Universitat de Barcelona

Facultat d'Economia i Empresa

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Facultat de Matemàtiques i Estadística

Treball Final de Grau

Integració de diferents fonts de dades òmiques i visualització de les variables originals mitjançant tècniques de *Machine Learning*

Grau d'Estadística

Autor: Laura Riba Archilla

Director: Esteban Vegas Lozano

Convocatòria: Setembre 2014

Resum

En l'última dècada s'han desenvolupat noves tecnologies d'alt rendiment, les quals generen un volum de dades biològiques tan gran que ha motivat la creació de nous algorismes en el camp de la bioinformàtica per analitzar les dades generades. Aquests avenços han revolucionat la biologia molecular i han conduït a una nova mentalitat en la qual es desenvolupa una visió global dels sistemes biològics. En aquest context, actualment hi ha dues grans vies d'investigació: la integració de dades òmiques i la visualització de les variables originals. L'anàlisi de dades òmiques de més d'un tipus de forma simultània combinada amb la visualització de les relacions entre els milers de variables biològiques pot portar a una millor comprensió dels processos biològics. En aquest projecte s'estudia la tècnica del Kernel PCA juntament amb procediments per a representar les variables originals, s'aplica a dos conjunts de dades òmiques i es presenta de forma accessible amb aplicacions web interactives.

Paraules clau: *dades òmiques*, *integració de dades*, *aprenentatge automàtic*, *mètodes kernel*, *visualització*, *Kernel PCA*, *Shiny*

Abstract

The development in the last decade of the high-throughput technologies, new techniques for measuring biological data, has dramatically changed our views on molecular biology. Whereas a few years ago each gene or protein was studied as a single entity, new technologies allow to analyse large numbers of genes or proteins simultaneously. As a result, biological processes are studied as complex systems of functionally interacting macromolecules. This new mindset has led to the rise of new disciplines, such as genomics, proteomics and transcriptomics, in the so-called "omics era". All of them have in common that are based on the analysis of a large volume of heterogeneous biological data. These datasets encourage researchers to develop new algorithms in the field of bioinformatics for its interpretation.

Within this context, there are currently two major research challenges: omics data integration and visualization of the input variables. The analysis at the same time of integrated omics data combined with the visualization of relationships between the thousands of biological variables generated may lead to a better understanding of the global functioning of biological systems. Although individual analysis of each of these omics data undoubtedly results into interesting findings, it is only by integrating them that one can gain a global insight into cellular behavior. A systems approach thus is predicated on the integration of multiple independent datasets. Visualization is a key aspect of both the analysis and understanding of the omics data. The challenge is to create clear and meaningful visualizations that give biological insight, despite the complexity of the data.

In this project, first we present the main types of omics data, the associated high-throughput technologies and the challenges that present its analysis, including the integration of omics data. After this, we give an overview of the discipline of machine learning, which provides algorithms and techniques to analyze omics data. In addition, special attention is paid to kernel methods, which are one of the most powerful methods for integrating heterogeneous data types. In the present work, we analyze the integration of data from several sources of information using the Kernel PCA technique together with a set of procedures to represent the input variables. Then we apply them to two different omics datasets. In addition, we provide this technique in an accessible way by the creation of interactive web applications.

Keywords: *omics data*, *data integration*, *machine learning*, *kernel methods*, *visualization*, *Kernel PCA*, *Shiny*

Índex

Índex de figures

Índex de taules

Notacions

x	Valor escalar
$\mathbf{x}$	Vector fila
$\mathbf{x}^T$	Vector transposat
$\mathbf{X}$	Matriu
$\mathbf{X}^{-1}$	Matriu inversa
n	Nombre d'observacions
p	Nombre de variables
$\mathcal{X}$	Espai d'entrada o *input space*
$\mathcal{F}$	Espai de característiques o *feature space*
$\phi(\mathbf{x})$	Funció que passa $\mathbf{x}$ de $\mathcal{X}$ a $\mathcal{F}$
$k(\mathbf{x}, \mathbf{x}')$	Funció kernel
$\mathbf{K}$	Matriu kernel

Capítol 1

INTRODUCCIÓ

El present document constitueix la memòria del Treball Final del Grau d'Estadística impartit per la Universitat de Barcelona i la Universitat Politècnica de Catalunya. En aquest primer capítol d'introducció es presenta la justificació, els objectius i la metodologia emprada en el desenvolupament d'aquest projecte, juntament amb l'estructura d'aquesta memòria.

1.1 Justificació

En els últims anys s'han desenvolupat noves tecnologies d'alt rendiment per a mesurar dades biològiques que han revolucionat la biologia molecular. Aquesta explosió en la quantitat de dades biològiques generades, les dades òmiques, ha motivat el desenvolupament de noves tècniques per al seu anàlisi dins del camp de la bioinformàtica. Aquests avenços aconseguits tant en el camp de la biologia com en el camp de la bioinformàtica han donat lloc a una nova mentalitat en la qual es desenvolupa una visió global dels processos biològics. Es passa de l'estudi dels gens o de les proteïnes individuals a l'estudi de tots els gens, de totes les proteïnes i d'altres molècules i de les relacions entre elles al mateix temps en un moment determinat.

Tot i els nous avenços, és relativament recent la investigació de la possibilitat d'analitzar dades òmiques de més d'un tipus de forma simultània. Agafar dos tipus de dades òmiques i analitzar-les de forma combinada segur que proporciona més informació sobre el sistema biològic d'un organisme que no pas si s'analitzen individualment. Aquesta possibilitat ha donat lloc a l'aparició de diverses tècniques per a la integració de grans conjunts de dades òmiques heterogènies.

En el context d'aquestes investigacions, un dels principals objectius amb aquests tipus de dades és poder visualitzar les relacions entre els milers de variables biològiques obtin-

gudes amb les tecnologies d'alt rendiment. La visualització és un aspecte clau tant per a l'anàlisi com per a la comprensió d'aquestes dades, però a la vegada existeix el repte de crear visualitzacions clares i integrades que donin una visió biològica significativa, tot i la complexitat intrínseca de les dades.

Aquest projecte es desenvolupa amb la finalitat de conèixer i proporcionar mètodes que permetin tant la integració de diferents fonts de dades òmiques com la visualització de les variables originals, dos dels principals reptes actuals en Bioinformàtica.

1.2 Objectius

L'objectiu principal que es persegueix amb aquest projecte és conèixer tota una nova vessant on aplicar l'Estadística, el camp de la Bioinformàtica, una àrea que no s'ha vist gaire al llarg del Grau. En el context de la Bioinformàtica, s'utilitzen un tipus especial de dades biològiques, les dades òmiques, i s'apliquen tècniques que provenen de la disciplina del *Machine Learning*. Com a part d'aquest projecte, es desenvolupen les principals característiques d'aquests camps d'estudi.

Per tal de tenir una visió general del món de la Bioinformàtica, en aquest projecte s'han plantejat els següents objectius específics:

- Descriure les característiques de les dades biològiques que s'obtenen dins de l'àmbit de la Bioinformàtica, i en especial els reptes que suposa el seu anàlisi i la seva integració des del punt de vista de l'Estadística.
- Proporcionar una visió general de la disciplina del *Machine Learning*, i en concret dels tipus de tasques a les que s'adrecen les seves tècniques.
- Introduir conceptes sobre les funcions kernel i els mètodes kernel, amb la finalitat de destacar la seva potència per a la integració de fonts de dades de diversos tipus.
- Comentar algunes tècniques estadístiques multivariants per a la integració de dades, i centrar-se en detall en el Kernel PCA amb visualització de les variables originals.
- Aplicar la tècnica del Kernel PCA juntament amb els procediments per a representar les variables originals a diversos conjunts de dades òmiques.
- Desenvolupar un paquet de R associat al Kernel PCA amb visualització de les variables originals, i crear aplicacions web interactives per a facilitar l'ús de la tècnica als no usuaris de R.

1.3 Metodologia

La principal característica d'aquest Treball Final de Grau és l'alt nivell d'aprenentatge de nous coneixements que implica. Les dades amb les quals es treballa, les tècniques que s'apliquen i la disciplina en la que s'engloba són camps que, abans de la realització d'aquest treball, m'eren totalment desconeguts. Així doncs, la primera fase d'aquest projecte, i segurament la més important, ha estat la de la recerca bibliogràfica i de recursos *online* referents a dades òmiques, *machine learning* i mètodes kernel, i sobretot al Kernel PCA amb visualització de les variables originals, que és la tècnica principal que es desenvolupa en el projecte. Tots aquests recursos es poden trobar a la bibliografia situada al final d'aquesta memòria.

Un cop assimilats els nous conceptes, la segona fase ha estat la de l'aplicació dels diferents mètodes estudiats a diversos conjunts de dades, tant simulades per a comprovar les propietats de les tècniques com a dades òmiques reals. La part fonamental en aquesta fase d'utilització dels mètodes ha estat l'ús del *software* R, i en concret de dos nous paquets especials per a aquests mètodes. Pel que fa a les dades òmiques utilitzades, un dels conjunts es pot trobar al paquet `mixOmics` de R, mentre que l'altre forma part de les dades suplementàries proporcionades per l'estudi de Tenenhaus et al. (2014).

Després d'assimilar i aplicar els nous coneixements, l'última fase del projecte s'ha dedicat a la divulgació d'aquests coneixements. Actualment ja no és suficient amb proporcionar eines i tècniques per analitzar la gran quantitat de dades que es produeixen, sinó que també és necessari que es faci d'una manera "fàcil", en el sentit que no només les puguin utilitzar experts sinó que estiguin a l'abast de qualsevol persona que vulgui desenvolupar el seu negoci. És per això que en aquesta etapa del projecte s'han desenvolupat aplicacions web interactives per a que qualsevol usuari pugui aplicar les tècniques explicades encara que no sàpiga R. Aquestes aplicacions s'han desenvolupat amb *Shiny*, un nou paquet de R creat pels autors de RStudio.

Paral·lelament a aquestes fases s'ha dut a terme la redacció de la present memòria, combinant l'ús de l'editor de Latex *Texmaker* amb el paquet *Sweave* de R, que permet integrar codi R amb instrucccions de Latex.

1.4 Estructura

La memòria del present Treball Final de Grau està formada per un total de set capítols, incloent el corresponent a aquesta introducció. El projecte està estructurat de forma que cada un dels capítols dóna resposta als objectius específics plantejats anteriorment.

En el segon capítol es presenten les característiques de les dades que es tractaran al projecte, les dades òmiques. S'expliquen quins són els principals tipus de dades òmiques, quines són les tecnologies d'alt rendiment necessàries per a obtenir-les i quins són els principals problemes que presenta el seu anàlisi. A més, es destaca el repte que suposa la integració de dades òmiques, que és un dels principals objectius d'aquest projecte.

El tercer capítol està dividit en dos apartats diferenciats. En primer lloc es dóna una visió general de la disciplina del *Machine Learning* o Aprenentatge Automàtic, incloent la seva descripció, una mica de la seva història i algunes de les aplicacions actuals, que mostren la seva importància en el món actual. Es para especial atenció als tipus de tasques a les que dóna solució aquest camp, i es comenten alguns dels conceptes clau a l'hora d'aplicar tècniques de *Machine Learning*.

El segon apartat del tercer capítol està dedicat als mètodes kernel, que proporcionen extensions no lineals d'un gran nombre de tècniques lineals mitjançant l'ús de funcions kernel. En aquest apartat es dóna una visió tant teòrica com pràctica de les funcions kernels i es destaquen els grans avantatges que ofereixen aquests mètodes.

El quart capítol està dedicat a les tècniques d'integració de dades, especialment al Kernel PCA. En un primer apartat s'ofereix una introducció a diferents tècniques multivariants estadístiques que permeten la integració de dades, per després passar a explicar en detall la tècnica utilitzada en aquest treball, el Kernel PCA. A més, es presenten un conjunt de procediments adreçats a la problemàtica de la visualització de les variables originals. Per últim, es mostra la implementació d'aquests mètodes en el *software* R a partir d'un conjunt de dades simulades.

Al cinquè capítol d'aquesta memòria es troba l'aplicació de les tècniques vistes en el capítol anterior a dos conjunts reals de dades òmiques. En el primer conjunt es combinen dades d'expressió dels gens i de concentracions d'àcids grassos en un estudi en ratolins, mentre que en el segon cas s'integren dades genòmiques i transcriptòmiques pertanyents a nens amb glioma.

El sisè capítol està adreçat a la implementació i divulgació del Kernel PCA amb visualització de les variables originals per dos vies diferents. La primera via consta de la creació d'un paquet de R amb les funcions necessàries per a que qualsevol usuari d'aquest *software* pugui aplicar les tècniques vistes en els capítols anteriors a qualsevol conjunt de dades. La segona via és la creació d'aplicacions web interactives amb Shiny, per a que qualsevol usuari pugui aplicar les tècniques encara que no tingui o no sàpiga utilitzar R.

En l'últim capítol de la memòria s'exposen les principals conclusions que s'han extret al llarg de la realització d'aquest projecte. Seguidament es troben totes les referències consultades per a l'elaboració del treball i a l'annex s'adjunta el codi R utilitzat en els diferents apartats del projecte.

Capítol 2

DADES ÒMIQUES

El terme "dades òmiques" fa referència a les dades que es tenen en disciplines com la genòmica, la proteòmica o la transcriptòmica. Aquestes tenen en comú que es basen en l'anàlisi d'un gran volum de dades biològiques, pel que fan ús de la bioinformàtica en la seva interpretació. En aquest capítol es presenten els principals tipus de dades òmiques, quines són les tecnologies d'alt rendiment necessàries per a obtenir-les i quins són els principals reptes que es donen en quant al seu anàlisi. Per últim, es destaca el repte que suposa la integració de dades òmiques.

2.1 Presentació de les dades òmiques

Els avenços aconseguits en l'última dècada tant en el camp de la biologia com en el camp de la bioinformàtica han obert la porta a una nova mentalitat en la qual es desenvolupa una visió global dels processos biològics. Hi ha hagut una explosió en la quantitat de dades biològiques generades degut a un nombre creixent de tècniques que permeten la detecció simultània d'un gran nombre de variables biològiques com gens, proteïnes o metabòlits. Aquest concepte de globalització es veu reflectit en el desenvolupament del que s'ha denominat com "l'era òmica". El sufix "-oma" té origen llatí i significa "conjunt de", pel que s'utilitza per a referir-se a l'estudi de la totalitat o del conjunt d'alguna cosa. Per tant, l'addició d'aquest sufix als diferents estudis biològics cobreix les noves aproximacions en les què s'està enfocant la biologia recentment.

