VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Molekulinės dinamikos programų integracija statistinės kalbos R aplinkoje : Kai kurių algoritmų realizacija panaudojant C/C++ sąsają

Molecular Dynamics Software Integration to R Statistical Environment : Implementation of Selective Algorithms Using C/C++ Interface

Magistro baigiamasis darbas

Atliko: Audrius Jakutis
Darbo vadovas: Irus Grinis
Recenzentas: Darius Baronas

Vilnius – 2016

Santrauka

Šiame darbe nagrinėjame molekulinės dinamikos algoritmų integracijos galimybes į R statistinę aplinką naudojant C/C++ sąsajas skirtas R aplinkai. Darbe pateikiame R sistemai siūlomų programinių sąsajų apžvalgas. Grafines analizės galimybes. Aptariame bendrą funkcionalumą pasiekiamą per R aplinką. Praktinės darbo dalies rezultatas – molekulinės dinamikos algoritmų paketas kurį galima naudoti windows ir linux sistemose.

Raktiniai žodžiai: R sistema, molekulinės dinamikos algoritmai, C/C++ algoritmai, R paketas, paketas, programinė sąsaja, molekulinės dinamikos algoritmai.

Summary

In this thesis we research R system's abilities to accept molecular dynamic algorithms written in C/C++ language via C/C++ interfaces (provided by R itself). We also review other interfaces created for R and R system's extended functionality contributed by packages and external programs. The practical value of this work, besides invaluable research results and conclusions, is R package with functions to execute selected molecular dynamic algorithms in windows and linux environments.

Keywords: R system, molecular dynamic algorithms, C/C++ algorithms, R package, package, interface, molecular dynamic simulations

Turinys

Įvadas

Tyrimuose įvairūs molekulinės dinamikos algoritmai taikomi sukurto metodo ar metodų patikimumui nustatyti prieš pradedant jį/juos taikyti in vivo ar in vitro. Ir programinė įranga, naudojama tokiems tyrimams modeliuoti, yra specializuota savo turimo funkcionalumo aibe. Todėl, kai kuriais atvejais, norėdami atlikti papildomus skaičiavimus ir analizes su gautomis išvestimis, turime rašyti atskiras funkcijas ir/ar programas, kurios, dažnu atveju, yra išorinės tai programinei įrangai su kuria atliekame simuliacijas.

Atliktų tyrimų korektiškumo patikrinimas keliomis to paties algoritmo implementacijomis gali padėti išvengti klaidingų prielaidų, kurios atsiranda tyrimų eigoje naudojant kurį nors vieną, tiriamosios srities, programinį produktą. Taikomosios programos, dažnu atveju, leidžia tik varijuoti parametrais, kuriuos paduodame metodams ar funkcijoms ir neturi paprastos prieigos leidžiančios keisti pačio metodo struktūros[1].

Naudojant molekulinės dinamikos algoritmus aplinkose, turinčiose didelį statistinės analizės potencialą, galėtume naudoti platų spektrą analizės metodų arba kurti savo tyrimo modelius; gautas išvestis galėtume validuoti ir verifikuoti ne vienu, o keliais statistiniais modeliais. Visus tyrimus jungtų viena programavimo aplinka.

Kurdami programinę įrangą ar tiesiog taikomąsias programas dažniausiai stengiamės pasinaudoti jau turimu ar egzistuojančiu funkcionalumu. Neretai, atliekant turimos funkcijos derinimą su naujai kuriamu produktu, nutinka taip, kad reikiamas funkcionalumas yra parašytas kita programavimo kalba, egzistuoja kaip nepriklausomas modulis ar paketas kitose programavimo aplinkose, o gal būt veikia kaip autonomiška programa. Dėl programavimo aplinkų nesuderinamumo tenka kurti ir/ar naudoti įvairias programines sąsajas, kurios padeda derinti skirtingas programines aplinkas, skirtingomis kalbomis parašytus algoritmus.

Šiame darbe nagrinėjame rinktinių molekulinės dinamikos algoritmų integraciją į R statistinę aplinką. MD grupei priskiriami algoritmai retai yra antros eilės [You14, p. 2], kartais pasitaikančios implementacijos yra ketvirtos eilės [Erc16], ir priklausomai nuo nagrinėjamos molekulinės sistemos reikalauja daugybės iteracijų rezultatams pasiekti. Galutinis algoritmo rezultatas yra molekulinę sistemą sudarančių objektų koordinatės (dažniausiai imama trimatė erdvė ir pateikiamos įvairios statistikos, kurios gaunamos iš koordinačių) per iš anksto apibrėžtą laiko tarpą. Gautus rezultatus, panaudojus MD algoritmus, galima statistiškai apibendrinti, atlikti kitus skaičiavimus ir tyrimus.

[1] Kai kuriose programinėse įrangose, kurios yra atviro kodo, pavyzdžiui, LAMMPS, Gromacs, įmanoma pakeisti kodo išvesčių failus (code sources), perkompiliuoti programą ir naudoti pakeistą metodą, tačiau toks scenarijus reikalauja gilių programavimo ir nagrinėjamos dalykinės srities žinių. Ir dažnu atveju yra nepriimtinas

Šio darbo tikslas yra ištirti R galimybes įsisavinti C/C++ kalbomis parašytus molekulinės dinamikos algoritmus naudojant išorines C/C++ programines sąsajas. Darbo eigoje, siekdami parodyti statistinės aplinkos galimybes, pateiksime bendrą ir išplėstinį R funkcionalumą pasiekiamą per paketus ir bazines funkcijas. Praktinėje dalyje aprašysime gautus rezultatus ir išvadas, kurias gavome integracijos eigoje. Taip pat planuojame pateikti R sistemai skirtą paketą MD simuliacijoms atlikti.

1. R sistema

R yra aplinka [Fou16v] ir kalba [Fou16v] skirta matematiniams skaičiavimams vykdyti, duomenų statistinei (neapsiribojama vien statistika) analizei atlikti ir rezultatų atvaizdavimui pateikti (grafikai, diagramos, lentelės,...). Nuostabu yra tai, kad taikomuosius analizės modelius, naudojant R sistemą, nereikia programuoti nuo nulio – galima pasitelkti pačioje sistemoje esančius metodus[2] tyrimams atlikti. Būdama atviro kodo[3] ši sistema suteikia, dar daugiau galimybių – modifikuoti esamus skaičiavimo metodus, pridėti naujus, dalintis savo sukurtais tyrimo modeliais organizuojant savo sukurtus metodus į paketus ir platinant juos kitiems naudotojams per taip vadinamą R tinklą (The Comprehensive R Archive Network [CRAN]).

Analizės (skaičiavimo) modeliai, R sistemoje, yra išreiškiami kuriant naujas arba papildant jau esamas R funkcijas[4]. Įprastu atveju R kalba yra interpretuojama [Fou16g], dėl to ne visada, atliekant intensyvius skaičiavimus, rašyti R funkcijas yra geras sprendimas [Ran12, p.18] ir labiau verta pagalvoti apie kitas skaičiavimo aplinkas, pavyzdžiui, MathLab, arba, tą kuriamo modelio dalį, kuri reikalauja intensyvių procesoriaus resursų, rašyti labiau tinkamose aplinkose – intensyvius skaičiavimus perkeliant į labiau tokio pobūdžio užduotims tinkamas aplinkas, pavyzdžiui, dalį kodo rašyti C/C++ aplinkoje ir galutinius žingsnius, jau turint visus reikiamus skaičiavimus, perkelti atgal į R aplinką ir pabaigti vykdyti sukurtą tyrimo modelį įvykdant reikiamas R funkcijas.

Suprantama, kad skaičiavimo aplinkų keitimas, optimizuojant tyrimo metodą, nėra itin pageidautina aplinkybė, nes, dažnu atveju, keičiant skaičiavimų aplinkas susiduriama su tipų/skaičių formato nesuderinamumu [dJvdL13, p. 18] dėl ko, pavyzdžiui, atsiranda paklaidos, iškreipiančios analizę ir arba pats metodas tampa nepastovus (skaičiavimų išvestys tampa nenuspėjamos arba metodas, jo vykdymo eigoje, sustoja, nes įvestis/išvestis tampa nebevalidi aplinkai, kurioje vykdomi tyrimai). R sistema, tam, kad nereikėtų derinti atsirandančius skirtumus tarp skaičiavimo aplinkų suteikia galimybę skaičiavimų (įvesčių ir išvesčių), nesuderinamumo su kitomis skaičiavimų aplinkomis problemą dalinai spręsti per programines sąsajas, kuriomis naudojantis, gautas išvestis galima pervesti atgal į R aplinką ir atvirkščiai. Įdomu pastebėti, kad R aplinkoje, kai kurios bazinės funkcijos, pavyzdžiui, *lapply* [Fou16q, žr. do_apply], *abs* [Fou16q, žr. do_abs], *dim* [Fou16q, žr. do_dim] ir kt. [Fou16q] yra parašytos būtent per programines sąsajas į kitas skaičiavimų aplinkas, pavyzdžiui, C/C++.

[2] metodo ir funkcijos sąvoka, R sistemoje, iš programavimo pusės yra tapati

[3] išsamų R sistemos licenzijų sąrašas galima rasti adresu : <https://www.r-project.org/Licenses/>

[4] pilną oficialų R teikiamų funkcijų sąrašą galima rasti adresu : <https://cran.r-project.org/web/packages/>

2. R programinės sąsajos

R programinėje aplinkoje, skaičiavimams atlikti, dažniausiai užtenka tuos skaičiavimus užrašyti R programavimo kalba ir iškviesti vieną ar kitą funkciją. Naudodami programines R aplinkos sąsajas, šią idėją galima išplėsti kur kas plačiau, pavyzdžiui, galima aprašyti tyrimo modelį kitoje programavimo aplinkoje su jai būdinga programavimo kalba ir R iškvietus R funkcijas gauti rezultatus atgal. Šiame skyriuje pateikiame kai kurias R programines sąsajas ir tų sąsajų panaudojimo atvejus.

2.1. C/C++ programinės sąsajos

.Inside – didelis R sistemos funkcionalumas yra organizuotas funkcijomis, kurios, pagal tam tikras sritis, yra apjungtos į paketus. Kita dalis funkcionalumo jau ateina kartu su R sistema be jokių papildomų paketų – tai vidinės R funkcijos, kuriomis naudojasi pati R sistema. Dėl to, kad, kai kurios vidinės R sistemos funkcijos yra parašytos C kalba ir yra sukompiliuotos į R sistemos kodą jų iškvietimui sukurta vidinė programinė sąsaja .Inside [Wic16a]. Vidinės funkcijos, kurios yra kviečiamos per .Inside nėra prieinamos naudotojui, kaip ir pati programinė sąsaja, jų negalima modifikuoti [Fou16f].

.Primitive [5]– kiekviena programavimo kalba yra sudaryta iš primityvių struktūrų, kurios gali būti apjungiamos, praplečiamos, perrašomos ir/ar kitaip modifikuojamos. R kalba parašytas kodas įprastu R sistemos rėžimu yra interpretuojamas ir .Primitive programinė sąsaja naudojama iškviesti/apdoroti įvairias R struktūras – primityvias funkcijas, operatorius, kitas kalbai būdingas struktūras. Skirtingai nei programinė sąsaja .Inside, kuri naudojama kaip programinė sąsaja vidinėms funkcijoms kviesti, .Primitive, pagal, konvencija yra naudojama įvairioms kalbos vidinėms konstrukcijoms apdoroti. Taip pat, kai kuriais atžvilgiais .Primitive sąsaja yra našesnė nei .Inside. Tačiau, tiek .Primitive tiek .Inside yra skirtos R sistemos vystytojams.

.C – priešingai nei .Primitive ar .Inside programinės sąsajos, .C skirta ne tik R sistemos vidiniams procesams vykdyti [Wic16d], bet ir sistemos naudotojo parašytoms ir sukompiliuotoms C ir dalinai C++ funkcijoms kviesti [Str13]. Naudojant šią programinę sąsają galima susidurti su keliais nepatogumais, kaip, kad pavyzdžiui, funkcija parašyta C kalba būtinai turi gražinti void tipo reikšmę [PdL02], ir sąsajai paduodamų objektų (parametrų) kiekis negali viršyti tam tikros ribos [Fou16o]. Tačiau, bent jau tipų suderinamumas tarp R aplinkos ir C funkcijos yra išlaikomas, nors ir ne visi R kalboje naudojami tipai gali būti pervesti į C programinę aplinką [Fou16o]. Taip pat naudojant šią programinę yra galimybė iškviesti kai kurias funkcijas, parašytas R aplinkai, kurios

[5] programinės sąsajos apžvalga paremta <2 .Internal vs .Primitive> žr. [Fou16f]

yra parašytos C/C++ kalbomis [Fou16t].

.Call – pagerinta .C programinės sąsajos versija [Fou16m]. Galima paduoti R kalbos objektus [Fou16u], kurti R objektus [Fou16n], kviesti R aplinkos funkcijas C kalbos programinėje aplinkoje [Fou16j]. Kaip ir .C programinės sąsajos atveju .Call yra ribojama paduodamų parametrų kiekiu [Fou16i]. .Call nuo .C skiriasi tuo, kad per .Call sąsają paduodami argumentai nėra kopijuojami kelis kartus [Fou16m], galima manipuliuoti sudėtingesnėmis duomenų struktūromis [Fou16j], tipų konvertavimas gali būti sprendžiamas C kodo aplinkoje, atminties valdymas vyksta nebe iš R programinės aplinkos, o iš C kodo. .Call taip pat reikalauja daugiau supratimo, apie R aplinkos programines struktūras ir tų struktūrų sąveika su C ir dalinai C++ kodu.

.External – patobulinta .Internal programinės sąsajos versija, savo funkcionalumu, ekvivalenti .Call programinei sąsajai. Skirtingai nei .Call gali priimti neribotą parametrų kiekį [Fou16i]. Praktikoje nėra labai dažnai taikoma, nes yra naujesnė nei .C ar .Call ir siūlomi privalumai beveik nesiskiria nuo .Call.

.External2 – naudojant .External2 sąsają, iškviestai C funkcijai, suteikiama galimybė pačiai spręsti ar aukštesnės eilės R išraiškos rezultatai bus spausdinami į konsolės langą [Fou16h]. .External2 yra pagerinta .External programinės sąsajos versija, su papildomais R sistemos vidiniais nustatymais [Fou16r].

.Rcpp – visos aukščiau aprašytos programinės sąsajos yra labiau orientuotos į jau sukompiliuoto C ir dalinai C++ kodo iškvietimą. Kai kurios iš jų leidžia panaudoti vidines R aplinkos funkcijas, parašytas C kalba, naujose, C kalba, rašomose funkcijose. Programinė sąsaja Rcpp, skirtingai, nei jau minėtos sąsajos, suteikia galimybę vykdyti pilnavertį C++ kodą [EFA+16], naudojant duomenų struktūras iš standartinės C++ bibliotekos, kurių R kalboje nėra, yra galimybė kodą organizuoti į klases [EF16](taikyti objektinio programavimo paradigmą) R aplinkoje, rašyti ir vykdyti C/C++ kodą pačioje R sistemoje, nekuriant atskirų failų [6].

Laikoma, kad Rcpp naudojimas vietoj .Call, .C, .External ar .External2, išorinio kodo rašymui, yra gera praktika [Wic16b]. Kai kurie R paketai, pavyzdžiui, Amelia [HKB16a], JAGUAR [HKB16b], dauguma kitų paketų [EFA+16] dėl našumo sumetimų yra perrašę dalį R funkcionalumo C/C++ kalbomis panaudojant Rcpp programinę sąsają. Rcpp, skirtingai nei jau minėtos sąsajos, nėra R sistemos sudėtinė dalis ir todėl neateina kartu su baziniu R sistemos funkcionalumu. Norint naudotis Rcpp teikiamais privalumais reikia atsisiųsti keletą papildomų įrankių ir funkcinių paketų. Per Rcpp ir kitas, aukščiau minėtas, programines sąsajas yra įmanoma vykdyti ne tik C, bet ir C++ kalba parašytas funkcijas ar programas.

[6] papildomai naudojant inline paketą. [SMS+16]

2.2. Fortran programinės sąsajos

.Fortran – programinė sąsaja iškviečiama per R sistemą. Leidžia vykdyti išorinį [Fou16p], ir vidinį R sistemos funkcionalumą parašytą FORTRAN kalba. Ši programinė sąsaja yra dalis R sistemos bazinio funkcionalumo.

2.3. Java programinės sąsajos

.JRI – programinė sąsaja [Urb16a] leidžianti iškviesti R išplėstinį funkcionalumą iš Java aplinkos [Ora16]. Ši programinė sąsaja nėra sudėtinė R sistemos dalis ir norint ją naudoti reikia sudiegti papildomas bibliotekas Java aplinkos pusėje taip pat reikia nustatyti kai kuriuos parametrus, nusakančius R sistemos buvo vietą kompiuteryje ir pan. Sąsaja yra ribota tuo, kad ji nėra naudojama iš R aplinkos pusės .RJI nuo 0.4-0 versijos yra sudėtinė kitos R-Java programinės aplinkos sąsajos dalimi pavadinimu rJava.

.rJava – programinė sąsaja [Urb16b] ekvivalenti .C ir .Call (žr. C/C++ programinės sąsajos). Leidžia kurti Java objektus kviesti metodus iš R sistemos. Ši programinė sąsaja – tai .JRI sąsajos atvirkštinė versija, jos naudojimas galimas tik iš R sistemos pusės. Kaip ir .JRI sąsaja, nėra bazinio R sistemos funkcionalumo dalis ir yra prieinama per paketų sistemą CRAN.

.RJ – atviro kodo biblioteka suteikianti įrankius ir programines sąsajas [Wal16] R sistemos funkcionalumo integravimui į Java ir Java taikomosioms programos kurti. Skirtingai, nei anksčiau minėtos programinės sąsajos, ši nėra R sistemos bazinio funkcionalumo dalis ir nėra pasiekiama per paketų sistemą CRAN.

2.4. Python programinės sąsajos

.rpy – programinė sąsaja [MW16] leidžianti naudoti ir kurti įvairius R objektus Python aplinkoje [Fou16b], iškviesti R sistemos funkcijas. Suteikia klaidų, atsirandančių vykdant R funkcijas ir naudojant R objektus, valdymą. Ši programinė sąsaja nėra bazinio R sistemos funkcionalumo dalis ir nėra prieinama per paketų sistemą CRAN.

.rpy2 – patobulinta .rpy programinės sąsajos versija, naujesniuose projektuose turėtų būti naudojama kaip .rpy sąsajos pakaitalas turi visą funkcionalumą būdingą .rpy ir suteikia papildomas galimybes [Gau16] išnaudoti R sistemos grafinėms bibliotekoms, ir kito pobūdžio vizualizavimo paketams pasiekti. Suteikia galimybę naudoti R tipo struktūras Python aplinkoje ir taip pat kaip ir rpy nėra bazinio R sistemos funkcionalumo dalis ir nėra pasiekiamas per paketų sistemą. Beje, veikia tik Python aplinkos pusėje.

.rPython – programinė sąsaja [Bel15] veikianti R sistemos aplinkos pusėje. Suteikia galimybę iškviesti Python kalba parašytas funkcijas. Rašyti python kodą R aplinkoje. Nėra bazinio R sistemos funkcionalumo dalis, tačiau gali būti pasiekiama per CRAN paketų sistemą.

2.5. .NET programinės sąsajos

R.NET suteikia galimybę .Net karkasui sąveikauti su R aplinka [JM15]. Leidžia R sistemos funkcijas ir objektus pasiekti iš .NET aplinkos pusės. R.NET nepriklauso baziniam R sistemos funkcionalumui ir nėra prieinamas per paketų sistemą.

R (D)COM Server – specialios paskirties programa kuri suteikia COM [Mic16d] programinę sąsają ir galimybę naudoti kitus COM objektus bei Active X valdiklius įvairioms Windows operacinės sistemos programoms ir taip pat gali būti naudojamas kaip programinės įrangos Microsoft Excel papildinys, išplėsti Excel duomenų analizės galimybes R sistemos funkcionalumu. Nėra bazinė R sistemos funkcionalumo dalis ir nėra pasiekiama per paketų sistemą, tačiau galima atsisiųsti kaip papildomą programinę įranga [Fou16s].

.rClr – programinė sąsaja [Mic16b] kuri leidžia vykdyti .NET karkaso pagrindu parašytas programas. Ši sąsaja yra ekvivalenti rJava programinei sąsajai. Veikia iš R sistemos aplinkos. Nėra sudėtinė R funkcionalumo dalis ir vis dar nepasiekiama per paketų sistemą.

2.6. Kitos programinės sąsajos

R sistema per programines sąsajas gali būti naudojama su programinėmis aplinkomis kaip, pavyzdžiui, Pascal [swi09], Ruby [DC09] ir kitomis, kurioms yra sukurtos programinės sąsajos. Sistemos funkcionalumą taip pat galima pritaikyti ir žiniatinklio taikomosioms programoms kurti panaudojant, pavyzdžiui, .Shiny [RSt14] ar .rApache [Uni16]. Arba galima tiekti/naudoti įvairias funkcijas per serverį panaudojant rServer [Mic16a].

Programinės sąsajos naudojamos tam, kad būtų galima R sistemos aplinkos bazines funkcijas iškviesti kitose aplinkose arba panaudoti išorinių aplinkų parašytas programas/metodus/funkcijas, siekiant apeiti tas sritis, kurios, pavyzdžiui, rašant kodą kita programavimo kalba gali būti geriau optimizuotos kitose aplinkose. Be kodo optimizavimo, R sistema suteikia galimybę atvaizduoti gautus/turimus duomenis diagramų, grafikų, lentelių, kitų objektų pavidalu. Sekantis skyrius skirtas R sistemos bazinio ir išplėstinio duomenų vizualizavimo galimybės ištirti.

3. Duomenų vaizdavimo galimybės R sistemoje

Ankstesniame skyriuje pateikėme kai kurias R programines sąsajas ir aptarėme jų panaudojimo atvejus R sistemos funkcionalumui išplėsti. Sekantis žingsnis – išanalizuoti galimybes, kurias suteikia pati R aplinka. Šį dalis skirta panagrinėti išplėstiniam R funkcionalumui – gautų ir/ar turimų duomenų perteikimui grafiškai (atlikti grafinę analizę) galimybes. Nagrinėdami duomenų vizualizavimą, pateiksime tiek bazinį tiek išplėstinį funkcionalumą – pasiekiama per paketų sistemą CRAN. Panagrinėsime atvejus, kai per R aplinką, duomenų grafinei analizei atlikti, naudosime išorines taikomąsias programas per jau anksčiau minėtas programines sąsajas.

