Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Bakalářská práce

2019 **Martina Moravcová**

Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Softwarová predikce neuropeptidů v nemodelových organismech

Bakalářská práce

Martina Moravcová

Školitel: Mgr. Jiří Pech, Ph.D.

Konzultant: RNDr. Jan Provazník

České Budějovice 2019

Moravcová, M., (2019): Softwarová predikce neuropeptidů v nemodelových organismech. [Software prediction of neuropeptides in non-model organisms. Bc. Thesis, in Czech] – 69p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Annotation
The bachelor thesis deals with methods by which neuropeptides can be identified. The aim of this work is to write a script that would facilitate searching of candidate sequences suitable for prediction of neuropeptides and allow to work with larger input data.

Prohlašuji, že svoji bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 2019

Podpis ……

Poděkování

Děkuji Mgr. Jiřímu Pechovi, Ph.D. za trpělivost, ochotu a čas, který mi věnoval při psaní této práce. Dále děkuji RNDr. Janu Provazníkovi za cenné rady a konzultace.

Obsah

Úvod

Neuropeptidy jsou nedílnou součástí spousty funkcí ovlivňujících řadu pochodů dějících se prakticky ve všech živých organismech. V dnešní době existuje řada metod, jak identifikovat funkce a umístění těchto bílkovin a umožňuje nám to sledovat například vliv na chování hmyzu, ale i dalších organismů. Řada neuropeptidů byla již objevena a alespoň částečně vyzkoumána, avšak mnoho z nich nebylo doposud objeveno, a proto je to stále aktuální téma. V této práci zmíním metody, pomocí kterých se neuropeptidy dají identifikovat.

Tato práce se zaměřuje na prozkoumání možnosti vyhledávání potenciálních neuropeptidů pomocí jednodušších metod s využitím znalostí o struktuře prekurzorových molekul neuropeptidů.

Cíle práce

Hlavním cílem této práce bylo napsat skript, který by uživateli zjednodušil a urychlil vyhledávání kandidátních sekvencí pro predikci neuropeptidů. Zjednodušení spočívá v tom, že skript bude provádět několik kroků, které by uživatel jinak musel provádět jeden po druhém ručně, nebo pomocí web-serverového rozhraní, které ovšem umožňuje vkládání jen omezeného množství dat. Pomocí skriptu tedy bude možné zkoumat rozsáhlejší data a výstupem bude, mimo jiné, výsledek s nalezenými kandidátními sekvencemi.

Skript by měl umožnit:

- Zjištění signálních sekvencí (SignalP)
- Vybrat pouze sekvence obsahující signální peptid
- Vyhledání pattern match
- Blast nalezených sekvencí proti databázi jiných zvířat
- SQL dotaz pro vytvoření tabulek a naplnění výslednými daty
- Výsledné soubory jednotlivých kroků

1. Neuropeptidy

1.1. Charakteristika

Neuropeptidy jsou krátké bílkoviny, které hrají velmi důležitou roli ve funkci a regulaci nervového systému prakticky všech organismů, které nervový systém mají. Jsou součástí růstu, reprodukce, metabolismu a také chování hmyzu. Některé neuropeptidy hrají roli jako neurotransmitery, přenáší tedy informaci z jednoho neuronu na druhý. Jiné neuropeptidy jsou uvolňovány do krevního řečiště a poté slouží jako hormony a některé jsou jako hormony uvolňovány endokrinními žlázami. Mnoho z nich bylo původně objeveno jako hormony, a to hormony hypofyzární a gastrointestinální. Lze je rozdělit podle chemické struktury, funkce v organismu, nebo podle tkáně, ve které vznikají. Ovšem je nutné mít na paměti, že ačkoliv mohou být strukturně opravdu velice podobné, nemůžeme říci, že jejich funkce jsou stejné. Ba naopak, mohou být někdy velice odlišné. Metabolických drah, které neuropeptidy ovlivňují je velké množství, ne-li všechny. Neuropeptidy jsou na rozdíl od „klasických" neuropřenašečů syntetizovány odlišným způsobem a vykazují i řadu odlišných charakteristik.

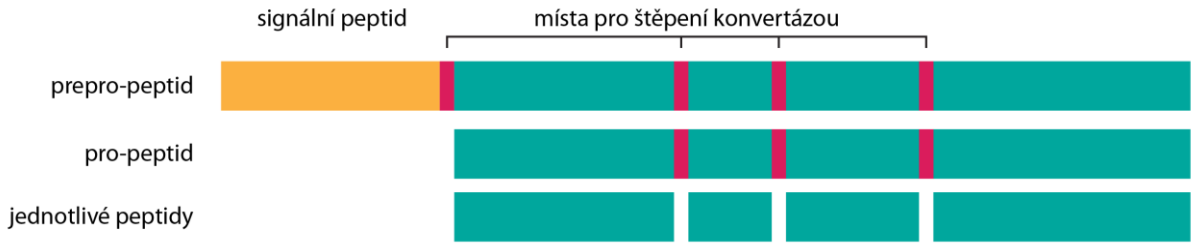
Je známo, že se peptidy dělí do skupin podle místa nebo oblasti, ve které byla poprvé popsána jejich funkce. Také je můžeme rozdělit podle jejich účinku. Neuropeptidy se zpravidla jmenují podle toho, kde byly původně objeveny a co dělají, neznamená to však, že se nachází jen v těchto místech a mají pouze tyto funkce.

Současná molekulární biologie a reverzní genetika umožňuje manipulovat s biologickými pochody modelových organismů a dávat tak odpovědi kupříkladu na otázky vzniku chorob, kontroly škůdců, tvorby ekonomicky zajímavých látek a podobně. Mnohdy je však výchozím bodem takového výzkumu znalost sekvence nukleotidů nebo aminokyselin. Moderní metody „high-throughput" sekvenování DNA a potažmo RNA nám, na jednu stranu, poskytují velké množství dat, na druhou stranu je mnohdy komplikované určit, co dané sekvence kódují. V tomto místě do hry vstupuje softwarová predikce funkce a struktury biologických sekvencí. (1) (2) (3) (4) (5)

1.2. Identifikace

V případě neuropeptidů existuje několik základních *in silico* metod, jak je možné je identifikovat. První z nich je celková podobnost (homologie) sekvence s již popsanými sekvencemi z jiných organismů. Tato metoda je velmi účinná ve chvíli, kdy se jedná o neuropeptid velmi blízkého organismu, nebo je to neuropeptid konzervovaný – tedy v evoluci jen nepatrně pozměněný a sekvenčně velmi podobný. Druhý způsob vychází z potřeby najít neuropeptidy, které jsou nové, dosud neobjevené a nepodobné ničemu již popsanému. Zde je možné použít třeba neurálních sítí jako v případě programu NeuroPID, o kterém se zmíním později.

Při identifikaci neuropeptidů je klíčovým faktorem jejich sekvenční struktura viz Obr. 1.1. a způsob, kterým jsou v organismu syntetizovány. Všechny neuropeptidy mají několik společných rysů, právě v tom, jak jsou vyráběny. Tvorba neuropeptidu začíná syntézou proteinu ve formě prepro-peptidu. Předpona "pre" označuje, že sekvence pro-peptidu obsahuje signální peptid – specifická sekvence, která buňce signalizuje, do jakého buněčného kompartmentu bude finální peptid putovat. Signální peptid je v příslušný čas enzymaticky odstraněn za vzniku pro-peptidu. Následující předpona "pro" naznačuje, že protein vzniklý po odstranění signální sekvence ještě není hotovým produktem, ale vyžaduje další (mnohdy i několikanásobné) sestřižení některým z enzymů z rodiny konvertáz. (4) (6)



Obr. 1.1. Schéma struktury neuropeptidu

2. Dostupné programy

Jak již bylo zmíněno, existuje několik způsobů, jak neuropeptidy identifikovat. Proto se v této části budu zabývat některými dostupnými programy pro tuto problematiku.

2.1. BLAST

BLAST, neboli Basic Local Alignment Search Tool, je algoritmus, který porovnává podobnost vkládané, neboli dotazované sekvence, se sekvencemi již známými, popsanými. Hovoříme tedy o homologii. BLAST původně vznikl pod záštitou databáze NCBI, o které se zmíním později. Důležité je zmínit, že BLAST nevyhledává celou délku sekvence, která nás zajímá, ale pouze její část (tzv. slovo), kterou poté prodlužuje. Nejprve je důležité ze sekvence odstranit oblasti, které mají nízkou komplexitu (to můžou být např. repetitivní úseky). Následně sekvenci rozdělit na krátká slova, která jsou poté vytříděna a seřazena a jsou z nich vybrána vhodná slova, která jsou lepší než zadaný práh (tzv. threshold). Z takto vybraných slov se poté sestaví efektivní vyhledávací strom a hledá se přesná shoda daného slova v databázi. V dalším kroku je zapotřebí prodloužit slovo a rozhodnout, zda má prodlužování s danými požadavky význam, nebo popřípadě spojit úseky, které byly nalezené vyhledáváním různých slov. V posledním kroku se z GenBank vypíší záznamy, které prošly rozhodováním a tím pádem jsou podobné naší, dotazované, sekvenci.

Citlivost prohledávání ovlivňuje délka slova. U dlouhých slov je rychlejší prohledávání, ovšem naleznou jen velice podobné záznamy. Naopak u krátkých slov je možné objevit i málo podobné sekvence, ale rychlost prohledávání je menší. Před samotným porovnáváním sekvence máme možnost zvolit si příslušný program pro spuštění, viz Obr.2.1, a vybíráme ho podle toho, jaké sekvence vkládáme a v jaké databázi hledáme – nukleotidy nebo aminokyseliny. Pokud algoritmus BLASTu narazí na některé z omezení, jako např. počet nalezených záznamů, vyhledávání se ukončí.

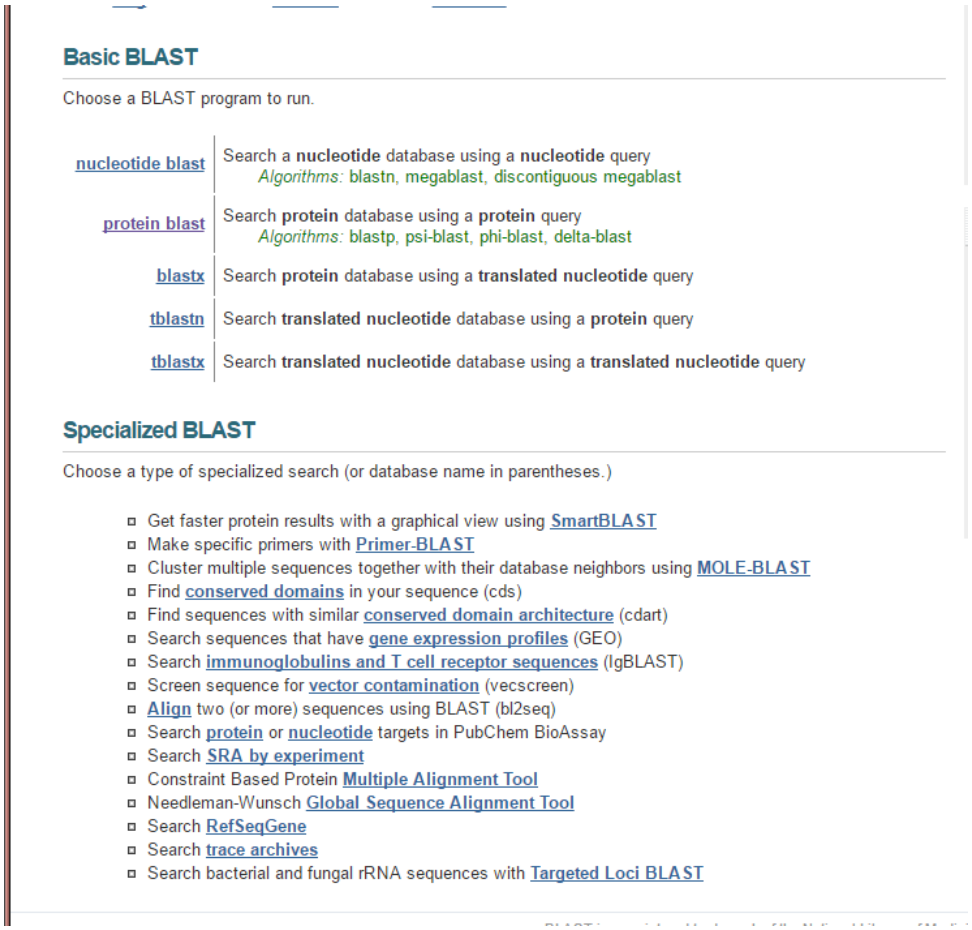
Tato nastaveni neboli parametry můžeme měnit. Jak už bylo zmíněno, můžeme nastavit např. počet nalezených záznamů, potom také velikost nebo délku slova, typ matice a cenu za vložení nebo prodloužení mezery (gap). Gap je jedním z prvků používaným k určování skóre alignmentu (dalšími a hlavními prvky jsou shody a neshody v aminokyselinách nebo nukleotidech). K určování skóre slouží také tzv. scoring matrix, což jsou tabulky, které popisují cenu záměny jedné aminokyseliny v jinou. Některé aminokyseliny mohou být do určité míry nahraditelné jinými. Podobné aminokyseliny následně dramaticky nemění strukturu a funkci proteinu, a proto z evolučního hlediska není

záměna dvou podobných aminokyselin tak závažná, jako záměna z jedné do strukturně a chemicky úplně jiné.

Za nezbytné považuji zmínit tzv. E-hodnotu (E-value), což je hodnota udávající statistické skóre, která odráží jak velikost databáze, tak použitý systém skóre. Je to hodnota, která je očekávaná na základě výskytu podobné sekvence v databázi, která má skóre S, nebo lepší. Čím větší je skóre S, tím nižší je hodnota E. Pro E-hodnotu je typické číslo 10, avšak pokud hodnotu zvýšíme, můžeme očekávat, že algoritmus najde větší počet podobných sekvencí. (7) (8)

Vzorec pro výpočet E-hodnoty je následující:

$$E = Kmne^{-\lambda S}$$



Obr. 2.1 - Blast

2.2. NeuroPred

Neuropred slouží jako webový nástroj, který byl navržen pro predikci neuropeptidů. Je možné zkoumat jak jeden řetězec aminokyselin, tak více sekvencí současně, které lze vybrat z několika predikovaných modelů a volitelných uživatelem definovaných funkcí. Zároveň také dokáže vypočítat hmotu predikovaných peptidů a umožnuje uživateli zvolit si post-translační modifikace. To umožnuje objev a potvrzeni nových neuropeptidů pomocí technik hmotnostní spektrometrie. (9) (10)

2.3. NeuroPID

NeuroPID je dalším nástrojem k predikci neuropeptidů. Funguje na základě 4 modelů strojového učení, včetně support vector machines (metoda podpůrných vektorů).

