**Praktická část**

! Veškeré řešení – skripty/odpovědi/použité funkce dávejte do nové složky “Odpovědi”.

Př.: po první úloze tam uložte script .R/.py s kódem a textový soubor s výsledky.

1. Načtěte soubor Data/transfery.csv a vypracujte následující statistiky:
   1. Dle věku matky “vek\_mother”, ve věkových kategoriích viz tabulka, vytvořte tabulku úspěšnosti embryotransferu v procentech dle sloupce “clinical\_gravidity”, kde 1 = transfer byl úspěšný a 0 = neúspěšný. Prázdné hodnoty do statistik nepočítejte.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| všechny věkové kategorie | do 29 | 30-34 | 35-39 | 40 a výše |
| x% | x% | x% | x% | x% |

* 1. Určete zda-li je věk matky statisticky významný na úspěch transferu.

Vyšším vekom matky klesá úspešnosť transferu.

* 1. Taktéž A-B proveďte i pro věk embrya “vek\_embryo”. Pokud bylo embryo darované ”f\_donor” = 1, takový transfer do statistiky nepočítejte.
  2. Vytvořte tabulku s počty transferů dle použité genetické metody "genetic\_method” viz tabulka.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PGT-A | PGT-SR | Karyomapping | OneGene | bez genetické metody (prázdná hodnota) | ostatní |
| x | x | x | x | x | x |

* 1. Určete statistickou významnost pohlaví embrya “sex” – XX/XY na úspěch klinické gravidity dle sloupce “clinical\_gravidity”, kde 1 = transfer byl úspěšný a 0 = neúspěšný. Prázdné hodnoty do statistik nepočítejte.
  2. Z výsledných tabulek z úkolu A a D vytvořte a uložte grafy ve formátu .png, kde na ose x bude první a na ose y druhý řádek tabulky.

BONUS) Vytvořte jednoduchou R shiny aplikaci, která udělá předchozí statistiky při spuštění a vykreslí tabulky nebo hodnoty, tak aby je uživatel viděl.

1. Vytvořte script, který bude vytvářet jednoduchý .docx dokument obsahující nadpis, tučným a zarovnaný na střed “Výsledný protokol genetického vyšetření”, a dále pak tabulku, která se vyplní dle tří vstupních argumentů následovně:

|  |  |
| --- | --- |
| Jméno a příjmení: | argument 1 |
| Rodné číslo: | argument 2 |
| Datum odběru: | argument 3 |

**Teoretická/vyhledávací část**

! Úkoly jsou koncipovány, tak abyste odpovědi hledali a získali tak trochu přehled, nejedná se o vědomostní test.

Nalezené výsledky vkládejte do libovolného textového dokumentu do složky “Odpovědi”.

Zde netřeba uvádět postup, kde a jak jste informaci našli.

1. Uveďte nukleotidovou sekvenci exonu 9 genu CFTR. (tip: hledejte “ensembl”)

NBD1

1. Uveďte, jaké onemocnění způsobují patogenní mutace v genu CFTR a některé uveďte.

Dedičné ochorenie spôsobené mutáciami v géne CFTR, ktorý riadi činnosť tzv. chloridového kanála v membráne buniek. Choroba postihuje množstvo orgánov, predovšetkým dýchací, tráviaci a reprodukčný systém.

Cystická fibróza

1. Uveďte, co znamená autosomálně recesivní přenašečtví.

Autozomálně recesivní je jeden z několika typů dědičnosti, jakými se může v rodině přenášet nějaký znak nebo i genetická porucha. U autozomálně recesivního onemocnění platí, že pro vznik onemocnění musí být přítomny dvě kopie abnormálního genu.

1. Rozepiště, jak byste postupovali při vytváření SQL datábaze na Linux serveru, tak aby byla zajištěna integrita dat.
2. Na svojom počítači so systémom Linux otvorte reláciu bash terminálu.
3. Použite sqlcmd na spustenie príkazu Transact-SQL CREATE DATABASE. Bash Copy. /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -Q 'CREATE DATABASE SampleDB'
4. Skontrolujte, či je databáza vytvorená uvedením databáz na vašom serveri. Bash Copy

Ak chcete pridať obmedzenia integrity v SQL, použite príkaz ALTER TABLE a zadajte typ obmedzenia, ako napríklad PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK alebo NOT NULL. Potom definujte potrebné stĺpce a obmedzujúce podmienky v rámci príkazu

1. Uveďte, co znamená SQL Injection a jak se tomu vyvarovat.

SQL Injection je zraniteľnosť založená na kóde, ktorá umožňuje útočníkovi čítať a pristupovať k citlivým údajom z databázy. Útočníci môžu obísť bezpečnostné opatrenia aplikácií a použiť SQL dotazy na úpravu, pridávanie, aktualizáciu alebo odstraňovanie záznamov v databáze. Úspešný útok SQL injection môže nepriaznivo ovplyvniť webové stránky alebo webové aplikácie používajúce relačné databázy, ako sú MySQL, Oracle alebo SQL Server. V posledných rokoch došlo k mnohým narušeniam bezpečnosti, ktoré boli výsledkom útokov SQL injection.

Ako zabrániť SQL Injection?

1. Použiť pripravené príkazy a parametrizované dotazy – Parametrizované príkazy zaisťujú bezpečné zaobchádzanie s parametrami odovzdanými do príkazov SQL.
2. Objektovo-relačné mapovanie – Väčšina vývojových tímov uprednostňuje použitie objektov objektovo-relačného mapovania na plynulejší preklad sád výsledkov SQL do objektov kódu.
3. Únikové vstupy – Je to jednoduchý spôsob ochrany pred väčšinou útokov SQL injection. Mnoho jazykov má štandardné funkcie na dosiahnutie tohto cieľa. Musíte si byť vedomí pri používaní escape znakov vo vašej kódovej báze, kde sa vytvára príkaz SQL.

Niektoré z ďalších metód používaných na zabránenie SQL Injection sú:

Hašovanie hesla

Overenie treťou stranou

Firewall webovej aplikácie

Kúpte si lepší softvér

Vždy aktualizujte a používajte opravy

Neustále monitorujte SQL príkazy a databázu

1. Zjistěte, co znamená error: “Error in .local: Cannot allocate a new connection: 16 connections already opened” a napiště jak byste postupovali při jeho opravě.

Napísať malú funkciu, ktorá zvládne celú vec otvárania a zatvárania

library(RMySQL)

sqlQuery <- function (query) {

DB <- dbConnect(MySQL(), user="youruser", password='yourpassword', dbname='yourdb', host='192.168.178.1')

on.exit(dbDisconnect(DB))

rs <- dbSendQuery(DB, query)

result <- fetch(rs, -1)

return(result)

}

1. Zjistěte, co znamená error: “Error in if: argument is of length zero” a napiště jak byste postupovali při jeho opravě.

If(!is.null(data[k]][[k2]]) & temp > data[[k]][[k2]]){}

1. Napište, proč se dělá Sekvenování nové generace (NGS)?

Umožňuje detekovať všetky typy variantov naprieč celým genómom, v kódujúcich (tzv. exom) aj nekódujúcich oblastiach, a tak umožňuje identifikovať aj nové, predtým nepopísané nádorové varianty