3.1. Duomenų vaizdavimas grafikais/diagramomis

Graphics – bazinis funkcijų rinkinys ateinantis kartu su įprastiniu R sistemos įdiegimo paketu [Fou16l]. Naudojamas kintamųjų tarpusavio ryšio diagramom, stulpelinėms diagramoms, histogramoms, stačiakampėms diagramoms, funkcijų grafikams, sklaidos diagramoms, skritulinėms diagramoms, poligonams, spinogramoms, žvaigždinėms diagramoms paišyti. Taip pat galima koreguoti diagramų, grafikų, atvaizdavimo galimybes, kaip, pavyzdžiui, pakeisti diagramų spalvą/-spalvas, padidinti diagramos mastelį, pridėti paaiškinamuosius stulpelių ir kitų diagramų/grafikų dalių aprašus. Didžiausias trūkumas yra tai, kad nėra galimybės grafiko koreguoti interaktyviai. trūksta trivialaus trimačio grafikų vaizdavimo funkcionalumo.

Lattice – Išplėstinis R sistemos funkcijų rinkinys (paketas) pasiekiamas per CRAN sistemą. Kaip ir su graphics paketu, galima daryti įvairiausio pobūdžio įprastus grafikus ir diagramas. Lattice funkcijų rinkinys turi metodus generuoti trimačius sklaidos grafikus ir dvimates bei trimates paviršiaus diagramas. funkcijų kvietimai grafikams ir diagramoms vaizduoti yra paprastesni (lyginant su įprastom R sistemos galimybėmis). Lattice paketas suteikia daugiau įvairių galimybių daugiamačių duomenų vaizdavimui ir yra pranašesnis savo siūlomų galimybių aibe [Sar16], lyginant su Graphics siūlomų metodų gausa, tačiau nėra absoliutus pakaitalas bazinėms grafikų braižymo funkcijoms – braižant tam tikro tipo diagramas, pavyzdžiui, spinogramas, paprasčiau naudoti graphics bazines funkcijas. Su lattice Kaip ir su graphics, nėra galimybės gautais grafikais ir diagramomis naudotis interaktyviai.

ggplot2 – grafikų/diagramų kūrimo sistema [WCR16], kuri, kaip ir lattice, yra pasiekiama per CRAN sistemą. Skirtingai nei Graphics ir lattice paketai, neturi išreikštinų funkcijų vienokio ar kitokio pobūdžio grafikams/diagramos gauti. ggplot paketas vaizdinei duomenų analizei aprašyti naudojasi grafikų aprašymo gramatikos principais [Wil07]. Lyginant su jau anksčiau minėtais grafikų/diagramų braižymo paketais, suteikia kur kas profesionalesnę išvaizdą naudojant įprastinius

parametrus. Visas grafiko/diagramos dalis galima koreguoti. Yra galimybė išgauti trimačius grafikus/diagramas. Nėra galimybės su grafikais sąveikauti interaktyviai ir yra kur kas sudėtingesnė nei graphics bazinė sistema. Norint braižyti sudėtingesnius grafikus reikia itin gerai išmanyti šią sistemą ir su ja susijusius nuansus.

ggvis – išplėstinis R sistemos funkcijų rinkinys [CWR16] pasiekiamas per paketų sistemą CRAN. Grafikai/diagramos yra iškviečiami(-os) panašia sintakse į ggplot2 paketo. Tačiau skirtingai nei ggplot2 suteikia galimybę duomenų vizualizavimą atlikti interaktyviai. Duomenų vizualizavimui atlikti naudojasi grafikų gramatikos principais [Wil07]. Gautus grafikus/diagramas galima panaudoti internetiniuose puslapiuose išlaikant visas ggvis siūlomo interaktyvumo galimybes. Įmanoma paišyti trijų matmenų grafikus ir diagramas.

rCharts – kaip ir ggvis paketas, rChart yra papildomas R sistemos funkcionalumas [Vai16]. R sistemos pagalba gauti grafikai ir diagramos yra apdorojamos taip, kad jas būtų galima naudoti žiniatinklyje, kaip ir ggvis atveju interaktyvumas pridedamas per javascript bibliotekas, ir skirtingai nei ggvis paketo atveju, grafikai ir diagramos generuojami ne grafikų gramatikos principais, o tiesiog kaip ir lattice ar graphics atvejais per funkcijų kvietimus.

plotly – kaip ir ggvis taip ir plotly yra pasiekiamas per paketų sistemą CRAN [SPH+16] ir gauti grafikai/diagramos gali būti panaudoti internetiniuose puslapiuose. Galima generuoti trimačius grafikus ir diagramas, o atliekant duomenų analizę su, pavyzdžiui, demografiniais duomenimis, gautus rezultatus pavaizduoti pasaulio, šalių žemėlapiuose. Suteikia galimybę pavaizduoti daugybę diagramų/grafikų tipų – nuo sklaidos grafikų iki mokslinio tipo ir statistikoje naudojamų diagramų/grafikų. Suteikia galimybę interaktyviai sąveikai. Jau sugeneruotą grafiką/diagramą galima padidinti, sumažinti, išplėsti reikiamas intervalo sritis, kitus veiksmus.

googleVis – pridėtinės vertės R sistemos paketas [GdCC16] pasiekiamas per CRAN sistemą. Naudojantis googleVis paketu galima generuoti per naršyklę pasiekiamus grafikus ir diagramas. Įmanoma generuoti įprastas diagramas, histogramas, skritulines ir stulpelines diagramas ir kitas diagramas ir grafikus, kaip ir rCharts, ggvis ir plotly sugeneruotiems vaizdiniams suteikia galimybę interaktyviai sąveikauti, tačiau nėra galimybės generuoti trijų dimensijų grafikus/diagramas. Lyginant su plotly, nėra įprastinio funkcionalumo interaktyviai sutraukti ir išskleisti grafiko/diagramos. Naudojant googlevis be javascript funkcionalumą turinčių grafikų dar galima kurti flash tipo interaktyvias diagramas/grafikus.

rgl – dar vienas R sistemos paketas [AM16], kurį galima parsisiųsti per paketų sistemą CRAN. rgl skirtas braižyti trimačiams (ir kai kuriems dvimačiams) grafikams ir diagramoms. Galima kurti interaktyvias vizualizacijas. Paketas kurtas panaudojant OpenGL teikiamas galimybes. Ir skirtingai nei paketai rCharts, plotly, ggvis, googleVis nesuteikia galimybės dalintis gautomis diagramomis

ir grafikais per žiniatinklio puslapius išlaikant interktyvumo galimybes, tačiau yra galimybė eksportuoti gautus rezultatus įvairių tipų paveiksliukų formatais.

scatterplot3d – funkcijų rinkinys [LMS16], pasiekiamas per paketų sistemą CRAN. Trimačiams grafikams ir diagramoms išpiešti naudojasi paketo Graphics galimybėmis. Orientuotas į trimačių sklaidos diagramų generavimą ir suteikia tik bazinį funkcionalumą, pavyzdžiui, naudojantis scatterplot3d paketu galima nuspalvinti ir kitaip sužymėti erdvėje išsibarsčiusius taškus. Jokia interaktyvi sąveika nėra įmanoma. Lyginant su kitais R sistemos paketais, kurie siūlo tiek trimačius vaizdinius tiek interaktyvią sąveiką. scatterplot3d paketas savo siūlomų funkcijų ir galimybių gausa yra itin skurdus, lyginant pavyzdžiui, su googleVis, rgl, plotly, lattice, ggvis, ggplot, rCharts.

3.2. Kitos duomenų vaizdavimo galimybės

R sistema duomenų vizualizavimui grafikais ir diagramomis suteikia tikrai itin didelį pasirinkimą – pradedant nuo paprastų vaizdinių rezultatams pateikti ir baigiant interaktyvia sąveika žiniatinklio puslapiuose ar pačioje R sistemoje. Tačiau yra atvejų, kai R siūlomos duomenų vaizdavimo galimybės, lyginant su kitose sistemose esančiomis vizualizavimo galimybėmis yra pernelyg abstrakčios – tarsi galima išgauti visus reikiamus grafiko/diagramos parametrus, bet tam reikia gerai išmanyti R sistemą ir/ar naudojamo paketo galimybes dėl ko geriau naudotis nedidele programa, nors ir su ribotomis duomenų vaizdavimo galimybėmis. Dar galima susidurti su situacija kai nagrinėjamoji dalykinė sritis, duomenims atvaizduoti, diagramas ar grafikus naudoja labiau kaip antraeilį dalyką ir pats tyrimo objektas nėra tinkamas vaizduoti diagramomis, histogramomis ir grafikais. Be jau minėto bazinio ir išplėstinio R sistemos funkcionalumo, pasiekiamo per paketų sistemą dar įmanoma pasinaudoti R programinėmis sąsajomis (žr. 2. R programinės sąsajos) ir susieti išorinės sistemos galimybes su R aplinka arba padaryti taip, kad išorine sistema galima būtų naudotis per R sistemą. Visų išorinių programų, kviečiamų per R aplinką, bendra savybė yra ta, kad gali būti nepilnai įgyvendintas tarpusavio suderinamumas – kartais nutinka taip, kad išorinės sistemos kvietimas nepavyksta, nes, pavyzdžiui, naudojamos skirtingos versijos, ne ta programinė aplinka ir apskritai trūksta dar kelių kitų papildomų programų ir ar papildinių. Toliau aptariame R duomenų vaizdavimo galimybes per programines sąsajas.

ggobi – nuo R sistemos nepriklausoma taikomoji programa [SCLB16] skirta duomenų vizualizavimui. Naudojantis teikiamu funkcionalumu įmanoma duomenis apžiūrėti iš vienmatės ir dvimatės erdvės perspektyvų. Duomenų nagrinėjimas atliekamas interaktyviai. Taip pat įmanoma koreguoti vaizduojamus duomenis – kalibruoti, pervadinti, susiaurinti nagrinėjamą duomenų aibę pasirenkant duomenų poaibį, nustatyti įprastines reikšmes, reikšmių nebuvimo atvejais, įmanoma kurti

animacijas. Deja, tačiau pati Ggobi programa nėra sistema, kurios funkcionalumą būtų įmanoma išplėsti kaip, kad galima praplėsti R aplinkos teikiamas galimybes. Tačiau Ggobi teikiamą funkcijų rinkinį įmanoma naudoti per R aplinką – įsidiegiant papildomą paketą – rggobi [LSWL16]. Naudojant Rggobi galima užkrauti duomenų aibes, nustatyti duomenų vaizdavimo ypatybes, atlikti kitus veiksmus, kuriuos galima daryti naudojantis GGobi grafine sąsaja. Tam, kad per rggobi būtų įmanoma iškviesti ggobi funkcijas, rggobi naudojasi .Call ir .C programinėmis sąsajomis, kuriomis yra iškviečiamos funkcijos – esančios rggobi.dll[7] faile.

iplots – interaktyviam duomenų vizualizavimui skirtas funkcijų rinkinys [UW16], parašytas java programavimo kalba, iškviečiamas per R aplinką panaudojant rJava programinę sąsają. Turi galimybę atvaizduoti diagramas, histogramas, stačiakampes diagramas, įmanoma atvaizduoti duomenis pagal šalių žemėlapius, daryti sklaidos, paralelių koordinačių grafikus/diagramas. Interaktyvumas, naudojant iplots paketą, galimas tiek nagrinėjant grafikus juos padidinant/sumažinant, pasirenkant apžiūrėti poaibį taškų, tiek siejant kelias diagramas/grafikus. Kaip ir ggobi atveju, iškvietus iplots paketo kurią nors funkciją atsidaro grafinė naudotojo sąsaja, kurioje galima atlikti įvairius pakeitimus gautiems grafikams ir diagramoms. Tačiau kaip ir ggobi taikomosios programos atveju, pridėti papildomas funkcijas, kurios nėra sukurtos – neįmanoma nepapildant/neperdarant paketo, kurioje saugomos funkcijos, šiuo atveju, iplots.jar[8] failo. Reiktų atkreipti dėmesį, kad šiam funkcijų rinkinio veikimui neužtenka įsidiegti vien tik paketo per CRAN sistemą, taip pat reikia ir Java aplinkos. Naudojantis iplots teikiamomis galimybėmis įmanoma generuoti tik dviejų matmenų vizualizacijas. Gautas diagramas ir grafikus įmanoma išsaugoti į *.pdf, *.svg, *.postScript, *.pgs ir *.png formatais.

Tiek ggobi tiek iplots yra išorinės taikomosios programos skirtos įprastam duomenų vizualizavimui – funkcijų grafikais ir diagramomis, leidžia atlikti kai kuriuos interaktyvius veiksmus. Ir yra suderintos su R aplinka per įvairias programines sąsajas. Apsikeitimas duomenimis su R sistema yra vienkryptis – iš R pusės galima paduoti įvairius parametrus ir grafikų/diagramų nustatymus, bei duomenis, tačiau išorinėse programose pakeistus duomenis gražinti atgal į R, funkcionalumas nėra įgyvendintas. Be grafikų ir diagramų dar galima analizuoti duomenis grafiškai ne vien iš grafikų ir diagramų – galima pasirinkti ir kitokią grafinę išraišką, pavyzdžiui, unikaliai pagal dalykinę sritį ir duomenų prasmę – vien grafikų ir diagramų gali nepakakti.

igraph – nuo R sistemos nepriklausantis funkcijų rinkinys [Csa16] skirtas objektų tinklų analizei ir tinklo mazgų vizualizavimui atlikti. Kaip ir anksčiau aprašytuose paketuose taip ir naudojantis igraph galima paišyti grafikus ir diagramas – įmanoma kurti skritulines ir dendogramamas.

[7] *.dll failus galima rasti paketų kodo išvestyse.
[8] .jar failas gali būti rastas paketo kode.

Pagrindinis šio paketo tikslas yra atvaizduoti tinklų sistemas, tas sistemas sudarančius mazgus ir ryšius tarp mazgu remiantis grafų teorijomis. Taigi, pirminiai duomenys vaizdavimui yra mazgai ir ryšiai įeinantys/išeinantys į kitus mazgus, o vizualizacijose – paišomi grafai. Gautas tinklų diagramas galima modifikuoti taip, kad būtų rodomas grafas iš poaibio mazgų ir atlikti kitus veiksmus ir modifikacijas. Igraph projektas yra vystomas trimis kryptimis – viena šaka skirta funkcijų derinimui su R sistema, kitos dvi – atitinkamai python ir C/C++ aplinkoms. Su R aplinka igraph paketas sąveikauja per C/C++ programines sąsajas ir naudoja kitus, R pritaikytus paketus – rgl, tcltk. Deja, tačiau naudojantis igraph teikiamais privalumais – gauti grafai ir diagramos nesuteikia įrankių interaktyvumui ir trijų matmenų vizualizavimui atlikti.

Cytoscape – taikomoji programa [Con16] savo panaudojimo galimybėms artima igraph paketui, tačiau dėmesys yra skirtas kitokio pobūdžio tinklų ir sąveikų atvaizdavimui ir modeliavimui – cytoscape orientuojasi į molekulių tarpusavio ryšių, kitų biologinių sąveikų atvaizdavimus. Cytoscape kaip ir R sistema turi savo programavimo/modeliavimo aplinką. Veikia java aplinkoje, papildomos funkcijos pridedamos per papildinius. Cystoscape funkcionalumas nėra priklausomas nuo R aplinkos, tačiau per papildomą funkcijų paketą – RCystoscape [Sha16], pasiekiamą per CRAN sistemą, galima suderinti abiejų aplinkų teikiamas galimybes, nors tas suderinamumas ir yra ribotas – įmanoma iš R aplinkos kontroliuoti ir derinti tinklo, sąveikų, ir ryšių parametrus Cytoscape aplinkoje, tačiau atgalinis ryšys – perduoti duomenis į R aplinką, šio darbo rašymo metu, vis dar nėra įmanomas. Skirtingai nei igraph paketas, suteikia galimybe grafus matyti trimatėje erdvėje, taip pat yra galimybė interaktyviai sąveikai.

Aptarėme daugybę įrankių, kuriais galima panaudoti R sistemą duomenų vizualizavimui ir grafinei analizei atlikti. Aprašėme esminius panaudojimo būdus ir pateiktų įrankių ribotumą. Nurodėme papildomas (nuo R nepriklausomos programinės įrangos) susiejimo atvejus, duomenims tirti. Pavyko atrasti ir tuos atvejus, kur R sistema išplečia savo galimybes per programines sąsajas, panaudodama išorines taikomąsias programas.

R sistema be grafinės analizės ir duomenų atvaizdavimo galimybių dar gali būti pritaikyta ir duomenų matematinei, statistinei, funkcinei, fizikinei, cheminei, biologinei, geografinei, demografinei duomenų analizei. Sekančiuose skyriuose analizuosime R sistemos galimybes manipuliuoti duomenimis, taikant matematinius ir statistinius duomenų tyrimo modelius.

4. R sistemos duomenų analizės galimybės

Skyrius skirtas aptarti R galimybes, kurios padeda atlikti įvairias duomenų matematines ir statistines analizes (grafines, R sistemos, analizės subtilybes jau aptarėme). Kaip ir ankstesniuose skyriuose panagrinėsime bazinius ir papildomus funkcijų rinkinius.

4.1. Įprastiniai funkcijų rinkiniai

stats – R sistemos funkcinis paketas [Fou16k] skirtas statistinei duomenų analizei atlikti. Galima apskaičiuoti atsitiktinių didžių pasiskirstymo funkcijas pagal binominį, beta, Koši, chi-kvadrato, eksponentinį, gamma, geometrinį, hipergeometrinį, Poasono, Studento, ir kitus modelius. Dar vienas svarbus funkcionalumas – hipotezių tikrinimas. Stats paketas siūlo hipotezių tikrinimą pagal t-kriterijų nepriklausomoms imtims, t-kriterijų priklausomoms imtims taip pat ir pagal Kolmogorovo-Smirnovo kriterijų (skirstinio normalumui tikrinti), kitus kriterijus. Su duomenimis galima atlikti analizes remiantis regresiniais modeliais.

datasets – paketas duomenų rinkinių [Tcw16] su kuriais galima atlikti statistinius tyrimus. Į paketą įeina realių duomenų ištraukos iš įvairių sričių – ekonomikos, biologijos, medicinos, chemijos, matematikos, kitų sričių. Šis paketas, apart, duomenų aibių, nesuteikia jokio kito funkcionalumo.

MASS – funkcijų ir duomenų aibių rinkinys [RVB+16] pagrįstas "Modern Applied Statistics with S"knygoje išdėstytais ir aprašytais tyrimo metodais ir funkcijomis. Naudojant MASS galima daryti veiksmus su matricomis, grafinę analize, atlikti duomenų kalibracija, diskriminantinę analizę, kvadratinę diskriminantinę analizę, tirti duomenis naudojant regresinės analizės modelius.

Matrix – R aplinkoje galima atlikti įprastus veiksmus su matricomis – jas sudėti, sudauginti, gauti vektorinę sandaugą, transponuoti, kurti diagonalines matricas, spręsti tiesines lygtis, atlikti dekompozicijas, veiksmus su pavieniais matricų stulpeliais ir eilutėmis. Paketas matrix [BM16] dar labiau išplečia R sistemos galimybes manipuliuoti matricomis – efektyviau išnaudoja procesoriaus resursus dirbant su nepilnai inicializuotomis matricomis, suteikia daugiau metodų matricos dekompozicijai atlikti, turi daugiau funkcijų įvairioms factorizacijoms, turi papildomas funkcijas patikrinti matricos simetriškumą, nustatyti kitas savybes. Suteikia daugiau algoritmų efektyvesniai matricų sandaugai, sudėčiai atlikti. Skirtingai nei bazinės R aplinkos galimybės darbui su matricomis, atsižvelgia į matricų dydžius ir algoritmai veiksmams su matricomis yra perrašyti C/C++ kalba. Iki R 2.9.0 sistemos versijos, matrix paketas buvo priskiriamas išplėstinėms R galimybėms, tačiau vėlesnėse versijose ateina kartu su įprastiniu R sistemos instaliavimo paketu ir yra pasiekiamas per paketų sistemą CRAN.

boot – R sistemos paketas [CR16a] turintis duomenų rinkinius ir funkcijas duomenims gene-

ruoti remiantis *bootsrap* metodais aprašytais "Bootstrap Methods and Their Applications"knygoje. Duomenų generavimą galima atlikti imant atsitiktines (arba remiantis kitokiais atrinkimo metodais) reikšmes iš vektorių, duomenų masyvų, matricų. Yra galimybė parinkti duomenų atrinkimo metodus ir parinkti įvairias statistikas. Kitus kriterijus. Pasiekiamas per paketų sistemą CRAN ir yra įprastinėje R sistemos instaliavimo programoje.

class – funkcijų rinkinys [CR16b] skirtas duomenų klasifikavimui atlikti. Pakete įgyvendintas saviorganizuojančių neuroninių tinklų (SOM) teorijomis pagrįstas klasifikatorius ir jo variacijos, k artimiausių kaimynų klasifikatorius, artimiausio kaimyno, lvq klasifikatorius ir jo variacijos. Įgyvendinti tik kai kurie klasifikatoriai, o klasifikatorių pasirinkimas yra nedidelis. Dar naudojant class paketą įmanoma atvaizduoti rezultatus gautus naudojant SOM klasifikatorių. Kaip ir boot paketas, class yra pasiekiamas per paketų sistemą CRAN ir ateina kartu su įprastine R sistemos instaliavimo programa.

cluster – paketas [MRS+16] kuriame yra metodai, duomenų rinkiniai ir klasterių vaizdavimo funkcijos duomenų analizei atlikti remiantis klasterizavimo (duomenų grupavimo) teorijomis išdėstytomis knygoje "Finding Groups in Data: An Introduction to Cluster Analysis". Su įgyvendintais metodais, duomenims, galima atlikti hierarchine klasterizavimo analizę, padalinti aibes į n grupių, apskaičiuoti atstumus ir kitus įverčius tarp gautų klasterių, kiti įgyvendinti klasterizavimo algoritmai – diana *(DIvisive ANAlysis Clustering)*, fanny *(Fuzzy Analysis Clustering)*, mona *(MONothetic Analysis Clustering of Binary Variables)*, pam *(Partitioning Around Medoids)*. Paketas ateina kartu su R sistema ir gali būti pasiektas per paketų tinklą CRAN.

codetools – kartais paketuose esantis funkcionalumas nėra išbaigtas ir ko pasekoje gali prireikti pasirašyti papildomą funkciją ar funkcijas R kalba, kuri/kurios atliktų kokius nors pridėtinės vertės skaičiavimus ar veiksmus. codetools paketas [Tie16] suteikia įrankius (funkcijų pavidalu) analizuoti kodą parašyta R kalba. Įmanoma patikrinti ar R kodas parašytas laikantis standartų. Galima surasti visas globaliai aprašytas funkcijas ir kintamuosius. Išspausdinti R išraiškos medį. Funkcijų rinkinys ateina kartu su R sistemos įdiegimo failu ir gali būti parsiųstas per paketų tinklą CRAN.

nnet – metodų rinkinys [RV16] skirtas tiesioginio sklidimo *(feed-forward)* neuroniniams tinklams ir polinominiams tiesiniams logaritminiams modeliams apmokyti. Tiesa, kalbėdami apie neuroninius tinklus galvoje turime tik vienasluoksnį neuronų tinklą. Naudojant nnet paketą galima apskaičiuoti neuronų tinklų Hessiano matricą. Apmokius tinklą galima naudoti naujų reikšmių klasifikavimui. Savo siūlomų funkcijų gausa nnet nėra itin turtingas. Siūlo tik du modelius, kuriais galima klasifikuoti duomenis. Ateina kartu su standartine R instaliavimo programa ir gali būti pasiektas per paketų sistemą CRAN.

rpart – metodų ir duomenų rinkinys [TAR16] suteikiantis galimybę rekursyviai skaidyti duo-

menis klasifikavimo, regresijos uždavinių sprendimui ir/ar analizei atlikti. Paketas ateina kartu su R sistemos instaliavimo failu ir gali būti pasiektas per CRAN tinklą.

spatial – dar vienas paketas [RBV16] skirtas duomenų analizei atlikti. Suteikia metodus interpoliacijai vykdyti ir taškų išsidėstymo modeliams erdvėje tirti. Funkcijų rinkinys ateina kartu su R sistemos diegimo paketu ir yra prieinamas per paketų sistemą CRAN.

survival – R sistemos paketas [TL16] suteikiantis duomenų rinkinius, metodus, ir modelius išlikimo analizei atlikti. Tarp įgyvendintų analizės metodų yra Aleno regresinio modelio, sąlyginės regresijos, cox regresinio modelio implementacijos. Gautus rezultatus galima atvaizduoti grafikais. Duomenų rinkiniai, kurie ateina su paketu, yra klinikinio pobūdžio – ir gauti, pavyzdžiui, atlikus medicininius vėžio, cirozės, kiaušidžių vėžio, kitų ligų tyrimus. Kaip ir anksčiau minėtas funkcionalumas, šis taip pat yra pasiekiamas per R tinklą CRAN ir ateina kartu su R sistemos diegimo programa.