2.1.1 Tipus de dades òmiques

Les dades òmiques cada cop s'estenen a una més àmplia gamma de camps biològics, tot i que els principals són:

Genòmica Es denomina genòmica al conjunt de ciències i tècniques dedicades a l'estudi

integral del funcionament, el contingut, l'evolució i l'origen del genoma. La genòmica té com a objectiu predir la funció dels gens a partir de la seva seqüència o de les seves interaccions amb altres gens. Es destaquen les següents branques:

- **Genòmica funcional**: és la disciplina que s'orienta cap a la recol·lecció sistemàtica d'informació sobre les funcions exercides pels gens.
- **Genòmica estructural**: és la branca de la genòmica orientada a la caracterització i localització de les seqüències que conformen el ADN dels gens, permetent d'aquesta manera l'obtenció de mapes genètics dels organismes.
- **Genòmica comparativa**: es basa en la comparació de grans parts del genoma de diferents organismes per a estudiar les similituds i les diferències biològiques bàsiques, així com les relacions evolutives entre aquests organismes.

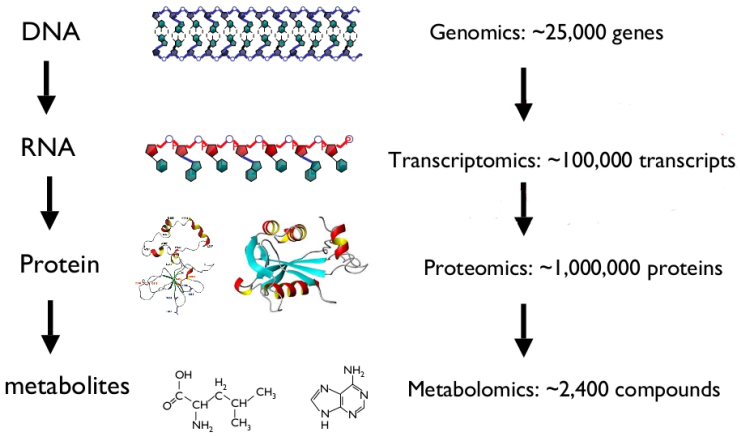
Transcriptòmica La transcriptòmica cobreix l'estudi del transcriptoma, és a dir, del conjunt de totes les molècules d'ARN, incloent l'ARN missatger, l'ARN ribosòmic, l'ARN de transferència i altres ARN no codificants, produïts en una cèl·lula o en una població de cèl·lules.

Proteòmica La proteòmica és l'estudi a gran escala de les estructures i funcions de totes les proteïnes d'una cèl·lula. Inclou també la determinació de les interaccions i dels perfils d'expressió.

Metabolòmica La metabolòmica és l'estudi i comparació del metaboloma, és a dir, la col·lecció de tots els metabòlits (molècules de baix pes molecular) presents en una cèl·lula, teixit o organisme en un moment donat.

Tots aquests camps biològics tenen en comú la gran quantitat de dades que genera el seu estudi, tal i com es mostra en la següent figura, extreta de [22]:

Figura 2.1: Tipus principals de dades òmiques



A part dels quatre tipus principals definits anteriorment, altres camps són la nutrigenòmica, la farmacogenòmica o la interactòmica. De fet, existeixen centenars de termes "òmics". Una llista exhaustiva dels diferents tipus de dades òmiques es pot trobar a `http://www.genomicglossaries.com/content/omes.asp`, proporcionada per l'Institut de Salut de Cambridge (CHI).

2.1.2 On trobar dades òmiques

En aquest apartat es descriuen breument alguns dels projectes biològics més rellevants actualment, donant prioritat a aquells en els que els conjunts de dades resultants estan (o estaran) a disposició del públic.

***Gene Expression Omnibus* (GEO)** És un repositori públic de dades de genòmica funcional, que exigeix que les dades a compartir compleixin les directrius MIAME (*Minimum Information About a Microarray Experiment*). El repositori GEO requereix que tant les dades en brut (*raw data*) com les normalitzades estiguin disponibles, que les mostres estiguin anotades, i que els protocols de laboratori i de processament de dades estiguin descrits. A més, proporciona eines de consulta per ajudar als usuaris a descarregar experiments i perfils d'expressió gènica de forma acurada. Es pot accedir en `http://www.ncbi.nlm.nih.gov/geo/`.

***The Cancer Genome Atlas* (TCGA)** L'objectiu principal d'aquest projecte és generar coneixements sobre l'heterogeneïtat dels diferents subtipus de càncer mitjançant la creació d'un mapa de les alteracions moleculars per a cada tipus de càncer en múltiples nivells. Disposa d'un portal de dades que proporciona una plataforma per als investigadors per a buscar, descarregar i analitzar conjunts de dades generades pel projecte TCGA. Aquest repositori conté informació clínica, dades de caracterització genòmica i anàlisis de seqüències d'alt nivell dels genomes tumorals, les quals es troben a `https://tcga-data.nci.nih.gov/tcga/`.

***Encyclopedia of DNA Elements* (ENCODE)** Aquest consorci és una col·laboració internacional de grups de recerca finançats per l'Institut Nacional d'Investigació del Genoma Humà (NHGRI), que té com a objectiu identificar tots els elements funcionals en el genoma. Inclou la descripció dels elements que actuen a nivell de ARN i de proteïnes, i els elements reguladors que controlen les cèl·lules i les circumstàncies en què un gen està actiu. Està disponible a `http://genome.ucsc.edu/ENCODE/`.

A més dels recursos anteriors, es destaca ***Pathguide: the Pathway Resource List*** (`http://pathguide.org`), que conté informació sobre aproximadament 550 recursos (a data d'agost del 2013) relacionats amb processos biològics i amb la interacció molecular.

2.2 Materials: Tecnologies d'alt rendiment

Les tecnologies òmiques utilitzen tècniques de detecció d'alt rendiment (*high-throughput techniques*) per a generar les grans quantitats de dades necessàries per permetre una comprensió a nivell de sistema de les correlacions i les dependències entre els components moleculars.

Hi ha una gran varietat de definicions del terme *high-throughput* (HT), tot i que simplificant es pot aplicar als casos en què s'utilitza l'automatització per augmentar el rendiment d'un procediment experimental, capacitant-lo per dur a terme un gran nombre de processos en paral·lel. Aquest tipus de tecnologies sorgeixen de camps com la robòtica, l'òptica, la química, la biologia i la investigació en anàlisi d'imatges.

Les ciències òmiques requereixen l'ús de tecnologies d'alt rendiment ja que poden manejar mostres biològiques extremadament complexes en grans quantitats amb alta sensibilitat i especificitat. A continuació es comenten breument les tecnologies més utilitzades per als tipus de dades òmiques abans destacats:

Genòmica En el camp de la genòmica, les tecnologies d'alt rendiment tenen com a principal objectiu la seqüenciació de l'ADN. Els mètodes actuals poden seqüenciar directament fragments d'ADN relativament curts (entre 300 i 1000 nucleòtids de llarg) en una sola reacció, pel que els costos s'han reduït dràsticament. Alguns d'aquests mètodes de seqüenciació són les *next-generation sequencing (NGS) technologies* i la *Illumina sequencing*.

Transcriptòmica L'anàlisi global de l'expressió gènica es duu a terme tant per hibridació amb *microarrays* d'oligonucleòtids com mitjançant el recompte d'etiquetes de seqüències. Recentment han sorgit altres tècniques revolucionàries que generen els perfils d'expressió de tot el genoma, com per exemple la *ChIP-Seq* i la *ChIP-chip*.

Proteòmica Hi ha vàries tècniques d'alt rendiment que s'apliquen en aquesta àrea, tot i que les dues principals són les basades en espectrometria de masses, com per exemple el *MALDI-TOF*, i els *microarrays* per a proteïnes (*Protein Chips*).

Metabolòmica Hi ha múltiples tècniques que es poden utilitzar per a caracteritzar els metabòlits, i donat que cada tècnica té avantatges i desavantatges associats, la combinació de diferents tecnologies d'anàlisi és una bona opció. Les tècniques analítiques que s'utilitzen en aquest camp inclouen la ressonància magnètica nuclear (RMN), i la combinació de l'espectrometria de masses (MS) amb la cromatografia de gasos (GC-MS), amb la cromatografia líquida (LC-MS), i amb l'electroforesi capil·lar (CE-MS).

Actualment existeixen mètodes d'alt rendiment per pràcticament tots els diferents dominis òmics. El repte és evitar els obstacles que apareixen en l'emmagatzematge, l'anotació i l'anàlisi de les dades tan heterogènies que produeixen aquestes tecnologies. Com que cada cop s'utilitzen dades d'aquest tipus en un context més global, cal fixar normes i protocols per a poder compartir aquesta informació de manera adequada. I en quant a l'anàlisi d'aquestes dades, aquest seria impossible sense la bioinformàtica.

2.3 Mètodes: Bioinformàtica

La bioinformàtica es pot definir com el camp científic interdisciplinari que desenvolupa mètodes i programari per a emmagatzemar, recuperar, organitzar i analitzar dades biològiques. Es basa en coneixements derivats de la informàtica, l'estadística, les matemàtiques i l'enginyeria per a donar sentit a les grans quantitats de dades produïdes en la investigació òmica.

El paper de la bioinformàtica, doncs, és doble: per un costat ha de proporcionar les estructures en les quals emmagatzemar la informació d'una manera que sigui recuperable i comparable tant amb dades similars com amb altres tipus d'informació; per una altra banda, s'encarrega del desenvolupament d'algorismes i tècniques estadístiques per a analitzar els grans conjunts de dades biològiques.

En particular, tal i com explica [23], l'anàlisi d'aquestes dades biològiques suposa un repte considerable per les següents cinc raons:

(a) Problema de les comparacions múltiples

Les tecnologies d'alt rendiment donen lloc a un vector de milers de variables per a cada mostra biològica analitzada. Per exemple, en un experiment amb *microarrays* es poden obtenir les expressions de 10 000 gens per a un individu. Si es volen determinar quins gens s'expressen diferencialment entre dues condicions diferents, utilitzant el nivell de significació habitual del 5% es trobarien 500 falsos positius. Si en realitat hi ha un petit nombre de, per exemple, 100 gens que són realment regulats diferencialment, es barrejarien amb els 500 falsos positius anteriors sense cap informació a priori que permeti discriminar els veritables positius del falsos positius.

Aquesta dificultat per a distingir els veritables positius del falsos positius en l'anàlisi de conjunts de gran dimensió es coneix com el *problema de les comparacions múltiples*, un problema que es torna més greu com més gran sigui el nombre de variables a comparar. És per això que encara que un sistema informàtic tingui la potència com per manegar

tasques computacionals d'aquest tipus, s'ha d'evitar la recerca i comparació exhaustiva.

La solució passa per recórrer a algorismes heurístics que controlin altes taxes de falsos positius i que investiguin una porció molt petita de totes les solucions. La major part d'aquests algorismes es basen en models probabilístics i tècniques d'inferència estadística que maximitzen la potència estadística per identificar vertaders positius al mateix temps que controlen les taxes de falsos positius.

(b) Gran dimensionalitat de les dades biològiques

El segon repte és fer front a la naturalesa de les dades biològiques, ja que aquestes són dades de gran dimensió. Això fa que siguin molt difícils de visualitzar gràficament i que necessitin anàlisis complexos amb dificultat en quant a la interpretació. Per això, és necessari entendre i utilitzar tècniques de reducció de la dimensió, amb les quals passar de problemes amb dades de grans dimensions a problemes de dimensions inferiors, però conservant la variabilitat d'interès en les dades biològiques.

(c) El paradigma de la n-petita i la p-gran

Els mètodes estadístics convencionals estan pensats o funcionen bé quan es tenen molts individus (n gran) i poques variables (p petita), és a dir, quan el nombre d'observacions independents és bastant superior al nombre de paràmetres el valor dels quals cal estimar. El problema és que la situació en la majoria de problemes bioinformàtics és exactament la inversa. Per exemple, en experiments amb *microarrays*, per a un individu es tenen desenes de milers de gens, mentre que el nombre de mostres independents (de diferents pacients) és sovint com a màxim de poques desenes. I a vegades fins i tot més baix, degut als alts costos experimentals i a la limitació de materials biològics. Per tot això, és important seleccionar eines d'anàlisi estadística que proporcionin alta especificitat i sensibilitat sota aquestes circumstàncies.

(d) Soroll en les dades biològiques "d'alt rendiment"

Les grans bases de dades biològiques tenen inevitablement soroll, ja que la informació biològica i els senyals d'interès s'observen sovint amb molts altres factors aleatoris o parcials que poden obstruir les senyals principals i la informació d'interès. Cal utilitzar algorismes capaços de reduir i descompondre diverses fonts d'error, i també cal fer un control de qualitat acurat de les dades inicials.

(e) Integració de fonts d'informació múltiples i heterogènies

Un dels reptes de la biologia i la genòmica funcional és la integració de la informació proteòmica, transcriptòmica, metabolòmica i clínica per donar una imatge més completa dels organismes vius. El problema és que aquests conjunts d'informació contenen dades amb característiques i formats molt diferents, la integració dels quals mitjançant les tècniques estadístiques d'inferència no és feina fàcil. En el següent apartat s'explica amb més detall aquesta problemàtica.

2.4 Integració de dades òmiques

En aquest apartat es para especial atenció al repte que suposa la integració de dades òmiques. Primer es defineix el concepte d'integració de dades i després es contextualitza en l'àmbit de les dades òmiques.

2.4.1 Integració de diferents fonts de dades

El concepte de *data integration* està sent actualment molt utilitzat en la recerca científica. Mentre que al 2006 hi havia uns mil articles en els quals es mencionava aquest concepte en el títol o en l'abstract, aquesta quantitat s'ha més que doblat al 2013 (Gómez-Cabrero et al., 2014 [11]). Tot i això, encara no hi ha una definició unificada del terme *data integration*.

La integració de dades es pot definir com l'ús de múltiples fonts d'informació per tal de proporcionar una millor comprensió d'un sistema / situació / associació / etc. Es poden definir dos reptes principals associats a la integració de dades: el descobriment de dades i l'explotació de les dades.

Data discovery El descobriment de fonts d'informació es defineix com la identificació de fonts de dades rellevants. Trobar fonts de dades biològiques publicades i disponibles és fàcil, però no ho és tant trobar dades apropiades. Un dels principals problemes és la diversitat dels tipus de dades i de formats existents, donat que cadascun compleix amb un estàndard diferent, cosa que dóna com a resultat la gran heterogeneïtat de dades.

