvareno je pozivom funkcije `main`. Pozivom funkcije `main` pokreće se rad klase `DataApp_()` kojoj je zadatak prikaz programskih okvira(frame).

```
from DataApp import DataApp_

def main():
app = DataApp_()
app.mainloop()

if __name__ == '__main__':
main()
```

Ispis kôda 4.1 Funkcija main

U klasi `DataApp_()` potrebno je deklarirati sve okrive koje želimo pozivati na ekran prilikom rada programa. Pomoću `for` petlje sadržaj okvira se učitava u varijablu `frame` naredbom `frame = F(container, self)`, potom se sadržaj pohranjuje u listu kojoj se pristupa prilikom rada programa `self.frames[F] = frame`. Osnovni okvir koji se prikazuje prilikom pokretanja programa je `DataPage_` klasa, naredbom `self.show_frame(DataPage_)`.

```
self.frames = {}
for F in (DataPage_, GraphPage_, IndividualReport_,
    CompareExperiments_, HourAve_):
frame = F(container, self)
self.frames[F] = frame
frame.grid(row=0, column=0, sticky='nsew')

self.show_frame(DataPage_)
```

Ispis kôda 4.2 Prikaz okvira u klasi DataApp_

Prikaz na korisnikovom ekranu ostvaren je funkcijom `show_frame()`. Ova funkcija učitava sadržaj koji je potreban prikazati i naredbom `frame.tkraise()` ga prikazuje na ekranu.

```
def show_frame(self, cont):
frame = self.frames[cont]
frame.tkraise()
```

Ispis kôda 4.3 Funkcija za prikaz okvira

4.2.1 Programski okviri (eng.frames)

Grafički element okvira(Frame Widget) vrlo je važan za proces grupiranja i organiziranja drugih elemenata(widgeta) sučelja na nekako prijateljski način. Djeluje kao spremnik, koji je odgovoran za uređenje položaja drugih grafičkih elemenata. Koristi pravokutna područja na zaslonu za organiziranje izgleda i za pružanje popunjavanja istih. Okvir se također može koristiti kao temeljni razred za implementaciju složenih elemenata sučelja. Okviri su vrlo važni za proces grupiranja i organiziranja drugih grafičkih elemenata na nekako prijateljski način. Djeluje kao spremnik, koji je odgovoran za uređenje položaja drugih grafičkih elemenata.
Okviri se prikazuju na zaslonu ovisno o odabiru korisnika. Osnovni okvir koji se prikazuje kao početni prilikom pokretanja programa je okvir iz klase `DataPage_`. U nastavku je prikazan osnovni dio programskog kôda koji to omogućuje.

```
class DataPage_(tk.Frame):
```

```
def __init__(self, parent, controller):
tk.Frame.__init__(self, parent)
```

Ispis kôda 4.4 Primjer programskog okvira(framea) u biblioteci TkInter

4.3 Implementacija korisničkog sučelja

Za prikaz oznaka u sučelju koristi se `Label` naredba (Slika 4.2). Ovaj grafički element implementira okvir za prikaz gdje se može prikazati tekst ili slika. Tekst koji se prikazuje može se ažurirati u bilo kojemu trenutku rada programa.

```
tk.Label(self, text='Data edit').grid(pady=10,padx=10)
```

Ispis kôda 4.5 Primjer grafičkog elementa za oznaku

Slika 4.2 Prikaz grafičkog elementa za prikaz u korisničkom sučelju

Za dodavanje gumba u aplikaciji koristi se ključna riječ `Button`. Na gumbima može biti prikazan tekst ili slika koji govore ulogu gumba (Slika 4.3). Na gumb se može dodati funkcija ili metoda koja se automatski poziva pritiskom gumba. U nastavku je prikazan primjer kôda za gumb koji poziva funkciju `save_to_xlsx()`.

```
ttk.Button(self, text='Save XLSX', command=lambda: DataEdit_.
    save_to_xlsx(self.controller.df)).grid()
```

Ispis kôda 4.6 Kôd koji će generirati gumb za poziv funkcije save_to_xlsx()

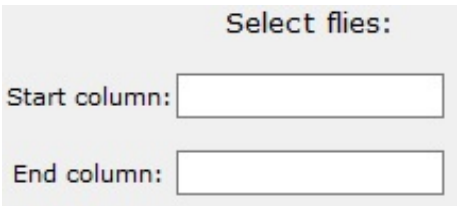
Unos od strane korisnika u sučelju ostvaren je pomoću grafičkog elementa `Entry`. Ovaj grafički element koristi se za unos tekstualnih nizova duljine jednog retka od strane korisnika. U nastavku je prikazan način unosa početnog i krajnjeg stupca u podatcima koji se obrađuju. Prvo je potrebno deklarirati varijablu za pohranu

Slika 4.3 Prikaz grafičkog elementa za gumb u korisničkom sučelju

unesene vrijednosti, potom se pomoću elementa `Entry` ostvaruje unos (Slika 4.4). Nakon unosa i pohrane u varijablu u programu se može koristiti unesena vrijednost.

```
1 start_col = tk.StringVar()
2 end_col = tk.StringVar()
3
4 self.start_col = ttk.Entry(self, textvariable=start_col).grid()
5 self.end_col = ttk.Entry(self, textvariable=end_col).grid()
```

Ispis kôda 4.7 Primjer kôda za korištenje grafičkog elementa Entry



Slika 4.4 Prikaz grafičkog elementa za unos u korisničkom sučelju

Grafički prikaz više od jedne linije teksta ostvaren je grafičkim elementom `Text` (Slika 4.5). Tekstualni grafički elementi pružaju napredne mogućnosti uređivanja i formiranja višelinijskoga teksta. Promjena boje teksta, format i veličina fonta također je moguća. Prikaz sadržaja proširen je na više od jednoga stupca i retka opcijom `grid(row=2, column=4, columnspan=14, rowspan=14)`. Naredbom `insert` omogućeno je ažuriranje prikazanoga teksta. Prilikom rada programa ako se događa novi odabir podataka rezultat odabira prikazan je u na korisničkom ekranu. Prikaz podataka u jednom redu, odnosno prijelom prikaza (page break) ostvaren je naredbom `to_string()`. U nastavku je primjer prikaza učitanih podataka iz DataFrame-a

i njihov prikaz pomoću grafičkog elementa `Text`.