4.2. Papildomi funkcijų rinkiniai

Be paketų repozitorijos [CRAN], funkcijų rinkinius pritaikytus R sistemai, galima parsisiųsti iš įvairių vietų, kurias apima funkcijų kūrėjų puslapiai, kitos projektų vystymo repozitorijos ir galiausiai papildomomis funkcijomis galima dalintis iš „rankų į rankas“. Šis skyrelis skirtas papildomų metodų aptarimui iš paketų sistemos *https://r-forge.r-project.org/*. Projektas *r-project.org*, kaip ir CRAN, yra R sistemos paketų repozitorija ir platforma su R sistema dirbantiems žmonėms, kurioje galima dalintis sukurtais paketais, kurti naujas R funkcijas.

TraMineR – metodų rinkinys [GMRS16] suteikiantis galimybes atlikti sekų analizę. Pateikia sekų panašumo ir neatitikimų matricas pagal įvairias metrikas. Yra įmanoma atlikti sekų formato konvertavimus. Pirminis paketo tikslas yra socialinių mokslų duomenų sekų tyrimas. Tačiau, gali būti pritaikytas ir kitokio pobūdžio sekoms, pavyzdžiui, genomo, DNR, RNR. Suteikia, kai kurias galimybes, kurias siūlo BLAST [oM16].

TSP – funkcijų rinkinys skirtas keliaujančio prekeivio tipo uždaviniams spręsti, trumpiausio Hamiltono kelio radimui, klasterių analizei atlikti. Paketas siūlo kelias metrikas pagal kurias galima ieškoti trumpiausio kelio – artimiausio kaimyno, įterpimo, k-opt, ir kitus. Deja, tačiau šio darbo rašymo metu TSP metodų rinkinys nebėra atnaujinamas *https://r-forge.r-project.org/* projekte, o naujausią versiją galima rasti *github* tinklapyje adresu *https://github.com/mhahsler/ TSP*.

ade4 – paketas skirtas daugiamačių ekologinių duomenų aibių analizei ir atvaizdavimui. Siūlo daugybę metodų tarp kurių yra paprogramės skirtos principinių komponenčių, principinių koordinačių, duomenų sklaidos modelių, duomenų koreliacijos atsitiktinumo, duomenų sklaidos tarp

augalų bendruomenių, tiesinės diskriminantinės analizės ir kitiems tyrimams atlikti. Šis funkcijų rinkinys apima ir ekologinius duomenų rinkinius, bei pagalbines funkcijas įvairioms transformacijoms atlikti. Rinkinys funkcijų gali būti pasiektas per paketų sistemą CRAN [DDT16] ir *https://r-forge.r-project.org/*.

rms – funkcijų rinkinys [Jr16] regresinei analizei ir rezultatų vaizdavimui atlikti. Labiausiai Orientuojasi į ordinalių ir binarinių kintamųjų regresiją. Deja, tačiau [https://r-forge.r-project.org/] yra paketo versija tinkama senesnei R sistemai (<3.2.0), tačiau CRAN sistemoje radome paketą, kuris tiko ir naujausiai (=3.2.3) R sistemos versijai.

caret – suteikia funkcijas klasifikavimo ir regresijos modelių treniravimui ir gautų rezultatų atvaizdavimui [Kuh16]. Į funkcijų paketą įeina metodai skirti k-artimiausių kaimynų klasifikavimui ir regresijai atlikti. Turi pagalbinius metodus panašumo matricoms gauti, bei duomenų rinkinius.

klaR – paketas turintis funkcijas duomenų klasifikavimui, vizualizavimui, lokaliai diskriminantinei analizei, RDA, kitiems klasifikavimo ir regresijos modeliams, atlikti. Paketas pasiekiamas per CRAN [RRL+16] ir *https://r-forge.r-project.org/*.

Aptarėme, kai kuriuos R sistemos funkcijų rinkinius ir tų rinkinių teikiamą funkcionalumą, tačiau dar daugybė liko nepaminėtų. Iš jau aprašytų paketų, galima susidaryti įspūdį, kad R sistemoje, yra funkcinių rinkinių, kurie dubliuojasi ir/arba papildo vienas kitą. Taip pat pastebėjome, kad ne visos paketų versijos, skirtingose repozitorijose, turi visas paketų versijas pritaikytas visoms R sistemos – pavyzdžiui, paketų sistemoje *https://r-forge.r-project.org/* savo R sistemai neradome tinkamos paketo **rms** versijos, tačiau viskas susitvarkė, kai paieškojome to paties paketo CRAN sistemoje. Dar atkreipėme dėmesį į paketų tarpusavio ryšius – kai kurie iš jų naudojasi kitų teikiamais metodais ir todėl atsinaujinant visą sistemą gali atsirasti tam tikrų nepatogumų, kai, pavyzdžiui, turime papildomas funkcijas, parašytas tam tikrai sistemos versijai (po atnaujinimo kas veikė, gali nebeveikti, nes ne visi paketai gali atsinaujinti, nes yra išplėstinio funkcionalumo, nepalaikomo R komandos, dalimi arba gali tiesiog duoti klaidingus rezultatus). Dažnu atveju, funkcijas galima naudoti nežinant implementacijos detalių, bent jau kalbant apie R komandos įgyvendintus metodus, tačiau naudojantis paketais iš kitų šaltinių, gali būti pavojinga – neužtenka žinoti vien metodo uždaviniui spręsti, reikia gerai išmanyti ir metodo įgyvendinimo detales, todėl gali teki panagrinėti ir kodą.

5. R sistema ir molekulinė biologija

Iki šiol aptarėme tik įprastinį R sistemos funkcionalumą, nurodėme kai kurių funkcinių paketų panaudojimo atvejus ir dalinai apžvelgėme išplėstines galimybes. Šiame skyriuje toliau gilinsimės į išplėstines R galimybes, visą dėmesį skirdami molekulinės dinamikos sričiai ir metodų rinkiniams sukurtiems spręsti problemas keliamas molekulinėje biologijoje.

Molekulinė biologija – sritis nagrinėjanti molekules, molekulinius mechanizmus, molekulių, molekulinių mechanizmų tarpusavio sąveiką organizmuose ir poveikį organizmams [Dic16].

5.1. Funkcijų rinkiniai molekulinės biologijos uždaviniams spręsti

OOMPA – paketų rinkinys sukurtas R sistemai ir skirtas genų ekspresijos analizei atlikti, padeda įvertinti kiekį chromosomų esančių ląstelėje (iš SNP masyvo), nustatyti pasikartojančias sekas SNP masyvuose ir sekvenavimo metu gautuose duomenyse, turi algoritmus duomenų klasterizavimui ir kryžminei patikrai mikromasyvams atlikti. Suteikia genetinius ir kitokio pobūdžio modelius požymių paieškai ir spėjimas atlikti. Asistuoja sekų išlyginimo (sugretinimo) uždaviniuose. Paketas turi ir statistikas biožymeklių atrinkimui iš mikromasyvų. Pasiekiamas per projekto repozitoriją *http://oompa.r-forge.r-project.org/*.

MotIV – funkcijų rinkinys skirtas motyvų analizei, padeda sekose identifikuoti transkripcijos faktorius. Motyvus ir jų pasiskirstymą taip pat galima vizualizuoti motyvų pasiskirstymą. Paketas pasiekiamas per biocoductor projekto repozitoriją [MG16].

Rsamtools – suteikia metodų rinkinius ir programines sąsajas sekų išlyginimo failams manipuliuoti. Pasiekiamas per paketų sistemą bioconductor [MPVO16].

BioMart – programinių sąsajų rinkinys įvairioms biologinių duomenų bazėms pasiekti. Suteikia įrankius atlikti įvairias užklausas. Ir duoda galimybę parsisiųsti aktualias biologinių organizmų sekas. Paketas gali būti pasiektas per biocoductor projekto repozitoriją [DH16].

GenomicFeatures – paketas kuris leidžia parsiųsti aktualias transkriptorių vietas išsaugoti jas lokalioje duomenų bazėje iš UCSC ir BioMart funkcijų paketo pasiekiamų duomenų bazių. Suteikia paieškos galimybes pagal tyrėjui aktualius požymius. Paketas pasiekiamas per bioconductor projekto repozitoriją [CPA+16].

biovizBase – turi metodus ir funkcijas žymėti sekoms, nukleotidams, ryšiams, turi spalvų schemas kitiems biologiniams duomenims (sekoms) žymėti. Vienas iš paketo keliamų tikslų – suteikti bendras konvencijas duomenų vaizdavimui ir sekų žymėjimui. Naudojamas kaip papildinys kituose paketuose ir funkcijose. Kaip ir ankstesni paketai, gali būti pasiektas per projekto bioconductor repozitoriją [YLC16].

GenomicAlignments – funkcijų rinkinys kurio pagalba galima manipuliuoti genomo sekomis, aptikti genomo sekų persidengimo vietas sutampančias su transkriptoriais, turi metodus sekų išlyginimams daryti, sekų neatitikimams ir tarpams rasti. Paketas pasiekiamas per bioconductor repozitoriją [POM16].

motifStack – R sistemai sukurtas paketas, kuriuo galima atvaizduoti DNR, RNR, aminorūgščių sekas sudarančių elementų dažnius ir reikšmingumą remiantis informacijos teoriją. Funkcijų rinkinys prieinamas per projekto bioconductor repozitoriją [OBWZ16].

BSgenome – padeda atrasti nukleotidų seką/sekas arba ieškomų/-(os) sekų/sekos skaičių chromosomoje. Paieškai naudoti galima pagal sudarytą sekų žodyną. Metodų rinkinys pasiekiamas per projektą bioconductor [Pag16].

Biostrings – turi funkcijas daryti sekų išlyginimui, sekų nepriklausomumo analizei; elementų, esančių sekoje, dažniui ir ilgiausios sekos radimui. Pakete taip pat yra metodų DNR sekos išvertimui į RNR seką, sekos elementų sukeitimui, iškirpimui, įterpimui ir sujungimui. Turi kai kurias biologinių organizmų sekas. Paketas gali būti rastas bioconductor projekto puslapyje [PAGD16].

GenomeInfoDb – funkcijų rinkinys turintis metodus manipuliuoti biologinių sekų informacija. Padeda išversti sekas pagal skirtingas sekų žymėjimo konvencijas. Paketas gali būti rastas bioconductor projekto repozitorijoje [AMCP16].

GenomicRanges – Paketas suteikiantis funkcijas genomo sekų intervalų ir genomo kintamųjų reprezentacijai (reikalinga efektyviam sekų palyginimui ir paieškai atlikti). Šis funkcijų rinkinys taip pat yra sudėtinė paketo *GenomicAlignments* dalis. Funkcijų rinkinys gali būti pasiektas per bioconductor puslapį [APL16].

Gviz – paketas skirtas biologinių sekų paieškai ir atrinkimui iš Ensembl ir UCSC duomenų bazių, gautų sekų statistiniam vizualizavimui pagal įvairius kriterijus. Funkcijų rinkinys yra sudėtinė bioconductor projekto dalis [HDI+16].

rtracklayer – programinė sąsaja tarp R sistemos ir genomo paieškos sistemų. Padeda išsaugoti genomo sekų anotacijas įvairių failų formatu. Yra galimybė naršyti genomo paieškos sistemose iš R pusės. Paketą galima parsiųsti iš bioconductor projekto [LCG16].

VariantAnnotation – suteikia metodus genomų sekų variantams anotuoti. Turi funkcijas aminorūgščių kitimams variantuose nuspėti, variantų tipų identifikacijai, ir variantų radimui. Paketas yra bioconductor projekto dalis [OMLG16].

TFMPvalue – funkcijų rinkinys kurio pagalba galima apskaičiuoti genomo sekoje rastos transkriptoriaus vietos reikšmingumą remiantis *p-value* metodu. Paketas savyje turi tik vieną metodą ir kelias pagalbines funkcijas. Kaip ir anksčiau minėtus paketus, šį galima atsisiųsti per CRAN projekto repozitoriją [Tan16].

motifbreakR – R paketas kuris turi metodus sekų, kuriose užfiksuotas polimorfizmas, patikimumo patikrinimui. Padeda įvertinti sekos informacijos kiekį. Turi metodus sekos reikšmingumui įvertinti pagal *p-value* metodą, turi kai kurių genomų motyvų duomenų bazes. Naudojantis paketu galima vizualiai atvaizduoti nagrinėjamą seką. Paketą galima parsisiųsti iš bioconductor projekto repozitorijos [CH16].

seqPattern – suteikia funkcijas vizualizuoti analizuojamos sekos dalis. Galima ieškoti sekose atitikimų pagal užduotą šabloną. Geba vizualiai pateikti šablono pasirodymo dažnį, kitas statistikas. Funkcijų rinkinį galima rasti biocondutor projekto puslapyje [Hab16].

genomation – metodų rinkinys skirtas genomo sekų statistiniam apibendrinimui ir grafiniam atvaizdavimui. Galima parsisiųsti iš bioconductor projekto puslapio [AFWIS16].

Visi šiame skyriuje išvardinti R sistemos funkcijų paketai vienaip ar kitaip yra susiję su molekuline biologija. Visus nagrinėtus paketus galima suskirstyti į kelias grupes textendash paketai skirti duomenų grafiniam vaizdavimui, biologinių sekų ir kitokių biologinių objektų statistinei analizei, duomenų paieškai išorinėse duomenų bazėse.

Ieškodami ir nagrinėdami paketus, kurie padėtų vizualizuoti biologinius duomenis, pastebėjome, kad vaizdavimas yra itin primityvus – statistinių duomenų grafikams išpiešti užtenka elementarių R galimybių. Paketų, kurie statistiniam vaizdavimui naudotųsi kitų taikomųjų programų teikiamomis funkcijomis, neradome (nagrinėjome tik rinktinius paketus esančius bioconductor ir CRAN projekto repozitorijose.).

Tie funkcijų rinkiniai, kurie atlieka statistinius skaičiavimus taip pat remiasi tik R sistemos galimybėmis – naudojasi jau sukurtų išorinių ir vidinių paketų siūlomais metodų rinkiniais; intensyviam skaičiavimui atlikti kviečia paprogrames per C/C++ programines sąsajas. Šiame skyriuje aptarti paketai turi išsamią dokumentaciją ir panaudojimo atvejų aprašymus. Norime atkreipti dėmesį į tai, kad paketai yra kurti skirtingų autorių ir todėl gali turėti skirtingas naudojimo licenzijas, kurios gali riboti pačių paketų naudojimo aplinkybes.

R sistemai skirti paketai gali atlikti ir programinių sąsajų su išorinėmis duomenų bazėmis funkcijas. Pasinaudojant išplėstinėmis R galimybėmis, kurios padeda paduoti užklausas, palaikyti abipusį ryšį, yra įmanoma atlikti skaičiavimus, statistinę analizę ir gautus rezultatus atvaizduoti grafiškai.

Iki šiol skyriuje nagrinėjome tik funkcionalumą, kuri galima įgyvendinti panaudojant vien tik R galimybes daryti skaičiavimus per C/C++ sąsajas, atlikti bazinį grafinį duomenų vaizdavimą. Šio skyriaus pabaigoje pateiksime dar vieną R sistemai skirtą paketą, kuris apjungia R galimybes vaizduoti duomenis, daryti statistinę analizę ir sąveikauti su išorinėmis taikomosiomis programomis.

bio3d – R sistemos paketas [GYSI16] suteikiantis įrankius biomolekulių, biologinių sekų, bei biomolekules sudarančių komponentų trajektorijos analizei atlikti. Funkcijų rinkinys turi metodus biologinių duomenų formato failams (PDB [rcs16]) skaityti, rašyti, atlikti failų padalinimą ir suliejimą. Geba išrinkti aktualius duomenis (atomų koordinates, kitą papildomą informaciją). Bio3d taip pat gali išlyginti biologinių struktūrų sekas ir tam tikslui naudojasi išorinės taikomosios programos MUSCLE [Edg04] funkcionalumu.

Po sekos išlyginimo dar suteikia funkcijas sekai išgryninti (išima tas sekos dalis kurios turi didelį nuokrypį bendros sekos struktūros atžvilgiu, atliekant skaičiavimus remiasi trajektorijų duomenimis). Pakete yra įgyvendintas funkcionalumas daugybinių biologinių objektų (molekulių, atomų) ryšiams identifikuoti ir žymėti, invariantams ieškoti. Detalesnei analizei atlikti turi funkcijas, kuriomis galima konstruoti atskirus biologinius objektus į junginius. Geba atlikti įprastą statistinę analize ir gautus duomenis atvaizduoti grafikais, histogramomis. Biologinių struktūrų konformacijų ryšiams tirti galima pasinaudoti įgyvendinta principinių komponenčių metodu, arba atlikti kryžminę koreliacijos, biomolekulinių struktūrų persidengimo analizę.

Naudojantis bio3D galimybėmis galima atvaizduoti ne vien analizės rezultatus. Dažnu atveju, nagrinėjant biomolekules ir įvairių kitų biologinių darinių sudėtines dalis svarbu atvaizduoti ir patį tiriamąjį objektą. Naudojantis teikiamu funkcionalumu galima atvaizduoti biomolekulių tinklus, pavienius, biomolekules sudarančius, komponentus ir tų komponentų dalis, tačiau atvaizdavimui neužtenka vien R sistemos galimybių – yra keletas išorinių taikomųjų programų, skirtų biologinių struktūrų peržiūrai, kurioms bio3d gali sugeneruoti failus, kuriuos vėliau galima užkelti į PyMol, Jmol, ar netgi dalintis grafikais ir kitais analizės rezultatais žiniatinklyje.

Iš visų skyriuje nagrinėtų paketų, kurie padėtų atlikti užduotis susijusias su molekuline biologija, paketas bio3d suteikia gausiausią funkcijų rinkinį ir yra labiau panašus į atskirą taikomąja programa, nei paketą, nes naudojasi ne vien R sistemos teikiamomis galimybėmis. Bio3d funkcijų rinkinį galima parsisiųsti iš *http://thegrantlab.org/bio3d* puslapio.

6. Literatūros apžvalgos išvados

Išanalizavę R sistemos teikiamas programines sąsajas galime teigti, kad R turi tikrai daugybę programinių sąsajų su įvairiomis programinėmis aplinkomis, tačiau tik Fortan, C/C++ programinėms aplinkoms skirtos sąsajos (išskyrus Rcpp) yra palaikomos R sistemos kūrėjų, kitų programavimo aplinkų sąsajos, pavyzdžiui, java, .NET, Pascal yra palaikomos tik išorinių projektų, ir programuotojų pastangomis ir todėl naudoti kitas programines sąsajas negu Fortan ir C/C++ gali būti pavojinga – projektas, vykdantis kurios nors aplinkos sąsajos kūrimą ir palaikymą paprasčiausiai gali nustoti egzistavęs.

Ištyrę kai kurių R sistemos paketų galimybes galime drąsiai teikti, kad R sistemą siūlo labai įvairaus spektro funkcijas duomenų tyrimui ir analizei atlikti. Paruoštas funkcionalumas turi dokumentaciją ir panaudojimo atvejų aprašymus. Tačiau vargu, kad itin rimtiems moksliniams tyrimams visi siūlomi paketai yra tinkami – nėra aišku kas tuos paketus kūrė – programuotojas turintis tos srities gilias žinias, ar programuotojas mėgėjas. Taip pat pastebėjome, kad paketai nėra apibrėžti viena konkrečia licenzija, todėl gali atsirasti sunkumu norint naudoti, pavyzdžiui, komerciniuose projektuose.