Používá jako vstupní data databázi známých neuropeptidů a databázi kandidátních sekvencí. Autoři uvádějí, že tento nástroj umožňuje detekci i nových neuropeptidů. (11)

3. Databáze

3.1. NCBI

NCBI (National Center for Biotechnology Information) je součástí Národního institutu zdraví (the National Institute of Health) ve Spojených státech amerických. NCBI zprostředkovává sérii databází, které jsou důležité pro biotechnologie, biomedicínu a jsou důležitým zdrojem pro bioinformatické nástroje. Jednou z nejhlavnějších databází je tzv. PubMed, který obsahuje různé vědecké články a texty, které jsou hlavně využívané v biomedicíně. Dalšími databázemi jsou například i GenBank a Refseq. Všechny databáze jsou uživatelům k dispozici online prostřednictvím vyhledávače záznamů. (12)

3.1.1. Refseq

RefSeq, neboli The Reference Sequence, je databáze s tzv.otevřeným přístupem (open access), která vznikla pod záštitou NCBI. Obsahuje anotované, spravované, a hlavně striktně ověřené sbírky nukleotidových sekvencí a jejich proteinové produkty, které jsou veřejně dostupné. Na rozdíl od GenBank poskytuje pouze jeden zápis pro každou molekulu hlavních organismů (od virů, bakterií, až po eukaryota). (13)

3.1.2. GenBank

GenBank je další databází, která vznikla pod záštitou NCBI. Je to databáze DNA sekvencí, která koordinuje s jednotlivými laboratořemi a ostatními sekvenčními databázemi, jako např. s tzv. DDBJ (the DNA Data Bank of Japan). Na rozdíl od Refseq databáze, zde nejsou data tak striktně ověřována a databáze není limitovaná pouze na hlavní modelové organismy. (14)

3.2. SwissProt

SwissProt je databáze zaměřená hlavně na proteiny a snaží se poskytnout opravdu vysokou úroveň anotací, jako např. popisy funkcí proteinu, jejich doménové struktury a další. Zároveň se snaží poskytnout minimální úroveň nadbytečnosti, a naopak zase vysokou úroveň integrace s jinými databázemi. Dbá se zde na to, aby byly sekvence hodně ověřované a hlídané. (15)

3.3. Prosite

Prosite je proteinová databáze, která se skládá ze záznamů, které popisují proteinové rodiny, domény a funkční stránky tak dobře, jako modely aminokyselin a profily v nich. Tyto záznamy jsou ověřovány a kontrolovány týmem ze švýcarského institutu bioinformatiky a následně integrovány do proteinových anotací ve SwissProt. Využití Prosite spočívá v identifikaci možných funkcí nově objevených proteinů a v analýze již známých proteinů pro zatím neurčenou aktivitu. Vlastnosti dobře studovaných genů mohou být propagovány k biologicky příbuzným organismům a pro rozdílné nebo málo známé geny mohou být biochemické funkce predikovány z podobností. Prosite nabízí nástroje pro analýzu proteinových sekvencí a pro detekování motivů. (16) (17)

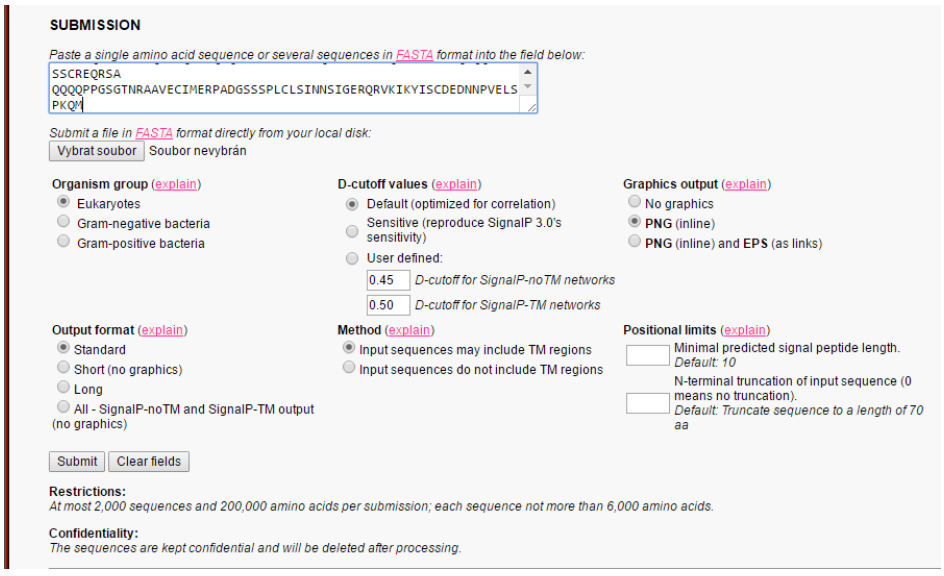
4. Detekce signálního peptidu

Při predikci proteinových sekvencí se nevyhneme práci s velkým množství jednotlivých proteinů. Pokud jsme striktní tak v průměrném eukaryotickém genomu jsou miliony až desítky milionů potenciálních proteinů. Je samozřejmé, že velká většina z těchto sekvencí nemá reálný biologický význam a je tedy třeba tento velký balík sekvencí profiltrovat. Prvním takovým krokem v této práci je detekce signálního peptidu pomocí SingalP.

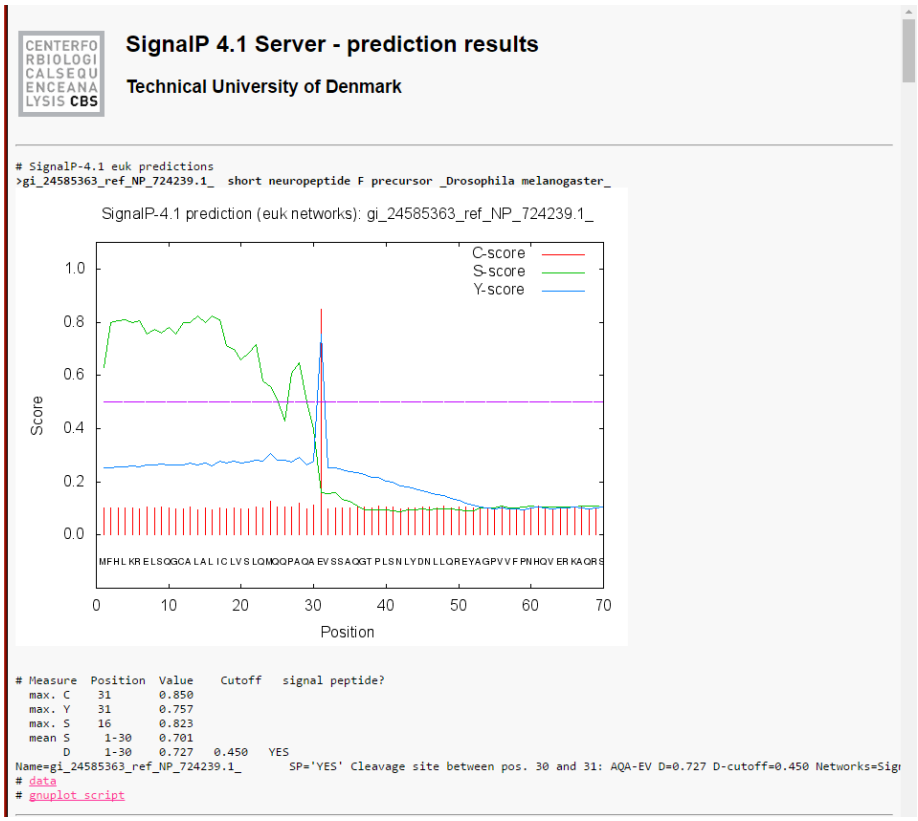
4.1. SignalP

SignalP je jedním z produktů CBS (The Center for Biological Sequence Analysis). Je to server s webovým rozhraním, viz Obr. 4.1, díky kterému můžeme predikovat přítomnost a umístění štěpných míst signálních peptidů v aminokyselinových sekvencích od různých organismů (gram-pozitivních prokaryot, gram-negativních prokaryot a eukaryot). Tato metoda zahrnuje predikci štěpných míst a signálních/non-signálních peptidů, založenou na kombinaci několika umělých neuronových sítí. SignalP je možné stáhnout i jako software a používat ho pomocí příkazové řádky, avšak stáhnout ho můžeme, pouze pokud jsme na seznamu akademických institucí. Pro uživatele je pochopitelně pohodlnější použít webové rozhraní, kam vloží sekvenci, nastaví potřebné parametry a poté si počká na výsledek. Nicméně problém nastává v momentě, kdy je sekvencí příliš mnoho. Do tohoto webového rozhraní je totiž možné vložit pouze omezený počet sekvencí. Takže pokud jsme členem akademické instituce, která na seznamu je, a pokud potřebujeme testovat rozsáhlejší data, je výhodnější stáhnout si software.

Výstupem SignalP je buďto graf s informacemi o pravděpodobnosti výskytu signálního peptidu v dané sekvenci, viz Obr. 4.2, nebo v případě softwarové verze s ovládáním přes příkazový řádek tabulka obsahující informace o pravděpodobnosti výskytu signálního peptidu v dané sekvenci viz Obr. 4.3. (18) (19) (20)



Obr. 4.1 - webové rozhraní SignalP



Obr. 4.2

```
martina@martina-VirtualBox:~/signalp-4.1$ ./signalp -c 0 -f short -u 0.34 -U 0.34 euk10.fsa
# SignalP-4.1 euk predictions
# name                    Cmax  pos  Ymax  pos  Smax  pos  Smean   D     ?  Dmaxcut    Networks-used
IPI:IPI00000001.2         0.327  42  0.213  42  0.373  66  0.119   0.163 N  0.340      SignalP-noTM
IPI:IPI00000005.1         0.173  92  0.132  32  0.164  28  0.128   0.130 N  0.340      SignalP-noTM
IPI:IPI00000006.1         0.162  90  0.132  32  0.162  28  0.126   0.129 N  0.340      SignalP-noTM
IPI:IPI00000012.4         0.830  30  0.770  30  0.845  15  0.722   0.751 Y  0.340      SignalP-TM
IPI:IPI00000013.1         0.774  18  0.840  18  0.943  15  0.909   0.877 Y  0.340      SignalP-noTM
IPI:IPI00000015.2         0.127 136  0.115 136  0.119  14  0.100   0.107 N  0.340      SignalP-noTM
IPI:IPI00000017.1         0.112  22  0.113  12  0.140  34  0.110   0.111 N  0.340      SignalP-noTM
IPI:IPI00000020.1         0.119  66  0.140  12  0.227   4  0.177   0.160 N  0.340      SignalP-noTM
IPI:IPI00000021.5         0.165  72  0.180  11  0.350   2  0.286   0.237 N  0.340      SignalP-noTM
IPI:IPI00000023.4         0.769  40  0.810  40  0.916  35  0.532   0.699 Y  0.340      SignalP-TM
```

Obr. 4.3

5. Použitý programovací jazyk

Pro tuto bakalářskou práci byl využit programovací jazyk Perl s jeho dostupnými moduly. Zde se mu chvíli budeme věnovat.

5.1. Historie

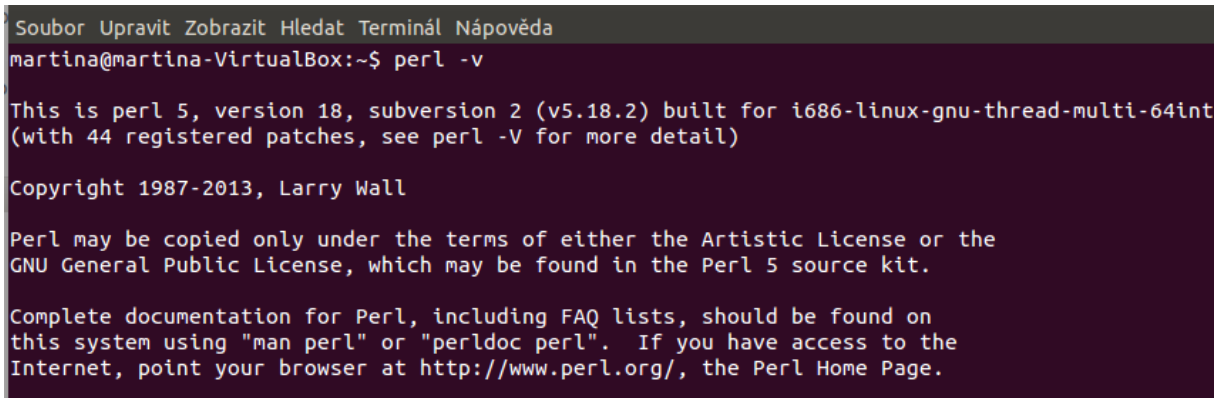
Perl je skriptovací jazyk, který byl vyvinut jako nástroj pro zpracování textu. Nejprve vzniknul pro vlastní potřebu jeho autora, jímž je Larry Wall, a to proto, že chtěl nástroj, který zvládne více požadavků a bude jednodušší než dostupné nástroje. V roce 1987 dal první verzi Perlu k dispozici veřejnosti a zájem byl k jeho udivení tak veliký, že se Perl začal vyvíjet natolik, že se z jednoduchého nástroje na zpracování textu stal programovací jazyk, který disponoval mnoha doplňky. Stal se populárním také kvůli jeho použitelnosti pro tvorbu tzv. CGI (Common Gateway Interface) skriptů, které byly velice důležitou součástí většiny webových aplikací.

Z Perlu se stal jazyk, kterým lze, oproti jiným jazykům, rychle a efektivně programovat aplikace, a to jak strukturovaně, tak objektově. Je také volně k dispozici, a tak si ho každý může stáhnout a zdarma ho používat i pro komerční projekty. Původně byl vyvinut pro Unix, avšak nyní ho lze využívat i na jiných platformách a lze mezi nimi kód přenášet, což je v dnešní době velikou výhodou. Můžeme se však setkat s výjimkami, kdy přenesený kód z jedné platformy na jinou bude nutno upravit. (21) (22) (23)

5.2. Fungování

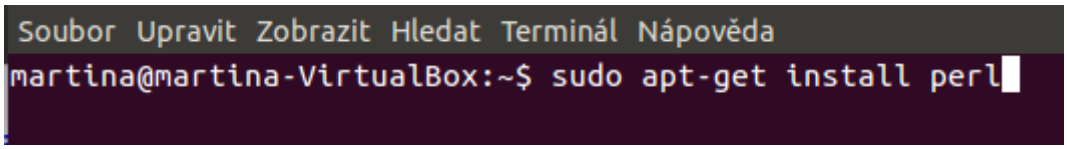
U Perlu je velikou výhodou, že pro programování v něm nám postačí textový editor a interpret Perlu. Existuje samozřejmě i kompilátor, který pro psaní není nutný, avšak poslouží nám v momentě, kdy chceme program poskytnout někomu dalšímu a nechceme, aby viděl kód.

Instalace Perlu (v rámci Linuxu) ve většině případů není nutná, protože jeho interpret je pravděpodobně již ve vašem systému obsažen. Pokud však nevíte, zda tomu tak je, postačí zadat do konzole jednoduchý příkaz viz Obr. 5.1.

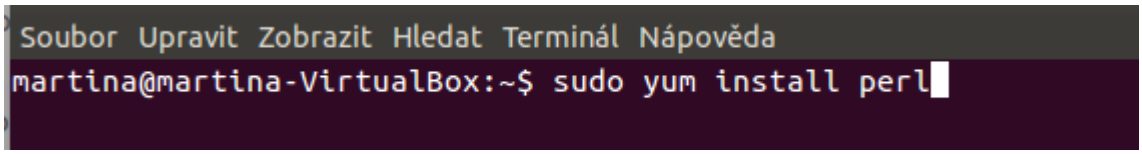


Obr. 5.1

Pakliže v systému interpret perlu není, je možné ho nainstalovat pomocí příkazu v konzoli viz. Obr. 5.2 a Obr.5.3.



