Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Connexió de CoDaPack amb R

Document: Memòria

Alumne: Daniel Re Lartigue

Tutor: Santiago Thio Fernandez de Henestrosa
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Estadística i Investigació Operativa

Convocatòria (mes/any) Juny/2020

Escola Politècnica Superior

Treball de final de Grau

Connexió de CoDaPack amb R

Daniel Re Lartigue

Índex

Índex de figures

Índex de taules

1 Introducció

Aquest treball de final de grau sorgeix de les pràctiques que vaig realitzar a l'estiu del 2018 i la col·laboració posteriorment amb contractes de recerca, juntament amb el Grup de recerca en Estadística i Anàlisi de Dades Composicionals (GR-EADC), fent ús de l'aplicació desenvolupada pel grup, el CoDaPack.

Durant els darrers anys, s'ha desenvolupat una nova visió metodològica per l'anàlisi estadística de dades composicionals, després del plantejament introduït a principis dels anys vuitanta per John Aitchison. El Grup de Recerca de Dades Composicionals de Girona són els seus deixebles i actualment líders en el món en aquest camp. Aquesta metodologia no és senzilla per utilitzar-la amb paquets estadístics estàndard.

És per això que va ser desenvolupat el CoDaPack Figura 1 (Copositional Data Package), un software gratuït, independent i multiplataforma que implementa en aquest moment el més elemental del mètodes estadístics anomenats anteriorment. Aquest software està orientat a usuaris sense una amplia experiència en l'ús de diversos paquets informàtics **[1]**. Les funcions principals d'aquest software són les següents:

- Transformacions entre l'espai real al simplex o viceversa com les transformacions: alr, clr i ilr.
- Operacions dins el simplex com: centering, perturbation, power transformation, amalgamation, subcomposition (closure) o el rounded zero replacement.
- Sortides gràfiques en 2-D i 3-D com: diagrames ternaris, diagrames alr, diagrames clr, biplots o diagrames de components principals.
- Estadístiques descriptives de la composició.

Per altra banda trobaríem les característiques del software serien les següents:

- És un software userfriendly.
- Funciona bàsicament via menús.
- Sense necessitats de programació.
- Podem obtenir sortides gràfiques 3D interactives.

Figura 1: Logo del CoDaPack.

Un cop feta una petita introducció del CoDaPack, ara també cal explicar que també hi ha llibreries CODA per a R però que no tenen connexió amb el CoDaPack, ja que està fet amb JAVA, d'aquí és on sorgeix la necessitat de recerca per tal de buscar la manera de connectar R amb JAVA i poder incloure les llibreries al CoDaPack.

Això actualment és una pràctica que s'està estenent bastant, podem trobar-ne un exemple amb l'aplicació RapidMiner Figura 2, un programa per l'anàlisi i mineria de dades.

Figura 2: Logo de RapidMiner.

Aquest treball per tant intenta resoldre aquesta necessitat i fusionar els dos recursos que tenim en un mateix d'una manera transparent i senzilla per l'usuari de l'aplicació.

1.1 Motivacions

El desenvolupament d'una aplicació en JAVA és una matèria d'estudi al llarg del Grau d'Enginyeria informàtica, s'imparteix en l'assignatura de Projecte de Programació (3105G07011), on alumnes han de desenvolupar un projecte en grups de 2-3 persones amb el llenguatge JAVA.

Juntament també trobem en el segon curs del grau l'assignatura d'estadística (3105G07007), on durant el llarg del curs s'han de presentar un seguit de pràctiques fetes amb el llenguatge de programació R que ens permet treballar i estudiar un conjunt de dades.

A l'haver cursat aquestes assignatures, de la primera em va agradar molt programar en JAVA en ser un llenguatge que no coneixia, ja que fins al moment en el grau només havíem programat en el llenguatge C++, em va semblar un llenguatge més ben documentat, més utilitzat en el món de l'informàtica i més flexible i complet que el C++, per tant volia aprofundir encara més el llenguatge JAVA.

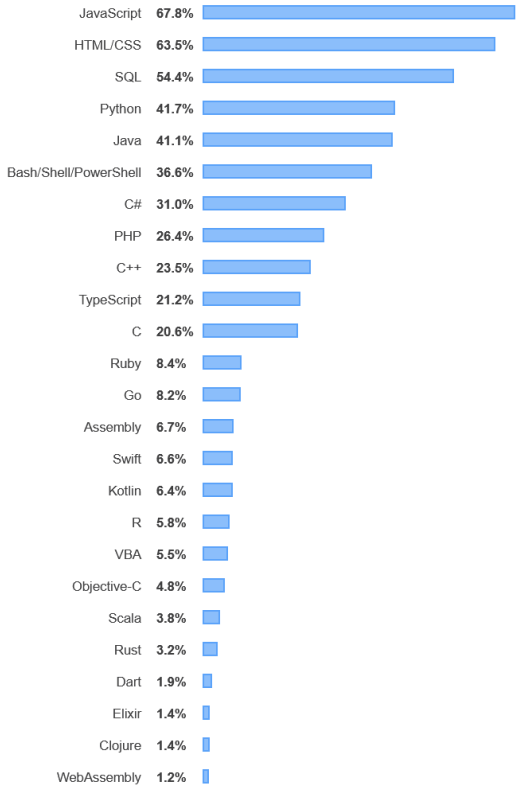


Figura 3: Tecnologies més populars stackoverflow 2019.

Com podem observar en la Figura 3, en les tecnologies més populars de 2019 en StackOverflow es troben tant JAVA en cinquena posició amb un 41,1% com el llenguatge R amb un 5,8%.

Per altra banda l'assignatura d'estadística el que més em va cridar és el llenguatge R, primer de tot no en sabia de l'existència d'aquest i segon no tenia coneixements de llenguatges especialitzats en el tractament de dades. Actualment el tractament de dades és un aspecte molt important en el món de la informàtica i també és molt important en el món empresarial com podem veure a continuació en la Figura 4.

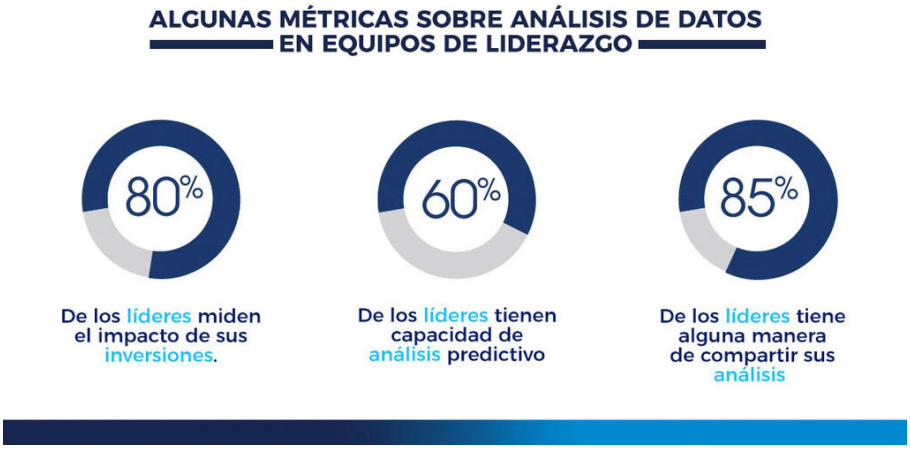


Figura 4: Mètriques sobre l'anàlisi de dades.

En un futur fins i tot m'agradaria seguir amb estudis que s'especialitzin en l'anàlisi i tractament de dades a causa de l'esmentat anteriorment.

Per tant en conclusió en veure que hi havia un projecte en el qual es combinaven els dos aspectes comentats anteriorment i a més era una connexió entre ells, no vaig dubtar en triar-ho per al meu projecte de final de grau. En ser un projecte amb un bon equilibri de recerca i gestió em va cridar l'atenció, ja que em considero una persona molt curiosa i que m'agrada indagar en els aspectes que trobo interessants.

1.2 Propòsit

El propòsit d'aquest projecte és aprofundir en el llenguatge JAVA, estudiant el programa base que se'ns ofereix en el projecte que és el CoDaPack, veure com està programat, les rutines que té i com està estructurat el programa per partir del mateix patró estructuralment.

També el que es vol és estudiar les llibreries que hi ha per incloure en els projectes de JAVA i aconseguir amb la utilització d'elles, incorporar R en el nostre programa CoDaPack.

Un cop hi hagi connexió, establir un protocol de comunicació entre JAVA i R, per tant també s'haurà d'estudiar les llibreries d'R així doncs com el mateix llenguatge de programació.

També es vol crear una interfície senzilla i genèrica per la utilització d'R en el CoDaPack.

Finalment també volem que el CoDaPack segueixi sent un programa senzill, transparent i multiplataforma, per tant caldran uns instal·ladors tant per Windows com per MacOs com a mínim i que siguin fàcils d'instal·lar per l'usuari.

La idea del producte final en conclusió és que ha de ser un producte tancat i transparent a l'usuari de tal manera que ell només s'hagi d'instal·lar el CoDaPack, sense necessitat d'haver d'instal·lar també R.

1.3 Objectius del projecte

L'objectiu principal d'aquest treball és analitzar, (dissenyar) i implementar una llibreria amb JAVA que permeti utilitzar les funcionalitats de les llibreries que ja tenim de CODA fetes amb R mitjançant menús del CoDaPack i de forma totalment transparent per l'usuari.

Un cop fet això també està l'objectiu de crear una interfície genèrica per tal que usuaris de l'aplicació puguin crear les seves pròpies rutines i executar-les amb el CoDaPack igual que fem amb les llibreries de CODA.

Per tal d'aconseguir aquests objectius s'haurà d'aconseguir fer les següents fites:

- Estudiar com obrir R i carregar paquets de forma automàtica des de JAVA.
- Implementar rutines de JAVA específiques que facin crides a les rutines CODA d'R.
- Incorporar aquestes rutines al programa CoDaPack.
- Establir una interfície que permeti als usuaris programar les seves pròpies rutines amb R i que les puguin executar de forma transparent en el CoDaPack.
- Fer un instal·lador integrat.

2 Estudi de viabilitat

A continuació es fa un estudi de viabilitat del projecte, per realitzar aquesta anàlisi, el dividirem en uns quants apartats, primer de tot veurem quins coneixements previs hauríem de tenir com a mínim per poder posar en marxa el projecte, seguidament els recursos humans i els recursos tecnològics, finalment en conclusió farem un estudi dels costos per veure quin hauria de ser el pressupost aproximat del projecte.

2.1 Coneixements previs

Aquest projecte requereix uns mínims de coneixements previs per poder procedir a realitzar-se, a continuació un petit llistat d'aquests:

- Enginyeria del software.
- Programació orientada a objectes amb el llenguatge JAVA.
- Nocions de programació amb el llenguatge R.
- Instal·lació i configuració de màquines virtuals.
- Nocions de programació de pàgines web amb HTML, CSS i JavaScript.
- Mínims en manipulació d'imatges.

2.2 Recursos humans

Pel bon desenvolupament d'aquest projecte, en el que s'ha de fer una recerca per connectar R amb el programa CoDaPack, hem estimat que el període de temps és aproximadament de 1152 hores (1080 hores per part del programador i unes 72 hores de direcció).

Com a recursos humans en principi necessitem el següent:

- **Programador Junior**: Aquest programador haurà de realitzar les 1080 hores aproximades en el projecte. Aquestes hores equivalen a nou mesos de feina treballant 30 hores setmanals.
- **Assessors del grup de recerca (GR-EADC)**: Aquests assessors el que faran és regular la feina i resoldre dubtes en l'àmbit més estadístic. Tindran una feina de dues hores setmanals també durant un període de nou mesos.

En principi amb aquests dos apartats en tenim prou com a recursos humans.

2.3 Recursos tecnològics

Seguidament trobem els recursos tecnològics necessaris per dur a terme el projecte:

- Per al desenvolupament del codi:
  - Un PC amb sistema operatiu Windows on instal·larem les eines necessàries per programar i desenvolupar el projecte.
  - Un PC amb sistema operatiu MacOs on també provarem el nostre projecte.
  - Una màquina virtual de Windows on poder desenvolupar el projecte sense comprometre la màquina real i així poder testejar amb tranquil·litat i comoditat.
  - Un entorn de desenvolupament integrat per al llenguatge JAVA.
  - Necessitarem instal·lar R que és l'entorn per la computació estadística i gràfica.
  - Un entorn de desenvolupament integrat per al llenguatge R que en aquest cas es tracta de RStudio.
  - Per últim un editor de text i codi font, per poder treballar amb diferents llenguatges.
- Eines per al desenvolupament del projecte:
  - Una eina pel control de versions del nostre projecte.
  - Una interfície més amigable per Git, amb integració a Github.
  - Una eina per l'administració del projecte per tal d'organitzar i planificar el nostre TFG.
  - Un programa per poder realitzar els executables per Windows.
  - Un programa que ens permeti poder realitzar els instal·ladors pel sistema operatiu Windows.
  - Un programa de manipulació i edició d'imatges.
- Per la realització de la documentació del projecte:
  - Un processador de textos per poder redactar la documentació del TFG.
  - Un visualitzador d'arxius PDF.

2.4 Costos

En aquest apartat descriurem els costos necessaris per poder realitzar el nostre projecte, cal dir que aquests costos no són exactes del tot per tant en farem una aproximació.

Dividirem els costos en dos apartats que veurem seguidament, un primer apartat que són els costs fixos i un altre apartat que són els costos variables.

2.4.1 Costos fixos

Els costos fixos del projecte són aquells tipus de costos que no varien en un període de temps curt i que són independents a la productivitat del projecte, generalment aquest tipus de cost no pot ser evitat.

Dividirem els costos fixos del projecte en dues taules, una taula representarà els costos fixos del hardware i l'altra taula representarà els costos fixos corresponents al software.
Seguidament en la Taula 1 podem veure els costos fixos referents només al hardware.

Nom	Descripció	Cost
PC Windows	PC amb SO Windows per instal·lar els programes necessaris.	500 €
PC MacOS	PC amb SO MacOs per provar CoDaPack multiplataforma.	729 €
Cost total		**1229 €**

Taula 1: Costos fixos de hardware

Un cop vistos els costos fixos de hardware ara passarem a veure els costos fixos de software, a la Taula 2 trobarem tots els elements d'aquests costos fixos de software, seguidament els anem a resumir un per un:

- **Màquina Virtual Windows**: Per muntar la màquina virtual de Windows, farem servir el programa VirtualBox de l'empresa Oracle, aquest software és un virtualitzador que ens permet muntar SO amb imatges ISO i poder-les simular en un entorn simulat amb part del nostre hardware real. Això ens permetrà muntar el SO Windows simulat per poder anar desenvolupant el projecte sense por de fer malbé alguna cosa a la màquina real. És un software de codi obert i la seva descarrega és gratuïta.
- **IDE per JAVA**: Per programar en JAVA utilitzarem un IDE que s'anomena NetBeans, és un entorn de desenvolupament integrat lliure, fet principalment per la programació en codi JAVA, és un software que a més a més inclou molts mòduls per estendre'l. Aquest software és lliure i gratuït i sense cap restricció d'ús.
- **R**: R és un entorn de llenguatge de programació que està orientat a l'anàlisi estadístic. R té una llicència GNU GPL, però el podem descarregar de forma gratuïta en el seu portal oficial.
- **IDE per R**: Per provar scripts i programar en R, utilitzarem RStudio, un entorn de desenvolupament per R, que està dedicat òbviament a la computació estadística i gràfica, aquest software inclou: una consola, un editor de sintaxi, eines de depuració de codi... En el nostre cas utilitzarem més concretament RStudio Desktop Open Source License, ja que és la versió gratuïta del software i amb aquesta ja en tenim prou per al desenvolupament del nostre projecte.
- **Editor de text i codi**: Per editar qualsevol text en qualsevol format d'una manera rapida i senzilla, utilitzarem Sublime Text, un editor de text i de codi font que té una llicència propietària però es pot descarregar de forma gratuïta.
- **Control de versions**: Per poder portar un registre del projecte pel que fa a codi, no hi ha millor eina que un control de versions com és Git. Aquest software ens permet guardar estats del nostre projecte en tot moment. El podem descarregar gratuïtament.

- **Client GUI de Git**: Per poder realitzar accions amb Git una mica més fàcilment ho podem fer gràcies a un client GUI de Git com és GitKraken, aquest software ens permet fer les tasques de Git de manera gràfica. El podem descarregar i fer servir gratuïtament amb un compte d'organització.
- **Planificador projecte**: Per portar al dia el nostre projecte hem de planificar i gestionar el projecte d'alguna manera, Trello és un software gratuït que ens permet fer aquestes tasques d'administració de projectes amb interfície Web i amb clients tant per Android com per iOS.
- **Software per fer executables de Windows**: Haurem de crear executables per Windows, Launch4j és una eina multiplataforma que ens permet empaquetar aplicacions JAVA i crear-ne exes, la podem trobar de forma gratuïta.
- **Software per realitzar instal·ladors per Windows**: En el nostre projecte haurem de crear instal·ladors per Windows, per tant necessitarem una eina, aquesta eina serà Inno Setup Compiler. Inno Setup Compiler és un sistema d'instal·lació basat en scripts de software lliure, el podem descarregar per al seu ús de manera gratuïta.
- **Software manipulació d'imatges**: Per manipular imatges, utilitzarem GIMP. GIMP és un programa d'edició d'imatges digitals en forma de mapa de bits. És lliure, gratuït i multiplataforma.
- **Processador de textos**: Com a processador de textos fem servir LibreOffice com a alterativa a altres processadors que no són gratuïts com Microsoft Office. LibreOffice és un paquet de software lliure i de codi obert per tant és gratuït.
- **Visualitzador PDF**: Per poder visualitzar arxius PDF necessitem un visualitzador, el que utilitzarem és Adobe Acrobat Reader DC, es pot descarregar gratuïtament.

Nom	Descripció	Cost
Màquina Virtual Windows	Màquina virtual VirtualBox amb un SO Windows.	0 €
NetBeans	IDE per programar en JAVA, producte lliure i gratuït.	0 €
R	Entorn i llenguatge de programació estadístic.	0 €
RStudio	IDE per programar amb R d'una manera més amena.	0 €
Sublime Text	Editor de text i codi font.	0 €
Git	Software de control de versions.	0 €
GitKraken	Client GUI de Git per simplificar l'ús de Git.	0 €
Trello	Software d'administració de projectes.	0 €
Launch4j	Eina per la creació de .exe a partir de jar.	0 €
Inno Setup Compiler	Eina que ens permetrà crear instal·ladors per Windows.	0 €
GIMP	Programa d'edició d'imatges, lliure i gratuït.	0 €
Adobe Acrobat Reader	Visualitzador d'arxius PDF.	0 €
LibreOffice	Paquet de software d'oficina lliure i de codi obert.	0 €
Cost total		**0 €**

Taula 2: Costos fixos del software

Un cop vista la Taula 2 podem observar que els costos totals fixos de software són nuls, per tant no ens haurem de gastar diners en el software, ja que tot el que fem servir és open source o ho tenim de forma gratuïta gràcies a comptes d'organització.

2.4.2 Costos variables

Els costos variables són aquells costos que varien en funció de producció, en el nostre cas, si ho mirem en el projecte, seria el temps que triguem i volem invertir en el projecte, p.ex no són els mateixos costos treballar en el projecte durant tres mesos que treballar-hi sis mesos, ja que els treballadors hauran de cobrar més mesos de feina.

En el nostre projecte els costos variables fan referència principalment als recursos humans que són els següents:

- **Programador Junior**: Estimem que el salari del programador Junior és d'uns 1500 € bruts mensuals, a part d'això també hem de pagar un 30% de seguretat social (450 €). Per tant, mensualment ens surt un total de 1950 €.
- **Assessors del grup de recerca (GR-EADC)**: Estimem que el salari d'un assessor del grup de recerca és aproximadament d'uns 4000 € bruts mensuals, com que només han de treballar dues hores setmanals, això equival a 200 € bruts mensuals. A part d'això també hem de pagar el 30% de seguretat social (60 €). Per tant, mensualment ens surt un total de 260 €.

Seguidament la Taula 3 amb els costos variables (recursos humans) multiplicats per la quantitat de mesos treballats i pel nombre de treballadors, obtinguin així el total de costos variable del projecte.

Nom	nº de mesos	nº de treballadors	Cost mensual	Cost total
Programador Junior	9	1	1950€/mes	17550 €
Assessors GR-EADC	9	2	260€/mes	4680 €
Cost total				**22230 €**

Taula 3: Costos variables (recursos humans)

2.4.3 Cost total del projecte

Finalment fent la suma tant del total dels costos fixos com el total de costos variables, com es mostra a continuació a la Taula 4 obtenim el cost tota del projecte.

Costos fixos	1229 €
Costos variables	22230 €
Cost total	**23459 €**

Taula 4: Cost total del projecte

Aquests costos han estat finançats per la Universitat de Girona, gràcies a beques de recerca per part del Ministeri.

3 Metodologia

La metodologia que s'ha dut a terme durant el desenvolupament del projecte de fi de grau, és una metodologia àgil, hi ha múltiples metodologies àgils pel desenvolupament de projectes de software, però la que més s'assembla a la utilitzada en el nostre cas és SCRUM.

Seguidament passarem a explicar en què consisteix una metodologia àgil, després ens centrarem justament en una d'elles que és SCRUM la més semblant a la que hem utilitzat. Finalment expondrem més específicament la utilització d'aquesta en el projecte.

3.1 Metodologia àgil

Per definició les metodologies àgils, són aquelles que permeten adaptar la forma de treballar a les condicions que requereix el projecte, aconseguint així més flexibilitat i feedback per adaptar el projecte i el seu desenvolupament a circumstàncies específiques de l'entorn.

Així doncs una metodologia àgil permet a un equip donar respostes ràpides a les valoracions que es rep del projecte. Crea oportunitats d'avaluar el projecte durant el seu cicle de desenvolupament, aquestes avaluacions del projecte es porten a terme en reunions regulars anomenades sprints o iteracions. Això aporta a les empreses la capacitat de fabricar un producte valuós, de manera que mantingui competitivitat amb altres empreses en el sector **[2]**.

Seguidament podem veure en la Figura 5 el procediment que segueix aquesta metodologia àgil.

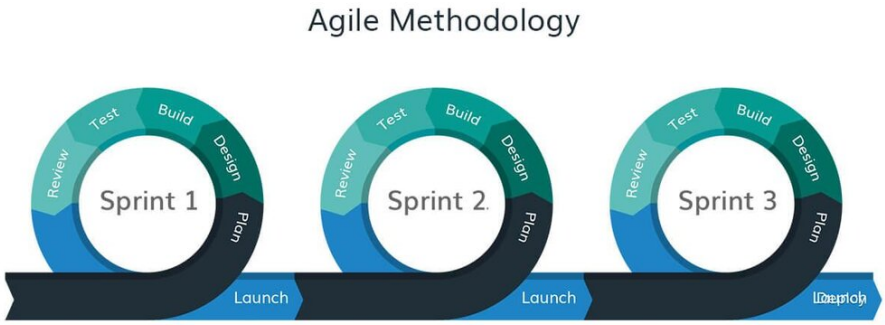


Figura 5: Metodologia àgil.

La definició moderna de metodologia àgil va aparèixer a mitjans de la dècada de 1990 com una reacció contra els mètodes de "pes pesat", molt estructurats i estrictes, del model de desenvolupament en cascada. L'any 2001, disset desenvolupadors de programes informàtics es van reunir a Utah per examinar mètodes de desenvolupament lleugers, junts van publicar el *Manifest pel Desenvolupament Àgil de Software*. Aquest manifest conté quatre valoracions importants:

- El focus s'ha de posar més sobre les persones i les interaccions que sobre els processos i eines.
- El software funcionant és més important que la documentació excessiva.
- La col·laboració amb el client ha de ser més important que la negociació contractual.
- El procés hauria de respondre davant el canvi, en lloc de seguir un pla.

Podem trobar molts beneficis en el desenvolupament amb una metodologia àgil de software, alguns d'aquests beneficis són els següents:

- **Transparència:** El mètode àgil involucra al client, inclòs en el plantejament de la iteració, les revisions i l'anunci de noves característiques en el software. Els clients igualment han d'entendre que encara que el projecte sigui transparent, el que estan veient és un treball en curs i no un producte final com a tal.
- **Rapida entrega i predictible:** Els sprints o iteracions es porten a terme en un horari fix durant 1 a 4 setmanes. Gràcies a aquest mètode les entregues del software són predictibles, ja que noves característiques poden ser entregades als clients d'una manera ràpida i freqüent. També això ajuda a poder testejar i desplegar abans el software si l'empresa en té els recursos suficients.
- **Es permet el canvi:** Encara que l'objectiu hauria de ser entregar el subconjunt acordat de funcions del producte, els processos àgils donen l'oportunitat de canviar prioritats i refinar el backlog del producte. Aquests canvis poden ser introduïts d'una manera relativament ràpida en poques setmanes, per exemple en el següent sprint.
- **Es millora la qualitat:** El projecte es divideix en unitats, així permet centrar-se en el desenvolupament de qualitat, en un bon testeig i en la col·laboració. En crear parts del producte final es poden realitzar proves i revisions al llarg de les diferents iteracions, és més fàcil trobar errors i defectes en el producte i solucionar-los com més aviat millor, fen així una millora general pel que fa a la qualitat del producte final.

En la Figura 6 podem veure un estudi comparant mètodes de desenvolupament tradicionals amb mètodes àgils, veient així els beneficis i avantatges que té.

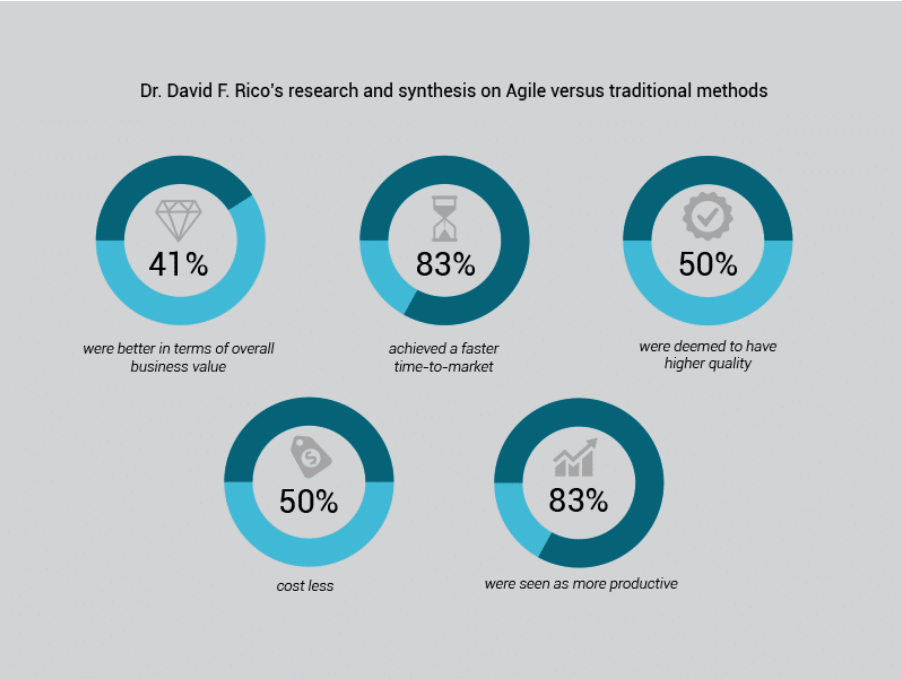


Figura 6: Metodologia àgil vs tradicional.

Finalment hi ha moltes metodologies àgils, però totes elles comparteixen la mateixa filosofia i característiques. Però en ser implementades cada metodologia té les seves pròpies pràctiques i tàctiques. Algunes de les principals metodologies de desenvolupament àgil de software són: Scrum, Kanban, Programació Extrema (XP), Crystal, mètode de desenvolupament de sistemes dinàmics (DSDM), desenvolupament basat en funcionalitats (FDD)...

3.2 Metodologia SCRUM

Com hem comentat anteriorment SCRUM és un marc de treball pel desenvolupament àgil de software.

És un procés en el qual s'apliquen d'una manera regular un conjunt de bones pràctiques per treballar de manera col·lectiva i obtenir així el millor resultat possible pel projecte. Aquestes pràctiques es recolzen les unes a les altres i tenen el seu origen en formes de treball d'equips altament productius **[3]**. Aquestes pràctiques estan caracteritzades per:

- Utilitzar un desenvolupament incremental en comptes de desenvolupar completament el producte final.
- Basar la qualitat del resultat en el coneixement tàcit de les persones en equips autoorganitzats en comptes de basar-la en els processos empleats.
- Superposar les fases del desenvolupament en comptes de realitzar-les seqüencialment.

A continuació veurem les principals característiques de la metodologia àgil SCRUM:

- Gestió regular del que el client espera obtenir, resultats ràpids, flexibilitat i adaptació, productivitat i qualitat, alineament entre el client i l'equip i motivació.
- Es fa ús d'equips autodirigits i autoorganitzats.
- Es fa reunions diàries que no duren més de quinze minuts per obtenir informació tant de les tasques de l'equip com dels possibles obstacles que sorgeixin.

En la metodologia SCRUM tenim el que s'anomenen rols (responsabilitats definides), és molt important per la implementació exitosa del projecte tenir ben clar els tres rols principals que hi ha.

SCRUM està format per tres rols principals: El *Product Owner* (l'amo del producte), el *Scrum Master* (l'amo del procés) i el *Team* (membres de l'equip de desenvolupament).

- **Product Owner:** Assegura que l'equip SRUM treballi correctament en la perspectiva de negoci. El *Product Owner* ajuda a l'usuari a escriure les històries a prioritzar-les i col·locar-les en el backlog.
- **Scrum Master:** La seva tasca primària consisteix a facilitar els possibles problemes que impedeixin aconseguir l'objectiu de l'sprint per part de l'equip. Actua com una protecció entre l'equip i les possibles distraccions. S'assegura que el procés SCRUM és realitzat correctament i que les regles es facin complir per part de l'equip.
- **Team:** Té bàsicament la tasca i responsabilitat d'entregar el producte, normalment el componen equips petits d'entre 3 a 9 persones, amb les habilitats per realitzar el producte (anàlisi, desenvolupament, disseny, testing, documentació ...).

També trobem un flux de treball en SCRUM, aquest flux de treball s'anomena Sprint. Els sprints normalment són d'una duració constant i ben definida per l'equip segons l'experiència que tinguin. Normalment aquesta duració sol ser de mínim dues setmanes i màxim quatre setmanes. En el sprint hi apareixen diferents etapes.

Primer trobem la planificació de l'sprint on es reuneix l'equip per decidir quina és la feina que es farà a l'sprint, els objectius que se n'espera obtenir i repartir la feina als components de l'equip. Seguidament també hi ha scrums diaris molt curts per estar en constant feedback amb l'estat actual del projecte. Després tenim La revisió de l'sprint, on quan ja s'ha acabat l'sprint l'equip presenta la feina feta durant aquest i es valora la feina feta per realitzar una millora continua del procés.

A continuació en la Figura 7 podem veure el procés i metodologia d'SCRUM.

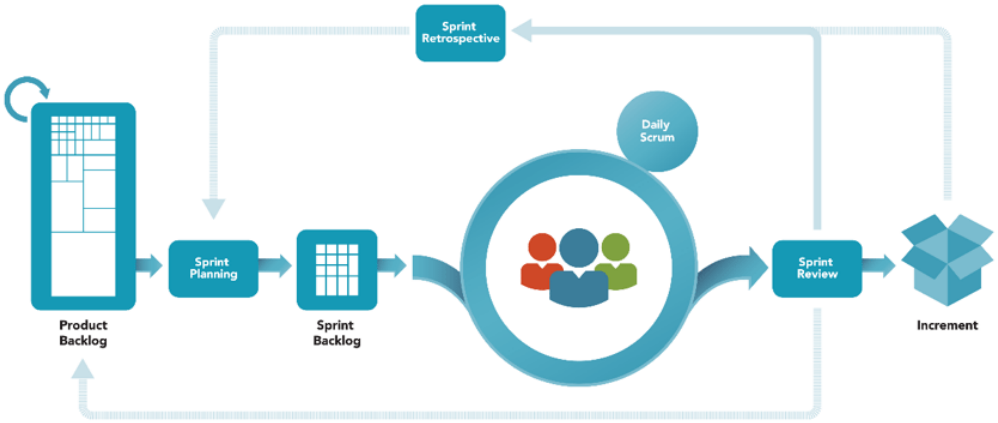


Figura 7: Procés i metodologia SCRUM.

Finalment comentar alguns dels beneficis d'utilitzar aquesta metodologia àgil:

- **Predicció de temps:** Amb el flux de treball comentat, es coneix la velocitat mitjana de l'equip per sprint, per tant és més fàcil poder estimar el temps que caldrà en fer una funcionalitat que encara està en el backlog.
- **Reducció del Time To Market:** El client del producte pot començar a utilitzar les funcionalitats principals i més importants abans que el producte estigui acabat del tot.
- **Reducció de riscos:** El fet de desenvolupar les funcionalitats amb més valor i saber la velocitat de treball de l'equip segons l'sprint, permet eliminar riscos de manera anticipada.

3.3 Implementació en el projecte

En el cas de la metodologia triada en el projecte s'ha triat fer servir una metodologia àgil basada en SCRUM, s'ha utilitzat aquesta deguda a què l'equip de desenvolupament és bastant reduït (un desenvolupador), més els assessors del grup de recerca que d'alguna manera feien el rol principal de client del producte o *Product Owner* a l'haver contractat i triat el desenvolupador per realitzar les tasques del projecte i així doncs controlar també la feina desenvolupada a cada un dels sprints o iteracions.

Aquests sprints eren realitzats setmanalment on quedàvem normalment el desenvolupador amb els assessors per comentar la feina feta durant aquell sprint, això incloïa: mostrar la feina feta, comentar possibles modificacions o millores, comentar alguns dubtes referents a algun aspecte en concret...

A l'inici del projecte es van definir unes funcionalitats principals i bàsiques que s'havien de realitzar, juntament amb tasques complementàries. Seguidament un cop acabada aquesta etapa es van definir unes tasques més enfocades en millora o crear de noves funcionalitats, correcció d'errors i generalitat a l'hora d'utilitzar el software. Aquestes dues etapes van ser organitzades en els sprints comentats anteriorment.

Notar també que el projecte ha estat mostrat i compartit amb altres membres del grup de recerca per d'alguna manera fer ells també de clients del producte per poder reportar un feedback, reportar errors o simplement suggerir canvis o millores. Finalment comentar que el projecte ha estat mostrat en el CoDaWork2019, un Workshop d'anàlisi de dades composicionals que es va realitzar a Terrassa.

4 Planificació

En aquesta etapa procedirem a mostrar l'estratègia duta a terme al llarg del projecte per poder arribar als objectius que s'han plantejat. En primer lloc dividirem la planificació en dues parts diferenciades, i seguidament mostrarem dos diagrames que correspondran a cada una de les etapes amb les tasques que s'han planificat i el temps estimat per cada una d'aquestes tasques, òbviament això estarà també relacionat amb el punt anterior (metodologia).

- **Etapa 1:** En aquesta etapa trobem les funcionalitats més importants del projecte, principalment correspon a la connexió entre el CoDaPack (Java) i R per poder executar scripts d'R en Java. Un cop fet això també cal establir un protocol de comunicació entre aquests dos, com s'envien i com reben dades l'un amb l'altre, això inclou dades numèriques, textuals i fins i tot sortides gràfiques. Aquesta seria la funcionalitat més important de l'etapa 1 i la principal.

  A part d'això també trobem rutines que cal implementar amb la connexió ja establerta, com poden ser per exemple la rutina ZPatters, o simplement fer una ordenació d'un conjunt de dades fent servir un criteri (sort).

  Finalment també trobem millores i noves funcionalitats implementades únicament utilitzant el mateix llenguatge Java, sense haver de fer una connexió amb R com per exemple la millora del panell de sortides de missatges del CoDaPack, o la creació d'un menú per la selecció de dos conjunts de variables.

- **Etapa 2:** En l'etapa 2 també trobem funcionalitats importants com per exemple: la creació dels instal·ladors amb R integrat tant per Windows com per MacOS, la creació de menús per desenvolupadors amb funcionalitats genèriques per poder crear scripts amb R i poder rebre els resultats obtinguts i mostrar-los en el CoDaPack, la creació de rutines fetes directament amb R i executades des del CoDaPack, la millora de les sortides gràfiques ara amb resize automàtic gràcies al format SVG, la incorporació de menús de help en cada una de les rutines del CoDaPack i finalment una de les tasques que més temps ha dut a terme que és la correcció de bugs i errors.

  A part d'aquestes tasques també cal comentar que s'han dut a terme algunes altres tasques simplement utilitzant el Java, com correccions o millores en alguns aspectes del CoDaPack i també noves petites funcionalitats com la neteja de la consola de missatges o poder simplement esborrar tots els conjunts de dades que es tenen carregats.

Seguint la metodologia comentada en l'apartat anterior, es va planificar la feina en sprint, on s'havien d'implementar les funcionalitats comentades en períodes aproximats d'entre una a dues setmanes, per poder mostrar la funcionalitat o afegir-hi millores o correccions.

El projecte va donar inici al Juny de l'any 2018 i va acabar en el mes de Març de l'any següent, en el 2019. Seguidament tant en la Figura 8 com en la Figura 9 podem veure la planificació de les tasques i el temps aproximat que s'ha consumit en implementar-les, respectivament podem observar la planificació tant de l'etapa 1 com de l'etapa 2.

En aquest apartat finalment també es vol tenir en compte la redacció de la memòria que ha estat d'aproximadament d'uns dos mesos.

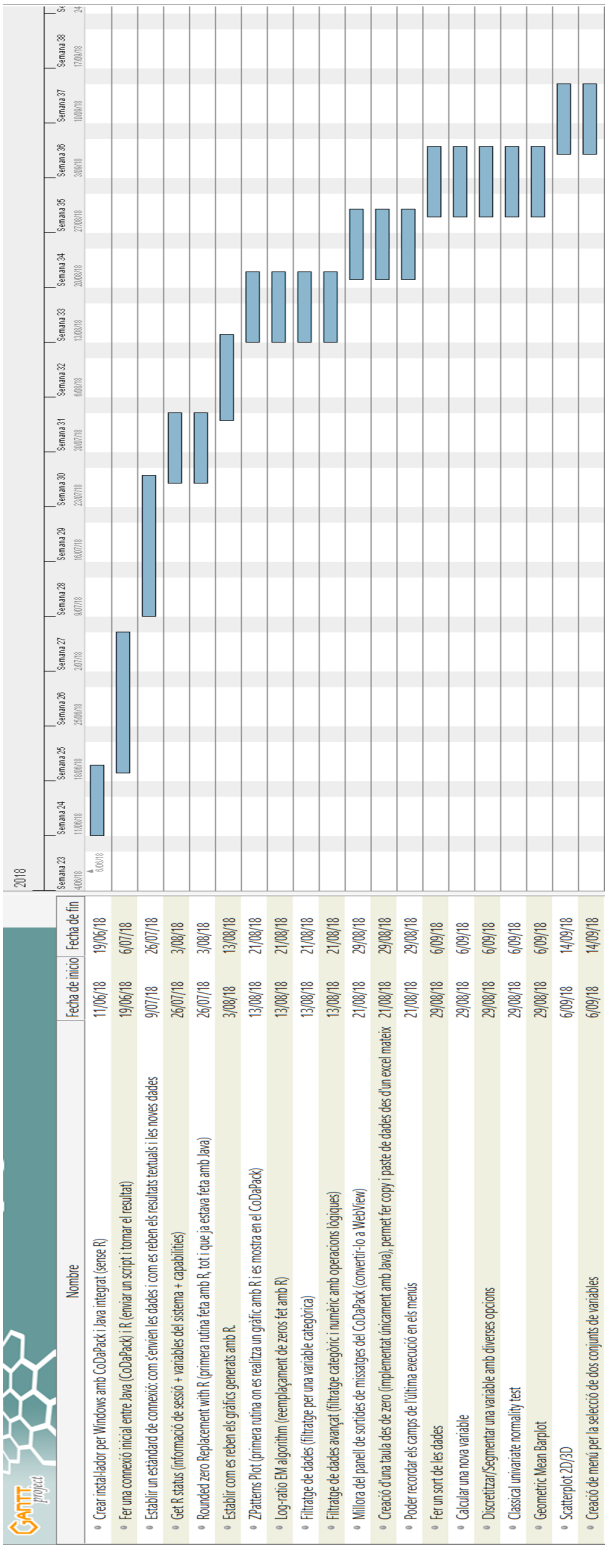


Figura 8: Planificació de l'etapa 1.

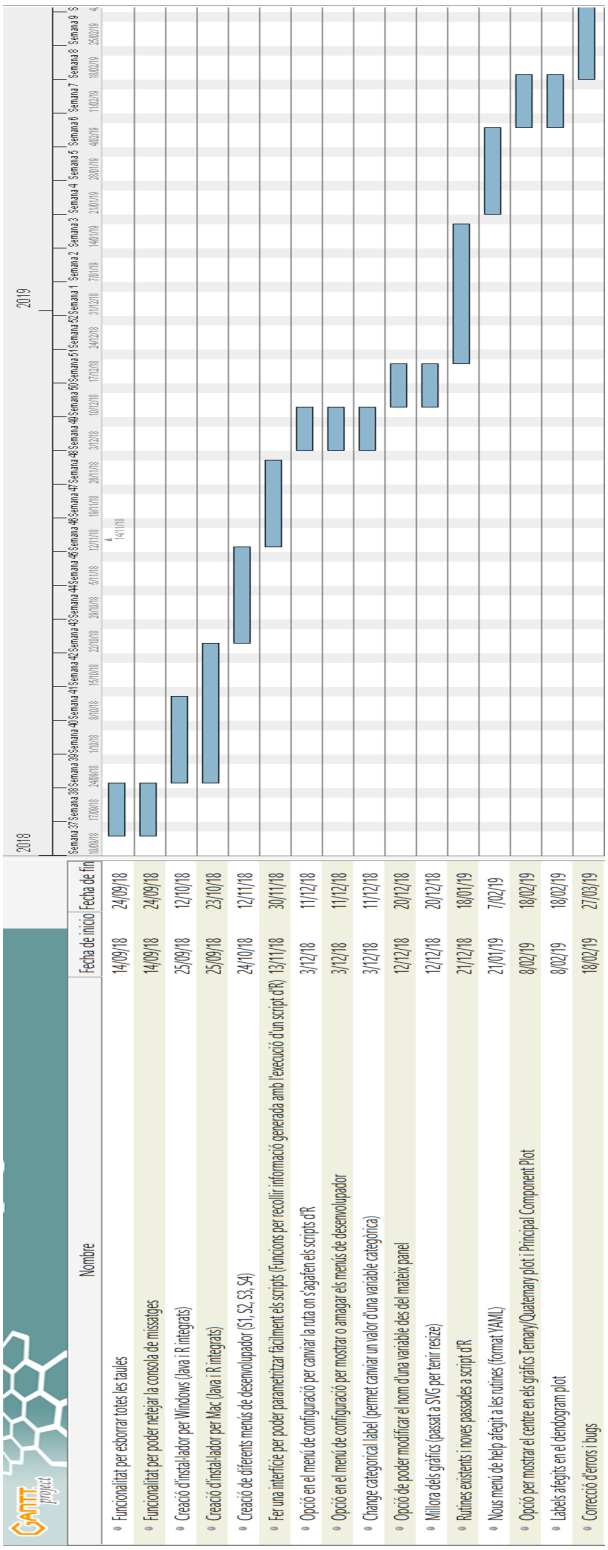


Figura 9: Planificació de l'etapa 2.

5 Marc de treball i conceptes previs

Abans de començar a tractar el projecte en si, s'ha d'estudiar i, després en la documentació, donar a conèixer una sèrie de conceptes necessaris per fer un bon seguiment del projecte.

En la memòria, per tal de situar el lector en la temàtica que engloba al treball, descriurem els diversos aspectes relacionats amb el desenvolupament general del projecte que permetran entendre molt millor els capítols següents.

En aquest apartat mencionarem l'equip de persones que col·laboren o tenen relació amb el desenvolupament del projecte.

5.1 Marc de treball

En aquesta secció procedirem a descriure detalladament el marc de treball del projecte, on trobarem el CoDaPack i en segon lloc el grup de recerca d'Estadística i Anàlisi de Dades Composicionals.

5.1.1 CoDaPack

Des de principis del segle XXI el grup de recerca d'Estadística i Anàlisi de Dades Composicionals ha desenvolupat un software anomenat CoDaPack, aquest conté un conjunt de rutines destinades a usuaris finals sense una gran experiència en informàtica **[4]**.

Mitjançant menús interactius del CoDaPack l'usuari es comunica amb l'aplicatiu i aquest retorna a l'usuari tan sortides numèriques, textuals i també sortides gràfiques. Les sortides gràfiques poden ser en 3D i s'hi poden aplicar tant rotacions com zooms.

Originalment CoDaPack estava associat, mitjançant rutines en VisualBasic, al software Excel de tal manera que s'executava com un menú més d'Excel i els resultats es guardaven també en fulls Excel. Més endavant es van millorar els gràfics i es van programar en OpenGL, encara amb sortida d'Excel.

Des de maig de 2011 hi ha una nova versió del CoDaPack, la versió 2.0, on ja no es depèn de l'Excel. Aquesta versió ja està programada en Java i només requereix tenir instal·lada una màquina virtual de Java (versió mínima 1.5). Això permet que el CoDaPack 2.0 sigui multiplataforma i es pugui executar en qualsevol sistema operatiu que pugui executar màquines virtuals de Java. Més concretament ara els sistemes operatius de MacOS com Unix podran executar CoDaPack 2.0.

Aquest software es va ampliant constantment amb noves rutines, millores i noves funcionalitats.

5.1.2 Grup de recerca d'Estadística i Anàlisi de Dades Composicionals

El Grup de Recerca d'Estadística i Anàlisi de Dades Composicionals el trobem en el Departament d'Informàtica, Matemàtica Aplicada i Estadística de la Universitat de Girona **[5]**.

El principal motiu de cohesió del grup de recerca com el mateix nom indica és l'anàlisi Estadística de les dades composicionals. Aquestes ja bé poden ser de Geologia, Petrologia, Química, Economia, Arqueometria... habitualment es treballa amb vectors de dades les components dels quals representen la contribució relativa de diferents parts amb relació al total, donant lloc a les mostres composicionals.

L'objectiu per tant del grup és el d'avançar en aquest estudi d'anàlisi estadística de les dades composicionals i la seva fonamentació matemàtica. El ventall d'aplicació de les tècniques desenvolupades és molt ampli.

Partint del treball del professor J. Aitchison en aquest camp, des d'un àmbit estrictament estadístic, s'ha vist que la fonamentació matemàtica i rigorosa d'aquestes anàlisis estadístiques es basa en la definició d'una geometria específica sobre el símplex (espai suport de les dades composicionals) a partir de la qual es poden anar desenvolupant amb rigor totes les anàlisis habituals (anàlisi clúster, anàlisi discriminant, anàlisi factorial, models de regressió, etc.).

Tot això fa que la temàtica del grup de recerca no es limiti només al desenvolupament de les tècniques pròpies de l'anàlisi estadística clàssica de dades composicionals, sinó que abordi també els aspectes estrictament matemàtics que fonamenten aquestes tècniques i que pertanyen als àmbits de la geometria, de la teoria de la mesura i del càlcul diferencial i integral sobre el símplex.

La manca d'un software específic i amigable per al tractament de les dades composicionals, ha portat al grup de recerca a implementar des de l'any 2001 aquest software (CoDaPack) i en la posterior incorporació dels nous desenvolupaments matemàtics de la teoria.

Els membres que componen el grup de recerca són els següents:

- Dra. Vera Pawlowsky Glahn
- Dr. Carles Barceló Vidal
- Dr. Santiago Thió Fdez. de Henestrosa
- Dr. Josep Antoni Martín Fernández
- Dr. Pepus Daunis i Estadella
- Dra. Glòria Mateu Figueras
- Dr. Marc Comas Cufí
- Dra. Marina Vives Mestres

5.2 Conceptes previs

En aquest apartat s'intentarà presentar i explicar alguns conceptes previs sobre: les llibreries d'enllaç dinàmic (DLL), conceptes de Java, conceptes d'R i per últim parlarem de les variables d'entorn. Aquests conceptes ens ajudaran a situar-nos en la temàtica del treball i a entendre molt millor els capítols següents.

5.2.1 Llibreries d'enllaç dinàmic (DLL)

Una biblioteca d'enllaç dinàmic, es refereix als arxius de codi executable que es carreguen sota demanda d'un programa per part del sistema operatiu. Aquesta denominació és exclusiva dels sistemes operatius Windows però el terme existeix en la majoria de sistemes operatius actuals **[6]**.

Algunes de les avantatges que ens aporten les DLL són:

- Reducció de la mida dels arxius executables.
- Poden estar compartides en diferents aplicacions.
- Faciliten la gestió i la utilització de la memòria del sistema.
- Ofereixen més flexibilitat davant de possibles canvis.

I alguns dels mals de caps que poden portar a terme són els següents:

- La instal·lació d'un programa pot sobreescriure una DLL amb una nova versió incompatible.
- La desinstal·lació d'un programa pot esborrar una DLL que és utilitzada per altres recursos.

En el cas que es produeixi algun d'aquests errors, el programa que utilitza les DLL es veurà afectat i deixarà de funcionar com ho hauria de fer.

5.2.2 JAVA

A continuació una petita introducció d'alguns conceptes del llenguatge Java i tot el que engloba.

Compilació i execució

Quan el codi en Java és compilat es genera el *bytecode*, un codi intermedi independent de la màquina (no és codi màquina) i es guarda en els fitxers *.class* (un per a cada classe). Per tant el llenguatge Java és un llenguatge interpretat òbviament **[7]**.

La variable d'entorn *PATH* ha d'incloure la ruta del directori *bin* de la instal·lació de la màquina virtual de Java.

La variable d'entorn *CLASSPATH* indica al Java on ha d'anar a buscar els fitxers *.class* que s'ha generat en la compilació (a part del directori actiu).

Packages

Els packages en Java, són un conjunt de classe, aquests s'organitzen jeràrquicament. Aquesta jerarquia queda marcada en forma de directoris (carpetes).

Algunes de les utilitats dels packages en Java poden ser les següents:

- **Encapsulament:** Es poden definir així diferents nivells i permisos d'accés.

- **Nomenclatura:** En el cas que vulguem tenir classes amb el mateix nom ho podrem fer amb packages si aquestes estan ubicades en packages diferents.

P.ex per utilitzar la llibreria *Vector*, cal el nom complet de la classe, que es refereix a la jerarquia de packages:

```
import java.util.Vector;
```

Figura 10: Exemple d'import per veure package.

Import

Els imports en Java s'utilitzen bàsicament per abreujar la referència que es fa a la classe, seguint amb l'exemple de l'apartat anterior, si utilitzem l'import de la Figura 10.

Així ara ens podem referir a la classe Vector com a Vector, sense haver d'indicar tota la ruta del package.

Una altra cosa que és interessant és si per exemple el que volem és importar totes les classes que hi ha en un package determinat, així doncs si volguéssim importar totes les classes del package util de Java hauríem de fer el que es mostra a la Figura 11.

```
import java.util.*;
```

Figura 11: Import de totes les classes del package util de Java.

Com podem observar el que s'utilitza és el símbol * per importar totes les classes.

Modificadors d'accés

Aquests modificadors d'accés de Java, poden ser tan aplicats a mètodes, constructors, atributs, etc. Els modificadors són els següents ordenats de més a menys permissiu:

- **Public:** Accessible des de tot arreu del projecte.
- **Protected:** Només accessible des de les classes que estan al mateix package o de les classes que s'han derivat d'aquesta.
- **Default:** És el modificador per defecte, quan no s'hi posa res, i només en tenen accés les classes del mateix package.
- **Private:** Només s'hi pot accedir des de la mateixa classe.

Les classes també cal comentar que poden ser publiques, privades o per defecte (sense posar-hi res).

Fitxers .jar

Els fitxers *.jar* són fitxers comprimits (format ZIP) que el que fa és agregar els fitxers *.class* d'una aplicació Java i eventualment també s'hi poden incloure altres recursos com poden ser text o imatges. Aquests fitxers són molt útils per a la portabilitat de les aplicacions Java. Alguns dels beneficis que ens pot aportar un fitxer JAR són els següents: Seguretat, reducció del temps de descàrrega, empaquetatge de les extensions, segellat del paquet, versionat de paquets i portabilitat.

Dins d'un fitxer *.jar* podem tenir-hi un fitxer de manifest, aquest fitxer el que farà és indicar al JAR com s'ha d'executar (quina és la classe principal, quins packages hi ha en el jar...)

Herència en Java

Per expressar l'herència de classes amb Java, per exemple el fet que una classe B es deriva d'una classe A es fa com es mostra en la Figura 12.

```
public class B extends T {
    // Constructors de la classe B
    // Atributs de la classe B (si en fan falta)
    // Mètodes de la classe B (si en fan falta)
    // Reescriptura de mètodes de la classe A (si fa falta)
}
```

Figura 12: Exemple d'herència amb Java.

Els objectes de la classe B hereten tots els atributs i mètodes de la classe A. Però des de B no es pot accedir als elements privats de la classe A. L'herència és transitiva, per tant si B deriva de A i C deriva de B, podem dir que C també deriva de A.

Els constructors en Java no s'hereten, això si és recomanable que els constructors de la classe derivada comencin invocant el constructor de la classe base. Per fer això s'utilitza el mètode super() com es mostra a la Figura 13.

```
public B (...){
    super(...); // Inicialitzem els atributs heretats
    // Inicialitzem els atributs específics de la classe B en el cas que fos
        necessari
}
```

Figura 13: Exemple constructor d'una classe derivada en Java.

En un objecte de Java diferenciem dos tipus:

- **Tipus aparent:** Tipus de variable que referència a l'objecte (temps de compilació).
- **Tipus real:** Tipus amb el que ha estat creat l'objecte (temps d'execució).