Ištyrę duomenų grafinio atvaizdavimo galimybes. Galime užtikrinti, kad jos tikrai gana plačios. Įmanoma naudoti tiek bazinį funkcionalumą įprastiems grafikams braižyti, tiek išplėstinį textendas piešti grafus, vizualizuoti kompiuterių tinklus. Apmaudu, tačiau tiek bazinės tiek išplėstinės galimybės nepasiūlo grafinės naudotojo sąsajos (neskaitant žiniatinklio aplikacijose naudojamo vizualizavimo) grafikams manipuliuoti realiu laiku. Ir reikia naudoti išorines/vidines sąsajas perduoti duomenis kitoms programoms.

R sistema turi repozitorijas, kuriose yra saugomas paketas ir informacija apie paketą. Gaila, tačiau susidūrėme su problema, kai negalėjome parsisiųsti tinkamos paketo versijos dėl naujausios R sistemos nesuderinamumo su senesniais paketais. Gali nutikti tokių situacijų, kad po sistemos atnaujinimo senesnius paketus reikės derinti prie naujausios R aplinkos dėl ko atsiranda rizika gauti klaidingus duomenis atlikus analizę.

Šio darbo autoriaus nuomone, R sistemą šaunu naudotis, kai reikia greitai įgyvendinti kokį nors algoritmą ir kai reikia kurti kokį nors funkcionalumą nuo pradžios iki pabaigos asmeniniam naudojimui, tačiau rimtiems tyrimams, jei leidžia biudžetas, geriau rinktis tą programinę įrangą ir taikomąsias programas, kurios turi autoritetą vienokioje ar kitokioje tyrimo srityje, nes tokiu atveju tampa aiškiau kur kreiptis iškilus problemoms, kitiems nesklandumams.

7. Praktinėje darbo dalyje naudoti įrankiai ir terpės

Šiame skyriuje pateikiame papildomų įrankių, kuriuos naudojome molekulinės dinamikos algoritmų integracijai atlikti, apžvalgas, panaudojimo galimybes ir nepatogumus su kuriais teko susidurti, nepatogumų sprendimų būdus.

7.1. Integruotos programų kūrimo aplinkos

Kuriant programinę įrangą ar taikomąsias programas dažnu atveju siekiame pasirinkti tokį programų kūrimo įrankį ar įrankius kurie padėtų greitai ir efektyviai naršyti po įvairias kalbos struktūras ar jau egzistuojantį kodą, taip pat siekiame, kad parinktas įrankis gebėtų tikrinti rašomo kodo eilutes nuo klaidų ir kad padėtų užtikrinti kodo korektiškumą bei estetinį vaizdą (kodo eilučių formatavimas), sumažinti kompiliavimo ir programų veikimo metu gautų klaidų ir kitokių netikslumų skaičių. Tam, kad galėtume programų kūrimo procese naudotis anksčiau išvardintais patogumais paprastai yra naudojamos integruotos programinės aplinkos.

C/C++ kodui rašyti naudojamos integruotos programų kūrimo aplinkos yra Visual Studio [Mic16c] (skirta tik Windows operacinei sistemai), NetBeans su C/C++ palaikymu [Cor16b], Code::Blocks [cod16], Eclipse su C/C++ palaikymu [Fou16a], Dev-C++ [Mes16]. Nors šios aplinkos ir padeda organizuoti C/C++ kodą, suteikia kitus privalumus, jos nėra niekaip susijusios su R aplinką ir dėl to sukompiliavus C/C++ kodą reikia atlikti kitus papildomus veiksmus [Fou16e] norint susieti R sistemą su C/C++ gautomis išvestimis. R projekto kūrėjai, be bendrų gairių kaip susieti išorines aplinkas su R, nėra sukūrę jokių integruotų programų kūrimo aplinkų, kuriomis pasinaudojant būtų įmanoma derinti C/C++ kodą su R aplinką.

Išnagrinėję esamų programavimo aplinkų teikiamus privalumus galime sakyti, kad C/C++ programavimui yra sukurta tikrai daug programavimo aplinkų, tačiau apžvelgiant R programavimo aplinkas, kurios padėtų kurti ir organizuoti R kodą galime teigti, kad yra galimybė naudoti papildomus įrankius ir nustatymus aprašytus aplinkų suderinamumo gairėse arba naudoti Rstudio – R kodo, paketų ir funkcijų kūrimo aplinką [Rst16a]. Tiek vienu tiek kitu atveju galime įžvelgti daugybę privalumų ir trukumų – naudoti Rstudio galima tik licenzijos apibrėžtose ribose, pasitaikydavo atvejų, kai ši integruota programų kūrimo aplinka tiesiog užstrigdavo ir tekdavo viską pradėti iš naujo. Ruošiantis kodo integravimui naudojant bendras gaires galime gauti lankstumo, pavyzdžiui, galima pasirinkti programų kūrimo aplinką/aplinkas ir netgi R sistemos versiją[9]. Tačiau, kaip jau minėta anksčiau – programų kūrimo aplinkų – į kurias būtų galima integruoti R – tiesiog nėra.

[9] minimali versija reikalinga Rstudio veikimui yra = 2.11.1 [Rst16b]

Pasvėrę privalumus ir trūkumus, C/C++ kodo integracijai į R sistemą atlikti pasirinkome naudoti Rstudio įmonės sukurtą integruotą programų kūrimo aplinką tuo pačiu pavadinimu (Rstudio). Rstudio leidžia paruošti projektą pagal atitinkamus šablonus – suteikia įprastinius projekto ir programavimo aplinkos nustatymus. Deja, tačiau C/C++ kodo ir R aplinkos integravimui vien Rstudio aplinkos neužtenka ir reikia dar kai kurių papildomų išorinių paketų ir įrankių.

7.1.1. Rstudio, išoriniai paketai ir įrankiai

Rtools[*].exe[10] – įrankių rinkinys [Fou16c] skirtas kompiliuoti ir testuoti pačiai R sistemai, R sistemos paketams Windows operacinėse sistemose. Rtools įtraukia kai kurias[11] pagalbines programas iš Cygwin aplinkos [RH16]; taip pat savyje laiko MinGW_32 ir MinGW_64 programų kūrimo aplinkas [Min16], bei gcc 32/64 bitų platformos kompiliatorius (g++, gcc, gfortan); kitus įrankius skirtus kodo kokybei ir našumui įvertinti. Skaitant Rtools dokumentaciją aiškėja, kad įrankių, ateinančių kartu su Rtools, nepakanka norint pilnai sukompiliuoti R sistema ir papildomai reikia sudiegti Perl programavimo aplinką [Per16], ir Latex [Lp16] sistemą jei norime sugeneruoti R paketų naudojimo instrukcijas (manuals) ir vinjetes(vignettes). Svarbu paminėti, kad Rtools turi būti operacinėje sistemoje nepriklausomai nuo to ar naudojame programų kūrimo aplinkas tokias kaip Rstudio ar ne; Linux ir kitose *nix sistemose turi būti naudojamas/naudojami Rtools ekvivalentai.

devtools – R sistemos paketas [WCRt16a] pasiekiamas per CRAN ir GitHub sistemas (naujausia versija su pataisymais ir naujomis funkcijomis visada pasiekiama per Github. Naudodami senesnę versiją, susidūrėme su nesuderinamumo problema – versija esanti CRAN sistemoje nėra itin dažnai atnaujinama ir gali būti nesuderinama su naujausia Rstudio versija). Šis paketas nėra sudėtinis bazinės R sistemos funkcionalumo dalis, taip pat nėra privalomas jei R paketų kūrimas yra nesusijęs su Rstudio – integruota programų kūrimo aplinka.

Devtools suteikia papildomas funkcijas, kuriomis pasinaudojus galima automatizuoti R kodo sudėjimą į paketą, C/C++ kodo kompiliavimą ir perkompiliavimą (pavyzdžiui, galima nustatyti automatinį senų *.o failų ištrynimą), padeda parsiųsti ir įdiegti paketus iš GitHub, Bitbucket kodo versijavimo sistemų. Suteikia galimybę nustatyti papildomas konfigūracijas C/C++ kodo kompiliavimui.

Naudodami Rstudio turėjome devtools sudiegti atskirai, priešingu atveju ne visos funkcijos, esančios Rstudio programų kūrimo aplinkoje, būtų veikusios korektiškai ir būtų reikėję kodo kompiliavimą, senų failų ištrynimą, dll modulių ir R paketų sukūrimą tekę atlikinėti per komandinę eilutę.

[10] * = naudojamos R aplinkos versija

[11] Visos funkcijos gali būti rastos Rtools instaliacijos kataloge, pavyzdžiui, C:\RBuildTools\3.3\bin

Po suinstaliuoto įrankių rinkinio – Rtools, integruotos programų kūrimo aplinkos – Rstudio ir paketo – devtools darbinė aplinka jau buvo paruošta darbui ir galėjome pradėti algoritmų integraciją, tačiau tam, kad integracijos procese galėtume taikyti gerąsias praktikas – testų rašymą, kodo dokumentavimą, komentavimą ir formatavimą turėjome sudiegti dar keletą papildomų paketų.

roxygen2 – R sistemos paketas [WCRt16b] kurio pagalba galima generuoti dokumentaciją panašiai kaip C/C++ kodui, naudojant Doxygen įrankį [vH16], arba komentarus naudojant Javadoc [Cor16a] – java kalba parašytoms programoms. Projekte naudojama programų kūrimo aplinka naudojasi roxygen2 paketo teikiamu funkcionalumu. Apmaudu, tačiau roxygen2 paketas turi priklausomybių nuo kitų paketų[12], kuriuos taip pat reikėjo sudiegti, deja kai kurių priklausomų paketų nepavyko suderinti su projekte naudojama R sistemos versija ir todėl kai kuriuos priklausomus paketus teko diegti rankiniu būdu. Roxygen2 paketas gali padėti sugeneruoti *.Rd failus, kuriuos atpažįsta R sistema ir pateikia žmogui suprantamu formatu (HTML formatu), tačiau norint turėti dokumentaciją *.pdf faile reikia dar vienos priklausomybės projekte – LaTex.

Latex – dokumentų paruošimo sistema [Lp16], dažniausiai naudojama alternatyva tokioms sistemoms kaip Microsoft Office ar SoftMaker Office, Open Office, Libre office, kitoms. Latex sistema yra nemokama, o paruošti dokumentai savo išvaizda nieko nenusileidžia aukščiau minėtų dokumentų sistemų ruošiniams. Windows operacinėse sistemose dažniausiai naudojama latex sistema yra MikTex. Sistemos apimtis svyruoja iki 200MB ir priklausomai nuo papildinių dar gali plėstis. Projekte naudosime įprastinė MikTex sistemos instaliaciją [Sch16].

R Markdown – kartais vien funkcijų aprašymo neužtenka ir norisi pateikti daugiau informacijos apie parašytą paketą, aprašyti taikymo atvejus ir/ar sudėtingesnius metodų panaudojimo scenarijus, aprašyti metodiką. Roxygen2 paketas koncentruojasi ties R sistemos dokumentacijos failų (*.Rd) generavimu. R Markdown paketas [All16] padeda kurti vinjetes [Wic16e], kuriose galima išdėstyti panaudojimo atvejus, pateikti itin detalius aprašus. Šio darbo projektui atlikti apsiribojome tik roxygen2 paketo teikiamomis galimybėmis.

testthat – paketas [WR16] suteikiantis galimybę kurti testus R sistemos funkcijoms ir paketams. Tam, kad galėtume naudoti Rstudio – integruotoje programų kūrimo aplinkoje reikia parsiųsti iš CRAN paketų sistemos ir sudiegti.

lintr – R paketas [Hes16] leidžiantis patikrinti parašytą R kodą. Nurodo netikslumų vietas, padeda numatyti tas kodo vietas, kurios gali sukelti nepatogumų.

formatR – R paketas leidžiantis formatuoti R kalba parašytą kodą [Xie16].

[12] ‘magrittr’, ‘stringr’, ‘stringi’, ‘brew’, ‘digest’

8. Molekulinės dinamikos algoritmai. Integracija

Šis skyrius skirtas apžvelgti molekulinės dinamikos algoritmus, kuriuos pasirinkome integracijai atlikti. Pateiksime kriterijus, kuriais remiantis atlikome atranką, taip pat aprašysime tikslus, kuriuos siekėme pasiekti, bei atsiradusias problemas, su kuriomis susidūrėme.

8.1. Algoritmų atrinkimo kriterijai

Algoritmų implementacijos turi būti aprašytos mokslinėje literatūroje – yra daugybė molekulinės dinamikos algoritmų implementacijų[Rap04][FDSB96]. Vienuose jų naudojamos papildomos sąlygos ir kriterijai. Kiti remiasi įvairiomis aproksimacijomis. Naudojant algoritmų implementacijas iš mokslinės literatūros, tikėtina, kad implementacijose esantys ribojimai, aproksimacijos ir taisyklės, kuriomis jie yra pagrįsti turi mokslinį pagrindą ir yra naudingi tyrimams atlikti.

Aprašytos algoritmų implementacijos turi būti susietos su teorija – šaunu turėti algoritmą, kuris sprendžia iškeltą uždavinį ar uždavinius, tačiau algoritmo vertę gali sumenkinti prastai aprašyta algoritmo taikymo sritis ar per ne lyg abstrakti uždavinio formuluotė. Todėl vienas iš svarbių kriterijų renkantis algoritmą – teorija (aprašyta dalykinė sritis, uždavinio formuluotės, algoritmo/(-ų) gautų rezultatų aiškinimas.)

8.2. Atrinkti algoritmai

Algoritmų implementacijas integracijai atlikti išsirinkome iš Rapaport knygos "The Art of Molecular Dynamics Simulation"[Rap04]. Detalesnė informacija pateikiama žemiau esančioje lentelėje (žr. lent. nr. 1).

Nuoroda į literatūrą	Darbe atlikta implementacija	Nuoroda į literatūrą	Darbe atlikta implementacija
pr_02_1	maino	pr_08_5	maino28
pr_03_1	maino3	pr_09_2	maino30
pr_03_2	maino4	pr_06_3	maino19
pr_03_3	maino5	pr_12_3	maino39
pr_03_4	maino6	pr_13_2	maino41
pr_04_1	maino8	pr_15_2	maino46
pr_04_2	maino9	pr_16_1	maino47
pr_04_4	maino11	pr_16_2	maino48
pr_04_5	maino12		
pr_06_1	maino17		
pr_06_2	maino18		

1 lentelė.: Antrinkti algoritmai su nuorodomis į Rapaport knygą ir kodą. [Rap04, p. 515, p. 516, p. 517]

8.3. Algoritmų integracijos tikslai

Esminis tikslas – ištirti R aplinkos galimybes sąveikauti su C/C++ kodu per R sistemai sukurtas C/C++ programines sąsajas. Tam, kad galėtume išskaidyti tyrimą į konkretesnes dalis suformulavome keletą užduočių.

Modifikuoti algoritmus (žr. lentelė 1.) taip, kad jie tiktų R aplinkai – parinkti algoritmai gali turėti trūkumų ir gali būti nesuderinami su R aplinką. Atlikdami šią užduotį, ištirsime R aplinkos teikiamų C/C++ programinių sąsajų galimybes vykdyti išorinį C/C++ kodą, trūkumus, nesuderinamumo priežastis. Taip pat, turėsime galimybę apžvelgti C/C++ sąsajų integravimui teikiamas duomenų struktūras, tų struktūrų ribojimus, bei integracijos ypatybes.

Papildyti algoritmus funkcijomis C/C++ kodo atžvilgiu – turimus algoritmus, tam, kad galėtume išgauti norimą ar jau esamą funkcionalumą, gali tekti keisti – pridėti naujas funkcijas, pavyzdžiui, duomenų nuskaitymui, ar abipusiam duomenų apsikeitimui tarp C/C++ ir R aplinkų, palaikyti. Atlikdami šią užduotį turėsime galimybę panagrinėti naujų C/C++ funkcijų pridėjimo įtaką algoritmų integracijai į R aplinką.

Papildyti algoritmus funkcijomis R kalbos atžvilgiu – algoritmų integracijos eigoje gali prireikti papildomų R funkcijų. Šios užduoties tikslas išsiaiškinti būtinas ir papildomas R funkcijas, kurios reikalingos integracijai atlikti.

Panaudoti kai kurių R paketų funkcionalumą C/C++ algoritmams – be galimybės kurti papildomas funkcijas, dar galima pasinaudoti jau sukurtu funkcionalumu, pasiekiamu per įvairias funkcinių paketų sistemas. Šioje užduotyje išplėsime C/C++ turimų algoritmų galimybes pasinaudodami jau egzistuojančiu paketų pagalba. Panaudosime kai kuriuos R sistemos paketus aprašytus literatūros dalyje. Užduoties tikslas – išsiaiškinti jau turimų C/C++ algoritmų suderinamumo ypatybes su R sistemoje naudojamais funkciniais paketais.

Parašyti integruojamiems algoritmams testus – testus turime parašyti tiek R sistemos funkcijoms, tiek C/C++ algoritmams. Tikime, kad testai yra itin svarbi integracijos dalis, ir todėl testų rašymui/ kūrimui išskyrėme atskirą užduotį, kurios tikslas apžvelgti testų sudarymo ir vykdymo galimybėms integracijos eigoje.

Panaudoti gerąsias praktikas – apima kodo organizavimą, vykdymą ir iškvietimą. Atlikdami šią užduotį, išsiaiškinsime ir aprašysime tas praktikas kurios yra taikomos integracijos procese, kuriant, keičiant R ir C/C++ kodą.

Panaudoti dokumentavimo galimybes – konspektuoti kodo galimybes, funkcionalumą, aprašyti panaudojimo atvejus yra laikoma gera praktika. Atlikdami užduotį aprašysime ir ištirsime dokumentavimo galimybes iš R aplinkos pusės, panaudojant C/C++ ir R kodą.

Integraciją atlikti panaudojant papildomus įrankius (žr. 6. papildomi įrankiai) – užduoties tikslas – apžvelgti esamus įrankius, kuriais pasinaudoję galėtume palengvinti, pagreitinti ar kitais būdais įtakoti algoritmų integracijos eigą į R aplinką.

Paruošti diegimo paketus – tam, kad galėtume naudotis sukurtu funkcionalumu, tą funkcionalumą reikia kokiu nors formatu pasiūlyti naudotojams. Paprastai Windows operacinėse sistemose funkcionalumas siūlomas *.exe failų formatu, *nix sistemose kitais formatais, taip pat yra keletas formatų, pavyzdžiui, *.java, kurie nėra priklausomi nuo operacinės sistemos. Šios užduoties tikslas išsiaiškinti būdus, kuriais sukurtą funkcionalumą, molekulinės dinamikos algoritmų integracijos į R aplinką procese, galima būtų pasiūlyti naudotojams. Tai paskutinė užduotis.

9. C/C++ algoritmų pritaikymas R aplinkai

Šiame skyriuje pateikiame išvadas apie C/C++ algoritmų pritaikymo galimybes R aplinkai. Apžvelgiame reikiamus pakeitimus algoritmams.

R aplinka turi septynias C/C++ programines sąsajas (žr. lentelę nr. 2), tačiau pastebėjome, kad išorinių C/C++ algoritmų kvietimui dažniausiai naudojamos yra .C, .Call ir .Rcpp. Integracijai naudojant .C sąsają reikia papildomai paruošti programų kūrimo aplinką, nes nėra sukurta integruotų programų kūrimo įrankių, kurios gebėtų naudoti .C sąsają automatizuotam C/C++ programų kvietimui. Taip pat privalu valdyti paduotų parametrų gyvavimo ciklą – apsaugoti paduotus parametrus ir gautus rezultatus nuo ištrynimo, atlaisvinti atmintį. Pakeitus C/C++ kodą reikia perkompiliuoti su Rtools[*].exe įrankiu.

.Call sąsaja veikia panašiai kaip ir .C – būtina apsaugoti reikšmes nuo ištrynimo, atlaisvinti atmintį, tačiau tipų tinkamumas gali būti tikrinamas C/C++ kode, yra galimybė naudoti sudėtingesnes duomenų struktūras. C/C++ kodo pakeitimus reikia perkompiliuoti su Rtools[*].exe

.Rcpp programinė sąsaja, C/C++ kodo kvietimui naudojasi .Call sąsajos funkcionalumu, o visi veiksmai susiję su parametrų ir reikšmių gyvavimo ciklu yra valdomi automatiškai. C/C++ kodo pakeitimus perkompiliuoti galima pasinaudojus Rstudio – programų kūrimo aplinkos teikiamu funkcionalumu.

Atliekant C/C++ algoritmų integraciją į R aplinką naudojome .Rcpp programinę sąsają kartu su R studio teikiamais įrankiais. Privalumai – R studio apjungia visus įrankius, reikalingus integracijai atlikti. Pakeitimų perkompiliavimą galima automatizuoti. Testų paleidimą galima atlikti iš Rstudio aplinkos.

Nr.	**C/C++ programinė sąsaja**	**Aprašymas**
1.	Rcpp	žr. 2.1 C/C++ programinės sąsajos. *.Rcpp*
2.	.Call	žr. 2.1 C/C++ programinės sasajos. *.Call*
3.	.C	žr. 2.1 C/C++ programinės sasajos. *.C*
4.	.External2	žr. 2.1 C/C++ programinės sasajos. *.External2*
5.	.External	žr. 2.1 C/C++ programinės sasajos. *.External*
6.	.Primitive	žr. 2.1 C/C++ programinės sasajos. *.Primitive*
7.	.Inside	žr. 2.1 C/C++ programinės sasajos. *.Inside*

2 lentelė.: Programinės C/C++ sąsajos naudojamos R aplinkoje

C/C++ Programinės sąsajos nėra tapačios, skiriasi savo panaudojimo galimybėmis – paprasčiausias būdas iškviesti C/C++ kalba parašytą kodą – yra išorinių ar vidinių R kalbos paketų arba fundamentalių R sistemos funkcijų panaudojimas. Taip yra todėl, nes dauguma funkcijų, esančių R sistemoje yra parašytos C/C++ kalbomis, o R kalba parašytas kodas veikia kaip tarpininkas tarp C/C++ kodo ir R aplinkos. Sudėtingesniais atvejais, kai pavyzdžiui, norime iškviesti savo kodą, parašytą C/C++ kalba turime naudoti išorines programines sąsajas, kurias pagal panaudojimo prasmę, galima suskirstyti į dvi grupes (žr. lentelė nr. 3).