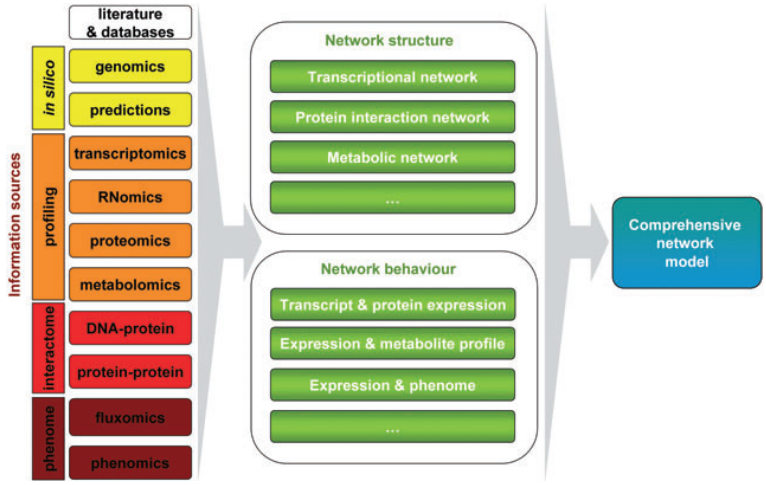
Data exploitation L'explotació de les dades es refereix a l'ús eficaç de la informació col·lectiva per tal d'obtenir nous coneixements. Implica l'ús de coneixements previs i el seu eficient emmagatzematge, el desenvolupament de mètodes estadístics per tal d'analitzar conjunts de dades heterogenis i la creació d'eines d'exploració de dades que incorporin noves eines de visualització.

2.4.2 Integració de diferents fonts de dades òmiques

En el context de les dades òmiques, la integració de dades suposa un repte més complex donada la gran quantitat de dades que es genera diàriament. Els científics han reconegut que els sistemes biològics no poden ser entesos analitzant conjunts de dades d'un sol tipus donat que la regulació del sistema es dóna en molts nivells. La integració de dades heterogènies és actualment un camp actiu d'investigació on els bioestadístics estan constantment proposant nous enfocaments per millorar l'ús de les dades i els descobriments científics.

La següent figura, extreta de [6], mostra el concepte d'integració de dades òmiques. Es volen integrar diferents fonts d'informació per inferir l'estructura de la xarxa transcripcional, la xarxa d'interacció de proteïnes o la xarxa metabolòmica. A més de desxifrar les estructures, les dades òmiques també permeten analitzar les respostes (perfils de metabòlits, proteïnes i ARNm) desencadenades per cadascuna d'aquestes xarxes i estudiar la seva relació mútua (el comportament de la xarxa). L'objectiu final de la integració de dades és combinar la xarxa transcriptòmica, la xarxa d'interacció de proteïnes i la metabolòmica per a construir models de xarxa integrals i comprensius.

Figura 2.2: Integració de dades òmiques



Per últim, es destaca la necessitat emergent de desenvolupar metodologies per a integrar les dades òmiques amb les dades clíniques dels pacients, amb l'objectiu, en última instància, d'assolir la medicina personalitzada. La medicina personalitzada, en relació amb el tractament davant d'una malaltia d'un pacient concret, és l'administració del fàrmac o conjunt de fàrmacs més idonis i en les dosis adequades per a cada pacient concret en vista de la seva individualitat química i genètica. Es recolza tant en el coneixement de la naturalesa molecular de les malalties com en la individualitat química que posseeix cada pacient. El que s'espera és que l'aplicació adequada de la medicina personalitzada pugui donar lloc a una millor prevenció i a un diagnòstic més precís d'una malaltia.

Capítol 3

MACHINE LEARNING I MÈTODES KERNEL

El Machine Learning es basa en la construcció i l'aplicació d'algorismes que aprenen a partir de dades d'exemple. En aquest capítol es proporciona una visió general sobre aquest camp, i es para especial atenció als mètodes kernel, que permeten construir versions no lineals d'algorismes lineals i així detectar relacions complexes en les dades gràcies a l'ús de funcions kernel.

3.1 Visió general del *Machine Learning*

Per a resoldre un problema en un ordinador cal un algorisme, és a dir, una seqüència d'instruccions que quan s'executin transformaran un *input* en un *output*. Però, per a algunes tasques, desenvolupar aquest algorisme no és gens senzill, com per exemple per diferenciar els missatges *spam* dels correus electrònics legítims. El que sí que es pot fer és compilar milers de correus dels quals es sap de quin tipus són i "aprendre" què significa *spam* a partir d'aquests correus. En altres paraules, el que es desitja és que el propi ordinador extregui per ell mateix l'algorisme, que "autoaprengui". El terme *Machine Learning* es pot traduir com ***aprenentatge automàtic***.

En aquesta secció es comença donant una definició més precisa de què és l'aprenentatge automàtic, que es completa afegint una breu explicació de la seva història. Més endavant es troben algunes de les múltiples aplicacions del *Machine Learning* en el món real, i per últim es comenten quins són els principals tipus de tasques que es poden realitzar i quines tècniques s'apliquen en l'aprenentatge automàtic.

3.1.1 Concepte i definició

El camp del *Machine Learning* tracta de respondre a la pregunta, tal i com va formular Mitchell (2006) [30]:

> "*How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?*"

Aquesta qüestió abasta una gran diversitat de tasques d'aprenentatge, com per exemple com dissenyar robots mòbils autònoms que aprenguin a navegar a partir de la pròpia experiència, com explotar les dades històriques de registres mèdics per saber quins futurs pacients respondran millor als tractaments, o com construir motors de cerca que automàticament es personalitzen segons els interessos dels seus usuaris.

Per ser més precisos, es diu que una màquina **aprèn** respecte a una tasca concreta **T**, una mesura de rendiment **R** i un tipus d'experiència **E**, si de forma fiable el rendiment **R** del sistema en la realització de la tasca **T** augmenta amb l'experiència **E**. En funció de com s'especifiquin **T**, **R** i **E** el *machine learning* també pot anomenar-se mineria de dades, descobriment autònom, actualització de bases de dades, programació per exemples, etc. Alguns exemples de problemes d'aprenentatge automàtic podrien ser:

Tasca T Jugar als escacs.
Mesura de rendiment R Percentatge de partides guanyades als adversaris.
Experiència E Partides jugades.

Tasca T Reconèixer paraules escrites a mà.
Mesura de rendiment R Percentatge de paraules ben reconegudes.
Experiència E Base de dades de paraules escrites a mà reconegudes prèviament.

Una de les característiques bàsiques de l'aprenentatge automàtic és la interdisciplinarietat. Segons Mitchell [30], l'aprenentatge automàtic és una conseqüència natural de la intersecció entre la Computació i l'Estadística. Mentre que la Computació s'ha centrat principalment en com programar manualment els ordinadors, l'aprenentatge automàtic es centra en com aconseguir que els ordinadors es programin ells mateixos, a partir de l'experiència i d'alguna estructura inicial. I pel que fa a l'Estadística, mentre que aquesta s'ha centrat en quines conclusions es poden inferir de les dades, l'aprenentatge automàtic incorpora preguntes addicionals sobre quins algorismes es poden usar per emmagatzemar, indexar, recuperar i combinar aquestes dades, com múltiples subtasques d'aprenentatge es poden incorporar en un sistema major, i també s'adreça a qüestions sobre la complexitat computacional dels problemes. Així doncs, l'aprenentatge automàtic vol construir màquines que, basant-se en l'estadística, es programin a sí mateixes per a resoldre determinades tasques.

Ara bé, per què cal *aprendre*? És necessari recórrer a l'aprenentatge automàtic, entre d'altres motius, perquè:

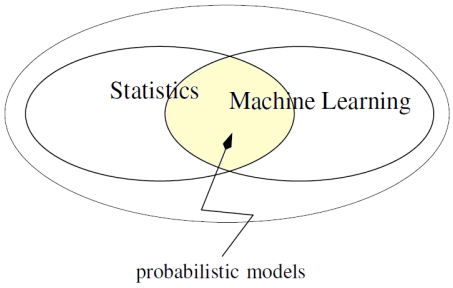
- Pot ser que no existeixi l'experiència humana, com per exemple en la navegació en Mart.
- Algunes tasques no es poden definir bé, excepte amb exemples. Som incapaços d'explicar la nostra experiència, per exemple, en tasques com el reconeixement de veu o de persones.
- Les relacions i les correlacions poden estar amagades entre grans quantitats de dades, tan grans que no és viable per a una persona examinar-les totes, però sí es poden fer algorismes que analitzin aquestes dades i extreguin automàticament informació sobre aquestes, és a dir, que aprenguin.
- L'entorn o la solució canvien amb el temps.
- La solució s'ha d'adaptar a casos particulars.

Enllaç entre l'Estadística i el *Machine Learning*

És fàcil trobar moltes discussions sobre quines diferències hi ha entre l'Estadística i el *Machine Learning* (només cal buscar "*statistics vs machine learning*" a Google). Els dos camps es preocupen per la mateixa pregunta: com podem aprendre de les dades?, però l'enfocament principal de les dues disciplines és diferent. L'Estadística emfatitza la *inferència* (es vol inferir el procés generador de les dades) mentre que l'Aprenentatge Automàtic emfatitza la *predicció* (es vol predir com seran les futures dades respecte a alguna variable). En Estadística es suposa que la mostra prové d'una certa població i es vol aprendre sobre aquesta població, mentre que en l'Aprenentatge Automàtic les dades d'exemple són equivalents a la mostra i originalment no es contempla el concepte de població.

Està clar que hi ha un solapament entre els dos camps: els dos enfocaments d'anàlisi de dades són complementaris, més que no pas contradictoris. Els algorismes de *Machine Learning* tenen una base matemàtica sòlida, i molts incorporen directament estadística. Les tècniques estadístiques s'han desenvolupat de forma independent, però moltes són fonamentalment similars a l'aprenentatge automàtic i produeixen un resultat similar. Les tècniques de validació dels models són les mateixes per a ambdós tipus d'anàlisi. Hi ha molts temes que comencen en un dels dos camps però que després es desenvolupen més en l'altre, i també hi ha temes que estan actius en les dues àrees. Ara bé, tot i el solapament, cal remarcar que l'Estadística cobreix àrees que són (per ara) de poc interès per a l'Aprenentatge Automàtic (disseny d'enquestes, mostratge, etc.), i per la seva part en l'Aprenentatge Automàtic hi ha molts mètodes que no es basen en models probabilístics o conceptes estadístics (com per exemple, els mètodes que optimitzen marges). Aquesta idea es mostra en la imatge a continuació, extreta de [49].

Figura 3.1: L'Estadística i el *Machine Learning*



Una de les raons per les quals hi ha més separació entre les dues disciplines són les diferències en la terminologia: s'usen termes diferents però que es refereixen al mateix concepte. En la taula 3.1 es mostra un petit recull de diversos termes estadístics i la seva traducció dins del món de l'aprenentatge automàtic.

Taula 3.1: Comparativa entre el vocabulari estadístic i el de l'aprenentatge automàtic

Estadística	***Machine Learning***
Variables	Atributs (Camps)
Individus	Instàncies (Registres)
Variable explicativa, predictor, ...	Input
Variable resposta	Output (target)
Model	Xarxa neuronal, arbre de decisió, ...
Coeficients	Pesos
Criteri d'ajust (MV, MQO, ...)	Funció de cost
Estimació	Aprenentatge
Clustering	Aprenentatge no supervisat
Classificació, Regressió, ...	Aprenentatge supervisat

Recentment ha aparegut el terme ***Statistical Learning***, que ve a ser l'enllaç entre l'Estadística i el *Machine Learning*. El seu objectiu és estudiar, en un marc estadístic, les propietats dels algorismes d'aprenentatge així com comprendre conjunts de dades complexos. És una àrea de recent desenvolupament en Estadística i que es barreja amb els desenvolupaments paral·lels en l'aprenentatge automàtic.

Si es vol aprofundir en aquest tema és recomanable donar un cop d'ull a l'entrada al blog de Brendan O'Connor anomenada "Statistics vs. Machine Learning, fight!" [31], així com a l'article "Statistical Modeling: The Two Cultures" de Leo Breiman [2]. També és curiós el glossari entre Estadística i *Machine Learning* fet per Robert Tibshiriani [45]. I per últim, destacar dos llibres per aprendre sobre *Statistical Learning*: *An Introduction to Statistical Learning* [17] i *The Elements of Statistical Learning* [13].

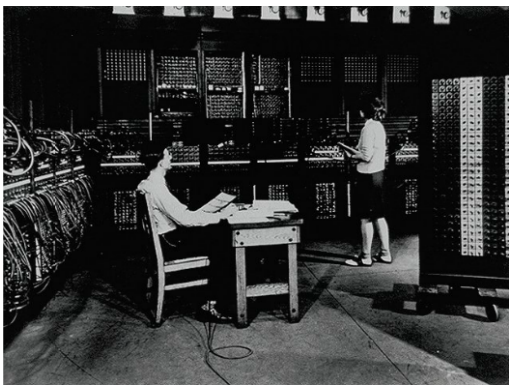
3.1.2 Una mica d'història

En aquest apartat s'expliquen de forma breu els aconteixements més rellevants que marquen l'origen i l'evolució de l'Aprenentatge Automàtic.

ENIAC (1946)

Es pot dir que el punt de partida del *Machine Learning* és el desenvolupament del primer ordinador completament electrònic. Aquest va ser construït en 1946 per John Mauchly i John Eckert, i el van anomenar **ENIAC** (*Electronical Numerical Integrator and Computer*). Pesava uns 30 000 Kg i estava situat en una paret de 3 metres d'alt per 24 metres de llarg. En aquell moment la paraula "ordinador" es referia a una persona fent operacions numèriques en paper, i no implicava una màquina. L'ENIAC es feia funcionar manualment, és a dir, requeria que una persona fes connexions entre les parts de la màquina per a realitzar càlculs. Dins d'aquest context neix el camp de la Intel·ligència artificial, i es concep la idea que el pensament humà i l'aprenentatge es poden descriure i representar de forma lògica amb una màquina d'aquell tipus.

Figura 3.2: ENIAC (1946)



El test de Turing (1950)

El 1950 Alan Turing va proposar una prova per a demostrar l'existència d'intel·ligència en una màquina, i a dia d'avui segueix sent un dels millors mètodes existents. La prova de Turing es basa en la idea que només es pot determinar si una màquina pot realment aprendre si al comunicar-se amb ella no es pot distingir d'un altre ésser humà.