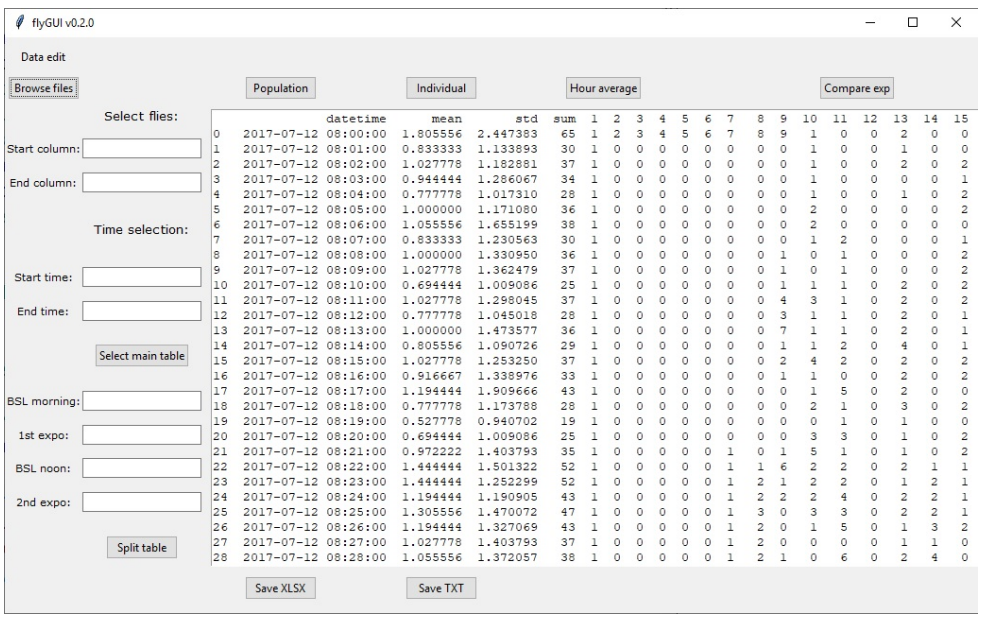
```
T = tk.Text(self, height=30, width=200, wrap=None)
T.grid(row=2, column=4, columnspan=14, rowspan=14, padx=10, pady=10)
T.insert(tk.END, self.controller.df.to_string())
```

Ispis kôda 4.8 Kôd koji će generirati grafički element Text

	datetime	mean	std	sum	1	2	3	4	5	6	7	8	9
0	2016-08-02 08:00:00	0.472222	0.774084	17	1	0	0	0	0	0	1	0	2
1	2016-08-02 08:01:00	0.583333	0.967323	21	1	0	0	0	0	0	1	1	2
2	2016-08-02 08:02:00	0.666667	1.014185	24	1	0	0	0	0	0	1	0	1
3	2016-08-02 08:03:00	1.166667	1.748469	42	1	0	0	0	0	0	1	0	3
4	2016-08-02 08:04:00	0.694444	0.980363	25	1	0	0	0	0	0	1	1	1
5	2016-08-02 08:05:00	0.916667	1.401530	33	1	0	0	0	0	0	1	0	2
6	2016-08-02 08:06:00	0.833333	1.444200	30	1	0	0	0	0	0	1	0	0
7	2016-08-02 08:07:00	0.611111	1.248491	22	1	0	0	0	0	0	1	0	2
8	2016-08-02 08:08:00	0.833333	1.424279	30	1	0	0	0	0	0	1	0	1
9	2016-08-02 08:09:00	0.750000	1.024695	27	1	0	0	0	0	0	1	2	1
10	2016-08-02 08:10:00	0.777778	1.197882	28	1	0	0	0	0	0	1	2	2

Slika 4.5 Primjer grafičkog elementa za prikaz tekstualnih podataka u korisničkom sučelju

Korisničko sučelje za unos i prikaz podataka nad kojima je moguće izvršiti odabir stupaca i redaka prikazano je na Slika 4.6. Elementi za prikaz deklarirani su u klasi `DataPage_` a metode za obradu podataka u pozivaju se iz klase `dataEdit_`.