Nr.	C/C++ programinė sąsaja	Išorinė	Vidinė
1.	Rcpp	✓	
2.	.Call	✓	
3.	.C	✓	
4.	.External2	✓	
5.	.External	✓	
6.	.Primitive		✓
7.	.Inside		✓

3 lentelė.: Vidinės ir išorinės programinės sąsajos

Vidinės C/C++ programinės sąsajos yra naudojamos pačios R sistemos vidinių struktūrų apdorojimui ir bazinių funkcijų kvietimui atlikti. Išorinės sąsajos padeda funkciniams paketams ir kitos pridėtinės vertės objektams sąveikauti su R aplinka (padeda suderinti įvestis/ išvestis, iškviesti R sistemoje esantį funkcionalumą). Išorinės sąsajos skirtos paketų kūrėjams ir asmenims dirbantiems R aplinkoje, kur vidinės sąsajos išimtinai skirtos tik R sistemos vidiniams procesams ir funkcijoms naudotis.

.C ir .Call, .External ir .External2 programinės sąsajos, lyginant su .Rcpp, yra senesnės, todėl C/C++ algoritmų kvietimas įtraukia daugiau žemo lygio elementų valdymo rutinų, o žemo lygio rutinų naudojimas gali suteikti atminties valdymo privalumų, tačiau sukelia nepatogumų kuriant programos logiką – nes programinės sąsajos struktūros persipina su programa, kurią norime priderinti prie pačios sąsajos. Manipuliacijos Žemo lygio struktūrų kvietimais, naudojant senesnes programines sąsajas[13] kodą padaro mažiau atsparų klaidoms, gali trukdyti programos ar algoritmo veikimą.

Naudodami .C sąsają galime iškviesti C/C++ kalba parašytą kodą be jokių papildomų pakeitimų, tačiau toks kvietimas neperduotų algoritmo išvesčių į R aplinką ir negalėtume pateikti jokių įvesčių iš R aplinkos. .Call programinė sąsaja, skirtingai nei .C leidžia tikrinti perduotus parametrus iš C/C++ kodo pusės. .External ir .External2 veikia taip pat kaip ir .Call tiesiog galima perduoti daugiau nei 65 parametrus. .Rcpp veikia kaip programinė sąsaja .Call sąsajai.

[13].C, .Call, .External, .External2

Svarbiausia integracijos dalis – susieti C/C++ kodo išvestis su R aplinkos įvestimis – tam, kad galėtume atlikti C/C++ algoritmų integraciją turėjome panaudoti C/C++ algoritmus, R sistemą, turėjome sukurti papildomas funkcijas R aplinkai ir pakoreguoti jau turimas C/C++ programas, kad jos galėtų sąveikauti su R aplinką, panaudoti R sistemos ir R sistemai siūlomas programines sąsajas. Atlikdami integraciją pastebėjome, kad esminis elementas – C/C++ algoritmų išvesčių su R aplinka ir R aplinkos įvesčių susiejimas su C/C++ algoritmais. Po tokio susiejimo belieka pasirašyti keletą šalutinių R funkcijų ir algoritmų integracija iš esmės baigta. Skirtingos programinės sąsajos siūlo skirtingas įvesčių/išvesčių susiejimo formas.

.C programinė sąsaja – kiekviena funkcija kviečiama .C programinės sąsajos turi būtinai gražinti void tipo reikšmę. Visi paduodami parametrai c funkcijai turi būti rodyklės tipo. C funkcijos kvietimas grąžina sąrašo tipo struktūrą, kurios elementai yra c funkcijai paduoti parametrai po c funkcijos iškvietimo.

.Call, .External, .External2, .Rcpp programinės sąsajos – C/C++ funkcijose parametrai, kurie yra paduodami iš R aplinkos, turi būtinai būti SEXP[14] tipo. C/C++ funkcijos gali gražinti void arba SEXP tipo reikšmes [Fou16d], kurios gali būti vektoriai, sąrašai ir elementarūs tipai tokie kaip boolean, string, integer, real.

.External, .External2 programinės sąsajos – veikia panašai kaip .Call, skiriasi tik parametrų padavimu c funkcijoms iš R aplinkos – .External programinės sąsajos neturi paduodamų parametrų skaičiaus ribojimo, o c funkcijos turi būti kuriamos taip, kad funkcijos antraštėje būtų tik vienas parametras – SEXP tipo elementas.

Po rezultatų, parametrų įvesčių ir išvesčių suderinimo apžvalgos galime teigti, kad paprasčiau yra naudoti .Rcpp programinę sąsaja, nes ji užslepia žemesnio lygio atminties valdymo rutinas (kintamųjų apsaugai nuo ištrynimo), parametrų perdavimui yra naudojamos SEXP tipo struktūros, kaip ir .Call .External ir .External2 sąsajų atvejais, o kadangi SEXP tipo struktūros apima sąrašo, vektorių tipo struktūras, naudodami .Rcpp sąsają, parametrus, perduotus C/C++ funkcijai, galime nuskaityti iš sąrašo ar vektoriaus tipo struktūros, tokiu būdu išvengdami .Call ir .C programinės sąsajos 65 parametrų limito. Taip pat norime atkreipti dėmesį, kad nuo pasirinktos programinės sąsajos priklauso C/C++ modifikacijos, kurios yra privalomos pagal nagrinėtų programinių sąsajų panaudojimo aprašymus.

Po išvesčių ir įvesčių susiejimo lieka apžvelgti dar keletą integracijos aspektų – tipų suderinamumo galimybes iš C/C++ kodo ir R aplinkos klaidų valdymo atžvilgiu. Paketų / išorinių modulių parengimo atvejus. Dokumentacijos rašymo funkcionalumą ir testavimo scenarijų rengimą integracijos procese.

[14] SEXP = S expression

10. Parinktų algoritmų derinimas prie R aplinkos

Šioje darbo dalyje apžvelgiame pakeitimus, kuriuos turėjome atlikti atrinktiems C/C++ algoritmams. Pateikiame išvadas, kurias priėjome darydami pakeitimus. Aptariame pakeitimų įtaką algoritmų veikimui.

10.1. Pakeitimai susiję su C/C++ algoritmais

Turimų algoritmų pakeitimams atlikti buvo galima pritaikyti kelias strategijas – pakeitimus (pridėti naujus kintamuosius, funkcijas, parametrus) daryti tik C/C++ algoritmuose ir per programines sąsajas kviesti jau sukompiliuotas programas. Kitu atveju papildomus priedus, jau turimiems algoritmams, įgyvendinti galima per R aplinką – kuriant papildomas R funkcijas. Trečiu atveju galima derinti pirmąją strategiją su antrąją, tačiau visais atvejais norėdami iškviesti C/C++ algoritmus taip, kad išvestys ar įvestys būtų matomos R aplinkai turime daryti pakeitimus C/C++ kodui. O, pasirinkdami trečiąją algoritmų integravimo į R sistemą strategiją galime minimizuoti reikalingų pakeitimų kiekį – papildymus įgyvendinant per R aplinką. Siekdami išsaugoti integruojamų algoritmų vientisumą ir loginę seką Šiame darbe algoritmų integracijai atlikti pasirinkome taikyti trečiąjį būdą. Ir integracijai atrinktus algoritmus turėsime pakeisti keliais atžvilgiais.

Visi atrinkti algoritmai turi kelias funkcijas kurios po integracijos į R aplinką tampa nebereikalingos – tai parametrų iš failo nuskaitymo funkcijos – jas pakeičia pagrindinės funkcijos (main) praplėtimas naujais parametrais, kur parametrų formatas priklauso nuo pasirinktos R programinės sąsajos. Kiekvieno atrinkto algoritmo pagrindinės funkcijos (main) pavadinimas pakeičiamas algoritmui unikaliu pavadinimu, kurį naudosime kvietimui iš R aplinkos. Pagrindinės funkcijos (main) gražinamas objektas iš void tipo pakeičiamas į SEXP objektą, kuris savyje turi visus paduotus parametrus, rezultatą, gautą po iteracijos, kitus algoritmui būdingus požymius. Šalutines funkcijas (kurios skaičiuoja rezultatus, kintamųjų reikšmes) perkeliame į unikalias skirtingas vardų sritis, o algoritmui lokalūs kintamieji yra perkeliami į anoniminę vardų sritį. Suteikdami šalutinėms funkcijoms unikalias vardų sritis ir lokalius kintamuosius perkėlę į anoniminę vardų sritį laimime keliais atžvilgiais – minimizuojame pakeitimų kiekį, reikalingą algoritmų veikimui R aplinkoje, Išlaikome originalią veiksmų seką, kuria remtasi įgyvendinant algoritmą. Algoritmo pakeitimus galima paprasčiau atversti, taip kad vėl juos būtų galima naudoti be R aplinkos. Ir svarbiausia – lengviau lyginti originalias kodo išvestis su pakeistomis.

Visus aukščiau aprašytus pakeitimus įgyvendinome panaudodami .Rcpp programinę sąsają. Sekančiu etapu panagrinėsime funkcionalumą ir pakeitimus, kurie padės išnaudoti C/C++ algoritmų galimybes R aplinkoje.

10.2. Papildomos R funkcijos C/C++ algoritmams naudoti

R funkcijomis galima reguliuoti kviečiamų C/C++ algoritmų veikimą – po C/C++ algoritmų modifikacijų lieka sukurti papildomas funkcijas R aplinkai, kurios kviestų C/C++ funkcionalumą. Išsiaiškinome, kad kurti tokias funkcijas galima pasinaudojant R aplinkai pritaikytų C/C+++ sąsajų teikiamomis galimybėmis, tačiau naudoti vien programines sąsajas, kvietimams atlikti, gali būti netikslinga – nes jos tik iškviečia funkcijas ir perduoda papildomus parametrus, tačiau netikrina ar perduoti parametrai yra tinkamo tipo, neturi klaidų apdorojimo mechanizmo. Netikslumų atsiradimų į kuriuos gali įeiti įvairūs tipų nesuderinamumai, paduotų parametrų trūkumai/perteklius galima kontroliuoti dvejais būdais – iš R aplinkos arba pačiuose C/C++ algoritmuose.

Atlikdami tyrimą išsiaiškinome, kad naudojant .C programinę sąsają, tipų suderinamumo problemas ir kitus atsirandančius netikslumus turėtume spręsti R aplinkoje, prieš paduodant duomenis programinei sąsajai (.C). Kitos nagrinėtos programinės sąsajos tokio ribojimo neturi ir tipų derinimą įmanoma atlikti tiek iš R aplinkos tiek C/C++ algoritme.

C/C++ algoritmo kvietimas turi gražinti R objektą arba null tipo reikšmę – visose nagrinėtose programinėse sąsajose pastebėjome bendrą bruožą – kviečiamos funkcijos privalo gražinti R objektą, C/C++ rodyklės tipą arba null reikšmę žymintį objektą. Visais atvejais, gražinus ne null tipo objektą ar reikšmę, objektai ir gražintos reikšmės gali būti pernaudotos tolesniems veiksmams atlikinėti R aplinkoje, tačiau iškvietus C/C++ funkciją, kuri gražina null tipo reikšmę – skaičiavimo rezultatų funkcija negrąžina, nors jei kviestoji funkcija ką nors spausdindavo į standartinę išvestį, tai toji išvestis spausdinama atidarytoje R konsolėje.

Papildomų funkcijų kūrimas iš R aplinkos leidžia daryti mažiau pakeitimų C/C++ kode – kuriant R funkcijas C/C++ algoritmams reikia atsižvelgti į tai, kokia R programinę sąsają naudojame kvietimams atlikti ir į tai ar tipų suderinamumas gali būti atliekamas R aplinkos ar kviečiamų C/C++ algoritmų pusėje. Taip pat turime numatyti sukurtų C/C++ algoritmų panaudojimo atvejus – parametrų derinius.

Darbe kurtas R funkcijas galime suskirstyti į dvi grupes –tai papildomos programinės sąsajos ir duomenų nuskaitymo (papildomo apdorojimo) funkcijos. Papildomomis programinėmis sąsajomis galime varijuoti funkcionalumą, siūlomą C/C++ algoritmų taip pat galime apsisaugoti nuo tipų nesuderinamumo problemos, kitų, nenumatytų, problemų.

Papildomomis funkcijomis galime paruošti duomenis apdorojimui, sukurti duomenų aibes, piešti duomenų aibių statistikas,atlikti kitas vizualizacijas. Algoritmų integracijos procese pastebėjome, kad tikrinti paduodamus duomenis C/C++ algoritmams R aplinkos pusėje yra kur kas paprasčiau nei tikrinti C/C++ kode.

11. Testų rašymas ir sudarymas

R sistema turi funkcionalumą rašyti testus C/C++ funkcijoms – testus turėjome rašyti tiek C/C++ algoritmams tiek R funkcijoms ir visai integracijai apskritai. Nors testų rašymas ir nėra privalomas. Nagrinėdami R sistemos galimybes radome, kad paketas testthat (1.0 versija) kartu su paketu devtools suteikia galimybe testus rašyti ir C/C++ algoritmams. Tačiau galiausiai nusprendėme testus C/C++ kurtiems algoritmams nerašyti, o testuoti juos netiesiogiai – per R aplinką, rašydami testus tik R kodui.

R funkcijoms testuoti galima naudoti RUnit, svUnit arba testthat paketų siūlomas galimybes – tiek RUnit [BJK16], svUnit [Gro16] ir testthat suteikia kuriamo R sistemos funkcionalumo testavimo galimybes kiekvienai R funkcijai rašant modulių testavimo atvejus. Šio darbo praktinės dalies testų kūrimui ir rašymui pasirinkome naudoti testthat funkcijų paketą. Naudodamiesi testthat paketu taip pat galėjome ištestuoti ir C/C++ algoritmų korektiškumą, netiesiogiai testuodami programinių sąsajų kvietimus su įvairiais parametrais. Naudodamiesi testthat teikiamomis galimybėmis galėjome įjungti automatinį testavimą funkcijų rašymo metu.

Testų integracija į kuriamą produktą pavyko be nesklandumų – projekto vystymui ir kūrimui naudojome integruotą programų kūrimo aplinką – Rstudio, taip pat daugiausiai algoritmų pritaikėme .Rcpp programinei sąsajai. Naudodami Rstudio turėjome keletą privalumų – projektui vystyti palaikėme Rstudio programavimo aplinkos siūlomą įprastą aplankų struktūrą ir turėjome galimybę kiekvieną kartą leisti automatinius testus vieno mygtuko paspaudimu. Taip pat Rstudio turi testthat paketo funkcionalumo pilną palaikymą.

Šalutinės pastabos – R funkcijų testų sudarymui naudojome tik testthat siūlomą funkcionalumą, nes naudojamas programų kūrimo įrankis – Rstudio turi šio paketo palaikymą. RUnit ir svUnit galimybių netestavome. Testų sudarymui jokių specifinių kriterijų neturėjome, tiesiog rašėme testus taip, kad skaičiavimai ir išvestys, lyginant su literatūros šaltiniu, visada būtų tapatūs.

Testuodami C/C++ algoritmus taip pat neturėjome jokių kriterijų ir kaip R funkcijų testavimo atveju – išvestis lyginome su šaltinyje pateiktomis išvestimis. Kadangi testavimas vyko tik su testthat paketu, palyginti funkcionalumo ir teikiamų galimybių su skirtingais testavimo paketais negalime.

Testus parašytus parašytus R funkcijoms turime leisti atskirai – būdo kaip automatizuoti radome. Tačiau neradome patogaus įrankio kuris savo galimybių aibe būtų ekvivalentus maven, ant, gradle sistemoms. Testavimą galėjome atlikti visoms kuriamoms R funkcijoms.

12. Gerųjų praktikų naudojimas

R funkcijoms ir integruojamiems C/C++ algoritmams nėra griežtų stiliaus reikalavimų – kurdami R funkcijas ir atlikdami C/C++ algoritmų integraciją į R aplinką neaptikome jokių privalomų reikalavimų kurių turėtume laikytis rašydami funkcijas R aplinkai ir bendrai – integruodami C/C++ kodą. Visos pastabos apie kodo stilių ir funkcijų vykdymo formatus yra pateikiamos įvairiuose šaltiniuose rekomendacijų forma [AEF16][Wic16c].

Klaidų atsiradimo metu vykdomas kodas neturėtų trikdyti R aplinkos veikimo – integruojant C/C++ algoritmus itin dažnai susidurdavome su situacija, kai C/C++ algoritmų veikimo metu sustingdavo R aplinka, ko pasekoje tekdavo R aplinką paleidinėti iš naujo. Vengdami tolesnių panašaus tipo nutikimų pasirūpinome, kad įvykus klaidai būtu išmestas pranešimas, o algoritmo vykdymas sustabdytas. Nors ir atlikome visus reikalingus pakeitimus visiškai išvengti nemalonių R aplinkos sutrikimų nepavyko.

Algoritmas negali veikti be trumpalaikių sustojimų – integruodami C/C++ algoritmus pastebėjome keletą iškviestų funkcijų veikimo bruožų – iškviesti algoritmai veikia iki tol kol negražina rezultato arba neįvykdo kode esančių instrukcijų. Taip pat viso algoritmo veikimo metu R sistema tampa neaktyvi naudotojo įvestims, ko pasekoje algoritmo veikimo sustabdymas įmanomas tik iš naujo paleidus R sistemą. Tam, kad galėtume grąžinti R aplinkos kontrolę naudotojui reikia kas kelias kelias iteracijas pertraukti funkcijos ar C/C++ kodo veikimą. Atkreipiame dėmesį, kad toks priverstinis funkcijos ar C/C++ algoritmo stabdymas atsiliepia ir algoritmo/funkcijos įvykdymo laikui. Kito būdo kaip grąžinti aplinkos kontrolę naudotojui – neradome. Ši gera praktika itin suprastino C/C++ algoritmų veikimo laiką, ko pasekoje C/C++ kodo pranašumas (vykdyti greitai sudėtingas algoritmo dalis) praktiškai išnyko. O, molekulinės dinamikos algoritmų integraciją į R aplinką galime vadinti beprasme – nei našumo, nei efektyvumo nebeliko.

Kodo formatavimui nėra jokių apribojimų – kaip ir C/C++ kodui taip ir R kodui nėra jokių papildomų formatavimo apribojimų yra tik rekomendacijos (ateina kartu su formatR paketu). Tačiau viena iš gerų praktikų – aiškiai skaitomas kodas. Kadangi, įvairios programavimo kalbos turi skirtingas kodo formatavimo taisykles prieš paviešinant kodą galima panaudoti kodo formatavimo įrankius – r kodui formatuoti – formatR. C/C++ kodo formatavimui – AStyle [Dav16]. Kodą formatavimo galimybes siūlo ir Rstudio programų kūrimo įrankis.

Be kodo formatavimo, įrankių, kurie būtų kurti R kodo minimizavimui, neradome. Poreikio minimizuoti kodą taip pat neaptikome. R kalba rašytos funkcijos yra interpretuojamos R interpretatoriaus. Be kodo formatavimo, klaidų apdorojimo, algoritmų vykdymo pertraukimo, kitų vien R kalbai būdingų specifinių savybių nepastebėjome.

Dar viena gera praktika su kuria susidūrėme ruošdami R funkcijas ir C/C++ algoritmus – tai kodo dokumentavimas – praktika kuri R sistemoje yra itin labai plačiai naudojama, nors ir nėra privaloma.

R sistema turi dokumentavimo galimybes kurios prilygsta Java ar C++ programavimo aplinkoms – dažniausiai norėdami komentuoti Java kodą, tam naudojame kelių tipų komentarus, jei planuojame komentarus įtraukti į kuriamą dokumentaciją tam naudojame papildomus įrankius, pavyzdžiui, doxygen (įrankis C++ kalbos dokumentacijai genruoti), javadoc (naudojamas java programų dokumentacijos ruošimui).

R sistema, skirtingai nei Java, neturi kartu su sistema ateinančio įrankio dokumentacijos ruošimui, tačiau panaudojus išorinius paketus, kaip, kad, pavyzdžiui, roxygen2 galima kurti dokumentacijas įvairiais formatais, pavyzdžiui, *.html ar *.pdf.

Kai kurios R sistemos dokumentacijos ruošimo galimybės remiasi latex sistemos funkcionalumu – ruošdami dokumentaciją susidūrėme su situacija, kai norėdami sugeneruoti *.pdf failą turėjome sudiegti latex sistemą. Taip pat atkreipėme dėmesį į tai, kad dokumentacija sugeneruota .html formatu nežymiai skiriasi nuo *.pdf versijos.

R sistema turi du pagrindinius dokumentacijos panaudojimo scenarijus – naudodami roxygen2 paketo galimybes iš komentarų parašytų R funkcijų kode ir C/C++ algoritmų kode galime kurti dokumentaciją, kurioje būtų įtraukti funkcijų parametrai, parametrų paaiškinimai, kita esminė informacija.

Norėdami įtraukti sukurtų funkcijų panaudojimo atvejus, naudodami roxygen2 komentavimo galimybes turėtume rašyti pakankamai ilgus tekstus virš funkcijų (ką ir padarėme visuose algoritmuose), todėl platesniam kodo komentavimui galima naudoti kitas dokumentavimo galimybes, pavyzdžiui, tas kurios yra markdown pakete. Naudodami markdown paketą būtume galėję kai kuriuos atvejus susijusius su funkcijų panaudojimu aprašyti plačiau. Vienas trūkumas į kurį atkreipėme dėmesį – tai roxygen2 ir markdown paketo sintaksės skirtumai.

Šalutinės pastabos – daryti sustojimus po kelių vienos ar kitos iškviestos funkcijos iteracijų nėra pats geriausias sprendimas, nes tokiu atveju pailginame algoritmo veikimo laiką, deja, tačiau kito varianto grąžinti naudotojui R aplinkos funkcionalumo neradome.

R sistemoje taip pat yra funkcionalumas suteikiantis galimybę rašyti platesnius funkcijų panaudojimo atvejus (markdown) ir bendro pobūdžio komentarus (roxygen2). Esminis komentarų rašymo paketų privalumas – gebėjimas generuoti dokumentaciją įvairiais formatais, pavyzdžiui, *.pdf, *.html. Didelis trūkumas – skirtinga sintaksė, kuria paketai remiasi.