Estructures de dades

En aquest apartat veurem les estructures de dades de Java més utilitzades (Set, List, Map), a més a més comentarem que hi ha diferents implementacions per cada un dels diferents tipus de col·leccions **[8]**.

Set: Aquesta defineix una col·lecció en la qual no hi poden haver elements duplicats i conté només els mètodes heretats de *Collection*, afegint això sí que no hi poden haver elements duplicats.

Per poder comprovar si els elements són duplicats o no cal per tant tenir implementades les funcions `equals()` i `hashCode()`. Dins de la interfície Set, trobem diferents tipus d'implementacions com són: HasheSet, TreeSet i LinkedHashSet.

List: La interfície List defineix una successió d'elements, a diferència de la vista anteriorment aquesta sí que admet elements repetits. Conté mètodes que permeten millorar el següent: accés indexat als elements, cerca d'elements, iteració sobre els elements i subset d'elements de la col·lecció.

Dins de la interfície List, existeixen diferents implementacions com són: ArrayList i LinkedList.

Map: La interfície Map el que fa és associar claus a valors. Per tant no hi poden haver claus duplicades, ja que són úniques, cada una d'aquestes claus només pot tenir un valor associat a ella.

Dins la interfície Map trobem també diferents implementacions com són: HashMap, TreeMap i LinkedHashMap.

En la Figura 14 podem observar les diferents col·leccions que hi ha en Java.

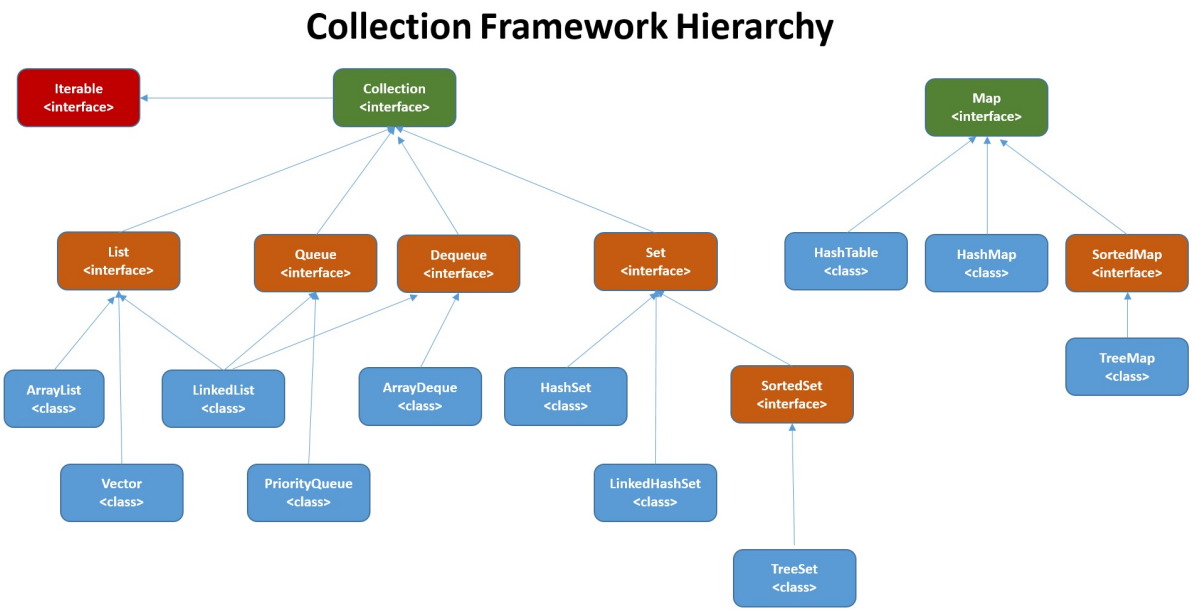


Figura 14: Col·leccions de dades en Java.

JavaFx

JavaFx es tracta d'un producte/tecnologia Java que va ser pensat per poder crear Rich Internet Applications (RIAs), aquest software pertany actualment a Oracle Corporation, els que són propietaris del mateix llenguatge Java.

JavaFx ens permet crear interfícies gràfiques d'usuari (GUI's) d'un aspecte molt més modern que les obtingudes amb versions anteriors com poden ser l'eina Swing. Tanmateix també permet incorporar animacions i gràfics d'entre altres coses.

Per poder-ne fer ús no cal instal·lar res d'especial, simplement cal tenir el JDK estàndard de Java i ja en tindríem prou.

Seguidament en la Figura 15 podem observar un exemple d'un GUI del típic Hello World.

Figura 15: Exemple GUI JavaFx.

Per últim una petita descripció dels elements bàsics que componen JavaFx:

- **Application:** Classe base per les aplicacions fetes amb JavaFx, aquesta ens aportarà totes les funcionalitats necessàries.
- **ActionEvent:** Objecte generat en produir-se una certa acció en l'aplicació, per exemple clicar un botó.
- **EventHandler:** Interfície genèrica que proporciona mètodes per gestionar els esdeveniments, aquests són rebuts com a paràmetre en les funcions que els gestionen.
- **Stage:** Representa el frame de l'aplicació JavaFx.
- **Scene:** Representa el contingut que hi ha en Stage.
- **Button:** Representa els botons.
- **StackPane:** Correspon a la disposició (layout) que permet col·locar els elements en l'escena.

Maven

Maven és una eina per la gestió de projectes de software, es basa en el concepte de **POM** (Proyect Object Model) **[9]**. Amb Maven, podrem compilar, empaquetar, generar documentació, passar tests, preparar les builds, etc.

En el fitxer *pom.xml* és on descriurem el nostre projecte, una de les coses més importants que ens permet fer Maven és posar-hi en aquest fitxer dependències, això ens permet fer ús de llibreries i importar-les des de Maven si ho indiquem correctament en aquest fitxer.

5.2.3 R

Una petita introducció a R

R és un llenguatge de programació interpretat, de distribució lliure sota llicència GNU, i té un ambient per al còmput estadístic i gràfic. És un software multiplataforma que pot corre sota Linux, MacOS, Windows i fins i tot Playstation3 **[10]**.

R està dividit en dues parts conceptuals, la primera seria el seu sistema base que és el que ens podem descarregar a **CRAN**. L'altra part consisteix en els paquets modulars que el poden fer extensible.

Algunes de les tasques que podem fer per exemple utilitzant aquests paquets són les següents: mineria de textos, processament d'imatges, visualització interactiva de dades i processament Big Data entre moles altres coses més que es poden fer.

Dades i els seus tipus

Els tipus de dades més comuns en R són els que podem observar en la figura Figura 16.

Tipo	Ejemplo	Nombre en inglés
Entero	1	integer
Numérico	1.3	numeric
Cadena de texto	"uno"	character
Factor	uno	factor
Lógico	TRUE	logical
Perdido	NA	NA
Vacio	NULL	null

Figura 16: Tipus de dades en R.

Per poder determinar el tipus d'una dada, en R podem utilitzar la funció `class()`, aquesta ens retornarà el nombre del tipus al qual pertany en anglès.

Matrius, llistes i data frames

Les matrius i arrays poden ser descrits com vectors multidimensionals. Igual que passa amb els vectors, únicament poden contenir dades d'un sol tipus. Les matrius són un cas especial d'un array, que es caracteritza per tenir específicament dues dimensions, una llargada i una altura. Per tant és una estructura rectangular amb files i columnes.

Per crear les matrius, en R tenim la funció `matrix()` com podem veure en la Figura 17, que conté dos arguments, nrow que representa el nombre de files i després tenim rcol que representa el nombre de columnes de la matriu. Les dades que intentem agrupar en la matriu, seran posades en ordre de dalt a baix i d'esquerra a dreta fins a formar un rectangle que representa la matriu en qüestió.

```
matrix(1:12, nrow = 3, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Figura 17: Exemple de com crear una matriu amb R.

Les llistes, igual que els vectors, són estructures de dades unidimensionals, només tenen una llargada, però a diferència d'aquests, cada element de la llista pot tenir un tipus de dades completament diferent, o inclús d'una classe diferent per tant són estructures heterogènies.

Per poder crear una llista en R simplement cal utilitzar la funció `list()` com es mostra en la Figura 18, simplement haurem de col·locar a dins els elements que vulguem incloure en la llista, i fins i tot tenim l'opció de donar un nom a cada element de la llista.

```
mi_vector <- 1:10
mi_matriz <- matrix(1:4, nrow = 2)
mi_df     <- data.frame("num" = 1:3, "let" = c("a", "b", "c"))

mi_lista <- list("un_vector" = mi_vector, "una_matriz" = mi_matriz, "un_df" = mi_df)

mi_lista
```

Figura 18: Exemple de com crear una llista amb R.

Els data frames són estructures de dades de dues dimensions com les matrius però a diferència d'aquestes, poden ser dades de diferents tipus, per tant són heterogènies. Aquesta estructura de dades és la més utilitzada per fer anàlisi de dades.

Els data frames permeten dades de diferents tipus com hem comentat anteriorment, però les columnes han de conservar la restricció de contenir dades d'un sol tipus. En termes generals, les files d'un data frame representen casos, individus o observacions, mentre que les columnes representen atributs, característiques o variables.

Per crear un data frame amb R simplement hem d'utilitzar la funció `data.frame()` com es mostra en la Figura 19, aquesta funció ens demana un nombre de vectors que ha de ser igual òbviament al nombre de columnes. Tots els vectors per tant han de tenir la mateixa longitud. És molt important destacar que un data frame està compost per vectors, cosa que aporta unes certes propietats al data frame. També tenim la funció `as.data.frame()` on li podem passar una matriu com paràmetre i ens la transformarà a un data frame.

```r
mi_df <- data.frame(
  "entero" = 1:4,
  "factor" = c("a", "b", "c", "d"),
  "numero" = c(1.2, 3.4, 4.5, 5.6),
  "cadena" = as.character(c("a", "b", "c", "d"))
)

mi_df
```

```
##   entero factor numero cadena
## 1      1      a    1.2      a
## 2      2      b    3.4      b
## 3      3      c    4.5      c
## 4      4      d    5.6      d
```

Figura 19: Exemple de com crear un data frame amb R.

Com hem comentat anteriorment podem assignar un nom a cada vector del data frame, que d'alguna manera es convertirà per tant en el nom de la columna.

Gràfics i dispositius

R compta amb un sistema de generació de gràfiques poderós i flexible.

Una de les funcions més utilitzades en R per la creació de gràfics, és la funció `plot()`, aquesta funció té un comportament especial, ja que depenent del tipus de dades que li donem d'arguments ens generarà un tipus diferent de gràfic. A més a més per cada tipus de gràfic també disposem de diferents paràmetres per controlar el seu aspecte des de la mateixa funció.

La funció `plot()` requereix d'un argument `x` que representarà **l'eix X** d'una gràfica. Aquest argument `x` requereix un vector. La resta d'arguments de la funció són opcionals, però l'argument més important és l'argument `y`. Igualment que l'altre argument aquest també requereix un vector i correspondrà a **l'eix Y** de la gràfica.

Depenent del tipus de dades que donem a la funció obtindrem un gràfic diferent, d'acord amb les regles que es mostren a continuació en la Figura 20.

`x`	`y`	Gráfico
Continuo	Continuo	Diagrama de dispersión (*Scatterplot*)
Continuo	Discreto	Diagrama de dispersión, `y` coercionada a numérica
Continuo	Ninguno	Diagrama de dispersión, por número de renglón
Discreto	Continuo	Diagrama de caja (*Box plot*)
Discreto	Discreto	Gráfico de mosaico (Diagrama de Kinneman)
Discreto	Ninguno	Gráfica de barras
Ninguno	Cualquiera	Error

Figura 20: Tipus de gràfics de la funció plot().

On els tipus de dades són:

- **Continu:** Un vector numèric, enter, lògic o complex.
- **Discret:** Un vector de factors o cadenes de text.

A més de la funció `plot()` hi ha altres funcions que generen tipus específics de gràfics. Per exemple per fer gràfics de barres podem utilitzar la funció `barplot()`.

Per poder exportar els gràfics que generem amb R, ho fem amb un procés que pot ser una mica confús.

Quan cridem a una funció que ens permet exportar els gràfics, el que li estem dient a R és que enviï el nostre gràfic a un dispositiu gràfic (graphic device), en el nostre ordinador. Una conseqüència d'això és que si enviem un gràfic a un dispositiu en ús, aquest el que farà és reemplaçar l'antic gràfic pel nou. És necessari enviar els nostres gràfics a un dispositiu per tant com pot ser JPG, PNG o qualsevol altre dispositiu que pugui ser emmagatzemat en el nostre disc dur.

Per poder exportar un gràfic podem utilitzar diferents funcions que corresponen a un tipus d'arxiu diferent com poden ser: `bpm()`, `jpeg()`, `pdf()`, `png()`, `tiff()` o `svg()`. Cada una d'aquestes funcions té els següents arguments:

- `filename`: El nom i ruta de l'arxiu imatge a crear (per defecte la ruta és el directori de treball).
- `width`: L'ample de l'arxiu imatge a crear, per defecte són píxels.
- `height`: L'altura de l'arxiu imatge a crear, per defecte són píxels.

Cal utilitzar aquestes funcions abans de generar un gràfic amb R, d'aquesta manera li indiquem que la sortida del gràfic s'ha de fer a través del dispositiu gràfic. Un cop fet això, cal tancar el dispositiu gràfic, això ho fem amb la funció `dev.off()`.

Seguidament en la Figura 21 podem veure un exemple de com exportaríem un gràfic en el format png.