	datetime	mean	std	sum	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2017-07-12 08:00:00	1.805556	2.447383	65	1	2	3	4	5	6	7	8	9	1	0	0	2	0	0
1	2017-07-12 08:01:00	0.833333	1.133893	30	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0
2	2017-07-12 08:02:00	1.027778	1.182881	37	1	0	0	0	0	0	0	0	0	1	0	0	2	0	2
3	2017-07-12 08:03:00	0.944444	1.286067	34	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
4	2017-07-12 08:04:00	0.777778	1.017310	28	1	0	0	0	0	0	0	0	0	1	0	0	1	0	2
5	2017-07-12 08:05:00	1.000000	1.171080	36	1	0	0	0	0	0	0	0	0	2	0	0	0	0	2
6	2017-07-12 08:06:00	1.055556	1.655199	38	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0
7	2017-07-12 08:07:00	0.833333	1.230563	30	1	0	0	0	0	0	0	0	0	1	2	0	0	0	1
8	2017-07-12 08:08:00	1.000000	1.330950	36	1	0	0	0	0	0	0	0	1	0	1	0	0	0	2
9	2017-07-12 08:09:00	1.027778	1.362479	37	1	0	0	0	0	0	0	0	1	0	1	0	0	0	2
10	2017-07-12 08:10:00	0.694444	1.009086	25	1	0	0	0	0	0	0	0	1	1	1	0	2	0	2
11	2017-07-12 08:11:00	1.027778	1.298045	37	1	0	0	0	0	0	0	0	4	3	1	0	2	0	2
12	2017-07-12 08:12:00	0.777778	1.045018	28	1	0	0	0	0	0	0	0	3	1	1	0	2	0	1
13	2017-07-12 08:13:00	1.000000	1.473577	36	1	0	0	0	0	0	0	0	7	1	1	0	2	0	1
14	2017-07-12 08:14:00	0.805556	1.090726	29	1	0	0	0	0	0	0	0	1	1	2	0	4	0	1
15	2017-07-12 08:15:00	1.027778	1.253250	37	1	0	0	0	0	0	0	0	2	4	2	0	2	0	2
16	2017-07-12 08:16:00	0.916667	1.338976	33	1	0	0	0	0	0	0	0	1	1	0	0	2	0	2
17	2017-07-12 08:17:00	1.194444	1.909666	43	1	0	0	0	0	0	0	0	0	1	5	0	2	0	0
18	2017-07-12 08:18:00	0.777778	1.173788	28	1	0	0	0	0	0	0	0	0	2	1	0	3	0	2
19	2017-07-12 08:19:00	0.527778	0.940702	19	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0
20	2017-07-12 08:20:00	0.694444	1.009086	25	1	0	0	0	0	0	0	0	0	3	3	0	1	0	2
21	2017-07-12 08:21:00	0.972222	1.403793	35	1	0	0	0	0	0	1	0	1	5	1	0	1	0	2
22	2017-07-12 08:22:00	1.444444	1.501322	52	1	0	0	0	0	0	1	1	6	2	2	0	2	1	1
23	2017-07-12 08:23:00	1.444444	1.252299	52	1	0	0	0	0	0	1	2	1	2	2	0	1	2	1
24	2017-07-12 08:24:00	1.194444	1.190905	43	1	0	0	0	0	0	1	2	2	2	4	0	2	2	1
25	2017-07-12 08:25:00	1.305556	1.470072	47	1	0	0	0	0	0	1	3	0	3	3	0	2	2	1
26	2017-07-12 08:26:00	1.194444	1.327069	43	1	0	0	0	0	0	1	2	0	1	5	0	1	3	2
27	2017-07-12 08:27:00	1.027778	1.403793	37	1	0	0	0	0	0	1	2	0	0	0	0	1	1	0
28	2017-07-12 08:28:00	1.055556	1.372057	38	1	0	0	0	0	0	1	2	1	0	6	0	2	4	0

Slika 4.6 Prikaz rada programa prilikom odabira nad podatcima u korisničkom sučelju

Poglavlje 5

Implementacija modula za obradu i vizualizaciju podataka

U ovom poglavlju prikazan je praktični primjer korištenja sustava i njegov rad nad stvarnim podatcima. U prvom djelu prikazana je obrada i vizualizacija osnovnog prikaza aktivnosti populacije za 4 vremenska perioda (mirovanje prije prve administracije, nakon prvog unosa, mirovanje prije druge administracije, nakon drugog unosa). U drugom djelu prikazana je obrada i vizualizacija podataka prosječne aktivnosti populacije po satu. U zadnjem djelu prikazan je način međusobne usporedbe rezultata individualne analize više eksperimenata.

5.1 Vizualizacija i obrada podataka individualne i populacijske aktivnosti po intervalima

5.1.1 Obrada podataka aktivnosti populacije po intervalima

Razdvajanje podataka na intervale ostvareno je funkcijom `split_for_graph()` ova funkcija kao parametre prima vrijednosti unesene pomoću grafičkog elementa za unos. Same vrijednosti dobavljaju se metodom `get()`. Vrijednosti se prosljeđuju funkciji i vrši se razdvajanje podataka na 4 manja podatkovna okvira koji će se naknadno vizualizirati. U nastavku je prikazana deklaracija funkcije `split_for_graph()`.

```
ttk.Button(self, text='Split table',command=lambda: DataEdit_.
    split_for_graph(self,
start_bsl_morning.get(),
start_1st_expo.get(),
start_bsl_noon.get(),
start_2nd_expo.get())).grid()
```

Razdvajanje podataka ostvareno je tako da se prvo podatci iz glavnog okvira kopiraju, iz razloga očuvanja početnih podataka. Potom se funkcijom `select_col()` ostvaruje odabir željenog vremenskog perioda. Funkcija `split_for_graph()` stvara 4 manja okvira i to za vrijednosti prije uzimanja psihostimulansa, nakon prvog uzimanja, prije drugoga uzimanja i nakon drugoga uzimanja. U nastavku je prikazan primjer odabira podataka za razdoblje prije prvoga uzimanja psihostimulansa.

```
df = self.controller.df.copy()
df_1 = DataEdit_.select_col(df, start_bsl_morning)
self.controller.df_1 = df_1
```

Za odabir podataka koristi se pomoćna metoda `"select_col()"` koja kao parametre prima varijable `df` i `start`. Period promatranja aktivnosti mušica iznosi 30 minuta iz tog razloga vrijednost `end_time` zadaje se na način `end_time = start_time + timedelta(minutes=30)`.

```
def select_col(df, start):
start_time = datetime.strptime(start, '%H:%M')
end_time = start_time + timedelta(minutes=30)

df['datetime'] = pd.to_datetime(df['datetime'])
df = df.loc[df['datetime'].dt.time.between(start_time.time(),
    end_time.time())]

df = df.reset_index(drop=True)

return df
```