El 7 juny de 2014, el robot conversacional Eugene Goostman (un programa que simula mantenir una conversa amb un humà), que participava en un concurs celebrat a la *Royal Society* per a commemorar el 60è aniversari de la mort de Turing, va guanyar en aconseguir que el 33% del jurat del concurs cregués que Goostman era humà. Més de 60 anys després de la creació de la prova, un programa informàtic ha passat el test de Turing per primer cop, tot i que cal destacar que no tothom està d'acord en aquest fet [25].

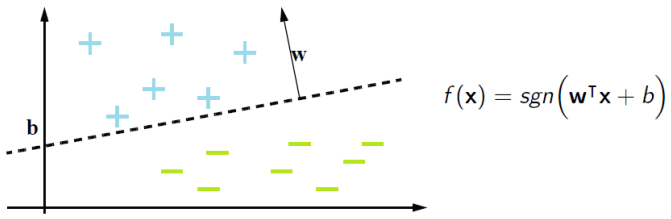
Programa per a jugar a les dames (1952)

Arthur Samuel era un científic d'IBM que va utilitzar el joc de les dames per a crear el primer programa capaç d'aprendre. Aquest programa es va convertir en millor jugador després de jugar molts cops contra ell mateix i contra altres jugadors humans. El programa observava quins moviments eren estratègies guanyadores i adaptava la seva programació per a incorporar aquests moviments. Amb aquest programa Samuel va refutar la idea general de que les màquines no podien anar més enllà dels codis escrits i que no podien aprendre patrons com els humans. Ell va encunyar el terme "machine learning", i el va definir com "*a field of study that gives computers the ability to learn without being explicitly programmed*".

El perceptró de Rosenblatt (1957)

Frank Rosenblatt [38] va dissenyar al 1957 el perceptró, que és el precursor de les xarxes neuronals. El perceptró es pot entendre com una neurona artificial, que té forma de discriminador lineal. És a dir, és una classificador lineal molt simple, un algorisme capaç de generar un hiperplà que separa de forma lineal les observacions d'un grup de les altres.

Figura 3.3: El perceptró de Rosenblatt



El model de perceptró va generar punts de vista sobre el camp de la Intel·ligència Artificial molt optimistes durant la dècada del 1960, ja que es disposava d'un algorisme que et donava la certesa de que si les teves dades eren linealment separables aleshores trobaries en un nombre finit de passos l'hiperplà que les separés. Però després que Marvin Minsky [29] assenyalés al 1969 la limitació d'aquest model en l'expressió de funcions complexes i la seva incapacitat per resoldre problemes no separables linealment, els investigadors van deixar de banda la investigació en xarxes neuronals en la següent dècada.

AI's winter (1970 – 1990)

Degut a la fallida en el model de perceptró, i d'altres decepcions més (com per exemple, el programa informàtic ELIZA [47]), s'entra en el període anomenat com "AI's winter" (l'hivern de la Intel·ligència Artificial): s'investiga menys en aquest camp i cada vegada es destinen menys recursos econòmics. Aquest estancament es va notar sobretot en la recerca

en el camp de les xarxes neuronals, que no va tornar a interessar fins una dècada després, quan es va desenvolupar el model de xarxes neuronals amb múltiples capes. Aquesta situació comença a canviar a mitjans de la dècada dels 80, quan es va inventar el model d'arbre de decisió [36] i es va distribuir com a programari. Aquest model es pot visualitzar, és fàcil d'explicar, versàtil i pot adaptar-se a problemes àmpliament diferents.

Statistical AI (1990's)

A principis dels 90 el camp del *Machine Learning* torna a ser popular i viu la seva gran reactivació gràcies a la intersecció de la Computació amb l'Estadística. Aquests dos camps havien estat molt separats històricament: els estadístics tenien una escola i un marc de pensament i un llenguatge propi, i els informàtics teòrics tenien un altre. Però la seva unió es va traduir en una nova manera de pensar: l'enfocament probabilístic, on s'introdueix incertesa en els paràmetres dels models i es canvia a un enfocament més impulsat per les dades. Moltes de les actuals aplicacions de l'aprenentatge automàtic que han tingut èxit són resultat de les idees desenvolupades en aquell moment.

Una de les tècniques amb més importància que va sorgir en aquesta dècada i que cal mencionar va ser l'algorisme de les Màquines de Vectors Suport (SVM), proposat per Vapnik i Cortes [4] l'any 1995. Ha resultat ser una tècnica amb una sòlida posició teòrica i molt bons resultats empírics. Als anys 2000 es va desenvolupar la versió kernel d'aquest algorisme, incrementant la rellevància i l'ús d'aquesta tècnica.

Estat actual

L'estudi en el camp de l'aprenentatge automàtic ha passat dels esforços d'un grapat d'enginyers informàtics per saber si els ordinadors podien aprendre a jugar i imitar el cervell humà, i del camp de l'Estadística que en gran part ignorava les consideracions computacionals, a una disciplina àmplia en la que s'han produït teories estadístiques i computacionals fonamentals sobre els processos d'aprenentatge.

Molts dels nous algorismes d'aprenentatge, com ara les màquines de vectors suport o les xarxes bayesianes, s'utilitzen de forma rutinària en els sistemes comercials. Cada vegada més el *Machine Learning* té un rol més important en una àmplia gamma d'àrees d'estudi, tal i com demostren les aplicacions que es comenten a continuació. Abans, però, si es vol ampliar una mica la informació sobre la història del *Machine Learning* és recomanable escoltar la xerrada de Hilary Mason anomenada *"Machine Learning: A Love Story"*, presentada a *Strange Loop*, una conferència sobre desenvolupadors de programari, i que és pot trobar a [28].

3.1.3 Algunes aplicacions

Una mesura del progrés en l'aprenentatge automàtic són les seves aplicacions en el món real, algunes de les quals es comenten a continuació. Tot i que ara moltes d'aquestes aplicacions es poden donar per fet, cal destacar que fins ben entrats els anys 80 gairebé no hi havia aplicacions comercials de l'aprenentatge automàtic. Aquestes aplicacions són un exemple del ventall de possibilitats que ofereix aquesta disciplina.

Reconeixement de veu Tots els sistemes comercials disponibles en l'actualitat per al reconeixement de veu utilitzen l'aprenentatge automàtic per entrenar el sistema a reconèixer la parla, ja que la precisió és més gran si s'entrena el sistema que si s'intenta programar-lo a mà. De fet, molts dels sistemes comercials de reconeixement de veu impliquen dues fases d'aprenentatge diferents: una abans que el programa és comprat (entrenament general de manera independent del parlant), i una segona fase després que l'usuari compra el programa (entrenament en funció de la manera de parlar de l'usuari). Potser un dels sistemes de reconeixement de veu més coneguts actualment és el sistema Siri dels Iphones d'Apple.

Visió artificial per ordinador Molts sistemes de visió actuals, des dels sistemes de reconeixement facial fins a sistemes que classifiquen automàticament les imatges microscòpiques de cèl·lules, es desenvolupen utilitzant l'aprenentatge automàtic, i novament la raó és que els sistemes resultants són més precisos que els programes fets a mà. Una aplicació a escala massiva és el seu ús per l'Oficina de Correus dels EUA per a classificar automàticament les cartes amb adreces escrites a mà. Una altra aplicació més popular és el sistema de reconeixement de cares de Facebook.

Control robòtic Les tècniques d'aprenentatge automàtic han estat utilitzades amb èxit en un gran nombre de sistemes robòtics. Per exemple, diversos investigadors han demostrat l'ús de l'aprenentatge automàtic per tal d'adquirir estratègies de control per a vols i acrobàcies aèries estables d'helicòpters. Un altre exemple molt actual és el del cotxe que és condueix sol de Google.

Finances i Indústria bancària La indústria bancària utilitza models de risc creditici per qualificar als sol·licitants segons el seu nivell de compliment esperat en el pagament de les quotes del crèdit. També hi ha models de frau de consum de targetes de crèdit i de predicció de comportament en el mercat de valors.

Diagnòstic mèdic S'utilitzen tècniques d'aprenentatge automàtic per assistir a metges en el diagnòstic segons la història clínica i els símptomes que presenta el pacient. També hi ha models que mesuren el risc de vida d'un accidentat i permeten decidir en cada cas si cal enviar una ambulància, un metge particular o si es pot tractar l'incident per telèfon.

Totes aquestes aplicacions mencionades i infinites més existents són exemples del poder en el món actual de l'aprenentatge automàtic. Les possibilitats són molt àmplies i amb el temps s'arribarà a una tecnologia més intel·ligent que tindrà la precisió d'un ordinador i l'adaptabilitat dels éssers humans més intel·ligents. Els cotxes que s'auto-condueixen, el reconeixement de veu i el reconeixement facial són només el començament de l'aprenentatge automàtic, que portarà a grans noves aplicacions que actualment ens són difícils d'imaginar.

En aquest context cal destacar també el rol tan important que té actualment el *Machine Learning* en el món del Big Data. El terme **Big Data** fa referència exactament al que el seu nom indica: a la recollida, emmagatzematge, tractament i anàlisi de bases de dades tan grans que resulta impossible tractar-les amb les eines convencionals. Aquest és un camp d'actualitat ja que cada any es registra un creixement exponencial en la quantitat de dades que es generen, degut a la proliferació de pàgines web, a les xarxes socials, els dispositius mòbils, etc. Aquests conjunts de dades són grans, diversos i estan en constant evolució, un marc on els mètodes d'aprenentatge automàtic són particularment eficaços, superant fàcilment als mètodes tradicionals en precisió, escala i velocitat. Últimament, però, ja no cal només que l'aprenentatge automàtic proporcioni aquestes eines sinó que cal també que ho faci d'una manera "fàcil", en el sentit que no només ho puguin utilitzar *data scientists* sinó que sigui assolible per a qualsevol persona que vulgui desenvolupar el seu negoci.

Un fenomen que està en auge actualment és el de les **competicions** de *Machine Learning*. Un exemple d'aquestes es troba en la comunitat de Kaggle. **Kaggle** és una plataforma en la que empreses i investigadors poden publicar els seus problemes científics juntament amb les dades per a que *data scientists* de tot el món competeixin per a produir els millors models. Kaggle té com a objectiu convertir "la ciència de dades" en un esport. El funcionament de la plataforma Kaggle és molt senzill, només cal que et registris per a poder competir en qualsevol dels problemes publicats. Et descarregues les dades d'exemple, construeixes un model utilitzant qualsevol mètode que prefereixis i puges les teves prediccions a Kaggle. Aquest dóna una puntuació a la teva solució en temps real i pots veure el teu lloc a la taula de classificació. Cada problema està obert durant un cert temps i té una certa recompensació econòmica per al primer classificat. Algunes de les empreses més importants a nivell mundial que participen a la xarxa Kaggle són Microsoft, General Electric o la NASA, fet que il·lustra la gran dificultat dels problemes que es publiquen així com el grau de professionalitat i qualificació dels participants en les competicions. Es pot trobar molta més informació a la pàgina web `www.kaggle.com`.

3.1.4 Tipus de tasques

L'enfocament clàssic diferencia les tasques d'aprenentatge supervisat de les tasques d'aprenentatge no supervisat. Actualment, però, existeixen d'altres tipus, com l'aprenentatge per reforç o el semi-supervisat. La taxonomia en base a la qual s'organitzen els algorismes d'aprenentatge automàtic es basa en el resultat desitjat de l'algorisme o en el tipus d'informació d'entrada disponible durant l'aprenentatge de la màquina. A continuació es comenten les idees principals d'aquestes quatre grans àrees diferents d'aprenentatge automàtic.

1. Aprenentatge supervisat

En el context de l'aprenentatge supervisat la situació general és disposar d'un conjunt de predictors $X_1, X_2, \ldots, X_p$ mesurats en n observacions, i una resposta Y associada a cada observació. El que es vol és ajustar un model que relacioni la resposta amb els predictors, amb l'objectiu de predir amb precisió la resposta per a futures observacions o bé millorar la comprensió de la relació entre la resposta i els predictors.

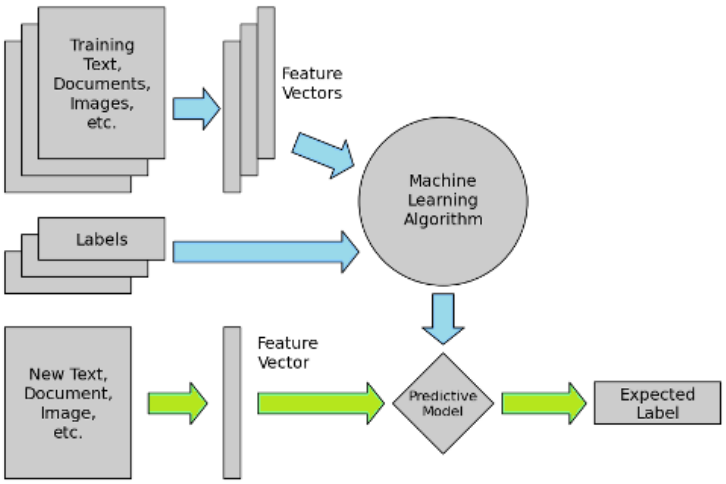
Així doncs, en el context de l'aprenentatge supervisat, es té l'*input space* $\mathcal{X}$ o espai de les observacions (vectors numèrics, text, imatges, etc.) i l'*output space* $\mathcal{Y}$ o espai de les etiquetes. Dins de l'aprenentatge supervisat es poden diferenciar dos grans tipus de problemes en funció de com sigui l'*output space* $\mathcal{Y}$:

- **Classificació**, en cas que la resposta Y sigui una variable qualitativa. Hi ha diversos tipus de classificació:
  - Binària: La resposta té dues categories ($\mathcal{Y} = \{-1, 1\}$).
  - Multiclasse: La resposta té més de dos modalitats ($\mathcal{Y} = \{1, 2, \ldots, K\}$).
  - Multi-label: Es tenen múltiples respostes, generalment binàries ($\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_L$).
- **Regressió**, en cas que la resposta Y sigui una variable quantitativa. Aquesta regressió pot ser lineal o polinòmica, entre d'altres.

Pel que fa als problemes de regressió, algunes tècniques a emprar dins d'aquest àmbit serien els models de regressió lineal o els polinòmics. I per al cas dels problemes de classificació es tenen tècniques de molts tipus, algunes de les quals poden ser la regressió logística o l'anàlisi de discriminació lineal. Tot i això, aquesta diferenciació entre problemes de regressió i de classificació és molt general i massa simple molts cops, ja que hi ha tècniques que es poden utilitzar en els dos casos. Algunes de les tècniques més utilitzades dins de l'aprenentatge supervisat i que es poden adaptar tant a regressió com a classificació són els arbres de decisió, les xarxes neuronals i les màquines de vectors suport.