```
png(filename = "loan_age.png", width = 800, height = 600)
plot(x = banco$age, y = banco$duration, col = banco$loan,
     main = "Edad y Duración", xlab = "Edad", ylab = "Duración")
legend(x = "top", legend = c("No", "Yes"), fill = c("Black", "Red"),
       title = "Loan")
dev.off()
```

Figura 21: Exemple export amb la funció png().

Els paquets en R

R pot ser expandit amb paquets. Cada paquet és una col·lecció de funcions dissenyades per poder realitzar tasques específiques. Aquests paquets es troben en **CRAN**, per on passen per un control abans d'estar disponibles pels usuaris finals.

Per instal·lar paquets en R ho fem amb la funció `install.packages()`, donant com a argument el nom del paquet que volem instal·lar entre cometes dobles.

Un cop feta la instal·lació d'un paquet, per utilitzar les funcions d'aquest hem d'utilitzar la funció `library()` on només li hem de passar per argument el nom del paquet aquesta vegada sense cometes.

5.2.4 Variables d'entorn

Independent del sistema operatiu que estiguem utilitzant, les variables d'entorn són la manera més senzilla de poder passar informació d'una aplicació a una altra **[11]**. Per tant en el cas que necessitem que una informació estigui disponible en alguna aplicació o eina del nostre sistema, simplement haurem de crear una variable d'entorn que la contingui.

Com el seu nom ens indica són variables per tant el seu valor pot variar, per tant poden ser alterades per un usuari, una aplicació o un script. Òbviament si en podem alterar el seu valor, també podem crear-ne o eliminar-ne segons sigui la necessitat que en tenim.

La importància de les variables d'entorn és que molta informació d'aplicacions o eines depèn d'elles per obtenir-ne informació i si alguna d'aquesta informació es trobés alterada llavors l'aplicació o eina deixaria de funcionar correctament.

6 Requisits del sistema

En aquest apartat el que farem és descriure els requisits que ha de complir el sistema, tant els requisits funcionals com els no funcionals. Aquí també descriurem tot allò que cal fer en el projecte perquè compleixi cadascun dels seus objectius, tant el punt de vista de software com de hardware.

Cal comentar que com el projecte ja té un punt de partida, els requisits que es comentaran són els requisits on només hi intervé la feina realitzada en el projecte.

6.1 Requisits funcionals

Els requisits funcionals defineixen una funció del sistema de software o dels seus components. Aquestes funcions són descrites com un conjunt d'entrades, uns comportaments i finalment unes sortides.

En definitiva els requisits funcionals estableixen el comportament del software.

Pel que fa als usuaris finals que utilitzin l'aplicació, els requisits funcionals del sistema dividit en grups són els següents:

6.1.1 Millores del CoDaPack amb Java

- Esborrar totes les taules que hi hagin carregades en el sistema.
- Netejar la consola de missatges en el cas que així ho vulgui l'usuari.
- S'ha de poder definir un closure To per defecte en el menú de configuració.
- Mostrar o amagar els menús de desenvolupador en el menú de configuració.
- S'ha de poder canviar la ruta del directori on es busquen els scripts per defecte.
- Canviar els valors seleccionats d'una variable categòrica.
- Poder filtrar observacions amb el mateix valor en la variable categòrica seleccionada.
- Poder crear un data frame de zero (podent fer copy paste des de un excel), indicant el nombre de columnes i de files.
- Poder visualitzar els centres en els ternary/quaternary plot.
- Poder guardar els gràfics generats amb Java en format PDF.
- Visualitzar etiquetes en el balance dendogram.
- Poder guardar l'estat de la consola en sortir de l'aplicatiu.
- Guardar els gràfics generats amb R amb el format SVG.
- Poder canviar els noms de les variables directament en la taula.
- Poder visualitzar help en cada un dels menús.

6.1.2 Connexió directa amb R

- Discretitzar/Segmentar una variable continua en una variable categòrica (factor) segons el següent:
  - Method
  - # de levels
  - Si es vol afegir noms als grups o no
- Calcular una nova variable a partir d'altres amb una expressió d'R.
- Ordenar les dades seleccionades segons si es vol en ordre ascendent o descendent.
- Poder fer un subconjunt del conjunt de dades que compleixen una certa condició en les variables seleccionades.
- Executar la rutina Non-parametric Replacement esperant obtenir com a resultats unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - DL proportion
  - Closure result
  - Closure to
- Executar la rutina Logratio-EM zero Replacement esperant obtenir com a resultats unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Rob Option
  - IniCov Option
  - DL proportion
- Executar la rutina Classical Univariate Noramlity test esperant obtenir com a resultat una sortida en la consola a partir dels següents paràmetres:
  - Selecció de variables
  - Normality Test
- Poder mostrar la següent informació amb R:
  - La versió d'R, el sistema operatiu i els paquets carregats
  - Obtenir els valors de les variables d'entorn
  - Informe sobre les característiques opcionals que han sigut compilades en la construcció d'R

6.1.3 Execució d'scripts d'R

- Executar la rutina zPatterns plot esperant obtenir com a resultats una sortida tant a la consola com una sortida gràfica a partir dels següents paràmetres:
  - Selecció de variables
  - Show means
  - Show percentages
  - Label
- Executar la rutina Bayesian Multiplicative Replacement esperant obtenir com a resultats unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Method
  - Output
- Executar la rutina Logratio-EM missing replacement esperant obtenir com a resultats unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Rob Option
- Executar la rutina Logatio-EM Zero & missing replacement esperant obtenir com a resultats unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Rob Option
  - IniCov Option
  - DL proportion
- Executar la rutina Atipicality Index esperant obtenir com a resultats una sortida en la consola i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Level of Confidence
- Executar la rutina X real Y composition regression esperant obtenir com a resultats una sortida en la consola, un nou data frame, una sortida gràfica i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selected X
  - Selected Y
  - Y partition
  - Residuals
  - Fitted
- Executar la rutina X composition Y real regression esperant obtenir com a resultats una sortida en la consola, un nou data frame, una sortida gràfica i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selected X
  - Selected Y

  - X partition
  - Residuals
  - Fitted

- Executar la rutina Cluster k means espeant obtenir com a resultat una sortida en la consola, una sortida gràfica i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Nombre de clusters ($>= 2$)
  - Find optimal number between 2 and (> 2)
  - Name of column of groups
  - Optimal Method
- Executar la rutina Manova esperant obtenir com a resultat una sortida en la consola i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - Residuals
  - Dif. between paris of groups
- Executar la rutina Discriminant Analysis esperant obtenir com a resultat una sortida en la consola, una sortida gràfica i unes noves variables en el data frame actual a partir dels següents paràmetres:
  - Selecció de variables
  - X partition
  - Use of uniform prior
  - Predicting for a new sample Z
  - Discriminant scores
  - Max. posteriori prob. classification
  - Posterior prob. for the classes
- Executar la rutina Boxplot esperant obtenir una sortida gràfica a partir de les variables seleccionades.
- Executar la rutina Scatterplot 2D/3D esperant obtenir una sortida gràfica a partir dels següents paràmetres:
  - Selecció de variables
  - Set same scale
- Executar la rutina Geometric mean barplot esperant obtenir una sortida gràfica a partir de les variables seleccionades.

6.2 Requisits no funcionals

Es tracten de requisits que no defineixen les funcions específiques que dóna el sistema (catacterístiques d'usuari), sino les propietats del sistema com poden ser: el rendiment, la seguretat o la disponibilitat.

Altrament, també defineixen restriccions del sistema com pot ser la capacitat dels dispositius d'entrada i sortida i la presentació de les dades utilitzades en la interfície del sistema.

Seguidament els requisits no funcionals del projecte:

- L'aplicació ha de funcionar en els diferents sistemes operatius: Windows, Linux i MacOs.
- És necessari tenir instal·lada una màquina virtual de Java per poder utilitzar l'aplicació (mínim Java 1.8.1).
- És necessari tenir una versió d'R (mínim 3.5) per poder realitzar les rutines programades amb R.
- No s'ha de requerir de software addicional per poder fer ús de l'aplicació, ja que Java i R venen amb la instal·lació.
- Les dades introduïdes per l'usuari no han de ser alterades en cas de no haver-se utilitzat per realitzar cap operació.
- L'usuari ha de seleccionar les dades coherentment pels diferents menús pel seu bon funcionament.
- Els scripts que utilitza l'usuari no ha de contenir errors, en cas que hi hagi error no es garanteix el bon funcionament de l'aplicació i el que fem és capturar l'error que reben des de R i mostrar-li a l'usuari per la consola del CoDaPack.
- Tanmateix també els formats d'arxius d'entrada han de ser vàlids.
- La quantitat d'operacions que l'usuari pot realitzar en l'aplicació ha de ser il·limitada a efectes pràctics.

7 Estudis i decisions

En aquest apartat descriurem el maquinari utilitzat, les diferents llibreries i/o programari utilitzats durant el desenvolupament del projecte i la justificació del seu ús. Tanmateix també parlarem de la versió de Java utilitzada.

7.1 Maquinari

Seguidament procedirem a descriure el maquinari utilitzat durant el desenvolupament del projecte, ho dividirem en els dos maquinaris utilitzats, en primer lloc una màquina virtual de Windows i en segon lloc un Mac mini.

7.1.1 Màquina Virtual de Windows 10

Primer de tot trobem una Màquina Virtual de Windows 10 que ha estat utilitzada per fer la major part del desenvolupament del projecte, en ella s'hi han instal·lat tots els elements necessaris per poder realitzar el projecte com pot ser per exemple un IDE de Java. Seguidament una petita descripció de la màquina virtual:

- **Processador:** Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 3.6GHz
- **Memòria RAM:** 10074MB
- **Espai Lliure al Disc:** 50GB
- **Sistema Operatiu:** Windows 10 Education 64 bits (10.0, compilación 18362)

7.1.2 Mac mini

En segon lloc trobem, el Mac mini que ha estat utilitzat molt menys que la màquina comentada anteriorment, aquesta bàsicament ha estat utilitzada per fer les compilacions del codi en una màquina amb sistema operatiu MacOS i poder testejar l'aplicació en aquest sistema operatiu en concret. Seguidament una petita descripció de les característiques que conté el Mac mini:

- **Processador:** 1,4 GHz Intel Core i5
- **Memòria RAM:** 8 GB 1600 MHz DDR3
- **Gràfica:** Intel HD Graphics 5000 1536 MB
- **Espai Lliure al Disc:** 500GB
- **Sistema Operatiu:** MacOs Mojave versió 10.14.6

7.2 Versió de Java

La versió que s'utilitzi en Java és una decisió important, ja que la versió de Java farà augmentar o baixar la compatibilitat amb altres versions de softwares i sistemes operatius, i així doncs també pot fer augmentar o disminuir l'abast en el mercat al públic. Per tant principalment el que s'intenta a l'hora d'escollir una versió de Java és agafar una versió que sigui estable i que sigui altament compatible amb la majoria de dispositius i software que la gent utilitzi en l'actualitat. Un cop dit això passarem a explicar el cas concret en aquest projecte.

Seguidament en la Figura 22 podem veure les diferents versions de Java amb la seva data de sortida, la data de fi d'actualitzacions i la data fins quan tenen suport:

Version	Release date	End of Free Public Updates[5][6]	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	**January 2019 for Oracle (commercial)** December 2020 for Oracle (personal use) At least May 2026 for AdoptOpenJDK At least June 2023[7] for Amazon Corretto	December 2030
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	At least August 2024[7] for Amazon Corretto October 2024 for AdoptOpenJDK	September 2026
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	TBA	TBA

Legend: Old version | Older version, still maintained | **Latest version** | Future release

Figura 22: Versions de Java.

Al principi del projecte ja hi havia un software del CoDaPack programat en Java i que estava fet i compilat amb la versió de Java 8. En començar a fer el projecte del CoDaPack amb R vam decidir fer un petit canvi en la versió de Java i utilitzar una més nova i actual, aquesta és la versió 10 de Java.

En principi aquesta versió estava funcional, operativa i funcionava correctament en sistemes operatius de Windows 10. Però en fer un instal·lador amb la versió de Java 10 per Mac, hi va haver un component

del grup de recerca que en el seu ordinador no funcionava bé per culpa d'incompatibilitats amb Java 10, el seu sistema operatiu no suportava la versió 10. Per tant vaig tornar a fer un instal·lador per mac però amb la versió 8 de Java i ja funcionava correctament.

Finalment per tant per temes d'incompatibilitats i, ja que Java 8 ja funciona correctament, és força compatible amb la majoria d'usuaris i a més a més és LTS (Long Time Suport), fins al Desembre de 2030 en principi. Vam decidir conservar la versió de Java 8 i a partir d'ara fer servir aquesta.

En la Figura 23 podem observar la distribució de les diferents versions de Java i podrem veure que la més extensa és la de Java 8:

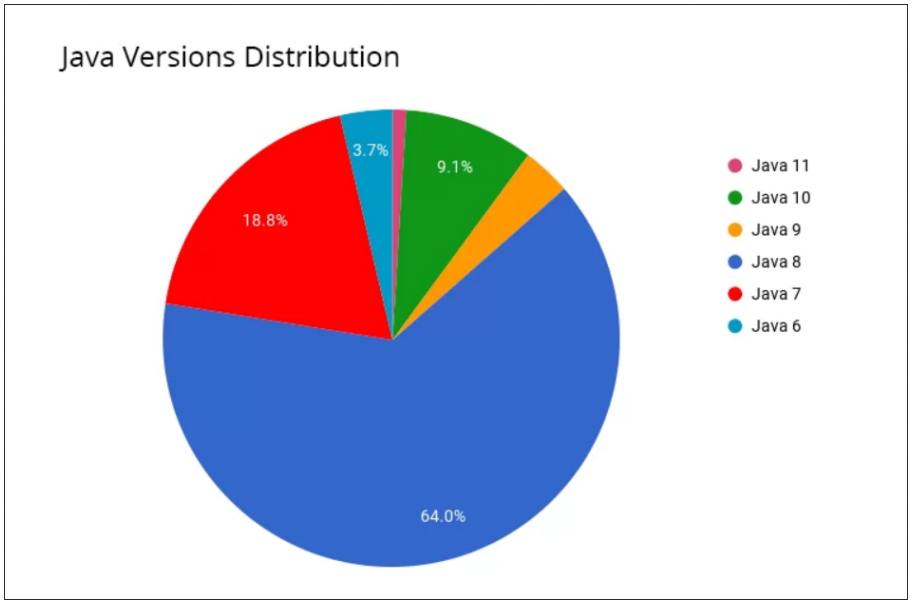


Figura 23: Distribució de versions de Java.

7.3 CoDaPack

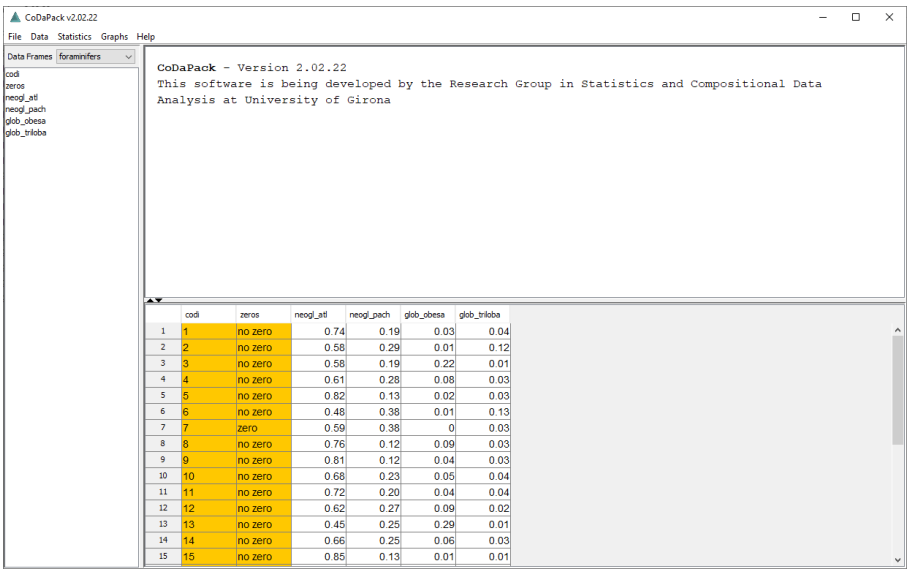
El projecte es basa en una aplicació de partida, aquesta aplicació és el CoDaPack, per tant a continuació es proporcionaran un seguit d'elements per tal d'entendre el funcionament principal de l'aplicació base.

Cal remarcar que els elements que s'explicaran a continuació formen part de la versió en la qual estava el CoDaPack abans d'iniciar el projecte és a dir la versió 2.02.22.

Per tal que funcioni CoDaPack, simplement s'ha d'instal·lar i obrir l'arxiu anomenat "CoDaPack.exe". CoDaPack està basat en menús, els resultats numèrics apareixen en la part d'outputs de la finestra, mentre que les sortides gràfiques apareixen en una nova finestra, en el cas de noves variables apareixeran al final de la taula de dades.

Les dades poden ser importades des d'arxius Excel, CSV, R o recuperats de sessions anteriors. Les observacions de les dades s'organitzen en files i les variables en les columnes. La pàgina principal del CoDaPack Figura 24, conté quatre parts. En la part superior trobem els menús interactius, a l'esquerra trobem el marc de les dades actives en el moment i el nom de les seves variables. En la part superior de la part dreta trobem el lloc on es mostren els resultats alfanumèrics, en canvi en la part inferior trobem les dades de l'arxiu carregat actualment.

Mitjançant els menús de la part superior es poden executar rutines que poden retornar: resultats numèrics en la part de sortida de la finestra o resultats gràfics com a finestres independents com hem comentat anteriorment.



	codi	zeros	neogl_atl	neogl_pach	glob_obesa	glob_triloba
1	1	no zero	0.74	0.19	0.03	0.04
2	2	no zero	0.58	0.29	0.01	0.12
3	3	no zero	0.58	0.19	0.22	0.01
4	4	no zero	0.61	0.28	0.08	0.03
5	5	no zero	0.82	0.13	0.02	0.03
6	6	no zero	0.48	0.38	0.01	0.13
7	7	zero	0.59	0.38	0	0.03
8	8	no zero	0.76	0.12	0.09	0.03
9	9	no zero	0.81	0.12	0.04	0.03
10	10	no zero	0.68	0.23	0.05	0.04
11	11	no zero	0.72	0.20	0.04	0.04
12	12	no zero	0.62	0.27	0.09	0.02
13	13	no zero	0.45	0.25	0.29	0.01
14	14	no zero	0.66	0.25	0.06	0.03
15	15	no zero	0.85	0.13	0.01	0.01

Figura 24: Pàgina principal CoDaPack v2.02.22.

Cada una de les rutines del menú, demana a l'usuari les dades que s'han de tenir en compte per realitzar la rutina. Alguns d'aquests menús tenen opcions a part de la selecció de dades per modificar els valors per defecte.

Per poder realitzar qualsevol de les rutines que el CoDaPack ofereix simplement cal anar als menús de la part superior i seleccionar el que es vulgui amb un clic del ratolí. En fer clic ens apareix una nova finestra com és mostra per exemple en la Figura 25, que és una finestra estàndard per la majoria de rutines de CoDaPack, en aquesta se'ns demana quines de les columnes de les dades actuals s'han

de seleccionar. La part esquerra de la finestra conté l'estructura de dades disponibles del data frame carregat actualment, la part central l'estructura de les dades seleccionades, l'estructura de grups si és que n'hi ha i un botó per reiniciar les dades seleccionades. En la part dreta de la finestra, trobem les opcions que estan disponible per cada rutina en concret en el cas que en fessin falta. Per últim, en la part inferior de la finestra es troben els botons d'acceptar i de cancel·lar, òbviament el d'acceptar realitza la rutina amb les dades proporcionades i el de cancel·lar torna a la finestra d'inici. Entre la part esquerra i la part dreta, trobem dues fletxes per intercanviar dades entre les dues seleccions de dades. També es pot passar una dada fent doble clic en ella.

Per tant el que hauria de fer un usuari, és primer de tot seleccionar les parts que s'utilitzaran del data frame per realitzar la rutina, per exemple en la Figura 25, les parts que s'utilitzaran per realitzar un gràfic amb la rutina dendograma. Per això s'han de marcar una o més variables de la llista de dades disponibles i passar-les com a dades seleccionades simplement fent un doble clic o amb les fletxes com hem comentat anteriorment. Aquesta operació s'ha de fer amb cada una de les dades que vulguem tenir com a seleccionades. En el cas que tinguem l'opció de seleccionar grups, s'hauria de seleccionar una variable categòrica del data frame actual en el cas que ho vulguem utilitzar.

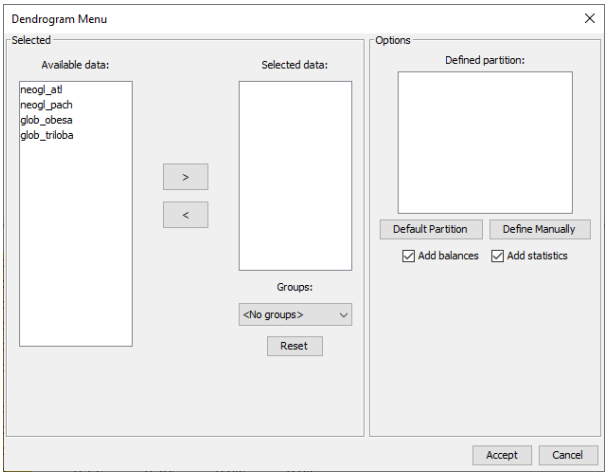


Figura 25: Exemple menú dendogram.

El CoDaPack com hem comentat anteriorment en realitzar una rutina, ens podria retornar tres tipus de sortides diferents:

- **Noves variables:** Aquestes són col·locades al final de la taula de dades.
- **Sortides alfanumèriques:** Aquestes apareixerien en la part de sortides en la part superior de la finestra.
- **Sortides gràfiques:** Aquestes apareixen en una nova finestra independent.

Com podem observar en la Figura 24, CoDaPack té cinc menús principals que són els següents: *File*, *Data*, *Statistics*, *Graphs* i *Help*.

CoDaPack emmagatzema un conjunt de dades carregat per l'usuari en un data frame o taula. Es pot tenir més d'un data frame obert al mateix temps. Aquest conjunt de taules o data frames pot ser guardat en el que s'anomena "*Espai de Treball*", i pot ser obert en una altra sessió. Aquest "*Espai de Treball*" té un format "*.cdp*".

Cada data frame conté emmagatzemat els noms de les variables i els seus valors numèrics o categòrics. Hi ha dos tipus de valors que falten, les dades no detectades o les dades no disponibles, i aquestes han de tenir un símbol per poder distingir-los. Les dades no detectades comencen amb un prefix de caràcter, per exemple "<", seguit del valor de detection limit oflow, mentre que les dades no disponibles han d'utilitzar un símbol com per exemple "*NA*".

7.4 Llibreries de Java utilitzades

Seguidament comentem les llibreries de Java, emmagatzemades en Maven, que s'han utilitzat per poder obtenir funcionalitats o característiques concretes d'aquests. Per cada llibreria comentarem el que ens aporten i una petita descripció.

JYAML

JYaml **[12]** és una llibreria de Java per poder treballar amb el format d'arxius YAML.

YAML és un llenguatge de serialització de dades que s'adapta molt bé a les dades estructurades, és molt més simple i llegible que per exemple altres llenguatges com pot ser XML. Normalment s'utilitza per arxius de configuració, d'inicialització i altres arxius específics d'aplicacions.

Amb JYaml, les entitats d'un arxiu YAML es poden mapejar directament amb objectes de programació, per exemple un map amb <clau,valor>. Això vol dir que la interacció amb aquesta llibreria és molt simple i en la majoria de casos simplement caldrà fer una crida i ja.

Batik-Swing

L'objectiu de la llibreria Batik Swing **[13]** és proporcionar un component Swing que es pugui utilitzar per mostrar documents amb un format SVG.

Amb la classe JSVGCanvas, es pot mostrar fàcilment un document SVG i permetre a l'usuari la seva manipulació com pot ser: rotar, fer zoom, desplaçar-se, seleccionar text o activar un enllaç.

Commons-Math

Commons-Math **[14]** és una llibreria de components matemàtics i estadístics lleugers i autònoms que tracten els problemes més comuns no disponibles en el llenguatge de programació Java.

Principis bàsics:

- Els casos d'ús de l'aplicació en el món real determinen la prioritat de desenvolupament.
- Aquest paquet fa èmfasis en petits components i fàcils d'integrar.
- Tots els algoritmes estan documentats i segueixen les millors pràctiques generalment acceptades.
- En casos d'existència de diversos algoritmes estàndard, s'utilitza un patró d'estratègia per sostenir diferents implementacions.
- No hi ha dependències més enllà dels components comuns i a plataforma central de Java.

Renjin-Script-Engine

Renjin **[15]** és un intèrpret del llenguatge de programació R per la computació estadística escrita en Java, com JRuby i Jython ho són per llenguatges de Ruby i Python. El projecte oficial R, d'aquí endavant anomenat GNU R, és la implementació de referència per al llenguatge R.

L'objectiu d'Renjine és ser eventualment amb GNU R de tal forma que la majoria de programes existents del llenguatge R s'executin en Renjin sense necessitat de fer cap canvi en el codi. No fa falta comentar que actualment Renjin no és 100% compatible amb GNU.

Poi

Apache POI **[16]** és una popular API que permet als programadors: crear, modificar i mostrar arxius MS Office utilitzant programes programats amb Java.

És una llibreria de codi obert desenvolupada i distribuïda per Apache Software Foundation per dissenyar o modificar arxius de Microsoft Office utilitzant programes Java.

Conté classes i mètodes per descodificar les dades d'entrada de l'usuari o un arxiu en documents de MS Office.

Swing-Layout

L'objectiu de Swing Layout Extensions **[17]** és facilitar la creació de dissenys professionals de multiplataforma amb Swing.

Aquest projecte té en compte les necessitats dels constructors de GUI, com NetBeans.

Aquest projecte consisteix en les següents peces:

- Capacitat per obtenir la base dels components.
- Capacitat per obtenir l'escletxa preferida entre els components.
- Un nou administrador de disseny que utilitza ambdós conceptes.

JLatexMath

JLaTeXMath **[18]** és una llibreria de Java, el seu propòsit és, el de mostrar formules matemàtiques escrites en LaTeX, és considerada la millor llibreria de Java per mostrar codi LaTeX.

Aquesta llibreria és utilitzada per diversos projectes importants com poden ser: Scilab, Geogebra, Freeplane, Mathpiper, CaRMetal, Ultrastrudio...

La codificació per defecte és UTF-8.

OpenCSV

OpenCSV **[19]** és una llibreria d'anàlisi CSV (valors separats per comes) fàcil d'utilitzar per Java.

Va ser desenvolupada perquè tots els analitzadors CSV en aquell moment no tenien llicencies comercials. Actualment Java 8 és la versió mínima suportada.

OpenCSV suporta totes les coses bàsiques de tipus CSV que probablement vulguis fer:

- Números arbitraris de valors per línia.
- Ignora les comes en els elements citats.
- Gestiona les entrades entre cometes amb els retorns de carro incrustats (entrades de diverses línies).
- Caràcters configurables de separador i de cometes.

JRI

JRI **[20]** és una interfície Java/R, que permet executar R dins d'aplicacions Java com un únic thread. Bàsicament carrega la llibreria dinàmica d'R en Java i proporciona una funcionalitat API de Java a R. Suporta tant crides simples a funcions d'R com una completa execució d'REPL.

En cert sentit, JRI és l'invers de rJava i els dos poden combinar-se. El projecte JGR fa ús complet tant de JRI com d'rJava per proporcionar una completa interfície gràfica de Java per R.

JRI utilitza codi natiu, però suporta totes les plataformes en les quals està disponible Java, és a dir: Windows, Mac OS, Sun i Linux (tots de 32 i 64 bits).

REngine

La classe Rengines **[21]** és la interfície entre una instància d'R i la màquina virtual de Java. Pel fet que R no té suport de threading, només es pot executar una instància d'R amb una aplicació multi-thread. Hi ha dues formes d'utilitzar R des de Java: crides individuals i bucle d'events complet.

JRIEngine

JRIEngine **[22]** és una implementació d'REngine utilitzant JRI (Java/R).

S'ha de tenir en compte que com a molt pot existir una instància JRI en el procés JVM, perquè R no suporta múltiples threads. JRIEngine és segur pels threads per tant és possible invocar els seus mètodes des de qualsevol thread. Però això és possible gràcies a la serialització de totes les entrades d'R, per tant s'han de tenir en compte totes les possibilitats de bloqueig si el codi d'R crida novament a Java.

ControlsFX

ControlsFX **[23]** és un projecte de codi obert per JavaFX que té com a objectiu proporcionar controls d'interfície d'usuari de molt alta qualitat i altres eines per complementar el nucli de la distribució de JavaFX.

Ha sigut desenvolupat per JavaFX 8.0 i versions posteriors, i té com a principi acceptar nous controls/funcions quan tot el codi existent està a un nivell altament acceptable, inclòs treballs com la documentació.

Això permet garantir un llançament d'alta qualitat en tot moment.

ITextPDF

Itext **[24]** és una llibreria Open Source per crear i manipular arxius: PDF, RTF i HTML en Java. Va ser escrita per Bruno Lowagie, Paulo Soares i altres i està distribuïda per Affero General Public License.

El mateix document pot ser exportant en diferents formats, o diferents instàncies del format.

Recentment ha estat estesa a una llibreria PDF de propòsit general, amb la possibilitat d'emplenar formularis, moure pàgines d'un PDF...

El suport de PDF d'iText és molt extensiu. Suporta signatures basades en PKI de PDF, xifrat de 40 i 128 bits, correcció de colors, PDF/X...

JZY3D-API

Jzy3d **[25]** és una llibreria Java de codi obert que permet dibuixar fàcilment dades científiques en 3D: superfícies, gràfics de dispersió, gràfics de barres i altres. L'api proporciona suport per gràfics interactius, amb barres de colors i consells d'eines. L'eix i el disseny dels gràfics poden ser totalment personalitzats i millorats.

Confiant en JOGL 2, es pot desplegar fàcilment gràfics OpenGL natius en Windows, Unix, MacOs i integrar-los en Swing, AWT o JavFX.

7.5 Llibreries d'R utilitzades

Durant el projecte, han sigut necessàries llibreries d'R que ens aporten funcions i funcionalitats que ens fan falta, a continuació comentem algunes d'elles amb una petita explicació del que fan i de què estan compostes.

Lattice

És un poderós i elegant sistema de visualització de dades d'alt nivell inspirat en els gràfics Trellis, amb èmfasis en les dades multivariades **[26]**.

L'entramat és suficient per a les necessitats típiques dels gràfics, i també és prou flexible per gestionar la majoria de requeriments que no són estàndard.

coda.base

Un conjunt mínim de funcions per realitzar l'anàlisi de dades composicionals utilitzant l'aproximació de relació logarítmica introduïda per John Aitchison (1982) <http://www.jstor.org/stable/2345821> **[27]**.

Les funcions principals s'han implementat en c++ per un millor rendiment.

zCompositions

Mètodes principals per la imputació de zeros, dades censurades a l'esquerra i dades que falten en conjunts de dades composicionals (Palarea-Albaladejo i Martin-Fernandez (2015)) **[28]**.

plyr

Conjunt d'eines que resolen un conjunt comú de problemes: necessites dividir un gran problema en peces petites i després tornar-les a muntar un altre cop. Per exemple, ajustar un model a cada punt de localització espacial o temporal en l'estudi, resumir les dades per panells o col·lapsar matrius d'alta dimensió per simplificar les estadístiques de resum **[29]**.

rJava

Interfície de baix nivell per màquines virtuals de Java molt semblant a .C/.Call. Permet la creació d'objectes, mètodes de crides i accés a camps **[30]**.

scatterplot3d

Dibuixa un núvol de punts tridimensionals (3D) **[31]**.

fpc

Diversos mètodes per clustering i validació de cluster. Clustering de punts fixos **[32]**. Clustering de regressió lineal. Projeccions simètriques i asimètriques per la visualització de la separació de grups. Estadística de variables per la interpretació de grups...

arules

Proporciona la infraestructura per representar, manipular i analitzar les dades i patrons de les transaccions. També proporciona implementacions dels algoritmes de mineria de dades d'associació Apriori i Eclat.Hahsler, Gruen i Hornik (2005) **[33]**.

7.6 Programes més importants utilitzats

Seguidament els programes més utilitzats en el desenvolupament del projecte, on trobarem una introducció de cada programa, el seu ús en el projecte i el perquè s'ha decidit utilitzar aquest software i no un altre.

NetBeans

NetBeans Figura 26 és un entorn de desenvolupament integrat lliure, ha estat fet principalment per al llenguatge de programació Java. A més a més hi ha mòduls per estendre'l. És un producte lliure i gratuït sense restriccions.

Algunes de les característiques de l'aplicació són:

- Gestió de la interfície de l'usuari.
- Gestió de configuració de l'usuari.
- Gestió d'emmagatzematge.
- Gestió de finestres.
- Llibreria visual de NetBeans.
- Eines de desenvolupament integrat.

Figura 26: Logo de NetBeans.

En el projecte s'ha utilitzat NetBeans com a IDE per programar l'aplicació en el llenguatge Java i poder utilitzar les eines que ens ofereix com: la utilització de llibreries de Maven, el testing i la compilació del programa...

La decisió d'utilitzar aquest software és per les eines que ens ofereix, perquè és un dels IDE de Java més utilitzat i més conegut i també perquè és de codi obert i sense restriccions d'utilització.

R Studio

RStudio Figura 27 és un entorn de desenvolupament integrat (IDE) per al llenguatge de programació R, dedicat a la computació estadística i gràfics. Aquest inclou una consola, un editor de sintaxi, eines per traçar, la depuració i la gestió de l'espai de treball.

Aquest IDE està disponible per Windows, Mac i Linux, fins i tot hi ha una versió per navegadors que estiguin connectats a RStudio Server. RStudio té el propòsit de proporcionar l'entorn informàtic estadístic d'R.

Permet analitzar i desenvolupar perquè qualsevol persona pugui analitzar dades amb el llenguatge d'R.

Figura 27: Logo d'RStudio.

En el projecte s'ha utilitzat RStudio per poder provar codi escrit en R i així poder desenvolupar rutines del projecte escrites en llenguatge R. Així doncs ha servit per poder realitzar scripts de rutines que després es criden en el CoDaPack des de Java.

La decisió d'utilitzar RStudio és perquè és el millor entorn de programació per al llenguatge R que utilitzem per realitzar scripts i rutines del nostre projecte, a més a més és gratuït.

Launch4j

Launch4j Figura 28 és una eina multiplataforma per empaquetar aplicacions Java (jar) en executables natius lleugers de Windows. Aquest executable es pot configurar per buscar una determinada versió de JRE o utilitzar una inclosa. El contenidor també proporciona una millor experiència d'usuari a través d'una icona d'aplicació, una pantalla de presentació nativa anterior a JRE i una pàgina de descàrrega de Java en el cas que no es trobi un JRE que sigui apte per executar l'executable.

Algunes de les característiques són les següents:

- No extreu el jar de l'executable.
- Admet aplicacions de consola i GUI.
- Admet manifestos d'aplicacions de Windows.
- Estableix variables d'entorn.
- És gratuït i pot utilitzar-se amb caràcter comercial.

Figura 28: Logo de Launch4j.

En el projecte s'ha utilitzat l'aplicació Launch4j per poder realitzar els executables natius per sistemes operatius Windows.

La decisió d'utilitzar Launch4j és perquè és una eina que ens permet fer un exe a partir del jar que es genera en compilar el CoDaPack i a més ens permet definir variables d'entorn.

Inno Setup

Inno Setup Figura 29 és un instal·lador gratuït per programes de Windows de Jordan Russell i Martijn Laan. Introduït per primer cop en 1997, actualment avui rivalitza i supera a molts instal·ladors comercials quant a característiques i estabilitat.

Algunes de les característiques són les següents:

- Suport per cada versió de Windows des de 2006.
- Ampli suport per la instal·lació d'aplicacions de 64 bits en les edicions de 64 bits de Windows.
- Admet la creació d'un sol exe per instal·lar el programa per una fàcil distribució en línia.
- Completa capacitat de desinstal·lació.
- Instal·lacions silencioses i desinstal·lacions també silencioses.

Figura 29: Logo de Inno Setup.

En el projecte s'ha utilitzat Inno Setup per poder realitzar instal·ladors per les versions de Windows de 64 bits amb l'exe realitzat amb Launch4j.

La decisió d'utilitzar Inno Setup és que és de les millors eines per crear instal·ladors de Windows amb moltes característiques i funcionalitats a més de ser gratuït.

7.7 Llenguatges utilitzats durant el desenvolupament del projecte

A continuació parlarem dels llenguatges de programació més utilitzats durant el desenvolupament del projecte, en farem una petita introducció, la seva utilització en el projecte i el perquè del seu ús.

Java

Java és un llenguatge de programació de propòsit general que està basat en classes, orientat a objectes i dissenyat per tenir la menor quantitat possible de dependències d'implementació. El seu objectiu és permetre que els desenvolupadors d'aplicacions simplement hagin d'escriure una vegada, i s'executi en qualsevol lloc, el que vol dir que un cop el codi Java compilat es pugui executar en qualsevol sistema que suporti Java sense la necessitat d'haver de tornar a compilar.

Les aplicacions de Java normalment es compilen en bytecode que permet executar-se en qualsevol màquina virtual de Java independentment de l'arquitectura informàtica. En el 2019, Java va ser un dels llenguatges més populars segons GitHub.

Java ha sigut el llenguatge principal en què està programat CoDaPack, per tant és fonamental en el desenvolupament del projecte. La decisió d'utilitzar Java és perquè CoDaPack ja estava programat amb aquell llenguatge.

R

R és un llenguatge i un entorn per la computació estadística i els gràfics. És un projecte GNU similar al llenguatge i entorn S. R pot considerar-se com una implementació diferent de S.

R proporciona una àmplia varietat de tècniques estadístiques i gràfiques a més de poder ser molt extensible. Un dels punts forts d'R és la facilitat de poder produir gràfics ben dissenyats, on es poden incloure símbols i formules matemàtiques sempre quan sigui necessari.

R és un Software LLiure sota Llicencia Pública GNU. Compila i s'executa sota moltes plataformes d'entre elles podem trobar: Windows, Unix i MacOS.

Podríem dir que R és el segon llenguatge en importància dins el projecte únicament darrere de Java. En el projecte s'utilitza R per crear rutines directament amb crides a R i també tenim rutines creades directament amb un script d'R que és crida en Java. Per tant és un llenguatge important en el projecte.

La decisió d'utilitzar R és clara, ja que en termes de computació estadística és el software més important i reconegut a més de poder incloure i utilitzar les llibreries fetes per R que contenen altres rutines necessàries per al desenvolupament del projecte.

HTML

HTML és un llenguatge d'etiquetes que bàsicament s'utilitza per a la construcció de la web i s'utilitza per definir el sentit i estructura del contingut d'una pàgina web. Altres tecnologies són utilitzades conjuntament per descriure l'aparença i el funcionament de la web com són CSS i JavaScript.

En el projecte s'utilitza HTML en una part molt important com són les sortides en el panell de sortides, qualsevol sortida està escrita amb HTML, ja sigui simple text com el seu format, taules...

JavaScript

JavaScript és un llenguatge de programació lleuger i interpretat, orientat a objectes amb funcions de primera classe, conegut com el llenguatge de script per pàgines web, però també és utilitzat en molts entorns que no utilitzen navegador com són: node.js, Apache CouchDB o Adobe Acrobat.

És un llenguatge script multi-paradigma, basat en prototips, dinàmic, suporta estils de programació funcional, orientat a objectes i imperatiu.

En el projecte quasi no s'utilitza JavaScript, únicament s'utilitza en el panell de sortida de missatges del CoDaPack, ja que cada cop que surt un missatge no feia scroll fins a baix, per tant es va haver d'utilitzar JavaScript per fer scroll fins a baix de tot cada cop que s'escriu una sortida per al panell de sortides.

Shell scripts

Els shell scripts són programes d'ordinador dissenyats per ser executats per al shell de Unix és a dir un intèrpret de línia de comandes. Les operacions típiques executades pels shell scripts inclou: manipulació d'arxius, execució de programes o sortida textual.

En el projecte fem ús dels shell scripts per poder executar correctament l'aplicació del CoDaPack en els sistemes operatius de Mac Os, ja que hem de definir prèviament uns paràmetres abans d'iniciar l'aplicació, cosa que ens permet fer el shell script.

YAML

YAML és un format de serialització de dades llegible per humans inspirat en llenguatges com són: XML, C, Python o Perl.

Va ser creat creient que totes les dades poden ser representades adequadament com a combinacions de llistes hashes i dades escalars. La sintaxi per tant és relativament senzilla és molt llegible i a la vegada fàcil de mapejar les dades més comunes de la majoria de llenguatges.

El llenguatge YAML l'utilitzem en el projecte per poder mapejar la informació de cada un dels menús de help, per després mostrar el contingut dels fitxers YAML en components Java.

LaTeX

LaTeXés un sistema de composició de textos, orientat a la creació de documents escrits que presenten una alta qualitat tipogràfica. Per les seves característiques i possibilitats, és utilitzat especialment per generar articles i llibres científics, ja que aquests solen incloure expressions matemàtiques.

En el projecte fem ús del llenguatge LaTeXper poder fer display de fórmules matemàtiques que volem representar en Java, juntament amb la biblioteca de Java (JLatexMath) que ens permet representar codi LaTeXen components de Java.

8 Anàlisi i disseny del sistema

En aquest apartat es realitzarà una anàlisi dels requisits funcionals del sistema mitjançant els diagrames de casos d'ús i les seves corresponents fitxes. Aquesta anàlisi ens servirà per estudiar de manera detallada les necessitats del sistema.

Per altra part, presentarem el disseny del sistema, mostrant les interfícies d'usuari, els packages utilitzats de Java i els seus respectius diagrames UML. Aquest disseny del sistema proposa una solució per al sistema.

8.1 Anàlisi dels requisits del sistema

Un cop fet l'apartat **[6]** on definim els requisits funcionals del sistema, a continuació mitjançant diagrames de casos d'ús, il·lustrarem d'alguna manera de forma gràfica la funcionalitat del sistema i com l'usuari interacciona amb ell. Juntament amb el diagrama de cas d'ús trobarem també la seva corresponent fitxa de cas d'ús per complementar la informació gràfica d'aquest. Mitjançant els diagrames de casos d'ús podem observar la seqüència d'accions que ocorren entre el sistema i l'usuari quan aquest iniciar una acció en el sistema.

8.1.1 Actors del sistema

Anomenem actor a tota entitat externa al sistema que té relació amb aquest i li demana funcionalitats. Això inclou actors humans però també pot incloure sistemes externs o entitats abstractes com el temps.

En el cas del projecte només tenim un actor que apareix en la Figura 30 i que correspon al que ve a ser l'usuari final de l'aplicació, aquest usuari pot ser qualsevol persona que disposi del software i l'executi correctament.

Figura 30: Actors del sistema.

8.1.2 Parts del sistema

A continuació en aquest apartat el que farem és comentar cada una de les pantalles en les quals l'usuari pot interactuar quan executa l'aplicació i accedeix al sistema. Cal comentar que una gran part important del CoDaPack ja estava fet i el que s'ha fet és afegir noves funcionalitats.

Les parts del sistema són les següents:

- **Principal:** Pantalla principal de l'aplicació, és la primera finestra que veu l'usuari a l'obrir l'aplicació.
  En aquesta pantalla trobem quatre parts:
  - **Menús interactius:** Ho trobem en la part superior i ens permet veure tots els menús disponibles per executar rutines o accions.

  - **Marc de dades actives:** Ho trobem en la part esquerra i conté les dades actives del moment i els seus corresponents noms de les seves variables.
  - **Consola de sortida:** En la part superior dreta trobem la consola de sortida on es mostren els resultats alfanumèrics obtinguts de les rutines o accions que realitzem amb l'aplicació.
  - **Taula de dades:** En la part inferior de la dreta, trobem la taula de dades que conté les dades de l'arxiu carregat actualment. Cal comentar que la taula ens permet modificar directament valors de les variables o canviar el nom d'una variable.

- **File chooser:** En aquesta pantalla el que es permet és poder seleccionar un fitxer per poder obrir o importar un arxiu o ve un directori per poder guardar un workspace o un gràfic... A més a més en alguns casos tenim diferents formats d'importació o exportació.

- **Configuration:** Aquesta pantalla el que ens permet és poder modificar alguns paràmetres de l'aplicació i així no haver d'utilitzar els paràmetres que venen per defecte. Alguns paràmetres com poden ser: el caràcter del decimal, el format de sortida o el format de la taula. Aquesta pantalla a més ens permet guardar els nous paràmetres i posar-los per defecte en l'aplicació o simplement guardar-ho per la sessió actual.

- **Menú amb opcions:** Aquesta pantalla ens permet realitzar una rutina d'un menú on ens apareixen les variables del data frame actual a l'esquerra, després a la dreta apareixen les variables seleccionades per realitzar la rutina. Després finalment trobaríem l'apartat d'opcions per cada una de les rutines en concret.
  Finalment trobaríem a la part dreta inferior una opció de help que ens portaria a una altra finestra.

- **Variable selector:** Aquesta pantalla ens permet simplement l'opció de seleccionar una o més d'una variable per realitzar una acció amb aquesta o aquestes com pot ser per exemple esborrar la/les variable/les.
  També trobem el seu corresponen botó de help per portar-nos a l'altra pantalla.

- **Help:** La pantalla de help és una pantalla on es mostra una informació relacionada amb la rutina que va associat el help com pot ser el funcionament o les opcions que componen la rutina perquè l'usuari la realitzi correctament i amb plena consciència del que fa i com ho fa.

- **Create new table:** La pantalla per crear una nova taula doncs ens permet poder crear el nostre propi data frame d'una forma manual sense haver d'importar cap arxiu de dades, es pot tant anar entrant els elements del data frame de forma manual com també es pot fer un copy and paste típic per copiar dades en aquest.
  També trobem el seu corresponen botó de help per portar-nos a l'altra pantalla.

- **Add numeric variables:** Aquesta pantalla ens permet importar les dades directament al conjunt de dades mitjançant un input manual o amb una simple acció de copy paste.
  També trobem el seu corresponent botó de help per portar-nos a l'altra pantalla.

- **Set partition:** Ens permet poder definir una partició ILR donat un conjunt de variables, aquesta definició de la partició pot ser de manera per defecte o de manera manual que obriria una nova finestra que comentem a continuació en el següent punt.

- **Define manually the partition:** La finestra ens permet definir manualment una partició ILR donat un conjunt de variables.

- **Graphics:** Les finestres de gràfics ens apareixen al realitzar una rutina que inclou un gràfic com per exemple pot ser la rutina *Boxplot*. També en tots els tipus de finestra de gràfics tenim l'opció de poder exportar els gràfics generats en diferents formats com por exemple el format SVG.

- **About:** La pantalla d'about simplement el que fa és mostrar un seguit d'informació referent a l'equip del CoDaPack que pot servir per exemple a què l'usuari pugui contactar en el cas que trobés algun bug en l'aplicació poder informar-ne al respecte.

8.1.3 Diagrames i fitxes de casos d'ús

En el següent apartat es mostren els diagrames de casos d'ús i les seves fitxes de casos d'ús associades de les diferents parts del sistema.

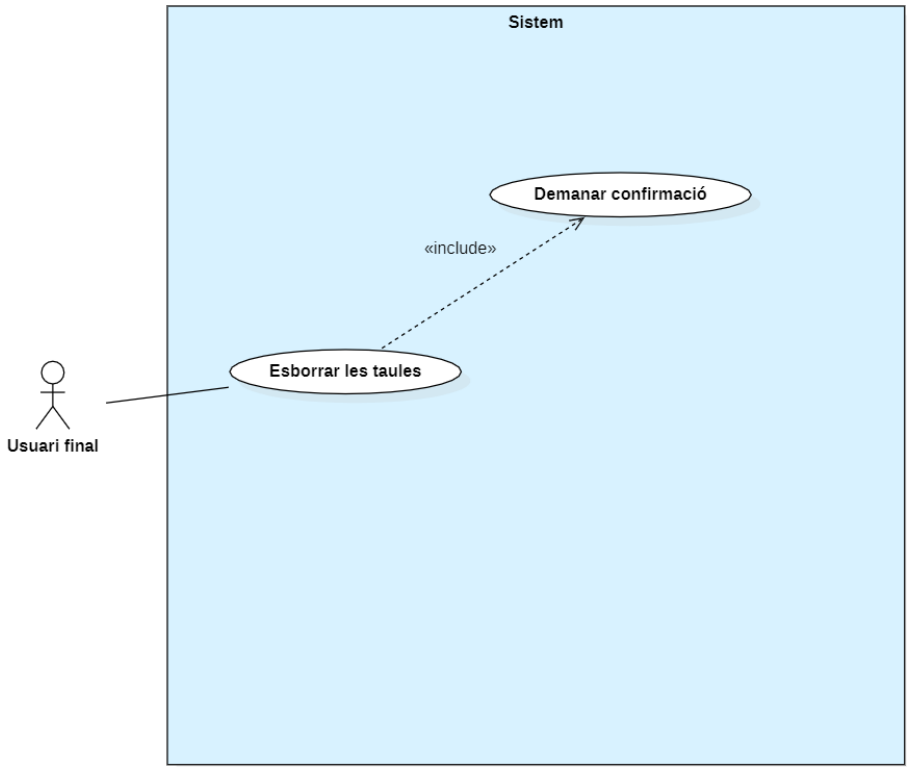


Figura 31: Diagrama de cas d'ús d'esborrar taules.

Casos d'ús	**Esborrar taules**
Descripció:	Usuari pot esborrar les taules que hi han carregades al sistema.
Actors:	Usuari final.
Precondició:	--
Flux principal:	1. L'usuari fa clic en el menú de "*File*". 2. L'usuari selecciona el submenú "*Delete All Tables*". 3. El sistema mostra una alerta per confirmar si l'usuari vol realment esborrar totes les taules o no. 4. L'usuari selecciona la opció "*Sí*". 5. El sistema mostra la pàgina inicial.
Flux alternatiu:	En el cas de que l'usuari seleccioni la opció de "*No*" en la confirmació, es torna a la pàgina inicial sense esborrar cap taula. En el cas de que no hi hagi cap taula carregada el sistema no esborrarà ras.
Postcondició:	S'han esborrat correctament totes les taules que hi havia carregades al sistema.

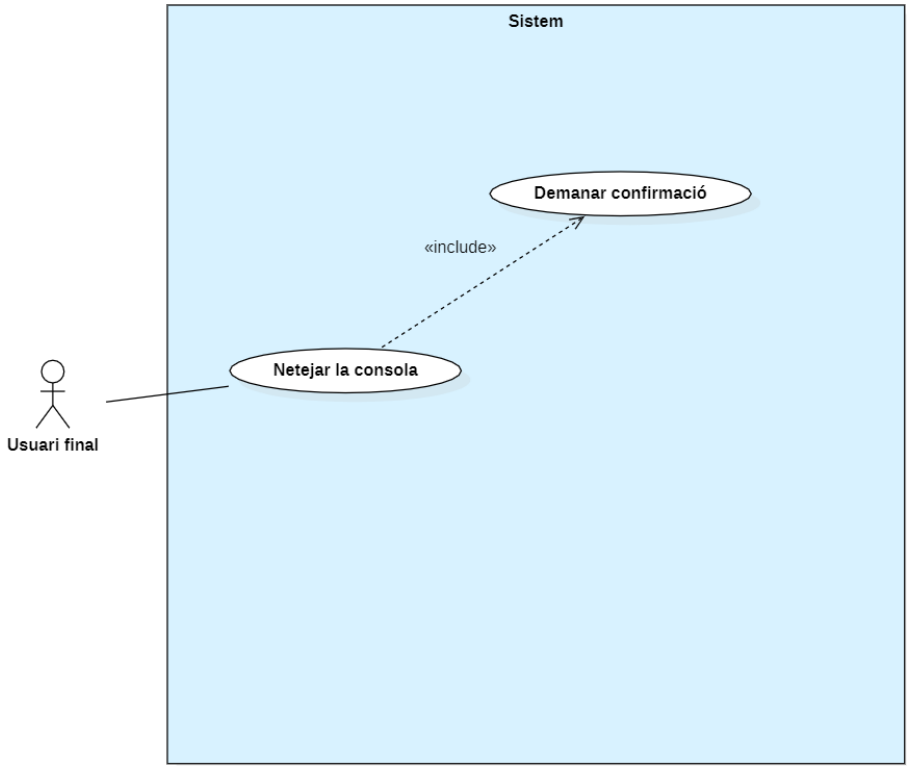


Figura 32: Diagrama de cas d'ús de netejar la consola.

Casos d'ús	**Netejar la consola de sortida**
Descripció:	Usuari pot netejar la consola de sortida de missatges.
Actors:	Usuari final.
Precondició:	--
Flux principal:	1. L'usuari fa clic en el menú de "*File*". 2. L'usuari selecciona el submenú "*Clear the output*". 3. El sistema mostra una alerta per confirmar si l'usuari vol realment netejar la consola o no. 4. L'usuari selecciona la opció "*Sí*". 5. El sistema mostra la pàgina inicial.
Flux alternatiu:	En el cas de que l'usuari seleccioni la opció de "*No*" en la confirmació, es torna a la pàgina inicial sense netejar la consola. En el cas de que no hi hagi missatge en la consola, la consola quedarà en el mateix estat.
Postcondició:	S'han netejat correctament la consola de missatges i només hi apareix el missatge inicial.

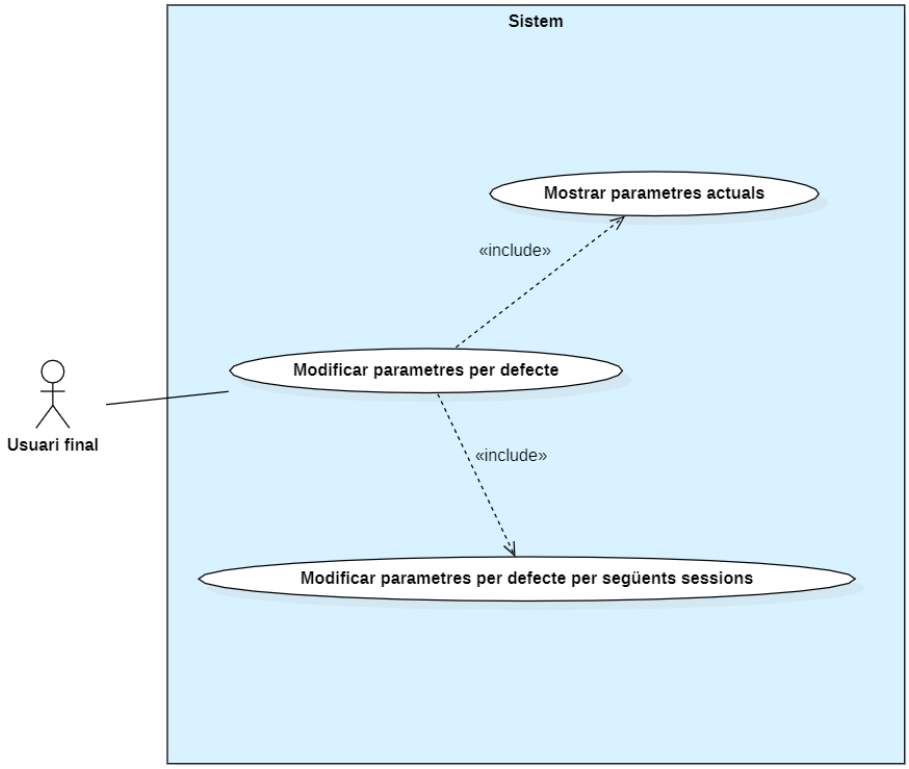


Figura 33: Diagrama de cas d'ús de modificar paràmetres per defecte de l'aplicació.

Casos d'ús	**Modificar paràmetres per defecte de l'aplicació**
Descripció:	Usuari pot modificar els paràmetres per defecte de l'aplicació.
Actors:	Usuari final.
Precondició:	Els paràmetres introduïts per l'usuari han de ser correctes i coherents al sistema.
Flux principal:	1. L'usuari fa clic en el menú de "*File*". 2. L'usuari selecciona el submenú "*Configuration*". 3. El sistema mostra un formulari amb els diferents paràmetres de l'aplicació. 4. L'usuari introdueix correctament i coherentment les dades que vol modificar en el sistema. 5. L'usuari selecciona "*Apply*" o "*Save as default*" segons la opció que vulgui. 6. El sistema mostra la pàgina inicial.
Flux alternatiu:	En el cas de que l'usuari seleccioni la opció de "*Apply*", es modifiquen els paràmetres per defecte del sistema per la sessió actual. En el cas de que l'usuari seleccioni "*Save as default*", es modifiquen els paràmetres per defecte en la sessió actual i en les que vinguin.
Postcondició:	S'han modificat els paràmetres per defecte de l'aplicació correctament per la sessió actual o per la sessió actual i les següents.

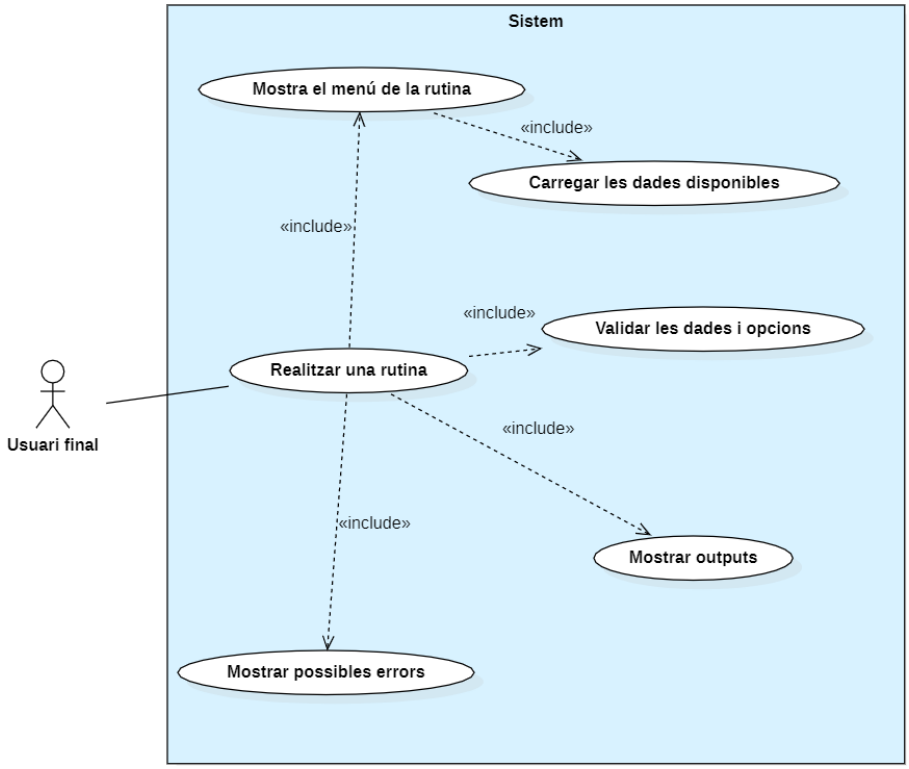


Figura 34: Diagrama de cas d'ús de realitzar una rutina.

Casos d'ús	Realitzar una rutina
Descripció:	Usuari pot realitzar una rutina en el sistema.
Actors:	Usuari final.
Precondició:	S'han de carregar dades amb prou informació en el sistema per poder realitzar la rutina que es vol fer.
Flux principal:	1. L'usuari fa clic en la menú de la rutina que vol realitzar. 2. El sistema mostra el menú corresponent a la rutina que s'ha seleccionat anteriorment. 3. L'usuari selecciona les variables amb les que vol realitzar les rutines i les opcions que vol utilitzar si es que es dona el cas. 4. El sistema comprova que les dades introduïdes per l'usuari i les opcions que ha seleccionat són valides per procedir a executar la rutina. a. Si les dades són valides llavors podem passar al punt nº5. b. Si les dades no són valides llavors hem de tornar al punt nº4. 5. El sistema realitza la rutina. 6. El sistema torna a la pàgina principal però retorna els resultats de la rutina que poden ser els següents: a. Sortides en la consola d'outputs. b. Sortides gràfiques amb finestres independents. c. Sortides en forma de noves variables en la taula de dades.
Flux alternatiu:	En el cas de que hi hagi algun error en la execució de la rutina per exemple amb R perquè hi ha un error en l'script llavors el que es fa es mostrar un error en la consola d'output informant-ne al respecte.
Postcondició:	S'han realitzat correctament la rutina que ha seleccionat l'usuari amb les variables seleccionades i les opcions indicades si es el cas.

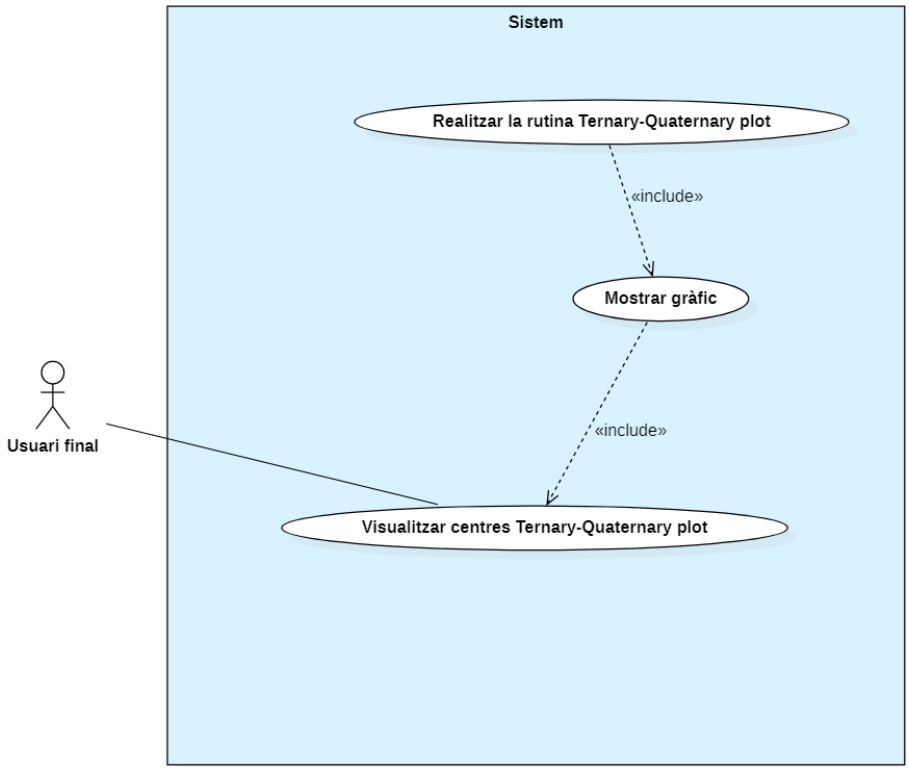


Figura 35: Diagrama de cas d'ús de visualitzar centres ternary/quaternary plot.

Casos d'ús	**Visualitzar els centres ternary/quaternary plot**
Descripció:	Usuari pot visualitzar els centres en els gràfics ternary/quaternary plot
Actors:	Usuari final.
Precondició:	Hi ha d'haver unes dades amb prou informació en el sistema.
Flux principal:	1. L'usuari fa clic en el menú de "*Graphs*". 2. L'usuari selecciona el submenú "Ternary/Quaternary plot". 3. El sistema mostra una un menú per poder seleccionar les dades per la realització de la rutina. 4. L'usuari selecciona tres o quatre variables disponibles i si vol també pot seleccionar una variable que representi els grups. 5. El sistema mostra el gràfic amb opcions per l'usuari. 6. L'usuari selecciona la opció "*Show the center*". 7. El sistema mostra els centres en el gràfic que s'està mostrant.
Flux alternatiu:	--
Postcondició:	ES mostra correctament el centre en els gràfics generats amb la rutina "*Ternary/Quaternary plot*".

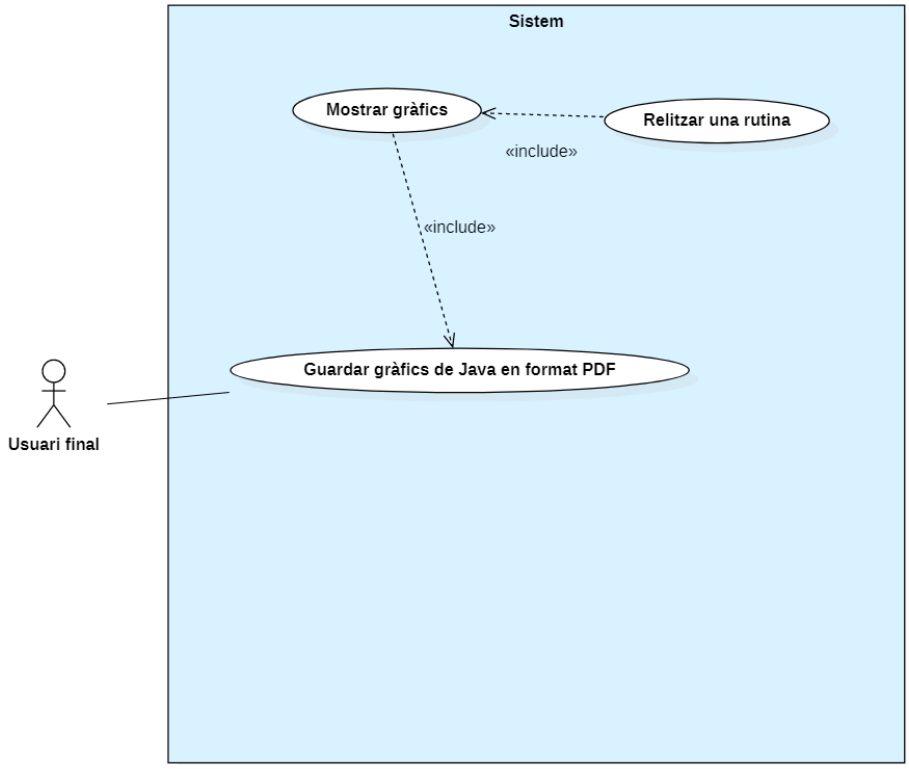


Figura 36: Diagrama de cas d'ús d'exportar gràfic generat amb Java en PDF.

Casos d'ús	**Guardar els gràfics generats amb Java en format PDF**
Descripció:	Usuari pot exportar els gràfics generats amb Java amb el format PDF.
Actors:	Usuari final.
Precondició:	S'han de carregar dades amb prou informació en el sistema.
Flux principal:	1. L'usuari genera un gràfic amb una rutina feta amb Java. 2. L'usuari fa clic en el menú "*File*". 3. L'usuari selecciona el submenú "*Save as*". 4. El sistema mostra un file chooser perquè l'usuari defineixi els paràmetres per exportar el gràfic. 5. L'usuari selecciona on vol exportar el gràfic amb el seu corresponent nom i a continuació selecciona com a tipus d'arxiu la opció "*PDF file*". 6. L'usuari fa clic en el botó "*Guardar*". 7. El sistema torna a mostrar el gràfic generat.
Flux alternatiu:	En el cas de que l'usuari seleccioni la opció de "Cancel" en el file chooser, es torna al gràfic generat sense exportar el gràfic en cap format.
Postcondició:	S'ha exportat correctament el gràfic generat amb Java amb el format PDF on l'usuari a seleccionat i amb el nom que ha definit.

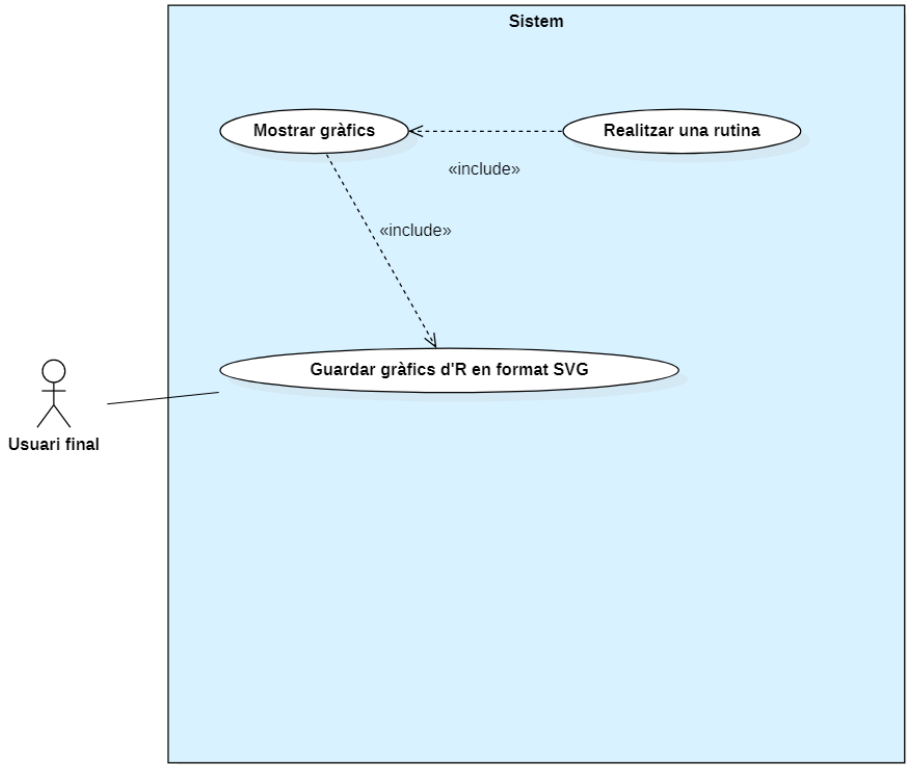


Figura 37: Diagrama de cas d'ús d'exportar gràfic generat amb R en SVG.

Casos d'ús	**Guardar els gràfics generats amb R en format SVG**
Descripció:	Usuari pot exportar els gràfics generats amb R amb el format SVG.
Actors:	Usuari final.
Precondició:	S'han de carregar dades amb prou informació en el sistema.
Flux principal:	1. L'usuari genera un gràfic amb una rutina feta amb R. 2. L'usuari fa clic en el menú "*File*". 3. L'usuari selecciona el submenú "*Export*". 4. L'usuari selecciona el submenú "*Export As SVG*". 5. El sistema mostra un file chooser perquè l'usuari defineixi els paràmetres per exportar el gràfic. 6. L'usuari selecciona on vol exportar el gràfic amb el seu corresponent nom. 7. L'usuari fa clic en el botó "*Guardar*". 8. El sistema torna a mostrar el gràfic generat.
Flux alternatiu:	En el cas de que l'usuari seleccioni la opció de "Cancel" en el file chooser, es torna al gràfic generat sense exportar el gràfic en cap format.
Postcondició:	S'ha exportat correctament el gràfic generat amb R amb el format SVG on l'usuari a seleccionat i amb el nom que ha definit.

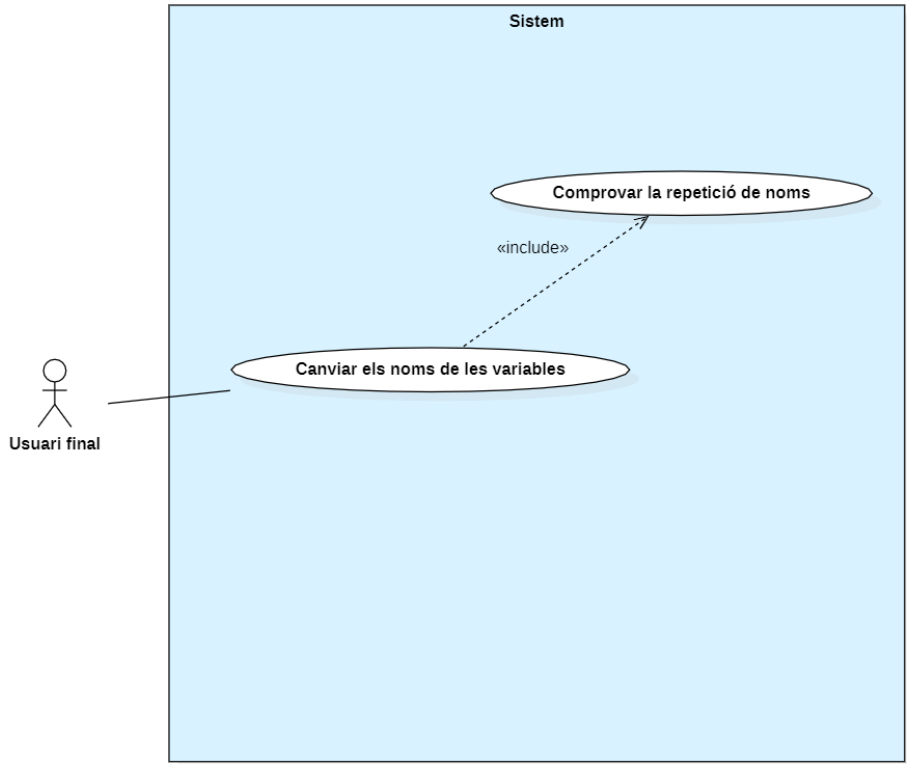


Figura 38: Diagrama de cas d'ús de canviar els noms de les variables en la taula.

Casos d'ús	**Canviar els noms de les variables en la taula**
Descripció:	Usuari pot canviar el nom de les variables directament de la taula de dades.
Actors:	Usuari final.
Precondició:	S'han de carregar dades en el sistema.
Flux principal:	1. L'usuari fa doble clic en un nom d'una variable en la taula de dades. 2. L'usuari escriu el nou nom de la variable. 3. Al acabar de posar el nom fa enter amb el teclat. 4. Es sistema mostra la taula amb el nou nom de la variable modificada.
Flux alternatiu:	En el cas de que el nom estigui repetit el sistema informa del problema i no modifica el nom de la variable.
Postcondició:	S'han netejat correctament la consola de missatges i només hi apareix el missatge inicial.

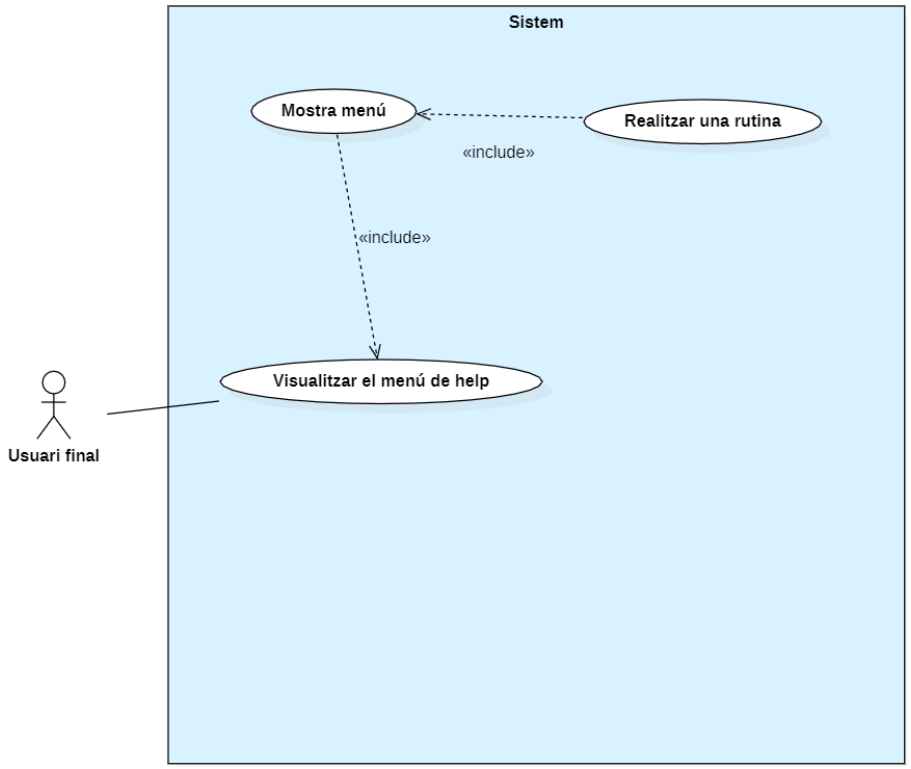


Figura 39: Diagrama de cas d'ús de visualitzar menú de help.

Casos d'ús	**Visualitzar el menú de help en les rutines**
Descripció:	Usuari pot visualitzar menús d'ajuda en les rutines quan ho vegi oportú.
Actors:	Usuari final.
Precondició:	S'han de carregar dades en el sistema.
Flux principal:	1. L'usuari fa clic en un dels menús per realitzar una rutina. 2. El sistema mostra el menú corresponent a la rutina seleccionada. 3. L'usuari fa clic en el botó on posa "*Help*" que normalment apareix en la part inferior a la dreta. 4. El sistema mostra el menú d'ajuda corresponent a la rutina que esta realitzant l'usuari en aquell moment. 5. En tancar el menú torna a aparèixer el menú de la rutina corresponent.
Flux alternatiu:	--
Postcondició:	S'ha mostrat correctament el menú d'ajuda associat al menú de la rutina que vol realitzar l'usuari en aquell moment.

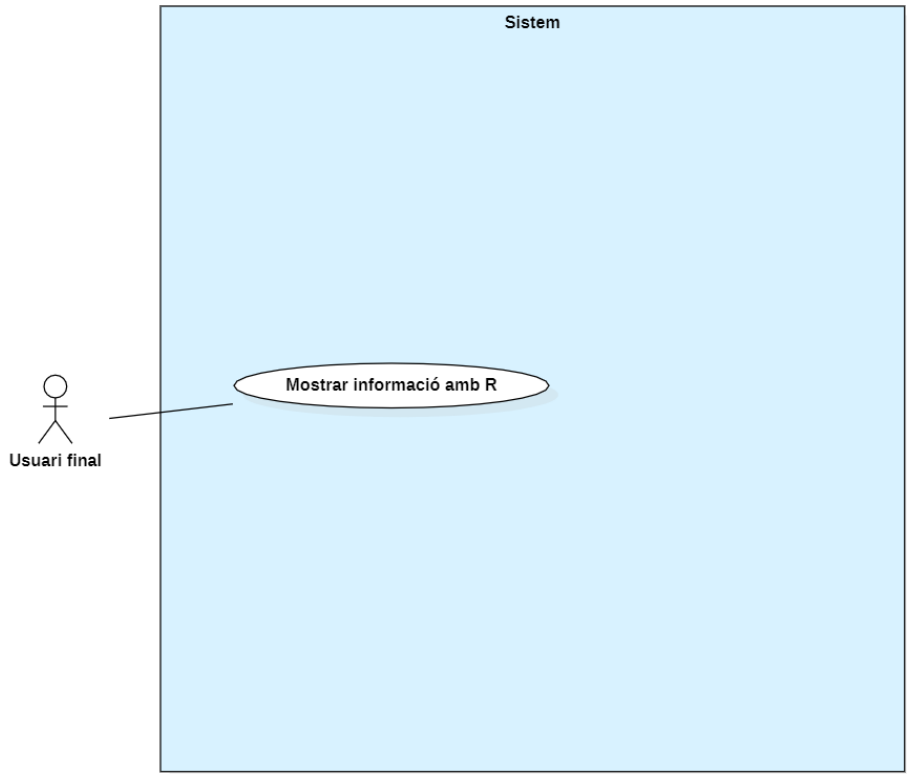


Figura 40: Diagrama de cas d'ús de mostrar informació amb R.

Casos d'ús	**Mostrar informació amb R**
Descripció:	Usuari pot visualitzar informació del sistema generada amb R.
Actors:	Usuari final.
Precondició:	--
Flux principal:	1. L'usuari fa clic en el menú "*Help*". 2. L'usuari selecciona el submenú "*Get R status*". 3. El sistema mostra informació obtinguda amb R en la sortida d'outputs en la pàgina principal.
Flux alternatiu:	--
Postcondició:	Es mostra correctament informació obtinguda d'R com: versió d'R, el sistema operatiu, els paquets carregats, les variables d'entorn...

8.2 Disseny del sistema

En aquest apartat procedirem a mostrar el disseny utilitzat per al desenvolupament del sistema. Primer de tot mostrarem les interfícies d'usuari i a continuació comentarem els packages utilitzats juntament amb els seus corresponents diagrames UML. Cal especificar que només es detallarà el que s'ha fet durant el projecte i hi han parts ja fetes anteriorment del projecte que es passaran per alt.

8.2.1 Interfícies d'usuari

A continuació, es mostraran els diferents dissenys d'interfícies d'usuari del projecte, cal comentar que moltes interfícies ja estaven definides en el programa inicial, per tant durant l'explicació de cada una d'elles comentarem canvis que s'hagin pogut produir durant el desenvolupament del projecte o comentar alguns dissenys que siguin nous.

La interfície d'usuari cal comentar que sempre s'ha d'intentar que sigui intuïtiva i simple d'utilitzar per a l'usuari final del producte, sense oblidar-nos de què amb ella es puguin realitzar totes les accions que siguin possible amb el potencial del sistema per complir amb els objectius. A continuació per tant cada una de les pantalles amb una petita descripció:

Finestres referents als menús amb opcions

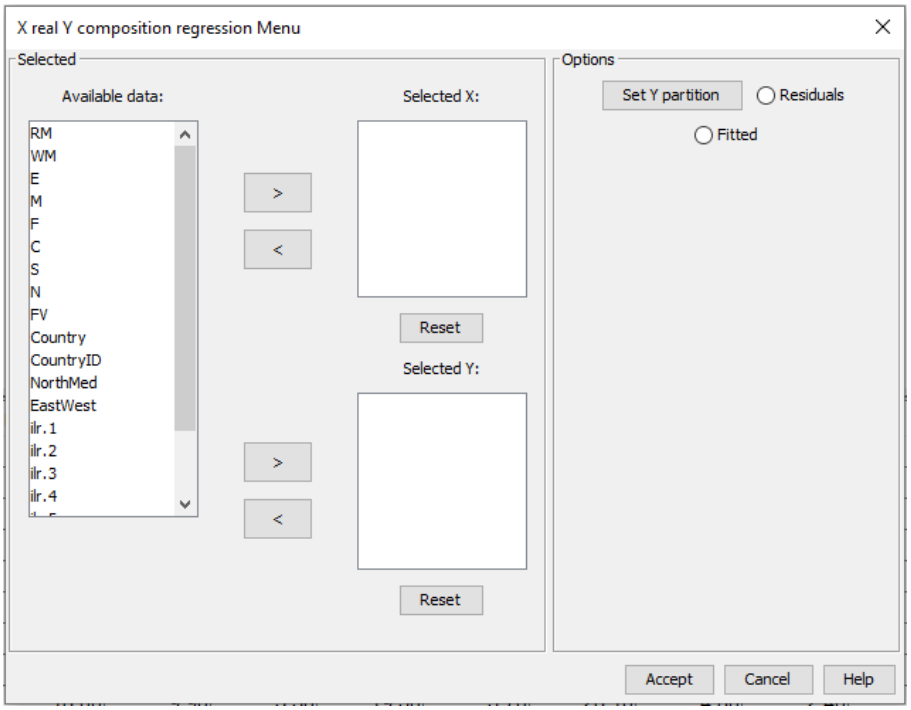


Figura 41: Pantalla de menú amb opcions.

En la Figura 41 podem observar el disseny dels menús amb opcions, el disseny ja estava fet en l'aplicació inicial, però s'han creat alguns nous menús amb dos selectors de variables en comptes d'un. Després els canvis que es produeixin de disseny de cada rutina dependrà de les opcions que hi hagi en la part dreta d'opcions segons convingui.

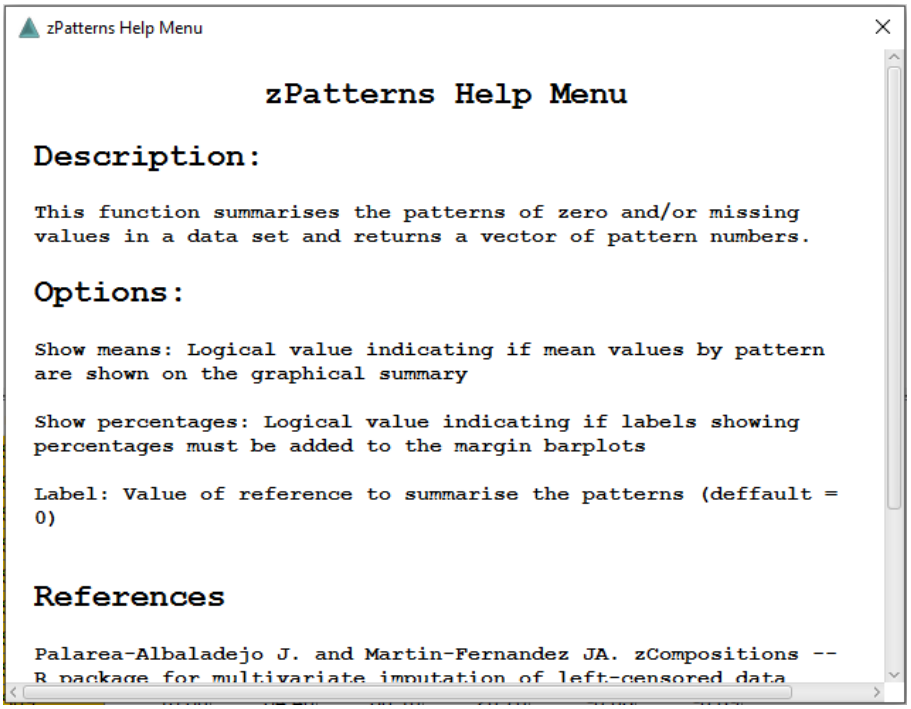


Figura 42: Pantalla del help menú.

En la Figura 42 podem observar el disseny del menú de help, aquest disseny és nou, ja que és una pantalla nova i un nou menú, el disseny simplement consisteix en un panell on es mostra informació sobre una rutina en concret, com per exemple: la descripció, les opcions o les referències.

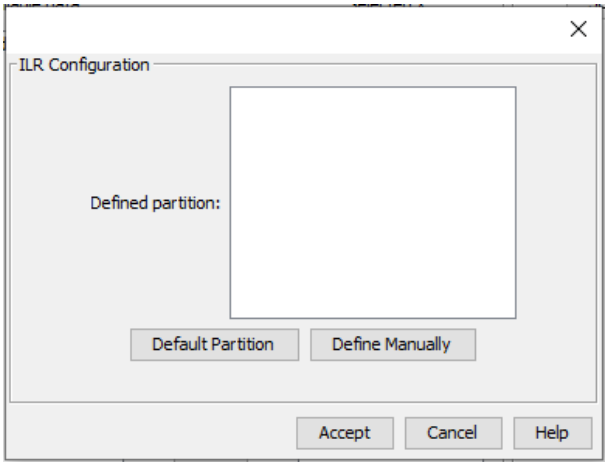


Figura 43: Pantalla per afegir una partició.

En la Figura 43 podem observar el disseny de la pantalla per poder crear noves taules des de zero, aquest disseny és nou, ja que és una pantalla nova.
El disseny consisteix en un panell on es mostra la partició, i dos botons tant per crear la partició de manera automàtica (Default Particion) com per crear la partició de manera manual que ens portaria a una nova finestra. Finalment en la part inferior trobem els típics botons d'acceptar, cancel·lar i de help.

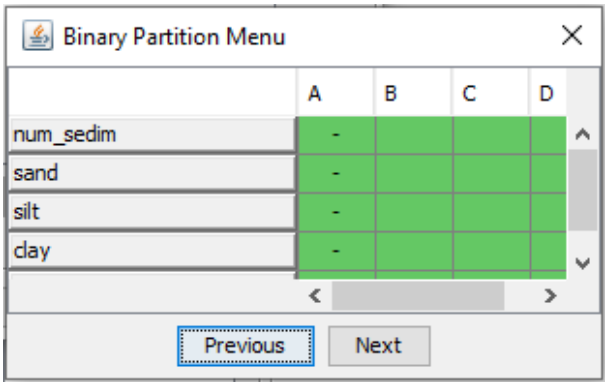


Figura 44: Pantalla per crear una partició manualment.

En la Figura 44 podem observar el disseny de la pantalla per poder crear una partició de forma manual, el disseny ja estava fet en l'aplicació inicial.
El disseny consisteix en una quadrícula on les files representen les parts i les columnes representen els passos de la partició.
També podem observar dos botons en la part inferior, el de "Next"que ens permetrà reordenar tota la informació (Etiquetes i partició) de tal forma que en les següents parts a dividir ens apareixen en una seqüència. En canvi l'altre botó "Previous", ens permet eliminar alguns passos de la partició.

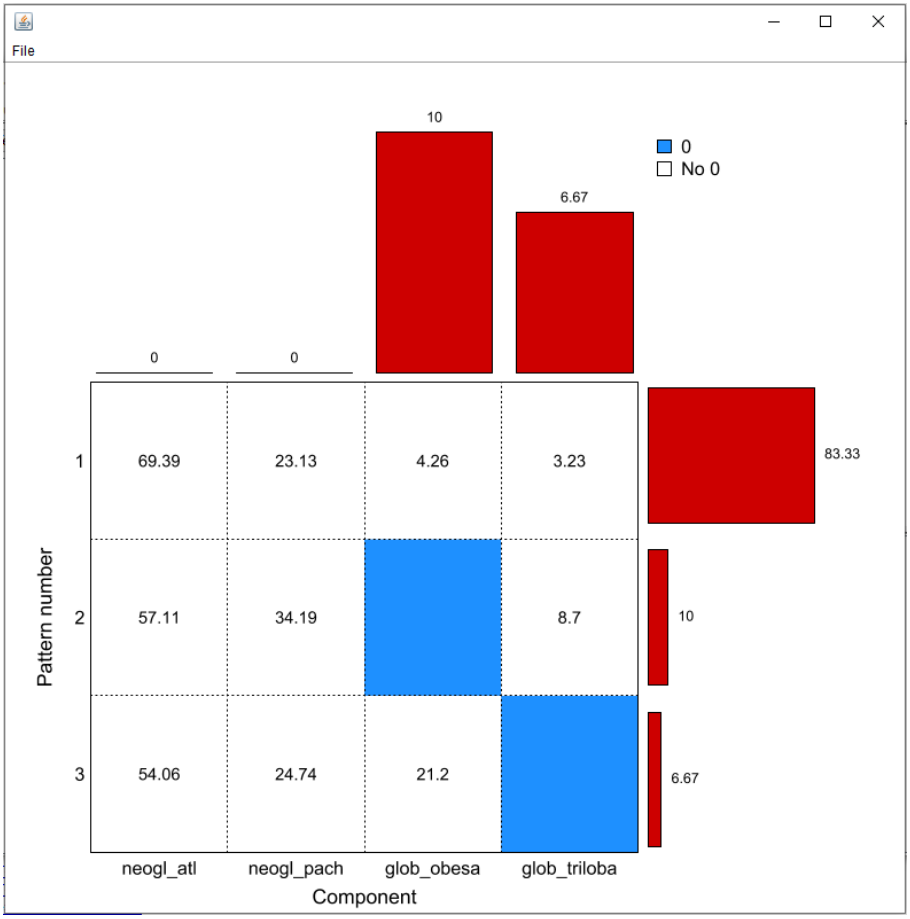


Figura 45: Pantalla de gràfics.

En la Figura 45 podem observar el disseny de la pantalla de gràfics, el disseny ja estava fet en l'aplicació inicial per gràfics en Java, però no per gràfics en R, en aquest cas de la imatge seria el nou disseny per als gràfics generats amb R. El disseny com podem observar consisteix simplement d'un panell amb el gràfic dibuixat.
També en la part superior trobem un menú per obtenir opcions a fer amb el gràfic com per exemple poder exportar el gràfic.

Finestres d'altres rutines sense menú amb opcions

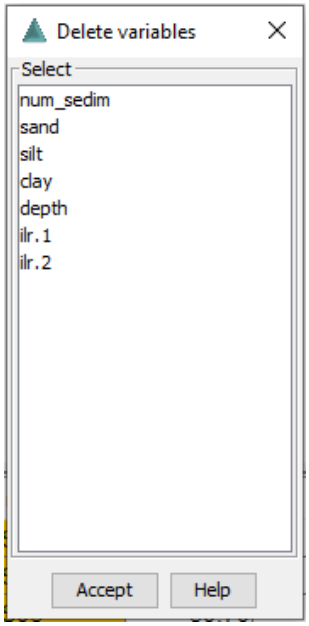


Figura 46: Pantalla del variable selector.

En la Figura 46 podem observar el disseny del variable selector, el disseny ja estava fet en l'aplicació inicial, però s'ha afegit un nou element que podem observar en la part inferior a la dreta. Consisteix en un botó que mostra el menú de help. La resta del disseny consisteix en mostrar un conjunt de variables per poder seleccionar-les.

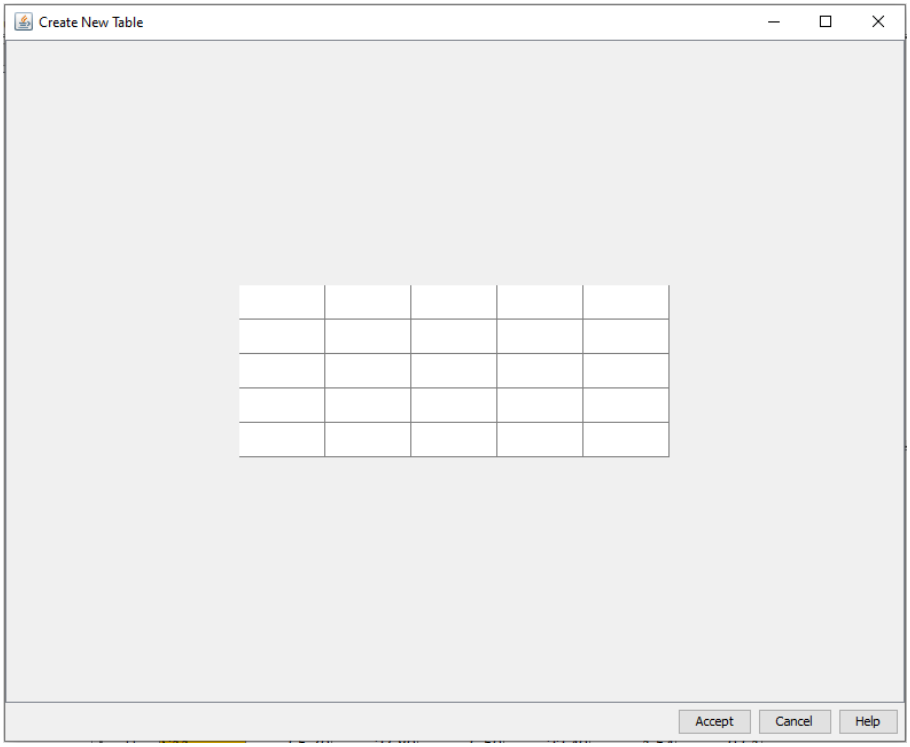


Figura 47: Pantalla per crear una taula nova.

En la Figura 47 podem observar el disseny de la pantalla per poder crear noves taules des de zero, aquest disseny és nou, ja que és una pantalla nova i un nou menú, el disseny consisteix en un panell amb una taula buida que representarà el nou data frame, on l'usuari pot posar informació en cada una de les cel·les. En la part inferior trobem els típics botons d'acceptar, cancel·lar i de help.

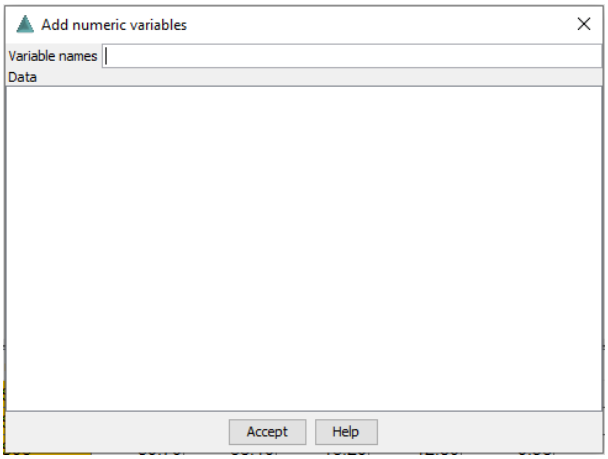
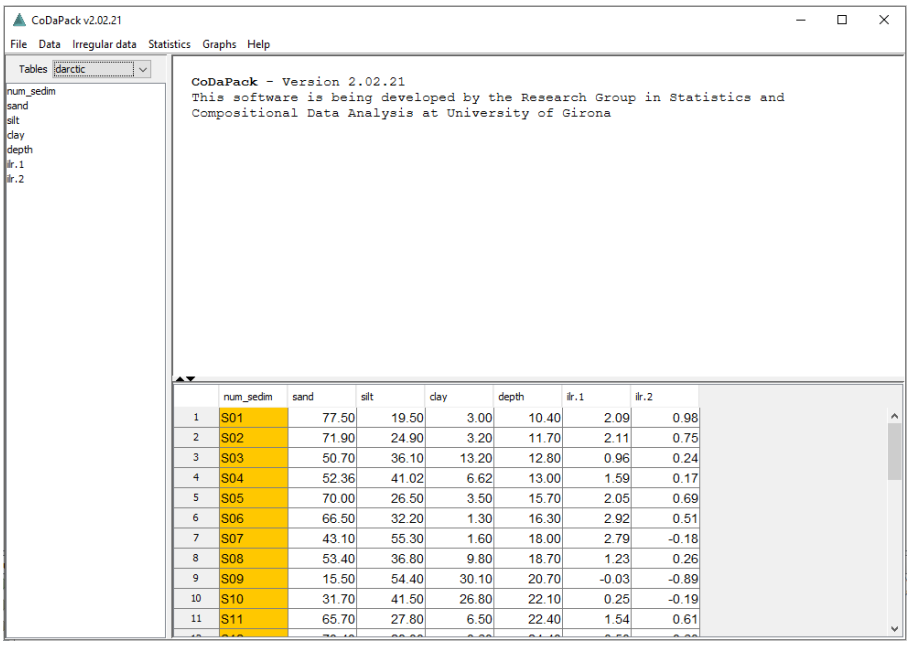


Figura 48: Pantalla per afegir noves variables numèriques.

En la Figura 48 podem observar el disseny de la pantalla per poder afegir noves variables numèriques, el disseny ja estava fet en l'aplicació inicial.
El disseny consisteix en un input de text per obtenir els noms de les variables, i a sota trobem un requadre de text molt més gran per poder introduir les dades referents a les respectives variables que hem afegit

a la part de dalt.
Comentar també que s'ha afegit el botó de help per mostrar l'ajuda de la rutina.

Altres finestres



	num_sedim	sand	silt	clay	depth	ilr.1	ilr.2
1	S01	77.50	19.50	3.00	10.40	2.09	0.98
2	S02	71.90	24.90	3.20	11.70	2.11	0.75
3	S03	50.70	36.10	13.20	12.80	0.96	0.24
4	S04	52.36	41.02	6.62	13.00	1.59	0.17
5	S05	70.00	26.50	3.50	15.70	2.05	0.69
6	S06	66.50	32.20	1.30	16.30	2.92	0.51
7	S07	43.10	55.30	1.60	18.00	2.79	-0.18
8	S08	53.40	36.80	9.80	18.70	1.23	0.26
9	S09	15.50	54.40	30.10	20.70	-0.03	-0.89
10	S10	31.70	41.50	26.80	22.10	0.25	-0.19
11	S11	65.70	27.80	6.50	22.40	1.54	0.61

Figura 49: Pantalla de la pàgina principal.

En la Figura 49 podem observar la finestra principal, el disseny ja estava fet en l'aplicació inicial, per tant contenia ja les 4 parts: menús interactius, sortida d'outputs, taula amb les dades i informació de variables a l'esquerra.

S'han produït alguns canvis de disseny en els menús interactius, ja que s'han reorganitzat en grups i s'han afegit noves rutines. També s'ha modificat el panell de sortida però el disseny segueix sent el mateix.

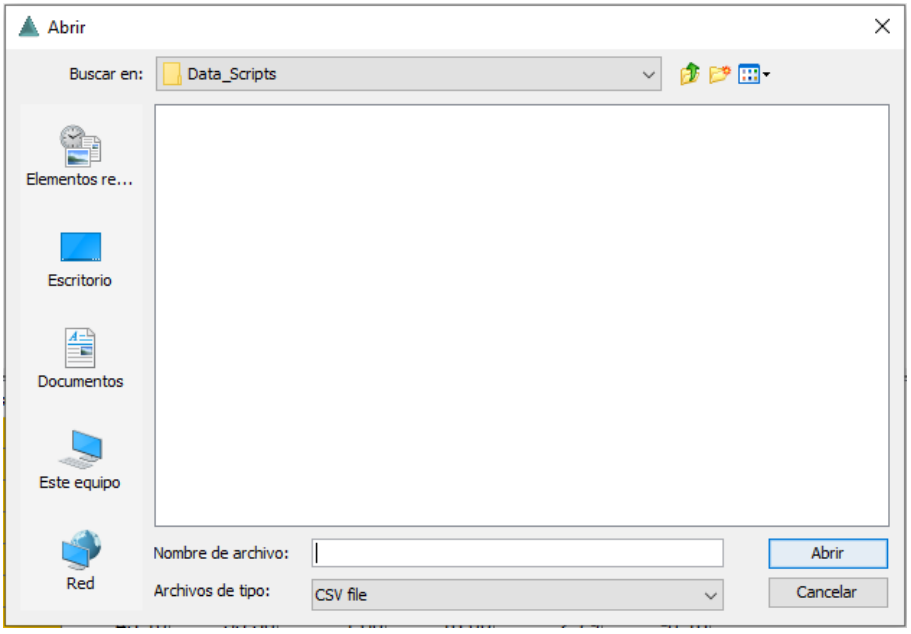


Figura 50: Pantalla del file chooser.

En la Figura 50 podem observar el disseny del file chooser, el disseny ja estava fet en l'aplicació inicial, aquest disseny és el disseny per defecte que utilitza la llibreria de Java per poder seleccionar fitxers de l'ordinador o seleccionar rutes de directori.

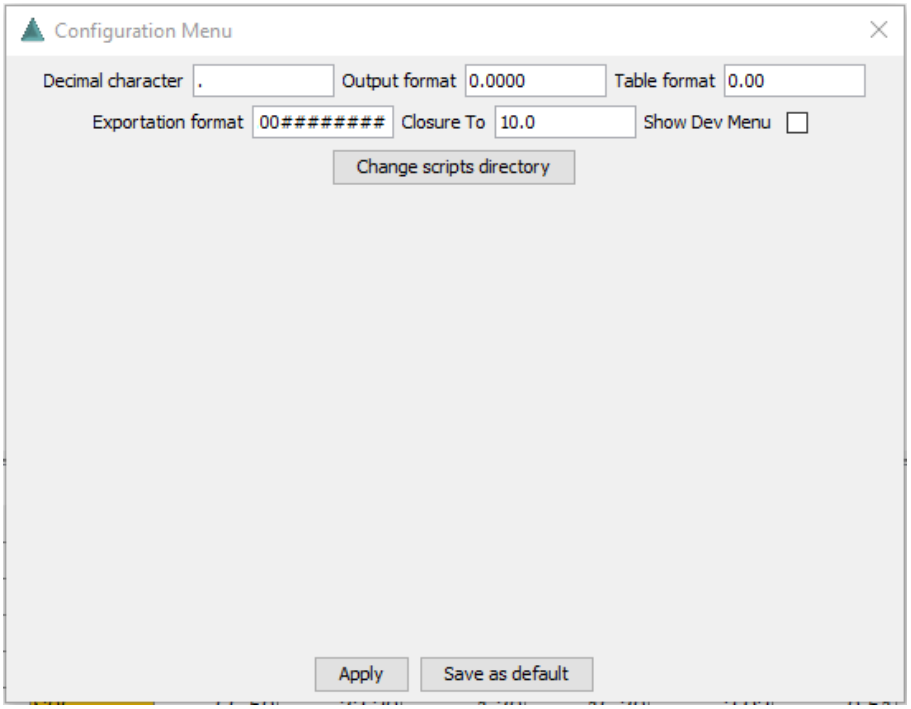


Figura 51: Pantalla de configuration.

En la Figura 51 podem observar el disseny de la finestra de configuració del CoDaPack, el disseny ja estava fet en l'aplicació inicial, com podem veure simplement conté un frame amb diferents elements per definir els paràmetres de l'aplicació.

S'han produït alguns canvis en el disseny, on simplement el que s'ha fet és afegir elements nous per poder modificar els seus paràmetres, com per exemple el valor per defecte del "*Closure To*".

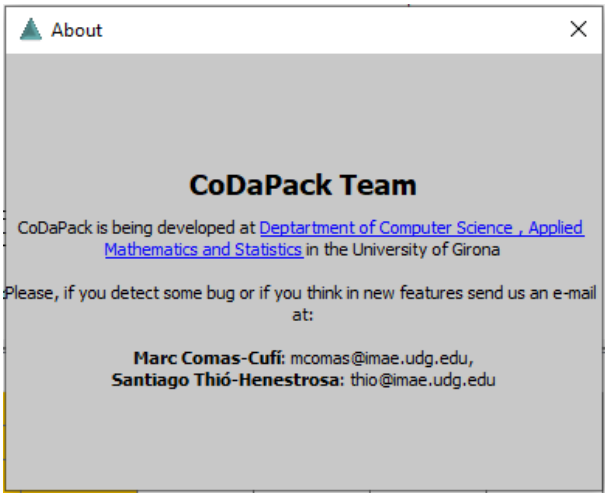


Figura 52: Pantalla d'about.

En la Figura 52 podem observar el disseny de la pantalla d'about, el disseny ja estava fet en l'aplicació inicial, consisteix en un panell on simplement apareix un seguit d'informació referent a l'aplicació per l'usuari.

8.2.2 Packages de Java

A continuació mostrarem tots els packages de Java que componen el projecte i en farem una petita descripció del que representen. Juntament amb la descripció també s'afegeix el diagrama UML corresponent, que com es podrà observar en alguns casos en concret, només apareixen les classes sense les seves funcions per manca d'espai i per pèrdua de la bona visualització. Cal remarcar que tots els packages que es mostren a continuació no s'han fet en el projecte sinó que són tots els packages que component el CoDaPack.

Package coda

- **coda:** El package coda correspon al package principal del programa i encapsula tots els package del projecte, per tant tots els package de continuació estan inclosos en aquest. A més també inclou elements bàsics del projecte com són els tipus de dades o el dataFrame. Podem observar el seu corresponent diagrama UML en la Figura 53.

Packages coda.ext

- **coda.ext.eps:** El package eps conté una llibreria externa que s'anomena EpsGraphics que és una biblioteca de Java per produir gràfics EPS. Podem observar el seu corresponent diagrama UML en la Figura 54.
- **coda.ext.jama:** El package jama conté una llibreria externa que s'anomena Jama, que és un paquet d'àlgebra lineal per Java. Proporciona classes en l'àmbit usuari per construir i manipular matrius reals i denses. Podem observar el seu corresponent diagrama UML en la Figura 55.
- **coda.ext.json:** El package json conté una llibreria externa de json per poder manipular objectes que utilitzin json. Podem observar el seu corresponent diagrama UML en la Figura 56.
- **coda.ext.triangle:** El package triangle conté una llibreria externa per poder manipular i construir triangles amb Java d'una manera fàcil i senzilla. Podem observar el seu corresponent diagrama UML en la Figura 57.

Packages coda.gui

- **coda.gui:** Conté les classes principals del gui com són la pantalla principal, el menú de configuració... Aquest package inclou altres packages de gui com són el d'output, table o menú. Podem observar el seu corresponent diagrama UML en la Figura 58.
- **coda.gui.menu:** Conté els menús de totes les rutines a més de les classes abstractes que després hereten els menús de les rutines concretes. Podem observar el corresponent diagrama UML en la Figura 59.
- **coda.gui.output:** Conté tots els tipus de sortides de la gui, com per exemple OutputForR per mostrar sortides obtingudes d'R o OutputElement com a classe abstracta. Podem observar el seu corresponent diagrama UML en la Figura 60.
- **coda.gui.table:** Conté les classes necessàries per mostrar correctament la taula de dades de la pàgina principal en la part inferior dreta. Podem observar el seu corresponent diagrama UML en la Figura 61.
- **coda.gui.utils:** El package utils del package gui conté uns sèrie de components gui que són complementaris per poder realitzar alguna tasca en concret com per exemple el tema de poder crear de manera manual les particions ILR. Podem observar el seu corresponent diagrama UML en la Figura 62.

Package coda.io

- **coda.io:** El package coda.io conté les classes necessàries per l'input i l'output de l'aplicació. Per tant s'encarrega de les importacions i exportacions que es realitzin en el sistema. Podem observar el seu corresponent diagrama UML en la Figura 63.

Packages coda.plot

- **coda.plot:** Package per mostrar gràfics en l'aplicació en Java, per exemple el gràfic Ternary Plot 2d i 3d. Podem observar el seu corresponent diagrama UML en la Figura 64.
- **coda.plot.window:** Package inclòs en el package coda.plot, aquest package conté diferents classes que representen diferents pantalles per diferents tipus de gràfics. Podem observar el seu corresponent diagrama UML en la Figura 65.

Packages coda.plot2

- **coda.plot2:** Package per mostrar gràfics en l'aplicació en Java, per exemple el gràfic Ternary Plot 2d i 3d. Podem observar el seu corresponent diagrama UML en la Figura 66.
- **coda.plot2.datafig:** Aquest package conté diferents figures per poder interactuar amb els gràfics que es generen. Podem observar el seu corresponent diagrama UML en la Figura 67.
- **coda.plot2.objects:** El següent package conté objectes per als gràfics Ternary 2D i Ternary 3D. Podem observar el seu corresponent diagrama UML en la Figura 68.
- **coda.plot2.window:**El següent package conté diferents finestres per als gràfics Ternary Plot 2D com per exemple el de Dialog Data Set. Podem observar el seu corresponent diagrama UML en la Figura 69.

Package coda.util

- **coda.util:** Package que conté només dues classes per representar tant nodes com arbres amb elements del tipus genèric. Podem observar el seu corresponent diagrama UML en la Figura 70.

Package coda

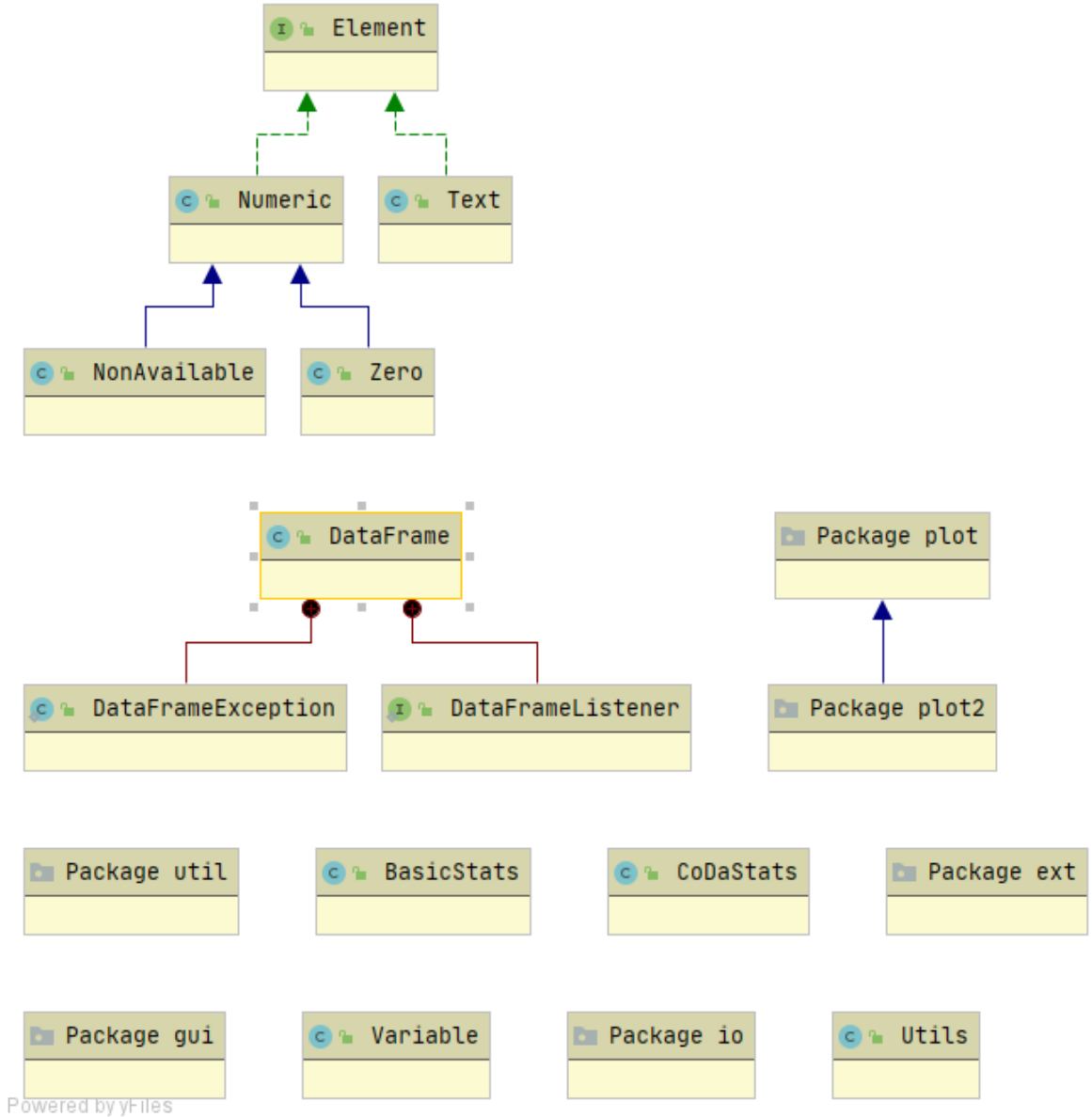


Figura 53: Diagrama de classes del package coda.

Package coda.ext.eps

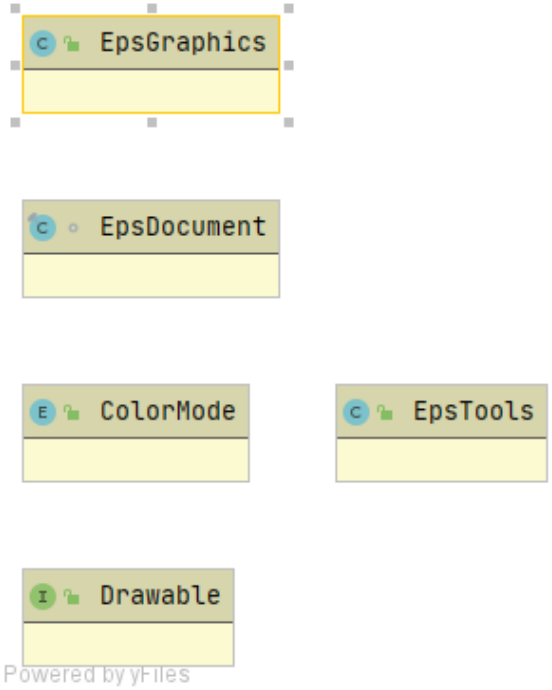


Figura 54: Diagrama de classes del package coda.ext.eps.

Package coda.ext.jama

Figura 55: Diagrama de classes del package coda.ext.jama.

Package coda.ext.json

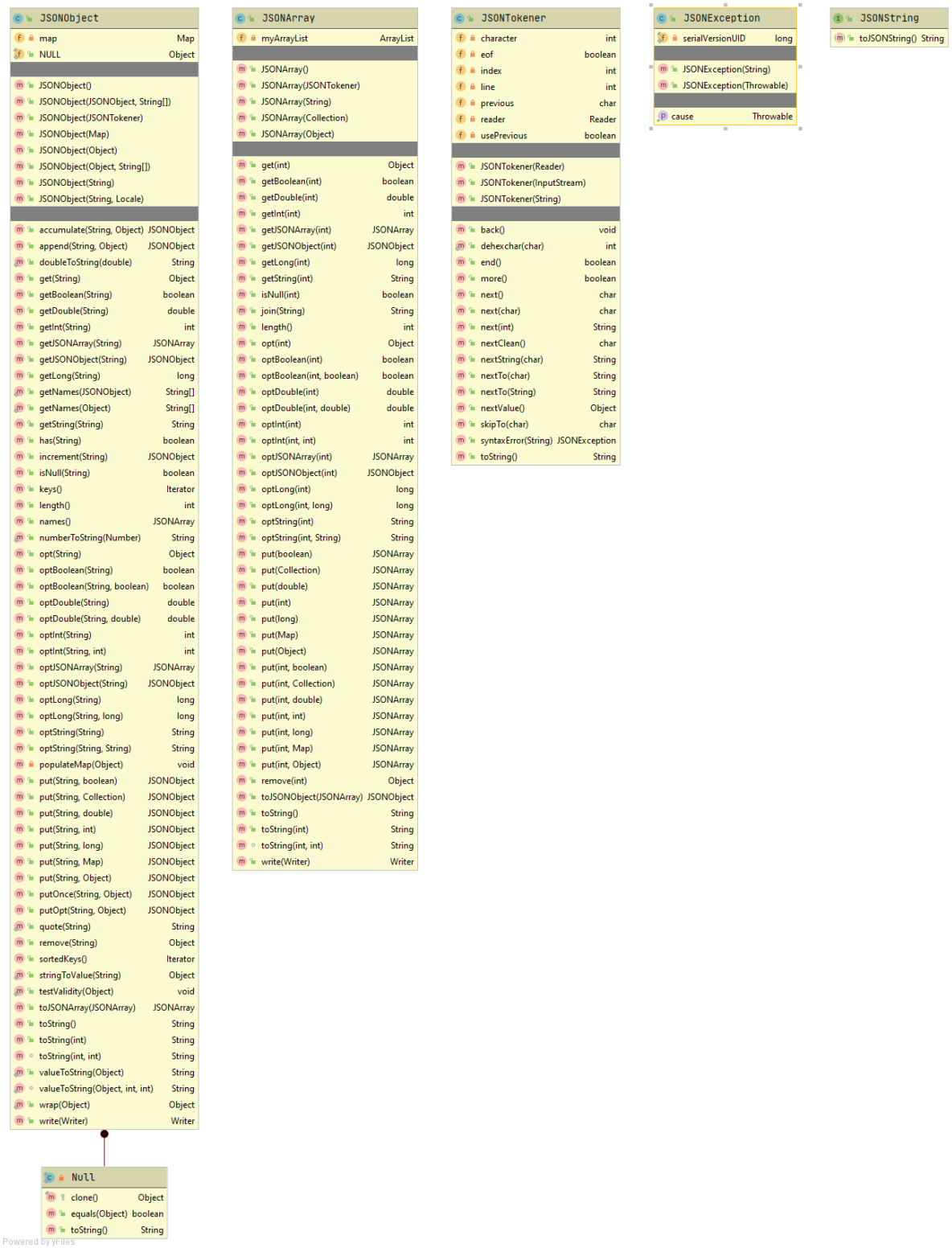


Figura 56: Diagrama de classes del package coda.ext.json.

Package coda.ext.triangle

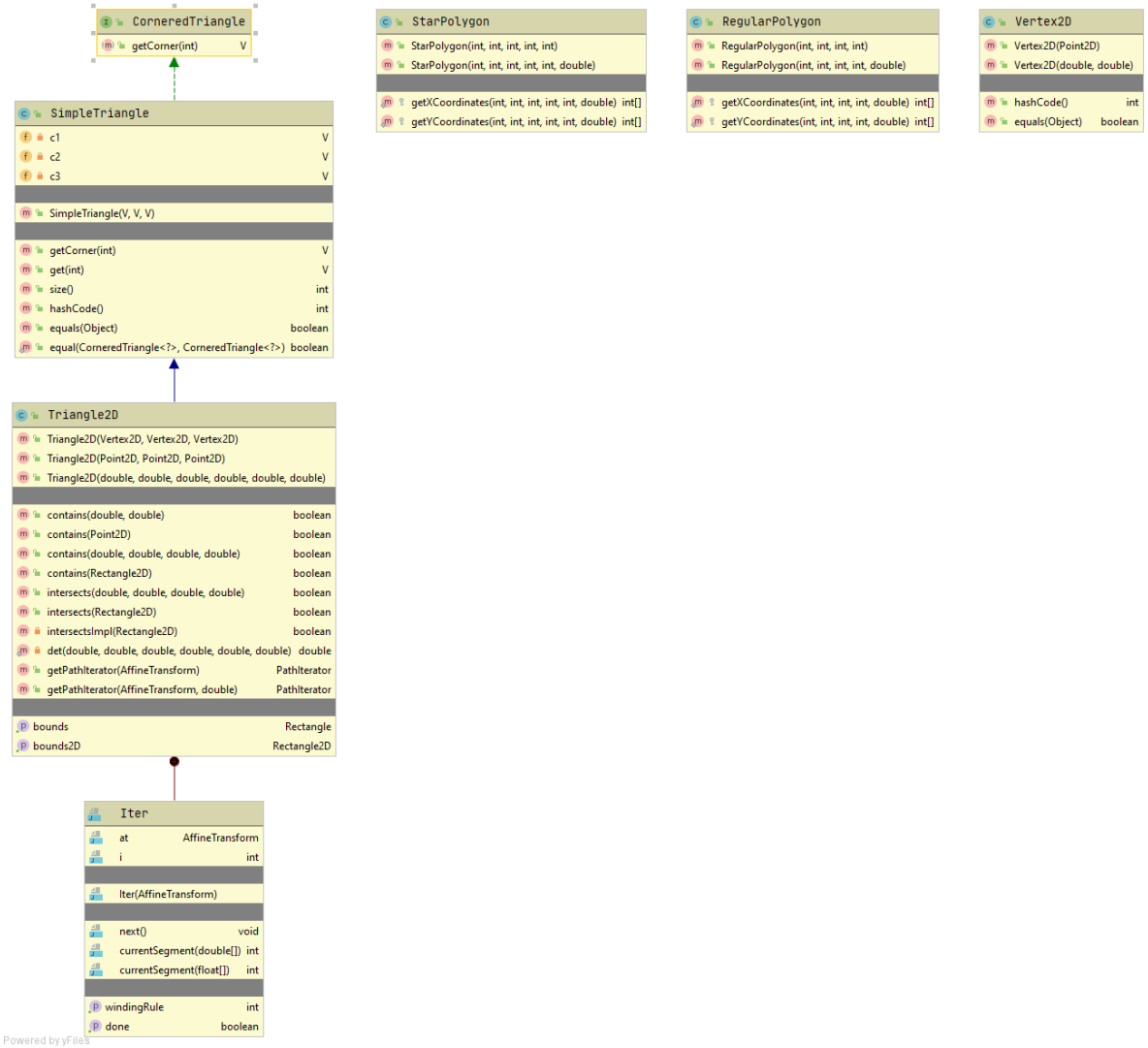


Figura 57: Diagrama de classes del package coda.ext.triangle.

Package coda.gui

Figura 58: Diagrama de classes del package coda.gui.

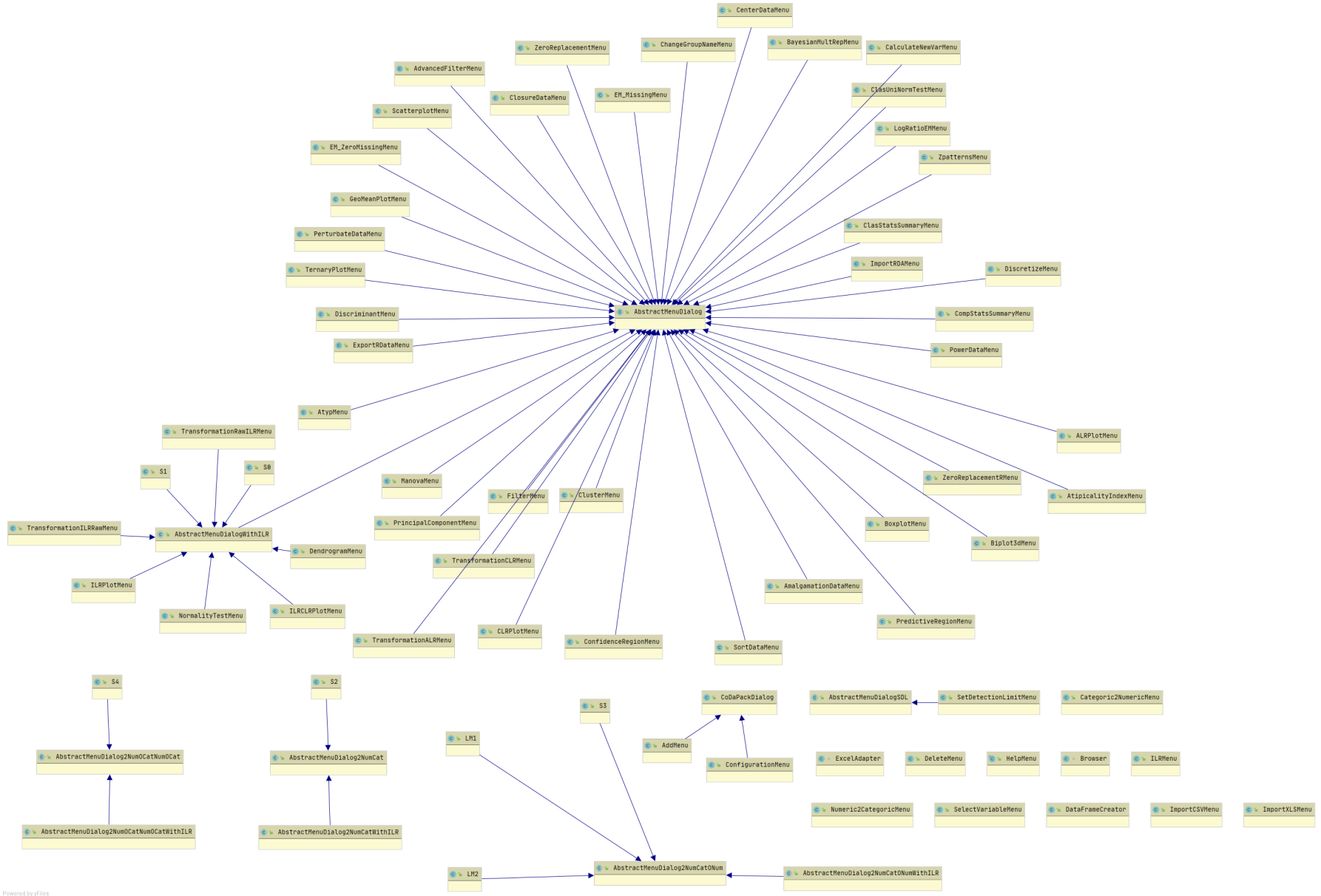


Figura 59: Diagrama de classes del package coda.gui.menu.

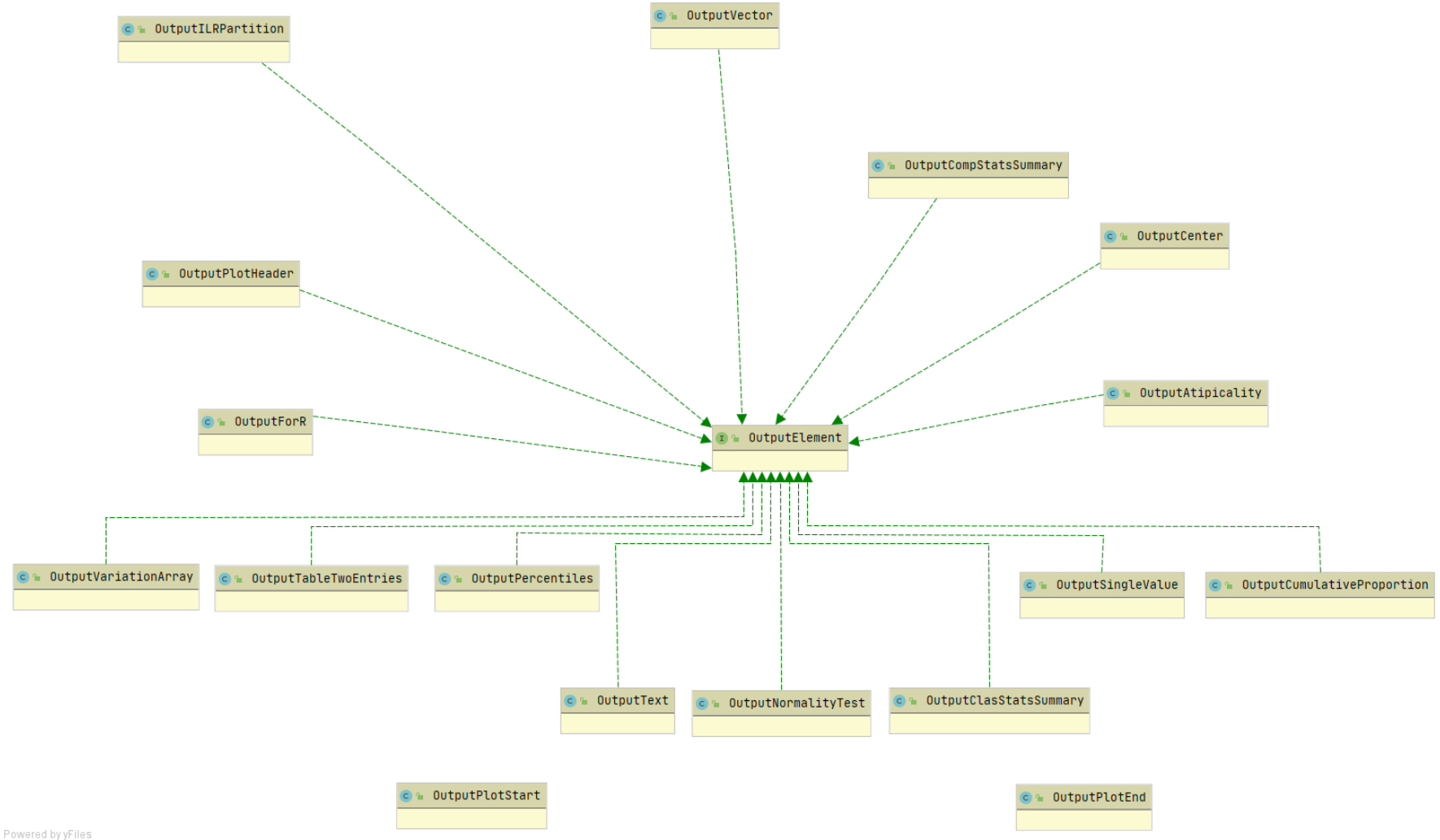


Figura 60: Diagrama de classes del package coda.gui.output.

Package coda.gui.table

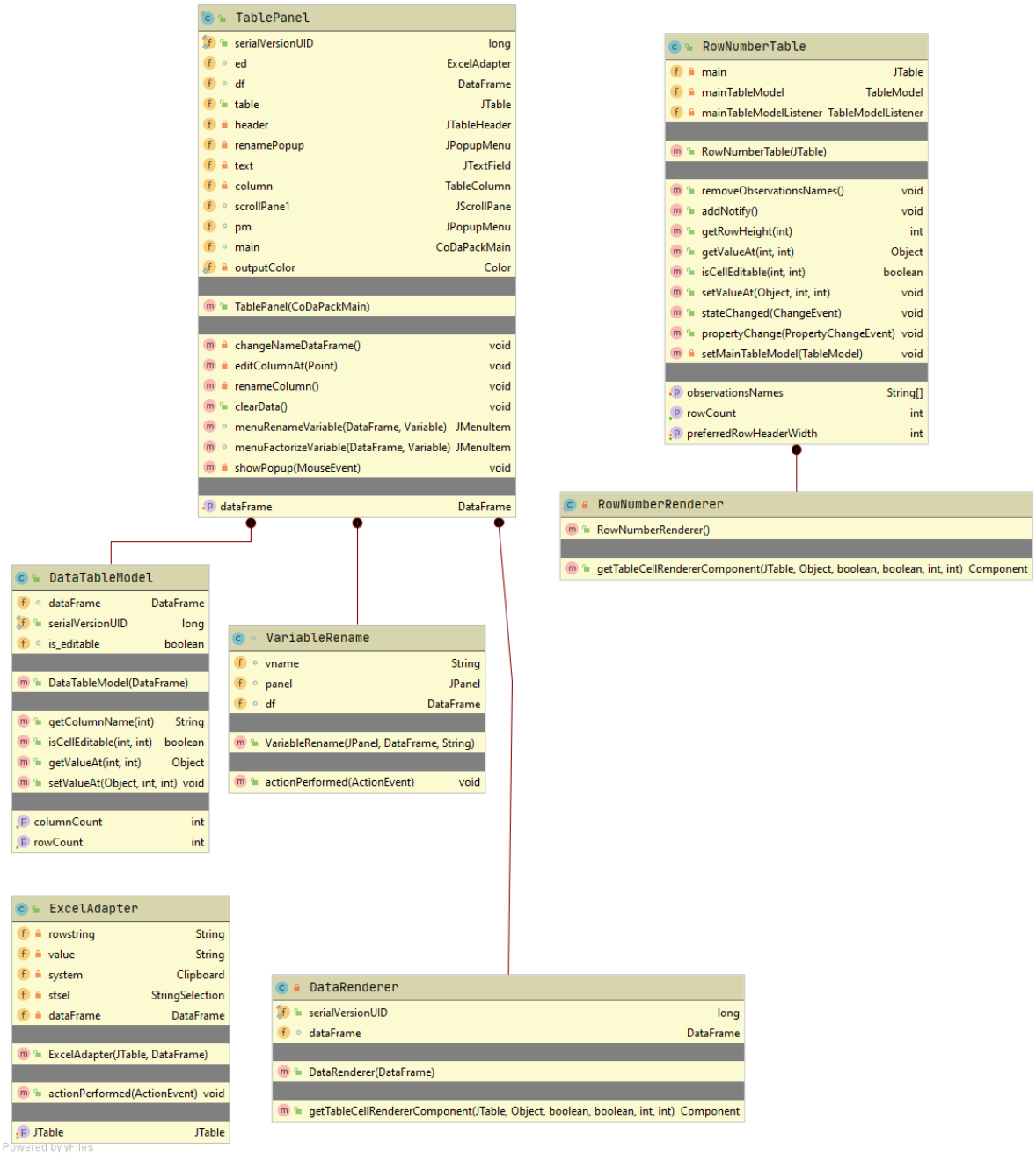


Figura 61: Diagrama de classes del package coda.gui.table.

Package coda.gui.utils

Figura 62: Diagrama de classes del package coda.gui.utils.

Package coda.io

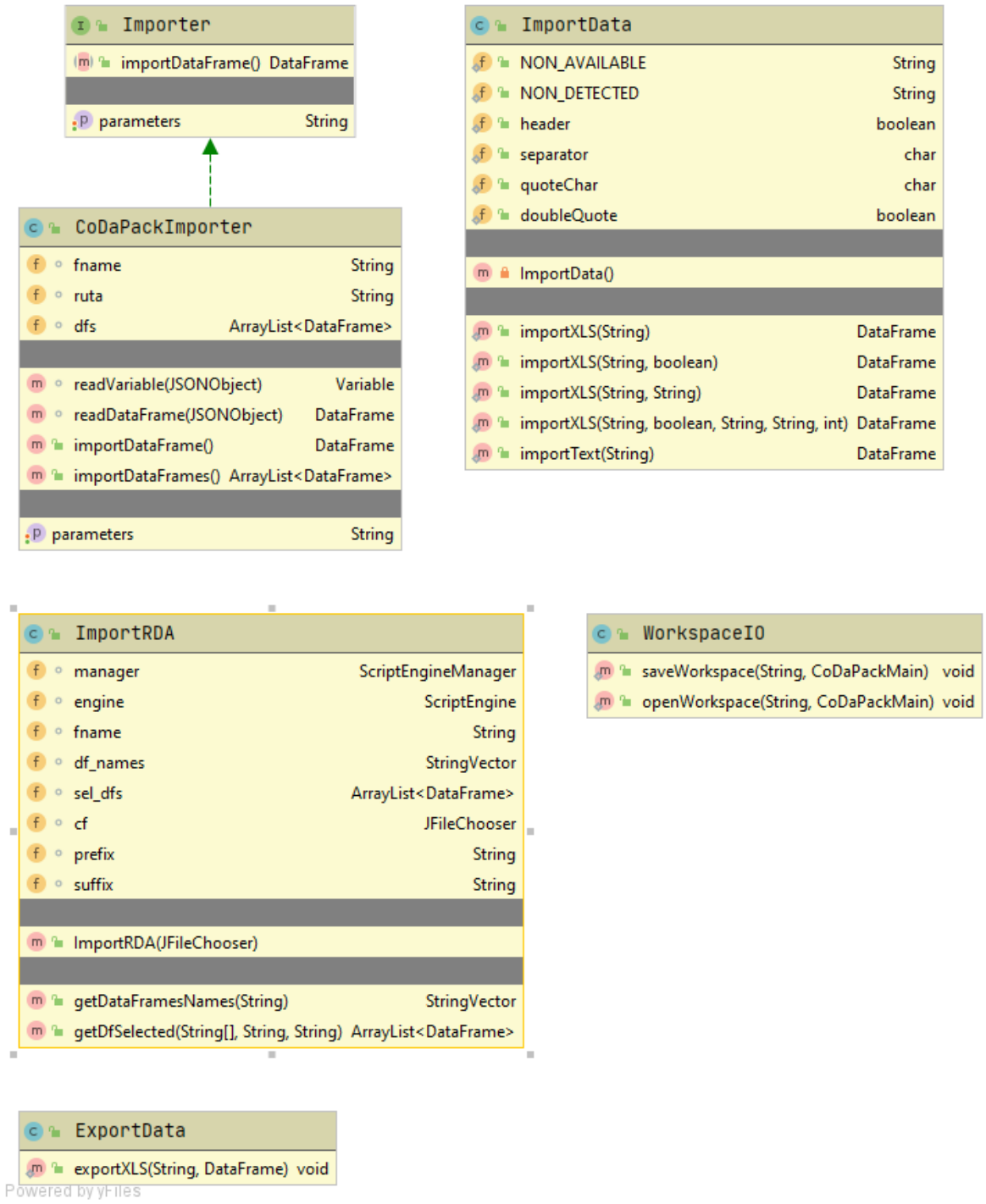


Figura 63: Diagrama de classes del package coda.io.

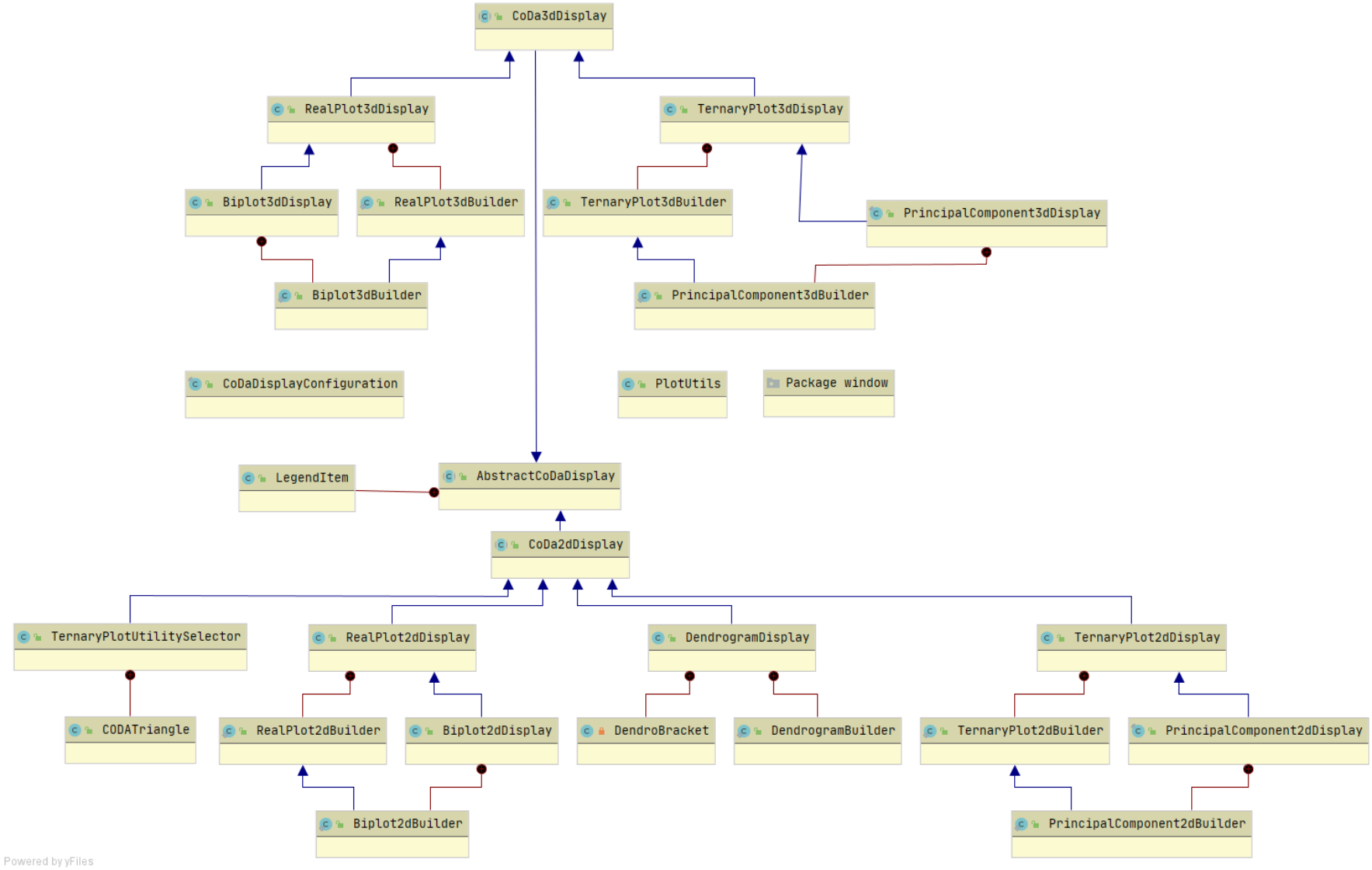


Figura 64: Diagrama de classes del package coda.plot.

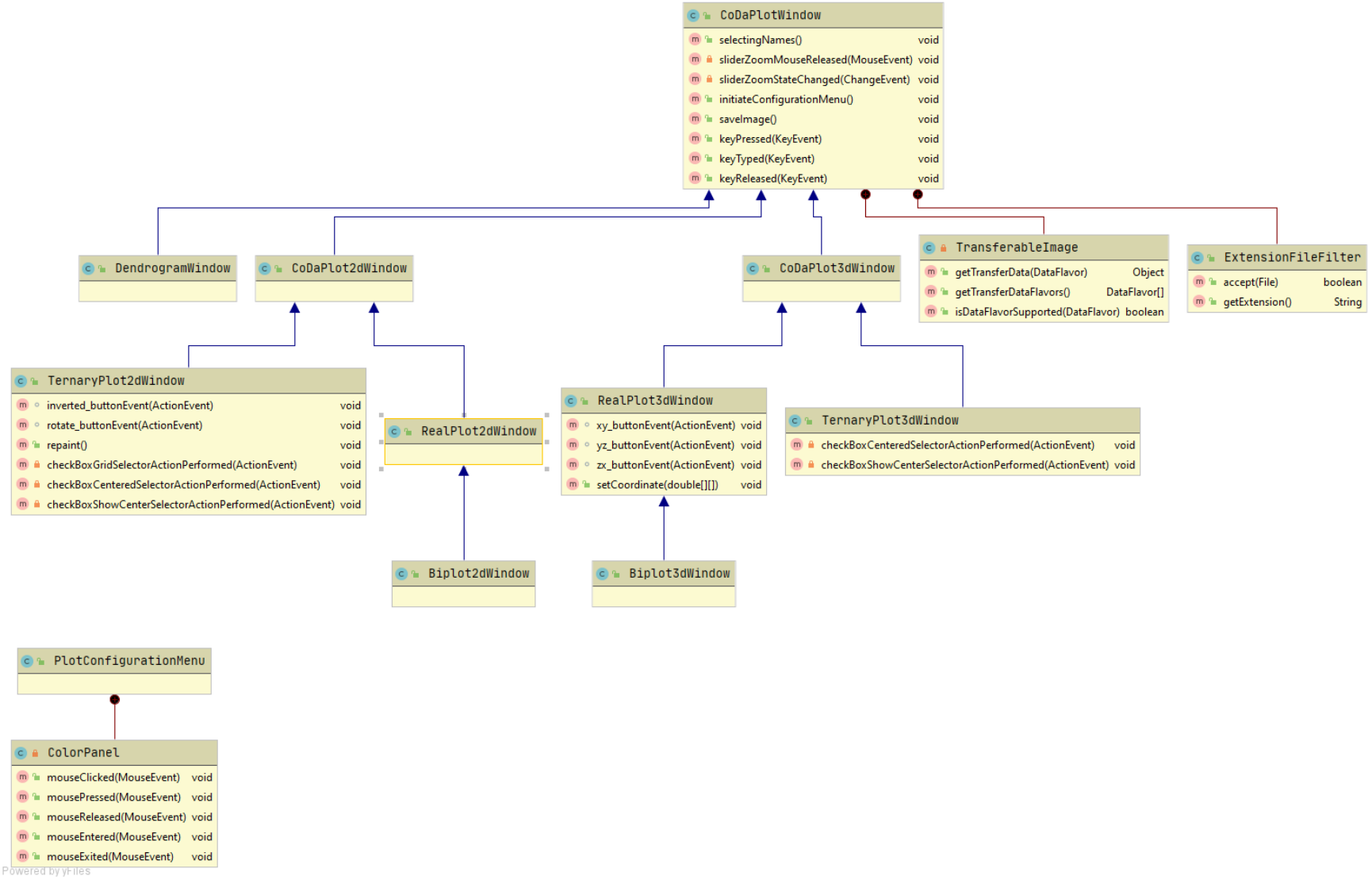


Figura 65: Diagrama de classes del package coda.plot.window.

Package coda.plot2

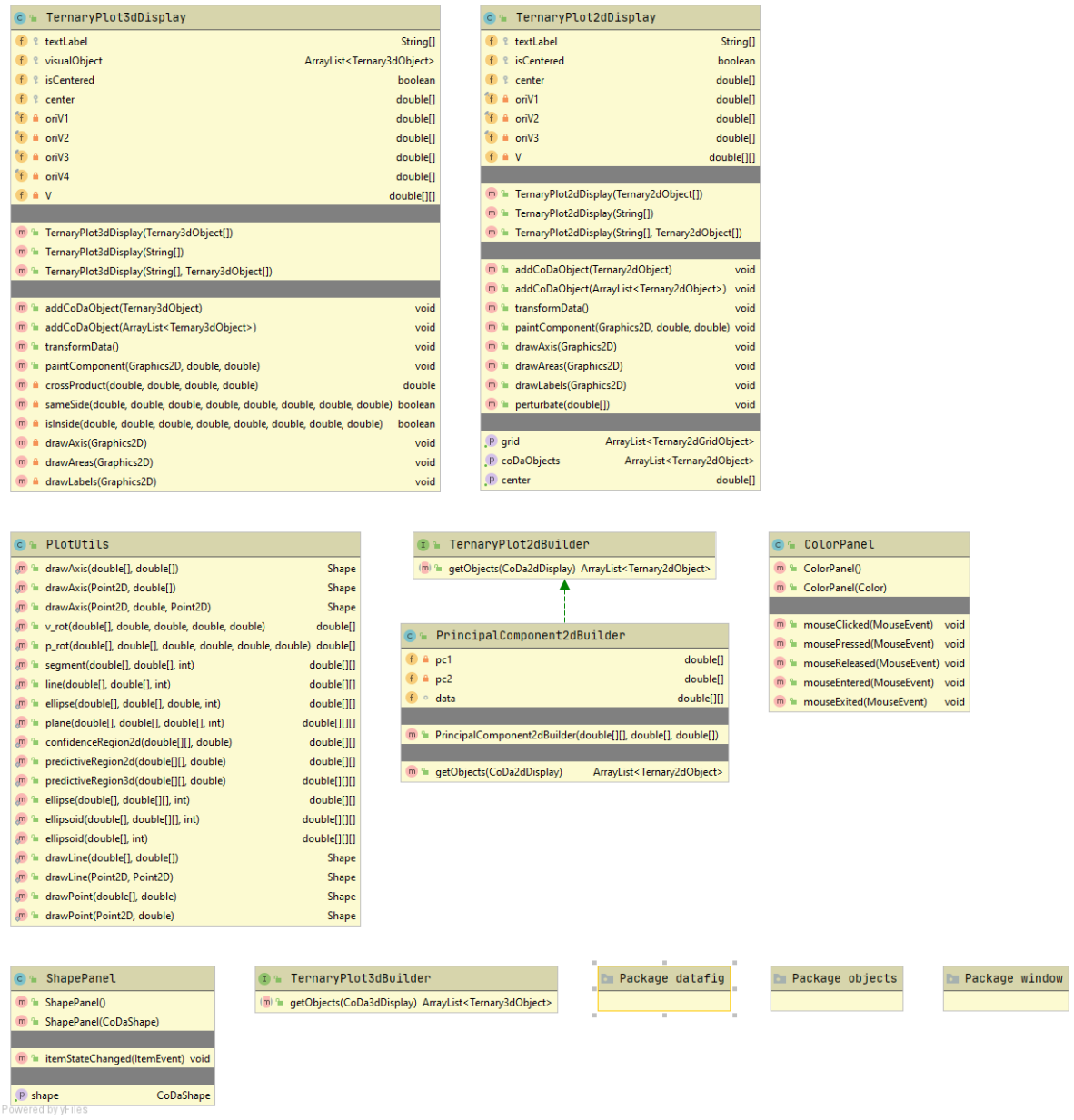


Figura 66: Diagrama de classes del package coda.plot2.

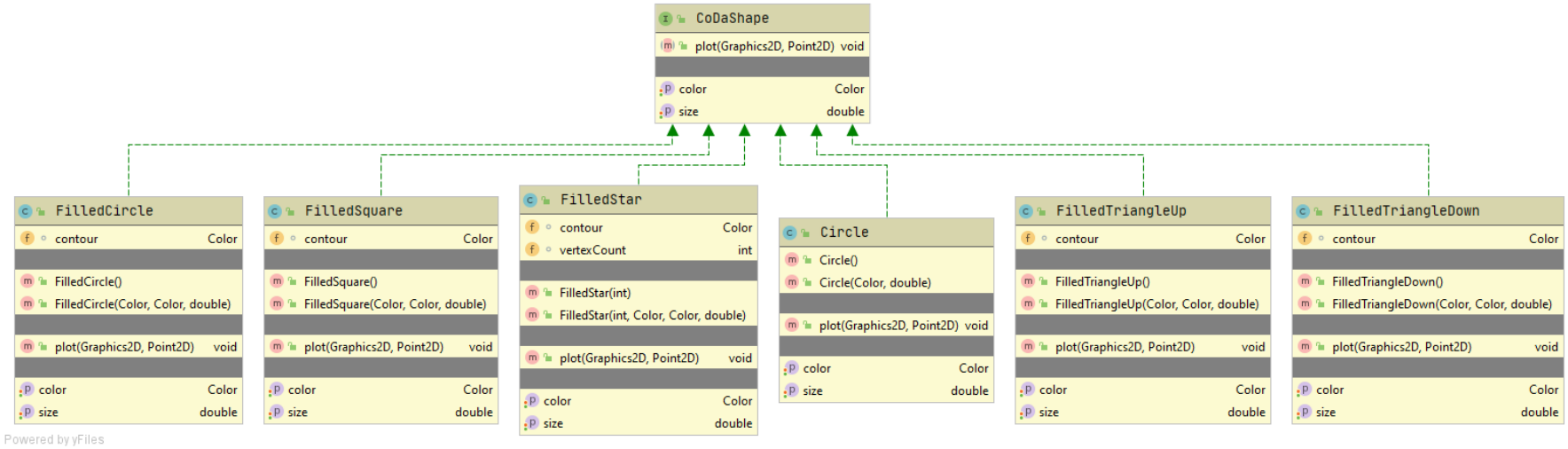


Figura 67: Diagrama de classes del package coda.plot2.datafig.

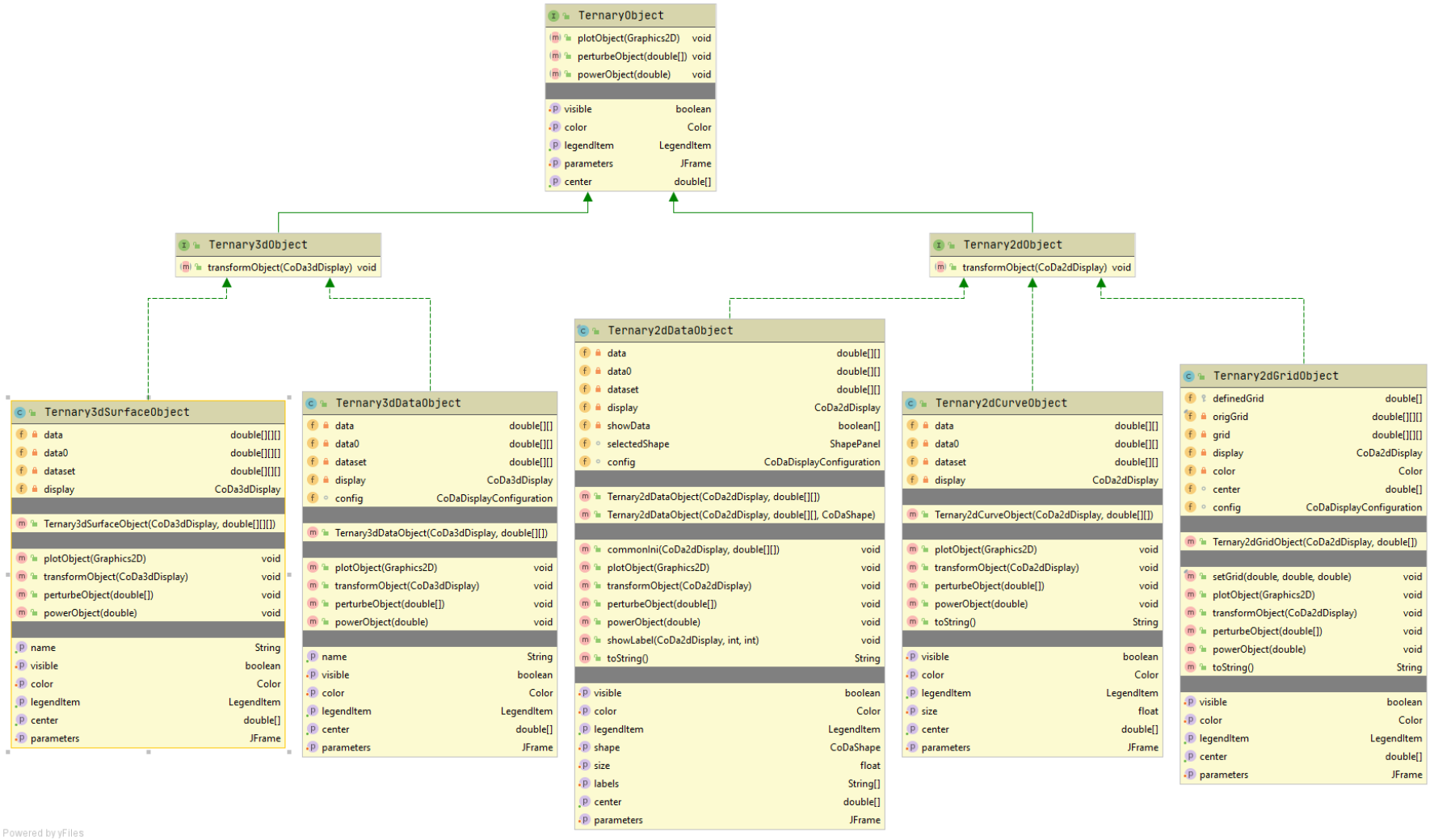


Figura 68: Diagrama de classes del package coda.plot2.objects.

Package coda.plot2.window

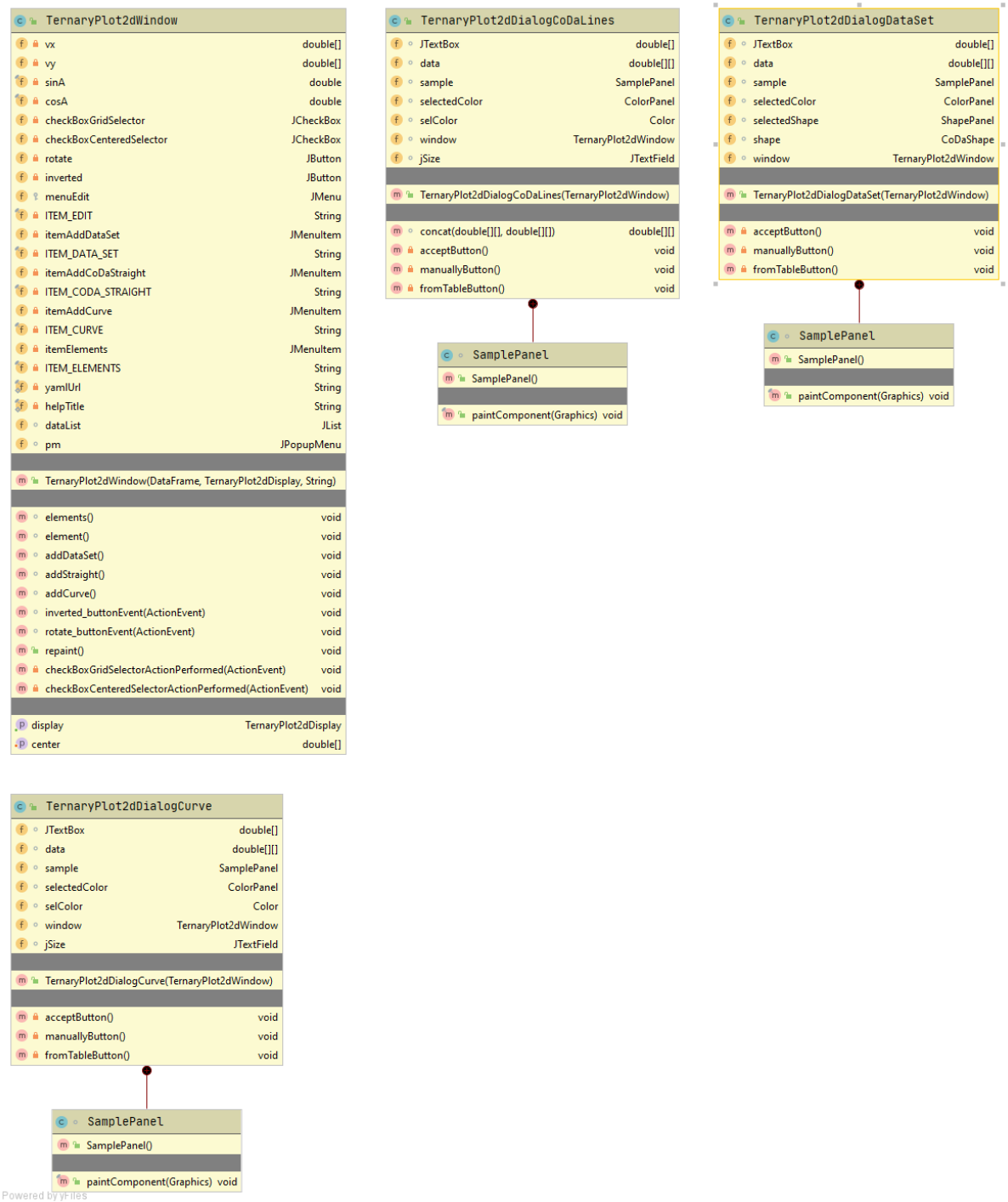


Figura 69: Diagrama de classes del package coda.plot2.window.

Package coda.util

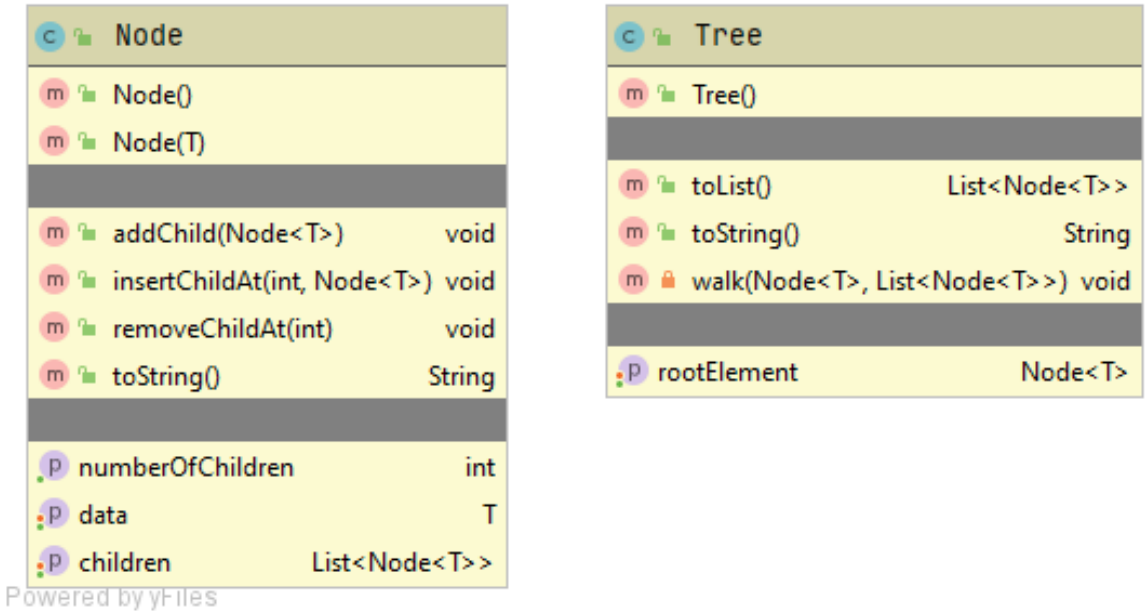


Figura 70: Diagrama de classes del package coda.util.

9 Implementació i proves

En aquest apartat es detallaran els problemes i les solucions apareguts al llarg del desenvolupament del projecte, així com la implementació d'alguns apartats en concret més rellevants.

Primer de tot comentarem la implementació on apareixeran els següents punts: la generació de data frames amb Java per poder ser enviats a R, la generació de paràmetres per ser enviats també a R i finalment com poder rebre els resultats obtinguts d'R des de Java i mostrar-los en el CoDaPack d'una manera clara i amb les interfícies adequades.

Finalment també es mostraran un seguit de proves que s'han fet per comprovar el funcionament de certes parts del projecte.

9.1 Implementació

9.1.1 Connexió entre Java i R

En el primer punt explicarem com s'ha fet per connectar Java amb R. Dividirem aquest apartat en fases, ja que durant la implementació d'aquest punt han aparegut diferents errors que comentarem:

- **Llibreries de Java i R:** Primer de tot cal tenir instal·lada les llibreries necessàries tant en Java com en R, aquestes són les comentades en els apartats **[7.4]** i **[7.5]** respectivament. En aquest apartat ens fixarem sobretot en la llibreria rJava d'R.
- **Inicialització d'Rengine:** El segon punt consisteix a inicialitzar el motor que s'utilitzarà per realitzar les crides a R des de Java, aquest motor és REngine. Podem veure aquesta inicialització en la Figura 71.