Ispis kôda 5.1 Metoda select_col()

5.1.2 Vizualizacija aktivnosti populacije po intervalima

Za vizualizaciju podataka potrebno je prvo obraditi podatke, iz toga razloga proces od učitavanja sirovih podataka do prikaza rezultata ostvaren je prvo pozivanjem funkcija za obradu, koje ujedno i aktiviraju određene događaje. Tako je za vizualizaciju po intervalima kreiran događaj `ShowGraph` koji je povezan sa slušačem koji pokreće funkciju crtanja grafa.

```
frame.event_generate("<<ShowGraph>>")
```

Ispis kôda 5.2 Kreiranje događaja ShowGraph

```
self.bind("<<ShowGraph>>", self.plot_graph)
```

Ispis kôda 5.3 Slušač za događaj ShowGraph

Podatci koji su prethodno razdvojeni na 4 okvira u funkciji `split_for_graph()` učitavaju se u funkciji `plot_graph()`. Prostor za kreiranje grafa prvo se deklarira naredbom `f = Figure(figsize=(5, 5), dpi=100)`, argumentom `dpi=100` određuje se rezolucija fotografije prikazanog grafa a funkcijom `plot()` kojoj se proslijedi podatkovni okvir se graf crta.

```
f = Figure(figsize=(5, 5), dpi=100)
a = f.add_subplot(111)
a.plot(df_1['mean'])
a.plot(df_2['mean'])
a.plot(df_3['mean'])
a.plot(df_4['mean'])
```

Legenda u grafu određuje se sa naredbom `legend()` koja prima parametre ovisno o broju nacrtanih grafova te im dodijele pripadajuće nazive.

```
a.legend(['bsl morning', '1st expo', 'bsl noon', '2nd expo'])
```

Prikaz grafa u korisničkom sučelju ostvaren je zadavanjem okvira za prikaz (canvas), u koji će se smjestiti prethodno nacrtani graf.

```
self.canvas = FigureCanvasTkAgg(f, self)
self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH,
    expand=True)
```

```
3 self.canvas.draw()
```

Treba istaknuti da bi svakim pozivom funkcije `plot_graph()` dogodilo ponovno crtanje istoga grafa ili grafa s promijenjenim vrijednostima koji bi se dodao ispod prethodno kreiranog grafa. Ovaj problem rješava se brisanjem prethodno nacrtanoga grafa naredbom `canvas.get_tk_widget().pack_forget()`. Ako graf ne postoji funkcija `canvas.get_tk_widget().pack_forget()` vratit će `AttributeError`, programu se tada prosljeđuje naredba `pass` za nastavak rada. Potpuni kôd prikazan je u nastavku.

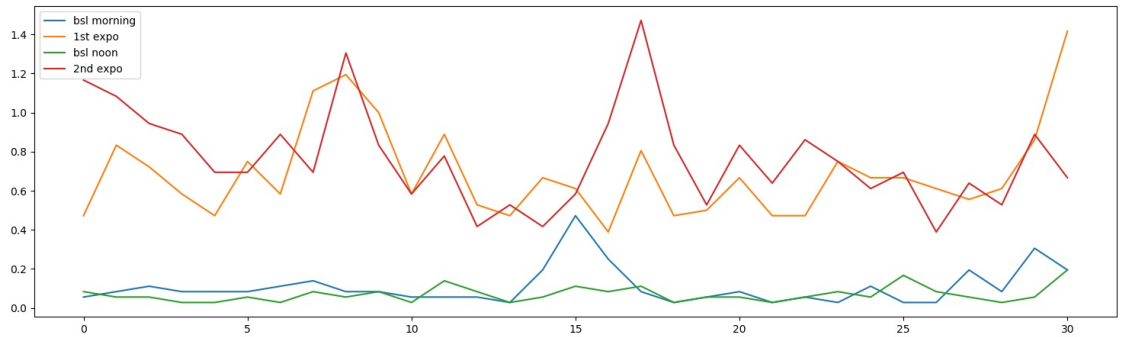
```
1 try:
2 self.canvas.get_tk_widget().pack_forget()
3 except AttributeError:
4 pass
```

Ispis kôda 5.4 Rješavanje problema dvostrukog crtanja grafa

U nastavku je prikazan cijeli kôd rada funkcije `plot_graph()` i rezultirajući graf (Slika 5.1) koji ona kreira.

```
1 def plot_graph(self,event):
2
3 df_1 = self.controller.df_1
4 df_2 = self.controller.df_2
5 df_3 = self.controller.df_3
6 df_4 = self.controller.df_4
7
8 try:
9 self.canvas.get_tk_widget().pack_forget()
10 except AttributeError:
11 pass
12
13 f = Figure(figsize=(5, 5), dpi=100)
14 a = f.add_subplot(111)
15 a.plot(df_1['mean'])
16 a.plot(df_2['mean'])
17 a.plot(df_3['mean'])
```

```
18 a.plot(df_4['mean'])
19 a.legend(['bsl morning', '1st expo', 'bsl noon', '2nd expo'])
20
21 self.canvas = FigureCanvasTkAgg(f, self)
22
23 self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH,
       expand=True)
24 self.canvas.draw()
```



Slika 5.1 Primjer vizualizacije podataka linijskim grafikonom za aktivnosti populacije po intervalima