La forma habitual de treballar en l'aprenentatge supervisat és separar les dades disponibles en dos conjunts: el *training set* i el *test set*. L'algorisme és construeix a partir dels exemples del conjunt *training*, i s'avalua el rendiment del model trobat amb el conjunt *test*. Aquest procés es resumeix de forma esquemàtica en la següent figura [1]:

Figura 3.4: Model d'aprenentatge supervisat



Un algorisme és un procediment que pren com a entrada les dades d'exemple i produeix com a resultat un classificador. Aquest classificador estima la relació entre l'espai les observacions $\mathcal{X}$ i l'espai de les etiquetes $\mathcal{Y}$. Ara bé, existeixen milers d'algorismes d'aprenentatge supervisat. La clau per saber quin escollir en cada situació concreta és adonar-se que aquests consisteixen en la combinació dels següents tres components:

Representació (Espai d'hipòtesis) Cada algorisme escull els possibles models d'un espai particular de funcions $\mathscr{F}$, anomenat espai d'hipòtesis. A vegades aquest es coneix explícitament i altres cops és implícit segons el mecanisme de l'algorisme. Si un classificador no està en l'espai d'hipòtesis, no pot ser après. Així doncs, el primer component és un conjunt de models possibles on buscar a fons.

Avaluació (Funció objectiu o de puntuació) Es necessita una funció d'avaluació per tal de distingir els bons classificadors dels dolents, és a dir, una manera d'avaluar si un model és bo.

Optimització Es necessita un mètode per buscar entre tots els classificadors aquell que tingui major puntuació (un valor de la funció d'avaluació major). L'elecció de la tècnica d'optimització és clau per a l'eficiència de l'algorisme, pel que cal buscar una manera intel·ligent de trobar un model realment bo amb poques proves.

[1] Imatge extreta de `http://www.astroml.org/sklearn_tutorial/general_concepts.html` [Consulta: 05/06/14]

La següent taula ([7]) mostra exemples habituals de cadascun d'aquests tres components. Combinant els elements de les tres columnes es formen diferents algorismes, tot i que per descomptat no totes les combinacions de components tenen el mateix sentit.

Taula 3.2: Els tres components dels algorismes d'aprenentatge

Representation	**Evaluation**	**Optimitzation**
Instances	Accuracy / Error rate	Combinatorial optimitzation
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimitzation
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Set of rules	Cost / Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional Random Fields		

2. Aprenentatge no supervisat

En el context de l'aprenentatge no supervisat la situació general és tenir només un conjunt de variables $X_1, X_2, \ldots, X_p$ mesurades en n observacions, i cap variable resposta, al contrari que en l'aprenentatge supervisat. No s'està interessat ni en predir ni en classificar ja que no es té una variable resposta Y associada. L'objectiu és, doncs, intentar descobrir característiques interessants sobre les variables ja sigui visualitzant les dades de forma informativa o intentant descobrir subgrups d'individus o de variables.

En comparació amb l'aprenentatge supervisat, el no supervisat ofereix normalment més reptes, ja que és més subjectiu, donat que no es té un objectiu definit per a l'anàlisi, com sí en el cas de la classificació o de la predicció. Molts cops les tècniques d'aprenentatge no supervisat es duen a terme com a part de l'anàlisi exploratori de dades. Una altra dificultat que comporta aquest tipus d'aprenentatge és l'avaluació dels resultats obtinguts. En el cas d'un model de predicció és fàcil comprovar com de bé (o de malament) el model prediu la resposta Y, però en el cas de l'aprenentatge no supervisat això no es pot realitzar donat que la "resposta vertadera" no es coneix.

Les tècniques d'aprenentatge no supervisat cada cop són més importants en una gran quantitat de camps diferents. En el context de la biologia, per exemple, un investigador pot disposar de l'expressió dels gens en pacients amb càncer de mama, i voldrà descobrir subgrups de gens o de mostres de càncer per a entendre millor la malaltia.

Algunes de les tècniques d'aprenentatge no supervisat més populars són les de reducció de la dimensió i de *clustering*. Ambdues volen simplificar les dades mitjançant l'ús d'un petit nombre de característiques, però això ho fan diferent:

Clustering El *clustering* es refereix a un conjunt molt ampli de tècniques que tenen com a objectiu principal descobrir patrons i subgrups en les dades. Quan s'agrupen les observacions, el que es busca és dividir-les en grups de manera que les observacions dins de cada grup siguin molt similars entre si, mentre que les observacions entre els diferents grups siguin molt heterogènies unes de les altres.

Reducció de la dimensió Fa referència al procés de reduir el nombre de variables en les dades en base a alguna consideració, com per exemple, maximitzar la quantitat de variació explicada. Aquest és el cas de la tècnica principal en aquest context, l'anàlisi de components principals. El PCA busca trobar una representació de les observacions en poques dimensions però que expliquin bona part de la variància de les dades. A l'apartat 4.2.1 es descriu aquesta tècnica més detalladament.

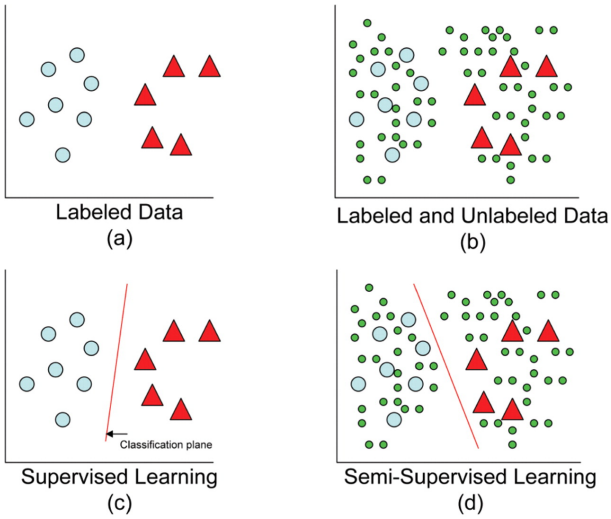
3. Aprenentatge semi-supervisat

Molts dels problemes cauen naturalment en els paradigmes d'aprenentatge supervisat o no supervisat. No obstant, de vegades la qüestió de si l'anàlisi ha de ser considerat amb o sense supervisió és menys clara. Per exemple, suposem que es té un conjunt de n observacions, de forma que per a m de les observacions ($m < n$) es disposa de la resposta però per a les restants $n - m$ no. Aquest escenari pot sorgir si els predictors es poden mesurar de forma relativament senzilla però en canvi les respostes corresponents són molt més difícils de recollir. Aquesta és la situació d'un problema d'aprenentatge semi-supervisat. En aquest context, es vol utilitzar un mètode d'aprenentatge que pugui incorporar les m observacions per a les quals es té la resposta així com les $n - m$ observacions per a les quals la resposta no està disponible. L'objectiu és que el resultat superi el rendiment que es pugui obtenir ja sigui descartant les dades no etiquetades i usant l'aprenentatge supervisat o bé descartant les etiquetes i usant l'aprenentatge no supervisat.

En la figura 3.5 [2] es mostra un exemple de la influència de les dades no etiquetades en l'aprenentatge semi-supervisat, concretament en un problema de classificació binària. En la figura (a) es mostren les dues classes que han de ser classificades a partir només de dades etiquetades, mentre que en (b) es mostra el mateix però juntament amb dades no etiquetades (punts verds). Tal i com es veu en (c), una tècnica d'aprenentatge supervisat pot triar una frontera de decisió incorrecta, mentre que en (d), al tenir en compte les dades extra, el pla separador porta a una millor classificació.

[2] Imatge extreta de `http://bioinformatics.oxfordjournals.org/content/24/6/783.full` [Consulta: 18/06/14]

Figura 3.5: Influència de les dades no etiquetades en l'aprenentatge semi-supervisat



A la pràctica es disposa d'una gran quantitat de dades sense resposta (sense etiqueta) i de moltes menys dades etiquetades. Les instàncies etiquetades són sovint difícils o cares d'obtenir o necessiten molt de temps, ja que requereixen d'un agent humà qualificat (per exemple, per transcriure un segment d'àudio) o d'un experiment físic (per exemple, la determinació de l'estructura 3D d'una proteïna). En canvi, l'adquisició de dades sense etiqueta és un procés relativament barat, ja que es poden trobar disponibles en una gran quantitat de llocs (text en llocs web, imatges, etc). És per això que l'aprenentatge semi-supervisat s'ha tornat recentment més popular i pràcticament imprescindible.

Existeixen bastants mètodes d'aprenentatge semi-supervisat, alguns dels més destacats són els models generatius, els mètodes basats en grafs, o els mètodes de co-entrenament. Ara bé, aquest és un camp que comporta molts reptes. Per exemple, l'ús de dades no etiquetades no sempre millora la precisió de l'algorisme (Cozman et al., 2003); també passa que atès que la quantitat de dades etiquetades és escassa, els mètodes d'aprenentatge semi-supervisat necessiten que es compleixin certs supòsits forts. Si es vol conèixer molt més sobre l'aprenentatge semi-supervisat es pot consultar [3].

4. Aprenentatge per reforç

L'aprenentatge per reforç és una de les àrees d'investigació més actives en el camp de la Intel·ligència Artificial. En l'aprenentatge per reforç s'aprèn què fer - com passar de les situacions a les accions - per tal de maximitzar un senyal numèrica de recompensa. Els dos trets distintius més importants de l'aprenentatge per reforç són:

Recerca en forma d'assaig i error No es diu quines són les accions a prendre sinó que s'ha de descobrir quines accions produeixen la màxima recompensa provant-les.

Recompensa amb retard Les accions poden afectar no només a la recompensa immediata sinó també a la següent situació i, en conseqüència, a totes les recompenses posteriors.

Els algorismes d'aprenentatge per reforç retenen selectivament els resultats que maximitzen la recompensa rebuda a través del temps. Per a acumular una gran quantitat de premis, el sistema d'aprenentatge ha de preferir les millors accions experimentades; tot i que també ha d'intentar noves accions per tal de descobrir millors accions per al futur. Així doncs, sempre hi ha un *trade-off* entre:

Explotació Utilitzar el coneixement que es té actualment per obtenir una recompensa.

Exploració Descobrir noves accions per poder fer millors seleccions en el futur.

L'aprenentatge per reforç té 3 components bàsics:

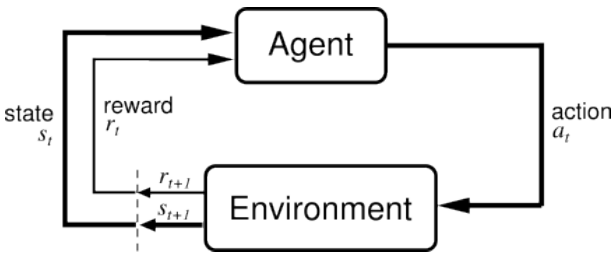
Agent Aquell que ha d'aprendre i prendre les decisions.

Entorn Tot el que interactua amb l'agent, és a dir, tot allò exterior a l'agent.

Accions Allò que l'agent pot fer. Cada acció està associada amb una recompensa. L'objectiu és que l'agent triï les accions que maximitzin la recompensa esperada sobre un cert període de temps.

En el model estàndard d'aprenentatge per reforç, un agent està connectat al seu entorn a través de la percepció i l'acció, tal i com es representa a la figura 3.6 [43]. A cada pas de la interacció l'agent rep com a entrada alguna indicació de l'estat actual (s) de l'entorn, i tria llavors una acció (a) com a resultat. L'acció canvia l'estat de l'entorn, i el valor d'aquesta transició d'estat es comunica a l'agent a través d'un senyal de reforç escalar (r). El comportament de l'agent ha de triar les accions que tendeixen a augmentar la suma a llarg termini dels valors de la senyal de reforç, guiat per una gran varietat d'algorismes. Alguns d'aquests algorismes són el *Q-learning*, el *Temporal difference learning* i el *Sarsa*.

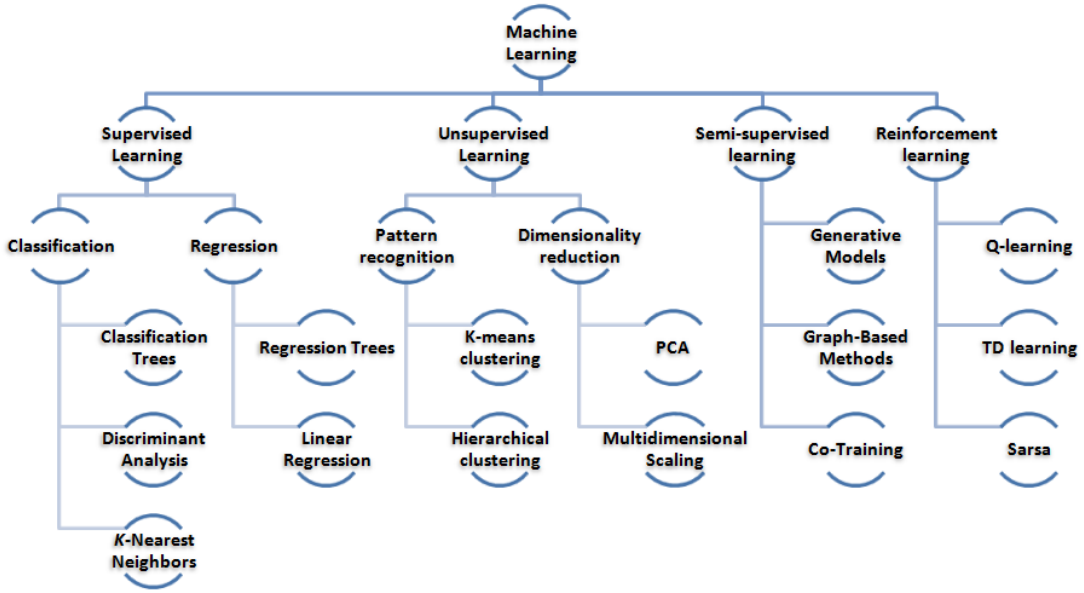
Figura 3.6: Esquema del model estàndard d'aprenentatge per reforç



Les aplicacions de l'aprenentatge per reforç s'han estès de la robòtica a la fabricació industrial, i als problemes de recerca combinatòria tals com els jocs d'ordinador. Es pot trobar informació més detallada sobre aquest camp d'estudi en [18] i [43].