```
/**
 * VARIABLE R
 */
public static String[] Rargs = {"--vanilla"};
public static Rengine re = new Rengine(Rargs, false, null);
```

Figura 71: Inicialització d'REngine.

  Aquesta inicialització es realitza en el fitxer "CoDaPackMain.Java", i un cop creat l'objecte es va passant on és necessari per fer crides a R, per tal de no tenir més d'una instància del motor REngine.

- **La llibreria dinàmica jri.dll:** En compilar el codi, no apareixia cap error, però en executar el programa apareixia el següent error de la Figura 72.

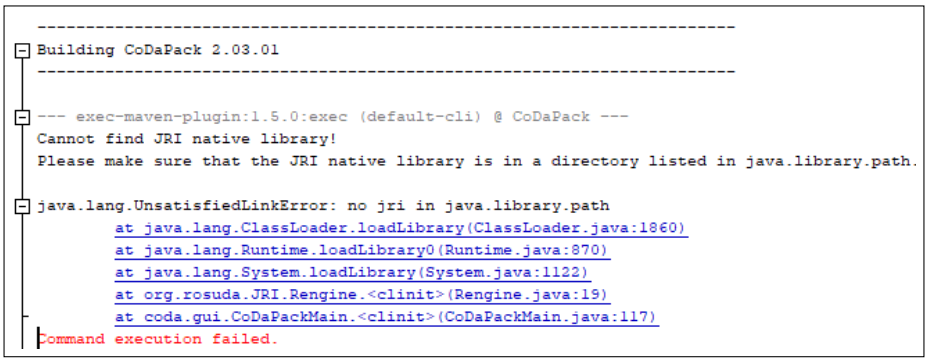


Figura 72: Error llibreria jri.dll.

Aquest error el que ens ve a dir és que no troba la llibreria JRI en el path de Java, per tant la solució que es va optar de fer és de copiar la llibreria dinàmica jri.dll de la Figura 73 a la carpeta bin del jdk de Java, ja que aquest directori està present en aquest path de Java comentat anteriorment.

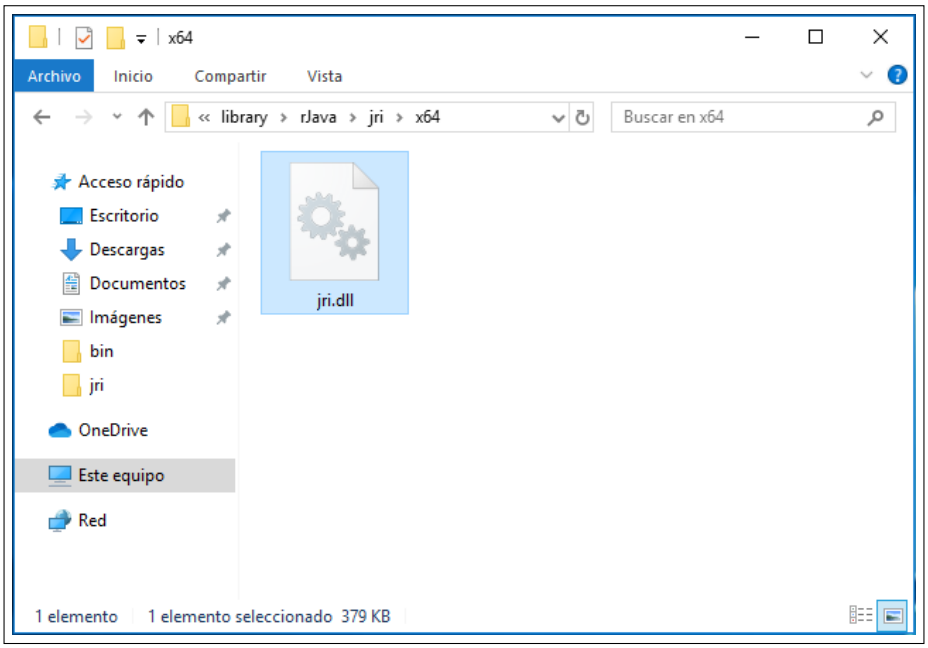


Figura 73: Llibreria jri.dll del package rJava d'R.

- **Llibreries dinàmiques d'R:** En fer el punt anterior, solucionem el problema que ens apareixia, però ara n'apareix un altre que es mostra en la Figura 74.

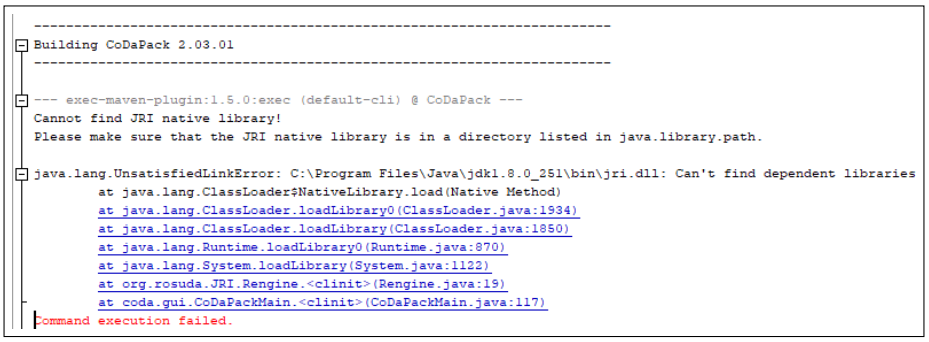


Figura 74: Error de dependències.

Per posar solució a aquest error es van agafar les llibreries dinàmiques d'R i es va fer el procediment que s'ha fet en el punt anterior, posar-les en la carpeta bin del jdk de Java. En la Figura 75 es pot apreciar les llibreries dinàmiques que s'han copiat d'R.

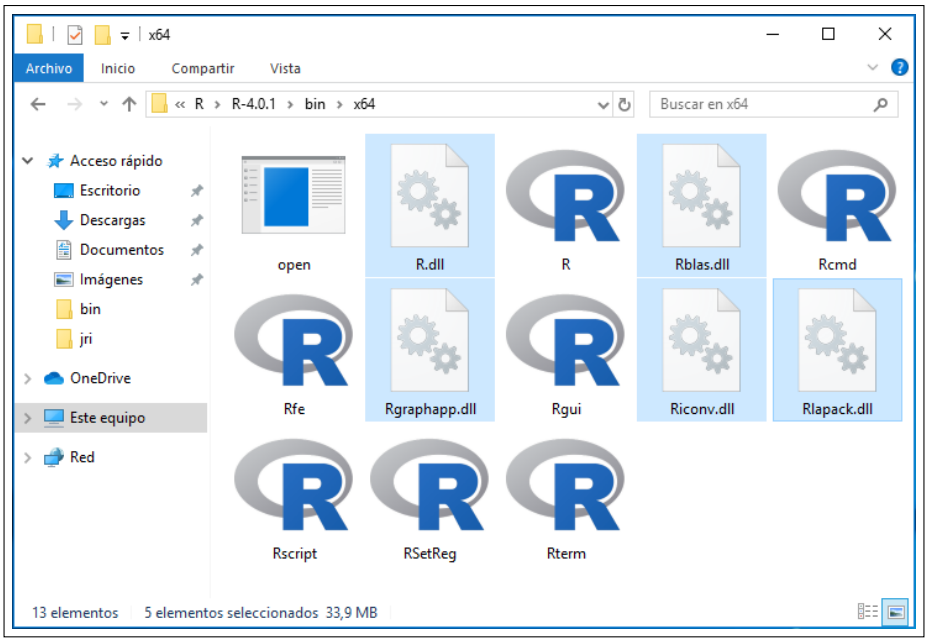


Figura 75: Llibreries dinàmiques d'R.

Un cop realitzat aquests passos ja podrem executar codi d'R des de Java i no apareix cap error d'execució del programa. Cal comentar que per realitzar la connexió de Java amb R des del sistema operatiu MacOS es realitzaria un procediment semblant, el que canvia és el format de les llibreries, ja que no són .dll.

9.1.2 Petita explicació de la funció eval()

La funció eval() és una funció de la llibreria JRI Engine té l'aspecte de la Figura 76, el que ens permet fer és analitzar i avaluar una expressió d'R i retornar el seu resultat **[34]**. Té el mateix efecte que utilitzar una altre funció d'eval que és: eval(s, true).