5.1.3 Obrada individualne aktivnosti po intervalima

Populacijska i individualna analiza odvajaju se istovremeno u istoj funkciji, jer se radi o analizi istih podataka na dva načina. U populacijskoj analizi se gleda ukupni zajednički prosjek aktivnosti, a u pojedinačnoj se gledaju jedinke i njihova reakcija. Vizualno gledano matricu aktivnosti populacije mušica za određeni interval možemo transponirati i analizirati na razini jedinke. Transponiranje podataka ostvareno je naredbom `transpose()`, podatci se potom čiste od stupaca s vrijednostima vremena i datuma, prosjeka aktivnosti itd. Nad pročišćenim podatcima ponovo se poziva funkcija računanja prosjeka `mean()`. Individualna razina se odvija tako da se uspoređuje prosjek aktivnosti jedinke prije uzimanja psihostimulansa (BSL morning), nakon

prvog uzimanja (1st adm), prije drugog uzimanja (BSL noon) i nakon drugog uzimanja (2nd adm). Iz ovog razloga izračunate prosjeke jedinki pohranjujemo u stupce 'BSL_morning ', 'first_adm', 'second_adm'. Ove vrijednosti će se naknadno usporediti u svrhu određivanja mušica koje pokazuju povećanu aktivnost nakon obje administracije psihostimulansa, one koje samo nakon jedne administracije ili one koje imaju smanjenu aktivnost unatoč konzumiranju. Primjer kreiranja stupca 'BSL_morning' prikazan je u nastavku.

```
df_1_t = df_1.transpose()
df_1_t = df_1_t.drop(df_1_t.index[0:4])
df_1_t = df_1_t.reset_index(drop=True)
df_1_t = df_1_t.iloc[:, 0:5]
df_1_t['BSL_morning'] = df_1_t.mean(axis=1)
df_1_t = df_1_t['BSL_morning']
```

Nakon obrade i izračuna nad podatcima od stupaca 'BSL_morning ', 'first_adm', 'second_adm' kreira se novi okvir podataka. Novi okvir podataka kreiran je u svrhu usporedbe rezultata više eksperimenata što je detaljno objašnjeno u poglavlju Usporedba rezultata individualne analize.

```
frames = [df_1_t, df_2_t, df_4_t]
result = pd.concat(frames, axis=1)
```

Individualna analiza sastoji se od međusobnih usporedbi stupaca prije i nakon prvog uzimanja te nakon prvog i drugog uzimanja psihostimulansa. Mušice koje ne pokazuju promijene u aktivnosti označuju se slovom "S", eng. Same, one s povećanom aktivnosti označene su slovom "I" od engleske riječi "Increased", dok su one sa smanjenom aktivnošću označene kao "D" od riječi "Decreased".

```
result['BSLvs1st'] = 'S'
result['BSLvs1st'] = result['BSLvs1st'].where(result['BSL_morning
    '] <= result['first_adm'], 'D')
result['BSLvs1st'] = result['BSLvs1st'].where(result['BSL_morning
    '] >= result['first_adm'], 'I')
```

Kôd funkcije `split_for_graph()` prikazan je u nastavku. Prvi dio funkcije podatke razdvaja na 4 intervala potrebna za populacijsku analizu. U sljedećem dijelu funkcije

isti podatci se obrađuju ali na razini individualne analize.

```
def split_for_graph(self, start_bsl_morning, start_1st_expo,
    start_bsl_noon, start_2nd_expo):

df = self.controller.df.copy()
df_1 = DataEdit_.select_col(df, start_bsl_morning)
df_2 = DataEdit_.select_col(df, start_1st_expo)
df_3 = DataEdit_.select_col(df, start_bsl_noon)
df_4 = DataEdit_.select_col(df, start_2nd_expo)


result['BSLvs1st'] = 'S'
result['BSLvs1st'] = result['BSLvs1st'].where(result['BSL_morning
    '] <= result['first_adm'], 'D')
result['BSLvs1st'] = result['BSLvs1st'].where(result['BSL_morning
    '] >= result['first_adm'], 'I')

result['1stvs2nd'] = 'S'
result['1stvs2nd'] = result['1stvs2nd'].where(result['first_adm']
    <= result['second_adm'], 'D')
result['1stvs2nd'] = result['1stvs2nd'].where(result['first_adm']
    >= result['second_adm'], 'I')

result['BSLvs1stvs2nd'] = 'B'
result['BSLvs1stvs2nd'] = result['BSLvs1stvs2nd'].where(
(result['BSLvs1st'] == 'I') & (result['1stvs2nd'] == 'I'), 'False
    ')

self.controller.result = result
```

Ispis kôda 5.5 Kod funkcije split_for_graph()

5.1.4 Prikaz rezultata analize individualne aktivnosti

Podatci koji se prethodno obrađuju u funkciji `split_for_graph()` pohranjeni su u varijablu `self.controller.result` kojoj se može pristupiti tijekom rada programa u iz bilo koje klase. Proces vizualizacije rezultata u samom sučelju ostvaren je tekstualnim grafičkim elementom. Potrebno je samo pozvati funkciju `T.insert(tk.END, self.controller.result)`. Prilikom svakog poziva za prikaz ove klase (prozora), osvježuje se prikaz iz razloga promijene podataka. Rezultirajući podatci prikazani su na Slici 5.3

	BSL_morning	first_adm	second_adm	BSLvs1st	1stvs2nd	BSLvs1stvs2nd
0	0.0	0.4	0.2	I	D	False
1	0.8	1.2	3.0	I	I	S
2	0.0	0.0	0.0	S	S	False
3	0.0	0.0	0.0	S	S	False
4	0.0	0.6	2.4	I	I	S
5	0.0	0.0	0.0	S	S	False
6	1.2	1.0	0.6	D	D	False
7	0.8	0.2	3.8	D	I	False
8	0.0	3.2	0.0	I	D	False
9	0.0	1.6	0.8	I	D	False
10	0.6	2.0	0.0	I	D	False
11	0.0	0.0	0.6	S	I	False
12	0.2	2.8	2.8	I	S	False
13	0.0	1.0	2.6	I	I	S
14	0.0	0.8	0.0	I	D	False
15	0.6	0.0	0.0	D	S	False
16	0.0	0.0	0.6	S	I	False
17	0.0	0.0	2.2	S	I	False
18	0.0	0.6	0.0	I	D	False
19	0.2	1.0	0.0	I	D	False
20	0.0	0.0	0.0	S	S	False
21	1.4	1.4	0.0	S	D	False
22	0.0	2.0	0.8	I	D	False
23	0.0	1.0	0.0	I	D	False
24	1.0	0.0	0.0	D	S	False
25	0.0	1.4	5.8	I	I	S
26	0.0	0.0	0.0	S	S	False
27	2.6	0.6	0.0	D	D	False
28	1.8	3.0	0.2	I	D	False
29	0.0	1.6	0.0	I	D	False
30	0.0	0.0	0.0	S	S	False
31	0.0	0.0	0.4	S	I	False