C/C++ algoritmų integravimo procese pastebėjome, kad nors ir įgyvendinus visas R aplinkos siūlomas kodo vykdymo gerąsias praktikas vis vien negalime būti visiškai tikri, kad įvykus kokiam

nors R sistemos sutrikimui (nors nuo daugumos sugebėjome apsisaugoti) sugebėsime apdoroti atsiradusius nepatogumus ir grįžti atgal į R aplinką, perduodami R aplinkos valdymą atgal R sistemos naudotojui.

13. Grafikos elementų integracija

Integruoti grafikos elementus į projektą galima keliais būdais – galima panaudoti bazinį funkcionalumą (žr. 3.1 Duomenų vaizdavimas grafikais/diagramomis) ir projekte sukūrus papildomas funkcijas kviesti bazines grafikos galimybes. Tą patį galima atlikti ir su papildomais grafikos paketais siūlomais R sistemai ar net išorinėmis programomis. Dar vienas metodas – sugeneruoti išvestis tokiu formatu, kad jas būtų galima atvaizduoti ne R sistemoje.

Kiekvienas iš anksčiau minėtų būdų turi didelių trūkumų ir ne daug privalumų – siedami savo projekto galimybes su baziniu R sistemos siūlomu grafikos funkcionalumu galime būti tikri, kad visose R sistemos versijose bus tos pačios grafikų vaizdavimo galimybės, tačiau tos galimybės nėra itin didelės ir apsiriboja vien tik įprastu grafikų braižymo funkcijomis.

Naudodami R aplinkos paketus (išplėstinį funkcionalumą) gauname lygiai tą patį ką ir naudodami įprastą funkcionalumą tik su gausesnę atvaizduojamų grafikų aibę ir geresnėmis atvaizdavimo galimybėmis. Minusas tas, kad prieš kviesdami funkcijas su išplėstiniu R funkcionalumu turime pasirūpinti, kad paketai reikalingi funkcijos veikimui, yra sudiegti, turi reikiamas versijas ir yra paruošti naudojimui.

Naudodamiesi išorinėmis programomis (žr. 3.2 Kitos duomenų vaizdavimo galimybės) galime mažiau laiko skirti funkcijų kūrimui, kurios kviestų grafines programų galimybes ir daugiau laiko skirti formato, kuriuo bus apsikeičiama tarp R aplinkos ir išorinės programos. Trūkumas naudojant išorines programas yra tas, kad jos, dažnu atveju, nėra R aplinkoje ir sugeneruoti vaizdiniai ir rezultatai turi būti perkelti atgal į R aplinką.

R sistemoje nėra funkcionalumo simuliacijų grafiniam vaizdavimui gyvai atlikti – ieškodami būdų kaip R aplinkoje atlikti duomenų vaizdavimą realiu laiku susidūrėme su keliais nepatogumais – funkcionalumo, kuris galėtų realiu laiku atvaizduoti duomenis, R sistemoje, neradome ir todėl turėjome įgyvendinti kelias funkcijas, kurios padavinėtų duomenis grafikų sistemai ir kas kart perpieštų grafiką. Taip pat neradome nagrinėjamai dalykinei sričiai (molekulinės dinamikos algoritmams) patogaus objektų atvaizdavimo galimybių ir todėl nagrinėjamos dalykinės srities objektų vaizdavimo teko atsisakyti – pagrindinė priežastis – C/C++ algoritmų veikimo privalomas pertraukimas.

Šalutinės pastabos – R sistema turi gausybę visokiausių paketų kuriais pasinaudojus yra įmanoma gauti įvairių tipų grafikus, kitus grafinei analizei būdingus elementus. Tačiau trūksta paketų kurie gebėtų atvaizduoti duomenis pagal skirtingas dalykines sritis, pavyzdžiui, nėra paketo ir funkcionalumo kuriuo galėtume atvaizduoti molekules, molekulinius junginius, todėl gali tekti improvizuoti arba supaprastinti vaizduojamos srities objektus arba nieko nevaizduoti.

14. Įrankiai integracijai atlikti

Rstudio programavimo aplinka palaiko visus šeštajame skyriuje aprašytus įrankius ir įrankių rinkinius – naudodami Rstudio turėjome kelis privalumus – visus papildomus įrankius (žr . 6.1.1 skyrių) galėjome paleidinėti iš vienos vietos ir turėjome galimybę minimizuoti R konsolės naudojimą[15], įrankius pasiekdami per Rstudio siūlomus funkcinių mygtukų rinkinius, arba naudodami meniu funkcionalumą.

R aplinka turi papildomus įrankius R kalbos stiliui ir formatavimui palaikyti – R kodo formatui palaikyti naudojome formatR ir R kodo korektiškumui tikrinti – lintr paketų teikiamas galimybes. Naudodami formatR galime užtikrinti, kad rašomas R kodas atitiks visas R kodo formatavimo rekomendacijas, o lintr naudojimas padėjo aptikti ir kai kuriais atvejais taisyti R kodo netikslumus. Tačiau naudoti šiuos paketus per R aplinkos komandinę eilutę nėra patogu. Rstudio programavimo aplinka leidžia automatizuoti formatR ir lintr teikiamą funkcionalumą.

Paketų, suteikiančių papildomą funkcionalumą, versija turi sutapti su naudojama R aplinkos versija – kurdami papildomas R funkcijas ir atlikdami molekulinės dinamikos algoritmų integraciją į R aplinką itin dažnai susidurdavome su nepatogumu kai paketai, kuriuos norime naudoti, yra nesuderinami versijomis su pasirinkta R aplinka. Toks versijų nesuderinamumas gali atsiliepti korektiškam paketų ir pačios R aplinkos veikimui.

Rstudio palaiko projekto versijavimo sistemas – vystant funkcionalumą, versijavimas padeda kurti projekto istoriją, saugoti pakeitimus taip pat padeda dalintis projektu ir pakeitimais su komanda. R sistema neturi papildomų galimybių integruoti rašomą kodą su versijavimo sistemomis, pavyzdžiui, git ar subversion.

Integraciją su versijavimo sistemomis galima atlikti keliais būdais – rankiniu būdu saugoti visus pakeitimus, pasirašyti keletą papildomų programų, kurios galėtų saugoti kodą kas tam tikrą laiko tarpą arba naudoti Rstudio kodo versijavimo galimybes. Pastebėjome, kad naudoti kodo versijavimo galimybes per komandinę eilutę (cmd, shell) yra kur kas paprasčiau nei per Rstudio.

Šalutinės pastabos – papildomi įrankiai, aprašyti šeštajame skyriuje padeda testuoti ir kurti R aplinkai skirtas funkcijas. Tačiau dauguma jų, kuriant bazinį funkcionalumą, nėra privalomi. Beveik visi papildomi įrankiai yra kurti ne R projekto kūrėjų, o išorinių programuotojų, siekiančių palengvinti programavimą R aplinkoje, todėl yra galimybė, kad iš minėtų įrankių, kai kurie bus nesuderinami su naujausiomis R sistemos versijomis. Rstudio – integruota programų kūrimo aplinka suteikia kodo formatavimo bei klaidų tikrinimo galimybes be papildomų paketų – formatR ar lintr todėl gali būti naudojama kaip alternatyva minėtiems paketams.

[15] reikia įrankių užkrovimui ir iškvietimui

15. Papildomų R aplinkos paketų panaudojimas

R paketus galima suskirstyti į dvi grupes – skirtus duomenų atvaizdavimui ir duomenų analizei – pirmajai grupei, duomenų vaizdavimui, skirti paketai tiek duomenų analizei skirti paketai ateina kartu su R sistema arba gali būti sudiegti parsisiuntus iš R repozitorijų, pavyzdžiui, CRAN arba gali būti sudiegti iš kitų trečių šaltinių.

Paprasčiau naudoti papildomus paketus per R aplinką nesusiejant paketų su projektu tiesiogiai – į vystomą projektą įdėti papildomus R aplinkai siūlomus paketus galėjome keliais būdais – padaryti keletą priklausomybių nuo reikalingų R paketų, be kurių projektas neveiktų. Aprašyti panaudojimo atvejus, pasinaudodami išplėstinėmis dokumentavimo galimybėmis. Arba sukurti minimalų funkcionalumą, kuris gebėtų pateikti išvestis į R aplinką, kurioje išvestis galėtume panaudoti kitiems skaičiavimams ar funkcijoms iš kitų paketų.

R paketų derinimas su R aplinka naudojant skirtingas R aplinkos ir R paketų versijas gali trikdyti kuriamo funkcionalumo veikimą – dažnai susidurdavome su nepatogumu, kai bandant įdiegti papildomus paketus į R aplinką, diegiami paketai būdavo nesuderinami, nes skirdavosi diegiamų paketų ir R aplinkos versijos. Toks versijų nesuderinamumas sukelia kelias problemas – destabilizuoja kviečiamas funkcijas (gali grąžinti nekorektiškus rezultatus). Apsunkina versijomis nesuderinamų paketų įdiegimą į R aplinką.

Išorinių paketų teikiamos galimybės gali vystytis nepriklausomai nuo kuriamo funkcionalumo – kurdami naujus paketus R aplinkai ir įtraukdami kitų išorinių paketų funkcijas negalime būti tikri, kad sekančiose pridėtinės vertės paketų versijose teikiamos galimybės bus suderinamos su sukurtu funkcionalumu. Taip pat naudojant išorinius paketus yra galimybė, kad rezultatai bus nekorektiški, nes dažnu atveju apie išorinio paketo funkcijų implementacijas yra žinoma nedaug. Taip pat nedaug žinoma apie autoriaus išsilavinimą, ketinimus ar profesionalumo lygį dalykinėje srityje.

Šalutinės pastabos – pastebėjome, kad daugiausiai lankstumo suteikianti papildomų R paketų integravimo į vystomą projektą strategija – yra neintegruoti paketus, o tiesiog pateikti kai kurių paketų panaudojimo atvejus aprašant juos pasinaudojus išplėstinėmis dokumentacijos galimybėmis. Naudodami šią strategiją laimime keliais atvejais – nepririšame savo projekto galimybių prie išorinio funkcionalumo. Pateikdami tik bazinį funkcionalumą (įvestys – išvestys) suteikiame kuriamam paketui daugiau pritaikymo ir panaudojimo atvejų, nors panaudojimo galimybes apsprendžia pats naudotojas.

16. Paketų paruošimas

R sistema gali būti dviejų tipų – 32-bit ir 64-bit – išoriniai C/C++ algoritmai prieš naudojimą turi būti sukompiliuoti dviejų tipų R platformoms – tai 32-bit ir 64-bit, todėl norint apimti šias dvi platformas reikia paruošti dvi alternatyvias kuriamo projekto versijas, kurios būtų suderinamos su atitinkama R aplinka (užtenka vieno paketo tačiau pakete turi būti dvi paruoštų algoritmų versijos –32-bit / 64-bit).

Paruoštas funkcionalumas gali būti pasiūlytas kitiems naudotojams paketo ar funkcijų pavidalu – R sistemoje galima kurti papildomas funkcijas ir tyrimo metodus. Norėdami pasiūlyti savo kurtą metodą kitiems naudotojams galime pateikti funkcijas arba jų rinkinius vieša forma, pavyzdžiui, forumuose, kituose internetiniuose puslapiuose. Naudotojams taip pat galima pasiūlyti susidiegti paruoštus paketus. Diegiant paketą į R sistemą susidiegia visi pagrindiniai ir šalutiniai įrankiai, kitos priklausomybės.

Skirtingoms operacinėms sistemoms R paketų turinys skiriasi – ruošiant diegimo paketus R aplinkai reikia atkreipti dėmesį į tai, kad C/C++ sukompiliuotas funkcijų failo formatas skirtingose operacinėse sistemose gali turėti skirtingas struktūras, failų pavadinimus – Windows operacinėse sistemose tai *.dll [Mic16e] formatas, *nix sistemose *.so [Ipp16]. Todėl priklausomai nuo operacinės sistemos gali prireikti paruošti aplinkai specifinius paketus.

Šalutinės pastabos – paruošti R sistemai skirtą funkcijų paketą pavyko ir sukurtos funkcijos veikia 64-bit ir 32-bit R sistemos variantuose. Diegimo paketus paruošėme Windows ir linux operacinės sistemos naudotojams (32-bit/64-bit)

Sukurtus paketus galima talpinti R paketų repozitorijose, pavyzdžiui, CRAN, taip galime galime būti garantuoti, kad naudotojai turės visas prieigas prie sukurto paketo ir kitų susijusių šaltinių. Dėl darbo panaudojimo apribojimų sukurto paketo netalpinome jokioje repozitorijoje.

Naudojant Rstudio teikiamas galimybes paketo struktūrą ir kitus reikiamus priedus turėjo galimybę sugeneruoti pačioje projekto vystymo pradžioje. Ir kitų priedų naudoti nereikėjo. Visas reikiamas funkcijas ir metodus paketų paruošimui turi ir Rtools[*].exe[16] įrankis.

Paketo paruošimas – tai paskutinė grandis R sistemos funkcionalumui paruošti, tačiau kaip pastebėjome, sukurtas paketas turi būti nuolatos atnaujintas ir derinamas prie kitų paketų ir ypač prie R sistemos versijos.

[16] * = projekte naudojama R versija

Išvados ir rezultatai

Molekulinės dinamikos algoritmai reikalauja daugybės skaičiavimo pajėgumų – esminė priežastis dėl kurios siekėme įgyvendinti našumo reikalaujančius algoritmus C/C++ kalba ir susieti juos su R aplinka. Algoritmo našumas priklauso nuo parinkto integratoriaus ir jėgų skaičiavimo metodo ir duomenų kiekio.

R aplinka neturi paketų molekulinės srities duomenų vizualizavimui atlikti – literatūros dalyje nagrinėdami R grafines galimybes radome gausybę išorinių paketų kurie siūlo įvairių grafikų ir diagramų pasirinkimą. Taip pat aptikome keletą paketų, kurių pagalbą galėjome vaizduoti grafus, dendogramas. Tačiau neaptikome jokio funkcionalumo, kuris galėtų vaizduoti molekules, molekulinius junginius, kitus molekulinės srities objektus. Todėl alternatyvus sprendimas – supaprastinti vaizduojamos dalykinės srities objektus arba per specialius failų formatus sąveikauti su išorinėmis programomis, pavyzdžiui, jmol [dt16].

Šalutinės pastabos – visus C/C++ algoritmų integracijos žingsnius darėme taip kaip jie buvo apibrėžti įvairiuose šaltiniuose. Algoritmų integracijos eigoje nepastebėjome esminių vien tik molekulinės dinamikos algoritmams būdingų išimčių, bet pastebėjome keletą R aplinkos trūkumų – gryna R kalba aprašytos funkcijos veikia lėčiau nei C/C++ kalba aprašytos funkcijos, funkcijas galima paspartinti perrašant intensyvių skaičiavimo resursų reikalaujančias dalis kitomis programavimo kalbomis ir tas dalis iškviečiant per atitinkamas programines sąsajas.

R funkciniai paketai leidžia dalintis sukurtomis funkcijomis ir metodais, tačiau tie patys paketai, jei juose esančios funkcijos remiasi išorinių kalbų galimybėmis, turi būti perkompiliuoti skirtingoms operacinėms sistemoms.

Ties šiuo skyriumi baigėme molekulinės dinamikos algoritmų integracijos į R aplinką tyrimą. Tyrimo eigoje išsiaiškinome, kad C/C++ algoritmų integraciją į R aplinką niekuo neišsiskiria nuo įprastų C/C++ algoritmų integracijos į R sistemą.

Pastebėjome, kad R aplinka neturi galimybių vaizduoti molekulinės biologijos objektų ir kaip alternatyvą galima sąveikauti su išorinėmis programomis, kurios gebėtų atvaizduoti nagrinėjamos dalykinės srities objektus. Taip pat pastebėjome, kad R aplinka turi itin didelį pasirinkimą metodų duomenų grafinei analizei atlikti.

Rezultatai

Tyrimo eigoje sukūrėme R sistemai skirtą paketą, kurio pagalba galima atlikti molekulinės dinamikos simuliacijas iš R aplinkos (naudojant klasikinius algoritmus). Literatūros apžvalgos dalyje pateikėme R programinių sąsajų, grafikos galimybių, ir papildomo funkcionalumo, kurto R

sistemai, aprašymus. Praktinės dalyje pateikėme išvadas kurias padarėme integruodami C/C++ algoritmus į R aplinką.

Literatūros sąrašas

[AEF16] J.J. Allaire, Dirk Eddelbuettel, and Romain Francois. Rcpp Attributes. <http://dirk.eddelbuettel.com/code/rcpp/Rcpp-attributes.pdf>, 2016.

[AFWIS16] Altuna Akalin, Vedran Franke, Katarzyna Wreczycka, and Liz Ing-Simmons. Summary, annotation and visualization of genomic data. <https://www.bioconductor.org/packages/release/bioc/html/genomation.html>, 2016.

[All16] JJ Allaire. rmarkdown: Dynamic Documents for R. <https://cran.r-project.org/web/packages/rmarkdown/index.html>, 2016.

[AM16] Daniel Adler and Duncan Murdoch. rgl: 3D Visualization Using OpenGL. <https://cran.r-project.org/web/packages/rgl/index.html>, 2016.

[AMCP16] Sonali Arora, Martin Morgan, Marc Carlson, and H. Pages. Utilities for manipulating chromosome and other 'seqname' identifiers. <https://www.bioconductor.org/packages/release/bioc/html/GenomeInfoDb.html>, 2016.

[APL16] P. Aboyoun, H. Pagés, and M. Lawrence. Representation and manipulation of genomic intervals and variables defined along a genome. <https://www.bioconductor.org/packages/release/bioc/html/GenomicRanges.html>, 2016.

[Bel15] Carlos J. Gil Bellosta. rPython: Package Allowing R to Call Python. <https://cran.r-project.org/web/packages/rPython/index.html>, 2015.

[BJK16] Matthias Burger, Klaus Juenemann, and Thomas Koenig. RUnit: R Unit Test Framework. <https://cran.r-project.org/web/packages/RUnit/index.html>, 2016.

[BM16] Douglas Bates and Martin Maechler. Matrix: Sparse and Dense Matrix Classes and Methods. <https://cran.r-project.org/web/packages/Matrix/index.html>, 2016.

[CH16] Simon Gert Coetzee and Dennis J. Hazelett. A Package For Predicting The Disruptiveness Of Single Nucleotide. <https://www.bioconductor.org/packages/release/bioc/html/motifbreakR.html>, 2016.

[cod16] codeblocks. The open source, cross platform, free C, C++ and Fortran IDE. <http://www.codeblocks.org/>, 2016.

[Con16] Cytoscape Consortium. cytoscape. <http://www.cytoscape.org/download.php>, 2016.

[Cor16a] Oracle Corporation. How to Write Doc Comments for the Javadoc Tool. <http://www.oracle.com/technetwork/articles/java/index-137868.html>, 2016.

[Cor16b] Oracle Corporation. NetBeans IDE Features. <https://netbeans.org/features/cpp/>, 2016.

[CPA+16] M. Carlson, H. Pagés, P. Aboyoun, S. Falcon, M. Morgan, D. Sarkar, and M. Lawrence. Tools for making and manipulating transcript centric annotations. <https://www.bioconductor.org/packages/release/bioc/html/GenomicFeatures.html>, 2016.

[CR16a] Angelo Canty and Brian Ripley. boot: Bootstrap Functions (Originally by Angelo Canty for S). <https://cran.r-project.org/web/packages/boot/index.html>, 2016.

[CR16b] Angelo Canty and Brian Ripley. Brian Ripley and William Venables. <https://cran.r-project.org/web/packages/class/index.html>, 2016.

[Csa16] Gabor Csardi. igraph: Network Analysis and Visualization. <https://cran.r-project.org/web/packages/igraph/index.html>, 2016.

[CWR16] Winston Chang, Hadley Wickham, and RStudio. ggvis: Interactive Grammar of Graphics. <https://cran.r-project.org/web/packages/ggvis/index.html>, 2016.

[Dav16] Tal Davidson. Artistic Style 2.05. <http://astyle.sourceforge.net/astyle.html>, 2016.

[DC09] David B. Dahl and Scott Crawford. RinRuby: Accessing the R Interpreter from Pure Ruby. Journal of Statistical Software, 29(4):1--18, 11 2009.

[DDT16] Stephane Dray, Anne-Beatrice Dufour, and Jean Thioulouse. ade4: Analysis of Ecological Data. <https://cran.r-project.org/web/packages/ade4/index.html>, 2016.

[DH16] Steffen Durinck and Wolfgang Huber. Interface to BioMart databases (e.g. Ensembl, COSMIC ,Wormbase and Gramene). <https://www.bioconductor.org/packages/release/bioc/html/biomaRt.html>, 2016.

[Dic16] Dictionary.com. molecular biology. <http://www.dictionary.com/browse/molecular-biology>, 2016.

[dJvdL13] Edwin de Jonge and Mark van der Loo. An introduction to data cleaning with R. 2013.

[dt16] Jmol development team. Jmol: an open-source Java viewer for chemical structures in 3D. <http://jmol.sourceforge.net/>, 2016.

[Edg04] R.C. Edgar. MUSCLE. <http://www.ncbi.nlm.nih.gov/pubmed/15034147>, 2004.

[EF16] Dirk Eddelbuettel and Romain François. Exposing C++ functions and classes with Rcpp modules. January 2016.

[EFA+16] Dirk Eddelbuettel, Romain Francois, JJ Allaire, Kevin Ushey, Qiang Kou, Douglas Bates, and John Chambers. Rcpp: Seamless R and C++ Integration. <https://cran.r-project.org/web/packages/Rcpp/index.html>, 2016.

[Erc16] Furio Ercolessi. The Verlet algorithm. <http://www.fisica.uniud.it/~ercolessi/md/md/node21.html#SECTION00431000000000000000>, 2016.

[FDSB96] Frenkel, Daan, Smit, and Berend. Understanding Molecular Simulation: From Algorithms to Applications. Academic Press, Inc., Orlando, FL, USA, 1st edition, 1996.

[Fou16a] Eclipse Foundation. Eclipse CDT (C/C++ Development Tooling). <https://eclipse.org/cdt/>, 2016.

[Fou16b] Python Software Foundation. Python. <https://www.python.org/>, 2016.

[Fou16c] R Foundation. Building R for Windows. <https://cran.r-project.org/bin/windows/Rtools/>, 2016.