```
public REXP eval(java.lang.String s)
```

Figura 76: Aspecte de la funció eval().

El paràmetre que rep és "s" com una expressió d'R (com un string) per parsejar i avaluar.

El que retorna és el resultat de l'expressió o null si alguna cosa va malament en la expressió.

9.1.3 Generació de data frames amb Java per R

La generació de data frames de Java a R és una de les parts més importants del projecte, ja que és de les informacions principals a enviar per poder tractar les dades després amb R a part dels altres paràmetres de les rutines.

Els data frames que s'envien de Java a R representen el conjunt de dades que s'han triat en el menú de la rutina que hem comentat en apartats anteriors, aquest data frame representarà les dades que han de ser tractades i/o analitzades, a continuació una petita explicació del codi.

El següent codi de la Figura 77, correspon a la generació d'un data frame que anomenarem "X" i que correspondrà al conjunt de dades seleccionades en el corresponent menú.

Primer de tot el que fem és agafar les dades numèriques a partir de les variables seleccionades, ja que aquesta rutina contindrà només variables numèriques i no categòriques, però com podrem veure la funció també serveix tant per dades numèriques com categòriques.

Seguidament el que fem és recórrer cada una de les variables seleccionades i fent distinció si és una variable numèrica o categòrica, en cada cas el que farem és crear una llista amb les dades de la variable amb el format que correspon. Per això utilitzem la funció `re.eval()` que ens permet posar codi d'R per ser executat amb Java.

Un cop tenim ja cada una de les llistes creades amb el nom de la variable i que conté les dades d'aquesta simplement hem de crear el data frame passant com a paràmetres els noms de les variables que representaran les llistes creades amb R anteriorment.

Fet això a R ja tenim guardada una variable anomenada "X" i que representa el data frame amb les variables que ha seleccionat l'usuari en el menú de la rutina.

```
double[][] numericData = df.getNumericalData(selectedNames);

// Create X matrix

// create dataframe on r

        for(int i=0; i < selectedNames.length;i++){
            re.eval(selectedNames[i] + " <- NULL");
            if(df.get(selectedNames[i]).isNumeric()){
                for(double j : df.get(selectedNames[i]).getNumericalData()){
                    re.eval(selectedNames[i] + " <- c(" + selectedNames[i] +
                        "," + String.valueOf(j) + ")");
                }
            }
            else{ // categorical data
                for(String j : df.get(selectedNames[i]).getTextData()){
                    re.eval(selectedNames[i] + " <- c(" + selectedNames[i] +
                        ",'" + j + "')");
                }
            }
        }

        String dataFrameString = "X <- data.frame(";
        for(int i=0; i < selectedNames.length;i++){
            dataFrameString += selectedNames[i];
            if(i != selectedNames.length-1) dataFrameString += ",";
        }

        dataFrameString +=")";

        re.eval(dataFrameString); // we create the dataframe in R
```

Figura 77: Generació de data frames amb Java per R.

9.1.4 Generació de paràmetres amb Java per R

La generació de paràmetres per R és com hem comentat anteriorment una part important del projecte juntament amb la creació dels data frames, ja que cada una de les rutines que generem té associat moltes vegades un conjunt no només de dades sinó també de paràmetres.

Aquests paràmetres que s'envien de Java a R poden ser de diferents tipus com per exemple els següents: lògics, textuals o inclús una matriu que representaria una base. Cal comentar que per estàndard hem decidit que: els paràmetres textuals s'anomenaran P1, P2..., els paràmetres lògics s'anomenaran B1, B2... i finalment els paràmetres de base s'anomenaran o BaseX o BaseY.

En el menú ZPatterns tenim tres opcions com podem veure en la Figura 78, una que correspon al label i que és una entrada textual i les dues altres corresponen a opcions per mostrar percentatges i mostrar mitjanes, aquestes dues opcions corresponen a paràmetres lògics, ja que es poden seleccionar (TRUE) o no seleccionar (FALSE).

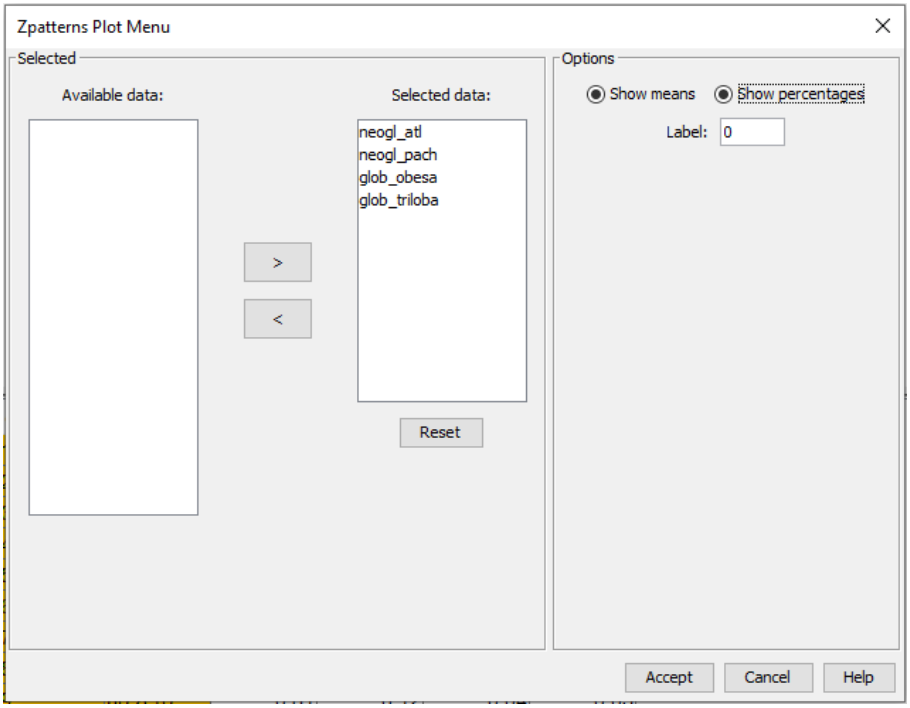


Figura 78: Menú Zpatterns.

A continuació mostrarem el significat de cada un d'aquests paràmetres segons la documentació de la funció Zpatterns d'R:

- **X:** Conjunt de dades (matriu o data frame).
- **label:** Etiqueta única (numèrica o de caràcters) utilitzada per identificar valors zero o no observats en X.
- **show.means:** Valor lògic que indica si els valors mitjans per patró es mostren en la taula de resum gràfic (per defecte show.means=FALSE).
- **bar.labels:** Valor lògic que indica si les etiquetes que mostren els percentatges s'han d'afegir als diagrames de barres del marge (per defecte bar.labels=FALSE).

El següent codi de la Figura 79, correspon a la generació de diferents paràmetres que correspondrà a les opcions del menú de la rutina ZPatterns.

Per tant simplement hem de mirar els paràmetres, en el cas de l'entrada de text si l'usuari ha introduït alguna cosa, mirem si correspon a "na", si és el cas, llavors creem la variable amb valor "NaN", si no fos el cas, llavors simplement creem la variable amb el que ha introduït l'usuari.
En el cas que l'usuari no hagi introduït res llavors agafem el valor per defecte que és "0".

Pels paràmetres lògics simplement hem de mirar si l'usuari ha seleccionat l'opció o no, en el cas que l'hagi seleccionat llavors creem la variable amb valor "TRUE", en cas contrari simplement creem la variable però amb valor "FALSE".

Fet això ja tindrem els tres paràmetres creats en R per poder realitzar la rutina amb ells.

```
void constructParametersToR(){
    /* construim parametre label */

    if(P1.getText().length() > 0){
        if(P1.getText().equals("na")) re.eval("P1 <- \"NaN\"");
        else re.eval("P1 <- " + P1.getText());
    }
    else re.eval("P1 <- 0"); // by default 0

    /* construim parametres logics */

    if(this.B1.isSelected()) re.eval("B1 <- TRUE");
    else re.eval("B1 <- FALSE");
    if(this.B2.isSelected()) re.eval("B2 <- TRUE");
    else re.eval("B2 <- FALSE");

}
```

Figura 79: Generació de paràmetres amb Java per R.

Finalment mostrem en la figura Figura 80 com en l'script d'R es realitzaria la crida a la funció Zpatterns amb els paràmetres ja definits en R.

```
zCompositions::zPatterns(X, label=P1, show.means=B1, bar.labels=B2)
```

Figura 80: Crida de la funció Zpatterns.

Podem observar que en les opcions de la crida, es posen directament les variables que s'han creat amb els valors definits en el menú Zpatterns del CoDaPack.

9.1.5 Rebre dades de script d'R des de Java i mostrar-les

Poder rebre dades generades amb R és l'altra tasca principal del projecte, ja que està bé poder enviar dades de Java a R per poder realitzar operacions, però també el que volem obtenir són els resultats que s'han obtingut al realitzar la rutina amb R.

Per generar una sortida amb els scripts d'R s'ha decidit crear una estructura de dades que hem anomenat "*cdp_res*" i que podem observar en la Figura 81.

Com podem observar en la imatge l'estructura és una llista amb els diferents elements que es retornen: text, dataframe, graphs i new_data. Cada un dels elements de la llista és ell mateix també una llista, ja que pot haver-hi per exemple més d'un gràfic a retornar.

```
# Ooutput
cdp_res = list(
  'text' = list(paste("ZERO PATTERNS"),sortida),
  'dataframe' = list(),
  'graph' = graphnames,
  'new_data' = list()
)
```

Figura 81: Estructura de dades dels scripts d'R.

A continuació anem a veure donada l'estructura "*cdp_res*" amb els resultats de l'script, com fem amb Java per recollir les dades.

- **Text**

  En la Figura 82 podem veure la funció que utilitzem en Java per obtenir el text d'un script d'R i posteriorment mostrar-lo en el CoDaPack.

  El que fa la funció és primer de tot mostrar el títol de la rutina per poder separar correctament en l'output la sortida de cada una de les rutines. Seguidament el que es fa és mirar quantes sortides de text s'han generat en l'script d'R.

  Finalment el que es fa és recórrer cada una de les sortides de text i guardar-ho com un array de strings per després mostrar aquest array de string prèviament convertit en el tipus de sortida "*OutputForR*".

  Un cop fet això en principi en el panell de sortides de text del CoDaPack visualitzaríem totes les sortides de text que s'han generat en l'script d'R.

```
void showText(){

    REXP result;
    String[] sortida;

    /* header output */

    outputPanel.addOutput(new OutputText("Zpatterns Plot:"));

    /* R output */

    int midaText = re.eval("length(cdp_res$text)").asInt();
    for(int i=0; i < midaText; i++){
        result = re.eval("cdp_res$text[[" + String.valueOf(i+1) + "]]");
        sortida = result.asStringArray();
        outputPanel.addOutput(new OutputForR(sortida));
    }
}
```

Figura 82: Rebre i mostrar text d'R.

- **Data frames**

  En la Figura 83 podem veure el codi per poder crear nous data frames a partir de les noves dades rebudes des d'un script d'R.

  El primer de tot és mirar la longitud per saber quants data frames nous hem d'afegir en el sistema. Un cop sabem quants data frames hem de crear el que fem és recórrer aquestes data frames que s'han generat en R i per cada un d'ells mirem quantes variables conté i creem un data frame buit que representarà el data frame que estem creant actualment.

  Ara el que fem és recórrer les variables del data frame que volem crear, i per cada una d'aquestes variables agafem el seu nom i mirem de quin tipus és, si correspon a una variable categòrica o correspon a una variable numèrica.
  En el cas que sigui categòrica guardarem les seves dades en una llista de string, en cas contrari guardarem les dades en una llista de double. Quan tenim el nom de la variable i les seves dades corresponents ja només ens queda crear la variable amb el seu nom i dades i afegir-la al nou data frame que estem generant.

  Quan ja hem repetit el procés per totes les variables del nou data frame, tindrem el data frame nou amb les variables que li corresponen, ara només falta posar el nom a aquest data frame. Per fer això agafem el nom que se li ha assignat en l'script d'R i mirem si està disponible en el CoDaPack (que no sigui repetit), si fos el cas, li apliquem "$+c$" al final del nom fins que estigui disponible.

  Per últim només queda assignar el nom al nou data frame i afegir-lo al sistema.

  Aquest nou data frame serà el que guardarem com una nova taula de dades del CoDaPack.

```
void createDataFrame(){

    int nDataFrames = re.eval("length(cdp_res$dataframe)").asInt();

    for(int i=0; i < nDataFrames; i++){

        int nVariables = re.eval("length(cdp_res$dataframe[[" +
            String.valueOf(i+1) + "]])").asInt();
        DataFrame newDataFrame = new DataFrame();

        for(int j=0; j < nVariables; j++){

            String varName = re.eval("names(cdp_res$dataframe[[" +
                String.valueOf(i+1) + "]][" + String.valueOf(j+1) +
                "])").asString();
            String isNumeric = re.eval("class(unlist(cdp_res$dataframe[[" +
                String.valueOf(i+1) + "]][" + String.valueOf(j+1) +
                "]))").asString();

            if(isNumeric.equals("numeric")){ /* crear una variable numerica */
                double[] data =
                    re.eval("as.numeric(unlist(cdp_res$dataframe[[" +
                    String.valueOf(i+1) + "]][" + String.valueOf(j+1) +
                    "]))").asDoubleArray();
                newDataFrame.addData(varName, data);
            }

            else{ /* crear una variable categorica */
                String[] data =
                    re.eval("as.character(unlist(cdp_res$dataframe[[" +
                    String.valueOf(i+1) + "]][" + String.valueOf(j+1) +
                    "]))").asStringArray();
                newDataFrame.addData(varName, new Variable(varName,data));
            }
        }

        String dataFrameName = re.eval("names(cdp_res$dataframe)[" +
            String.valueOf(i+1) + "]").asString();

        while(!mainApplication.isDataFrameNameAvailable(dataFrameName)){
            dataFrameName += "c";
        }

        newDataFrame.setName(dataFrameName);
        mainApplication.addDataFrame(newDataFrame);
    }
}
```

Figura 83: Rebre i mostrar un nou data frame d'R.

- **Graphs**

  Els gràfics que es generen amb R, el que fem és en la seva llista corresponent guardar les rutes a aquests gràfics que s'han generat en el PC.
  Per tant després simplement el que hem de fer és agafar aquesta ruta en Java i agafar la imatge.

  En la Figura 84 podrem observar com rebem i mostrem els gràfics que es generen des d'R en Java.

  El que fem com fem normalment és saber quants gràfics hem de mostrar, per tant mirem la longitud de la llista que conté les rutes dels gràfics. Un cop fet això el que fem és recórrer la llista que conté les rutes i anem guardant cada una de les rutes en una estructura que tenim feta a Java.

  Per últim simplement el que fem és cridar al mètode per mostrar els gràfics passant com a paràmetre quin és l'índex del gràfic actual. Aquest mètode el que farà és generar una finestra amb el gràfic que contingui l'índex indicat per paràmetre.

```
void showGraphics() throws IOException{

    int numberOfGraphics = re.eval("length(cdp_res$graph)").asInt(); /* num de
        grafics */

    for(int i=0; i < numberOfGraphics; i++){
        tempDirR = re.eval("cdp_res$graph[[" + String.valueOf(i+1) +
            "]]").asString();
        tempsDirR.add(tempDirR);
        plotZpatternsMenu(this.framesZpatternsMenu.size());
    }
}
```

Figura 84: Rebre i mostrar gràfics generats amb R.

Per tant en definitiva el que estem fent amb el gràfic, es generar-ho amb R en el nostre ordinador, seguidament agafem el path d'aquest gràfic i el mostrem en un frame del CoDaPack amb les opcions que ens interessen, en aquest cas amb les opcions per poder exportar aquest gràfic que s'ha generat.

- **New data**

  En la Figura 85 podem veure el codi per poder crear noves variables en el data frame actual a partir de les noves dades rebudes des d'un script d'R.

  El primer de tot és mirar la longitud per saber quantes variables noves hem d'afegir en el data frame.
  Un cop sabem quantes variables hem de crear el que fem és recórrer aquestes variables que s'han generat en R i per cada una d'elles agafem el seu nom. Després d'haver agafat el nom mirem si es tracta d'una nova variable numèrica o categòrica, depenent de si és numèrica o categòrica rebem les dades o com un array de doubles (numèrica) o com un array de strings (categòrica).

  Ara que ja tenim tant el nom de la variable com les seves dades simplement hem d'afegir la nova variable en el data frame actual i actualitzar la vista d'aquest en la pàgina principal del CoDaPack.
  En el cas de que el nom de la variable estigui duplicat s'afegeix automàticament un sufix en el nou nom de la variable.
  Aquest new data representaran les columnes noves que s'afegeixen a la taula de dades actual del CoDaPack.

```
void createVariables(){

    int numberOfNewVar =
        re.eval("length(colnames(cdp_res$new_data))").asInt(); /* numero de
        columnes nomes*/

    for(int i=0; i < numberOfNewVar; i++){
        String varName = re.eval("colnames(cdp_res$new_data)[" +
            String.valueOf(i+1) + "]").asString();
        String isNumeric =
            re.eval("as.character(is.numeric(cdp_res$new_data[["+
            String.valueOf(i+1) +"]]))").asString();
        if(isNumeric.equals("TRUE")){
            double[] data = re.eval("as.numeric(cdp_res$new_data[," +
                String.valueOf(i+1) + "])").asDoubleArray();
            df.addData(varName,data);
        }
        else{ // categoric
            String[] data = re.eval("as.character(cdp_res$new_data[," +
                String.valueOf(i+1) + "])").asStringArray();
            df.addData(varName, new Variable(varName,data));
        }
        mainApplication.updateDataFrame(df);
    }
}
```

Figura 85: Rebre i afegir noves variables d'R.

9.1.6 Crida al script d'R + tractament d'errors

A continuació mostrarem a on es fa la crida a un script d'R + com es fa per tractar els possibles errors que pugui contenir aquest. Seguidament mostrem el codi en la Figura 86.

```
// executem script d'R

String url = CoDaPackConf.rScriptPath + "ZPatterns.R";


re.eval("tryCatch({error <- \"NULL\";source(\"" + url + "\")}, error =
    function(e){ error <<- e$message})");

String[] errorMessage = re.eval("error").asStringArray();

if(errorMessage[0].equals("NULL")){
    /* executem totes les accions possibles */
    showText();
    createVariables();
    createDataFrame();
    try {
        showGraphics();
    } catch (IOException ex) {
        Logger.getLogger(ZpatternsMenu.class.getName()).log(Level.SEVERE,
            null, ex);
    }
}
else{
    OutputElement type = new OutputText("Error in R:");
    outputPanel.addOutput(type);
    OutputElement outElement = new OutputForR(errorMessage);
    outputPanel.addOutput(outElement);
}
```

Figura 86: Crida al script d'R + tractament d'errors.

Com podem veure en el codi de la part superior, primer de tot guardem el path on es troba l'script que volem executar. Seguidament fem un try per intentar executar aquest script, en el cas que hi hagi algun error en aquest try el que farem és guardar l'error corresponent en una variable que s'anomena "error". Si tot ha anat correctament la variable error valdrà NULL i per tant podrem mostrar els resultats obtinguts, en cas contrari llavors recollim els errors i simplement el que fem és mostrar-los per la pantalla de sortida de missatges.

9.1.7 Creació dels Instal·ladors

Un cop acabada l'aplicació, cal una posada en marxa del producte, és per això que cal fer uns instal·ladors de l'aplicació per la bona distribució. Durant el projecte s'han desenvolupat un total de dos instal·ladors, un per sistemes operatiu Windows i un altre per sistemes operatius MacOS.

A continuació mostrarem la implementació de cada un d'ells.

Windows

Primer de tot mostrarem la implementació de l'instal·lador per Windows. En primer lloc el que cal fer és un cop l'aplicació acabada és compilar-la amb totes les dependències que són necessàries per finalment generar un arxiu jar amb dependències.

Seguidament en la Figura 87 és mostra un directori amb tot el que fa falta per realitzar l'instal·lador.

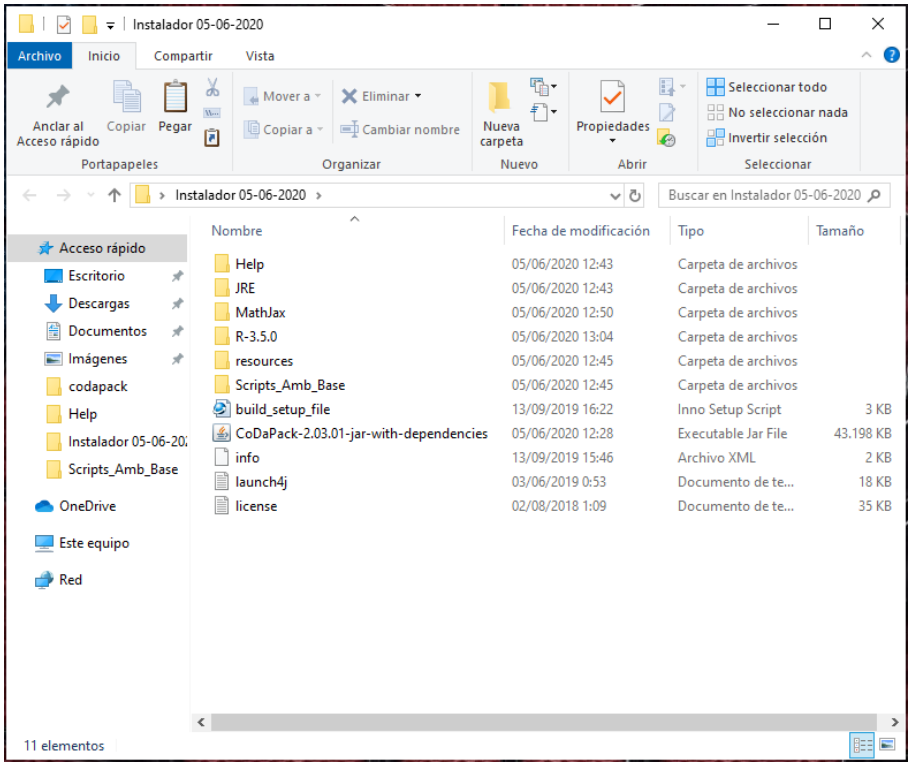


Figura 87: Fitxer inicials instal·lador Windows.

En la figura podem observar els següents elements:

- **Help:** Directori que conté cada un dels arxius YAML amb la informació dels menús de Help.
- **JRE:** Directori que conté la màquina virtual de Java.
- **MathJax:** Directori que conté les eines de MathJax necessàries per poder generar formules matemàtiques amb format web.
- **R-3.5.0:** Directori que conté l'R que s'utilitzarà amb el CoDaPack.
- **resources:** Directori que conté els recursos de l'aplicació com per exemple les icones.
- **Scripts_Amb_Base:** Directori que conté els scripts d'R que s'utilitzen en algunes rutines del CoDaPack.
- **buil_setup_file:** Fitxer de configuració per crear l'instal·lador.
- **CoDaPack-2.03.01-jar-with-dependencies:** Jar de l'aplicació que conté també les dependències necessàries.

- **info i launch4j:** Fitxers de configuració per crear l'executable.
- **license:** Llicencia del programari.

A continuació cal realitzar en primer pas l'executable amb el jar que acabem de realitzar. En la Figura 88 es mostra la configuració bàsica de l'aplicació lauch4j per realitzar l'executable.

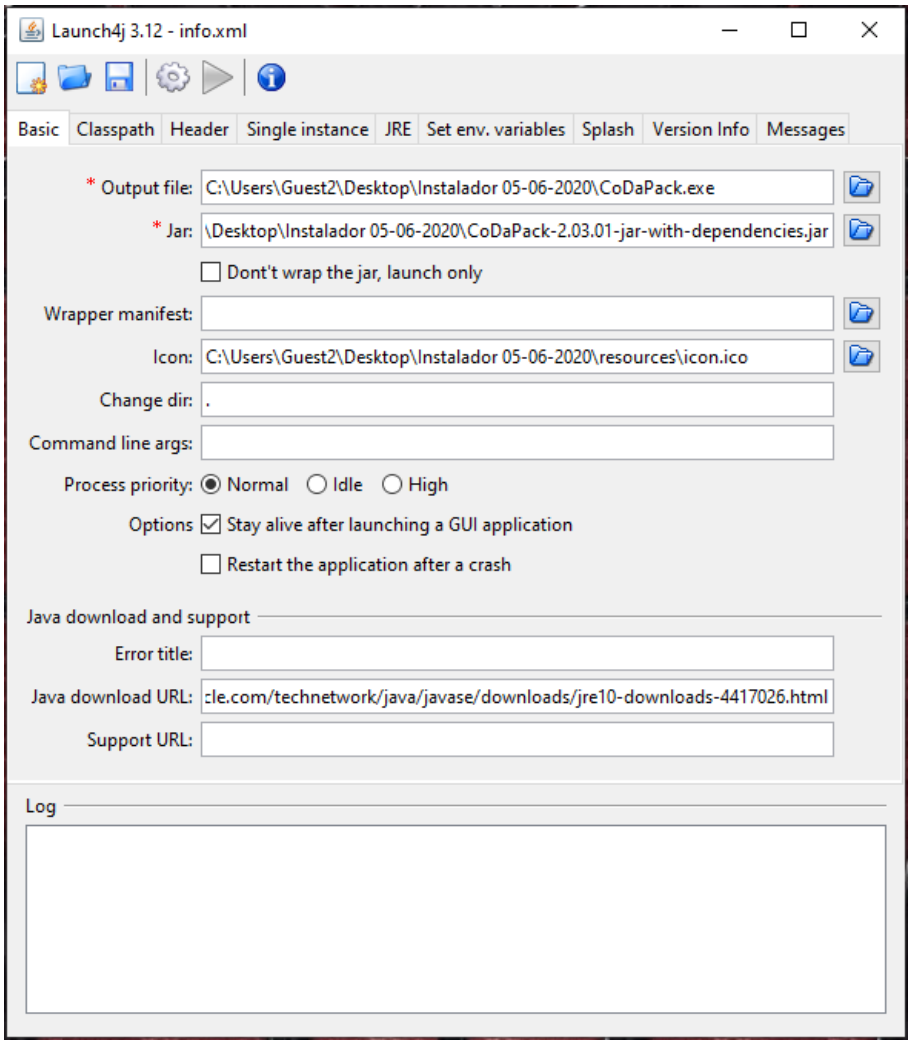


Figura 88: Configuració bàsica lauch4j.

Podem veure que en aquesta figura es defineix el nom i la localització de l'output de l'executable, la localització del jar que s'utilitzarà per realitzar l'executable, la icona de l'executable i finalment també es defineix un URL per descarregar una màquina virtual de Java en el cas que no es detecti cap versió de Java, que en principi no hauria de passar.

Continuem amb altres configuracions del launch4j, a continuació en la Figura 89, la configuració en aquest cas del JRE.

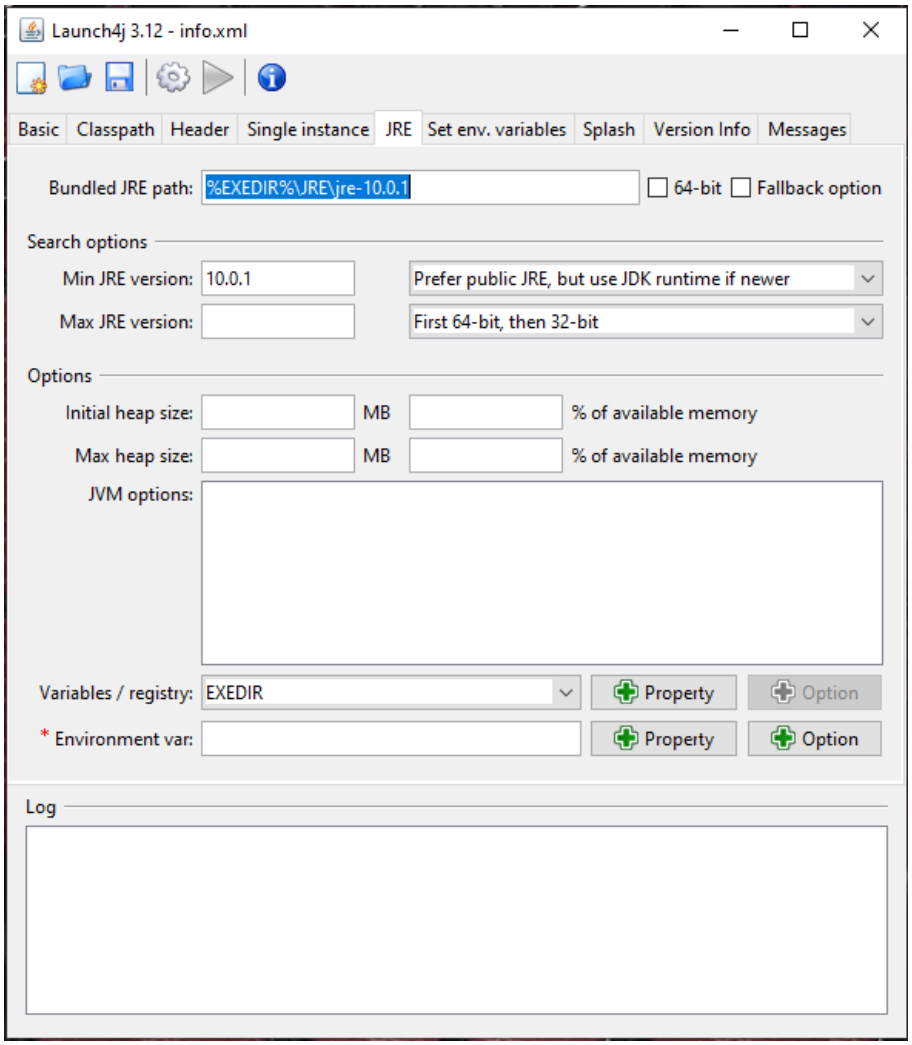


Figura 89: Configuració JRE lauch4j.

En aquest cas la configuració consisteix com podem veure en la imatge en configurar el path que agafarà l'aplicació com a JRE a utilitzar, en aquest cas el JRE de la versió 10.0.1 i juntament també definim que utilitzi com a mínim aquesta versió de JRE.

Finalment per acabar de configurar launch4j definim unes variables d'entorn a crear en iniciar l'executable, aquestes variables definides es poden veure en la Figura 90.

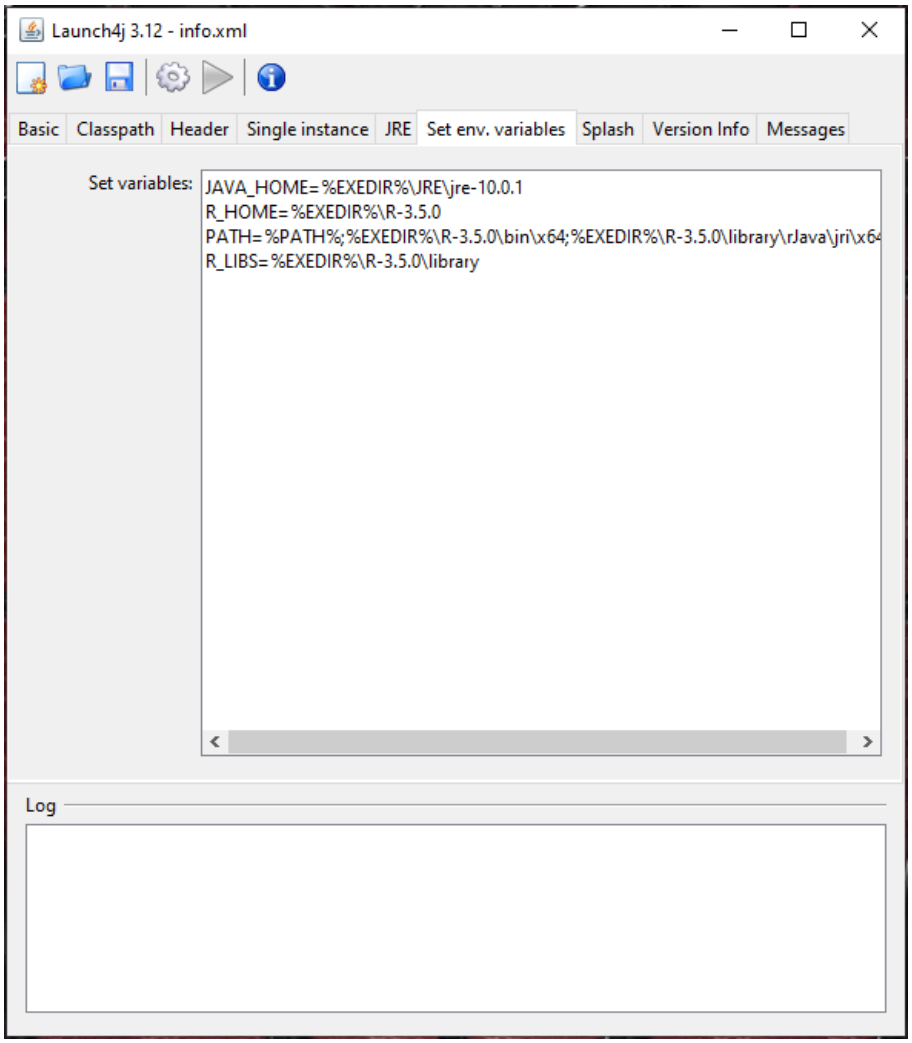


Figura 90: Configuració variable d'entorn lauch4j.

Podem veure que s'han definit les següents variables d'entorn: JAVA_HOME, R_HOME, PATH, R_LIBS.
Un cop configurat tot això ja podem executar l'aplicació perquè es creï l'executable amb la configuració definida, en la Figura 91 podem veure el directori on apareix ja l'executable que s'acaba de crear.

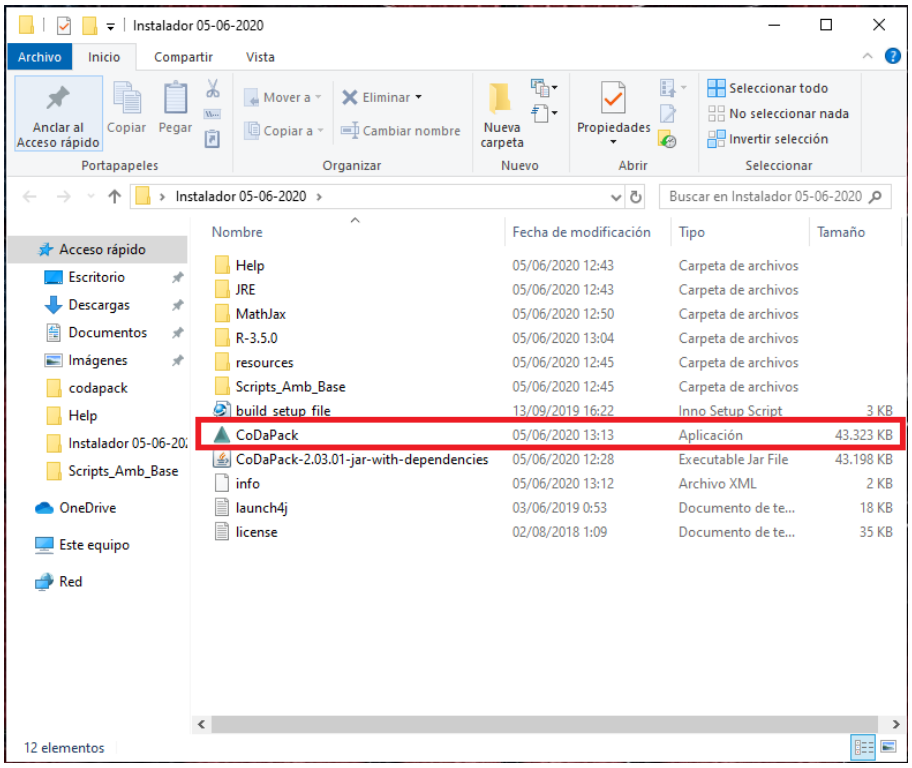


Figura 91: Executable CoDaPack.

Molt bé ara que ja tenim l'executable anem a fer l'instal·lador, per realitzar aquesta tasca utilitzarem l'aplicació Inno Setup que ja hem comentat anteriorment. Seguidament en la Figura 92 la configuració utilitzada.