Slika 5.2 Primjer tabličnog prikaza rezultata obrade podataka u individualnoj anailizi u korisničkom sučelju

5.2 Vizualizacija i obrada prosječne aktivnosti populacije po satu

5.2.1 Obrada podatak za prosječnu količinu aktivnosti po satu

Učitani podatci se obrađuju tako da se uzima izračunata vrijednost za prosjek. Potom se ista vrijednost računa za intervale od jednoga sata. Izračunati podatci se pohranjuju u novi okvir podataka.

```
1  def split_for_graph(self, start_bsl_morning, start_1st_expo,
       start_bsl_noon, start_2nd_expo):
2  df_ave = self.controller.df.copy()
3
4  df_ave['datetime'] = pd.to_datetime(df_ave['datetime'])
5  df_ave.index = df_ave['datetime']
6  df_p = df_ave.resample('H').mean()
7
8  df_p['Hour'] = df_ave['datetime'].dt.hour
9
10 df_hour_ave = pd.DataFrame(df_p['mean'])
11
12 self.controller.df_hour_ave = df_hour_ave
```

Ispis kôda 5.6 Funkcija main

5.2.2 Vizualizacija prosječne količine aktivnosti populacije po satu

U postupku vizualizacije funkcijom `plot_ave_hour_graph()` podatci su prethodno izračunati za prosjek po satu a potom se učitavaju iz podatkovnog okvira (DataFrame) i vizualiziraju. Pokretanje funkcije za crtanje grafa odvija se pomoću aktivacije slušača za događaj (event) `"<<ShowHourGraph>>"` koji je prethodno zadan u glavnom djelu klase `"HourAve_"` kôdom

```
self.bind("<<ShowHourGraph>>", self.plot_ave_hour_graph)
```

Ispis kôda 5.7 Slušač za događaj 'ShowHourGraph'

Vizualizacija se ostvarjue pomoću bilblioteke `matplotlib`. Podatci se prikazuju linijskim grafikonom za što je dovoljno samo pozvati funkciju `plot()`. Legenda u grafu prikazana je pomoću `a.legend(['hour average'])`.

```
f = Figure(figsize=(5, 5), dpi=100)

a = f.add_subplot(111)
a.plot(df['mean'])
a.legend(['hour average'])
```

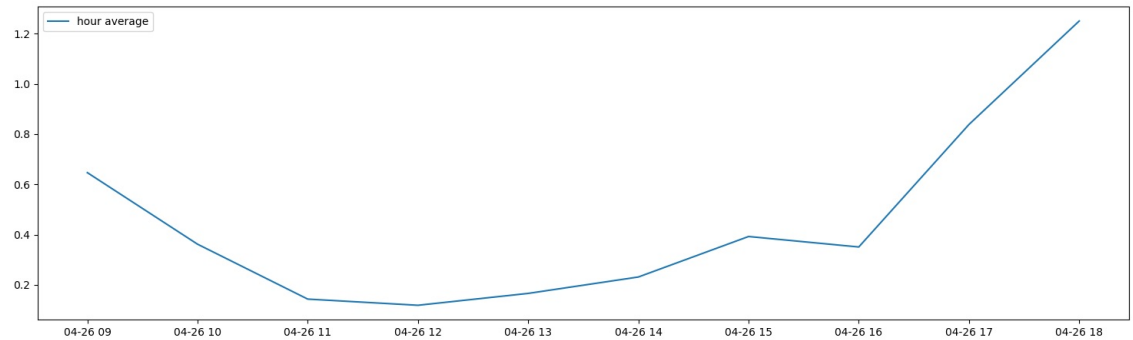
Ispis kôda 5.8 Vizualizacija aktivnosti po satu funkcijom plot_ave_hour_graph

Potpuni kôd funkcije `plot_ave_hour_graph()` prikazan je u nastavku.

```
def plot_ave_hour_graph(self, event):
try:
self.canvas.get_tk_widget().pack_forget()
except AttributeError:
pass

df = self.controller.df_hour_ave

f = Figure(figsize=(5, 5), dpi=100)
a = f.add_subplot(111)
a.plot(df['mean'])
a.legend(['hour average'])

self.canvas = FigureCanvasTkAgg(f, self)
self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH,
    expand=True)
self.canvas.draw()
```

Ispis kôda 5.9 Programski kôd za crtanje grafa prosiječne aktivnosti po satu

Rezultirajući graf prikazan je na Slici 5.3



Slika 5.3 Primjer vizualizacije podataka linijskim grafikonom aktivnosti populacije po satu

5.3 Usporedba rezultata individualne analize

Pored populacijske i individualne analize podataka pojedinog eksperimenta, u programu se nudi i mogućnost usporedbe individualnih rezultata tri eksperimenta. Razlog je praćenje aktivnosti jedinki za više od jednoga eksperimenta. Ovdje se jedinke promatraju kao ukupan broj jedinki koji pripada skupini ”I”, ”D” ili ”S”. Traži se udio jedinki s povećanom, smanjenom ili istom aktivnošću od ukupnoga broja mušica.