Obr. 5.2 - (Instalace pro Ubuntu, Debian)



Obr. 5.3 - (Instalace pro Centos, Red Hat)

V ostatních případech lze stáhnout aktuální verzi v podobě souboru, který je třeba rozbalit, zkompilovat a nainstalovat. V případě, že chceme Perl naistalovat pro operační systém Windows, stáhneme instalační soubor, který stačí po spuštění nainstalovat dle pokynů v instalačním prostředí a následně můžeme Perl používat klasicky, přes konzoli. Lze jej stáhnout na *https://www.perl.org/get.html*.

Jak už bylo zmíněno výše, pro psaní v perlu nám postačí textový editor. Výhodou pochopitelně je, pokud editor umí zvýraznit syntaxi a odsazovat. Abychom systému dali najevo, v čem program píšeme a jak ho tím pádem má spustit, je nutné v každém programu začít specifickou úvodní řádkou (`#!/usr/bin/perl`), která nám značí cestu k danému interpretu, který má být použit pro spuštění programu.

Kromě úvodní řádky používáme pro programy psané v perlu i specifickou příponu *.pl*, která sice není nutná, kvůli již zmíněné řádce, ale je vhodné ji používat kvůli přehlednosti.

5.3. BioPerl

BioPerl je sbírka modulů Perlu, které slouží pro snadnější vývoj bioinformatických aplikací. Je produktem snahy uživatelů o to, vytvořit takový kód Perlu, který bude použitelný pro biologii. Poskytuje softwarové moduly pro řadu úloh, které jsou typické pro bioinformatické programování.

- Přístup k datům nukleotidových a peptidových sekvencí z lokálních a vzdálených databází.
- Přeměna vstupního formátu na jiný formát (jak pro databáze, tak pro soubory).
- Manipulace s jednotlivými sekvencemi.
- Vytváření strojově čitelných sekvenčních anotací.
- Vyhledávání genů a jiných struktur genomové DNA.
- Vyhledávání podobných sekvencí.

Pro instalaci BioPerlu je potřeba mít nainstalovaný funkční Perl. Z důvodu aktuálnosti dat, je doporučeno instalovat veškeré moduly přes tzv. CPAN (Comprehensive Perl Archive Network). CPAN je Perlovský archiv, který obsahuje všechny moduly, které se dají využívat pro Perl. Výhodou tohoto archivu je, že obsahuje vždy aktuální verzi modulů, které si můžeme pomocí CPAN nainstalovat, popř. aktualizovat. (24)

6. Skript

Tento skript slouží ke zjednodušení vyhledávání kandidátních sekvencí, které jsou potřeba k predikci neuropeptidů. Uživateli postačí zadat do příkazové řádky příkaz pro spuštění skriptu, spolu s dvěma argumenty, což jsou v tomto případě název fasta souboru a název databáze pro výpočet Blast. Fasta soubor je textový soubor s definovaným formátováním, který slouží k ukládání dat sekvencí nukleotidů, nebo aminokyselin. Záznam sekvence se skládá ze dvou řádků. První řádek obsahuje znak „>“a jméno sekvence, případně další informace o sekvenci (anotace). Na následujícím řádku je pak samotná sekvence nukleotidu nebo aminokyselin používající jednopísmenové IUPAC označení.

Motivací pro vytvoření tohoto skriptu je primárně potřeba procházet velké množství peptidových sekvencí v případě nemodelových organismů. Vstupním souborem jsou predikované aminokyselinové sekvence vytvořené při skládání transkriptomických dat.

Takto predikované soubory často obsahují stovky tisíc sekvencí, a tedy nejsou vhodné pro ruční nebo web-serverové zpracování.

Očekává se, že uživatel již vytvořil vstupní sekvence, například pomocí programu Trinity (*https://github.com/trinityrnaseq/trinityrnaseq/wiki*) a následně predikoval peptidy pomocí TransDecoder (*https://github.com/TransDecoder/TransDecoder/wiki*).

Pro fungování skriptu je nutné mít nainstalovaný SignalP i Blast. Pro spuštění skriptu je třeba ho mít umístěný ve složce, kde se nachází skript SignalP. Vývojový diagram skriptu je přiložen k této práci jako příloha č.1.

6.1. Zdrojový kód

```
#!/usr/bin/perl
use strict;
use warnings;
use Bio::SeqIO;
use Time::Piece;

# Controlling if the user entered two arguments

if ($#ARGV != 1) {

print "\n\n************************************

The skript needs fasta file and Blast database name as an argument for
running.
for example: ./skript fastaFileName databaseName

************************************\n\n";
      exit;
}

my $testDate = localtime();

print "\n\nStart date and time: $testDate";

my $fastaFile = $ARGV[0]; # Fasta file name entered as an argument
my $databaseFile = $ARGV[1]; # Blast database
print "\n\n************************************

Successfully loaded fasta file - $fastaFile

Successfully loaded database file - $databaseFile
```

Skript byl napsán v jazyce Perl a využívá čtyři knihovny:

- `strict`
- `warnings`
- `Bio::SeqIO`
- `Time::Piece`

První dvě knihovny slouží k usnadnění práce při řešení chyb ve skriptu, třetí knihovna umí pracovat se sekvencemi – je to knihovna Bioperlu a čtvrtá knihovna nahrazuje standardní funkce localtime a gmtime implementacemi, které vracejí objekty.

Jak už bylo psáno výše, pro spuštění je třeba zadat dva argumenty, kterými jsou v tomto případě fasta soubor a databáze pro výpočet Blast. Proto následuje `if` statement, který kontroluje, zda byly zadány právě dva argumenty a pokud ne, vypíše se uživateli nápověda a skript se ukončí. Pokud ano, do proměnné `fastaFile` se načte název vstupního souboru, který bude v poli `ARGV` uložen na nulté pozici a do proměnné `databaseFile` se načte název databáze, který bude v poli `ARGV` uložen na první pozici. Uživateli se vypíše, že byly soubory úspěšně načteny. Dále je zde proměnná `testDate`, do které se pomocí funkce `localtime` ukládá aktuální čas a datum a tyto informace jsou uživateli vypsány do konzole.

```
# sensitivity setup for SignalP

print "Enter a D-cutoff value for noTM networks (default = 0.34): ";
my $inputNoTM = <STDIN>;
chomp $inputNoTM;
print "\n\nEnter a D-cutoff value for TM networks (default = 0.34): ";
my $inputTM = <STDIN>;
chomp $inputTM;

# If user do not enter any value the script uses the default one

if ($inputNoTM eq '') {

$inputNoTM = 0.34;

}
if ($inputTM eq '') {

$inputTM = 0.34;

}
```

V této části skriptu je po uživateli vyžadováno zadání dvou hodnot pro výpočet SignalP. Může zadat buďto vlastní hodnoty, nebo ponechat již defaultně nastavené. Dále jen `if` statement kontroluje, zda byly uživatelem zadány jiné hodnoty, nebo zda má skript pracovat s těmi defaultními.

```
# Creating new fasta file with data from SignalP
print "\n\nThe results will be moved to the directory named as an entered
prefix.

Enter prefix: ";
my $prefix = <STDIN>;
chomp $prefix;
```

Dále se uživateli vypíše, že se výsledky uloží do složky, která bude pojmenována dle prefixu, který si uživatel zvolí. Stejně tak budou mít daný prefix i všechny výstupní soubory. Poté bude uživatel dotázán na název prefixu, který se ukládá do proměnné `prefix` a je zde očekáván vstup z klávesnice.

```
# Motif selection - at least one must be selected

my $choice;
my $rangeRKKR;
my $rangeMETHIONIN;
my $rangeFURIN;
my $input;

while (1) {

      print "\n\n************************************";
      print "\n\nWhich of the motifs do you want to use?\n
            1 - RK/KR - RK/KR\n
            2 - METHIONIN - FURIN\n
            3 - FURIN - FURIN\n
            4 - All of them\n
            Enter value (1/2/3/4): ";

      $choice = <STDIN>;
      chomp $choice;

      if ($choice == 1) {

            print "\n\n************************************\n\n";
            print "Range for RK/KR - RK/KR: ";
            $input = <STDIN>;
            chomp $input;
            $rangeRKKR = $input;
            last;

      } if ($choice == 2) {

            print "\n\n************************************\n\n";
            print "Range for METHIONIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeMETHIONIN = $input;
            last;

      } if ($choice == 3) {

            print "\n\n************************************\n\n";
            print "Range for FURIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeFURIN = $input;
            last;

      } if ($choice == 4) {

            print "\n\n************************************\n";
            print "\nRange for RK/KR - RK/KR: ";
            $input = <STDIN>;
            chomp $input;
```

```
            $rangeRKKR = $input;

            print "\nRange for METHIONIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeMETHIONIN = $input;

            print "\nRange for FURIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeFURIN = $input;
            last;

      }

      print "\nWrong value!";

}
```

V tomto kroku bude skript vyžadovat po uživateli zvolení jedné z možností ohledně výběru motivu. Nejprve jsem si vytvořila proměnnou `choice`, která bude očekávat jakýkoliv vstup z klávesnice (`STDIN`). Dále proměnné `rangeRKKR`, `rangeMETHIONIN` a `rangeFURIN`, do kterých se budou pomocí proměnné `input` ukládat uživatelem zadané délky patternu – `range`, zvolené pro jednotlivé motivy. Proměnná `input` bude také očekávat vstup z klávesnice. Vzhledem k tomu, že na výběr je pouze ze čtyř možností, následují čtyři `if` statementy, které kontrolují, zda je zadána právě jedna z nabízených možností. Pokud ano, skript pokračuje dále a pokud ne, vypíše se uživateli nápověda „špatná hodnota“ a dá mu na výběr opět z nabízených čtyř možností. To dělá skript z toho důvodu, že je tento výběr v nekonečném `while` cyklu, který se bude opakovat tak dlouho, dokud nebude správně zadaná jedna ze čtyř možností. Až se tak stane, cyklus se ukončí příkazem `last`.

```
# SQL

print "\n\n************************************";

print "\n\nDo you want create an SQL file? (N/y): ";
my $SQL = <STDIN>;
chomp $SQL;
my $tableName;

if ($SQL eq "Y" | $SQL eq "y") {
      print "\n\nEnter a table name: ";
      $tableName = <STDIN>;
      chomp $tableName;
}
```

Dále bude uživatel dotázán, zda si přeje vytvořit SQL dotaz a do proměnné `SQL` se uloží vstup z klávesnice. Následuje `if` statement, který kontroluje, zda bylo zvoleno, že ano a je třeba napsat (zvolit si) název tabulky. Ten se ukládá do proměnné `tableName`.

Pokud si uživatel nepřeje vytvořit SQL dotaz a vstup z klávesnice je jiný, než `Y/y`, nebude se skript dotazovat na název tabulky a bude přeskočena část kódu pro SQL.

```
print "\n\n************************************";

my $start = localtime;

print "\n\nProcessing SignalP... Please wait...\n"; # Start of SignalP

system ("./signalp -c 0 -f short -u $inputNoTM -U $inputTM $fastaFile >
temp");
system ('grep "\<Y\>" temp > vysledek');
system ("rm temp");

print "Done... "; # End of SignalP

my $end = localtime;

my $durationSignal = ($end - $start)->pretty;

print "Processing time: $durationSignal\n";
```

Po zadání všech parametrů potřebných pro spuštění všech výpočtů se spouští SignalP. Nejprve mám vytvořenou proměnnou `start`, do které ukládám aktuální čas a datum (pro začátek výpočtu). Do konzole se vypíše, že běží výpočet SignalP, který se spouští pomocí příkazu, ve kterém jsou následující parametry:

- -c – „cut“: zkracuje od N-konce zvolenou délku sekvence – jelikož chci mít sekvenci kompletní, abych zkrácením nepřišla o hledaná data, je nastavena hodnota 0
- -f – „format“: dává výstupu specifický formát. Jelikož potřebuji zjistit pouze přítomnost signálního peptidu a očekává se veliké množství dat, postačí mi formát short, který vypíše pro každou sekvenci pouze jednořádkové skóre.
- -u a U cutoff – „user“: uživatelsky definovaný D-score cutoff pro non-neurální a neurální sítě (NoTM networks a TM networks) s defaultním nastavením 0,34 (sensitive) – uživatel může zadat své hodnoty. Získám tím pozitivní předpověď signálního peptidu (ve výsledku určuje odpověď Y/N)

Po dokončení výpočtu se výsledek uloží do souboru `temp` a jelikož hledám pouze ty sekvence, které obsahují signální peptid (ve výsledku označeny písmenem Y), použila jsem příkaz `grep`, pomocí jehož získám ze souboru `temp` pouze ty sekvence, které mě zajímají, ty se následně uloží do souboru `výsledek` a původní soubor `temp` se smaže. V konzoli se vypíše, že je výpočet hotový a následují ještě dvě proměnné. Jedna je `end`, do které se opět ukládá aktuální čas a datum (po dokončení výpočtu) a druhá `durationSignal`, ve které se odečtou data z proměnné `start` od dat z proměnné `end`.

Pomocí příkazu `pretty` (knihovna `Time::Piece`) se výsledná data přepíší do čitelné a srozumitelné podoby (přesná doba výpočtu) a následně se uživateli vypíše doba výpočtu s danými hodnotami.

```
# saving the sequence name into the memory

my $file = 'vysledek';
open my $info, $file or die "Could not open $file: $!";

my $counter = 0;
my @firstWord;

while( my $line = <$info>)  {
      my (@words) = split(/\s+/,$line);
      $firstWord[$counter] = $words[0];
      $counter++;
}

close $info;

system ("rm vysledek");

$counter = 0;
```

Dále bylo zapotřebí uložit si do paměti názvy sekvencí ze souboru `výsledek`, které budeme potřebovat v jednom z dalších kroků. Do proměnné `file` jsem si uložila název souboru `výsledek`. Dále jsem si otevřela proměnnou `info`, do které se okopíruje obsah souboru `výsledek`. Jelikož potřebuji pouze názvy sekvencí, nejprve jsem si vytvořila pole `firstWord`, do kterého je potom budu ukládat a počítadlo (`counter`), díky kterému se v poli mohu posouvat. Následuje `while` cyklus, ve kterém se do proměnné line uloží řádek ze souboru. Poté se do pole `words` uloží všechna slova z daného řádku a jelikož víme, že název sekvence je vždy na prvním místě, uložíme do pole `firstWord` slovo na nulté pozici z pole `words`. Poté se v `counteru` přičte číslo 1 a celý proces se opakuje, dokud neprojde celý soubor. V tomto momentě již nepotřebuji soubor `výsledek`, a proto jej mažu.

```
system ("mkdir $prefix"); # Creating directory for results
my $counterSignal = 0;
my $counterFastaFile = 0;
my $filenameResult = $prefix . "_vysledek_signalp.fsa";
      open(my $fh, '>', $filenameResult) or die;

      my $seqFile = Bio::SeqIO->new('-format' => 'fasta', '-file' =>
$fastaFile);
      while((my $seqObj = $seqFile->next_seq())){
            $counterFastaFile++;
            if($firstWord[$counterSignal]eq$seqObj->display_id) {
            print $fh ">" . $seqObj->display_id . "\n";
            print $fh "" . $seqObj->seq() . "\n";
            $counterSignal++;
            }
      }
```

```
close $fh;
```

Nyní využiji názvy sekvencí, které jsem si ukládala do pole `firstWord`. Jelikož potřebuji k dalším krokům skriptu celé sekvence a z výpočtu SignalP jsem získala pouze jejich názvy a k nim výsledné parametry (jako např. informaci o přítomnosti signálního peptidu), bylo třeba porovnat uložené názvy s původním fasta souborem a získat tak k názvům celé sekvence. Nejprve vytvářím složku s názvem již zadaného prefixu, do které se postupně uloží všechny výsledné soubory, které budou mít samozřejmě také stejný prefix. Dále proměnnou `counterSignal`, která je počítadlem pro sekvence obsahující signální peptid a `counterFastaFile`, která je také počítadlem, ale všech sekvencí ze vstupního fasta souboru. V této části kódu je využita, na začátku popisu zmíněná, knihovna bioperlu.