En la següent figura es resumeixen els tipus d'algorismes i de tècniques vistos en aquest subapartat.

Figura 3.7: Esquema de diferents algorismes de *Machine Learning*



Conceptes clau a tenir en compte quan es fa *Machine Learning*

Tal i com s'ha vist, els algorismes de *Machine Learning* intenten realitzar tasques complicades generalitzant a partir d'exemples, en comptes de programant "a mà". I tal com mostren les aplicacions comentades, el seu ús s'està estenent a una gran varietat de camps d'estudi. Ara bé, per a desenvolupar algorismes que realment funcionin cal tenir en compte certes qüestions, descrites per Pedro Domingos a [7], i algunes de les quals es resumeixen a continuació. Aquestes es centren en els problemes de classificació, tot i que es poden estendre a qualsevol altra tasca de l'aprenentatge automàtic.

Generalització L'objectiu fonamental de l'aprenentatge automàtic és aconseguir generalitzar els resultats més enllà dels exemples del conjunt *training*, ja que es vol que el model funcioni per a qualsevol conjunt de dades i no solament per al testejat. Així doncs, cal remarcar la necessitat de reservar una part de les dades com a dades test, i no tocar-les mai per a construir el model. També es poden utilitzar altres mètodes com la cross-validació o el *random splitting*.

Sobreajustament Un dels grans problemes en l'aprenentatge automàtic és el del sobreajustament (*overfitting*), que es dóna quan s'obté un classificador que és fiable, per exemple, al 100% en les dades d'entrenament però només al 50% en les dades de test. Algunes opcions per combatre el sobreajustament són l'ús de la cross-validació o l'addició d'un terme de regularització a la funció d'avaluació.

Dimensionalitat Després del sobreajustament, l'altre gran problema en l'aprenentatge automàtic és el conegut com la maledicció de la dimensionalitat (*the curse of dimensionality*). Construir un classificador en dues o tres dimensions és fàcil: es pot trobar una frontera raonable entre exemples de diferents classes només amb inspecció visual. Però en altes dimensions és difícil entendre el que està succeint. Això demostra la importància de les tècniques de reducció de la dimensió en el camp de l'aprenentatge automàtic.

La clau és el *feature engineering* El concepte de *feature engineering* significa que utilitzes el teu coneixement sobre les dades per a crear noves variables a partir de les originals que fan que l'algorisme funcioni millor. I a aquesta tasca és a la que s'ha de dedicar més esforç i recursos per a que el model sigui bo. No hi ha una única manera de fer-ho, però és segur que s'aconseguiran transformacions més interessants si es té un coneixement detallat de les dades específiques. Per tant, el treball conjunt amb alguna persona experta en el camp d'on provinguin les dades sempre és preferible.

Quantes més dades, millor Hi ha cada vegada més evidències que, en molts problemes, tècniques d'aprenentatge automàtic molt simples es poden convertir en classificadors increïblement potents només amb l'addició d'un munt de dades. Quants més exemples es tinguin i quanta major sigui la seva qualitat, millors models es trobaran.

És millor aprendre molts models, no només un Aquí Domingos es refereix al poder de tècniques com el *bagging*, el *boosting* o el *Random Forest*, en les quals en comptes d'aprendre un sol model, es troba un de més potent aprenent diversos classificadors sobre diferents mostres aleatòries de les dades.

Un model simple no té per què ser més precís Davant de dos models que s'ajusten a les dades igualment bé, molts algorismes d'aprenentatge automàtic tenen una manera matemàtica de preferir el més simple dels dos. Es pot creure que el motiu és que amb un model més simple és menys probable tenir sobreajustament, però en *machine learning* això no és tan clar: hi ha molts llocs on la complexitat addicional pot beneficiar el rendiment. Així doncs, si prefereixes un model simple que sigui perquè és més petit o més ràpid d'ajustar o més interpretable, però sabent que no necessàriament donarà lloc a un millor rendiment.

Correlació no implica causalitat Aquest és un tema ja après en Estadística, però que cal ressaltar. A l'hora d'interpretar els models, el fet que una cosa predigui una altra no vol dir que sigui causa d'aquesta, i prendre decisions en base a aquestes relacions s'ha de fer amb molta cura.

3.2 Mètodes kernel

Els mètodes kernel proporcionen extensions no lineals d'un gran nombre de tècniques lineals, mitjançant la projecció de les dades en un espai de característiques d'alta dimensió, cosa que augmenta la potència dels algorismes lineals. Cal que els algorismes lineals puguin expressar-se únicament en termes de productes escalars per poder aplicar l'extensió coneguda com el *kernel trick*.

Els primers mètodes que van utilitzar kernels daten dels anys 50. Posteriorment, a meitat dels 60, els kernels es van utilitzar d'una forma semblant al que ara es coneix com el *kernel trick*. No va ser fins als 90, quan les propietats dels kernels es van usar per a desenvolupar la tècnica de les màquines de vectors suport, una generalització de l'algorisme de l'òptim hiperplà. En aquells moments només s'usava amb dades numèriques, però uns anys més tard, cap al 2000, es van començar a utilitzar també en dades no vectorials, definint kernels més complexos. En els últims anys s'han dut a terme múltiples "*kernelitzacions*" de diversos algorismes.

En un primer subapartat es presenta la idea que hi ha darrere de l'ús de funcions kernel, per després donar pas a una definició formal tant de la funció kernel com de la matriu kernel. Posteriorment es detallen alguns dels kernels més habituals i s'explica com es poden utilitzar amb el software R. En últim lloc s'introdueixen els mètodes kernels de forma més detallada: què són, quins són i què els fa tan atractius.

3.2.1 Funcions kernel

Els kernels o funcions kernel són l'element bàsic que comparteixen tots els mètodes kernel. Proporcionen un marc general per a representar les dades i han de satisfer certes condicions matemàtiques.

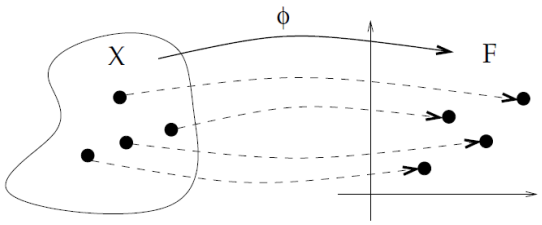
Idea general

Suposem que es té un conjunt de n individus que es volen analitzar ($S = (\mathbf{x_1}, ..., \mathbf{x_n})$), on cada objecte $\mathbf{x_i}$ és un element del conjunt $\mathcal{X}$, anomenat espai d'entrada o *input space*. En comptes d'aplicar les tècniques en aquest espai, el que es realitza és una transformació de les dades a un nou espai, anomenat espai de característiques o *feature space* $\mathcal{F}$, i que usualment és de dimensió major que $\mathcal{X}$ (i fins i tot pot ser infinita). Aquesta transformació es defineix per l'aplicació ϕ:

$$\begin{aligned} \phi : \mathcal{X} &\longrightarrow \mathcal{F} \\ \mathbf{x} &\longrightarrow \phi(\mathbf{x}) \end{aligned}$$

on $\phi(\mathbf{x})$ representa un *mapeig* no lineal. Això fa que a l'espai de característiques $\mathcal{F}$ es puguin aplicar tècniques lineals a les dades transformades $\phi(\mathbf{x})$ que en l'espai d'entrada $\mathcal{X}$ equivalen a tècniques no lineals.

Figura 3.8: Representació de les dades amb kernels [3]



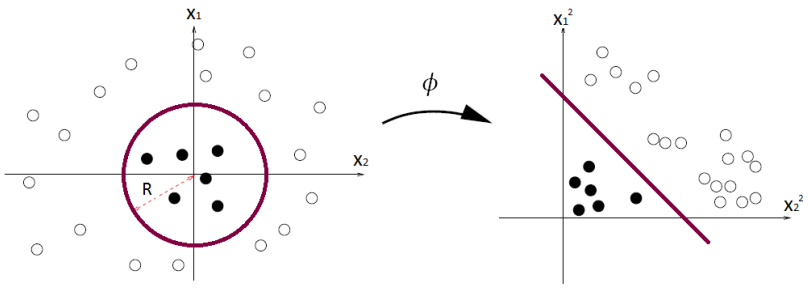
Ara bé, calcular les projeccions $\phi(\mathbf{x})$ en el nou espai pot resultar molt costós, sobretot si la dimensionalitat d'aquest és molt gran. És per això que en comptes de calcular aquestes projeccions el que s'utilitza és una funció per a calcular amb les dades de l'espai d'entrada els corresponents productes escalars en l'espai de característiques. Aquesta funció s'anomena funció kernel, i es defineix com:

$$k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$$
$$k(\mathbf{x}, \mathbf{x}') \longrightarrow \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

D'aquesta manera ja no cal calcular les coordenades de les dades explícitament, i, de fet, tampoc cal conèixer la funció de transformació. Aquesta transformació $\phi(\mathbf{x})$ queda definida de forma implícita per la funció kernel seleccionada. Com que no es tenen les coordenades dels punts en el nou espai, sinó que es tenen els seus productes escalars, les tècniques lineals que es poden aplicar en el *feature space* són aquelles en les quals les dades intervenen només com a productes escalars.

Un dels motius pels quals l'ús de kernels és tan atractiu s'exemplifica en la següent figura:

Figura 3.9: Motivació de l'ús de kernels [3]



[3]Imatges extretes de [46].

En l'espai original $\mathcal{X}$ els punts no són linealment separables, mentre que a l'espai de característiques $\mathcal{F}$ es passa a tenir un problema separable linealment. Així doncs, en el nou espai es poden aplicar mètodes lineals que donin resultats no lineals en l'espai original. El cas concret que es mostra en la figura anterior correspon a l'ús d'un kernel polinòmic (un tipus de kernel que es defineix posteriorment); en aquest cas concret la transformació $\phi(\mathbf{x})$ sí és coneix, tot i que no és necessari:

Per a $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$, es defineix $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$ com:

$$k(\mathbf{x}, \mathbf{x}') = x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2 = \langle \mathbf{x}, \mathbf{x}' \rangle^2$$

Definició formal de kernel

Per a que la funció k sigui una funció kernel cal que sigui simètrica i definida positiva. De manera més formal, una funció $k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ és una funció kernel si i només si és:

Simètrica És a dir, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ per a dos objectes $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ qualssevol.

Definida positiva És a dir,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

per a qualsevol $n > 0$, qualssevol $\mathbf{x_1}, ..., \mathbf{x_n} \in \mathcal{X}$ i qualssevol $c_1, ..., c_n \in \mathbb{R}$.

Tot i que el fet que la funció de comparació hagi de complir aquestes propietats limita en certa manera els tipus de funcions que es poden usar, a la vegada permet utilitzar els mètodes kernel, pel que val la pena.

Per tal d'aconseguir veure els kernels de forma una mica més intuïtiva, aquests es poden pensar com a mesures de **similitud**, en el sentit que $k(\mathbf{x}, \mathbf{x}')$ és "gran" quan $\mathbf{x}$ i $\mathbf{x}'$ són "similars". Per exemple, si es considera el següent kernel:

$$k(\mathbf{x}, \mathbf{x}') = exp\left(-\frac{d(\mathbf{x}, \mathbf{x}')^2}{2\sigma^2}\right)$$

on σ és un paràmetre i d és la distància euclídea, es pot veure que aquest és una funció decreixent de la distància euclidiana entre punts, de forma que quant més gran sigui $k(\mathbf{x}, \mathbf{x}')$, més a prop estan els punts $\mathbf{x}$ i $\mathbf{x}'$ en l'espai original. L'ús de funcions kernel implica una representació de les dades diferent. Ja no es defineix un individu pel seu vector de variables, sinó que es defineix en termes de com és de similar o de diferent de la resta d'individus.

Matriu kernel

La matriu kernel $\mathbf{K}$ és l'estructura central en els mètodes kernel, i conté tota la informació necessària per a aplicar les tècniques d'aprenentatge. Aquesta matriu conté els productes escalars en $\mathcal{F}$ de tots els individus, i es defineix com:

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} \langle\phi(\mathbf{x}_1),\phi(\mathbf{x}_1)\rangle & \langle\phi(\mathbf{x}_1),\phi(\mathbf{x}_2)\rangle & \cdots & \langle\phi(\mathbf{x}_1),\phi(\mathbf{x}_n)\rangle \\ \langle\phi(\mathbf{x}_2),\phi(\mathbf{x}_1)\rangle & \langle\phi(\mathbf{x}_2),\phi(\mathbf{x}_2)\rangle & \cdots & \langle\phi(\mathbf{x}_2),\phi(\mathbf{x}_n)\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle\phi(\mathbf{x}_n),\phi(\mathbf{x}_1)\rangle & \langle\phi(\mathbf{x}_n),\phi(\mathbf{x}_2)\rangle & \cdots & \langle\phi(\mathbf{x}_n),\phi(\mathbf{x}_n)\rangle \end{pmatrix}$$

D'aquesta forma de representar les dades es deriven alguns comentaris:

- Siguin del tipus que siguin les dades, sempre es representaran com una **matriu quadrada de valors reals**, tant si són vectors, com si són seqüències de caràcters com si són imatges o grafs. Això suggereix que un algorisme desenvolupat per a tractar aquest tipus de matriu pot analitzar qualsevol tipus de dades de la mateixa forma sempre que es pugui definir la funció k. Això és de vital importància en el món de la bioinformàtica on sovint cal treballar amb ADN o proteïnes, que es representen per seqüències de caràcters.

- La **mida de la matriu** utilitzada per a representar els n objectes sempre és $\mathbf{n} \times \mathbf{n}$, independentment de la naturalesa o de la complexitat de les dades. Per exemple, si es tenen les expressions de 10000 gens per a 21 individus, aquests es representaran amb una matriu de 21×21, cosa que en termes computacionals és molt atractiu.

- Donada la definició de kernel, la matriu kernel és una matriu simètrica i semidefinida positiva.

- S'evita haver de trobar una representació explícita de cada objecte. Molts algorismes d'anàlisis de dades necessiten que aquestes estiguin representades per vectors de valors reals, però biològicament parlant no hi ha una manera òbvia de representar, per exemple, seqüències de proteïnes com a vectors. Usant funcions kernel no cal trobar $\phi(\mathbf{x})$, i ni tan sols saber com calcular-ho.

3.2.2 Alguns tipus de kernels

A continuació es presenten alguns dels kernels més utilitzats per a dades vectorials i un kernel per a *strings*, tot i que també existeixen kernels per a grafs o per a funcions. Es poden trobar diverses parametritzacions per a un mateix kernel en diferents referències; la utilitzada aquí es correspon amb la que té implementada el software R i que es pot consultar per a més detalls a [19] i [20].