[Fou16d] R Foundation. SEXPTYPEs. <https://cran.r-project.org/doc/manuals/r-release/R-ints.html#SEXPTYPEs>, 2016.

[Fou16e] R Foundation. Writing R Extensions. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>, 2016.

[Fou16f] The R Foundation. 2 .Internal vs .Primitive. <https://cran.r-project.org/doc/manuals/r-release/R-ints.html#g_t_002eInternal-vs-_002ePrimitive>, 2016.

[Fou16g] The R Foundation. 2.1 What is R? <https://cran.r-project.org/doc/FAQ/R-FAQ.html>, 2016.

[Fou16h] The R Foundation. Autoprinting. <https://cran.r-project.org/doc/manuals/r-release/R-ints.html#Autoprinting>, 2016.

[Fou16i] The R Foundation. Calling .External. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Calling-_002eExternal>, 2016.

[Fou16j] The R Foundation. Details of R types. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Details-of-R-types>, 2016.

[Fou16k] The R Foundation. Documentation for package 'stats' version 3.4.0. <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html>, 2016.

[Fou16l] The R Foundation. Documentation for package 'graphics' version 3.4.0. <https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html>, 2016.

[Fou16m] The R Foundation. Handling R objects in C. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Handling-R-objects-in-C>, 2016.

[Fou16n] The R Foundation. Handling the effects of garbage collection. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Garbage-Collection>, 2016.

[Fou16o] The R Foundation. Interface functions .C and .Fortran. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>, 2016.

[Fou16p] The R Foundation. Interface functions .C and .Fortran. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Interface-functions-_002eC-and-_002eFortran>, 2016.

[Fou16q] The R Foundation. Internals.h. <https://github.com/wch/r-source/blob/e5b21d0397c607883ff25cca379687b86933d730/src/include/Internal.h>, 2016.

[Fou16r] The R Foundation. names.c. <https://svn.r-project.org/R/trunk/src/main/names.c>, 2016.

[Fou16s] The R Foundation. R (D)COM Server and RExcel. <https://cran.r-project.org/contrib/extra/dcom/00ReadMe.html>, 2016.

[Fou16t] The R Foundation. R Internals. <https://cran.r-project.org/doc/manuals/r-release/R-ints.html>, 2016.

[Fou16u] The R Foundation. Speed considerations. <https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Speed-considerations>, 2016.

[Fou16v] The R Foundation. What is R? <https://www.r-project.org/about.html>, 2016.

[Gau16] L. Gautier. rpy2 - R in Python. <http://rpy2.bitbucket.org/>, 2016.

[GdCC16] Markus Gesmann, Diego de Castillo, and Joe Cheng. googleVis: R Interface to Google Charts. <https://cran.r-project.org/web/packages/googleVis/index.html>, 2016.

[GMRS16] Alexis Gabadinho, Nicolas S. Müller, Gilbert Ritschard, and Matthias Studer. TraMineR: a toolbox for exploring sequence data. <http://traminer.unige.ch/>, 2016.

[Gro16] Philippe Grosjean. svUnit: SciViews GUI API - Unit testing. <https://cran.r-project.org/web/packages/svUnit/index.html>, 2016.

[GYSI16] Barry Grant, Xinqiu Yao, Lars Skjaerven, and Julien Ide. bio3d. <http://thegrantlab.org/bio3d/index.php>, 2016.

[Hab16] Vanja Haberle. Visualising oligonucleotide patterns and motif occurrences across a set of sorted sequences. <https://www.bioconductor.org/packages/release/bioc/html/seqPattern.html>, 2016.

[HDI+16] Florian Hahne, Steffen Durinck, Robert Ivanek, Arne Mueller, Steve Lianoglou, Ge Tan, Lance Parsons, and Shraddha Pai. Plotting data and annotation information along genomic coordinates. <https://www.bioconductor.org/packages/release/bioc/html/Gviz.html>, 2016.

[Hes16] Jim Hester. lintr: Static R Code Analysis. <https://cran.r-project.org/web/packages/lintr/index.html>, 2016.

[HKB16a] James Honaker, Gary King, and Matthew Blackwell. Amelia: A Program for Missing Data. <https://cran.r-project.org/web/packages/Amelia/index.html>, 2016.

[HKB16b] James Honaker, Gary King, and Matthew Blackwell. JAGUAR. <https://cran.r-project.org/web/packages/JAGUAR/index.html>, 2016.

[Ipp16] Greg Ippolito. Static, Shared Dynamic and Loadable Linux Libraries. <http://www.yolinux.com/TUTORIALS/LibraryArchives-StaticAndDynamic.html>, 2016.

[JM15] J-M. R.NET for users 1.6. <http://jmp75.github.io/rdotnet/about/>, 2015.

[Jr16] Frank E Harrell Jr. rms: Regression Modeling Strategies. <https://cran.r-project.org/web/packages/rms/index.html>, 2016.

[Kuh16] Max Kuhn. caret: Classification and Regression Training. <http://caret.r-forge.r-project.org/>, 2016.

[LCG16] Michael Lawrence, Vince Carey, and Robert Gentleman. R interface to genome browsers and their annotation tracks. <https://www.bioconductor.org/packages/release/bioc/html/rtracklayer.html>, 2016.

[LMS16] Uwe Ligges, Martin Maechler, and Sarah Schnackenberg. scatterplot3d: 3D Scatter Plot. <https://cran.r-project.org/web/packages/scatterplot3d/index.html>, 2016.

[Lp16] Latex-project. LaTeX – A document preparation system. <https://www.latex-project.org/>, 2016.

[LSWL16] Duncan Temple Lang, Debby Swayne, Hadley Wickham, and Michael Lawrence. rggobi: Interface between R and GGobi. <https://cran.r-project.org/web/packages/rggobi/index.html>, 2016.

[Mes16] Johan Mes. Dev-C++. <https://sourceforge.net/projects/orwelldevcpp/>, 2016.

[MG16] Eloi Mercier and Raphael Gottardo. Motif Identification and Validation. <http://www.dictionary.com/browse/molecular-biology>, 2016.

[Mic16a] Microsoft. Microsoft R Server. <https://www.microsoft.com/en-us/server-cloud/products/r-server/>, 2016.

[Mic16b] Microsoft. rClr - R package to access .NET. <https://rclr.codeplex.com/>, 2016.

[Mic16c] Microsoft. Visual Studio. <https://www.visualstudio.com/>, 2016.

[Mic16d] Microsoft. What Is a COM Interface? <https://msdn.microsoft.com/en-us/library/windows/desktop/ff485850(v=vs.85).aspx>, 2016.

[Mic16e] Microsoft. What is a DLL? <https://support.microsoft.com/en-us/kb/815065>, 2016.

[Min16] MinGW.org. MinGw. <http://www.mingw.org/>, 2016.

[MPVO16] Martin Morgan, Hervé Pagès, and Nathaniel Hayden Valerie Obenchain. Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import. <https://www.bioconductor.org/packages/release/bioc/html/Rsamtools.html>, 2016.

[MRS+16] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, Kurt Hornik, Matthias Studer, and Pierre Roudier. cluster: Finding Groups in Data. <https://cran.r-project.org/web/packages/cluster/index.html>, 2016.

[MW16] Walter Moreira and Gregory Warnes. RPy (R from Python). <https://sourceforge.net/projects/rpy/>, 2016.

[OBWZ16] Jianhong Ou, Michael Brodsky, Scot Wolfe, and Lihua Julie Zhu. Plot stacked logos for single or multiple DNA, RNA and amino acid sequence. <https://www.bioconductor.org/packages/release/bioc/html/motifStack.html>, 2016.

[oM16] National Library of Medicine. BLAST. <http://blast.ncbi.nlm.nih.gov/Blast.cgi>, 2016.

[OMLG16] Valerie Obenchain, Martin Morgan, Michael Lawrence, and Stephanie Gogarten. Annotation of Genetic Variants. <https://www.bioconductor.org/packages/release/bioc/html/VariantAnnotation.html>, 2016.

[Ora16] Oracle. Free Java Download. <https://java.com/en/download/>, 2016.

[Pag16] Herve Pages. Infrastructure for Biostrings-based genome data packages and support for efficient SNP representation. <https://www.bioconductor.org/packages/release/bioc/html/BSgenome.html>, 2016.

[PAGD16] H. Pages, P. Aboyoun, R. Gentleman, and S. DebRoy. String objects representing biological sequences, and matching algorithms. <https://www.bioconductor.org/packages/release/bioc/html/Biostrings.html>, 2016.

[PdL02] Roger D. Peng and Jan de Leeuw. An Introduction to the .C Interface to R. April 2002.

[Per16] Perl.org. perl. <https://www.perl.org/>, 2016.

[POM16] Hervé Pagès, Valerie Obenchain, and Martin Morgan. Representation and manipulation of short genomic alignments. <https://www.bioconductor.org/packages/release/bioc/html/GenomicAlignments.html>, 2016.

[Ran12] Aarne Ranta. Implementing Programming Languages. February 2012.

[Rap04] D.C. Rapaport. The Art of Molecular Dynamics Simulation. Cambridge University Press, 2004.

[RBV16] Brian Ripley, Roger Bivand, and William Venables. spatial: Functions for Kriging and Point Pattern Analysis. <https://cran.r-project.org/web/packages/spatial/index.html>, 2016.

[rcs16] rcsb. pdb. <http://www.rcsb.org/pdb/static.do?p=file_formats/pdb/index.html>, 2016.

[RH16] Inc Red Hat. Cygwin. <https://cran.r-project.org/bin/windows/Rtools/>, 2016.

[RRL+16] Christian Roever, Nils Raabe, Karsten Luebke, Uwe Ligges, Gero Szepannek, and Marc Zentgraf. klaR: Classification and visualization. <https://cran.r-project.org/web/packages/klaR/index.html>, 2016.

[RSt14] Inc RStudio. shiny: Easy web applications in R. <http://shiny.rstudio.com>, 2014.

[Rst16a] Inc Rstudio. Rstudio. <https://www.rstudio.com/>, 2016.

[Rst16b] Inc Rstudio. RStudio Will Not Start. <https://support.rstudio.com/hc/en-us/articles/200488508-RStudio-Will-Not-Start>, 2016.

[RV16] Brian Ripley and William Venables. nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models. <https://cran.r-project.org/web/packages/nnet/index.html>, 2016.

[RVB+16] Brian Ripley, Bill Venables, Douglas M. Bates, Kurt Hornik, Albrecht Gebhardt, and David Firth. MASS: Support Functions and Datasets for Venables and Ripleyś MASS. <https://cran.r-project.org/web/packages/MASS/index.html>, 2016.

[Sar16] Deepayan Sarkar. lattice: Trellis Graphics for R. <https://cran.r-project.org/web/packages/lattice/index.html>, 2016.

[Sch16] Christian Schenk. Miktex. <http://miktex.org/>, 2016.

[SCLB16] Debby Swayne, Di Cook, Duncan Temple Lang, and Andreas Buja. Ggobi. <http://www.ggobi.org/>, 2016.

[Sha16] Paul Shannon. Display and manipulate graphs in Cytoscape. <https://www.bioconductor.org/packages/release/bioc/html/RCytoscape.html>, 2016.

[SMS+16] Oleg Sklyar, Duncan Murdoch, Mike Smith, Dirk Eddelbuettel, Romain Francois, and Karline Soetaert. inline: Functions to Inline C, C++, Fortran Function Calls from R. January 2016.

[SPH+16] Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. plotly: Create Interactive Web Graphics via 'plotly.js'. <https://cran.rstudio.com/web/packages/plotly/>, 2016.

[Str13] Bjarne Stroustrup. C and C++: Siblings. April 2013.

[swi09] swissr. rpascal. <https://github.com/swissr/rpascal>, 2009.

[Tan16] Ge Tan. TFMPvalue: Efficient and Accurate P-Value Computation for Position Weight Matrices. <https://cran.r-project.org/web/packages/TFMPvalue/index.html>, 2016.

[TAR16] Terry Therneau, Beth Atkinson, and Brian Ripley. rpart: Recursive Partitioning and Regression Trees. <https://cran.r-project.org/web/packages/rpart/index.html>, 2016.

[Tcw16] R Core Team and contributors worldwide. The R Datasets Package. <https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/datasets-package.html>, 2016.

[Tie16] Luke Tierney. codetools: Code Analysis Tools for R. <https://cran.r-project.org/web/packages/codetools/index.html>, 2016.

[TL16] Terry M Therneau and Thomas Lumley. survival: Survival Analysis. <https://cran.r-project.org/web/packages/survival/index.html>, 2016.

[Uni16] Vanderbilt University. Web Application Development with R and Apache. <http://rapache.net/>, 2016.

[Urb16a] Simon Urbanek. JRI - Java/R Interface. <https://rforge.net/JRI/index.html>, 2016.

[Urb16b] Simon Urbanek. rJava: Low-Level R to Java Interface. <https://cran.r-project.org/web/packages/rJava/index.html>, 2016.

[UW16] Simon Urbanek and Tobias Wichtrey. iplots: iPlots - interactive graphics for R. <https://cran.r-project.org/web/packages/iplots/index.html>, 2016.

[Vai16] Ramnath Vaidyanathan. rCharts. <http://rcharts.io/>, 2016.

[vH16] Dimitri van Heesch. Doxygen. <http://www.stack.nl/\~dimitri/doxygen/>, 2016.

[Wal16] WalWare. RJ & RServi. <http://www.walware.de/;jsessionid=2a594325d1cb2b8d4b002d2b9854?page=/it/rj/index.mframe>, 2016.

[WCR16] Hadley Wickham, Winston Chang, and RStudio. ggplot2: An Implementation of the Grammar of Graphics. <https://cran.r-project.org/web/packages/ggplot2/index.html>, 2016.

[WCRt16a] Hadley Wickham, Winston Chang, RStudio, and R Core team. devtools: Tools to Make Developing R Packages Easier. <https://cran.r-project.org/web/packages/devtools/index.html>, 2016.

[WCRt16b] Hadley Wickham, Winston Chang, RStudio, and R Core team. roxygen2: In-Source Documentation for R. <https://cran.r-project.org/web/packages/roxygen2/index.html>, 2016.

[Wic16a] Hadley Wickham. Finding the C source code for a function. <http://adv-r.had.co.nz/C-interface.html>, 2016.

[Wic16b] Hadley Wickham. High performance functions with Rcpp. <http://adv-r.had.co.nz/Rcpp.html>, 2016.

[Wic16c] Hadley Wickham. R packages. <http://r-pkgs.had.co.nz/src.html>, 2016.

[Wic16d] Hadley Wickham. R's C interface. <http://adv-r.had.co.nz/C-interface.html>, 2016.

[Wic16e] Hadley Wickham. Vignettes: long-form documentation. <http://r-pkgs.had.co.nz/vignettes.html>, 2016.

[Wil07] Leland Wilkinson. The Grammar of Graphics. Journal of Statistical Software, 17:1--7, 01 2007.

[WR16] Hadley Wickham and RStudio. testthat: Unit Testing for R. <https://cran.r-project.org/web/packages/testthat/index.html>, 2016.

[Xie16] Yihui Xie. formatR: Format R Code Automatically. <https://cran.r-project.org/web/packages/formatR/index.html>, 2016.

[YLC16] Tengfei Yin, Michael Lawrence, and Dianne Cook. Basic graphic utilities for visualization of genomic data. <https://www.bioconductor.org/packages/release/bioc/html/biovizBase.html>, 2016.

[You14] Peter Young. The leapfrog method and other "symplectic"algorithms for integrating Newton's laws of motion. April 2014.

Priedas. Į R aplinką integruotų algoritmų taikymo pavyzdžiai

Visi integruoti algoritmai prasideda žodžiu maino[00] (angl. main output). Skaičiukai esantys gale padeda sukurti unikalų pavadinimą algoritmo kvietimui.

A. maino panaudojimo būdai

maino – algoritmas naudojamas sistemai sudarytai iš kietų sferų simuliuoti. Simuliacija vyksta dvimačiame plote apribotu x ir y parametrais, sistemoje esančių objektų kiekis priklauso nuo pasirinktų x ir y parametrų ir yra lygus jų sandaugai (molekulių kiekis = x * y). Kviečiant algoritmą maino, sistemai taip pat galima suteikti tankį, pradinę temperatūrą. Algoritmo veikimui parametras stepEquil jokios įtakos nedaro (neįgyvendintas funkcionalumas). DeltaT parametru nustatome laiko pokyti, per kurį yra atliekamas simuliacijos pilnas ciklas (apskaičiuojamos objektų koordinates, įvertinami pagreičiai, laiko pokytis, visuotinė sistemos energija, kinetinė energija). Simuliacija trunka iki kol sistemos simuliacijos ciklų skaičius (stepCount) nepasiekia nustatyto stepLimit parametro reikšmės.

Algoritmas grąžina įvykdytų žingsnių (ciklų) skaičių (stepCount). Esamą laiko momentą simuliacijos eigoje (timeNow = delta * stepCount, sistemą sudarančių objektų greičių sumas (vcSum)), pilną sistemos energijos vidurkį (totEnergySum), sistemos vidutinę kinetinę energiją (kinEnergySum), kitus parametrus [17]. Sistemos sistemos evoliucijai atlikti yra naudojamas leapfrog metodas (integratorius). Sąveikos tarp kitų sistemos objektų yra skaičiuojamos rCut = pow (2., 1./6.) spinduliu.

B. Panaudojimo atvejis nr.1. sistemos savybių tyrimas

simuliuojamą sistemą riboja plokštuma (20 x 20), sistemos tankis 0.9, sistema evoliucionuoja kas 0.05 s. Sistemos įverčiai apskaičiuojami kas 200 ciklų. Pradinė temperatūrą 1. Kaip kis tokios sistemos vidutinė pilnutinė energija (totEnergySum) įvykdžius 100000 ciklų?

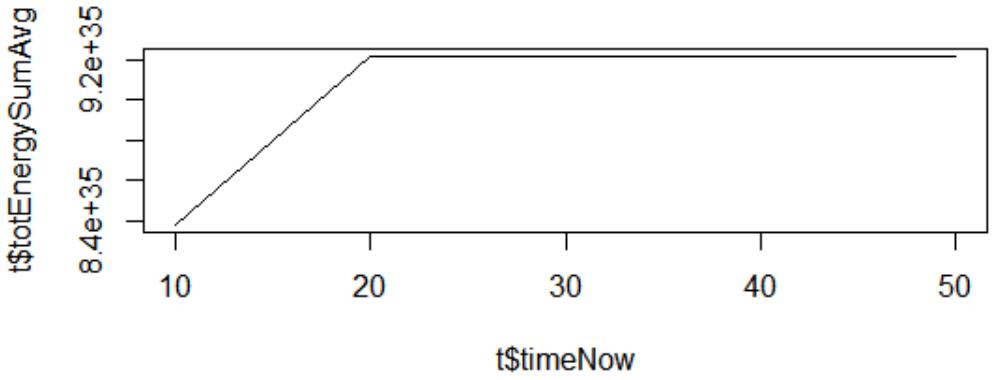
B.1. Tyrimo eiga

Surenkame pradinius duomenis – deltaT = 0.05, density = 0.9, initUCell.x = 20, initUCell.y = 20, stepAvg = 200, stepEquil = 0 (nenaudojamas), stepLimit = 100000, temperature = 1;

kviečiame maino funkciją – t <- maino(0.05, 0.9, 20, 20, 200, 0, 100000, 1);

[17] pilną aprašymą galima pamatyti suvedus R konsolėje ?maino

stepCount	timeNow	vcSum	totEnergySumAvg	totEnergySumAvgSquare
200	10	40021778	4.132931e+50	6.990302e+49
400	20	40021778	4.255270e+50	0
600	30	40021778	4.255270e+50	0
<...>	<...>	<...>	<...>	<...>



1 pav.: pav. 1 Rezultatai įvykdžius maino funkciją

darome išvadas – po 100000 ciklų matyti, kad sistema nėra pusiausvyros būsenoje (visos reikšmės itin didelės. Pusiausvyros būsenoje greičiai (vcSum) yra artimi 0.). Galime teigti, kad pilnutinė sistemos energija kurį laiką didėja, o paskui išlieka stabili. Taip pat iš gautų rezultatų galime daryti prielaidą, kad duotieji parametrai buvo nekorektiški (per didelis laiko pokytis deltaT).

C. maino3 panaudojimo būdai

maino3 – skirtingai nei maino funkcija, maino3 sistemos simuliaciją atlieką trimatėje erdvėje, turi papildomą parametrą (stepIntlTemp) sistemos temperatūrai koreguoti sistemos vykdymo metu. Sistemos temperatūros koregavimas leidžia sumažinti pradinių nustatymų įtaką galutiniams rezultatams. Sistemos objektų sąveikos modeliuojamos gretimų langelių metodu (cell subdivison) – esminis šio algoritmo išskirtinumas. Kaip ir maino funkcijos atveju taip ir su maino3 galime atlikti tuos pačius tyrimus. Tik maino3 algoritmas naudoja kitokią tarpusavio sąveikų skaičiavimo mechanizmą (cell subdivison). O, maino skaičiuoja sąveikas iš anksto apibrėžtu spinduliu.

C.1. Panaudojimo atvejis nr.2 skirtingų sąveikų skaičiavimo metodų palyginimas

Simuliuojamos dvi sistemos, pirmos plotas plotas 20 x 20, antrosios 20 x 20 x 20, sistemų tankiai 0.7, sistemų įverčiai skaičiuojami kas 20 ciklų. Pradinės temperatūros 1. Palyginkite pil-

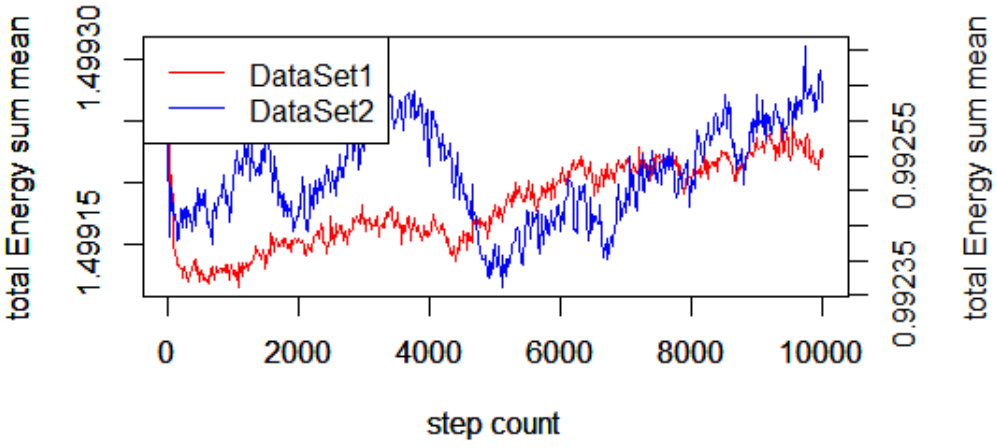
nutinės energijos pokytį naudodami maino ir maino3 algoritmus. Sistemos simuliacija atliekama 10000 kartų, laiko pokytis deltaT = 0.005, stepInitlzTemp = 99999,

Surenkame pradinius duomenis – InitUCell.x = 20, InitUCell.y = 20, InitUcell.z = 0 (naudosime dvimatę erdvę). deltaT = 0.005, stepAvg = 20; stepLimit = 10000 ; stepInitlzTemp = 99999, density = 0.7.; temperature = 1.