```
; Command line call:
; C:\Program Files (x86)\Inno Setup 5\ISCC.exe build_setup_file.iss
; Script generated by the Inno Setup Script Wizard.
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!

#define MyAppName "CoDaPack"
#define MyAppVersion "2.03.01"
#define MyAppPublisher "Universitat de Girona"
#define MyAppURL "http://ima.udg.edu/codapack/"
#define MyAppExeName "CoDaPack.exe"

[Setup]
; NOTE: The value of AppId uniquely identifies this application.
; Do not use the same AppId value in installers for other applications.
; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)
AppId={{C1C9D163-40EA-40F2-81C2-877DA22B1570}
AppName={#MyAppName}
AppVersion={#MyAppVersion}
;AppVerName={#MyAppName} {#MyAppVersion}
AppPublisher={#MyAppPublisher}
AppPublisherURL={#MyAppURL}
AppSupportURL={#MyAppURL}
AppUpdatesURL={#MyAppURL}
DefaultDirName={pf}\{#MyAppName}
DefaultGroupName={#MyAppName}
LicenseFile=license.txt
OutputDir=SetUp\
OutputBaseFilename=CoDaPack-setup-{#MyAppVersion}
SetupIconFile=resources\icon.ico
Compression=lzma
SolidCompression=yes
ChangesAssociations=yes
ArchitecturesInstallIn64BitMode=x64
ArchitecturesAllowed=x64

[Languages]
Name: "english"; MessagesFile: "compiler:Default.isl"

[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[Files]
Source: "CoDaPack.exe"; DestDir: "{app}"; Flags: ignoreversion
;Source: "target\codapack-{#MyAppVersion}.jar"; DestDir: "{app}"; Flags: ignoreversion
Source: "license.txt"; DestDir: "{app}"; Flags: ignoreversion
; NOTE: Don't use "Flags: ignoreversion" on any shared system files
Source: "JRE\*"; DestDir: "{app}\JRE"; Flags: ignoreversion recursesubdirs
Source: "R-3.5.0\*"; DestDir: "{app}\R-3.5.0"; Flags: ignoreversion recursesubdirs
Source: "Scripts_Amb_Base\*"; DestDir: "{app}\Scripts_Amb_Base"; Flags: ignoreversion recursesubdirs
Source: "Help\*"; DestDir: "{app}\Help"; Flags: ignoreversion recursesubdirs
Source: "MathJax\*"; DestDir: "{app}\MathJax"; Flags: ignoreversion recursesubdirs

[Icons]
Name: "{group}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"
Name: "{commondesktop}\{#MyAppName}"; Filename: "{app}\{#MyAppExeName}"; Tasks: desktopicon

[Run]
Filename: "{app}\{#MyAppExeName}"; Description: "{cm:LaunchProgram,{#StringChange(MyAppName, '&', '&&')}}"; Flags: nowait postinstall skipifsilent

[Registry]
Root: HKCR; Subkey: ".cdp"; ValueType: string; ValueName: ""; ValueData: "cdpfile"; Flags: uninsdeletevalue
Root: HKCR; Subkey: "cdpfile"; ValueType: string; ValueName: ""; ValueData: "CoDaPack File"; Flags: uninsdeletekey
Root: HKCR; Subkey: "cdpfile\DefaultIcon"; ValueType: string; ValueName: ""; ValueData: "{app}\CoDaPack.exe,0"
Root: HKCR; Subkey: "cdpfile\shell\open\command"; ValueType: string; ValueName: ""; ValueData: """{app}\CoDaPack.exe"" ""%1"""
```

Figura 92: Configuració Inno Setup.

En aquesta configuració es defineixen diferents paràmetres com l'idioma, els arxius que instal·larà l'instal·lador, les icones...

Finalment en la Figura 93 podem veure que un cop compilada la configuració de l'arxiu s'acaba generant l'instal·lador.

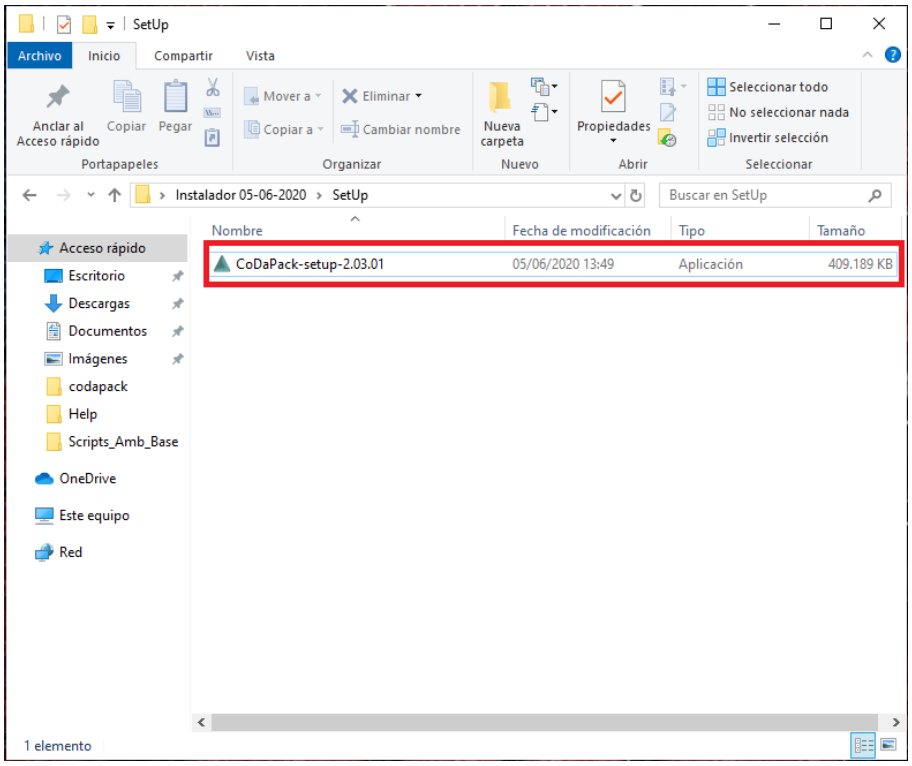


Figura 93: Instal·lador CoDaPack per Windows.

MacOS

Ara mostrarem en aquest apartat la implementació de l'instal·lador però per al sistema operatiu MacOS.

Primer de tot com en l'apartat anterior el que fem és compilar l'aplicació amb les dependències per generar el jar corresponent amb les dependències incloses.

A continuació en la Figura 94 i la Figura 95 els diferents elements que conté l'instal·lador.

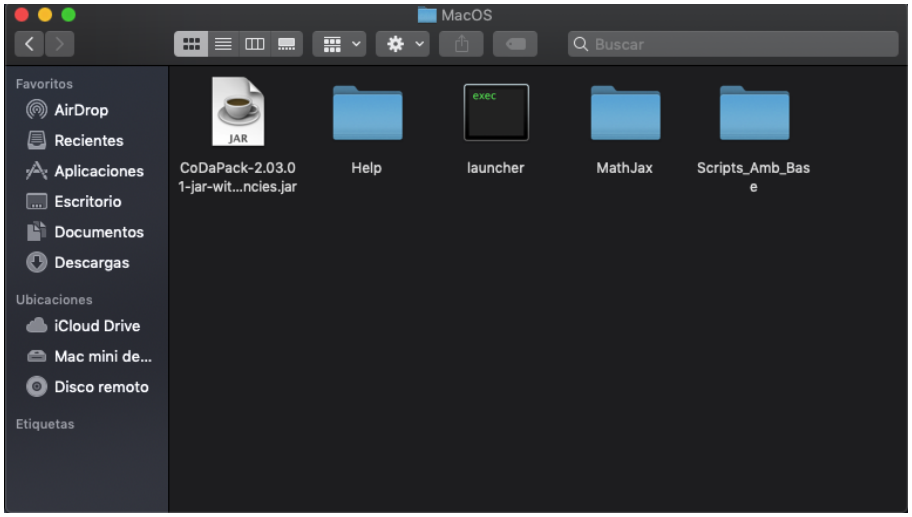


Figura 94: Fitxers Instal·lador MacOS nº1.

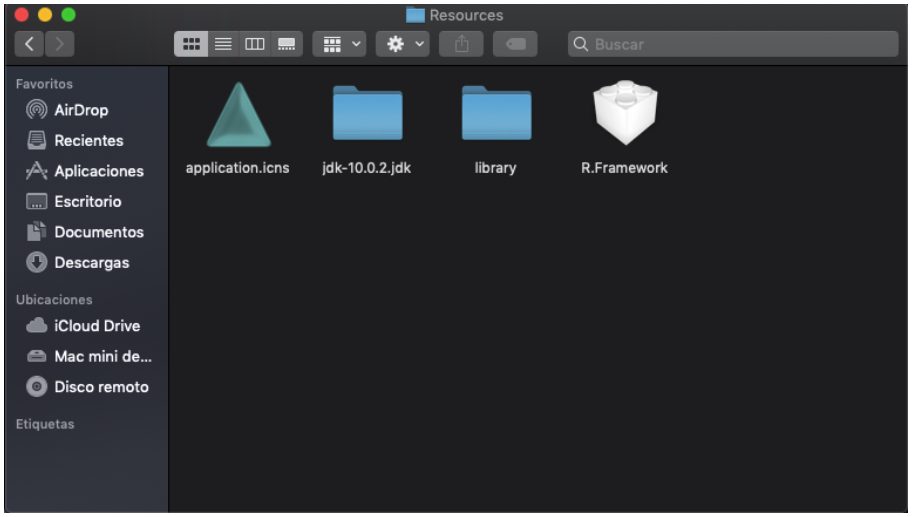


Figura 95: Fitxers Instal·lador MacOS nº2.

En les figures podem observar els següents elements:

- **CoDaPack-2.03.01-jar-with-dependencies:** Jar de l'aplicació que conté també les dependències necessàries.
- **Help:** Directori que conté cada un dels arxius YAML amb la informació dels menús de Help.
- **launcher:** Executable que inicialitza l'aplicació.
- **MathJax:** Directori que conté les eines de MathJax necessàries per poder generar formules matemàtiques amb format web.
- **Scripts_Amb_Base:** Directori que conté els scripts d'R que s'utilitzen en algunes rutines del CoDaPack.
- **application.icns:** Icona de l'aplicació.
- **jdk-10.0.2.jdk:** Directori que conté la màquina virtual de Java.
- **library:** Llibreries necessàries d'R.
- **R.Framework:** Directori que conté l'R que s'utilitzarà amb el CoDaPack.

Seguidament el que hem de fer és configurar el fitxer launcher, ja que aquest fitxer és el que farà que s'executi l'aplicació, seguidament la configuració del fitxer en la Figura 96.

```
# Author: Sri Harsha Chilakapati

# Constants
JAVA_MAJOR=10.0.1
JAVA_MINOR=8
APP_JAR="CoDaPack-2.03.01-jar-with-dependencies.jar"
APP_NAME="CoDaPack"
VM_ARGS=""

# Set the working directory
DIR=$(cd "$(dirname "$0")"; pwd)

# SETTING R CONFIG FOR CoDaPack

DIRECTORY_R_GOOD_VERSION="/Library/Frameworks/R.Framework/Versions/3.5"
DIRECTORY_R_OTHER_VERSION="/Library/Frameworks/R.Framework"
DIRECTORI_TO_PUT_R="/Library/Frameworks"

if [ -d $DIRECTORY_R_GOOD_VERSION ]; then
    echo "Nothing to do"
elif [ -d $DIRECTORY_R_OTHER_VERSION ]; then
    osascript -e "do shell script \"mkdir -p
        $DIRECTORI_TO_PUT_R/R.Framework/Versions; ln -s
        /Applications/CoDaPack.app/Contents/Resources/R.Framework/Versions/3.5
        $DIRECTORI_TO_PUT_R/R.Framework/Versions/3.5\" with administrator
        privileges"
else
    osascript -e "do shell script \"mkdir -p
        $DIRECTORI_TO_PUT_R/R.Framework/Versions; ln -s
        /Applications/CoDaPack.app/Contents/Resources/R.Framework/Versions/3.5
        $DIRECTORI_TO_PUT_R/R.Framework/Versions/3.5\" with administrator
        privileges"
fi

# SETTING ENVIRONMENT VARIABLES

export R_HOME="$DIRECTORY_R_GOOD_VERSION/Resources"
export R_LIBS_USER="$DIR/../Resources/library"
export PATH="$PATH:$R_HOME:$R_HOME/bin:$R_LIBS_USER/rJava/jri/libjri.jnilib"
export JAVA_HOME="$DIR/../Resources/jdk-10.0.2.jdk/Contents/Home"
export SCRIPTS_DIRECTORY="$DIR/" #directory actual a veure si l'agafa
export HELP_DIRECTORY="$DIR/" #directory actual a veure si l'agafa
export MATHJAX_DIRECTORY="$DIR/MathJax/"
```

Figura 96: Configuració fitxer launcher.

En la imatge podem veure que es fan un seguit de configuracions, també el que es mira és la versió d'R que es té en el sistema, en el cas que ja tingui la versió bona no es fa res, altrament si es té una altra versió o no es té cap versió instal·lem la nostra versió d'R al sistema.

Finalment definim les següents variables d'entorn que ens són necessaries: R_HOME, R_LIBS_USER, PATH, JAVA_HOME, SCRIPTS_DIRECTORY, HELP_DIRECTORY i MATHJAX_DIRECTORY.

Un cop això fet es va procedir a realitzar l'instal·lador gràcies al procediment que es realitza a par-

tir del vídeo que s'adjunta en la bibliografia **[35]**.

A continuació per acabar mostrem en la Figura 97 com és l'aspecte final d'aquest instal·lador.

Figura 97: Instal·lador CoDaPack per MacOS.

Simplement s'ha de fer drag and drop de l'aplicació a la carpeta d'aplicacions.

9.2 Proves

S'ha decidit dividir aquest apartat de proves en dues parts, primer de tot les proves que s'han realitzat durant el desenvolupament de les parts principals del projecte, per poder comprovar que aquestes parts funcionen correctament. Per altre banda la segona part correspon a les proves post desenvolupament un cop les tasques bàsiques acabades per poder trobar bugs en l'aplicació.

9.2.1 Proves durant el desenvolupament

Les proves que s'han realitzat durant el desenvolupament cal remarcar que són proves concretes per poder comprovar el bon funcionament individual de cada una de les parts importants del projecte, per tant el bon funcionament de cada part no depèn de res més que d'ella mateixa. Aquest tipus de proves ens garanteixen que tant si funciona bé com si no funciona bé l'element del qual s'està realitzant la prova, l'error o el bon funcionament seran seus.

Connexió de Java amb R

La connexió de Java amb R és una de les parts més importants del projecte per tant era important fer una bona comprovació del bon funcionament entre Java i R, a continuació es mostraran un seguit de proves que s'han realitzat en el projecte per demostrar això.

Primers missatges d'R mostrats a Java

La primera prova que es va realitzar per comprovar la bona connexió entre Java i R, va ser una simple crida perquè R generes una sortida textual, i des de Java poder recollir aquesta informació i mostrar-la per la pantalla de sortida. A continuació en la Figura 98 veiem el codi de Java per realitzar la prova.

```
// first we get the session info
re.eval("a <- capture.output(sessionInfo())");
OutputElement e = new OutputForR(re.eval("a").asStringArray());
outputPanel.addOutput(e);


// after we get the system variables

re.eval("a <- capture.output(Sys.getenv())");
e = new OutputForR(re.eval("a").asStringArray());
outputPanel.addOutput(e);

// finally the capabilities

re.eval("a <- capture.output(capabilities())");
e = new OutputForR(re.eval("a").asStringArray());
outputPanel.addOutput(e);
```

Figura 98: Codi dels primers missatges rebuts d'R.

Com podem veure en la imatge, es fan tres crides a R per rebre diferent informació: la funció `sessionInfo()`, la funció `Sys.getenv()` i finalment la funció `capabilities()`.

A continuació en la Figura 99 la prova per verificar que es rep correctament la informació d'R i que es mostra també correctament en la finestra de sortida de missatges.

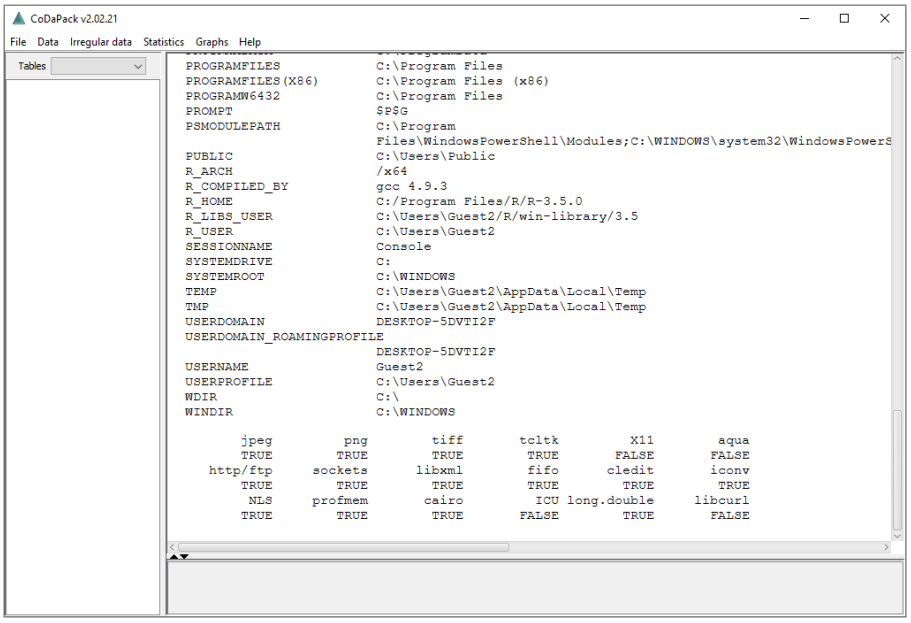


Figura 99: Prova dels primers missatges rebuts d'R.

Primers data frames creats des de Java per R

A continuació de comprovar que es rebien dades d'R correctament també s'havia de fer una comprovació de com s'enviaven les dades de Java a R, més concretament un dels tipus de dades més importants que són els data frames. Per tant aquesta prova consisteix en la realització d'un data frame, una crida a R per retornar-nos el data frame que hem creat i que en principi està guardat en la sessió d'R actual i finalment mostrar-lo per pantalla en la finestra de sortides.
A continuació en la Figura 100 el codi per recollir el data frame que hem creat i mostrar-lo per pantalla.

```
re.eval("output <- capture.output(X)");
String[] output = re.eval("output").asStringArray();
outputPanel.addOutput(new OutputForR(output));
```

Figura 100: Codi dels primers data frames creats per R.

Per verificar que el data frame s'ha creat correctament en R, mostrem el data frame d'R per la finestra de sortida de missatges, i com coincideix amb el data frame que volíem crear ens indica que s'ha creat correctament a R i amb les dades introduïdes. Podem veure'n un exemple en la Figura 101.

	neogl_atl	neogl_pach	glob_obesa	glob_triloba
1	0.74	0.19	0.03	0.04
2	0.58	0.29	0.01	0.12
3	0.58	0.19	0.22	0.01
4	0.61	0.28	0.08	0.03
5	0.82	0.13	0.02	0.03
6	0.48	0.38	0.01	0.13
7	0.59	0.38	0.00	0.03
8	0.76	0.12	0.09	0.03
9	0.81	0.12	0.04	0.03
10	0.68	0.23	0.05	0.04
11	0.72	0.20	0.04	0.04
12	0.62	0.27	0.09	0.02
13	0.45	0.25	0.29	0.01
14	0.66	0.25	0.06	0.03
15	0.85	0.13	0.01	0.01
16	0.75	0.09	0.15	0.01
17	0.69	0.25	0.00	0.06
18	0.76	0.10	0.11	0.03
19	0.66	0.29	0.01	0.04
20	0.66	0.24	0.06	0.04
21	0.50	0.46	0.00	0.40
22	0.65	0.25	0.05	0.05
23	0.60	0.35	0.02	0.03
24	0.40	0.27	0.01	0.32
25	0.60	0.10	0.30	0.00
26	0.60	0.10	0.29	0.01
27	0.59	0.39	0.01	0.01
28	0.58	0.39	0.01	0.02
29	0.61	0.34	0.02	0.03
30	0.39	0.49	0.12	0.00

Figura 101: Prova dels primers data frames rebuts d'R.

WebView

Una de les següents proves realitzades en el projecte, és amb el canvi de la finestra de sortida, el que es va fer és modificar la tecnologia i actualitzar la consola a WebView, que el que fa bàsicament és mostrar contingut web, per tant codi com HTML, CSS, JavaScript...

Per tal de poder fer proves i no fer-les en el mateix projecte, es va crear un subprojecte per poder comprovar el bon funcionament d'aquesta nova tecnologia, a continuació en la Figura 102, una prova del funcionament de la nova consola amb WebView.

Figura 102: Prova de la consola amb WebView.

En la imatge poder veure uns quants missatges de sortida de prova. Aquests missatges de prova es troben en un fitxer HTML que es carrega en la consola per mostrar-ho. A continuació en la Figura 103 es mostra el corresponent arxiu HTML associat a la sortida de la figura Figura 102.

Figura 103: HTML associat a la prova de la sortida del WebView.

MathJax

Finalment l'última prova que comentarem, és la creació dels menús de help amb els arxius YAML i l'eina MathJax que ens permet crear formules matemàtiques amb llenguatge Tex.

Per tal de fer proves, es va crear un arxiu YAML amb el format de help que volem molt senzill i amb un exemple de fórmula matemàtica per poder provar el bon funcionament de l'eina. A continuació en la Figura 104 veurem el menú de Help de prova.

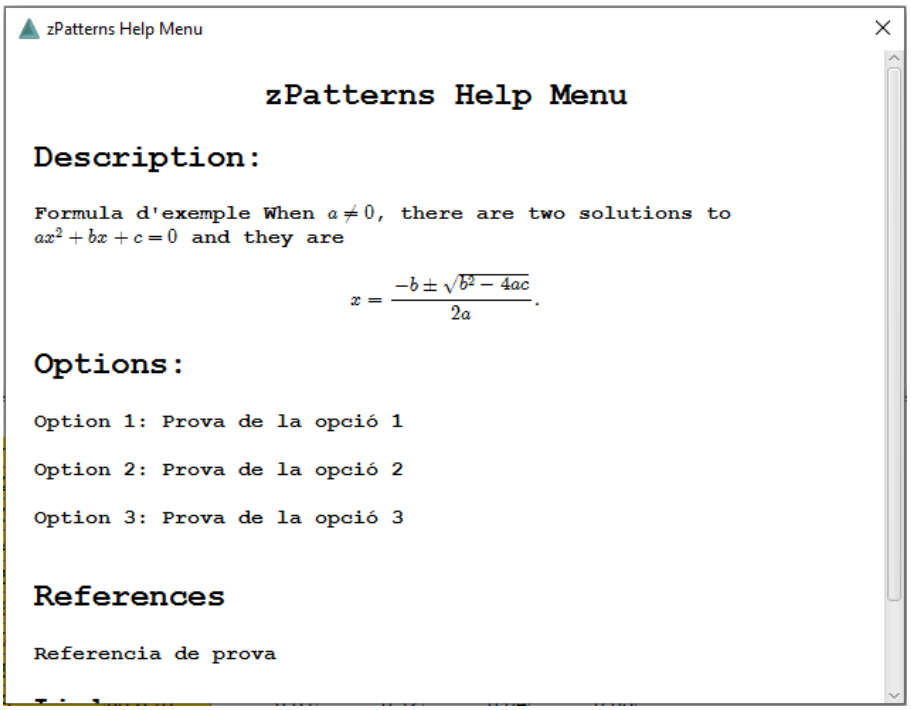


Figura 104: Prova del menú de help amb MathJax.

Com podem observar en la imatge, veiem el menú de help amb diferents parts i alguns elements matematics com formules generades amb MathJax. Finalment mostrem en la Figura 105 l'arxiu YAML corresponent a aquesta sortida de prova que acabem de mostrar.

```
---
name: Títol de prova
description: "Formula d'exemple When $a \\ne 0$, there are two solutions to
    \\(ax^2 + bx + c = 0\\) and they are $$x = {-b \\pm \\sqrt{b^2-4ac} \\over
    2a}.$$"
options:
  - Option 1: Prova de la opció 1
  - Option 2: Prova de la opció 2
  - Option 3: Prova de la opció 3
references: Referencia de prova
links:
  - link1: https://www.linkdeprova.com
```

Figura 105: YAML associat al menú de prova amb MathJax.

9.2.2 Proves post desenvolupament

Per comprovar que tot funciona correctament un cop el producte acabat, s'han realitzat un seguit de proves que comentarem seguidament en aquest apartat.

Un cop el producte acabat, es van realitzar un seguit d'iteracions. La primera iteració que es va realitzar, sorgien errors amb scripts d'R i es van haver d'introduir els missatges d'errors per poder depurar millor els errors.

Després d'aquesta iteració, es va produir una versió beta del producte final on els professors del grup anaven provant les diferents versions del CoDaPack i s'anaven depurant de mica en mica els errors que s'anaven trobant.

Cal comentar que les proves es van fer descarregant cada tester la versió beta i utilitzant els seus propis conjunts de dades, sense cap indicació per part nostre i diferents dels utilitzats en les proves de la primera fase.

10 Implantació i resultats

10.1 Implantació

La implantació que correspon a aquest apartat fa referència en la posada en marxa del producte, per tant aquí comentarem els aspectes que han fet falta per posar en marxa el projecte i per tant la nova versió del CoDaPack.

Primer de tot comentarem els instal·ladors que s'ha fet de la nova versió del CoDaPack implementada en el projecte, aquesta nova versió és la 2.03.01. S'han realitzat instal·ladors tant per Windows com per MacOS com es comenta en la secció **[9.1.6]**.

Aquests instal·ladors integren tant la màquina virtual de Java que farà servir l'aplicació com la seva pròpia versió d'R. Per tant l'usuari només ha d'instal·lar CoDaPack i les dependències que fan falta ja s'instal·len juntament de manera transparent per l'usuari. En el cas del sistema operatiu MacOS només s'instal·laria R si l'usuari no tingués la versió que es fa servir en el CoDaPack.

Un cop fet els instal·ladors per les diferents versions de sistemes operatius, el que s'ha fet és pujar-los en el servidor de manera pública perquè els usuaris finals puguin gaudir del producte simplement baixant-se l'aplicatiu com es pot veure en la Figura 106.

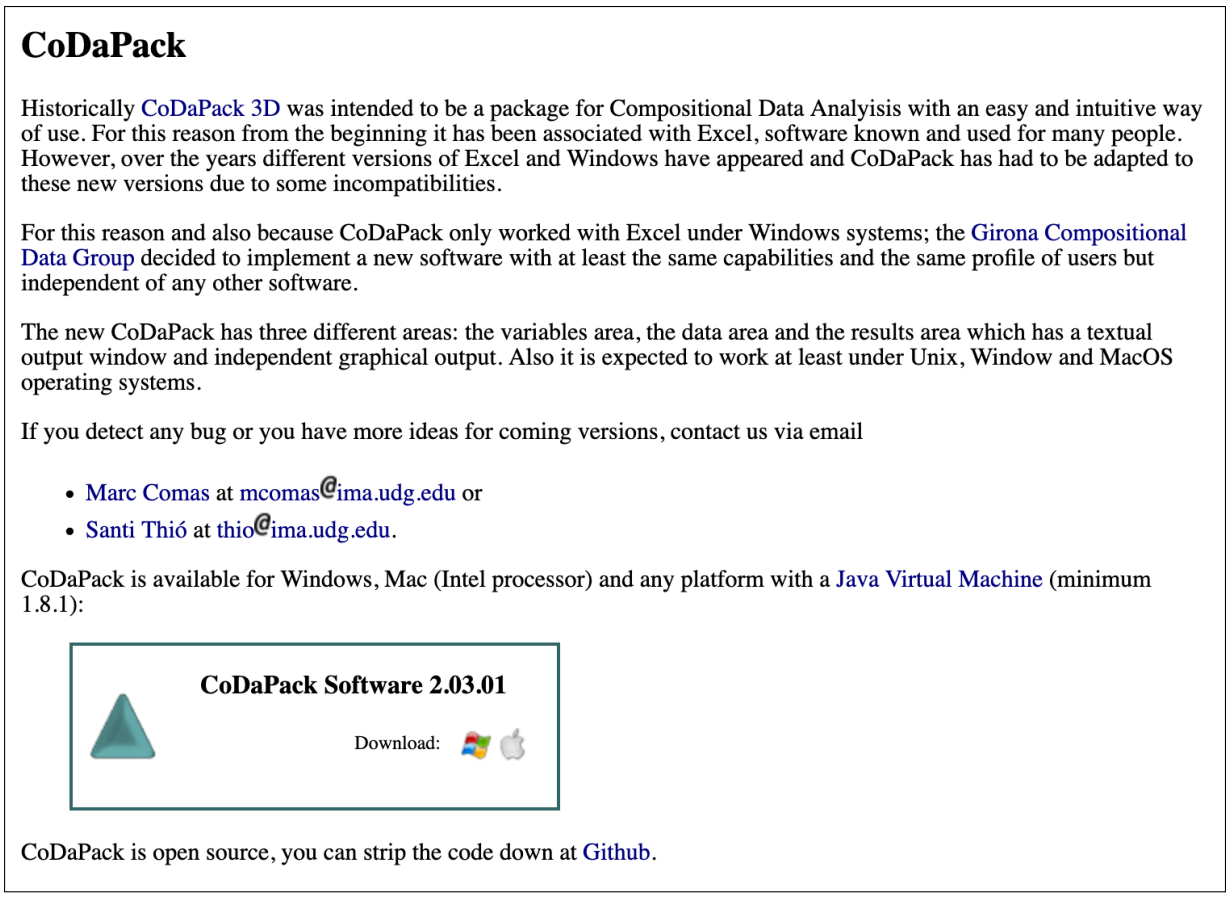


Figura 106: Web per descarregar el CoDaPack.

10.2 Resultats

En aquest apartat el que es pretén és mostrar els resultats finals obtinguts en aquest TFG. El que farem per tant és mostrar clarament el grau d'assoliment dels objectius que s'han plantejat des d'un inici en el projecte, intentarem realitzar tot això mitjançant exemples de funcionament de què s'ha fet.

Cal remarcar que en aquest apartat no es mostraran tots i cada un dels requisits funcionals que s'han esmentat en l'apartat **[6.1]**, ja que són molts, per tant s'ha decidit dividir els resultats d'igual manera però triar alguns requisits per cada un dels apartats. També s'ha afegit un apartat extra dedicat als menús de desenvolupador que s'han realitzat.

10.2.1 Millores del CoDaPack amb Java

Create data frame

Aquesta funcionalitat consisteix en crear un data frame de zero, per tant l'usuari pot triar el nombre de variables que contindrà el nou data frame, i el nombre d'observacions que hi haurà, tenint en compte que la prima fila s'utilitza per definir el nom de la variable.

En la Figura 107 podem veure un exemple de com un usuari podria introduir unes dades per crear un data frame, les podria anar creant manualment o simplement fent copy i paste des d'un excel per exemple. En aquest cas l'usuari ha definit cinc variables, on dues són variables categòriques i les altres són variables numèriques.

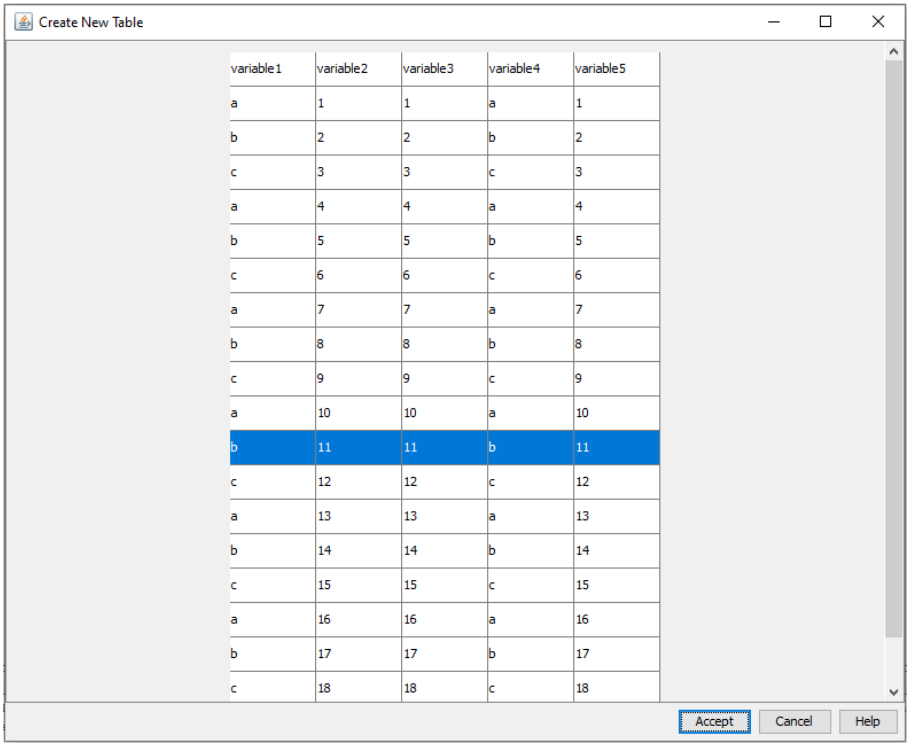


Figura 107: Dades introduïdes per crear un data frame.

Un cop introduïdes les dades, l'usuari simplement ha de fer clic en acceptar i el nou data frame apareixerà creat en la taula de dades com es mostra en la Figura 108.

	variable1	variable2	variable3	variable4	variable5
1	a	1.00	1.00	a	1.00
2	b	2.00	2.00	b	2.00
3	c	3.00	3.00	c	3.00
4	a	4.00	4.00	a	4.00
5	b	5.00	5.00	b	5.00
6	c	6.00	6.00	c	6.00
7	a	7.00	7.00	a	7.00
8	b	8.00	8.00	b	8.00
9	c	9.00	9.00	c	9.00
10	a	10.00	10.00	a	10.00
11	b	11.00	11.00	b	11.00
12	c	12.00	12.00	c	12.00
13	a	13.00	13.00	a	13.00
14	b	14.00	14.00	b	14.00
15	c	15.00	15.00	c	15.00
16	a	16.00	16.00	a	16.00
17	b	17.00	17.00	b	17.00
18	c	18.00	18.00	c	18.00
19	a	19.00	19.00	a	19.00

Figura 108: Data frame creat amb les dades introduïdes.

Filtrar observacions

La següent funcionalitat el que ens permet és poder filtrar dades d'un data frame a partir del valor d'una variable categòrica. Aquesta funcionalitat ja que és molt útil poder separar dades per certs valors d'una variable. En la figura Figura 109 observem un exemple de dades inicials, i ens imaginem que el que volem és només tenir les dades que fan referència a zones de l'est, per tant el que hem de fer és simplement filtrar pel valor "*East*" de la variable "*EastWest*.

	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	10.10	1.40	0.50	8.90	Albania	AL	Med	East
2	8.90	14.00	4.30	19.90	Austria	AT	North	West
3	13.50	9.30	4.10	17.50	Belgium	BE	North	West
4	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
5	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
6	10.60	10.80	3.70	25.00	Danemark	DK	North	West
7	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
8	9.50	4.90	2.70	33.70	Finland	FI	North	West
9	18.00	9.90	3.30	19.50	France	FR	Med	West
10	10.20	3.00	2.80	17.60	Greece	GR	Med	West
11	5.30	12.40	2.90	9.70	Hungary	HU	North	East
12	13.90	10.00	4.70	25.80	Ireland	EI	North	West
13	9.00	5.10	2.90	13.70	Italy	IT	Med	West
14	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
15	9.40	4.70	2.70	23.30	Norway	NO	North	West
16	6.90	10.20	2.70	19.30	Poland	PL	North	East
17	6.20	3.70	1.10	4.90	Portugal	PO	Med	West
18	6.20	6.30	1.50	11.10	Romania	RO	Med	East
19	7.10	3.40	3.10	8.60	Spain	ES	Med	West
20	9.90	7.80	3.50	24.70	Sweden	SE	North	West
21	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
22	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
23	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
24	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
25	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East

Figura 109: Dades inicials Filtrar observacions.

Un cop introduït el valor per al qual volem filtrar, obtindríem un nou data frame, només amb les observacions que continguin el valor indicat de la variable categòrica seleccionada, com es pot observar com a exemple en la Figura 110.

	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	10.10	1.40	0.50	8.90	Albania	AL	Med	East
2	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
3	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
4	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
5	5.30	12.40	2.90	9.70	Hungary	HU	North	East
6	6.90	10.20	2.70	19.30	Poland	PL	North	East
7	6.20	6.30	1.50	11.10	Romania	RO	Med	East
8	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
9	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East

Figura 110: Dades un cop utilitzat el filtre.

Guardar gràfics en format PDF

En aquesta funcionalitat, se'ns permet en els gràfics que són generats amb Java, poder exportar-los amb un format PDF. Seguidament en la Figura 111 podem veure un exemple d'un gràfic Dendogram, generat amb Java.

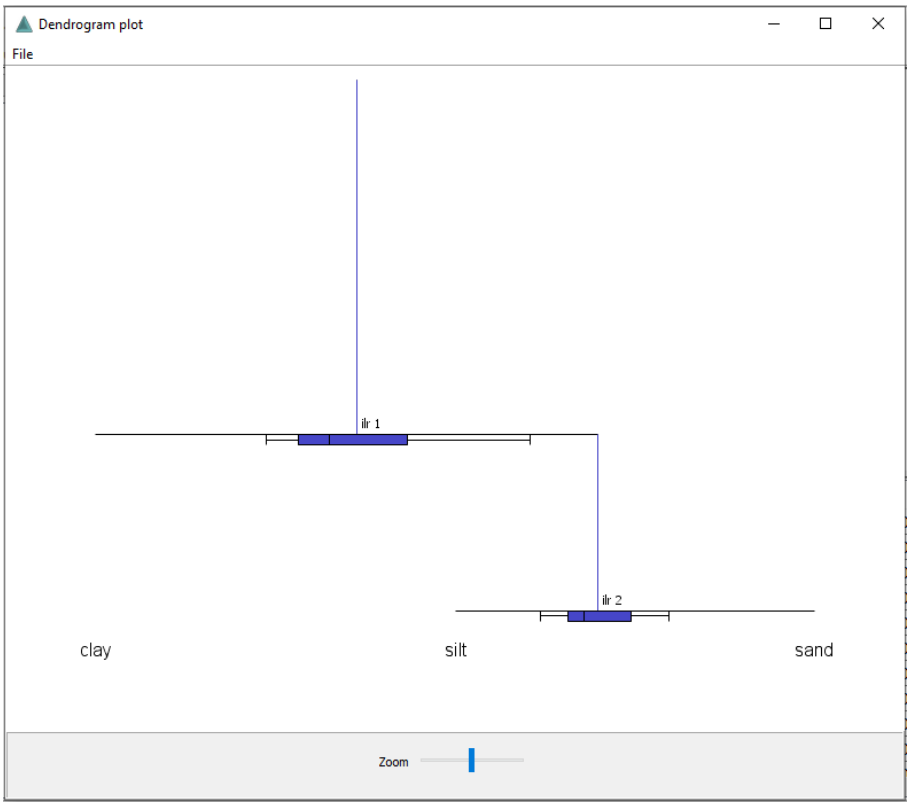


Figura 111: Gràfic inicial generat amb Java.

L'usuari simplement hauria d'exportar el gràfic, i simplement seleccionar el format PDF indicant el nom del nou arxiu i la localització on es guardarà l'arxiu. A continuació en la Figura 112 podem veure el gràfic que s'havia generat amb Java exportant en format PDF i obert amb un visualitzador d'arxius PDF.

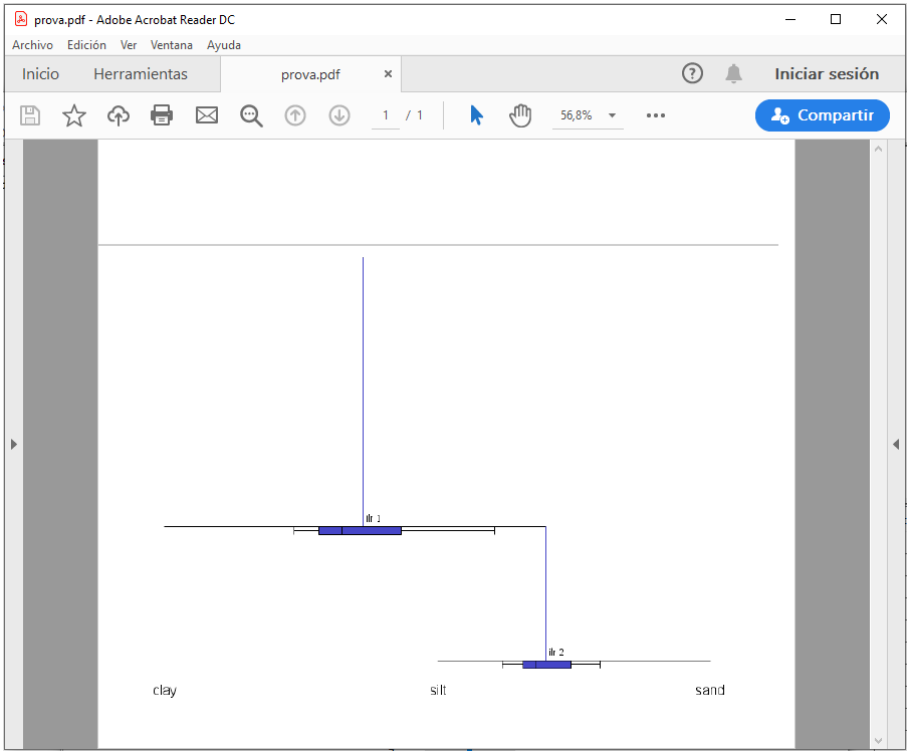


Figura 112: Gràfic exportat amb format PDF.

10.2.2 Connexió directa amb R

Classical Univariate Normality test

Aquesta rutina ens permetrà, poder realitzar dos tests sobre una variable numèrica seleccionada, el primer test que es pot realitzar és el test de Shapiro-Wilk, i el segon test és el test de Kolmogorov-Smirnov.
A continuació, en la Figura 113 les dades inicials que es fan servir per mostrar el funcionament, si ens fixem seleccionem la variable "M".

	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	10.10	1.40	0.50	8.90	Albania	AL	Med	East
2	8.90	14.00	4.30	19.90	Austria	AT	North	West
3	13.50	9.30	4.10	17.50	Belgium	BE	North	West
4	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
5	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
6	10.60	10.80	3.70	25.00	Danemark	DK	North	West
7	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
8	9.50	4.90	2.70	33.70	Finland	FI	North	West
9	18.00	9.90	3.30	19.50	France	FR	Med	West
10	10.20	3.00	2.80	17.60	Greece	GR	Med	West
11	5.30	12.40	2.90	9.70	Hungary	HU	North	East
12	13.90	10.00	4.70	25.80	Ireland	EI	North	West
13	9.00	5.10	2.90	13.70	Italy	IT	Med	West
14	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
15	9.40	4.70	2.70	23.30	Norway	NO	North	West
16	6.90	10.20	2.70	19.30	Poland	PL	North	East
17	6.20	3.70	1.10	4.90	Portugal	PO	Med	West
18	6.20	6.30	1.50	11.10	Romania	RO	Med	East
19	7.10	3.40	3.10	8.60	Spain	ES	Med	West
20	9.90	7.80	3.50	24.70	Sweden	SE	North	West
21	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
22	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
23	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
24	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
25	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East

Figura 113: Dades inicials Classical Univariate Normality test.

A continuació en la Figura 114 podem veure les sortides textuals després de la realització de cada un dels tests amb les dades seleccionades.