5.3.1 Obrada podataka aktivnosti jedinki iz više eksperimenata

Procesi koji su potrebni izvršiti se nad podatcima su:

- usporedba ukupnog broja mušica prije i nakon prve administracije psihostimulansa (SENS),
- usporedba ukupnog broja nakon prve i druge administracije psihostimulansa (1st adm vs 2nd adm),
- usporedba prije, nakon prve i nakon druge administracije (LS),
- kreiranje tablice za prikaz mušica po skupinama količine aktivnosti.

Učitavanje indivudalnih podataka u korisničkom sučelju ostvaruje se pomoću funkcije `browse()`. Učitani podatci pohranjuju se u tri zasebne varijable `df_1`, `df_2` i `df_3` koje se prosljeđuju funkciji `compare_experiments()` za daljnju obradu. Nad svakim od triju rezultata eksperimenta provodi se ista analiza nakon koje se kreira zajednička tablica. Tako da se iz svake obrade kreiraju tri tablice koje se naknadno spajaju u jednu zajedničku. Osim broja jedinki koje pripadaju određenoj skupini računa se i postotak mušica. Prebrojavanje ukupnog broja mušica po skupinama prije i nakon prve administracije i pohrana u zajedničku tablicu prikazana je u nastavku.

```
bsl_vs_first_exp_1 = df_1['BSLvs1st'].value_counts()
bsl_vs_first_exp_2 = df_2['BSLvs1st'].value_counts()
bsl_vs_first_exp_3 = df_3['BSLvs1st'].value_counts()

values_bsl_vs_first = [bsl_vs_first_exp_1, bsl_vs_first_exp_2,
    bsl_vs_first_exp_3]
result_bsl_vs_first = pd.DataFrame(values_bsl_vs_first)
result_bsl_vs_first = result_bsl_vs_first.reset_index(drop=True)
result_bsl_vs_first['Number of flies'] = result_bsl_vs_first.iloc
    [:, :].sum(axis=1)
```

Ispis kôda 5.10 Programski kôd za količinu mušica po skupinama sa smanjenom povećanom ili istom aktivnošću

Računanje postotka za jedinke po pripadnosti skupinama ostvareno je na sljedeći način.

```
result_bsl_vs_first['Decrease'] = result_bsl_vs_first['D'].div(
    result_bsl_vs_first['Number of flies'])
result_bsl_vs_first['Decrease'] = result_bsl_vs_first['Decrease']
    * 100
result_bsl_vs_first['Same'] = result_bsl_vs_first['S'].div(
    result_bsl_vs_first['Number of flies'])
result_bsl_vs_first['Same'] = result_bsl_vs_first['Same'] * 100
result_bsl_vs_first['Increase'] = result_bsl_vs_first['I'].div(
    result_bsl_vs_first['Number of flies'])
result_bsl_vs_first['Increase'] = result_bsl_vs_first['Increase']
```

```
    * 100
```

Ispis kôda 5.11 Programski kôd za izračun postotka po skupinama aktivnosti

Ista analiza prvodi se i za usporedbe podataka nakon prve i druge administracije i usporedbu prije prve administracije, nakon prve i nakon druge administracije. Potom se izvršava podjela mušica u skupine ovisno o količini aktivnosti. Količina aktivnost se gleda kao prosiječna vrijednost aktivnosti u tom periodu koja je najčešće u raspodnu od 0 do 10, no zbog izuzetaka gornja granica je pomaknuta na 15. Kôd za podjelu po količini aktivnosti prikazan je u nastavku.

```
ranges = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 7, 9, 15]

bsl_morning_1 = df_1.BSL_morning.groupby(pd.cut(df_1.BSL_morning,
    ranges)).count()
first_exp_1 = df_1.first_adm.groupby(pd.cut(df_1.first_adm,
   ranges)).count()
second_exp_1 = df_1.second_adm.groupby(pd.cut(df_1.second_adm,
   ranges)).count()

frames_1 = [bsl_morning_1, first_exp_1, second_exp_1]
result_1 = pd.concat(frames_1, axis=1)
```

Ispis kôda 5.12 Programski kôd za podjelu po aktivnosti po skupinama

	BSL_morning	first_adm	second_adm	BSL_morning	first_adm	second_adm	BSL_morning	first_adm	second_adm
(0.0, 0.5]	2	5	0	1	7	6	2	2	0
(0.5, 1.0]	3	3	0	4	8	8	7	3	0
(1.0, 1.5]	0	0	0	0	2	3	2	4	0
(1.5, 2.0]	0	0	0	0	2	4	1	2	0
(2.0, 2.5]	0	0	0	0	4	4	0	0	0
(2.5, 3.0]	0	1	0	0	0	0	1	1	0
(3.0, 3.5]	0	0	0	0	0	1	0	0	0
(3.5, 4.0]	0	0	0	0	0	1	0	0	0
(4.0, 4.5]	0	0	0	0	0	0	0	0	0
(4.5, 5.0]	0	0	0	0	0	0	0	0	0
(5.0, 7.0]	0	0	0	0	0	0	0	0	0
(7.0, 9.0]	0	0	0	0	0	0	0	0	0
(9.0, 15.0]	0	0	0	0	0	0	0	1	0

Slika 5.4 Prikaz korisničkog sučelja s učitanim podacima

Obrađeni podatci se zatim pohranjuju u varijablu kojoj se može pristupati prilikom vizualizacije podataka.

```
main_frames = [result_1, result_2, result_3]
sort_by_range_table = pd.concat(main_frames, axis=1)

self.controller.exp_comp_results = sort_by_range_table
```

Ispis kôda 5.13 Pohrana rezultata u varijablu kojoj se pristupa prilikom vizualizacije