Kernels per a vectors

Kernel lineal El kernel lineal, ja mencionat anteriorment, es defineix com el producte escalar de dos vectors. És el primer i el més simple dels kernels.

$$k_L(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'^T = \langle \mathbf{x}, \mathbf{x}' \rangle$$

Aquest es pot pensar com el kernel corresponent a la identitat, pel que si s'utilitza s'estarà fent una tècnica lineal. Així doncs, els kernels també proporcionen un marc on realitzar les tècniques lineals clàssiques.

Kernel polinòmic El kernel polinòmic es pot expressar d'una forma molt general com:

$$k_P(\mathbf{x}, \mathbf{x}') = (\alpha \cdot \langle \mathbf{x}, \mathbf{x}' \rangle + c)^d,$$

on d és el grau del polinomi ($d > 0$), α és un paràmetre d'escala ($\alpha > 0$) i c és l'*offset* ($c \geq 0$). Si el grau i l'escala són 1 i l'*offset* és 0, el kernel polinòmic es redueix al kernel lineal. En l'exemple de la figura 3.9 s'ha aplicat un kernel polinòmic de grau $d = 2$, amb $\alpha = 1$ i $c = 0$. El kernel polinomial es sol utilitzar en problemes de classificació d'imatges.

Kernel gaussià El kernel gaussià (*Gaussian radial basis function (RBF) kernel*) es defineix com:

$$k_G(\mathbf{x}, \mathbf{x}') = exp\left(-\sigma \parallel \mathbf{x} - \mathbf{x}' \parallel^2\right),$$

on σ és un paràmetre a fixar per l'usuari. Aquest és un dels kernels més utilitzats a la pràctica, ja que permet generar funcions de classificació no paramètriques i és el que s'usa habitualment quan no es té coneixement previ sobre les dades.

En funció del kernel triat, els valors de la matriu kernel seran d'una forma o d'una altra. Per exemple, la matriu del kernel gaussià és una matriu normalitzada, acotada: si la similitud entre dos individus és nul·la, el valor de la matriu serà un 0, mentre que si és màxima serà 1. Així doncs, els valors de la diagonal sempre seran 1, cosa que no passa amb el kernel lineal o el polinòmic. En altres paraules, el rang de valors de la matriu del kernel gaussià sempre estarà entre 0 i 1 siguin quines siguin les dades, mentre que per als altres dos kernels aquest rang dependrà de cada conjunt de dades en particular.

Aquestes funcions kernels són tres exemples de la diversitat de funcions existents per a vectors. Altres són el kernel sigmoidal, el kernel Laplace o el kernel ANOVA, tots inclosos en el paquet *kernlab* de R. Cada funció kernel té la seva qualitat, i la tria d'una o altra depèn molt de les dades a analitzar, així com dels objectius de l'anàlisi: no és el mateix voler classificar que voler trobar clústers. No hi ha cap regla general, sinó que més aviat

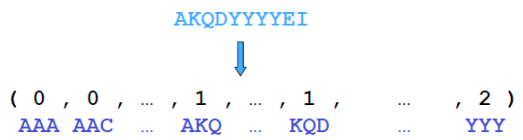
s'ha d'anar provant fins a trobar el millor, tenint molt clar com són les dades a tractar. I el mateix passa en la tria dels paràmetres de cada kernel en particular: no hi ha una manera concreta d'optimitzar aquests paràmetres, sinó que cal anar provant, i aquest ajust sempre dependrà de les dades que es tinguin.

Kernels per a strings

En el camp de la biologia és molt habitual trobar-se amb *strings*, com per exemple les cadenes de nucleòtids o les cadenes peptídiques. Un dels problemes centrals en la biologia computacional és la classificació de les seqüències de proteïnes en famílies funcionals i estructurals en base a l'homologia de les seqüències. Aquest fet fa que recentment s'hagi parat molt atenció a com afrontar el problema dels kernels per a cadenes, ja que amb el seu ús es poden estendre els mètodes de classificació i discriminació dissenyats per a treballar amb vectors numèrics, com ara el *Support Vector Machines*, i que han demostrat la seva eficàcia, per a treballar amb cadenes de caràcters com a dades.

Els *string kernels* es poden entendre de forma intuïtiva com funcions que mesuren la similitud entre parells de cadenes: quant més semblants siguin dues cadenes a i b més alt serà el valor per al kernel string $k(a, b)$. Una manera de mesurar aquesta similaritat és comptant quantes subcadenes comunes d'una certa longitud tenen les dues cadenes. Els diferents kernels que hi ha es diferencien per com valoren les coincidències entre les cadenes (alguns compten les coincidències exactes, altres permeten espais, etc). Un cop es tenen aquests comptatges, d'aquests es deriva un vector al *feature space* i el kernel es defineix com el producte escalar entre aquests vectors.

Figura 3.10: *String kernel:* pas de la cadena de caràcters en $\mathcal{X}$ al vector en $\mathcal{F}$



Com a exemple, a continuació es presenta un dels kernels per a cadenes més simple, el *spectrum kernel*. Es vol representar cada seqüència $\mathbf{x} \in \mathcal{X}$ per un vector numèric de longitud fixa $\phi(\mathbf{x}) \in \mathbb{R}^n$. L'enfocament es basa en indexar el *feature space* per cadenes de longitud fixa, és a dir:

$$\phi(\mathbf{x}) = (\phi_u(\mathbf{x}))_{u \in A^k}$$

on:

- u = cada subseqüència de longitud k ("*k-mer*").
- A = alfabet.
- $\phi_u(\mathbf{x})$ = número de cops que u es dóna en la seqüència $\mathbf{x}$ (sense espais).

El *k-spectrum* de $\mathbf{x}$, on $k \geq 1$, és el conjunt de totes les subseqüències contigües de longitud k que conté $\mathbf{x}$. I el *k-spectrum kernel* és:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{u \in A^k} \phi_u(\mathbf{x})\phi_u(\mathbf{x}')$$

En l'apartat 3.2.3 es detalla un exemple de càlcul d'aquest kernel pas a pas per entendre millor el seu ús.

Combinació de kernels

Els kernels tenen la propietat de la modularitat, és a dir, que és possible construir nous kernels mitjançant la combinació de kernels simples. Per exemple, si $k_1(\mathbf{x}, \mathbf{x}')$ i $k_2(\mathbf{x}, \mathbf{x}')$ són kernels vàlids i a i b són constants, aleshores els següents kernels també són vàlids.

$$k(\mathbf{x}, \mathbf{x}') = \begin{cases} ak_1(\mathbf{x}, \mathbf{x}') \\ k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\ ak_1(\mathbf{x}, \mathbf{x}') + bk_2(\mathbf{x}, \mathbf{x}') \\ k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}') \end{cases}$$

Es poden utilitzar kernels diferents per a diferents subconjunts de $\mathcal{X}$. Per tant, el fet de combinar kernels es pot veure com una manera de fusionar la informació de diferents fonts de dades, on cada kernel mesura la similitud de cada font. Aquest és un ús molt potent de les funcions kernel, ja que la integració de fonts de dades diferents és un dels reptes actuals en molts camps, com per exemple el de la bioinformàtica.

3.2.3 Kernels amb R

Una manera de treballar amb kernels i d'utilitzar algorismes basats en kernels és amb el paquet *kernlab* de R. Aquest paquet conté diversos kernels, així com els algorismes de les màquines de vectors suport, del Kernel PCA i del Kernel CCA, entre moltes d'altres característiques. En aquest apartat es mostra com usar diferents kernels i calcular la matriu kernel associada a unes dades concretes, tant numèriques com *strings*.

Kernels per a vectors

Es parteix d'unes dades molt simples: una matriu amb només 5 individus i 2 columnes. És a dir, es vol estudiar un conjunt S de $n = 5$ individus on $S = (\mathbf{x_1}, ..., \mathbf{x_5})$ de forma que cada $\mathbf{x_i}$ és un element del conjunt $\mathcal{X} = \mathbb{R}^2$. Així doncs, cada individu està representat per un vector de dues variables numèriques. Aquesta matriu és:

```
> set.seed(29)
> x <- round(rnorm(5), 2)
> y <- round(rnorm(5), 2)
> xy = cbind(x, y)
> xy
```

```
         x     y
[1,] -1.28  2.11
[2,] -1.26 -0.52
[3,]  0.21 -0.93
[4,]  0.95  0.42
[5,] -1.18  1.11
```

Donat que la matriu original té 5 registres, la matriu kernel serà una matriu de mida 5×5, i donades les propietats dels kernels, serà una matriu simètrica i semidefinida positiva. Per a calcular-la, s'utilitza la funció `kernelMatrix`. Aquesta funció calcula la matriu $\mathbf{K}$ on $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ i $\mathbf{x_i}$ és un vector fila per al kernel que s'especifiqui.

Kernel lineal En primer lloc, es calcula la matriu kernel per al kernel lineal. Aquest kernel està implementat en la funció `vanilladot`, i no cal especificar el valor de cap paràmetre.

```
> val = vanilladot()
> val
```

```
Linear (vanilla) kernel function.
```

La matriu kernel és:

```
> kernelMatrix(val, xy)
```

```
An object of class "kernelMatrix"
        [,1]    [,2]    [,3]    [,4]    [,5]
[1,]  6.0905  0.5156 -2.2311 -0.3298  3.8525
[2,]  0.5156  1.8580  0.2190 -1.4154  0.9096
[3,] -2.2311  0.2190  0.9090 -0.1911 -1.2801
[4,] -0.3298 -1.4154 -0.1911  1.0789 -0.6548
[5,]  3.8525  0.9096 -1.2801 -0.6548  2.6245
```

Així doncs, $K_{11} = k(\mathbf{x_1}, \mathbf{x_1})$ on $\mathbf{x_1} = [-1.28, 2.11]$. I com es tracta del kernel lineal, el valor de K_{11} es correspon amb el producte escalar entre $\mathbf{x_1}$ i $\mathbf{x_1}$.

$$K_{11} = \mathbf{x_1} \cdot \mathbf{x_1}^T = [-1.28, 2.11] \cdot [-1.28, 2.11]^T = (-1.28)^2 + 2.11^2 = 6.0905$$

De la mateixa manera, $K_{12} = k(\mathbf{x_1}, \mathbf{x_2})$ on $\mathbf{x_1} = [-1.28, 2.11]$ i $\mathbf{x_2} = [-1.26, -0.52]$. I el valor de K_{12} és:

$$K_{12} = \mathbf{x_1} \cdot \mathbf{x_2}^T = [-1.28, 2.11] \cdot [-1.26, -0.52]^T = (-1.28) \cdot (-1.26) + 2.11 \cdot (-0.52) = 0.5156$$

Kernel polinòmic En segon lloc es calcula la matriu kernel per al kernel polinòmic. Aquest kernel està implementat en la funció `polydot`, i en aquest cas hi ha paràmetres a especificar: el grau del polinomi d, l'escala α i l'offset c. Els tres per defecte estan fixats a 1. En R, els paràmetres dels kernels s'anomenen *Hyperparameters*.

```
> pol <- polydot(degree = 2)
> pol
```

```
Polynomial kernel function.
 Hyperparameters : degree =  2  scale =  1  offset =  1
```

La matriu kernel és:

```
> kernelMatrix(pol, xy)
```

```
An object of class "kernelMatrix"
          [,1]      [,2]       [,3]      [,4]        [,5]
[1,] 50.275190 2.2970434 1.51560721 0.4491680 23.54675625
[2,]  2.297043 8.1681640 1.48596100 0.1725572  3.64657216
[3,]  1.515607 1.4859610 3.64428100 0.6543192  0.07845601
[4,]  0.449168 0.1725572 0.65431921 4.3218252  0.11916304
[5,] 23.546756 3.6465722 0.07845601 0.1191630 13.13700025
```

Igual que abans, es mostra com obtenir un parell de valors d'aquesta matriu. Sent $\mathbf{x_1}$ i $\mathbf{x_2}$ els mateixos vectors que abans, es té:

$$K_{11} = k(\mathbf{x_1}, \mathbf{x_1}) = (\mathbf{x_1} \cdot \mathbf{x_1}^T + 1)^2 = (6.0905 + 1)^2 = 50.2752$$

$$K_{12} = k(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{x_1} \cdot \mathbf{x_2}^T + 1)^2 = (0.5156 + 1)^2 = 2.2970$$

I si es fixa el grau i l'escala a 1 i l'*offset* a 0, aleshores el resultat és equivalent a utilitzar un kernel lineal.

```
> pol_bis <- polydot(degree = 1, offset=0)
> pol_bis

Polynomial kernel function.
 Hyperparameters : degree =  1  scale =  1  offset =  0

> kernelMatrix(pol_bis, xy)

An object of class "kernelMatrix"
        [,1]    [,2]    [,3]    [,4]    [,5]
[1,]  6.0905  0.5156 -2.2311 -0.3298  3.8525
[2,]  0.5156  1.8580  0.2190 -1.4154  0.9096
[3,] -2.2311  0.2190  0.9090 -0.1911 -1.2801
[4,] -0.3298 -1.4154 -0.1911  1.0789 -0.6548
[5,]  3.8525  0.9096 -1.2801 -0.6548  2.6245
```

Kernel gaussià En últim lloc es calcula la matriu kernel per al kernel gaussià. Aquest kernel està implementat en la funció `rbfdot`, i cal especificar el valor del hiperparàmetre σ.

```
> rbf <- rbfdot(sigma = 2)
> rbf

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  2
```

La matriu kernel és:

```
> kernelMatrix(rbf, xy)

An object of class "kernelMatrix"
             [,1]         [,2]         [,3]         [,4]         [,5]
[1,] 1.000000e+00 9.810916e-07 1.107883e-10 1.584226e-07 1.326555e-01
[2,] 9.810916e-07 1.000000e+00 9.485414e-03 9.777761e-06 4.860568e-03
[3,] 1.107883e-10 9.485414e-03 1.000000e+00 8.736899e-03 5.094240e-06
[4,] 1.584226e-07 9.777761e-06 8.736899e-03 1.000000e+00 4.423474e-05
[5,] 1.326555e-01 4.860568e-03 5.094240e-06 4.423474e-05 1.000000e+00
```

Igual que abans, es mostra com obtenir un parell de valors d'aquesta matriu. Sent $\mathbf{x_1}$ i $\mathbf{x_2}$ els mateixos vectors que abans, es té:

$$K_{11} = k(\mathbf{x_1}, \mathbf{x_1}) = exp(-\sigma \parallel \mathbf{x_1} - \mathbf{x_1} \parallel^2) = exp(0) = 1$$

$$K_{12} = k(\mathbf{x_1}, \mathbf{x_2}) = exp(-\sigma \parallel \mathbf{x_1} - \mathbf{x_2} \parallel^2) = exp(-2 \cdot \parallel [-0.02, 2.63] \parallel^2) =$$

$$K_{12} = exp(-2 \cdot 2.63^2) = exp(-13.8346) = 9.81 \cdot 10^{-7}$$

S'observa com la matriu per al kernel gaussià amb $\sigma = 2$ té valors molt petits. Per a $\sigma = 0.25$, en canvi, aquesta matriu té valors més alts:

```
> rbf_bis <- rbfdot(sigma = 0.25)
> rbf_bis

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.25

> kernelMatrix(rbf_bis, xy)

An object of class "kernelMatrix"
           [,1]      [,2]       [,3]      [,4]      [,5]
[1,] 1.00000000 0.1774041 0.05695892 0.1412463 0.7768562
[2,] 0.17740412 1.0000000 0.55864001 0.2364721 0.5138494
[3,] 0.05695892 0.5586400 1.00000000 0.5529294 0.2179641
[4,] 0.14124631 0.2364721 0.55292937 1.0000000 0.2855752
[5,] 0.77685621 0.5138494 0.21796408 0.2855752 1.0000000
```

Comprovació de les propietats de les matrius kernel Tal i com ja s'ha comentat anteriorment, les matrius kernel són quadrades, de mida $n \times n$, simètriques i semidefinides positives. Només observant les matrius obtingudes es poden comprovar les tres primeres propietats, i per a comprovar que són semidefinides positives es calculen els seus valors propis per veure que tot són positius o iguals a 0.