Vytvořila jsem si proměnnou `filenameResults`, do které se uloží výsledný název souboru s prefixem. Následuje funkce `open`, která v tomto případě obsahuje tři parametry. Prvním je tzv. logická proměnná `fh` pro zapisování konkrétních dat do souboru. Druhým parametrem je znak „>“, který značí, že se jedná o výstupní soubor, což znamená, že má právo přepsat jakýkoliv soubor se stejným názvem, jako vytvářený.

A třetím parametrem je proměnná s názvem výstupního souboru. Jako další krok je vytvoření proměnné `seqFile`, do které se pomocí knihovny bioperlu uloží původní fasta soubor. Je potřeba knihovně zadat formát čteného souboru, aby v něm mohla správně číst. Následuje `while` cyklus, ve kterém jsou procházeny všechny sekvence ze vstupního souboru a porovnávány s názvy, které byly uloženy do pole `firstWord`. Pokud je shoda, uloží se název s příslušnou sekvencí do souboru a příkazem `close` se uzavře zapisování.

```
# Blast from the results of SignalP

$start = localtime;

print "\n\n************************************";
print "\n\nProcessing Blast... Please wait...";

my $blastFile = $prefix . "_blast.out";
my $query = "blastp -query $filenameResult -db $databaseFile -
num_alignments 1 -outfmt " . '"' . "6 qseqid sseqid pident length mismatch
qend sstart send evalue bitscore" . '"' . " -out $blastFile";

system ($query);

$end = localtime;

my $durationBlast = ($end - $start)->pretty;

print "\nDone... Processing time: $durationBlast";
print "\n\n************************************";
```

Tato část kódu provádí výpočet v Blastu. Přesněji řečeno porovnává sekvence, které prošly SignalP a mají signální peptid s databází již známých a prozkoumaných sekvencí a hledá mezi nimi shodu. Opět zde mám proměnné `start` a `end`, do kterých se ukládá aktuální čas a datum (pro začátek a konec výpočtu). Dále proměnnou `blastFile`, do které se ukládá název souboru s prefixem. Pro lepší přehlednost jsem si vytvořila proměnnou `query`, ve které je uložen příkaz pro spuštění Blastu (příkaz je poměrně dlouhý, do proměnné jsem si ho dala jen pro mou lepší orientaci). Poté je pomocí příkazu `system` spuštěn obsah proměnné `query`. Tato část končí proměnnou `durationBlast`, do které je uložena doba výpočtu a v konzoli se vypíše, že je výpočet hotov a jakou dobu trval.

```
# Counters and variables for saving the name and data from the motifs

my $RKKRcounter = 0;
my @RKKR;
my @RKKRdata;
my $METHIONINcounter = 0;
my @METHIONIN;
my @METHIONINdata;
my $FURINcounter = 0;
my @FURIN;
my @FURINdata;

my $filenameRKKR;
my $filenameMETHIONIN;
my $filenameFurin;
```

V této části jsem si vytvořila `countery` jako počítadla pro nalezené patterny, dále pole s informacemi o názvu sekvencí, ve kterých byly patterny nalezeny a poté pole s daty, která byla nalezena. A nakonec proměnné s názvy výsledných souborů.

```
# RKKR-RKKR motif

if ($choice == 1 || $choice == 4) {

      $filenameRKKR = $prefix . "_RKKR";
      open(my $fh, '>', $filenameRKKR) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -format =>
"fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~
/(((RK)|(KR)){1}[FLSYCWLPHQIMTNSVADEGKR]{$rangeRKKR}((RK)|(KR)){1})/gim)
                  {

                  print $fh ">",$seq_obj->id,"\n";
                  print $fh $1,"\n";
                  $RKKR[$RKKRcounter] = $seq_obj->id;
                  $RKKRdata[$RKKRcounter] = $1;

                  $RKKRcounter++;
```

```
                }
            }

      close $fh;

      print "\nNumber of RKKR found: $RKKRcounter ";

      if ($RKKRcounter) {

      system ("mv $filenameRKKR $prefix");

      } else {

      system ("rm $filenameRKKR");

      }
}
```

Tato část kódu vyhledává patterny, neboli části sekvencí, které jsou vyznačeny (ohraničeny) určitými aminokyselinami (v tomto případě se jedná o R-arginin a K-lysin) a které mají určitou délku. Pokud zvolil uživatel na začátku při volbě motivu možnost č.1 (RKKR) nebo č.4 (všechny tři motivy), do proměnné `filenameRKKR` se uloží název výsledného souboru s prefixem a pomocí funkce `open` se otevře soubor k zapisování. Poté se pomocí knihovny bioperlu uloží výsledný fasta soubor ze SignalP a ten je poté ve `while` cyklu procházen po jednotlivých sekvencích. Pokud je v sekvenci nalezen pattern podle zadaných parametrů, je spolu s názvem sekvence, ve které byl nalezen zapsán do výstupního souboru a do vytvořených proměnných. Příkazem `close` se uzavře zapisování do souboru. Do konzole se vypíše počet nalezených patternů a následuje ještě `if` statement, který kontroluje, zda byly nějaké patterny podle zadaného motivu nalezeny. Pokud ano, výsledný soubor s daty se přesune do vytvořené složky a pokud ne, soubor se maže, jelikož je prázdný.

```
# Methionin-furin motif

if ($choice == 2 || $choice == 4) {

      $filenameMETHIONIN = $prefix . "_METHIONIN_FURIN";
      open(my $fh, '>', $filenameMETHIONIN) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -format =>
"fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~ /^[A-Z]*?(M+?[A-
Z]{$rangeMETHIONIN}R{1}[A-Z]{1}[R|K]{1}R{1})/gim)
                  {

                        print $fh ">",$seq_obj->id,"\n";
                        print $fh $1,"\n";
```

```
                        $METHIONIN[$METHIONINcounter] = $seq_obj->id;
                        $METHIONINdata[$METHIONINcounter] = $1;

                        $METHIONINcounter++;

                  }
            }

      close $fh;

      print "\nNumber of Methionin_Furin found: $METHIONINcounter ";

      if ($METHIONINcounter) {

      system ("mv $filenameMETHIONIN $prefix");

      } else {

      system ("rm $filenameMETHIONIN");

      }
}
```

Pokud si uživatel zvolil možnost č.2 nebo č.4, provede se totéž, co bylo popsáno u předešlé možnosti. Jediný rozdíl je v parametrech pro vyhledání patternu a pochopitelně v názvech (methionin – furin). Na konci se opět uzavře soubor pro zapisování, vypíše se počet nalezených patternů a pokud byl nějaký pattern nalezen, výsledný soubor se přesune do vytvořené složky. V opačném případě je opět smazán.

```
# Furin-Furin motif

if ($choice == 3 || $choice == 4) {

      $filenameFurin = $prefix . "_FURIN_FURIN";
      open(my $fh, '>', $filenameFurin) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -format =>
"fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~ /(R[A-Z][R|K]R.{$rangeFURIN}R[A-
Z][R|K]R)/gim)
                  {

                        print $fh ">",$seq_obj->id,"\n";
                        print $fh $1,"\n";

                        $FURIN[$FURINcounter] = $seq_obj->id;
                        $FURINdata[$FURINcounter] = $1;

                        $FURINcounter++;

                  }
            }

      close $fh;
      print "\nNumber of Furin_Furin found: $FURINcounter ";
```

```
	if ($FURINcounter) {

	system ("mv $filenameFurin $prefix");

	} else {

	system ("rm $filenameFurin");

	}

}
```

A zde je kód pro vybranou možnost č.3 nebo č.4, který vyhledává opět podle rozdílných parametrů (oproti předchozím motivům) a je taktéž zakončen uzavřením zapisování do souboru, vypsáním počtu nalezených patternů a přesunem výsledného souboru do vytvořené složky, pokud byl nějaký pattern nalezen. V opačném případě je opět smazán.

```
$counter = 0;
my $counterWhile = 0;

if ($SQL eq "Y" | $SQL eq "y") {

# SQL for the SignalP result

	my $fileName = $prefix . "_SQL";
	open (my $fh, '>', $fileName) or die;

	print $fh
"CREATE TABLE $tableName (
ID INT NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $tableName (SEQ_ID, SEQ) VALUES\n";

my $seqFile = Bio::SeqIO->new(-file => $filenameResult, -format => "fasta"
);
while((my $seqObj = $seqFile->next_seq())){
	if ($counterWhile < $counterSignal - 1) {
		print $fh "('" . $seqObj->display_id . "',";
		print $fh "'" . $seqObj->seq() . "'),\n";
		$counterWhile++;
	} else {
		print $fh "('" . $seqObj->display_id . "',";
		print $fh "'" . $seqObj->seq() . "')\n";
		$counterWhile = 0;
	}
}

print $fh ";\n";
```

Pokud uživatel při dotázání zadal, že si přeje vytvořit SQL dotaz, začne se provádět tato část kódu. Mám zde vytvořené dvě proměnné – `counter` a `counterWhile`.

Následuje `if` statement, který kontroluje, zda je v proměnné SQL uloženo Y/y (vstup z klávesnice při dotazu na vytvoření SQL). Dále se opět vytváří proměnná, do které je uložen název výsledného souboru s prefixem a otvírá se soubor k zapisování. Následují SQL příkazy, které jsou vygenerovány do výstupního souboru. Kód prochází pomocí knihovny bioperlu fasta soubor (výsledný soubor po SignalP). Ve `while` cyklu se prochází jednotlivé sekvence, které jsou počítány proměnnou `counterWhile` a proměnná `counterSignal`, kde je uložený počet výsledných sekvencí ze signalp slouží ke kontrole, zda cyklus došel k poslední sekvenci. Každá sekvence je zapisována do výstupního souboru. Pokud dojde `counterWhile` k poslední sekvenci, kód pokračuje k části `else`, kde už za poslední zapisovanou sekvencí nebude čárka.

```
if (($choice == 1 || $choice == 4) && $RKKRcounter > 0) {

# SQL for RKKR table

      print $fh
"CREATE TABLE $filenameRKKR (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameRKKR (SEQ_ID, SEQ) VALUES\n";

until($counter == $RKKRcounter) {

      if ($counterWhile < $RKKRcounter - 1) {
            print $fh "('" . "$RKKR[$counter]" . "',";
            print $fh "'" . "$RKKRdata[$counter]" . "'),\n";
            $counter++;
            $counterWhile++;
      } else {
            print $fh "('" . "$RKKR[$counter]" . "',";
            print $fh "'" . "$RKKRdata[$counter]" . "')\n";
            $counter++;
            $counterWhile = 0;
      }
}

print $fh ";\n";

$counter = 0;
}
```

Pokud byla u výběru motivu vybrána možnost č.1, nebo č.4, a zároveň je hodnota `RKKRcounteru` (obsahuje počet nalezených patternů daného motivu) větší než nula, do souboru se zapisují SQL příkazy. Hodnota `RKKRcounteru` musí být větší než nula, protože nechci vytvářet prázdnou tabulku, do které se nebude mít co vložit.

V tomto případě se prochází nalezené výsledky podle motivu RKKR, které jsou opět počítány proměnnou counterWhile.

Do výstupního souboru se zapisuje název sekvence, ve které byl pattern nalezen (proměnná RKKR) a samotný pattern (proměnná RKKRdata). Pokud dojde RKKRcounter k poslednímu patternu, opět přechází do části else, ve které už za pattern nevkládá čárku.

```
if (($choice == 2 || $choice == 4) && $METHIONINcounter > 0) {

# SQL for Methionin table

      print $fh
"CREATE TABLE $filenameMETHIONIN (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameMETHIONIN (SEQ_ID, SEQ) VALUES\n";

until($counter == $METHIONINcounter) {
      if($counterWhile < $METHIONINcounter - 1) {
            print $fh "('" . "$METHIONIN[$counter]" . "',";
            print $fh "'" . "$METHIONINdata[$counter]" . "'),\n";
            $counter++;
            $counterWhile++;
      } else {
            print $fh "('" . "$METHIONIN[$counter]" . "',";
            print $fh "'" . "$METHIONINdata[$counter]" . "')\n";
            $counter++;
            $counterWhile = 0;
      }
}

print $fh ";\n";

$counter = 0;
}
```

Zde je postup stejný, jako u prvního motivu, jen se pracuje s daty a s proměnnými z druhého motivu (methionin – furin).

```
if (($choice == 3 || $choice == 4) && $FURINcounter > 0) {

# SQL for Furin table

      print $fh
"CREATE TABLE $filenameFurin (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameFurin (SEQ_ID, SEQ) VALUES\n";

until($counter == $FURINcounter) {
```

```
      if ($counterWhile < $FURINcounter - 1) {
            print $fh "('" . "$FURIN[$counter]" . "',";
            print $fh "'" . "$FURINdata[$counter]" . "'),\n";
            $counter++;
            $counterWhile++;
      } else {
            print $fh "('" . "$FURIN[$counter]" . "',";
            print $fh "'" . "$FURINdata[$counter]" . "')\n";
            $counter++;
            $counterWhile = 0;
      }
}

print $fh ";";

}
      close $fh;

      system ("mv $fileName $prefix");

}

system ("mv $filenameResult $prefix");
system ("mv $blastFile $prefix");
```

Zde je postup opět stejný, jako u prvního a druhého motivu, jen se pracuje s daty a s proměnnými ze třetího motivu (furin – furin). Příkazem `close` se uzavře zapisování do souboru a výsledný soubor je přesunut do již vytvořené složky. Poté se do složky přesunou i výsledné soubory z předchozích výpočtů.

```
my $summaryFileName = "Summary_" . $prefix;

open($fh, '>', $summaryFileName) or die;

print $fh "
**********  Summary report from $testDate  **********

Name of the tested fasta file: $fastaFile

Number of sequences tested: $counterFastaFile

**************************************************

SignalP processing time - $durationSignal

Number of sequences containing signal peptide: $counterSignal

**************************************************

Name of the used database - $databaseFile

Blast processing time - $durationBlast

**************************************************

Motifs used in test:\n\n";

if ($choice == 1) {
print $fh "********** RKKR-RKKR **********
```

```
Range: $rangeRKKR
Found: $RKKRcounter";
}
if ($choice == 2) {
print $fh "********** Methionin-Furin **********
Range: $rangeMETHIONIN
Found: $METHIONINcounter";
}
if ($choice == 3) {
print $fh "********** Furin-Furin **********
Range: $rangeFURIN
Found: $FURINcounter" ;
}
if ($choice == 4) {
print $fh "********** RKKR-RKKR **********
Range: $rangeRKKR
Found: $RKKRcounter

********** Methionin-Furin **********
Range: $rangeMETHIONIN
Found: $METHIONINcounter

********** Furin-Furin **********
Range: $rangeFURIN
Found: $FURINcounter";
}

close $fh;

system ("mv $summaryFileName $prefix");


print "\n\nTesting ended...\n";
```

Jako poslední přichází na řadu výstupní soubor se souhrnnými informacemi o průběhu všech výpočtů, které skript provedl. Vytvořila jsem si proměnnou `summaryFileName`, do které ukládám název výsledného souboru s prefixem a otevírám soubor k zapisování. Ten bude obsahovat:

- Datum
- Název vstupního fasta souboru, který byl uložen do proměnné `fastaFile`
- Počet testovaných sekvencí (`counterFastaFile`)
- Dobu trvání výpočtu SignalP (`durationSignal`)
- Počet sekvencí obsahujících signální peptid (`countersignal`)
- Název databáze pro výpočet Blast (`databaseFile`)
- Dobu trvání výpočtu Blast (`durationBlast`)
- Které motivy byly použity pro výpočet, jaký byl zadán range (např.`rangeRKKR`) a kolik jich bylo nalezeno (např.`RKKRcounter`)

Poté se uzavírá zápis do souboru, výsledný soubor se přesouvá do složky a uživateli se vypíše do konzole, že výpočet skončil.