```
Shapiro-Wilk Test

        Shapiro-Wilk normality test

data:  M
W = 0.96088, p-value = 0.4323


Kolmogorov-Smirnov Test

        One-sample Kolmogorov-Smirnov test

data:  M
D = 0.12126, p-value = 0.8559
alternative hypothesis: two-sided
```

Figura 114: Sortida textual Classical Univariate Normality test.

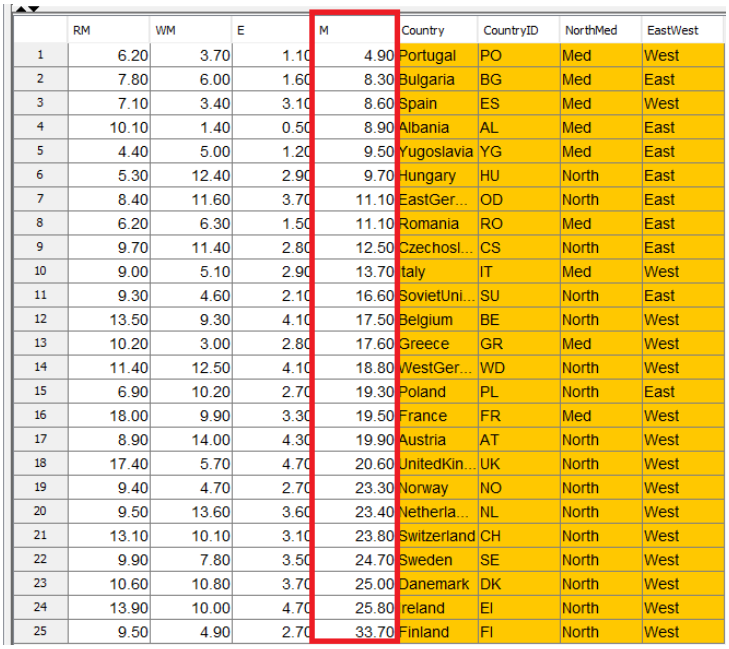
Ordenar les dades

Aquesta funcionalitat que mostrarem a continuació ens permet poder ordenar un conjunt de dades, de manera creixent o de manera decreixent. En aquest exemple que mostrarem a continuació, utilitzarem el conjunt de variables que es mostra en la Figura 115, i la variable que utilitzarem per ordenar és la variable anomenada "M".

	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	10.10	1.40	0.50	8.90	Albania	AL	Med	East
2	8.90	14.00	4.30	19.90	Austria	AT	North	West
3	13.50	9.30	4.10	17.50	Belgium	BE	North	West
4	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
5	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
6	10.60	10.80	3.70	25.00	Danemark	DK	North	West
7	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
8	9.50	4.90	2.70	33.70	Finland	FI	North	West
9	18.00	9.90	3.30	19.50	France	FR	Med	West
10	10.20	3.00	2.80	17.60	Greece	GR	Med	West
11	5.30	12.40	2.90	9.70	Hungary	HU	North	East
12	13.90	10.00	4.70	25.80	Ireland	EI	North	West
13	9.00	5.10	2.90	13.70	Italy	IT	Med	West
14	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
15	9.40	4.70	2.70	23.30	Norway	NO	North	West
16	6.90	10.20	2.70	19.30	Poland	PL	North	East
17	6.20	3.70	1.10	4.90	Portugal	PO	Med	West
18	6.20	6.30	1.50	11.10	Romania	RO	Med	East
19	7.10	3.40	3.10	8.60	Spain	ES	Med	West
20	9.90	7.80	3.50	24.70	Sweden	SE	North	West
21	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
22	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
23	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
24	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
25	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East

Figura 115: Dades inicials Ordenar les dades.

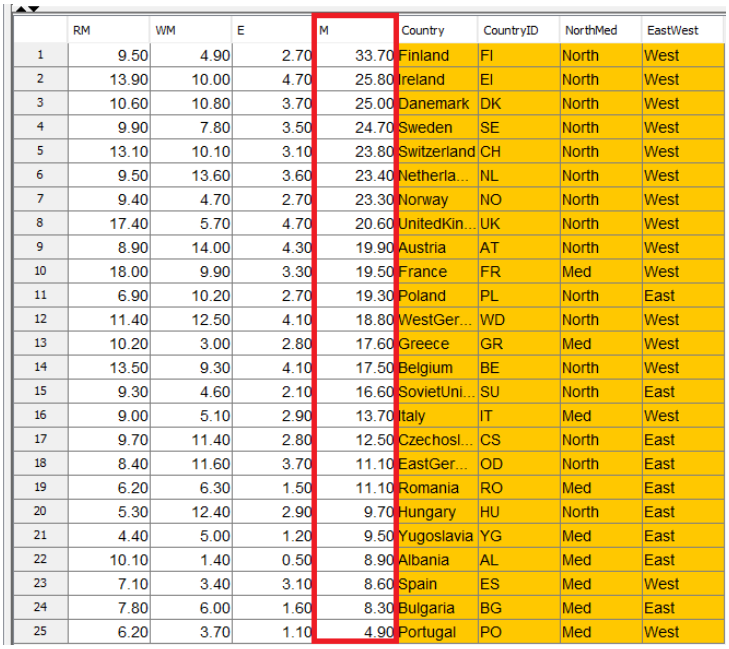
A continuació es mostra en la Figura 116 les dades ordenades creixentment, per tant la variable que s'ha seleccionat anirà dels valors més petits als valors més grans.



	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	6.20	3.70	1.10	4.90	Portugal	PO	Med	West
2	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
3	7.10	3.40	3.10	8.60	Spain	ES	Med	West
4	10.10	1.40	0.50	8.90	Albania	AL	Med	East
5	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East
6	5.30	12.40	2.90	9.70	Hungary	HU	North	East
7	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
8	6.20	6.30	1.50	11.10	Romania	RO	Med	East
9	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
10	9.00	5.10	2.90	13.70	Italy	IT	Med	West
11	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
12	13.50	9.30	4.10	17.50	Belgium	BE	North	West
13	10.20	3.00	2.80	17.60	Greece	GR	Med	West
14	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
15	6.90	10.20	2.70	19.30	Poland	PL	North	East
16	18.00	9.90	3.30	19.50	France	FR	Med	West
17	8.90	14.00	4.30	19.90	Austria	AT	North	West
18	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
19	9.40	4.70	2.70	23.30	Norway	NO	North	West
20	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
21	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
22	9.90	7.80	3.50	24.70	Sweden	SE	North	West
23	10.60	10.80	3.70	25.00	Danemark	DK	North	West
24	13.90	10.00	4.70	25.80	Ireland	EI	North	West
25	9.50	4.90	2.70	33.70	Finland	FI	North	West

Figura 116: Dades un cop ordenades creixentment.

A continuació es mostra en la Figura 117 les dades ordenades decreixentment, per tant la variable que s'ha seleccionat anirà dels valors més grans als valors més petits.

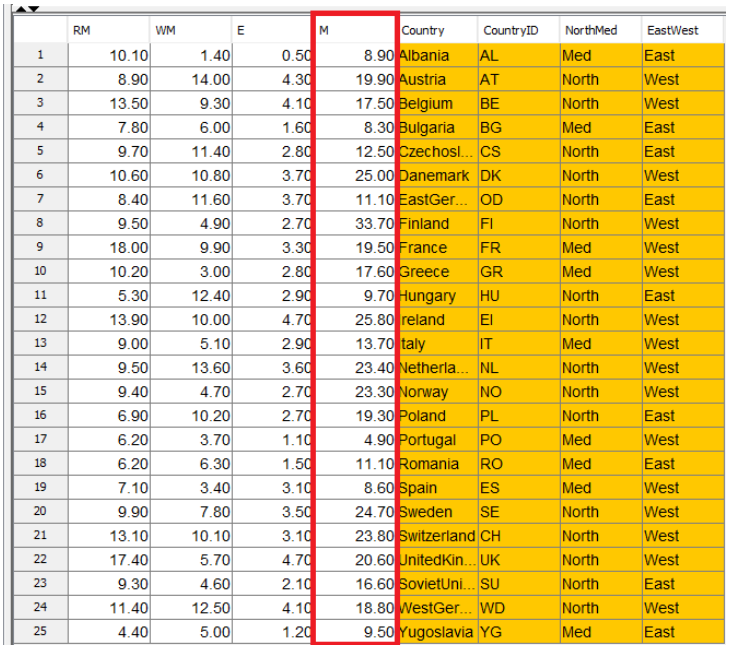


	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	9.50	4.90	2.70	33.70	Finland	FI	North	West
2	13.90	10.00	4.70	25.80	Ireland	EI	North	West
3	10.60	10.80	3.70	25.00	Danemark	DK	North	West
4	9.90	7.80	3.50	24.70	Sweden	SE	North	West
5	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
6	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
7	9.40	4.70	2.70	23.30	Norway	NO	North	West
8	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
9	8.90	14.00	4.30	19.90	Austria	AT	North	West
10	18.00	9.90	3.30	19.50	France	FR	Med	West
11	6.90	10.20	2.70	19.30	Poland	PL	North	East
12	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
13	10.20	3.00	2.80	17.60	Greece	GR	Med	West
14	13.50	9.30	4.10	17.50	Belgium	BE	North	West
15	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
16	9.00	5.10	2.90	13.70	Italy	IT	Med	West
17	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
18	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
19	6.20	6.30	1.50	11.10	Romania	RO	Med	East
20	5.30	12.40	2.90	9.70	Hungary	HU	North	East
21	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East
22	10.10	1.40	0.50	8.90	Albania	AL	Med	East
23	7.10	3.40	3.10	8.60	Spain	ES	Med	West
24	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
25	6.20	3.70	1.10	4.90	Portugal	PO	Med	West

Figura 117: Dades un cop ordenades decreixentment.

Subconjunt de dades

Aquesta funcionalitat és molt interessant perquè ens permet fer un subconjunt de dades a partir d'una expressió regular que apliquem a unes variables. En aquest cas realitzarem un exemple de com realitzaríem un subconjunt de dades utilitzant les dades que es mostren en la Figura 118 i dient que volem només les dades on el valor de la variable "M" sigui més gran de quinze com es mostra l'expressió en la Figura 119.



	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	10.10	1.40	0.50	8.90	Albania	AL	Med	East
2	8.90	14.00	4.30	19.90	Austria	AT	North	West
3	13.50	9.30	4.10	17.50	Belgium	BE	North	West
4	7.80	6.00	1.60	8.30	Bulgaria	BG	Med	East
5	9.70	11.40	2.80	12.50	Czechosl...	CS	North	East
6	10.60	10.80	3.70	25.00	Danemark	DK	North	West
7	8.40	11.60	3.70	11.10	EastGer...	OD	North	East
8	9.50	4.90	2.70	33.70	Finland	FI	North	West
9	18.00	9.90	3.30	19.50	France	FR	Med	West
10	10.20	3.00	2.80	17.60	Greece	GR	Med	West
11	5.30	12.40	2.90	9.70	Hungary	HU	North	East
12	13.90	10.00	4.70	25.80	Ireland	EI	North	West
13	9.00	5.10	2.90	13.70	Italy	IT	Med	West
14	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
15	9.40	4.70	2.70	23.30	Norway	NO	North	West
16	6.90	10.20	2.70	19.30	Poland	PL	North	East
17	6.20	3.70	1.10	4.90	Portugal	PO	Med	West
18	6.20	6.30	1.50	11.10	Romania	RO	Med	East
19	7.10	3.40	3.10	8.60	Spain	ES	Med	West
20	9.90	7.80	3.50	24.70	Sweden	SE	North	West
21	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
22	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
23	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
24	11.40	12.50	4.10	18.80	WestGer...	WD	North	West
25	4.40	5.00	1.20	9.50	Yugoslavia	YG	Med	East

Figura 118: Dades inicials Subconjunt de dades.

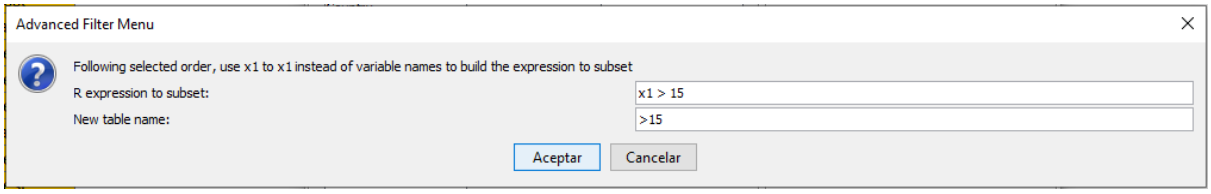
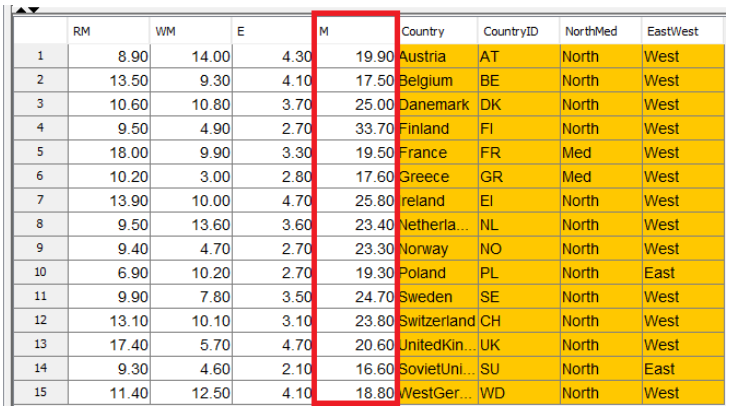


Figura 119: Expressió per realitzar el subconjunt.

Un cop indicada la funció, es crearà un nou data frame amb les observacions que compleixin l'expressió indicada en el formulari. Com podem observar en la Figura 120 s'ha creat un nou data frame on tots els valors de la variable "M" són superiors al valor quinze, per tant ha creat un subconjunt de dades amb l'expressió indicada per l'usuari.



	RM	WM	E	M	Country	CountryID	NorthMed	EastWest
1	8.90	14.00	4.30	19.90	Austria	AT	North	West
2	13.50	9.30	4.10	17.50	Belgium	BE	North	West
3	10.60	10.80	3.70	25.00	Danemark	DK	North	West
4	9.50	4.90	2.70	33.70	Finland	FI	North	West
5	18.00	9.90	3.30	19.50	France	FR	Med	West
6	10.20	3.00	2.80	17.60	Greece	GR	Med	West
7	13.90	10.00	4.70	25.80	Ireland	EI	North	West
8	9.50	13.60	3.60	23.40	Netherla...	NL	North	West
9	9.40	4.70	2.70	23.30	Norway	NO	North	West
10	6.90	10.20	2.70	19.30	Poland	PL	North	East
11	9.90	7.80	3.50	24.70	Sweden	SE	North	West
12	13.10	10.10	3.10	23.80	Switzerland	CH	North	West
13	17.40	5.70	4.70	20.60	UnitedKin...	UK	North	West
14	9.30	4.60	2.10	16.60	SovietUni...	SU	North	East
15	11.40	12.50	4.10	18.80	WestGer...	WD	North	West

Figura 120: Dades un cop aplicada l'expressió.

10.2.3 Execució d'scripts d'R

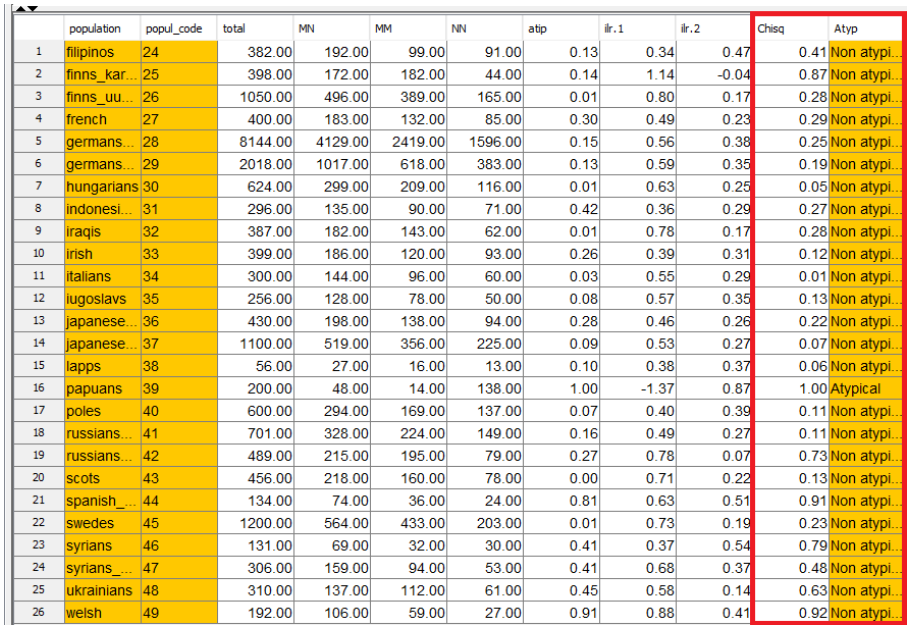
Atypicality test

Aquesta funcionalitat realitza la rutina Atypicality test, en la Figura 121 podem observar les dades inicials que s'utilitzaran per realitzar la rutina, recordem que aquesta rutina ens retorna dos elements. El primer element són noves variables en el data frame actual, i el segon element és una sortida textual.

	population	popul_code	total	MN	MM	NN	atip	ilr.1	ilr.2
1	filipinos	24	382.00	192.00	99.00	91.00	0.13	0.34	0.47
2	finns_kar...	25	398.00	172.00	182.00	44.00	0.14	1.14	-0.04
3	finns_uu...	26	1050.00	496.00	389.00	165.00	0.01	0.80	0.17
4	french	27	400.00	183.00	132.00	85.00	0.30	0.49	0.23
5	germans...	28	8144.00	4129.00	2419.00	1596.00	0.15	0.56	0.38
6	germans...	29	2018.00	1017.00	618.00	383.00	0.13	0.59	0.35
7	hungarians	30	624.00	299.00	209.00	116.00	0.01	0.63	0.25
8	indonesi...	31	296.00	135.00	90.00	71.00	0.42	0.36	0.29
9	iraqis	32	387.00	182.00	143.00	62.00	0.01	0.78	0.17
10	irish	33	399.00	186.00	120.00	93.00	0.26	0.39	0.31
11	italians	34	300.00	144.00	96.00	60.00	0.03	0.55	0.29
12	iugoslavs	35	256.00	128.00	78.00	50.00	0.08	0.57	0.35
13	japanese...	36	430.00	198.00	138.00	94.00	0.28	0.46	0.26
14	japanese...	37	1100.00	519.00	356.00	225.00	0.09	0.53	0.27
15	lapps	38	56.00	27.00	16.00	13.00	0.10	0.38	0.37
16	papuans	39	200.00	48.00	14.00	138.00	1.00	-1.37	0.87
17	poles	40	600.00	294.00	169.00	137.00	0.07	0.40	0.39
18	russians...	41	701.00	328.00	224.00	149.00	0.16	0.49	0.27
19	russians...	42	489.00	215.00	195.00	79.00	0.27	0.78	0.07
20	scots	43	456.00	218.00	160.00	78.00	0.00	0.71	0.22
21	spanish_...	44	134.00	74.00	36.00	24.00	0.81	0.63	0.51
22	swedes	45	1200.00	564.00	433.00	203.00	0.01	0.73	0.19
23	syrians	46	131.00	69.00	32.00	30.00	0.41	0.37	0.54
24	syrians_...	47	306.00	159.00	94.00	53.00	0.41	0.68	0.37
25	ukrainians	48	310.00	137.00	112.00	61.00	0.45	0.58	0.14
26	welsh	49	192.00	106.00	59.00	27.00	0.91	0.88	0.41

Figura 121: Dades inicials Atypicality test.

En la Figura 122 podem veure el primer dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge que han aparegut dues noves variables al final del data frame, una variable que és "Chisq" i és una variable numèrica i l'altre variable és "Atip" que és una variable categòrica que ens indica si l'observació és Atypica o no.



	population	popul_code	total	MN	MM	NN	atip	ilr.1	ilr.2	Chisq	Atyp
1	filipinos	24	382.00	192.00	99.00	91.00	0.13	0.34	0.47	0.41	Non atypi...
2	finns_kar...	25	398.00	172.00	182.00	44.00	0.14	1.14	-0.04	0.87	Non atypi...
3	finns_uu...	26	1050.00	496.00	389.00	165.00	0.01	0.80	0.17	0.28	Non atypi...
4	french	27	400.00	183.00	132.00	85.00	0.30	0.49	0.23	0.29	Non atypi...
5	germans...	28	8144.00	4129.00	2419.00	1596.00	0.15	0.56	0.38	0.25	Non atypi...
6	germans...	29	2018.00	1017.00	618.00	383.00	0.13	0.59	0.35	0.19	Non atypi...
7	hungarians	30	624.00	299.00	209.00	116.00	0.01	0.63	0.25	0.05	Non atypi...
8	indonesi...	31	296.00	135.00	90.00	71.00	0.42	0.36	0.29	0.27	Non atypi...
9	iraqis	32	387.00	182.00	143.00	62.00	0.01	0.78	0.17	0.28	Non atypi...
10	irish	33	399.00	186.00	120.00	93.00	0.26	0.39	0.31	0.12	Non atypi...
11	italians	34	300.00	144.00	96.00	60.00	0.03	0.55	0.29	0.01	Non atypi...
12	iugoslavs	35	256.00	128.00	78.00	50.00	0.08	0.57	0.35	0.13	Non atypi...
13	japanese...	36	430.00	198.00	138.00	94.00	0.28	0.46	0.26	0.22	Non atypi...
14	japanese...	37	1100.00	519.00	356.00	225.00	0.09	0.53	0.27	0.07	Non atypi...
15	lapps	38	56.00	27.00	16.00	13.00	0.10	0.38	0.37	0.06	Non atypi...
16	papuans	39	200.00	48.00	14.00	138.00	1.00	-1.37	0.87	1.00	Atypical
17	poles	40	600.00	294.00	169.00	137.00	0.07	0.40	0.39	0.11	Non atypi...
18	russians...	41	701.00	328.00	224.00	149.00	0.16	0.49	0.27	0.11	Non atypi...
19	russians...	42	489.00	215.00	195.00	79.00	0.27	0.78	0.07	0.73	Non atypi...
20	scots	43	456.00	218.00	160.00	78.00	0.00	0.71	0.22	0.13	Non atypi...
21	spanish_...	44	134.00	74.00	36.00	24.00	0.81	0.63	0.51	0.91	Non atypi...
22	swedes	45	1200.00	564.00	433.00	203.00	0.01	0.73	0.19	0.23	Non atypi...
23	syrians	46	131.00	69.00	32.00	30.00	0.41	0.37	0.54	0.79	Non atypi...
24	syrians_...	47	306.00	159.00	94.00	53.00	0.41	0.68	0.37	0.48	Non atypi...
25	ukrainians	48	310.00	137.00	112.00	61.00	0.45	0.58	0.14	0.63	Non atypi...
26	welsh	49	192.00	106.00	59.00	27.00	0.91	0.88	0.41	0.92	Non atypi...

Figura 122: Noves variables Atypicality test.

Per últim en la Figura 123 observem el segon element que ens retorna la rutina i que correspon a la sortida textual. Bàsicament la sortida ens indica quines observacions són atípiques, en aquest cas l'observació numero 16.

```
Atypicality Index:

POTENTIAL OUTLIERS

atypical observations  [1] 16
```

Figura 123: Sortida textual Atypicality test.

Cluster K-means

Aquesta funcionalitat realitza la rutina Cluster K-means, en la Figura 124 podem observar les dades inicials que s'utilitzaran per realitzar la rutina, aquesta rutina ens retorna tres elements però en aquest cas en concret amb les opcions triades en el menú no es mostra cap gràfica. El primer element són gràfics, el segon són noves variables en el data frame actual, i l'últim element és una sortida textual.

	RM	WM	E	M	F	C	S	N	FV
1	10.10	1.40	0.50	8.90	0.20	42.30	0.60	5.50	1.70
2	8.90	14.00	4.30	19.90	2.10	28.00	3.60	1.30	4.30
3	13.50	9.30	4.10	17.50	4.50	26.60	5.70	2.10	4.00
4	7.80	6.00	1.60	8.30	1.20	56.70	1.10	3.70	4.20
5	9.70	11.40	2.80	12.50	2.00	34.30	5.00	1.10	4.00
6	10.60	10.80	3.70	25.00	9.90	21.90	4.80	0.70	2.40
7	8.40	11.60	3.70	11.10	5.40	24.60	6.50	0.80	3.60
8	9.50	4.90	2.70	33.70	5.80	26.30	5.10	1.00	1.40
9	18.00	9.90	3.30	19.50	5.70	28.10	4.80	2.40	6.50
10	10.20	3.00	2.80	17.60	5.90	41.70	2.20	7.80	6.50
11	5.30	12.40	2.90	9.70	0.30	40.10	4.00	5.40	4.20
12	13.90	10.00	4.70	25.80	2.20	24.00	6.20	1.60	2.90
13	9.00	5.10	2.90	13.70	3.40	36.80	2.10	4.30	6.70
14	9.50	13.60	3.60	23.40	2.50	22.40	4.20	1.80	3.70
15	9.40	4.70	2.70	23.30	9.70	23.00	4.60	1.60	2.70
16	6.90	10.20	2.70	19.30	3.00	36.10	5.90	2.00	6.60
17	6.20	3.70	1.10	4.90	14.20	27.00	5.90	4.70	7.90
18	6.20	6.30	1.50	11.10	1.00	49.60	3.10	5.30	2.80
19	7.10	3.40	3.10	8.60	7.00	29.20	5.70	5.90	7.20
20	9.90	7.80	3.50	24.70	7.50	19.50	3.70	1.40	2.00
21	13.10	10.10	3.10	23.80	2.30	25.60	2.80	2.40	4.90
22	17.40	5.70	4.70	20.60	4.30	24.30	4.70	3.40	3.30
23	9.30	4.60	2.10	16.60	3.00	43.60	6.40	3.40	2.90
24	11.40	12.50	4.10	18.80	3.40	18.60	5.20	1.50	3.80
25	4.40	5.00	1.20	9.50	0.60	55.90	3.00	5.70	3.20

Figura 124: Dades inicials Cluster K-means.

En la Figura 125 podem veure les opcions que farem servir per mostrar l'exemple de funcionament de la rutina Cluster K-means. En aquest cas utilitzarem totes les variables del data frame, l'opció "Number of Clusters" amb un valor de 3 i el nom de la nova columna del grup ho deixem per defecte com a "Group".

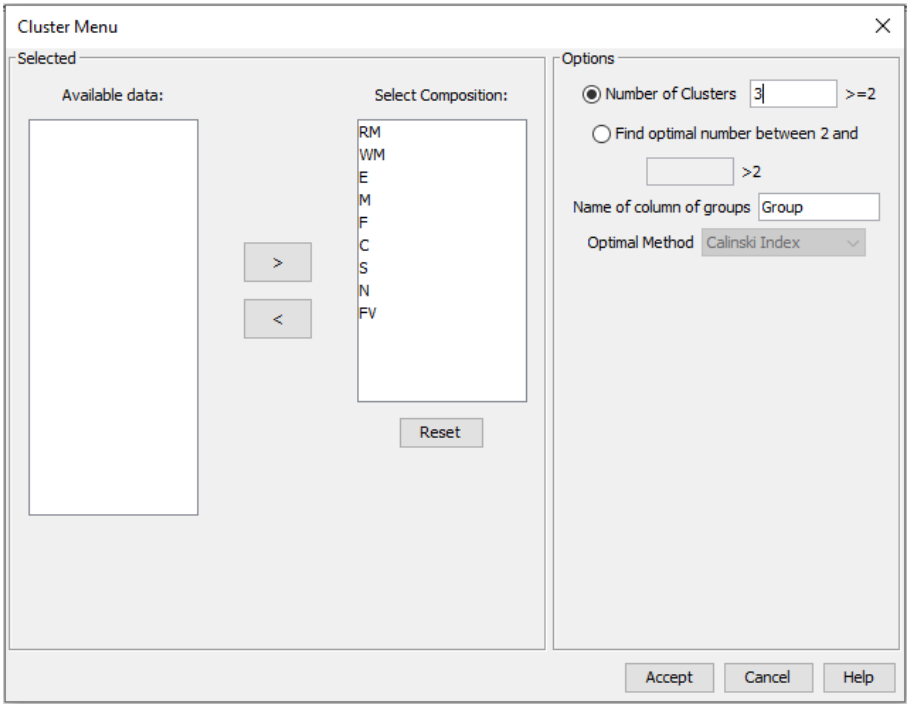


Figura 125: Opcions triades Cluster K-means.

En la Figura 126 podem veure el primer dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge que ha aparegut una nova variables al final del data frame, aquesta variable és l'anomenada "Group", que és una variable categòrica i que representa el grup de cluster que correspon la observació.

	RM	WM	E	M	F	C	S	N	FV	Group
1	10.10	1.40	0.50	8.90	0.20	42.30	0.60	5.50	1.70	3
2	8.90	14.00	4.30	19.90	2.10	28.00	3.60	1.30	4.30	2
3	13.50	9.30	4.10	17.50	4.50	26.60	5.70	2.10	4.00	2
4	7.80	6.00	1.60	8.30	1.20	56.70	1.10	3.70	4.20	3
5	9.70	11.40	2.80	12.50	2.00	34.30	5.00	1.10	4.00	2
6	10.60	10.80	3.70	25.00	9.90	21.90	4.80	0.70	2.40	2
7	8.40	11.60	3.70	11.10	5.40	24.60	6.50	0.80	3.60	2
8	9.50	4.90	2.70	33.70	5.80	26.30	5.10	1.00	1.40	2
9	18.00	9.90	3.30	19.50	5.70	28.10	4.80	2.40	6.50	2
10	10.20	3.00	2.80	17.60	5.90	41.70	2.20	7.80	6.50	1
11	5.30	12.40	2.90	9.70	0.30	40.10	4.00	5.40	4.20	3
12	13.90	10.00	4.70	25.80	2.20	24.00	6.20	1.60	2.90	2
13	9.00	5.10	2.90	13.70	3.40	36.80	2.10	4.30	6.70	1
14	9.50	13.60	3.60	23.40	2.50	22.40	4.20	1.80	3.70	2
15	9.40	4.70	2.70	23.30	9.70	23.00	4.60	1.60	2.70	2
16	6.90	10.20	2.70	19.30	3.00	36.10	5.90	2.00	6.60	2
17	6.20	3.70	1.10	4.90	14.20	27.00	5.90	4.70	7.90	1
18	6.20	6.30	1.50	11.10	1.00	49.60	3.10	5.30	2.80	3
19	7.10	3.40	3.10	8.60	7.00	29.20	5.70	5.90	7.20	1
20	9.90	7.80	3.50	24.70	7.50	19.50	3.70	1.40	2.00	2
21	13.10	10.10	3.10	23.80	2.30	25.60	2.80	2.40	4.90	2
22	17.40	5.70	4.70	20.60	4.30	24.30	4.70	3.40	3.30	2
23	9.30	4.60	2.10	16.60	3.00	43.60	6.40	3.40	2.90	2
24	11.40	12.50	4.10	18.80	3.40	18.60	5.20	1.50	3.80	2
25	4.40	5.00	1.20	9.50	0.60	55.90	3.00	5.70	3.20	3

Figura 126: Noves variables Cluster K-means.

Per últim en la Figura 127 observem el segon element que ens retorna la rutina i que correspon a la sortida textual. La sortida ens indica alguns paràmetres de la rutina com per exemple l'índex de calinski, el nombre d'elements en cada grup del cluster, les mitjanes de cluster o el vector de clustering.

```
Cluster K-means:

CLUSTER K-MEANS

calinski index =  [1] 15.09987

Average Silhouette =  [1] 0.3823071

K-means clustering with 3 clusters of sizes 4, 16, 5

Cluster means:
      ilr.1     ilr.2      ilr.3      ilr.4     ilr.5     ilr.6      ilr.7
1 0.5381605 0.7064872 -0.7789622 -0.2392575 -1.657801 0.6727841  0.1641811
2 0.1403497 0.8645517 -0.9337877  0.7317213 -1.116052 0.6152209  1.5581903
3 0.1760136 1.1902665 -0.8563386  1.9080868 -2.557898 0.8379683 -0.1922875
        ilr.8
1 -0.08609981
2  0.66448918
3  0.30447884

Clustering vector:
 [1] 3 2 2 3 2 2 2 2 2 1 3 2 1 2 2 2 1 3 1 2 2 2 2 2 3

Within cluster sum of squares by cluster:
[1]  4.222482 15.359240  7.243050
 (between_SS / total_SS =  57.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Figura 127: Sortida textual Cluster K-means.

Regressió X real Y compositional

Aquesta funcionalitat realitza la rutina Regressió X real Y compositional, en la Figura 128 podem observar les dades inicials que s'utilitzaran per realitzar la rutina, recordem que aquesta rutina ens retorna quatre elements. El primer element són noves variables en el data frame actual, el segon element són gràfics, el tercer és un nou data frame i per últim una sortida textual.

	num_sedim	sand	silt	clay	depth	ilr.1	ilr.2
1	S01	77.50	19.50	3.00	10.40	2.09	0.98
2	S02	71.90	24.90	3.20	11.70	2.11	0.75
3	S03	50.70	36.10	13.20	12.80	0.96	0.24
4	S04	52.36	41.02	6.62	13.00	1.59	0.17
5	S05	70.00	26.50	3.50	15.70	2.05	0.69
6	S06	66.50	32.20	1.30	16.30	2.92	0.51
7	S07	43.10	55.30	1.60	18.00	2.79	-0.18
8	S08	53.40	36.80	9.80	18.70	1.23	0.26
9	S09	15.50	54.40	30.10	20.70	-0.03	-0.89
10	S10	31.70	41.50	26.80	22.10	0.25	-0.19
11	S11	65.70	27.80	6.50	22.40	1.54	0.61
12	S12	70.40	29.00	0.60	24.40	3.53	0.63
13	S13	17.40	53.60	29.00	25.80	0.04	-0.80
14	S14	10.60	69.80	19.60	32.50	0.27	-1.33
15	S15	38.20	43.10	18.70	33.60	0.63	-0.09
16	S16	10.80	52.70	36.50	36.80	-0.35	-1.12
17	S17	18.40	50.70	30.90	37.80	-0.01	-0.72
18	S18	4.60	47.40	48.00	36.90	-0.96	-1.65
19	S19	15.60	50.40	34.00	42.20	-0.16	-0.83
20	S20	31.90	45.10	23.00	47.00	0.41	-0.24
21	S21	9.50	53.50	37.00	47.10	-0.40	-1.22
22	S22	17.10	48.00	34.90	48.40	-0.16	-0.73
23	S23	10.50	55.40	34.10	49.40	-0.28	-1.18
24	S24	4.78	54.43	40.80	49.50	-0.76	-1.72
25	S25	2.60	45.20	52.20	59.20	-1.28	-2.02

Figura 128: Dades inicials regressió X real Y compositional.

En la Figura 129 podem veure les opcions que farem servir per mostrar l'exemple de funcionament de la rutina Regressió X real Y compositional. En aquest cas utilitzarem la variable "depth" per la selecció X i les variables "sand", "silt" i "clay" per la selecció Y, després com a partició Y seleccionem la creació automàtica i finalment també seleccionem les dues opcions disponibles de "Residuals" i "Fitted".

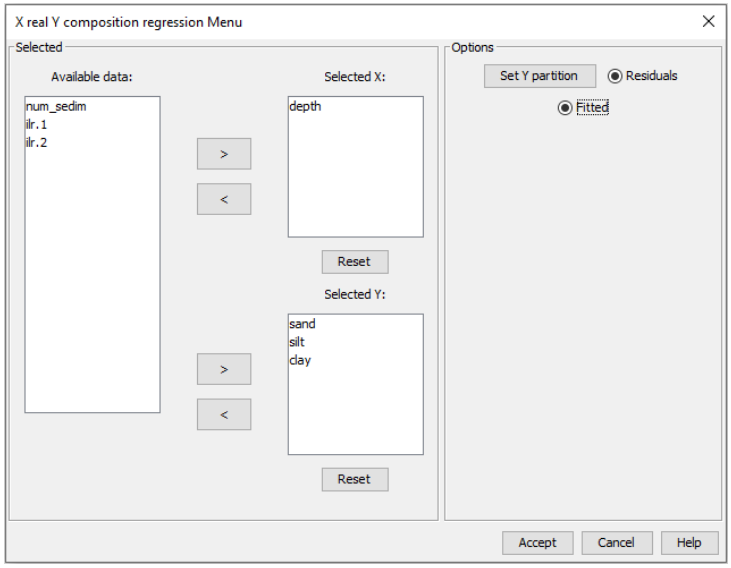


Figura 129: Opcions triades regressió X real Y compositional.

En la Figura 130 i la Figura 131 podem veure el primer dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge els gràfics de la regressió X real Y compositional, que ens indica ilr.1∼depth i ilr.2∼depth respectivament en les següents figures.

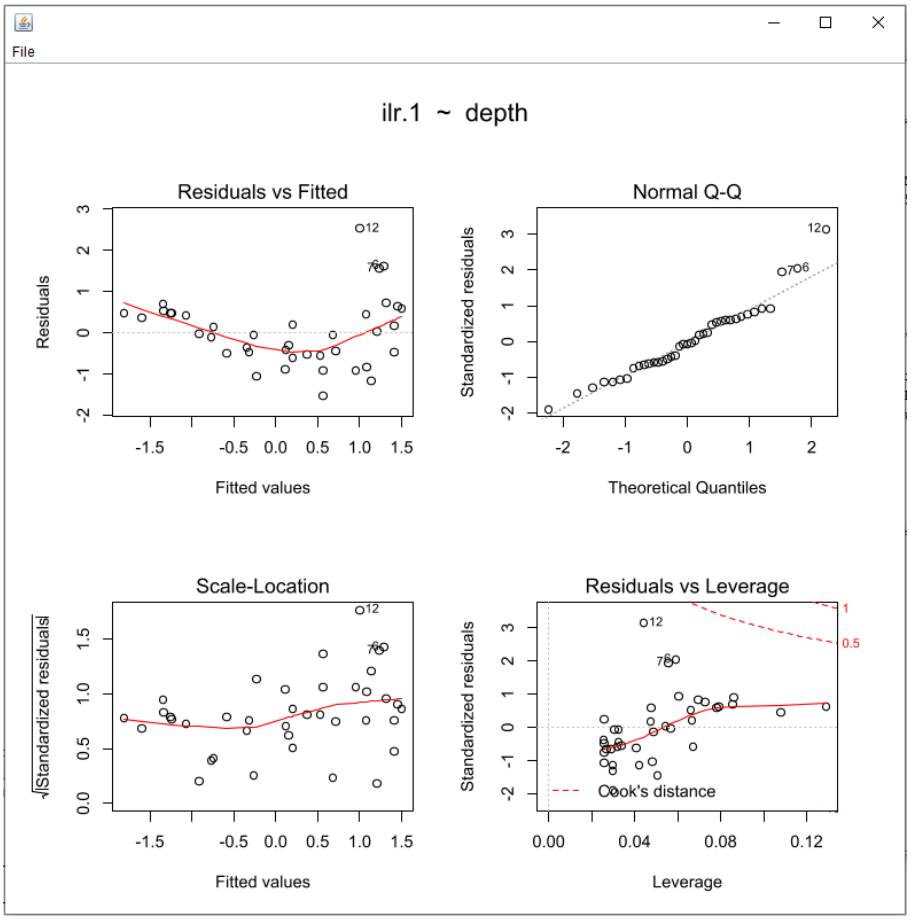


Figura 130: Gràfic nº1 regressió X real Y compositional.

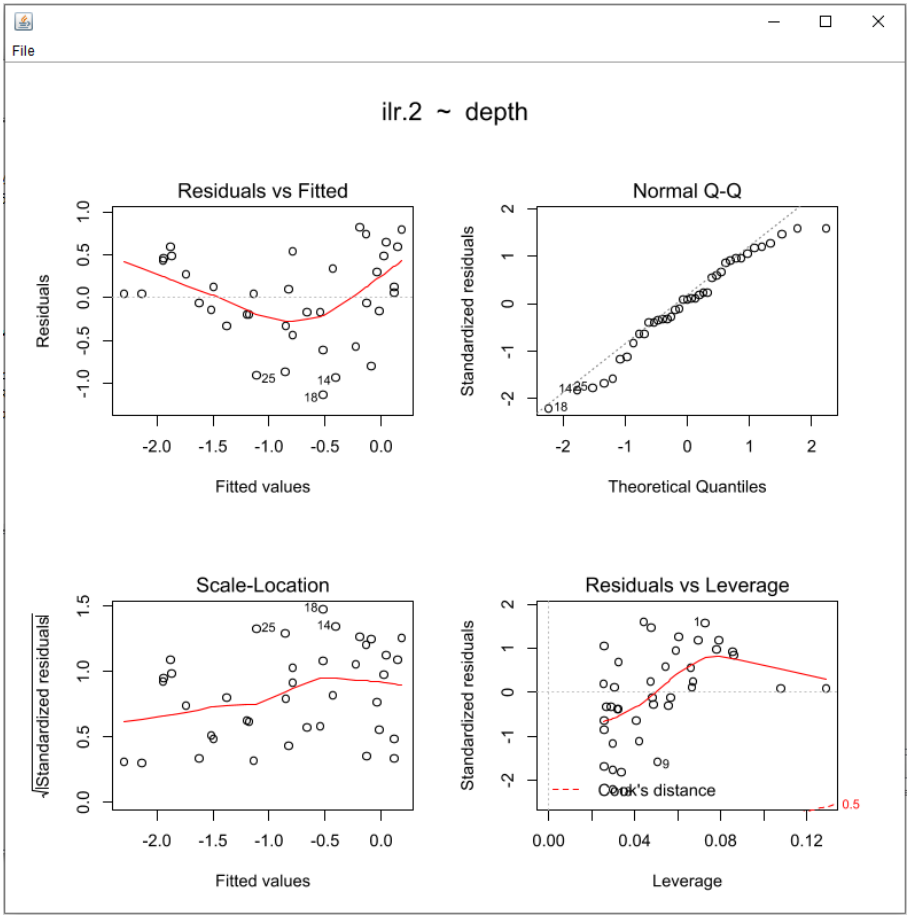


Figura 131: Gràfic nº2 regressió X real Y compositional.

En la Figura 132 podem veure el segon dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge, el nou data frame que conté tres variables, una categòrica i les altres numèriques, també podem veure que hi ha dues observacions una que representa l'intercept i l'altre que representa el depth.

	Coefficients	ilr.1	ilr.2
1	intercept	2.76	1.04
2	depth	-0.07	-0.05

Figura 132: Nou data frame regressió X real Y compositional.

En la Figura 133 podem veure el tercer dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge que han aparegut noves variables al final del data frame, aquesta són les que apareixen encerclades de color vermell.

	num_sedim	sand	silt	clay	depth	ilr.1	ilr.2	ilr.1.r	ilr.2.r	ilr.1.f	ilr.2.f
1	S01	77.50	19.50	3.00	10.40	2.09	0.98	0.04	0.43	2.05	0.54
2	S02	71.90	24.90	3.20	11.70	2.11	0.75	0.14	0.27	1.97	0.48
3	S03	50.70	36.10	13.20	12.80	0.96	0.24	-0.93	-0.19	1.89	0.43
4	S04	52.36	41.02	6.62	13.00	1.59	0.17	-0.29	-0.25	1.88	0.42
5	S05	70.00	26.50	3.50	15.70	2.05	0.69	0.35	0.40	1.69	0.29
6	S06	66.50	32.20	1.30	16.30	2.92	0.51	1.26	0.25	1.65	0.26
7	S07	43.10	55.30	1.60	18.00	2.79	-0.18	1.25	-0.36	1.54	0.18
8	S08	53.40	36.80	9.80	18.70	1.23	0.26	-0.26	0.12	1.49	0.15
9	S09	15.50	54.40	30.10	20.70	-0.03	-0.89	-1.39	-0.94	1.36	0.05
10	S10	31.70	41.50	26.80	22.10	0.25	-0.19	-1.01	-0.17	1.26	-0.02
11	S11	65.70	27.80	6.50	22.40	1.54	0.61	0.30	0.64	1.24	-0.03
12	S12	70.40	29.00	0.60	24.40	3.53	0.63	2.42	0.76	1.11	-0.13
13	S13	17.40	53.60	29.00	25.80	0.04	-0.80	-0.97	-0.60	1.01	-0.20
14	S14	10.60	69.80	19.60	32.50	0.27	-1.33	-0.29	-0.81	0.56	-0.52
15	S15	38.20	43.10	18.70	33.60	0.63	-0.09	0.15	0.49	0.48	-0.57
16	S16	10.80	52.70	36.50	36.80	-0.35	-1.12	-0.61	-0.40	0.27	-0.72
17	S17	18.40	50.70	30.90	37.80	-0.01	-0.72	-0.21	0.06	0.20	-0.77
18	S18	4.60	47.40	48.00	36.90	-0.96	-1.65	-1.22	-0.92	0.26	-0.73
19	S19	15.60	50.40	34.00	42.20	-0.16	-0.83	-0.06	0.15	-0.10	-0.98
20	S20	31.90	45.10	23.00	47.00	0.41	-0.24	0.83	0.97	-0.42	-1.21
21	S21	9.50	53.50	37.00	47.10	-0.40	-1.22	0.03	-0.00	-0.43	-1.22
22	S22	17.10	48.00	34.90	48.40	-0.16	-0.73	0.36	0.55	-0.52	-1.28
23	S23	10.50	55.40	34.10	49.40	-0.28	-1.18	0.30	0.15	-0.59	-1.33
24	S24	4.78	54.43	40.80	49.50	-0.76	-1.72	-0.16	-0.39	-0.59	-1.33
25	S25	2.60	45.20	52.20	59.20	-1.28	-2.02	-0.03	-0.22	-1.25	-1.80

Figura 133: Noves variables Regressió X real Y compositional.

Per últim en la Figura 134 observem el segon element que ens retorna la rutina i que correspon a la sortida textual. La sortida ens indica, entre altres coses, alguns paràmetres tant de la response ilr.1 com de la response ilr.2. També al final de tot ens indica el valor de r^2.

```
X real Y composition regression:

LINEAR REGRESSION

Response ilr.1 :

Call:
lm(formula = ilr.1 ~ as.matrix(X))

Residuals:
     Min       1Q   Median       3Q      Max
-1.38562 -0.28996 -0.03353  0.30368  2.42271

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.75750    0.40346   6.835 5.71e-07 ***
as.matrix(X) -0.06769    0.01212  -5.587 1.10e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8632 on 23 degrees of freedom
Multiple R-squared:  0.5757,    Adjusted R-squared:  0.5573
F-statistic: 31.21 on 1 and 23 DF,  p-value: 1.102e-05


Response ilr.2 :

Call:
lm(formula = ilr.2 ~ as.matrix(X))

Residuals:
     Min       1Q   Median       3Q      Max
-0.93682 -0.35501  0.05587  0.39758  0.96976

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.043614   0.245805   4.246 0.000305 ***
as.matrix(X) -0.048047   0.007382  -6.509 1.22e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5259 on 23 degrees of freedom
Multiple R-squared:  0.6481,    Adjusted R-squared:  0.6328
F-statistic: 42.36 on 1 and 23 DF,  p-value: 1.216e-06



r^2 =  [1] 59.81165
```

Figura 134: Sortida textual regressió X real Y compositional.

Zpatterns

Aquesta funcionalitat realitza la rutina Zpatterns, en la Figura 135 podem observar les dades inicials que s'utilitzaran per realitzar la rutina, recordem que aquesta rutina ens retorna dos elements. El primer element és un gràfic en una nova finestra independent, i el segon element és una sortida textual.

	codi	zeros	neogl_atl	neogl_pach	glob_obesa	glob_triloba
1	1	no zero	0.74	0.19	0.03	0.04
2	2	no zero	0.58	0.29	0.01	0.12
3	3	no zero	0.58	0.19	0.22	0.01
4	4	no zero	0.61	0.28	0.08	0.03
5	5	no zero	0.82	0.13	0.02	0.03
6	6	no zero	0.48	0.38	0.01	0.13
7	7	zero	0.59	0.38	0	0.03
8	8	no zero	0.76	0.12	0.09	0.03
9	9	no zero	0.81	0.12	0.04	0.03
10	10	no zero	0.68	0.23	0.05	0.04
11	11	no zero	0.72	0.20	0.04	0.04
12	12	no zero	0.62	0.27	0.09	0.02
13	13	no zero	0.45	0.25	0.29	0.01
14	14	no zero	0.66	0.25	0.06	0.03
15	15	no zero	0.85	0.13	0.01	0.01
16	16	no zero	0.75	0.09	0.15	0.01
17	17	zero	0.69	0.25	0	0.06
18	18	no zero	0.76	0.10	0.11	0.03
19	19	no zero	0.66	0.29	0.01	0.04
20	20	no zero	0.66	0.24	0.06	0.04
21	21	zero	0.50	0.46	0	0.40
22	22	no zero	0.65	0.25	0.05	0.05
23	23	no zero	0.60	0.35	0.02	0.03
24	24	no zero	0.40	0.27	0.01	0.32
25	25	zero	0.60	0.10	0.30	0
26	26	no zero	0.60	0.10	0.29	0.01
27	27	no zero	0.59	0.39	0.01	0.01
28	28	no zero	0.58	0.39	0.01	0.02
29	29	no zero	0.61	0.34	0.02	0.03
30	30	zero	0.39	0.49	0.12	0

Figura 135: Dades inicials Zpatterns.

En la Figura 136 trobem les opcions triades per l'usuari en aquest cas que com bé podem observar ha seleccionat les opcions de mostrar tant les mitjanes com els percentatges i el label que s'ha triat és el label 0.

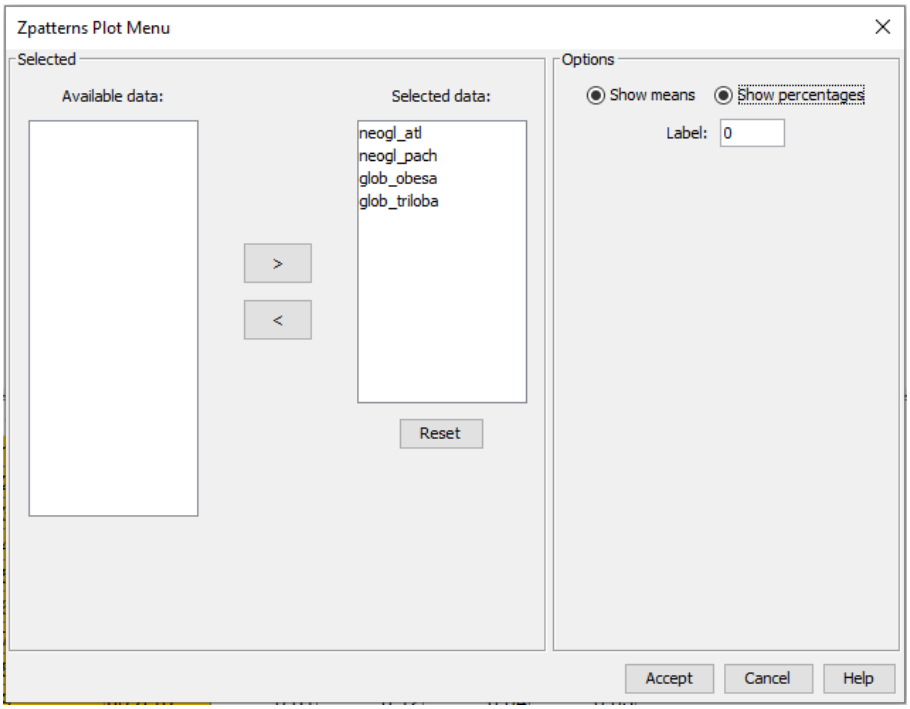


Figura 136: Opcions triades Zpatterns.

En la Figura 137 podem veure el primer dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge el gràfic Zpatterns, que ens indica el patró de zeros de les dades, indicant també les mitjanes i els percentatges. Com podem veure també en la imatge, el label que s'ha utilitzat en aquest cas és 0.

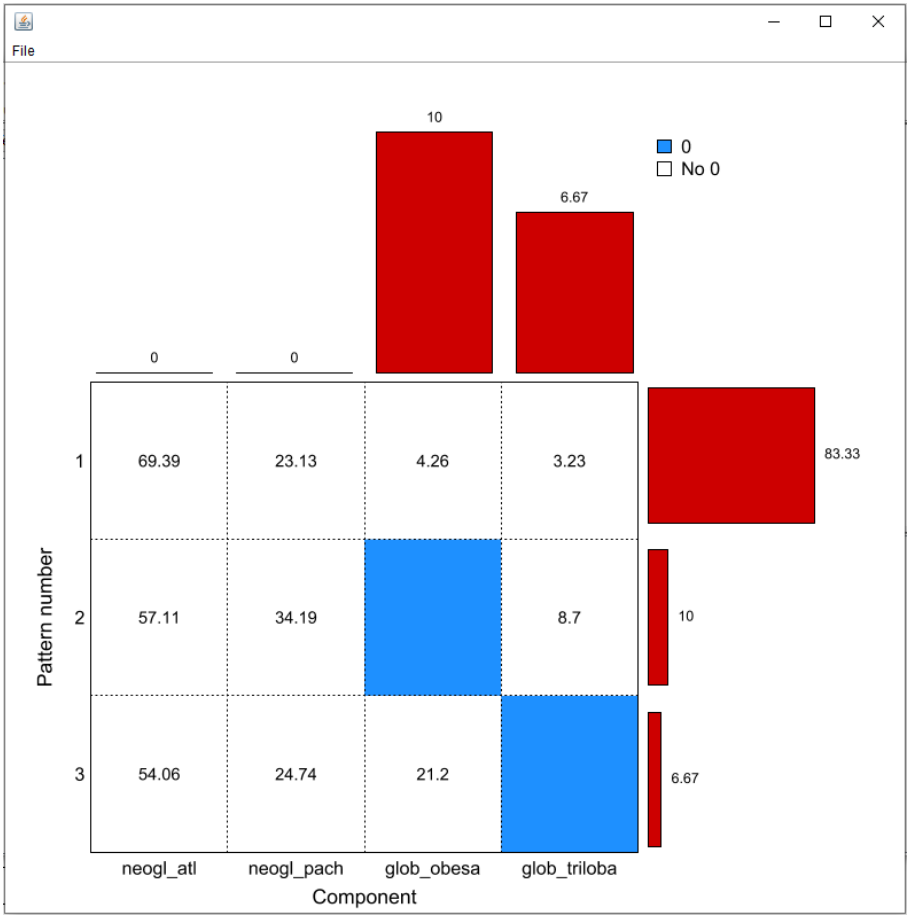


Figura 137: Gràfics Zpatterns.

Per últim en la Figura 138 observem el segon element que ens retorna la rutina i que correspon a la sortida textual. La sortida ens indica els patterns de zeros els percentatges per component i els percentatges per cel·la.

```
Zpatterns Plot:

ZERO PATTERNS

Patterns ('+' means 0, '-' means No 0)

 Patt.ID neogl_atl neogl_pach glob_obesa glob_triloba No.Unobs Patt.Freq
       1         -          -          -            -        0        25
       2         -          -          +            -        1         3
       3         -          -          -            +        1         2
 Percent
   83.33
   10.00
    6.67

Percentage cells by component
   neogl_atl   neogl_pach   glob_obesa glob_triloba
        0.00         0.00        10.00         6.67

Overall percentage cells: 4.17%
```

Figura 138: Sortida textual Zpatterns.

Bayesian Multiplicative Replacement

Aquesta funcionalitat realitza la rutina Bayesian Multiplicative Replacement, en la Figura 139 podem observar les dades inicials que s'utilitzaran per realitzar la rutina, recordem que aquesta rutina ens retorna només un element. Aquest element és un conjunt de noves variables que seran afegides al final del data frame actual.

	codi	zeros	neogl_atl	neogl_pach	glob_obesa	glob_triloba
1	1	no zero	0.74	0.19	0.03	0.04
2	2	no zero	0.58	0.29	0.01	0.12
3	3	no zero	0.58	0.19	0.22	0.01
4	4	no zero	0.61	0.28	0.08	0.03
5	5	no zero	0.82	0.13	0.02	0.03
6	6	no zero	0.48	0.38	0.01	0.13
7	7	zero	0.59	0.38	0	0.03
8	8	no zero	0.76	0.12	0.09	0.03
9	9	no zero	0.81	0.12	0.04	0.03
10	10	no zero	0.68	0.23	0.05	0.04
11	11	no zero	0.72	0.20	0.04	0.04
12	12	no zero	0.62	0.27	0.09	0.02
13	13	no zero	0.45	0.25	0.29	0.01
14	14	no zero	0.66	0.25	0.06	0.03
15	15	no zero	0.85	0.13	0.01	0.01
16	16	no zero	0.75	0.09	0.15	0.01
17	17	zero	0.69	0.25	0	0.06
18	18	no zero	0.76	0.10	0.11	0.03
19	19	no zero	0.66	0.29	0.01	0.04
20	20	no zero	0.66	0.24	0.06	0.04
21	21	zero	0.50	0.46	0	0.40
22	22	no zero	0.65	0.25	0.05	0.05
23	23	no zero	0.60	0.35	0.02	0.03
24	24	no zero	0.40	0.27	0.01	0.32
25	25	zero	0.60	0.10	0.30	0
26	26	no zero	0.60	0.10	0.29	0.01
27	27	no zero	0.59	0.39	0.01	0.01
28	28	no zero	0.58	0.39	0.01	0.02
29	29	no zero	0.61	0.34	0.02	0.03
30	30	zero	0.39	0.49	0.12	0

Figura 139: Dades inicials Bayesian Multiplicative Replacement.

En la Figura 140 trobem les opcions triades per l'usuari en aquest cas que com bé podem observar ha seleccionat com a method l'opció "GBM" i després com a Output s'ha seleccionat l'opció "prop".

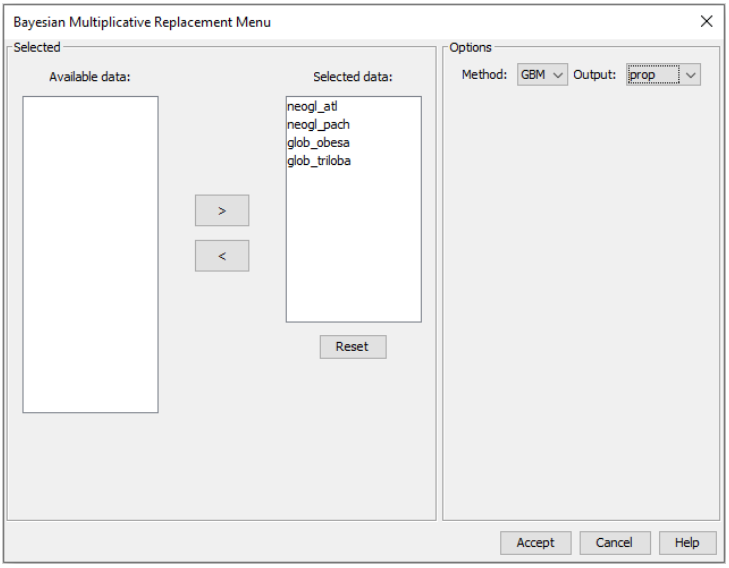


Figura 140: Opcions triades Bayesian Multiplicative Replacement.