Kviečiame maino ir maino3 funkcijas t <- maino(0.005, 0.8, 20, 20, 20, 0, 10000, 1); t1 <- maino3(0.005, 0.7, 20, 20, 20, 20, 0, 99999, 10000, 1);

Kviečiame funkcija, kuri braižo grafiką su dviem y ašimis compareTwoSets(t$stepCount, t$totEnergySumAvg, t1$totEnergySumAvg, "step count", "total Energy sum mean");

Gauname grafiką



2 pav.: DataSet1 (t) ir DataSet2 (t1) pilnutinių energijų vidurkio palyginimas

Darome išvadas iš grafiko (žr. pav. 2) matyti, kad abiejų sistemų pilnutinė energija, bėgant laikui, didėja. t1 atveju pilnutinė energija didėja ne taip nuosekliai kaip t.

D. maino4 panaudojimo būdai

maino4 – veikimo principas yra ekvivalentus maino3 funkcijai, skiriasi tik objektų tarpusavio sąveikų metodu. Maino4 atveju sistemos objektų tarpusavio sąveikoms skaičiuoti yra naudojamas artimiausių kaimynų sąrašo metodas. Atsiranda trys nauji kintamieji – nebrTabFac – skirtas didžiausiam galimam kaimynų skaičiui nustatyti. randomSeed – atsitiktinėms objektų pozicijoms generuoti, rNebrShell – kaimynystės spindulio ilgis.

D.1. Panaudojimo atvejis nr.3 Palyginti sistemų pilnutines energijas esant skirtingoms randomSeed reikšmėms

randomSeed – tai kintamasis, kuris naudojamas atomų (objektų) atsitiktiniam generavimui inicijuoti. Atlikdami šią užduotį turėtume išsiaiškinti ar kintamojo randSeed variavimas įtakoja sistemos pilnutinę energiją. Iškvieskite maino4 funkciją 5 kartus su tokiomis randomSeed reikšmėmis 10, 10000, 50000, 12, 1000000 laikydami, kad laiko pokytis deltaT = 0.005, sistemos tankis 0.8, simuliacijos plotas 5 x 5 x 5, nebrTabFac = 10, nebrShell = 0.5 stepAvg = 10, stepEquil = 0, stepInitlzTemp = 99999, stepLimit = 10000, temperature = 1 reikšmės yra pastovios.

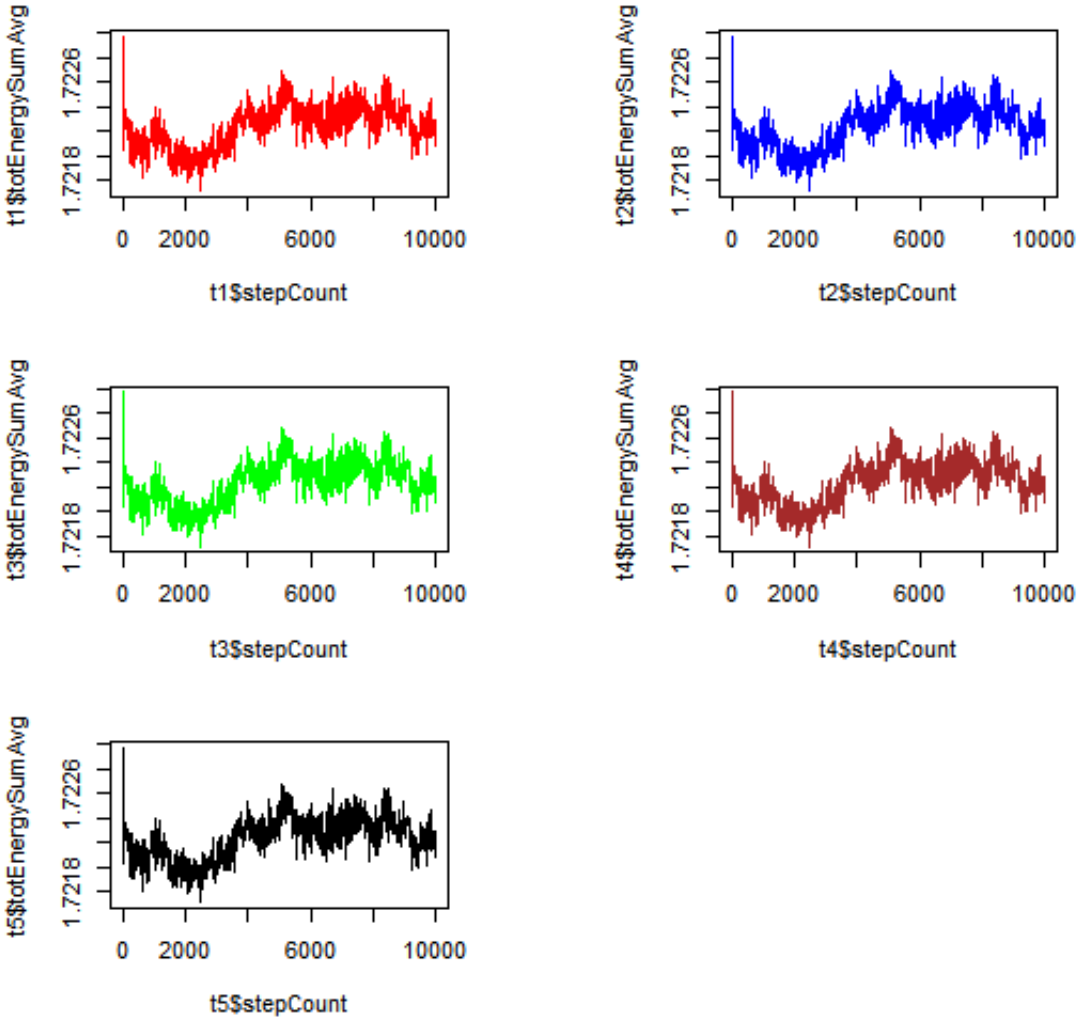
Surenkame pradinius duomenis – randSeed1 = 10, randSeed2 = 10000, randSeed3 = 50000, randSeed4 = 12, randSeed5 = 1000000; deltaT = 0.005, density = 0.8, InitUCell.x = 5, InitUCell.y = 5, InitUCell.z = 5, nebrTabFac = 10, nebrShell = 0.5 stepAvg = 10, stepEquil = 0, stepInitlzTemp = 99999, stepLimit = 10000, temperature = 1.

Kviečiame maino4 funkciją penkis kartus

```
t1 <- maino4 (0.005, 0.8, 5, 5, 5, 10, 10, 0.5, 10, 0, 0, 10000, 1);
t2 <- maino4 (0.005, 0.8, 5, 5, 5, 10000, 10, 0.5, 10, 0, 0, 10000, 1);
t3 <- maino4 (0.005, 0.8, 5, 5, 5, 50000, 10, 0.5, 10, 0, 0, 10000, 1);
t4 <- maino4 (0.005, 0.8, 5, 5, 5, 12, 10, 0.5, 10, 0, 0, 10000, 1);
t5 <- maino4 (0.005, 0.8, 5, 5, 5, 1000000, 10, 0.5, 10, 0, 0, 10000, 1);
```

Gauname grafikus

```
par(mfrow=c(3,2))
plot(t1$stepCount, t1$totEnergySumAvg, type="l", col="red");
plot(t2$stepCount, t2$totEnergySumAvg, type="l", col="blue");
plot(t3$stepCount, t3$totEnergySumAvg, type="l", col="green");
plot(t4$stepCount, t4$totEnergySumAvg, type="l", col="brown");
plot(t5$stepCount, t5$totEnergySumAvg, type="l", col="black");
```



3 pav.: randomSeed kintamojo įtaka pilnutinei sistemos energijai

Darome išvadas kaip matyti iš pateiktų grafikų (žr. pav.3) randSeed parinkimas pilnutinei sistemos energijai įtakos neturi.

E. maino5 panaudojimo būdai

maino5 – algoritmas yra panašus į jau anksčiau minėtus. Tik sistemos evoliucija remiasi ne leapfrog metodu, o predictor/corrector integravimo metodu, sistemoje esančių objektų sąveikoms apskaičiuoti remiamasi gretutinių langelių metodu. Kadangi sistemos evoliucijai naudojame predictor/ corrector metodą, galime vykdyti sistemos vidinius procesus (objektų tarpusavio sąveikos modeliavimą) didesniu laiko žingsniu (lyginant su leapfrog).

F. maino6 panaudojimo būdai

maino6 – algoritmas panašus maino5 algoritmui, tačiau objektų sąveikom apskaičiuoti yra naudojamas kaimynų sąrašo metodas, integratorius – predictor/corrector metodas. Naudodami šį algoritmą, galime lyginti skirtingų sąveikų įtakas, simuliuojamoms sistemoms, įvertinti, lyginant, pavyzdžiui, su maino5 ir kitais algoritmais.

G. maino8 panaudojimo būdai

maino8 – algoritmas skirtas simuliuoti sistemą, kurios objektai yra minkštos sferos (soft spheres). Sistemai evoliucionuoti yra naudojamas leapfrog metodas, objektų tarpusavio sąveikos skaičiuojamos remiantis kaimynų sąrašu. Galima naudoti minkštų ir ketų sferų sistemoms palyginti.

H. maino9 panaudojimo būdai

maino9 – algoritmas skirtas studijuoti kristalų gardelės periodiškumo savybėms. Objektų tarpusavio sąveikoms įvertinti naudojamas kaimynų sąrašo metodas, sistemos integratorius – leapfrog metodas.

I. maino11 panaudojimo būdai

maino11 – algoritmas ekvivalentus maino9 algoritmui. Skirtumas – paduodamų parametrų kiekyje – maino11 leidžia pasirinkti sąveikos su objektais spindulį rCut. Dažnu atveju, visuose algoritmuose naudojame empiriniu būdu parinktą rCut reikšmę. Šį algoritmą galima taikyti norint tirti kristalų gardelės periodiškumo savybėm, esant skirtingoms rCut reikšmėms.

J. maino12 panaudojimo būdai

maino12 – algoritmas klasterių susidarymą sistemoje, naudojant kaimynų sąrašo metodą, ir leapfrog integratorių. Algoritmas naudojamas, kai norime atspausdinti į R konsolę esamos sistemos konfigūraciją (einama laiko momentą, molekulių kiekį, esamų molekulių padėtis. Daugiau informacijos žiūrėti dokumentacijoje). Šį algoritmą galime naudoti, kai norime studijuoti klasterių susidarymą sistemoje.

K. maino17 panaudojimo būdai

maino17 leidžia simuliuoti kristalo evoliuciją, veikiamą pastovaus išorinio slėgio ir temperatūros. Sistemos integratorius – predictor/corrector metodas, sąveikos skaičiuojamos – gretutinių langelių metodu.

L. maino18 panaudojimo būdai

maino18 – leidžia simuliuoti sistemą, kurios temperatūrą, visą simuliacijos laiką, yra pastovi. Kitaip tariant maino18 leidžia simuliuoti taip vadinamą NVT ansamblį.

M. maino19 panaudojimo būdai

maino19 – šio algoritmo pagalba galime simuliuoti NPT ansamblį, kur sistemą veikia pastovi temperatūra ir pastovus išorinis slėgis. Sistemos integratorius predictor/corrector metodas, sąveikos skaičiuojamos – gretutinių langelių metodu.

N. maino23 panaudojimo būdai

maino23 – šio algoritmo pagalba galima tirti sistemas, kurios nėra pusiausvyros būsenoje. Sistemos integratorius predictor/corrector metodas. Sąveikoms skaičiuoti naudojamas kaimynų sąrašo metodas.

O. maino28 panaudojimo būdai

maino28 – algoritmas naudojamas standžių[18] molekulių simuliacijoms atlikti. Sistemos integratorius – leapfrog metodas. Objektų tarpusavio sąveikos skaičiuojamos kaimynų sąrašo metodu.

P. maino30 panaudojimo būdai

maino30 – algoritmas naudojamas simuliuoti molekules sujungtas paslankiais[19] ryšiais. Sistemos integratorius – leapfrog metodas. Objektų tarpusavio sąveikos skaičiuojamos kaimynų sąrašo metodu.

[18] molekulės sudarytos iš atomų. Kai kurie atomai formuoja tokius ryšius, kurie nėra paslankūs. Tokie ryšiai tarp molekulių ar jos dalių neleidžia joms ar jų dalims judėti.

[19] Be standžių ryšių dar yra paslankūs ryšiais – tai tokie ryšiai, kurie nesurakina molekulių ar jų dalių kartu. Molekulės gali laisvai judėti, pavyzdžiui, apie savo ašį, temptis

Q. maino39 panaudojimo būdai

maino39 – algoritmas skirtas simuliuoti objektų sąveikas remiantis embedded-atom metodu. Sistemos integratorius – leapfrog metodas. Sąveikos skaičiuojamos artimų langelių metodu. Simuliacijos išvestys atspindi sistemos evoliuciją, į kurios vyksmą yra įtraukiamos ilgo nuotolio sąveikos

R. maino41 panaudojimo būdai

maino41 – algoritmas simuliuoja ilgo nuotolio sąveikas sistemoje naudojant Evaldo sumavimo (Ewald summation) metodą. Sistemos integratorius – leapfrog metodas. Objektų sąveikos skaičiavimo metodas – artimų langelių metodas. Simuliuoja ilgo nuotolio objektų įtaką sistemos evoliucijai.

S. maino46 panaudojimo būdai

maino46 – algoritmas skirtas skysčių simuliacijai atlikti (simuliuoja skysčio tekėjimą (minkštų sferų) per kliūtį). Sistemos integratorius – leapfrog metodas. Sąveikos skaičiuojamos remiantis kaimynų sąrašo metodu.

T. maino47 panaudojimo būdai

maino47 – algoritmas simuliuoja objektų vibruojantį sluoksnį dvimatėje erdvėje. Sąveikų metodas – kaimynų sąrašas. Sistemos integratorius – leapfrog metodas. Simuliacija pagrįsta minkštų sferų sąveika.

U. maino48 panaudojimo būdai

maino47 – algoritmas simuliuoja atomų vibruojantį sluoksnį dvimatėje erdvėje. Sąveikų metodas – kaimynų sąrašas. Sistemos integratorius – leapfrog metodas. Simuliacija pagrįsta minkštų sferų sąveika.

V. Pastabos

Visi integruoti algoritmai grąžina reikšmes (daugiau informacijos žr. dokumentacijoje), kurias, pavyzdžiui, galima vaizduoti grafikais, todėl galima spręsti panašaus tipo uždavinius kaip apra-

šyta pirmame antrame ir trečiame pritaikymo būduose. Algoritmus taip pat galima panaudoti ir platesniame kontekste derinant juos su R sistemos teikiamomis galimybėmis.

Priedas. Parametrų generavimas

Simuliacijos sėkmę gali įtakoti daugybę veiksnių – parinktas sistemos evoliucijos metodas (leapfrog, Runge-Kutta, Verlet, kt.), sistemos evoliucijos žingsnis (*timeStep*), kiti, sistemai paduodami parametrai. Sėkmė taip pat gali turėti skirtingą reikšmę atliekant sistemos modeliavimą ir pavyzdžiui gali būti susietą su sistemos integratoriaus paklaidą, sistemos pusiausvyros būsena, potencinės / kinetinės energijos virsmais ar visuotinės sistemos energijos dydžiu, kitais parametrais.

Šioje darbo dalyje parodysime kaip R aplinka gali būti panaudota tinkamų[20] parametrų, simuliacijos vykdyti, atrankai atlikti.

W. Problema

Ne visų, sistemai paduodamų, parametrų reikšmės yra tinkamos simuliacijoms atlikti, pavyzdžiui, modeliuojant nedidelę sistemą (20 x 20 objektų), parinkus didžiules sistemos temperatūros, tankio reikšmes atsiranda rizika, kad sistema praras (nepasieks) pusiausvyros būseną ir gautos išvestys neatspindės rezultatų, kuriuos galėtume gauti, gyvai, atlikus eksperimentą.

X. Uždavinio formuluotė

Surasti tokią aibę A parametrų, kad simuliuojama 20 x 20 objektų sistema po 10 000 iteracijų vis dar būtų pusiausvyros būsenoje. Simuliuojamą sistemą sudaro kietos sferos, integratorius – leapfrog metodas, sistemos tankio reikšmė parenkama iš [0,008; 0,10] intervalo ir gali kisti 0.001 žingsniu. Leistinas Energijos kitimo intervalas [0,001; 0,01] pokyčio žingsnis 0,003. Temperatūra parenkama iš intervalo [1; 1,18] ir gali kisti 0,05 žingsniu. Sistemos įverčiai apskaičiuojami kas 100 iteracijų, laiko žingsnis 0,005.

X.1. Tyrimo eiga

Surenkame parametrus iš duoto uždavinio:

deltaT	0.005	stepLimit	10000
density	[0,008; 0,10]	temperature	[1; 1,18]
initUcell.x	20	energyDriftStep	0,003
initUcell.y	20	densityStep	0,001
stepAvg	1000	temperatureStep	0,05
stepEquil	[0,001; 0,01]		

[20] tinkamumas yra apibrėžiamas keliamos problemos ar suformuluoto uždavinio

Užduoties sąlygas tenkina algoritmas maino. Todėl jį naudosime tyrimui atlikti. Tačiau maino algoritmas negali simuliuoti sistemos duotų rėžių ribose, t.y vienu metu galime atlikti simuliacijas su viena temperatūros, tankio ir energijos pokyčio reikšme duotu laiko momentu. Pasinaudodami R sistemos galimybėmis sukursime papildomą R algoritmą, kuris atliks simuliacijas su visomis, intervaluose esamomis, reikšmėmis.

X.2. R algoritmo pagrindinės dalys

Pirmiausiai apsibrėžiame duomenų objektą, kurį naudosime R funkcijai paduoti, kaip parametrą. Funkcijai paduodamo parametro objektas pateiktas pavyzdyje nr.1. Uždavinį galima spręsti išrenkant visas reikšmes iš intervalų ir darant atskiras simuliacijas su gautais reikšmių rinkiniais. Intervalo padalinimui į aktualias reikšmes galime naudoti R funkciją seq (žr. pavyzdys nr. 2 intervalo išskaidymas reikšmėmis)

Pavyzdys nr. 1: R funkcijai paduodamas objektas

```
molBioObject <- list(
  timeStep = 0.005,
  simulationBox = c(20, 20),
  stepAvg = 100,
  totalSteps = 10000,
  densityStep = 0.001,
  energyDriftStep = 0.003,
  temperatureStep = 0.05,
  paramSetSize = 10,
  densityInterval = c(0.008, 0.10),
  energyDriftInterval = c(0.001, 0.01),
  temperatureIntarval = c(1, 1.18)
)
```

Pavyzdys nr. 2: intervalo išskaidymas reikšmėmis

```
densityValues <-
  seq(molObject$densityInterval[1],
      molObject$densityInterval[2],
      by = molObject$densityStep
  );

energyDriftValues <-
  seq(
    molObject$energyDriftInterval[1],
    molObject$energyDriftInterval[2],
    by = molObject$energyDriftStep
  );

temperatureValues <-
  seq(
    molObject$temperatureIntarval[1],
    molObject$temperatureIntarval[2],
    by = molObject$temperatureStep
  );
```

Tam, kad gautume visas galimas reikšmes su kuriomis, vykdant simuliacijas, po 10 000 tūkstančių iteracijų, sistema būtų pusiausvyros būsenoje, turime atlikti simuliacijas su visomis įmanomomis reikšmėmis iš intervalų. Pastaba: tai, kad sistema yra pusiausvyros būsenoje galime spręsti iš sistemą sudarančių objektų greičių sumos (maino algoritmo atveju tai vcSum išvestis). Jei sistemos objektų greičių pokytis, vykstant sistemos evoliucijai, kinta daugiau nei 0.00001, darome prielaidą, kad sistema nebėra pusiausvyros būsenoje. Bendra funkcija pusiausvyros sąlygoms nustatyti (žr. pavyzdys nr.3 pusiausvyros sąlygoms aptikti).

Pavyzdys nr. 3: pusiausvyros sąlygoms aptikti

```
function(results) {
  if ((mean(results$vcSum) < 0.00001) &&
      (mean(results$totEnergySumAvgSquare) < 0.00001)) {
    return (TRUE);
  }
  return (FALSE);
}
```

Atlikus visas galimas iteracijas gauname aibę parametrų, kurie buvo pritaikyti uždavinyje apibrėžtoms sistemos simuliuoti. Klasei *valid* priklausantys paremtrų rinkiniai tenkina užduotyje apibrėžtas sąlygas, klasei *invalid* ptiklauso tie parametrų rinkinai, kurie netenkina užduotyje apibrėžtų sąlygų. Pilną rezultatų rinkinį galima peržiūrėti disko faile Audrius_Jakutis_2016_magistrinis_darbas / PRIEDAI / R_OUTPUT_NR_1.txt. Pilnas R algoritmas šiam tyrimui atlikti fali būti iškviečiamas R aplinkoje (*generateParamsForMaino()*)

timeStep	density	energyDriftTreshold	temperature	class
0.005	0.008	0.001	1	valid
0.005	0.008	0.001	1.05	valid
0.005	0.008	0.001	1.1	valid
0.005	0.088	0.001	1.1	invalid
0.005	0.088	0.001	1.15	invalid
0.005	0.088	0.004	1	invalid
0.005	0.088	0.004	1.05	invalid

4 lentelė.: Gautų parametrų rinkiniai (iškarpa iš R_OUTPUT_NR_1.txt)