6.2. SQL

Pokud uživatel zvolí vytvoření SQL, bude vygenerován SQL dotaz do výstupního souboru pro přidání do vlastní databáze.

```
CREATE TABLE test1 (
ID INT NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);
```

První část je příkaz pro vytvoření tabulky. Název tabulky pro výsledky SignalP si uživatel zvolil na začátku skriptu. Pro ostatní tabulky jsou generovány podle prefixu a použitého motivu. Příkaz pro vytvoření tabulky je `CREATE TABLE`, ve kterém je potřeba definovat položky tabulky (sloupce). Sloupce určují název (`ID`, `SEQ_ID`, `SEQ`) a datový typ (`INT`, `VARCHAR`, `MEDIUMTEXT`). Dále jsou specifikovány vlastnosti daného sloupce, `NOT NULL`, kde sloupec musí vždy obsahovat data a `AUTO_INCREMENT` kde bude automaticky přičítat hodnotu +1. Poslední řádek zde určuje primární klíč pro tuto konkrétní tabulku (`ID` a `SEQ_ID`), primární klíče musí vždy mít jedinečný název.

- ID – unikátní identifikátor pro každou sekvenci; typ INT
- SEQ_ID – název sekvence; typ VARCHAR o maximální velikosti
- SEQ – sekvence; typ MEDIUMTEXT
- PRIMARY KEY – ID a SEQ_ID jsou primárními klíči (jsou unikátní a nikdy nesmí být 2 stejné)

```
INSERT INTO test1 (SEQ_ID, SEQ) VALUES
('gi|677581542|ref|XP_009070020.1|','MLQRRGLLWLAVLITLWVSSNAQDGDKEEE
TTFDLLQVSNINRKTIGAKLFRGPDPTIPAYRFIRFDHIPPCKPEKLKKIIKLIRQNEGFILSATLR
'),
('gi|677581980|ref|XP_009081647.1|','GKIRLATLELGCLLLKQLVFSKNSSIIKDV
HLACLEGAREESVHLLRRFYKGEEIFLDMFEDEYRSMTIKPMNVEYLMMDASILLPPTGTPLTGIDF
');
```

V další části je příkaz `INSERT INTO`, který vkládá data do tabulky. Data pro vložení jsou generována z výsledků ze SignalP.

Tímto způsobem jsou vytvořeny příkazy i pro motivy RKKR, METHIONIN a FURIN, ale pouze v případě, že jsou pro jednotlivé motivy dostupná data. (25)

7. Testování

K testování skriptu byl použit FASTA soubor, který obsahoval 14 535 sekvencí, z čehož 3 589 obsahovalo signální peptid. V těch byly poté vyhledávány pattern match. Pro tento test byla zvolená čtvrtá možnost, tedy všechny 3 motivy a všem byl zadán range 32. Celkem bylo nelezeno:

- u motivu RKKR-RKKR 80 patternů
- u motivu Methionin-Furin 97 patternů
- u motivu Furin-Furin 29 patternů

Zároveň proběhlo porovnání našich nalezených signálních sekvencí s databází jiných zvířat (v tomto případě se jednalo o hmyzí neuropeptidy), kterou si uživatel zadal spolu se vstupním souborem jako parametr (ukázka výsledků výpočtu Blast viz příloha C).

Takto veliká data zpracovával kód celkem přes 1 hodinu a 28 minut (podrobné časy výpočtů viz příloha H). Vzhledem k tomu, že bylo zapotřebí otestovat, jak fungují všechny kroky, byl záměrně vybrán takový vstupní soubor, který ověří funkčnost všech částí skriptu.

Testování proběhlo na serveru CentOS 7, v EMBL GeneCore v Heidelbergu a prokázalo, že skript dokáže vyhledat data dle zadaných parametrů/požadavků. Ukázky všech výsledků z daného testu viz Přílohy.

8. Závěr

V rámci bakalářské práce byl napsán skript, který zjednodušuje vyhledávání kandidátních sekvencí, které jsou potřeba k predikci neuropeptidů. Kód provádí na základě zadaného vstupního souboru několik výpočtů, díky kterým dostaneme vhodné kandidátní sekvence pro další zkoumání. Sekvence jsou zároveň porovnány pomocí Blastu s databází jiných zvířat, kterou si sám uživatel zadal. Výstupem je složka s jednotlivými soubory, které obsahují výsledky jednotlivých kroků, včetně dotazu SQL pro případné vložení dat do vlastní databáze a včetně souhrnného souboru s podrobnostmi o výpočtech.

Když uživatel získá takto připravené a profiltrované sekvence, zbývá mu už jen vložit je do NeuroPredu, který primárně poslouží k „profesionálnímu“ vyhodnocení predikce neuropeptidů a k výpočtu hmoty peptidů. Do skriptu tento krok nemohl být zahrnut, protože NeuroPred funguje bohužel jen jako webové rozhraní a nemá k dispozici žádný lokálně spustitelný kód, jako např. SignalP.

Citovaná literatura

1. **Schoofs, Liliane, De Loof, Arnold a Van Hiel, Matthias Boris.** *Neuropeptides as Regulators of Behavior in Insects.* místo neznámé : Annual Review of Entomology, 2017. Annual Review of Entomology.

2. **Altstein, Miriam a Nässel, Dick R.** *Neuropeptide signaling in insects.* místo neznámé : Advances in Experimental Medicine and Biology, 2010. Advances in Experimental Medicine and Biology. 0065-2598.

3. **Mains, Richard E. a Eipper, Betty A.** *The Neuropeptides.* místo neznámé : Basic Neurochemistry: Molecular, Cellular and Medical Aspects. 6th edition, 1999. Basic Neurochemistry: Molecular, Cellular and Medical Aspects. 6th edition.

4. **Švandová, Ivana.** *Neuropřenašeče: neuropeptidy a puriny.* Přírodovědecká fakulta, Univerzita Karlova. Přednáška.

5. **Burbach, J. Peter H.** *What are neuropeptides?* místo neznámé : Methods in Molecular Biology (Clifton, N.J.), 2011. Methods in Molecular Biology (Clifton, N.J.).

6. **Duckert, Peter, Brunak, Søren a Blom, Nikolaj.** *Prediction of proprotein convertase cleavage sites.* 01 2004. Protein Engineering, Design and Selection, Sv. 17. 1741-0134, 1741-0126.

7. **Altschul, S.F., a další.** *Basic local alignment search tool.* místo neznámé : Journal of Molecular Biology, 1990.

8. **NCBI.** BLAST: Basic Local Alignment Search Tool. *National Center for Biotechnology Information.* [Online] [Citace: 10. 4 2019.] http://blast.ncbi.nlm.nih.gov.

9. **Southey, Bruce R., a další.** *NeuroPred: a tool to predict cleavage sites in neuropeptide precursors and provide the masses of the resulting peptides.* místo neznámé : Nucleic Acids Research, 2006. Nucleic Acids Research.

10. **UIUC NIDA Center for Neuroproteomics.** [Online] http://stagbeetle.animal.uiuc.edu/neuropred.html.

11. **Ofer, Dan a Linial, Michal.** *NeuroPID: A Predictor for Identifying Neuropeptide Precursors from Metazoan Proteomes.* místo neznámé : Bioinformatics, 2013. Bioinformatics.

12. **National Center for Biotechnology Information.** National Center for Biotechnology Information. [Online] 1988. https://www.ncbi.nlm.nih.gov/.

13. **NCBI.** NCBI Reference Sequence Database. [Online] https://www.ncbi.nlm.nih.gov/refseq/.

14. —. GenBank. [Online] https://www.ncbi.nlm.nih.gov/genbank/.

15. **Bairoch, Amos a Apweiler, Rolf.** *The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000.* místo neznámé : Nucleic Acids Research, 2000. Nucleic Acids Research. 0305-1048.

16. **Sigrist, Christian J. A., a další.** New and continuing developments at PROSITE. *Nucleic Acids Research.* místo neznámé : Nucleic Acids Research, 2012. 1362-4962.

17. **Prosite.** PROSITE: Database of protein domains, families and functional sites. [Online] https://prosite.expasy.org/.

18. **DTU Bioinformatics.** Manual page for signalp. *DTU Bioinformatics.* [Online] http://www.cbs.dtu.dk/cgi-bin/nph-runsafe?man=signalp.

19. **Nielsen, Henrik.** *Predicting Secretory Proteins with SignalP.* místo neznámé : Methods in Molecular Biology (Clifton, N.J.), 2017. Methods in Molecular Biology (Clifton, N.J.).

20. **DTU Bioinformatics.** SignalP 4.1 Server. *DTU Bioinformatics.* [Online] http://www.cbs.dtu.dk/services/SignalP-4.1/.

21. **Perl Programming Documentation.** Perl programming documentation. [Online] https://perldoc.perl.org/index.html.

22. **Perl.** The Perl Programming Language. [Online] https://www.perl.org/.

23. **Tutorials Point.** Learn Perl Programming. [Online] https://www.tutorialspoint.com/perl/index.htm.

24. **BioPerl.** BioPerl. [Online] https://bioperl.org/index.html.

25. **MySQL.** MySQL :: MySQL 8.0 Reference Manual. [Online] https://dev.mysql.com/doc/refman/8.0/en/.

Seznam obrázků

Přílohy

A) Zdrojový kód
B) Výstup SignalP (protříděný)
C) Výstup Blast
D) Výstup FURIN-FURIN
E) Výstup METHIONIN-FURIN
F) Výstup RKKR
G) Výstup SQL
H) Výstup Summary
I) Vývojový diagram skriptu
J) Vstupní fasta soubor (pouze CD)
K) Blast databáze (pouze CD)