En la Figura 141 podem veure l'únic dels elements que ens retorna la rutina un cop realitzada. Observem en la imatge les noves variables afegides al final del data frame, més concretament són les ultimes quatre variables marcades de color vermell.

	codi	zeros	neogl_atl	neogl_pach	glob_obesa	glob_triloba	neogl_atl.imp	neogl_pac...	glob_obes...	glob_trilob...
1	1	no zero	0.74	0.19	0.03	0.04	0.74	0.19	0.03	0.04
2	2	no zero	0.58	0.29	0.01	0.12	0.58	0.29	0.01	0.12
3	3	no zero	0.58	0.19	0.22	0.01	0.58	0.19	0.22	0.01
4	4	no zero	0.61	0.28	0.08	0.03	0.61	0.28	0.08	0.03
5	5	no zero	0.82	0.13	0.02	0.03	0.82	0.13	0.02	0.03
6	6	no zero	0.48	0.38	0.01	0.13	0.48	0.38	0.01	0.13
7	7	zero	0.59	0.38	0	0.03	0.59	0.38	0.01	0.03
8	8	no zero	0.76	0.12	0.09	0.03	0.76	0.12	0.09	0.03
9	9	no zero	0.81	0.12	0.04	0.03	0.81	0.12	0.04	0.03
10	10	no zero	0.68	0.23	0.05	0.04	0.68	0.23	0.05	0.04
11	11	no zero	0.72	0.20	0.04	0.04	0.72	0.20	0.04	0.04
12	12	no zero	0.62	0.27	0.09	0.02	0.62	0.27	0.09	0.02
13	13	no zero	0.45	0.25	0.29	0.01	0.45	0.25	0.29	0.01
14	14	no zero	0.66	0.25	0.06	0.03	0.66	0.25	0.06	0.03
15	15	no zero	0.85	0.13	0.01	0.01	0.85	0.13	0.01	0.01
16	16	no zero	0.75	0.09	0.15	0.01	0.75	0.09	0.15	0.01
17	17	zero	0.69	0.25	0	0.06	0.69	0.25	0.01	0.06
18	18	no zero	0.76	0.10	0.11	0.03	0.76	0.10	0.11	0.03
19	19	no zero	0.66	0.29	0.01	0.04	0.66	0.29	0.01	0.04
20	20	no zero	0.66	0.24	0.06	0.04	0.66	0.24	0.06	0.04
21	21	zero	0.50	0.46	0	0.40	0.37	0.34	0.01	0.29
22	22	no zero	0.65	0.25	0.05	0.05	0.65	0.25	0.05	0.05
23	23	no zero	0.60	0.35	0.02	0.03	0.60	0.35	0.02	0.03
24	24	no zero	0.40	0.27	0.01	0.32	0.40	0.27	0.01	0.32
25	25	zero	0.60	0.10	0.30	0	0.60	0.10	0.30	0.01
26	26	no zero	0.60	0.10	0.29	0.01	0.60	0.10	0.29	0.01
27	27	no zero	0.59	0.39	0.01	0.01	0.59	0.39	0.01	0.01
28	28	no zero	0.58	0.39	0.01	0.02	0.58	0.39	0.01	0.02
29	29	no zero	0.61	0.34	0.02	0.03	0.61	0.34	0.02	0.03
30	30	zero	0.39	0.49	0.12	0	0.39	0.49	0.12	0.01

Figura 141: Noves variables Bayesian Multiplicative Replacement.

10.2.4 Menús de desenvolupador

A continuació en aquest apartat es mostraran els diferents menús de desenvolupador que s'han creat amb una petita descripció de cada un d'ells i un exemple d'execució amb un d'ells, més concretament amb el menú S3.

S0

El menú de desenvolupador S0, esta dissenyat per variables que siguin numèriques per tant la variable "X" serà numèrica. A part d'això conté una base per la "X" i unes opcions tant de text com lògiques. Podem veure el seu aspecte en la Figura 142.

Els elements que pot retornar són: text, nous data frames, noves variables en el data frame actual i gràfiques.

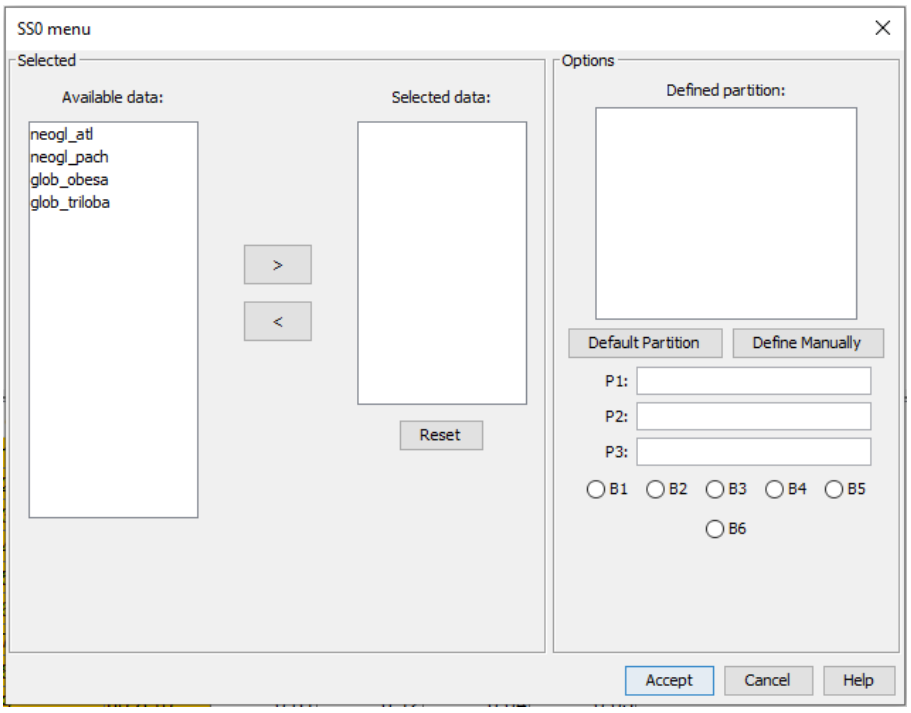


Figura 142: Menú de desenvolupament S0.

S1

El menú de desenvolupador S1, esta dissenyat per variables que siguin numèriques i positives per tant la variable "X" serà numèrica i estrictament amb variables positives. A part d'això conté una base per la "X" i unes opcions tant de text com lògiques. Podem veure el seu aspecte en la Figura 143.

Els elements que pot retornar són: text, nous data frames, noves variables en el data frame actual i gràfiques.

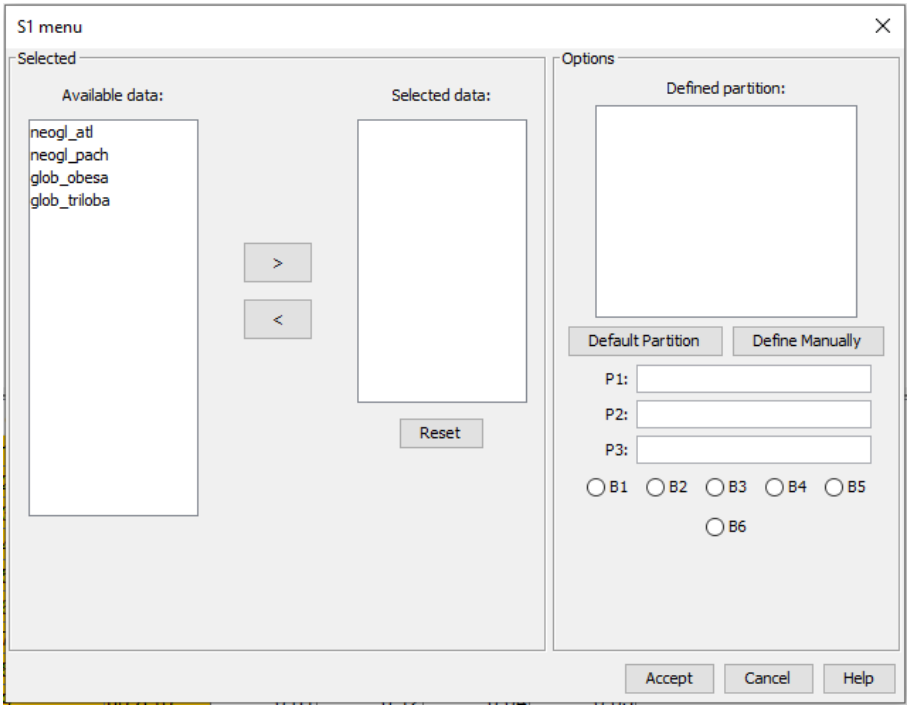


Figura 143: Menú de desenvolupament S1.

S2

El menú de desenvolupador S2, esta dissenyat per variables on la "X" sigui numèrica i on la variable "Y" sigui categòrica. A part d'això conté una base per la "X", un base també per la "Y" i unes opcions tant de text com lògiques. Podem veure el seu aspecte en la Figura 144.

Els elements que pot retornar són: text, nous data frames, noves variables en el data frame actual i gràfiques.

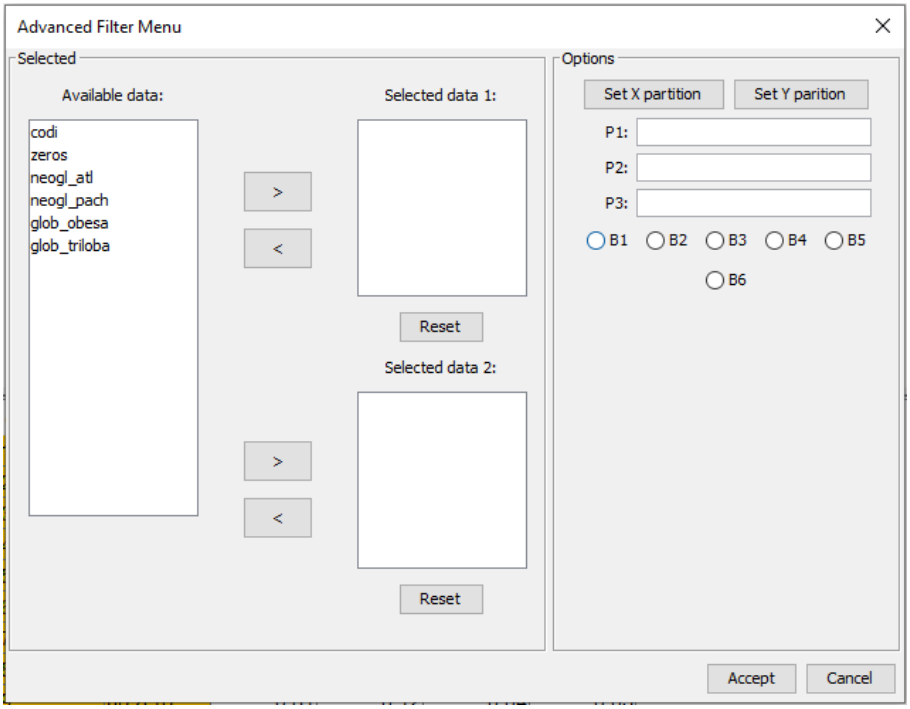


Figura 144: Menú de desenvolupament S2.

S3

En aquest menú farem una petita demo d'una execució amb un script creat amb R i amb el menú de desenvolupador S3. En la Figura 145 podem veure les dades inicials que utilitzarem per realitzar la demo.

	pinus	abies	quercus	group	ilr.1	ilr.2
1	0.30	0.58	0.12	1.0	1.02	-0.48
2	0.18	0.48	0.33	1.0	-0.08	-0.68
3	0.48	0.44	0.09	1.0	1.36	0.07
4	0.39	0.54	0.07	1.0	1.60	-0.23
5	0.20	0.51	0.29	1.0	0.07	-0.65
6	0.31	0.53	0.16	1.0	0.73	-0.38
7	0.24	0.60	0.16	1.0	0.72	-0.65
8	0.20	0.59	0.21	1.0	0.41	-0.75
9	0.18	0.58	0.23	1.0	0.28	-0.81
10	0.17	0.52	0.31	1.0	-0.03	-0.79
11	0.24	0.36	0.40	2.0	-0.24	-0.29
12	0.36	0.45	0.20	2.0	0.59	-0.16
13	0.18	0.44	0.37	2.0	-0.22	-0.63
14	0.35	0.50	0.15	2.0	0.84	-0.26
15	0.36	0.46	0.18	2.0	0.68	-0.16
16	0.22	0.48	0.30	2.0	0.07	-0.56
17	0.22	0.45	0.33	2.0	-0.04	-0.50
18	0.34	0.47	0.19	2.0	0.62	-0.24
19	0.25	0.45	0.31	2.0	0.06	-0.42
20	0.17	0.43	0.40	2.0	-0.31	-0.66
21	0.42	0.44	0.14	3.0	0.89	-0.04
22	0.34	0.48	0.18	3.0	0.66	-0.24
23	0.44	0.42	0.14	3.0	0.89	0.04
24	0.61	0.37	0.03	3.0	2.33	0.36
25	0.67	0.30	0.03	3.0	2.10	0.57
26	0.55	0.36	0.09	3.0	1.27	0.29
27	0.49	0.41	0.11	3.0	1.16	0.12
28	0.47	0.45	0.08	3.0	1.42	0.03
29	0.71	0.25	0.04	3.0	1.93	0.73
30	0.49	0.40	0.11	3.0	1.13	0.16

Figura 145: Dades inicials demo menú S3.

El menú de desenvolupador S3, esta dissenyat per variables on la "X" sigui numèrica i on la variable "Y" pugui ser numèrica o categòrica. A part d'això conté una base per la "X", un base també per la "Y" i unes opcions tant de text com lògiques. Podem veure el seu aspecte i les opcions triades per realitzar la demo en la Figura 146.

Els elements que pot retornar són: text, nous data frames, noves variables en el data frame actual i gràfiques. En aquesta demo en concret simplement retornarà noves variables i una sortida textual en la consola que mostrarem a continuació.

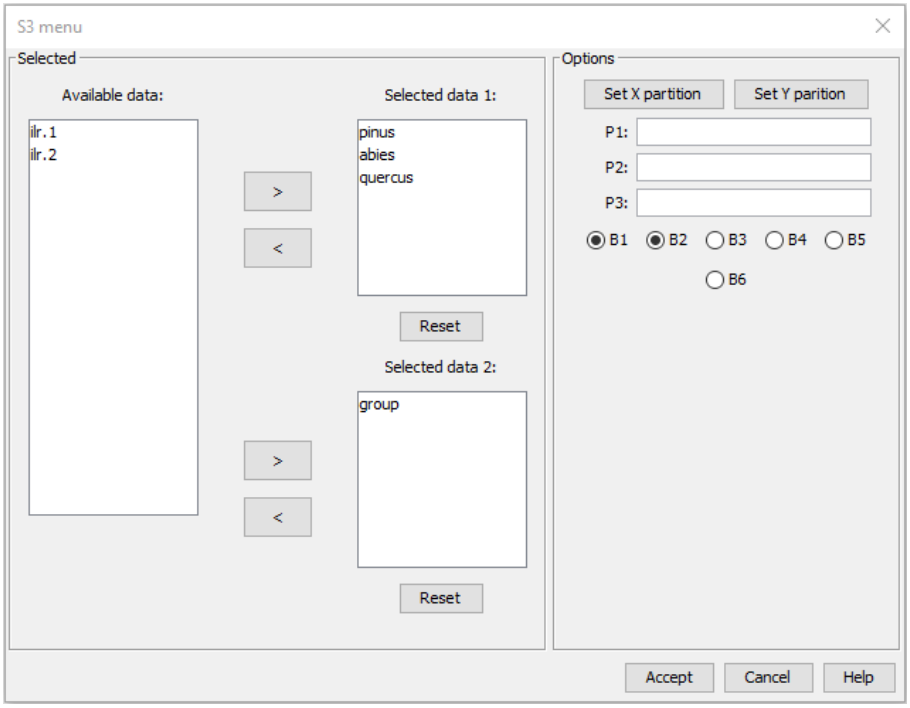


Figura 146: Menú S3 amb les opcions seleccionades.

Una de les coses que es interessant dels menús de desenvolupador es que un cop seleccionat el menú que es vol juntament amb les dades i opcions, un cop es fa clic en acceptar es permet poder seleccionar l'arxiu R que vols carregar per realitzar l'execució com podem veure en la Figura 147.

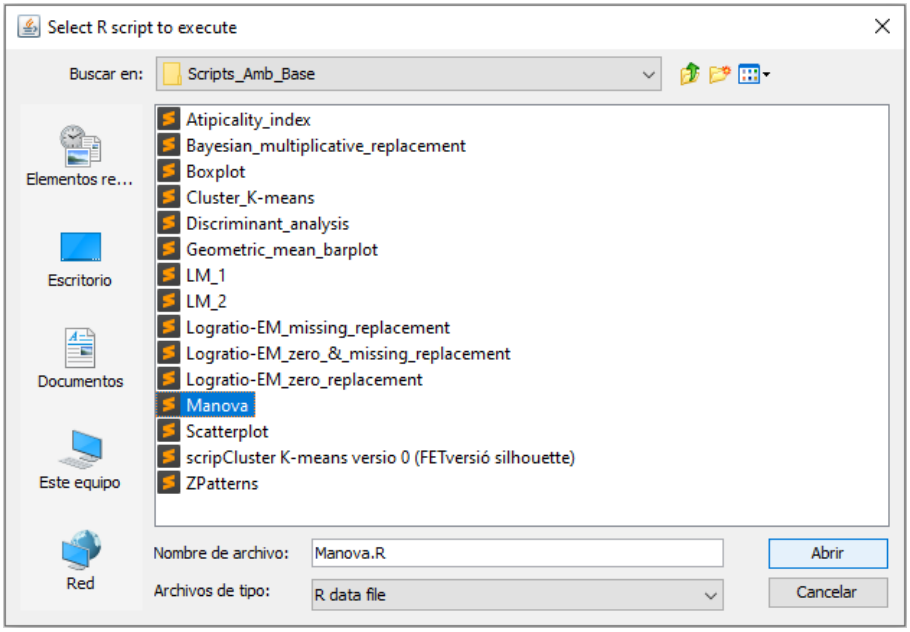


Figura 147: Selecció de l'script a executar amb el menú S3.

En la Figura 148 podem veure el primer dels elements que es retornen en aquesta demo que són les

noves variables en el data frame actual. En aquest cas són dues noves variables al final del data frame que apareixen encerclades de color vermell en la imatge.

	pinus	abies	quercus	group	ilr.1	ilr.2	ilr.1.r	ilr.2.r
1	0.30	0.58	0.12	1.0	1.02	-0.48	0.06	0.41
2	0.18	0.48	0.33	1.0	-0.08	-0.68	-0.15	-0.69
3	0.48	0.44	0.09	1.0	1.36	0.07	0.60	0.76
4	0.39	0.54	0.07	1.0	1.60	-0.23	0.31	0.99
5	0.20	0.51	0.29	1.0	0.07	-0.65	-0.12	-0.54
6	0.31	0.53	0.16	1.0	0.73	-0.38	0.16	0.13
7	0.24	0.60	0.16	1.0	0.72	-0.65	-0.12	0.11
8	0.20	0.59	0.21	1.0	0.41	-0.75	-0.21	-0.20
9	0.18	0.58	0.23	1.0	0.28	-0.81	-0.27	-0.33
10	0.17	0.52	0.31	1.0	-0.03	-0.79	-0.26	-0.64
11	0.24	0.36	0.40	2.0	-0.24	-0.29	0.10	-0.45
12	0.36	0.45	0.20	2.0	0.59	-0.16	0.23	0.38
13	0.18	0.44	0.37	2.0	-0.22	-0.63	-0.24	-0.43
14	0.35	0.50	0.15	2.0	0.84	-0.26	0.13	0.64
15	0.36	0.46	0.18	2.0	0.68	-0.16	0.23	0.48
16	0.22	0.48	0.30	2.0	0.07	-0.56	-0.17	-0.13
17	0.22	0.45	0.33	2.0	-0.04	-0.50	-0.11	-0.25
18	0.34	0.47	0.19	2.0	0.62	-0.24	0.15	0.42
19	0.25	0.45	0.31	2.0	0.06	-0.42	-0.03	-0.15
20	0.17	0.43	0.40	2.0	-0.31	-0.66	-0.27	-0.51
21	0.42	0.44	0.14	3.0	0.89	-0.04	-0.24	-0.49
22	0.34	0.48	0.18	3.0	0.66	-0.24	-0.44	-0.72
23	0.44	0.42	0.14	3.0	0.89	0.04	-0.16	-0.48
24	0.61	0.37	0.03	3.0	2.33	0.36	0.16	0.96
25	0.67	0.30	0.03	3.0	2.10	0.57	0.37	0.72
26	0.55	0.36	0.09	3.0	1.27	0.29	0.09	-0.11
27	0.49	0.41	0.11	3.0	1.16	0.12	-0.08	-0.21
28	0.47	0.45	0.08	3.0	1.42	0.03	-0.17	0.04
29	0.71	0.25	0.04	3.0	1.93	0.73	0.53	0.55
30	0.49	0.40	0.11	3.0	1.13	0.16	-0.04	-0.25

Figura 148: Noves variables S3.

En la Figura 149 podem veure el segon dels elements que es retornen en aquesta demo que és una sortida textual en la consola. En aquest cas hem agafat un fragment de la sortida, ja que era una mica llarga.

```
MANOVA ANALYSIS

          Df   Wilks approx F num Df den Df    Pr(>F)

Y          2 0.20308   15.848      4     52 1.527e-08 ***

Residuals 27

---

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

          Df Pillai approx F num Df den Df    Pr(>F)

Y          2 1.0582   15.168      4     54 2.258e-08 ***

Residuals 27

---

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

          Df Hotelling-Lawley approx F num Df den Df    Pr(>F)

Y          2           2.6378   16.487      4     50 1.123e-08 ***

Residuals 27

---

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

          Df    Roy approx F num Df den Df    Pr(>F)

Y          2 1.9921   26.893      2     27 3.753e-07 ***

Residuals 27

---
```

Figura 149: Sortida textual S3.

S4

El menú de desenvolupador S4, esta dissenyat per variables on la "X" pugui ser numèrica o categòrica i on la variable "Y" també pugui ser numèrica o categòrica. A part d'això conté una base per la "X", un base també per la "Y" i unes opcions tant de text com lògiques. Podem veure el seu aspecte en la Figura 150.

Els elements que pot retornar són: text, nous data frames, noves variables en el data frame actual i gràfiques.

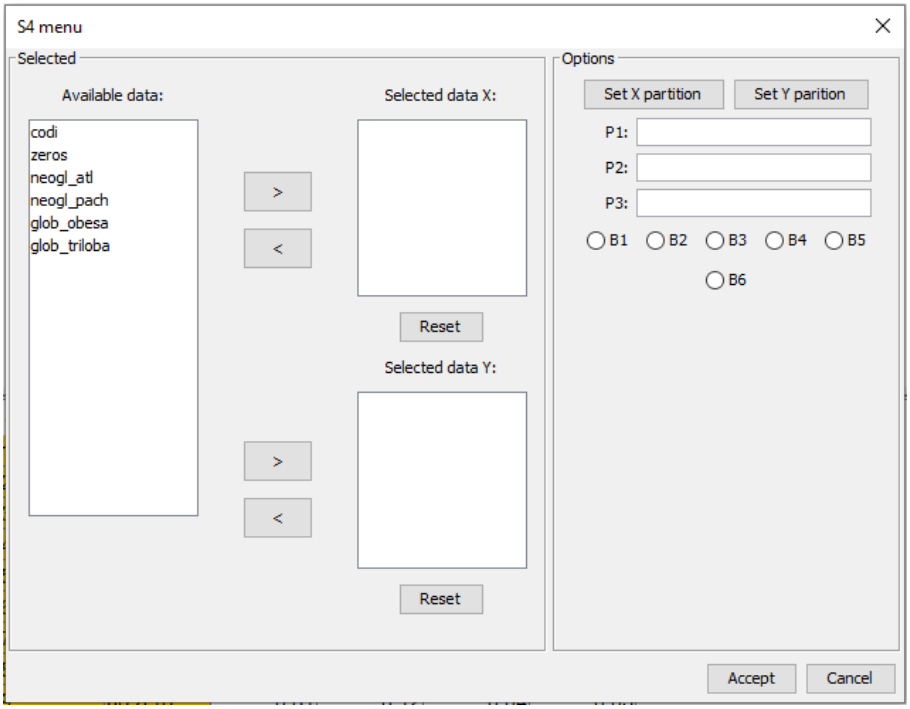


Figura 150: Menú de desenvolupament S4.

11 Conclusions

Els objectius del projecte inicialment consistien a analitzar, dissenyar i implementar una llibreria amb Java que permeti utilitzar llibreries de CODA desenvolupades amb R. Per tant també crear una interfície genèrica per tal que usuaris poguessin crear les seves pròpies rutines escrites amb R i executar-les en el CoDaPack.

Com que els objectius i funcionalitats definides al principi del projecte estan presents en l'aplicació desenvolupada fins ara, es considera que els objectius s'han assolit correctament. A més a més s'han desenvolupat diferents tipus de menús de desenvolupador, segons l'usuari vulgui utilitzar alguns paràmetres per la seva rutina o uns altres.

L'aplicació ha evolucionat molt favorablement i l'objectiu d'aconseguir ajuntar R amb Java s'ha assolit. Aquest projecte podríem dir que té moltes possibilitats i un futur al davant amb possibles noves millores i/o rutines noves.

L'aplicació actualment està disponible i operativa. Aquesta nova versió del CoDaPack ha permès ampliar considerablement les coses que es poden fer amb el CoDaPack gràcies al fet que ara té connexió amb R i per tant moltes coses que es poden fer amb R també ara es poden fer amb el CoDaPack. Aquestes millores i ampliacions els usuaris i la comunitat les agrairan bastant.

A continuació comentaré una sèrie de coneixements i experiència que he assolit al llarg del desenvolupament del projecte:

- Coneixements complementaris amb programació de Java, ja que ara sé treballar molt millor amb dependències i llibreries. També comentar que he après a incorporar llenguatges dins de Java, ja sigui HTML, R, JavaScript, CSS o Yaml.
- He après també a crear instal·ladors per als diferents sistemes operatius amb la creació de variables d'entorn sigui Windows o MacOS.
- Manipular llibreries d'R i saber-les utilitzar correctament en l'àmbit de Java i rJava.
- Treballar en un equip de recerca en un àmbit professional.

La meva conclusió personal final ha estat bona, ja que he après moltes coses i també a saber tractar certs problemes des de diferents punts de vista, també hem sento molt satisfet d'haver desenvolupat correctament el projecte i haver aconseguit els objectius marcats al principi del projecte.

Finalment vull comentar que des d'un principi he rebut una molt bona acollida en el departament i per qualsevol cosa sempre he pogut comptar amb qualsevol persona. Amb qui més contacte he tingut ha sigut amb en Marc Comas i en Santi Thió que han estat en tot moment al meu costat per ajudar-me per resoldre dubtes o simplement anar al meu costat durant el desenvolupament del projecte. Per tot això vull expressar el meu agraïment a tot el departament que m'han fet sentir un més del departament, en especial a en Santi Thió i en Marc Comas.

12 Treball futur

En aquest apartat, es comentaran les possibles ampliacions, millores o treballs futurs que es poden realitzar del projecte que s'ha desenvolupat en aquest TFG.

Aquest apartat s'ha decidit dividir-lo simplement en una part que consisteix en noves funcionalitats que es podrien afegir al projecte implementat.

12.1 Noves funcionalitats

- Ampliar les funcionalitats d'alguns dels menús ja existents.
- Fer que els enllaços que apareixen en els menús de help siguin clicables i per tant et porti a la pàgina web a la qual fa referència.
- Poder exportar els gràfics generats amb R en diferents formats com: PDF, PNG, JPEG...
- Desenvolupar un editor de menús per tal que l'usuari pugui crear els seus propis scripts d'R i pugi per tant ampliar les funcionalitats del seu propi CoDaPack.

En un futur fins i tot es podria arribar a utilitzar l'aplicació del CoDaPack amb les noves millores i funcionalitats per realitzar pràctiques de l'assignatura d'estadística de la carrera d'una manera més senzilla per als alumnes, ja que la interacció es realitza amb menús interactius.

13 Bibliografia

[1] CoDaWeb. *CoDaPack*, http://www.compositionaldata.com/codapack.php

[2] Luís Gonçalves. *Qué es la metodología Ágil*, https://luis-goncalves.com/es/que-es-la-metodologia-agil/

[3] Wikipedia. *Scrum (desarrollo de software)*, https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[4] Grup de Recerca en Estadística i Anàlisi de Dades Composicionals. *Recerca → Línies de recerca → Anàlisi de dades composicionals → CoDaPack*, http://ima.udg.edu/Recerca/EIO/inici_cat.html

[5] Grup de Recerca en Estadística i Anàlisi de Dades Composicionals. *Presentació*, http://ima.udg.edu/Recerca/EIO/inici_cat.html

[6] Wikipedia. *Biblioteca de enlace dinámico*, https://es.wikipedia.org/wiki/Biblioteca_de_enlace_din%C3%A1mico

[7] Apunts projecte de programació. *Introducció al llenguatge Java*, https://moodle2.udg.edu/pluginfile.php/781829/mod_resource/content/6/java.pdf

[8] Rafael Vindel Amor. *Introducción a Colecciones en Java*, https://www.adictosaltrabajo.com/2015/09/25/introduccion-a-colecciones-en-java/

[9] Alejandro Pérez García. *Maven, nunca antes resultó tan fácil compilar, empaquetar, ...*, https://www.adictosaltrabajo.com/2006/09/19/maven/

[10] Juan Bosco Mendoza Vega. *R para principiantes*, https://bookdown.org/jboscomendoza/r-principiantes4/

[11] Sergio G.B.. *Variables de entorno: ¿qué son y para qué sirven?*, https://www.sololinux.es/variables-de-entorno-que-son-y-para-que-sirven/

[12] JYAML, http://jyaml.sourceforge.net/

[13] Batik-Swing https://xmlgraphics.apache.org/batik/using/swing.html

[14] Commons-Math, http://commons.apache.org/proper/commons-math/

[15] RENJIN-SCRIPT-ENGINE, http://docs.renjin.org/en/latest/introduction.html#about-renjin

[16] POI, https://www.tutorialspoint.com/apache_poi/apache_poi_overview.htm

[17] SWING-LAYOUT, https://mvnrepository.com/artifact/net.java.dev.swing-layout/swing-layout/1.0.2

[18] JLATEXMATH, https://github.com/opencollab/jlatexmath

[19] OPENCSV, http://opencsv.sourceforge.net/

[20] JRI,
https://www.rforge.net/JRI/

[21] RENGINE,
https://www.rforge.net/org/docs/org/rosuda/JRI/Rengine.html

[22] JRIENGINE,
https://www.rforge.net/org/docs/org/rosuda/REngine/JRI/JRIEngine.html

[23] CONTROLSFX,
https://github.com/controlsfx/controlsfx

[24] ITEXTPDF,
https://es.wikipedia.org/wiki/IText

[25] JZY3D-API,
http://www.jzy3d.org/

[26] Deepayan Sarkar [aut, cre] (<https://orcid.org/0000-0003-4107-1553>),Felix Andrews [ctb],Kevin Wright [ctb] (documentation),Neil Klepeis [ctb],Paul Murrell [ctb]. *Package 'lattice'*,
https://cran.r-project.org/web/packages/lattice/lattice.pdf

[27] Marc Comas-Cufí. *Package 'coda.base'*,
https://cran.r-project.org/web/packages/coda.base/coda.base.pdf

[28] Javier Palarea-Albaladejo and Josep Antoni Martin-Fernandez. *Package 'zCompositions'*,
https://cran.r-project.org/web/packages/zCompositions/zCompositions.pdf

[29] Hadley Wickham. *Package 'plyr'*,
https://cran.r-project.org/web/packages/plyr/plyr.pdf

[30] Simon Urbanek. *Package 'rJava'*,
https://cran.r-project.org/web/packages/rJava/rJava.pdf

[31] Uwe Ligges, Martin Maechler and Sarah Schnackenberg. *Package 'scatterplot3d'*,
https://cran.r-project.org/web/packages/scatterplot3d/scatterplot3d.pdf

[32] Christian Hennig. *Package 'fpc'*,
https://cran.r-project.org/web/packages/fpc/fpc.pdf

[33] Michael Hahsler [aut, cre, cph],Christian Buchta [aut, cph],Bettina Gruen [aut, cph],Kurt Hornik [aut, cph],Ian Johnson [ctb, cph] and Christian Borgelt [ctb, cph]. *Package 'arules'*,
https://cran.r-project.org/web/packages/arules/arules.pdf

[34] Rengine eval() function description,
http://www.rforge.net/org/doc/org/rosuda/JRI/Rengine.html#eval(java.lang.String)

[35] Kenjiloc. *How to create a dmg disk installer file on mac 2016*
https://www.youtube.com/watch?v=8bkUMKX4cZ0

14 Annexos

L'apartat dels annexos serveix per ampliar alguns dels conceptes utilitzats, mostrar codi, altres resultats, o afegir explicacions complementàries que no són necessàries per a la comprensió global del projecte.

En aquest cas s'utilitzarà el projecte per adjuntar dos enllaços, el primer dels enllaços correspon a la branca de GitHub que s'ha utilitzat per al desenvolupament del projecte i on es troba tot el codi actual de l'última versió del CoDaPack.

El segon enllaç correspon a la pàgina web on es poden descarregar els diversos instal·ladors, tant per Windows com per MacOS, de l'última versió del CoDaPack que s'ha desenvolupat en el projecte.

A continuació els dos enllaços:

- **Enllaç del GitHub:** `https://github.com/mcomas/codapack/tree/daniel_branch`
- **Enllaç per descarregar els instal·ladors:** `http://ima.udg.edu/codapack/`

15 Manual d'usuari i/o instal·lació

En aquest apartat es mostrarà tant el manual d'usuari de l'aplicació desenvolupada en el projecte com també l'explicació de com realitza la instal·lació del programa tant en sistemes operatius Windows com sistemes operatius MacOS.

15.1 Manual d'usuari

El manual d'usuari el dividirem en dues parts, la primera part correspon a un manual d'usuari del CoDaPack d'una versió anterior al projecte, que va ser realitzat per Thio-Henestrosa, S. and Comas, M.. Aquest manual d'usuari pot servir per a alguns aspectes de l'aplicació actual que encara són presents i alguns funcionaments en comú. A continuació l'enllaç corresponent al manual.

- `http://ima.udg.edu/codapack/assets/codapack-manual.pdf`

La segona part del manual d'usuari correspon als menús de help per cada una de les rutines del CoDaPack. Aquests menús que s'han desenvolupat per ajudar als usuaris a saber com utilitzar cada una de les rutines ho considerem també com a manual d'usuari.

15.2 Instal·lació

15.2.1 Windows

Per instal·lar l'aplicació en Windows, primer de tot òbviament el que hem de fer és baixar-nos l'aplicació del CoDaPack des de l'enllaç que s'ha adjuntat anteriorment en la secció **[14]**. Un cop baixada l'aplicació hauríem de tenir l'aspecte que es mostra en la Figura 151.

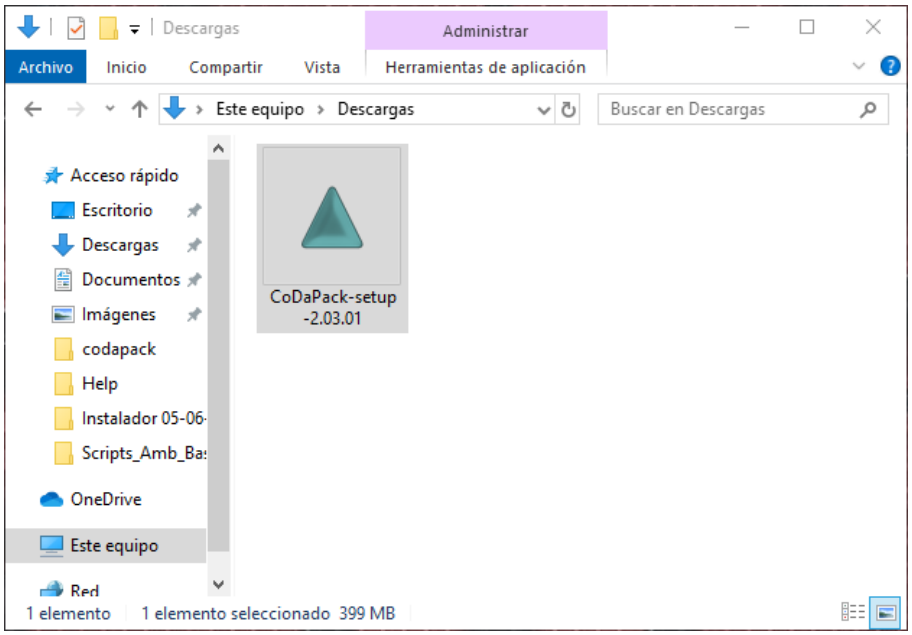


Figura 151: Arxiu exe de l'instal·lador per Windows.

Simplement en fer doble clic en l'arxiu apareix la següent finestra que es mostra en la Figura 152.

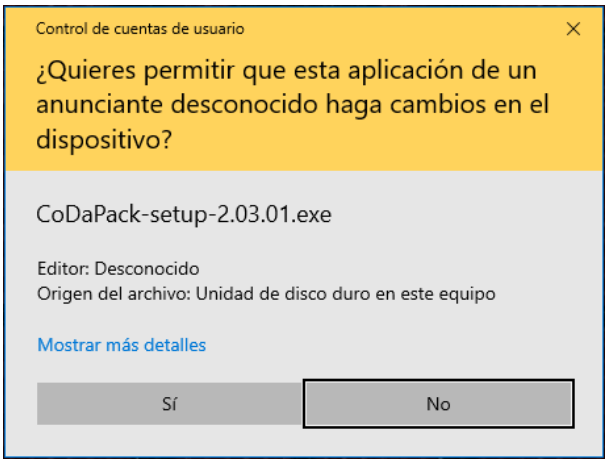


Figura 152: Alert instal·lador de Windows.

En aquesta finestra que ens apareix hem de fer clic en "Sí" per poder procedir a realitzar la instal·lació del programa. En fer aquest clic ens apareix la següent finestra de la Figura 153.

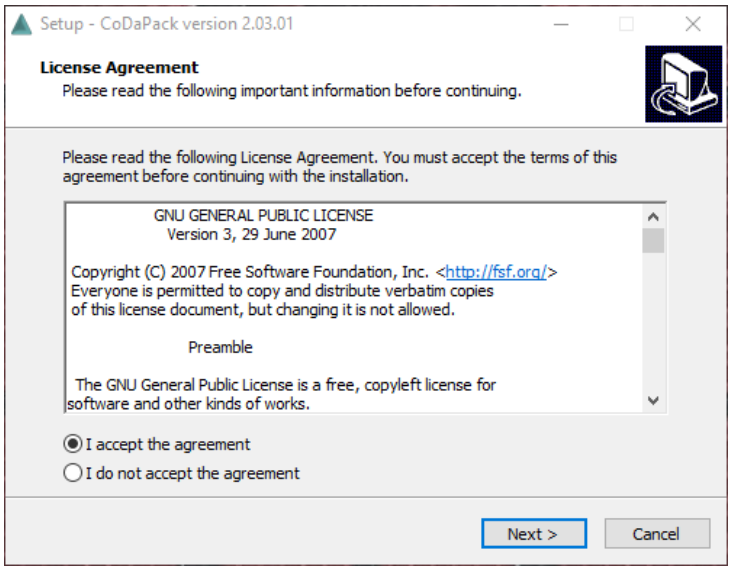


Figura 153: Pàgina 1 instal·lador de Windows.

Aquí simplement hem d'acceptar el contracte de la llicència del programa, en fer clic en l'opció d'acceptar se'ns habilita l'opció de "Next >;" que és on farem clic per seguir amb la següent finestra de la Figura 154.

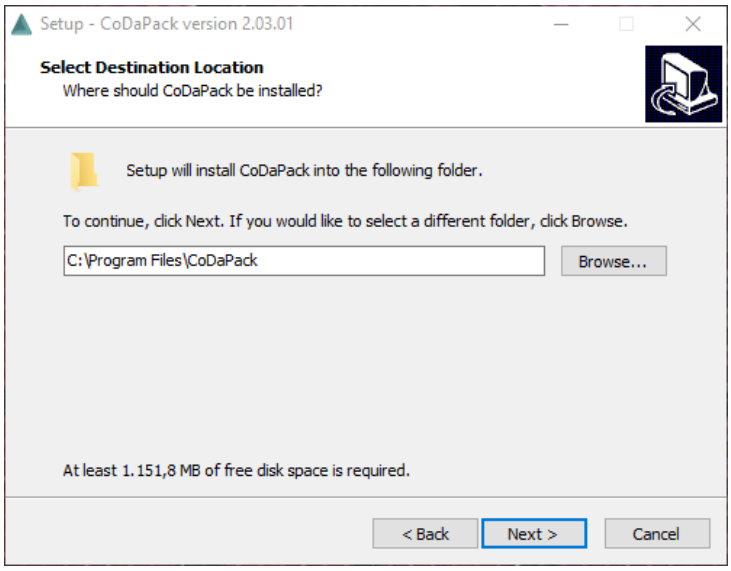


Figura 154: Pàgina 2 instal·lador de Windows.

En aquesta finestra se'ns demana la ruta on volem que s'instal·li el programa, en aquest cas deixem la ruta per defecte i fem clic en següent per mostrar la següent finestra de la Figura 155.

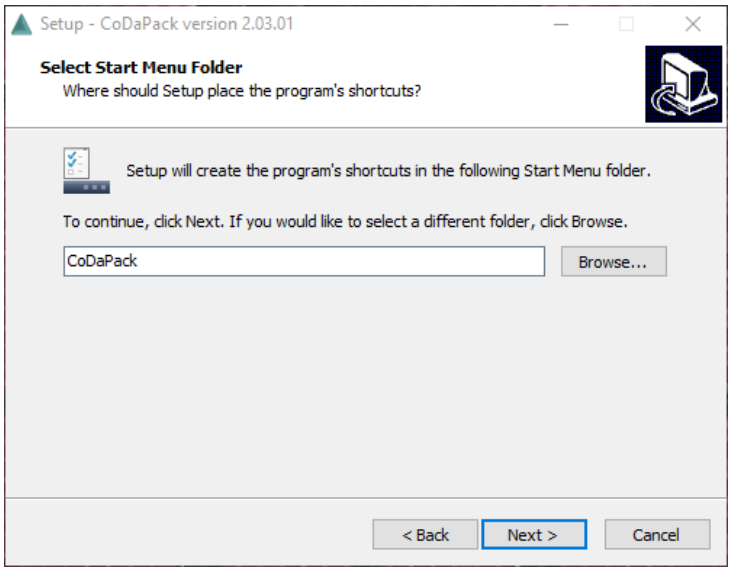


Figura 155: Pàgina 3 instal·lador de Windows.

Aquesta part simplement és per l'Start Menu Folder que en aquest cas no tocarem res i simplement farem clic en "Next >". A continuació la següent finestra en la Figura 156.

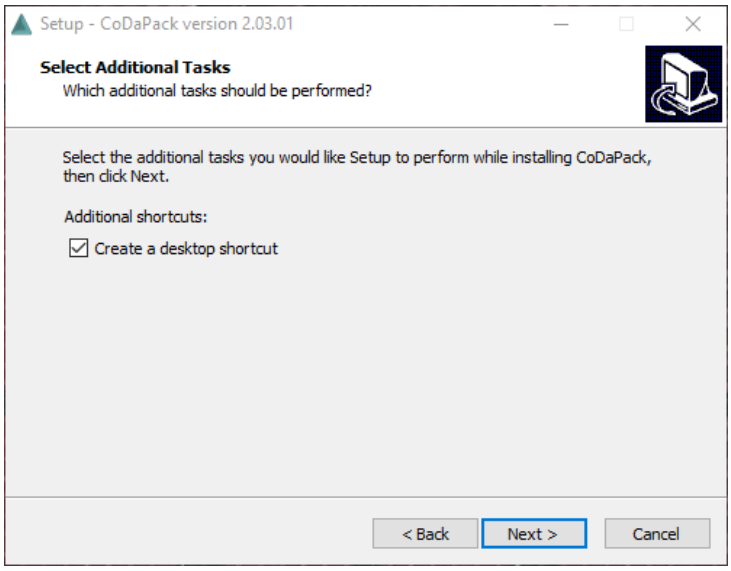


Figura 156: Pàgina 4 instal·lador de Windows.

Aquí el que se'ns dona l'opció de seleccionar si volem crear un accés directe en l'escriptori de l'aplicació o no, en aquest cas diem que sí que el volem. Passem a la següent finestra que es mostra en la Figura 157.

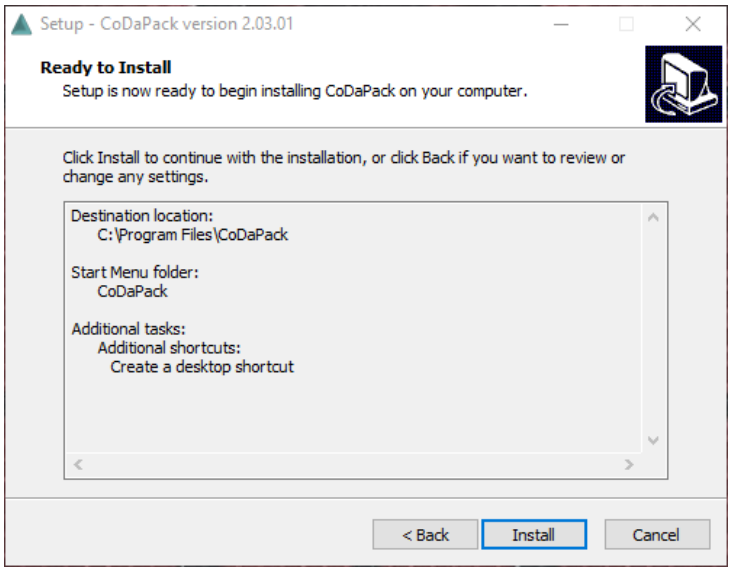


Figura 157: Pàgina 5 instal·lador de Windows.

Ara en aquesta finestra es mostra un petit resum de la instal·lació que es farà en el sistema. Finalment tenim l'opció per procedir ja a la instal·lació del programa, per tant fem clic en aquesta opció.

En la Figura 158 podem veure com s'està realitzant la instal·lació del programa en el sistema.

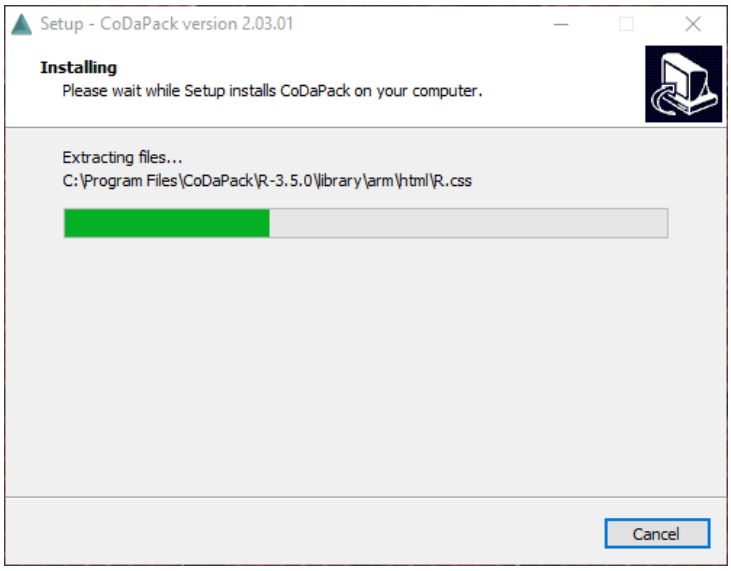


Figura 158: Pàgina 6 instal·lador de Windows.

Ara només queda esperar que acabi el procés d'instal·lació, quan això passi apareixerà la següent finestra que es mostra en la Figura 159.

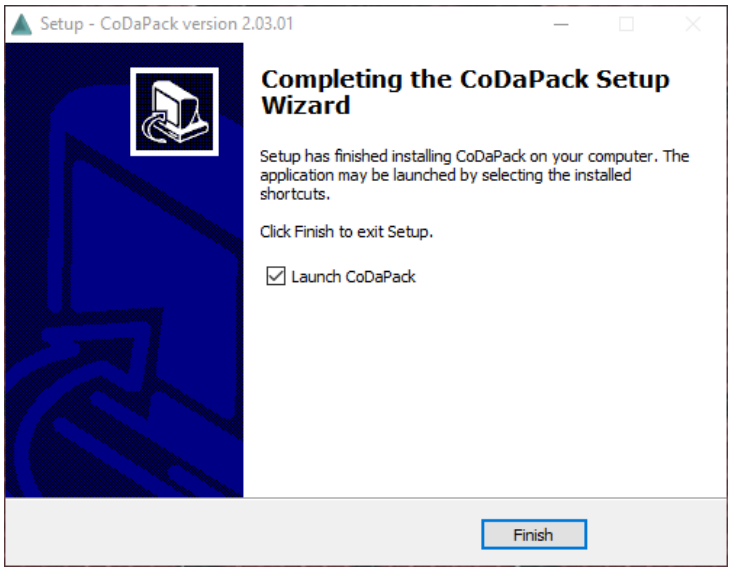


Figura 159: Pàgina 7 instal·lador de Windows.

Per mostrar que ja està instal·lat el programa en el sistema, mostrem l'escriptori on apareix la icona de l'aplicació del CoDaPack, podem veure això en la Figura 160.

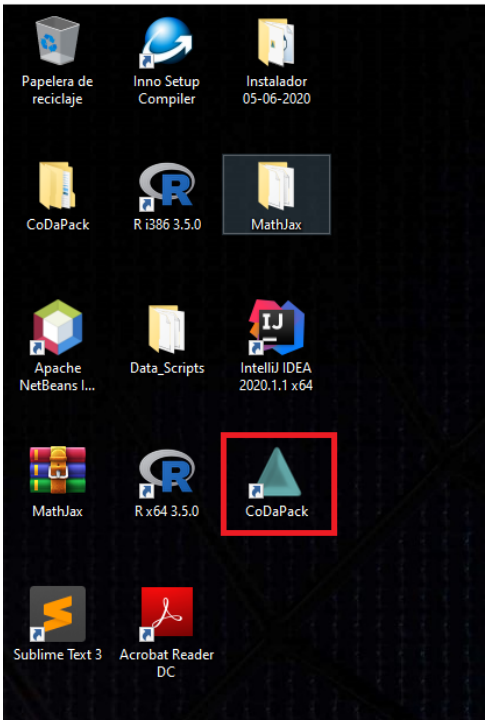


Figura 160: Escriptori de Windows amb el CoDaPack.

15.2.2 MacOS

Per instal·lar l'aplicació en MacOS, primer de tot òbviament el que hem de fer és baixar-nos l'aplicació del CoDaPack des de l'enllaç que s'ha adjuntat anteriorment en la secció **[14]**. Un cop baixada l'aplicació hauríem de tenir l'aspecte que es mostra en la Figura 161.

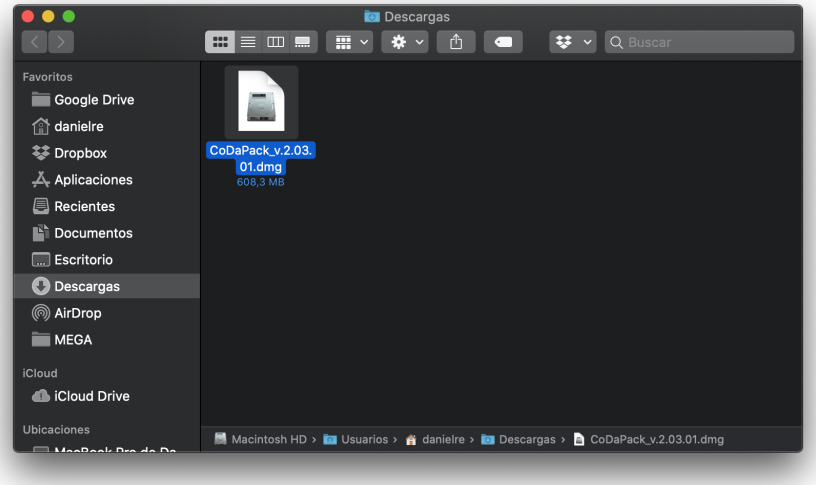


Figura 161: Arxius DMG de l'instal·lador per MacOS.

Simplement en fer doble clic en l'arxiu se'ns descomprimeix l'arxiu automàticament i apareix la següent

finestra que es mostra en la Figura 162.

Figura 162: Instal·lador MacOS.

En la imatge podem veure l'instal·lador per MacOS, on apareix una icona amb el CoDaPack i un àlies de la carpeta d'Aplicacions. Per instal·lar el CoDaPack en el sistema és tan fàcil com fer drag del CoDaPack a la carpeta d'aplicacions perquè es copiï en ella.
Un cop fet això si anem a mirar a la carpeta d'aplicacions, ja tindríem el CoDaPack instal·lat en el sistema com es pot veure en la Figura 163.

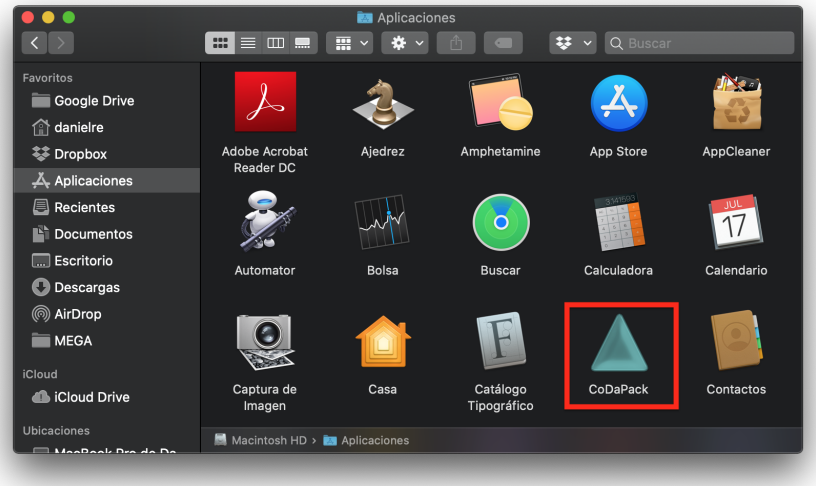


Figura 163: Aplicacions MacOS.