5.3.2 Pohrana obrađenih rezultata usporedbe više eksperimenata

Podatci se pohranjuju u tabličnom zapisu u formatu .xlsx. Podatci su podjeljeni na način da svaka od četiri analize je pohranjena na zasebnu stranicu u dokumentu. U nastavku je prikazan kôd koji to omogućuje.

```
with pd.ExcelWriter('results/exp_compare.xlsx') as writer:
result_bsl_vs_first.to_excel(writer, sheet_name='
    result_bsl_vs_first')
result_first_second.to_excel(writer, sheet_name='
    result_first_second')
result_bsl_first_second.to_excel(writer, sheet_name='
    result_bsl_first_second')
sort_by_range_table.to_excel(writer, sheet_name='
    sort_by_range_table')
```

Ispis kôda 5.14 Programski kôd za pohranu podataka usporedbe u .xlsx formatu

Poglavlje 6

Zaključak

Tema ovoga diplomskog rada je implementacija sustava za obradu i vizualizaciju sirovih podataka praćenja aktivnosti vinske mušice pomoću programskog jezika Python. Vinske mušice važan su modalni organizam u istraživanju gena, razlog tome je što djele 75% gena sa ljudima. Veliki broj genskih bolesti kod ljudi može se istraživati na mušicama. Praćenje aktivnosti vinskih mušica važan je proces u promatranju njihovog ponašanja i reagiranju na konzumaciju psihostimulansa. Obrada tabličnih podataka stvorenih praćenjem aktivnosti vinske mušice često zahtjeva dugotrajan proces obrade. Iz toga razloga u okviru ovog diplomskog rada osmišljen je i dizajniran sustav koji ubrzava i pojednostavljuje taj proces. Postupci koji se mogu automatizirati nad podatcima su odabir stupaca i radaka, podjela ukupnih podataka praćenja na intervale i statistička obrada odabranih vrijednosti.
Sustav je razvijen u programskom jeziku Python koji podržava paradigmu objektno orijentiranog programiranja. Sustav je podjeljen na tri osnovna dijela: glavni dio koji pokreće aplikaciju i vrši poziv klasa za prikaz na korisnikovom ekranu, modul za obradu i selekciju podataka i treći dio gdje je deklariran izgled svakog pojedinog elementa sučelja. Naglasak u analizi podataka je na praćenju individualne i poulacijske aktivnosti i podjela po intervalima. Ovime procesom olakšava se uvid u rezultate eksperimenata koji se provode davanjem psihostimulansa vinskim mušicama i njihova reakcija na njih. Omogućeno je praćenje u promjeni aktivnosti prije i nakon konzumacije psihostimulansa. Rezultati promijene aktivnosti prikazani su tablično i grafički. Stvoreno je grafičko sučelje koje korisniku omogućava jednostavan i brz

način za interpretaciju podataka.
Sam sustav razvijen u sklopu ovoga diplomskoga rada može se dalje razvijati. Na sustavu je moguće dodati nove funkcionalnosti ovisno o željenim potrebama u analizi podataka. Izmjene i dopune jednostavne su za implementaciju zbog objektno orijentirane strukture kôda. Ovisno o analizi koju želimo provesti nad podatcima kreira se nova metoda unutar klase za obradu podataka i njezin rezultat se prikazuje pomoću odrežene klase za vizualizaciju. Nad sustavom se mogu ostvariti estetska i vizualna poboljšanja prilikom čega treba obratiti na pozornost da se kreira intuitivno sučelje. Ovaj rad je samo početak razvoja programa za analizu podataka vezanih za genetska istraživanja vinskih mušica čime se želi ubrzati i pojednostaviti proces provedbe analize podataka i smanjiti potrebno virijeme obrade.

Bibliografija

[1] (2019) TIOBE Index for July 2019 , s Interneta, https://www.tiobe.com/tiobe-index/

[2] (2018, Mar.) A brief history of python. , s Interneta, https://medium.com/@johnwolfe820/a-brief-history-of-python-ca2fa1f2e99e

[3] (2019, Jun.) The zen of python. , s Interneta, https://www.python.org/dev/peps/pep-0020/

[4] (2019) General python faq. , s Interneta, https://docs.python.org/2/faq/general.html#why-is-it-called-python

[5] (2019) Object oriented programming. , s Interneta, https://python.swaroopch.com/oop.html

[6] (2019, Jun.) pypi.org. , s Interneta, https://pypi.org/

[7] (2019, May) Tkinter. , s Interneta, https://wiki.python.org/moin/TkInter

[8] J. W. Shipman, *Tkinter 8.4 reference: aGUI for Python.* New Mexico Tech, Computer Center, 2010.

[9] (2019, Jun.) pandas: powerful python data analysis toolkit. , s Interneta, https://pandas.pydata.org/pandas-docs/stable/#pandas-powerful-python-data-analysis-toolkit

[10] (2019, Jun.) Matplotlib documentation. , s Interneta, https://matplotlib.org/

[11] (2019) Vinegar flies. , s Interneta, https://ento.psu.edu/extension/factsheets/vinegar-flies

[12] J. Sang, "Drosophila melanogaster: The fruit fly," *Encyclopedia of Genetics*, pp. 157–162, 01 2001.

[13] C. Pfeiffenberger, *Locomotor activity level monitoring using the Drosophila Activity Monitoring (DAM) System,*, 2010.

[14] A. Filosevic, "Behavioral and genetic characteristics of psychostimulant-induced neuronal plasticity in drosophila melanogaster," 2018.

[15] V. Buffalo, *Bioinformatics Data Skills*, 2015.

Popis slika

Popis tablica

Pojmovnik

GUI Graphical User Interface. 7

LS Lokomotorna senzitizacija. 12, 33, 42

SENS Osjetljivost. 12, 33, 42