Kompletní znění všech příloh viz CD

A) Zdrojový kód

```
#!/usr/bin/perl
use strict;
use warnings;
use Bio::SeqIO;
use Time::Piece;

# Controlling if the user entered two arguments

if ($#ARGV != 1) {

print "\n\n************************************

The skript needs fasta file and Blast database name as an argument
for running.
for example: ./skript fastaFileName databaseName

************************************\n\n";
     exit;
}

my $testDate = localtime();

print "\n\nStart date and time: $testDate";

my $fastaFile = $ARGV[0]; # Fasta file name entered as an argument
my $databaseFile = $ARGV[1]; # Blast database
print "\n\n************************************

Successfully loaded fasta file - $fastaFile

Successfully loaded database file - $databaseFile

************************************\n\n";

#*********************************************************#

# sensitivity setup for SignalP

print "Enter a D-cutoff value for noTM networks (default = 0.34):
";
my $inputNoTM = <STDIN>;
chomp $inputNoTM;
print "\n\nEnter a D-cutoff value for TM networks (default = 0.34):
";
my $inputTM = <STDIN>;
chomp $inputTM;

# If user do not enter any value the skript uses the default one

if ($inputNoTM eq '') {

$inputNoTM = 0.34;
```

```
}
if ($inputTM eq '') {

$inputTM = 0.34;

}

#*********************************************************#

# Creating new fasta file with data from SignalP
print "\n\nThe results will be moved to the directory named as an
entered prefix.

Enter prefix: ";
my $prefix = <STDIN>;
chomp $prefix;

#*********************************************************#

# Motif selection - at least one must be selected

my $choice;
my $rangeRKKR;
my $rangeMETHIONIN;
my $rangeFURIN;
my $input;

while (1) {

      print "\n\n************************************";
      print "\n\nWhich of the motifs do you want to use?\n
            1 - RK/KR - RK/KR\n
            2 - METHIONIN - FURIN\n
            3 - FURIN - FURIN\n
            4 - All of them\n
            Enter value (1/2/3/4): ";

      $choice = <STDIN>;
      chomp $choice;

      if ($choice == 1) {

            print "\n\n************************************\n\n";
            print "Range for RK/KR - RK/KR: ";
            $input = <STDIN>;
            chomp $input;
            $rangeRKKR = $input;
            last;

      } if ($choice == 2) {

            print "\n\n************************************\n\n";
            print "Range for METHIONIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
```

```
            $rangeMETHIONIN = $input;
            last;

      } if ($choice == 3) {

            print "\n\n*************************************\n\n";
            print "Range for FURIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeFURIN = $input;
            last;

      } if ($choice == 4) {

            print "\n\n*************************************\n";
            print "\nRange for RK/KR - RK/KR: ";
            $input = <STDIN>;
            chomp $input;
            $rangeRKKR = $input;

            print "\nRange for METHIONIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeMETHIONIN = $input;

            print "\nRange for FURIN - FURIN: ";
            $input = <STDIN>;
            chomp $input;
            $rangeFURIN = $input;
            last;

      }

      print "\nWrong value!";

}

#*********************************************************#

# SQL

print "\n\n*************************************";

print "\n\nDo you want create an SQL file? (N/y): ";
my $SQL = <STDIN>;
chomp $SQL;
my $tableName;

if ($SQL eq "Y" | $SQL eq "y") {
      print "\n\nEnter a table name: ";
      $tableName = <STDIN>;
      chomp $tableName;
}
#*********************************************************#

print "\n\n*************************************";
```

```
my $start = localtime;

print "\n\nProcessing SignalP... Please wait...\n"; # Start of
SignalP

system ("./signalp -c 0 -f short -u $inputNoTM -U $inputTM
$fastaFile > temp");
system ('grep "\<Y\>" temp > vysledek');
system ("rm temp");

print "Done... "; # End of SignalP

my $end = localtime;

my $durationSignal = ($end - $start)->pretty;

print "Processing time: $durationSignal\n";

#*********************************************************#

# saving the sequence name into the memory

my $file = 'vysledek';
open my $info, $file or die "Could not open $file: $!";

my $counter = 0;
my @firstWord;

while( my $line = <$info>)  {
     my (@words) = split(/\s+/,$line);
     $firstWord[$counter] = $words[0];
     $counter++;
}

close $info;

system ("rm vysledek");

$counter = 0;

#*********************************************************#



system ("mkdir $prefix"); # Creating directory fo results
my $counterSignal = 0;
my $counterFastaFile = 0;
my $filenameResult = $prefix . "_vysledek_signalp.fsa";
     open(my $fh, '>', $filenameResult) or die;

     my $seqFile = Bio::SeqIO->new('-format' => 'fasta', '-file' =>
$fastaFile);
     while((my $seqObj = $seqFile->next_seq())){
          $counterFastaFile++;
          if($firstWord[$counterSignal]eq$seqObj->display_id) {
```

```
            print $fh ">" . $seqObj->display_id . "\n";
            print $fh "" . $seqObj->seq() . "\n";
            $counterSignal++;
            }
      }

close $fh;

#*********************************************************#

# Blast from the results of SignalP

$start = localtime;

print "\n\n************************************";
print "\n\nProcessing Blast... Please wait...";

my $blastFile = $prefix . "_blast.out";
my $query = "blastp -query $filenameResult -db $databaseFile -
num_alignments 1 -outfmt " . '"' . "6 qseqid sseqid pident length
mismatch qend sstart send evalue bitscore" . '"' . " -out
$blastFile";

system ($query);

$end = localtime;

my $durationBlast = ($end - $start)->pretty;

print "\nDone... Processing time: $durationBlast";
print "\n\n************************************";

#*********************************************************#

# Counters and variables for saving the name and data from the
motifs

my $RKKRcounter = 0;
my @RKKR;
my @RKKRdata;
my $METHIONINcounter = 0;
my @METHIONIN;
my @METHIONINdata;
my $FURINcounter = 0;
my @FURIN;
my @FURINdata;

my $filenameRKKR;
my $filenameMETHIONIN;
my $filenameFurin;

#*********************************************************#

# RKKR-RKKR motif

if ($choice == 1 || $choice == 4) {
```

```
      $filenameRKKR = $prefix . "_RKKR";
      open(my $fh, '>', $filenameRKKR) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -
format => "fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~
/(((RK)|(KR)){1}[FLSYCWLPHQIMTNSVADEGKR]{$rangeRKKR}((RK)|(KR)){1})
/gim)
                  {

                  print $fh ">",$seq_obj->id,"\n";
                  print $fh $1,"\n";
                  $RKKR[$RKKRcounter] = $seq_obj->id;
                  $RKKRdata[$RKKRcounter] = $1;

                  $RKKRcounter++;

                  }
            }

      close $fh;

      print "\nNumber of RKKR found: $RKKRcounter ";

      if ($RKKRcounter) {

      system ("mv $filenameRKKR $prefix");

      } else {

      system ("rm $filenameRKKR");

      }
}

#*********************************************************#

# Methionin-furin motif

if ($choice == 2 || $choice == 4) {

      $filenameMETHIONIN = $prefix . "_METHIONIN_FURIN";
      open(my $fh, '>', $filenameMETHIONIN) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -
format => "fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~ /^[A-Z]*?(M+?[A-
Z]{$rangeMETHIONIN}R{1}[A-Z]{1}[R|K]{1}R{1})/gim)
                  {
```

```
                        print $fh ">",$seq_obj->id,"\n";
                        print $fh $1,"\n";

                        $METHIONIN[$METHIONINcounter] = $seq_obj->id;
                        $METHIONINdata[$METHIONINcounter] = $1;

                        $METHIONINcounter++;

                  }
            }

      close $fh;

      print "\nNumber of Methionin_Furin found: $METHIONINcounter ";

      if ($METHIONINcounter) {

      system ("mv $filenameMETHIONIN $prefix");

      } else {

      system ("rm $filenameMETHIONIN");

      }
}

#*********************************************************#

# Furin-Furin motif

if ($choice == 3 || $choice == 4) {

      $filenameFurin = $prefix . "_FURIN_FURIN";
      open(my $fh, '>', $filenameFurin) or die;

      my $seqio_obj = Bio::SeqIO->new(-file => $filenameResult, -
format => "fasta" );
            while (my $seq_obj = $seqio_obj->next_seq)
            {

                  if ($seq_obj->seq =~ /(R[A-
Z][R|K]R.{$rangeFURIN}R[A-Z][R|K]R)/gim)
                  {

                        print $fh ">",$seq_obj->id,"\n";
                        print $fh $1,"\n";

                        $FURIN[$FURINcounter] = $seq_obj->id;
                        $FURINdata[$FURINcounter] = $1;

                        $FURINcounter++;

                  }
            }
```

```
      close $fh;
      print "\nNumber of Furin_Furin found: $FURINcounter ";

      if ($FURINcounter) {

      system ("mv $filenameFurin $prefix");

      } else {

      system ("rm $filenameFurin");

      }

}

#*********************************************************#

$counter = 0;
my $counterWhile = 0;

if ($SQL eq "Y" | $SQL eq "y") {

# SQL for the SignalP result

      my $fileName = $prefix . "_SQL";
      open (my $fh, '>', $fileName) or die;

      print $fh
"CREATE TABLE $tableName (
ID INT NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $tableName (SEQ_ID, SEQ) VALUES\n";

my $seqFile = Bio::SeqIO->new(-file => $filenameResult, -format =>
"fasta" );
while((my $seqObj = $seqFile->next_seq())){
      if ($counterWhile < $counterSignal - 1) {
            print $fh "('" . $seqObj->display_id . "',";
            print $fh "'" . $seqObj->seq() . "'),\n";
            $counterWhile++;
      } else {
            print $fh "('" . $seqObj->display_id . "',";
            print $fh "'" . $seqObj->seq() . "')\n";
            $counterWhile = 0;
      }
}

print $fh ";\n";

#*********************************************************#

if (($choice == 1 || $choice == 4) && $RKKRcounter > 0) {
```

```
# SQL for RKKR table

     print $fh
"CREATE TABLE $filenameRKKR (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameRKKR (SEQ_ID, SEQ) VALUES\n";

until($counter == $RKKRcounter) {

     if ($counterWhile < $RKKRcounter - 1) {
          print $fh "('" . "$RKKR[$counter]" . "',";
          print $fh "'" . "$RKKRdata[$counter]" . "'),\n";
          $counter++;
          $counterWhile++;
     } else {
          print $fh "('" . "$RKKR[$counter]" . "',";
          print $fh "'" . "$RKKRdata[$counter]" . "')\n";
          $counter++;
          $counterWhile = 0;
     }
}

print $fh ";\n";

$counter = 0;
}

#*********************************************************#

if (($choice == 2 || $choice == 4) && $METHIONINcounter > 0) {

# SQL for Methionin table

     print $fh
"CREATE TABLE $filenameMETHIONIN (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameMETHIONIN (SEQ_ID, SEQ) VALUES\n";

until($counter == $METHIONINcounter) {
     if($counterWhile < $METHIONINcounter - 1) {
          print $fh "('" . "$METHIONIN[$counter]" . "',";
          print $fh "'" . "$METHIONINdata[$counter]" . "'),\n";
          $counter++;
          $counterWhile++;
     } else {
```

```
            print $fh "('" . "$METHIONIN[$counter]" . "',";
            print $fh "'" . "$METHIONINdata[$counter]" . "')\n";
            $counter++;
            $counterWhile = 0;
      }
}

print $fh ";\n";

$counter = 0;
}

#*********************************************************#

if (($choice == 3 || $choice == 4) && $FURINcounter > 0) {

# SQL for Furin table

      print $fh
"CREATE TABLE $filenameFurin (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO $filenameFurin (SEQ_ID, SEQ) VALUES\n";

until($counter == $FURINcounter) {
      if ($counterWhile < $FURINcounter - 1) {
            print $fh "('" . "$FURIN[$counter]" . "',";
            print $fh "'" . "$FURINdata[$counter]" . "'),\n";
            $counter++;
            $counterWhile++;
      } else {
            print $fh "('" . "$FURIN[$counter]" . "',";
            print $fh "'" . "$FURINdata[$counter]" . "')\n";
            $counter++;
            $counterWhile = 0;
      }
}

print $fh ";";

}
      close $fh;

      system ("mv $fileName $prefix");

}

system ("mv $filenameResult $prefix");
system ("mv $blastFile $prefix");

#*********************************************************#
```

```
my $summaryFileName = "Summary_" . $prefix;

open($fh, '>', $summaryFileName) or die;

print $fh "
**********  Summary report from $testDate  **********

Name of the tested fasta file: $fastaFile

Number of sequences tested: $counterFastaFile

**************************************************

SignalP processing time - $durationSignal

Number of sequences containing signal peptide: $counterSignal

**************************************************

Name of the used database - $databaseFile

Blast processing time - $durationBlast

**************************************************

Motifs used in test:\n\n";

if ($choice == 1) {
print $fh "********** RKKR-RKKR **********
Range: $rangeRKKR
Found: $RKKRcounter";
}
if ($choice == 2) {
print $fh "********** Methionin-Furin **********
Range: $rangeMETHIONIN
Found: $METHIONINcounter";
}
if ($choice == 3) {
print $fh "********** Furin-Furin **********
Range: $rangeFURIN
Found: $FURINcounter" ;
}
if ($choice == 4) {
print $fh "********** RKKR-RKKR **********
Range: $rangeRKKR
Found: $RKKRcounter

********** Methionin-Furin **********
Range: $rangeMETHIONIN
Found: $METHIONINcounter

********** Furin-Furin **********
Range: $rangeFURIN
Found: $FURINcounter";
}
```

```
close $fh;

system ("mv $summaryFileName $prefix");


print "\n\nTesting ended...\n";
```

B) Výstup SignalP (protříděný)

>AAK84160.1
MFSPLLFFAVSISCVLANSNEIKDGGSDRGAHSDRAGLWFGPRLGKRSLRISTEDNRQAFFKLLEAA
DALKYYYDRLPYEMQADEPETRVTKKVIFTPKLGRSLAYDDKVFENVEFTPRLGRRLADDMPATPAD
QELYRPDPDQIDSRTKYFSPRLGRTMNFSPRLGRELSYDMLPSKLRLVRSTNRTQST
>sp|Q95P48.1|PBAN_SPOLI
MFSPLLFFAVSISCVLANSNEIKDGGSDRGAHSDRAGLWFGPRLGKRSLRISTEDNRQAFFKLLEAA
DALKYYYDRLPYEMQADEPETRVTKKVIFTPKLGRSLAYDDKVFENVEFTPRLGRRLADDMPATPAD
QELYRPDPDQIDSRTKYFSPRLGRTMNFSPRLGRELSYDMLPSKLRLVRSTNRTQST
>ABD52277.1
MTLSAPPIDEFEDPFVVMNTTNVSSHPAAYDEPYTLDLVVPLTVTYVVIFVAGILGNTSTCVVIARN
RSMHTATNFYLFSLAISDLILLVCGLPFEVHRLWNPDTYPLGEAHCIAIGLASETSANATVLTITAF
TVERYIAICRPFMSHTMSKLSRAVRFIIAIWVFALCTAVPQAMQFGIVSYVDNGQNVSACTVKGVGV
HQVFVISSFVFFVVPMSMISVLYALIGIKLRTSRVLHPVKKLSVDSNERASGQMQYRNGASQRRVIR
MLVAVALSFFICWAPFHVQRLLAIYGKSLEHPSDTFYLVYIVLTFLSGVLYFLSTAINPFLYNIMSN
KFRNAFKVSFCYIHD
>AFW19647.1
MSPSFVFALAVCGFAAICSLPVTNAQALVSPYESSAAADSQNGWGALGGLYALLAQHDALGGHALAR
KSVRSPSRRLRFGRRSDPDMPPQAPLDEMNELLSLREVRTPVRLRFGRRSEERAVPHIFPQEF
>AAR03495.1
MARVTLALAVSLAAYAAYYYVHIAEAQPIGSLVLAAPHSQQQQPGSSTSDEATINHLQQQHQRLKDT
NVYRARSKMRPHDRKYPGVIGAYEAYRRTVQGPQLMQRNPATADRFADDPGVDEQDQMRFSLEGFLT
GARTPTLLNDDEEEEEDEDHEQGGDGLVKRFDDYGHMRFGKRGGEGDQFDDYGHMRFGR
>AAT81601.1
TMASGTFTQRLLVALMIFALIADLSTLVAARPQDSDAASVAAAIRYLQELETKHAQHARPRFGKRGG
YLNPAIFGQDEQENLYRLIGRIQHFRDEQLPTNI
>sp|Q7PTL2.2|PBAN_ANOGA
MSRFYFFFNLICLYLAIKSALSAELDTNDQKYADLRTTGRGESPDSTGPDSDTLRRDDGAEGLNKRA
AAMWFGPRLGKRTIAADLHDDLVEEFDAEPLGYAGEPPQKLATELVQGAPYMVLLVTAKPRKPQPIF
YHTTSPRLGRRDSVGENHQRPPFAPRLGRNLPFSPRLGRSYNGGYPLPFQFAY
>ABD96048.1
MYRINLTTFTLLLVLAVGSLMSESLHPSDGAINDLYEYLLQREYAAPVSYADHQIKRKAVRSPSLRL
RFGRRSDPSVPLRPEEDELIDQKAIRAPQLRLRFGRNDPLWTSFNENALLEENFEKRAPSQRLRWGR
SNLFGNLVNQFQQDDVMQQKTIRAPQLRLRFGRTDPSWAMYNEHQLTTGQQAQPANEASEKRAPTQR
LRWGRSDPALAKDSSEDKALDVEESENTNADDK
>ABD96049.1
MLTPVSGLVSLVGAVTVATATSTSPAAMASLVLDHTELPLAGTIPPAALMPARVLLPSNATNLTLTL
EELLRPNSSTVAPPNGDNDIIFSNKLVQIVFCVLYSSIFVLGVFGNVLVCYVVFRNKAMQTVTNLFI
TNLALSDILLCVLAVPFTPSYTFMRRWVFGKLLCHTVPLAQGCSVYISTLTLTSIAIDRFFVIIYPF
HPRMKLSTCITIIVLIWSFAIMVTMPYGLYMKLHGVALNGTDNATGPLSSAMYCEELWPSEEMRKTF
SIVTSILQFVLPFIIMAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKRTNRMLISMVAIFGIS
WLPLNVVNMCNDFNSDINSWRFYNLIFFIAHLTAMSSTCYNPFLYAWLNDNFRKEFKQVLPCFDPSR
GRAGTVGGNRGAGGGWRSERTCNGNNDTVQETLIPSSQVLPSSRSSQPHQQPQQLSSSSSQGQGNQP
TVDSILLSEVVPPLSLPPPLTGAMLPSVQSAETVILPSGVLETPFDVQLIPSVGAPVSNGTSDATNL
TPSAGVRNGLNHACTNPKLSSLILINDGTTADSKVPAIL
>sp|Q7Q7R8.3|NPF_ANOGA
MASGTFTQRLLVALMIFALIADLSTLVAARPQDSDAASVAAAIRYLQELETKHAQHARPRFGKRGGY
LNPAIFGQDEQEVDWQDSTFSR
>sp|A0SIF1.1|SNPF_ANOGA
MYRINLTTFTLLLVLAVGSLMSESLHPSDGAINDLYEYLLQREYAAPVSYADHQIKRKAVRSPSLRL
RFGRRSDPSVPLRPEEDELIDQKAIRAPQLRLRFGRNDPLWTSFNENALLEENFEKRAPSQRLRWGR
SNLFGNLVNQFQQDDVMQQKTIRAPQLRLRFGRTDPSWAMYNEHQLTTGQQAQPANEASEKRAPTQR
LRWGRSDPALAKDSSEDKALDVEESENTNADDK

>sp|A0NDK8.1|CORZ_ANOGA
MLHTRTIALLLVGLVVLVNAQTFQYSRGWTNGKRSPLSSSSSSPSSSAAMEPLTANQLLASALSSGG
LNSLKPSEKALLRRFLRNPCDLRVASLLAAAHPTKELFPLAGNSFDSAESAGAAFVLPPFLMDPDES
NGGIGGSNLANGRSMEDELRFKRGTATGFSDHRQKIA
>ACH99845.1
MAKVSAACVLLVCLWLRASAALPAFEDDRDLDRELYIRQLAEWLADQSTDFLNELTSFPPCRPCSSY
EHTRQPIAVVPRAPYAKRNSELINSLLSLPKTMNDAGK
>AFE02890.1
MLHTRTIALLLVGLVVLVNAQTFQYSRGWTNGKRSPSSSSSSSPSSSAAMEPLTANQLLASALSPGG
LNSLKPSEKALLRRFLRNPCDLRVASLLAAAHPTKELFPLAGNSFDSAESAGAAFVLPPFLMDPDES
NGGIGGSNLANGRSMEDELRFKRGTATGFSDHRQKIA
>AFE02891.1
MLHTRTIALLLVGLVVLVNAQTFQYSRGWTNGKRSPSSSSSSSPSSSAAMEPLTANQLLASALSPGG
LNSLKPSEKALLRRFLRNPCDLRVASLLAAAHPTKELFPLAGNSFDSAESAGAAFVLPPFLMDPDES
NGGIGGSNLANGRSMEDELRFKRGTATGFSDHRQKIA
>AGG10332.1
MTHRTAAKLLLAVVSLFCVLQMLECGVVDRQPRAYKQYNTEPQKRPFCNAFTGCGKKRSSASSPPTA
AAAAAAAAMLQRHLQTMDPGRKDRMGSDFDPNEESISSLLDLNTEPAVEDLLRQIMSEAKLWEAIQE
ANREIYLQKSGMKDQRNDFPLTFSTQ
>AVR59279.1
MLTPVSGLVSLVGAVTVATATSTSPAAMASLVLDHTELPLAGTIPPAALMPARVLLPSNATNLTLTL
EELLRPNSSTVAPPNGDNDIIFSNKLVQIVFCVLYSSIFVLGVFGNVLVCYVVFRNKAMQTVTNLFI
TNLALSDILLCVLAVPFTPSYTFMRRWVFGKLLCHTVPLAQGCSVYISTLTLTSIAIDRFFVIIYPF
HPRMKLSTCITIIVLIWSFAIMVTMPYGLYMKLHGVALNGTDNATGPLSSAMYCEELWPSEEMRKTF
SIVTSILQFVLPFIIMAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKRTNRMLISMVAIFGIS
WLPLNVVNMCNDFNSDINSWRFYNLIFFIAHLTAMSSTCYNPFLYAWLNDNFRKEFKQVLPCFDPSR
GRAGTVGGNRGAGGGWRSERTCNGNNDTVQETLIPSSQVLPSSRSSQPHQQPQQLSSSSSQGQGNQP
TVDSILLSEVVPPLSLPPPLTGAMLPSVQSAETVILPSGVLETPFDVQLIPSVGAPVSNGTSDATNL
TPSAGVRNGLNHACTNPKLSSLILINDGTTADSKVPAIL
>XP_003699223.1
MLSASCMKAIFIFAMIGFVFGVESYMDYGDEISDKTPAENIHELYRLLLQRAGLENPGYNEAPFEHL
MIRKSQRSPSLRLRFGRSDPHLAMRLLSRQMSAIAPPRFEDN
>XP_003703358.1
MLDSIVISPNRATFVCALLCFFCVVSSTSGEYEGRESPSSVSNERTAGNEFGSCTDGKCIKRTTQDI
SSGMWFGPRLGRRRRADRKLETSPDLEALANFLDGSRWTVITIPGGEKRQPTQFTPRLGRESGEDFF
SYGFPKDQDELYAEEQMLPPLFAPRLGRRVPWTPSPRLGRQLHMLEKSRQYPEDSRF
>XP_012134940.1
MTYVFRNTSRHVPTTILVVLSILYGTISFLAVIGNSLVMWIVISTKRMQNVTNFYIANLALADIVIG
IFAIPFQFHAALLQRWNLPYFMCAFCPFVQALSVNVSVFTLTAIAVDRHRAILKPLSAKPRKLTAKI
IVAGIWFLAVCLAAPMAIARRVVMEPEGRGRYKPFCKEVNMSKSSILIYSAILFFMQYLTPLSVISC
AYVRMALKLWSNKAPGNAEDSRDATLMRNKMRVIKMLIIVVALFAICWLPLQTYNFFRYFYPQINQY
EYIHYIFFSSDWLAMSNSCYNPFIYGIYNENFRREFQQKCSCIFQRRLNNTAGDNVDTDKTQSSRTS
VRYEWKRTISSSYPVVSACYKGATREEPNESLIGEHQTSARNKKGIGYYIRSRNHCSLPSNRNEELS
VFPTRKLKHVEDSGVKELCL
>XP_012135020.1
MISTTSSSSFNDLYNDTTMSNKSFQVDITEEILWYTYSRFNETNYLLIILYVPVIALAVTANVLVIA
VVIKYHYMRSVTNYFVVNLSVADLLVTTICMPVAVSQAISMVWTYGEVMCKLSSYLQGVAVAASVFT
ITAMSIDRYLAIRSPMAFRRVFNRKSTVFVIVALWLVALIIFAPVLKAMTLRDPSQELSNITLHGSW
IMAGNFSENISPMSQRPPPFYVCWEDYKLLGVREHLFGTVCFVLVYAIPGFVVILSYSMMGRTLCSR
KPPFDCDSVKGSASSQQSFRLVRERRRIAWILLLLAVLFALCWLPYNVLMLLIDLSVIGEETVTTDA
LSYCLFLGHANSALNPVVYCFMTRNFRRSVAEILRRGPRALARRRPRRKSVQGTAVIDDMCAGCNAG
GTMRRGLLRKRRMLPGCGCGLPIGGHHAVLTVKRTATSSSGYDSFYSRHSPHRRCYMLQSIRKKPHV
PVDQAQNANKRQETSYQKNVTTSQPRVNTLVTTDEQR

C) Výstup Blast

```
AAK84160.1 sp|Q95P48.1|PBAN_SPOLI     100.000    191   0     191   1
     191   1.27e-141  390
sp|Q95P48.1|PBAN_SPOLI     sp|Q95P48.1|PBAN_SPOLI     100.000    191
     0     191   1     191   1.27e-141  390
ABD52277.1 gb|ABD52277.1|  100.000    350   0     350   1     350   0.0
     717
AFW19647.1 gb|AFW19647.1|  100.000    130   0     130   1     130
     2.30e-93   263
AAR03495.1 gb|AAR03495.1|  100.000    193   0     193   1     193
     2.87e-144  396
AAT81601.1 gb|AAT81601.1|  100.000    101   0     101   1     101
     3.58e-72   207
sp|Q7PTL2.2|PBAN_ANOGA     ref|XP_307885.2|100.000    187   0     187
     1     187   1.22e-138  382
ABD96048.1 sp|A0SIF1.1|SNPF_ANOGA     100.000    234   0     234   1
     234   4.90e-175  478
ABD96049.1 gb|AVR59279.1|  100.000    575   0     575   1     575   0.0
     1172
sp|Q7Q7R8.3|NPF_ANOGA sp|Q7Q7R8.3|NPF_ANOGA 100.000    89    0     89
     1     89    5.15e-63   182
sp|A0SIF1.1|SNPF_ANOGA     sp|A0SIF1.1|SNPF_ANOGA     100.000    234
     0     234   1     234   4.90e-175  478
sp|A0NDK8.1|CORZ_ANOGA     sp|A0NDK8.1|CORZ_ANOGA     100.000    171
     0     171   1     171   2.79e-123  342
ACH99845.1 gb|ACH99845.1|  100.000    105   0     105   1     105
     2.28e-75   215
AFE02890.1 gb|AFE02891.1|  100.000    171   0     171   1     171
     1.99e-123  342
AFE02891.1 gb|AFE02891.1|  100.000    171   0     171   1     171
     1.99e-123  342
AGG10332.1 gb|AGG10332.1|  100.000    160   0     160   1     160
     6.31e-121  335
AVR59279.1 gb|AVR59279.1|  100.000    575   0     575   1     575   0.0
     1172
XP_003699223.1  ref|XP_003699223.1|  100.000    109   0     109   1
     109   4.45e-79   225
XP_003703358.1  ref|XP_003703358.1|  100.000    191   0     191   1
     191   3.25e-142  391
XP_012134940.1  ref|XP_012134940.1|  100.000    422   0     422   1
     422   0.0   877
XP_012135020.1  ref|XP_012135020.1|  100.000    506   0     506   1
     506   0.0   1046
```

D) Výstup FURIN-FURIN

>XP_013189721.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPLMGRRKR
>XP_013189722.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPLMGRRKR
>XP_028173908.1
RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR
>XP_028173909.1
RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR
>XP_028173910.1
RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR
>XP_028173911.1
RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR
>XP_013179738.1
RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_026744122.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_026744123.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_013148126.1
RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISDRRKR
>XP_014371239.1
RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_021198427.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_021198428.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>PZC71758.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>OWR51862.1
RQKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_011547828.1
RRKRGSGTKVGGAAASAKTATKNNGNKKNFRPLTNDRRKR
>XP_011547829.1
RRKRGSGTKVGGAAASAKTATKNNGNKKNFRPLTNDRRKR
>NP_001153662.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>sp|B9WZ56.1|NEUPP_BOMMO
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_026756293.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_026756294.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR
>XP_026321326.1
RRKRGSGTKVGGAAASAKTATKNSGGGKKNFRPISERRKR
>XP_026321329.1
RRKRGSGTKVGGAAASAKTATKNSGGGKKNFRPISERRKR
>XP_022823038.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPTSERRKR
>XP_022823039.1
RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPTSERRKR
>XP_023937144.1
RRKRGSGTKVGGAAASAKSATKNSGGNKKNFRPISERRKR

E) Výstup METHIONIN-FURIN

>ABD96049.1
MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR
>AVR59279.1
MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR
>ETN60053.1
MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR
>KFB49968.1
MAFCYVCVSIRLNDRARTKPGSKTSRREEADRDRKKR
>XP_022122389.1
MALAQGLPDLYQVDPNEVARRYEEARRKRMYANRMKR
>XP_022122390.1
MALAQGLPDLYQVDPNEVARRYEEARRKRMYANRMKR
>sp|Q7M3V5.1|CSK_CALVO
MLMAPKDYQQKLHAKIPLNLDLMDFLLEYEDEDRSKR
>XP_023019356.1
MAFCYICVSIKLNDRARSKPGSKNARKEEADRERKRR
>OWR40906.1
MDYDGNWYLKRGAVGMPAHGLYRPIYIETNGNTRNKR
>NP_001037049.1
MRFNYSHMVSMTVYSVLMVISATGNLTVLYQLVRRRR
>BAG68409.1
MAFCYTCVSIKLNDRLKSRPGSKNSKKEDAERERKRR
>NP_001127707.1
MAFCYTCVSIKLNDRLKSRPGSKNSKKEDAERERKRR
>XP_016959826.1
MMAPNEQVAVPPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_016959828.1
MMAPNEQVAVPPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017084663.1
MMTPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017084664.1
MMTPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017046540.1
MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR
>XP_017046541.1
MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR
>XP_017046542.1
MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR
>SPP87876.1
MLAAQVSQLSSLGDANSLLESDVMDVGRHYASGRAKR
>XP_017021401.1
MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017021402.1
MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017021404.1
MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_017021405.1
MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR
>XP_022221896.1
MLAAQVSQLSSLGDANSLLESDAVDVGRHYASGRAKR
>XP_002014041.2
MLAAQVSQLSSLGDANSLLETEVVDVGRHYSSVRAKR

F) Výstup RKKR

```
>AZT81893.1
KRHVGVLARTGQLPFQGKRSYSSLVRNGDLPFFIRK
>AWT50606.1
RKVCSYFKDPLVCIGPVASDLQFKLNDAERTAGEKR
>AWT50616.1
KRNGMDEIDRSSFSGFFKKKNFDEIDRSGFDSFVKR
>XP_013165451.1
RKVDDRTSALLLTFYGILVVIGAVGNALVVISVVRK
>sp|C0HKR2.1|ALLA_AGRIP
KRLPLYNFGLGKRARSYNFGLGKRLASKFNFGLGKR
>AAC72894.1
KRSLSASPYDTSASEEDVDEFARLIRRPFNFGLGKR
>XP_022116255.1
RKVDDPTRSLLLTFYGILVVIGAVGNALVVLSVVRK
>XP_022122389.1
KRAVSKNTDTSSAKLTTKKSAEKKKIFRLISDRRKR
>XP_022122390.1
KRAVSKNTDTSSAKLTTKKSAEKKKIFRLISDRRKR
>XP_023023025.1
RKYFKRYLFCRKQKRYFSSNYRNQQTTSMSLVSMKR
>RXG54557.1
KRGPKQNFLRFGKSDNPIAAALNNRNIDELTEEDKR
>JAB57353.1
RKEQRAPSLRLRYGRMDPLLQSITNENALEDLIDRK
>OWR40781.1
RKESIPIISKTEKNISILTKDVNFCTEEWPSPETRK
>OWR46958.1
KRNRFMNDSYEDGQSYRTRVLSIRSTNERSIYSTRK
>AAC72893.1
KRLPVYNFGLGKRSKMYGFGLGKRDGRMYSFGLGKR
>AJM76779.1
RKNRARDHFLRLGRDSEELNDTSAEEEFSDEERRKR
>AJM76767.1
KRARPYSFGLGKRFDDDIDMSEEKRARGYDFGLGKR
>NP_001295987.1
KRARPYSFGLGKRFDDDIDMSEEKRARGYDFGLGKR
>NP_001037036.1
KRARMYSFGLGKRARSYSFGLGKRLSSKFNFGLGKR
>ABX52401.1
RKIFCIRKEPTVLPGIKRPETITLVDQLRISQSRKR
>BAG49563.1
KRSISILAKNGMLPTYRSPYVGTDKQEHEDESQEKR
>BAG68411.1
RKETIPIISKGDKNISIETRDIHFCIEEWPSPETRK
>NP_001124353.1
KRSISILAKNGMLPTYRSPYVGTDKQEHEDESQEKR
>NP_001127709.1
RKETIPIISKGDKNISIETRDIHFCIEEWPSPETRK
>XP_018321863.2
KRHEDDEAIAKEEMDAVMKLVKETPWAVVALNNGKR
>CAD87596.1
KRLPMYNFGLGKRARSYNFGLGKRLSSKFNFGLGKR
```

G) Výstup SQL

```
CREATE TABLE test_final (
ID INT NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO test_final (SEQ_ID, SEQ) VALUES
('AAK84160.1','MFSPLLFFAVSISCVLANSNEIKDGGSDRGAHSDRAGLWFGPRLGKRSLRIS
TEDNRQAFFKLLEAADALKYYYDRLPYEMQADEPETRVTKKVIFTPKLGRSLAYDDKVFENVEFTPR
LGRRLADDMPATPADQELYRPDPDQIDSRTKYFSPRLGRTMNFSPRLGRELSYDMLPSKLRLVRSTN
RTQST'),
('sp|Q95P48.1|PBAN_SPOLI','MFSPLLFFAVSISCVLANSNEIKDGGSDRGAHSDRAGLWF
GPRLGKRSLRISTEDNRQAFFKLLEAADALKYYYDRLPYEMQADEPETRVTKKVIFTPKLGRSLAYD
DKVFENVEFTPRLGRRLADDMPATPADQELYRPDPDQIDSRTKYFSPRLGRTMNFSPRLGRELSYDM
LPSKLRLVRSTNRTQST'),
('ABD52277.1','MTLSAPPIDEFEDPFVVMNTTNVSSHPAAYDEPYTLDLVVPLTVTYVVIFVA
GILGNTSTCVVIARNRSMHTATNFYLFSLAISDLILLVCGLPFEVHRLWNPDTYPLGEAHCIAIGLA
SETSANATVLTITAFTVERYIAICRPFMSHTMSKLSRAVRFIIAIWVFALCTAVPQAMQFGIVSYVD
NGQNVSACTVKGVGVHQVFVISSFVFFVVPMSMISVLYALIGIKLRTSRVLHPVKKLSVDSNERASG
QMQYRNGASQRRVIRMLVAVALSFFICWAPFHVQRLLAIYGKSLEHPSDTFYLVYIVLTFLSGVLYF
LSTAINPFLYNIMSNKFRNAFKVSFCYIHD'),
('AFW19647.1','MSPSFVFALAVCGFAAICSLPVTNAQALVSPYESSAAADSQNGWGALGGLYA
LLAQHDALGGHALARKSVRSPSRRLRFGRRSDPDMPPQAPLDEMNELLSLREVRTPVRLRFGRRSEE
RAVPHIFPQEF'),
('AAR03495.1','MARVTLALAVSLAAYAAYYYVHIAEAQPIGSLVLAAPHSQQQQPGSSTSDEA
TINHLQQQHQRLKDTNVYRARSKMRPHDRKYPGVIGAYEAYRRTVQGPQLMQRNPATADRFADDPGV
DEQDQMRFSLEGFLTGARTPTLLNDDEEEEEDEDHEQGGDGLVKRFDDYGHMRFGKRGGEGDQFDDY
GHMRFGR'),
('AAT81601.1','TMASGTFTQRLLVALMIFALIADLSTLVAARPQDSDAASVAAAIRYLQELET
KHAQHARPRFGKRGGYLNPAIFGQDEQENLYRLIGRIQHFRDEQLPTNI'),
('sp|Q7PTL2.2|PBAN_ANOGA','MSRFYFFFNLICLYLAIKSALSAELDTNDQKYADLRTTGR
GESPDSTGPDSDTLRRDDGAEGLNKRAAAMWFGPRLGKRTIAADLHDDLVEEFDAEPLGYAGEPPQK
LATELVQGAPYMVLLVTAKPRKPQPIFYHTTSPRLGRRDSVGENHQRPPFAPRLGRNLPFSPRLGRS
YNGGYPLPFQFAY'),
('ABD96048.1','MYRINLTTFTLLLVLAVGSLMSESLHPSDGAINDLYEYLLQREYAAPVSYAD
HQIKRKAVRSPSLRLRFGRRSDPSVPLRPEEDELIDQKAIRAPQLRLRFGRNDPLWTSFNENALLEE
NFEKRAPSQRLRWGRSNLFGNLVNQFQQDDVMQQKTIRAPQLRLRFGRTDPSWAMYNEHQLTTGQQA
QPANEASEKRAPTQRLRWGRSDPALAKDSSEDKALDVEESENTNADDK'),
('ABD96049.1','MLTPVSGLVSLVGAVTVATATSTSPAAMASLVLDHTELPLAGTIPPAALMPA
RVLLPSNATNLTLTLEELLRPNSSTVAPPNGDNDIIFSNKLVQIVFCVLYSSIFVLGVFGNVLVCYV
VFRNKAMQTVTNLFITNLALSDILLCVLAVPFTPSYTFMRRWVFGKLLCHTVPLAQGCSVYISTLTL
TSIAIDRFFVIIYPFHPRMKLSTCITIIVLIWSFAIMVTMPYGLYMKLHGVALNGTDNATGPLSSAM
YCEELWPSEEMRKTFSIVTSILQFVLPFIIMAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR
TNRMLISMVAIFGISWLPLNVVNMCNDFNSDINSWRFYNLIFFIAHLTAMSSTCYNPFLYAWLNDNF
RKEFKQVLPCFDPSRGRAGTVGGNRGAGGGWRSERTCNGNNDTVQETLIPSSQVLPSSRSSQPHQQP
QQLSSSSSQGQGNQPTVDSILLSEVVPPLSLPPPLTGAMLPSVQSAETVILPSGVLETPFDVQLIPS
VGAPVSNGTSDATNLTPSAGVRNGLNHACTNPKLSSLILINDGTTADSKVPAIL'),
('sp|Q7Q7R8.3|NPF_ANOGA','MASGTFTQRLLVALMIFALIADLSTLVAARPQDSDAASVAA
AIRYLQELETKHAQHARPRFGKRGGYLNPAIFGQDEQEVDWQDSTFSR'),
…
;
```

```
CREATE TABLE test_final_RKKR (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO test_final_RKKR (SEQ_ID, SEQ) VALUES
('AZT81893.1','KRHVGVLARTGQLPFQGKRSYSSLVRNGDLPFFIRK'),
('AWT50606.1','RKVCSYFKDPLVCIGPVASDLQFKLNDAERTAGEKR'),
('AWT50616.1','KRNGMDEIDRSSFSGFFKKKNFDEIDRSGFDSFVKR'),
('XP_013165451.1','RKVDDRTSALLLTFYGILVVIGAVGNALVVISVVRK'),
('sp|C0HKR2.1|ALLA_AGRIP','KRLPLYNFGLGKRARSYNFGLGKRLASKFNFGLGKR'),
('AAC72894.1','KRSLSASPYDTSASEEDVDEFARLIRRPFNFGLGKR'),
('XP_022116255.1','RKVDDPTRSLLLTFYGILVVIGAVGNALVVLSVVRK'),
('XP_022122389.1','KRAVSKNTDTSSAKLTTKKSAEKKKIFRLISDRRKR'),
('XP_022122390.1','KRAVSKNTDTSSAKLTTKKSAEKKKIFRLISDRRKR'),
('XP_023023025.1','RKYFKRYLFCRKQKRYFSSNYRNQQTTSMSLVSMKR'),
('RXG54557.1','KRGPKQNFLRFGKSDNPIAAALNNRNIDELTEEDKR'),
('JAB57353.1','RKEQRAPSLRLRYGRMDPLLQSITNENALEDLIDRK'),
('OWR40781.1','RKESIPIISKTEKNISILTKDVNFCTEEWPSPETRK'),
('OWR46958.1','KRNRFMNDSYEDGQSYRTRVLSIRSTNERSIYSTRK'),
('AAC72893.1','KRLPVYNFGLGKRSKMYGFGLGKRDGRMYSFGLGKR'),
('AJM76779.1','RKNRARDHFLRLGRDSEELNDTSAEEEFSDEERRKR'),
('AJM76767.1','KRARPYSFGLGKRFDDDIDMSEEKRARGYDFGLGKR'),
('NP_001295987.1','KRARPYSFGLGKRFDDDIDMSEEKRARGYDFGLGKR'),
('NP_001037036.1','KRARMYSFGLGKRARSYSFGLGKRLSSKFNFGLGKR'),
('ABX52401.1','RKIFCIRKEPTVLPGIKRPETITLVDQLRISQSRKR'),
('BAG49563.1','KRSISILAKNGMLPTYRSPYVGTDKQEHEDESQEKR'),
('BAG68411.1','RKETIPIISKGDKNISIETRDIHFCIEEWPSPETRK'),
('NP_001124353.1','KRSISILAKNGMLPTYRSPYVGTDKQEHEDESQEKR'),
('NP_001127709.1','RKETIPIISKGDKNISIETRDIHFCIEEWPSPETRK'),
('XP_018321863.2','KRHEDDEAIAKEEMDAVMKLVKETPWAVVALNNGKR'),
('CAD87596.1','KRLPMYNFGLGKRARSYNFGLGKRLSSKFNFGLGKR'),
('AAF89172.1','KRAPTGFTGMRGKRPALLAGGDDAEADEATELQQKR'),
('sp|Q9VGE8.1|TACHY_DROME','KRAPTGFTGMRGKRPALLAGDDDAEADEATELQQKR'),
('AAC72892.1','KRDYDDYYGDDDEEDHQTSADEDIEDADSVDLMDKR'),
('PSN42328.1','RKLRYRYVNSSIGSKNKCETCSTTSTNRRKPKDNKR'),
('AAY82901.1','KRGQYAFGLGKREDEEKRSKTFSFGLGKRVPDDEKR'),
('ANF04992.1','KRAGWSSMRGAWGKRDDSSDQGLQVSEDKRNNNWRK'),
('XP_017788751.1','RKVFVDLPGTPDSVGTIIGKAANPRNSQLDAIHGKR'),
('NP_001161308.1','KRTLAASGLGGLKAALIEEEKPSRSNTLNNAFYDRK'),
('AEI91710.1','KRFGEDLQSAIFFFGMSNSLVNPLIYGAFHLCPMKR'),
('NP_001295994.1','KRFGEDLQSAIFFFGMSNSLVNPLIYGAFHLCPMKR'),
('XP_026322916.1','RKVDDPIRAILMTFYGILVVIGAVGNALVVISVVRK'),
('AEX86939.1','KRGQYAFGLGKREDEEKRSKTFSFGLGKRVPDDEKR'),
('ENN70850.1','RKISPWTFAKSIKPQEGCFSTYMTLTGLILRLYIKR'),
('ERL88321.1','RKISPWTFAKSIKPQEGCFSTYMTLTGLILRLYIKR'),
('XP_019758999.1','RKHFNRYLLCKKMKRNRCNTCTGQQATSMSLVSTKR'),
('XP_022911265.1','RKHFNRYLLCIKPKRVRCDTCQGNRATSMSMASSKR'),
('JAC52197.1','KRWFGDVNQKPIRSPSLRLRFGRRSDPDMYLPAEKR'),
('JAC52199.1','KRWFGDVNQKPIRSPSLRLRFGRRSDPDMYLPAEKR'),
('sp|P12764.2|ALLS_DIPPU','KRLPVYNFGLGKRSKMYGFGLGKRDGRMYSFGLGKR'),
('ROT83139.1','KRDFAFSPRLGKRDFAFNPRLGKRDGFAFNPRLGKR'),
…
;
```

```
CREATE TABLE test_final_METHIONIN_FURIN (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO test_final_METHIONIN_FURIN (SEQ_ID, SEQ) VALUES
('ABD96049.1','MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR'),
('AVR59279.1','MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR'),
('ETN60053.1','MAFCYICVSIRLNDRARTKPGSKTSRREEADRDRKKR'),
('KFB49968.1','MAFCYVCVSIRLNDRARTKPGSKTSRREEADRDRKKR'),
('XP_022122389.1','MALAQGLPDLYQVDPNEVARRYEEARRKRMYANRMKR'),
('XP_022122390.1','MALAQGLPDLYQVDPNEVARRYEEARRKRMYANRMKR'),
('sp|Q7M3V5.1|CSK_CALVO','MLMAPKDYQQKLHAKIPLNLDLMDFLLEYEDEDRSKR'),
('XP_023019356.1','MAFCYICVSIKLNDRARSKPGSKNARKEEADRERKRR'),
('OWR40906.1','MDYDGNWYLKRGAVGMPAHGLYRPIYIETNGNTRNKR'),
('NP_001037049.1','MRFNYSHMVSMTVYSVLMVISATGNLTVLYQLVRRRR'),
('BAG68409.1','MAFCYTCVSIKLNDRLKSRPGSKNSKKEDAERERKRR'),
('NP_001127707.1','MAFCYTCVSIKLNDRLKSRPGSKNSKKEDAERERKRR'),
('XP_016959826.1','MMAPNEQVAVPPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_016959828.1','MMAPNEQVAVPPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017084663.1','MMTPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017084664.1','MMTPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017046540.1','MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017046541.1','MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017046542.1','MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('SPP87876.1','MLAAQVSQLSSLGDANSLLESDVMDVGRHYASGRAKR'),
('XP_017021401.1','MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017021402.1','MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017021404.1','MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_017021405.1','MMAPNEQVAVAPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_022221896.1','MLAAQVSQLSSLGDANSLLESDAVDVGRHYASGRAKR'),
('XP_002014041.2','MLAAQVSQLSSLGDANSLLETEVVDVGRHYSSVRAKR'),
('XP_016983828.1','MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('XP_016983829.1','MMAPNEQVAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('XP_020808506.1','MMAPNEQVAVSPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_020808510.1','MMAPNEQVAVSPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_020808517.1','MMAPNEQVAVSPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_016935891.1','MMAPNEQVAVPPEGDFNDYPARQVLYKIMRSWFNRPRR'),
('XP_016993506.1','MMAPNEQGAVAPEADFNDYPARQVLYKIMRSWFNRPRR'),
('XP_026762441.1','MAFCYTCVSIKLNDRLKSRPGSKNSKKEDAERERKRR'),
('XP_022672325.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672326.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672327.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672328.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672329.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672330.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672331.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672332.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672333.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672334.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672335.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
('XP_022672336.1','MALCYVRVCSRLAVRAKYMPGAKSQQKEELERKRAKR'),
…
;
```

```
CREATE TABLE test_final_FURIN_FURIN (
ID int NOT NULL AUTO_INCREMENT,
SEQ_ID VARCHAR(255) NOT NULL,
SEQ MEDIUMTEXT NOT NULL,
PRIMARY KEY (ID, SEQ_ID)
);

INSERT INTO test_final_FURIN_FURIN (SEQ_ID, SEQ) VALUES
('XP_013189721.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPLMGRRKR'),
('XP_013189722.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPLMGRRKR'),
('XP_028173908.1','RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR'),
('XP_028173909.1','RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR'),
('XP_028173910.1','RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR'),
('XP_028173911.1','RRKRGSGSKVSGAAASAKTATKNSGGNKKNFRPLSERRKR'),
('XP_013179738.1','RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_026744122.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_026744123.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_013148126.1','RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISDRRKR'),
('XP_014371239.1','RLKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_021198427.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_021198428.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('PZC71758.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('OWR51862.1','RQKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_011547828.1','RRKRGSGTKVGGAAASAKTATKNNGNKKNFRPLTNDRRKR'),
('XP_011547829.1','RRKRGSGTKVGGAAASAKTATKNNGNKKNFRPLTNDRRKR'),
('NP_001153662.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('sp|B9WZ56.1|NEUPP_BOMMO','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRK
R'),
('XP_026756293.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_026756294.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR'),
('XP_026321326.1','RRKRGSGTKVGGAAASAKTATKNSGGGKKNFRPISERRKR'),
('XP_026321329.1','RRKRGSGTKVGGAAASAKTATKNSGGGKKNFRPISERRKR'),
('XP_022823038.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPTSERRKR'),
('XP_022823039.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPTSERRKR'),
('XP_023937144.1','RRKRGSGTKVGGAAASAKSATKNSGGNKKNFRPISERRKR'),
('XP_026495617.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRSVSERRKR'),
('XP_026495618.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRSVSERRKR'),
('XP_028027448.1','RRKRGSGTKVGGAAASAKTATKNSGGNKKNFRPISERRKR')
;
```

H) Výstup Summary

```
**********  Summary report from Sat Apr 13 21:19:35 2019
**********

Name of the tested fasta file: test.fasta

Number of sequences tested: 14535

**************************************************

SignalP processing time - 1 hours, 15 minutes, 48 seconds

Number of sequences containing signal peptide: 3589

**************************************************

Name of the database used – db_insect_neuropeptides

Blast processing time - 12 minutes, 29 seconds

**************************************************

Motifs used in test:

********** RKKR-RKKR **********
Range: 32
Found: 80

********** Methionin-Furin **********
Range: 32
Found: 97

********** Furin-Furin **********
Range: 32
Found: 29
```

I)Vývojový diagram skriptu