- Per al kernel lineal, cal tenir en compte que en realitat no hi ha cap transformació de l'espai. És a dir, l'*input space* $\mathcal{X}$ i el *feature space* $\mathcal{F}$ són el mateix. Per això, hi haurà tants valors propis com dimensió tingui l'espai d'entrada, que en aquests cas és $\mathbb{R}^2$. La resta de valors propis seran 0, tal i com es mostra a continuació:

  ```
  > round(eigen(kernelMatrix(val, xy))$values, 4)

  [1] 9.5123 3.0486 0.0000 0.0000 0.0000
  ```

- Per als kernels polinòmic i gaussià, es té:

  ```
  > round(eigen(kernelMatrix(pol, xy))$values, 4)

  [1] 61.9854  8.9246  4.6872  3.3291  0.6201

  > round(eigen(kernelMatrix(rbf, xy))$values, 4)
  ```

```
[1] 1.1327 1.0129 1.0000 0.9871 0.8673
```

Comparació de les matrius kernel S'observa com, a nivell numèric, les matrius per als diferents kernels són bastant diferents. Una matriu kernel és una matriu de similituds entre els individus, i en funció del kernel que s'utilitzi, s'està creant / deformant una similitud diferent. Un exemple anàleg es pot trobar quan es vol mesurar la distància entre individus; es té la distància euclidiana, la distància de Mahalanobis, la distància de Manhattan, etc. Escollir un kernel o un altre dependrà de l'objectiu de l'anàlisi i del tipus de dades a analitzar. A continuació es comenten alguns aspectes que es deriven de les anteriors matrius:

- El kernel lineal i el kernel polinòmic tenen en compte el valor numèric de les dades originals, al contrari que el kernel gaussià, ja que tot ho normalitza respecte a 1. Per això la diagonal de la matriu kernel del kernel gaussià és un vector d'1's: té un màxim acotat, cosa que no passa amb els altres dos kernels. En el cas del kernel gaussià, si es tracta del mateix individu es té un 1 mentre que si són individus molt diferents es tendirà cap al 0.
- Els rangs de valors de les matrius són molt diferents, tal i com mostren els següents resultats:

Taula 3.3: Rangs de valors per a les diferents matrius kernel

Kernel	Hiperparàmetres	Mín.	Q1	Mediana	Mitjana	Q3	Màx.
Lineal	-	-2.231	-0.655	0.219	0.454	0.910	6.090
Polinòmic	$d = 2, \alpha = c = 1$	0.079	0.449	1.516	5.899	3.647	50.280
Gaussià	$\sigma = 2$	0.000	$5.1{\cdot}10^{-6}$	0.005	0.213	0.133	1.000

Figura 3.11: Histogrames dels valors de les matrius kernel

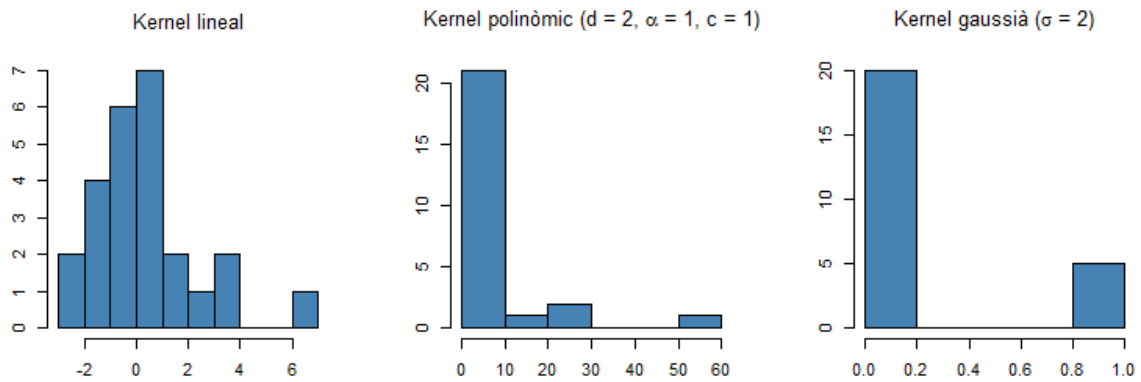
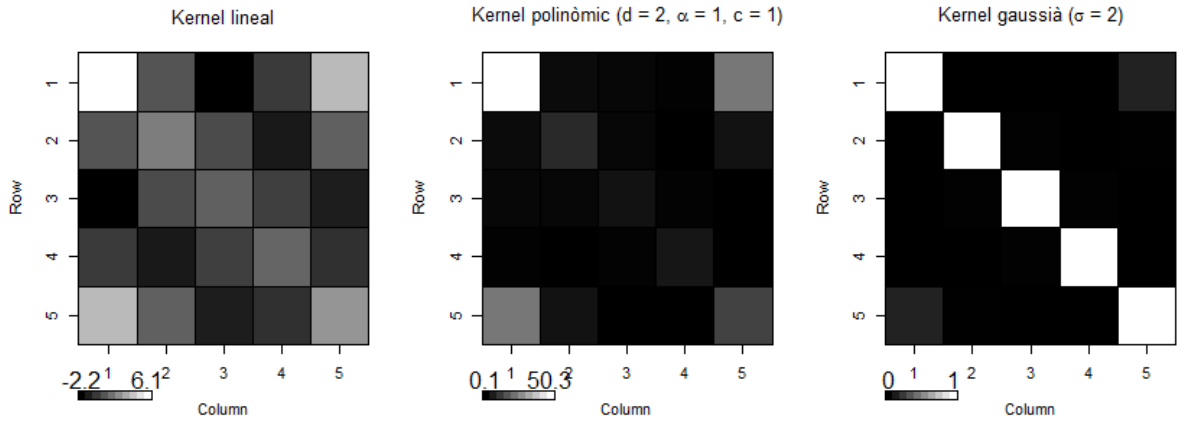


Figura 3.12: Representació dels valors de les matrius kernel



Kernels per a strings

Els kernels per a *strings* estan implementats en la funció `stringdot`. Un dels arguments d'aquesta funció és el tipus de kernel a utilitzar. Actualment hi ha 4 kernels diferents implementats. Informació detallada sobre aquests kernels es pot trobar a les pàgines 96-97 de [20].

Per a assimilar el concepte del *spectrum kernel* es mostra a continuació un petit exemple on les seqüències d'entrada són dues paraules i es calcula el *2-spectrum kernel*. En aquest context, l'alfabet $A^k = A^2$ està format per qualsevol parella de dues lletres. Si es té en compte que l'alfabet català té 26 lletres, el cardinal del nostre alfabet A^2 és $26^2 = 676$ seqüències u. Cada una d'aquestes subseqüències u de longitud 2 és un 2-mer. Així doncs, el *2-spectrum* és el conjunt de totes les subseqüències contigües de longitud *k=2* que contingui cadascuna de les dues paraules $\mathbf{x}$. Per últim, $\phi_u(\mathbf{x})$ és el nombre de cops que cada 2-mer es dóna en la paraula $\mathbf{x}$ (sense espais).

$\mathbf{x}$ = RADAR El 2-spectrum per a $\mathbf{x}$ és (RA, AD, DA, AR). Per tant,

u	RA	AD	DA	AR
$\phi_u(\mathbf{x})$	1	1	1	1

I $\phi_u(\mathbf{x}) = 0$ per a tots els altres u fora del 2-spectrum. És a dir, per a la resta de possibles parelles de lletres (per exemple AA, AB, BC, ...) $\phi_u(\mathbf{x})$ valdrà zero. Aleshores, el kernel és:

$$k(\mathbf{x}, \mathbf{x}) = \sum_{u \in A^k} \phi_u(\mathbf{x})\phi_u(\mathbf{x}) = 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4$$

$\mathbf{x}' =$ **ABRACADABRA** El 2-spectrum per a $\mathbf{x}'$ és (AB, BR, RA, AC, CA, AD, DA). Per tant,

$\mathbf{u}$	AB	BR	RA	AC	CA	AD	DA
$\phi_u(\mathbf{x}')$	2	2	2	1	1	1	1

Aleshores, el kernel és:

$$k(\mathbf{x}', \mathbf{x}') = \sum_{u \in A^k} \phi_u(\mathbf{x}')\phi_u(\mathbf{x}') = 2 \cdot 2 + 2 \cdot 2 + 2 \cdot 2 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 16$$

I per últim, tenint en compte que les subseqüències comunes entre $\mathbf{x}$ i $\mathbf{x}'$ són (RA, AD, DA), es té:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{u \in A^k} \phi_u(x)\phi_u(\mathbf{x}') = 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 1 = 4$$

Així doncs, la matriu kernel per a aquest exemple és:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \begin{bmatrix} 4 & 4 \\ 4 & 16 \end{bmatrix}$$

Això mateix es calcula amb R amb el següent codi:

```
> sk <- stringdot(type="spectrum", length=2, normalized=FALSE)
> sk

String kernel function.  Type =  spectrum
 Hyperparameters : sub-sequence/string length =  2
 Not Normalized

> kernelMatrix(sk, c('radar','abracadabra'))

An object of class "kernelMatrix"
     [,1] [,2]
[1,]    5    4
[2,]    4   17
```

S'observa com no coincideix exactament amb el calculat a mà. Això és perquè per a la funció `stringdot` els strings contenen un caràcter addicional amagat al final, de manera que els 2-spectrum són:

- Per a $\mathbf{x}$ = RADAR, es té (RA, AD, DA, AR, R_).
- Per a $\mathbf{x}'$ = ABRACADABRA, es té (AB, BR, RA, AC, CA, AD, DA, A_).

I com que el nou 2-mer no coincideix per a $\mathbf{x}$ i $\mathbf{x}'$, $k(\mathbf{x}, \mathbf{x}')$ no canvia, però sí $k(\mathbf{x}, \mathbf{x})$ i $k(\mathbf{x}', \mathbf{x}')$.

A continuació es mostra un exemple més real, utilitzant com a seqüències d'entrada 5 proteïnes:

```
$`P00269|SwissProt|Cytoplasmic|Rubredoxin`
[1] "MKKYVCTVCGYEYDPAEGDPDNGVKPGTSFDDLPADWVCPVCGAPKSEFEAA"

$`P22187|SwissProt|Periplasmic/InnerMembrane|Cell`
[1] "MISRVTEALSKVKGSMGSHERHALPGVIGDDLLRFGKLPLCLFICIILTAVTVVTTAHHTRLLT
AQREQLVLERDALDIEWRNLILEENALGDHSRVERIATEKLQMQHVDPSQENIVVQK"

$`P33582|SwissProt|Inner`
[1] "MYEALLVVFLIVAIGLVGLIMLQQGKGADMGASFGAGASATLFGSSGSGNFMTRMTALLAT
LFFIISLVLGNINSNKTNKGSEWENLSAPAKTEQTQPAAPAKPTSDIPN"

$`P02903|SwissProt|Cytoplasmic|Citrate`
[1] "MEMKIDALAGTLESSDVMVRIGPAAQPGIQLEIDSIVKQQFGAAIEQVVRETLAQLGVK
QANVVVDDKGALECVLRARVQAAALRAAQQTQLQWSQL"

$`P02908|SwissProt|Cytoplasmic|"PTS`
[1] "GLFDKLKSLVSDDKKDTGTIEIVAPLSGEIVNIEDVPDVVFAEKIVGDGIAIKPTG
NKMVAPVDGTIGKIFETNHAFSIESDSGIELFVHFGIDTVELKGEGFKRIAEEGQRVKVGDPVIEF
DLPLLEEKAKSTLTPVVISNMDEIKELIKLSGSVTVGETPVIRIKK"
```

Tal i com s'observa, les proteïnes són seqüències d'aminoàcids, i en total hi ha 20 aminoàcids diferents, cadascun representat per una lletra del conjunt {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. S'utilitza un kernel del tipus espectre amb k = 3, pel que l'alfabet està format per tots els conjunts de 3 aminoàcids, amb un cardinal de $20^3 = 8000$ cadenes. Un vector del *feature space* és un vector de comptatges de longitud 8000 on cada posició correspon a una de les seqüències de 3 aminoàcids i el valor de cada posició equival al nombre de cops que aquesta seqüència de 3 aminoàcids es dóna en la proteïna. La matriu kernel que s'obté és:

```
> sk <- stringdot(type="spectrum", length=3, normalized=FALSE)
> sk

String kernel function.  Type =  spectrum
 Hyperparameters : sub-sequence/string length =  3
 Not Normalized
```

```
> kernelMatrix(sk,x)

An object of class "kernelMatrix"
     [,1] [,2] [,3] [,4] [,5]
[1,]   53    1    0    1    2
[2,]    1  124    3    1    4
[3,]    0    3  121    4    5
[4,]    1    1    4  102    1
[5,]    2    4    5    1  181
```

El més habitual és normalitzar els valors del string kernel. Aquesta normalització fa que el rang de valors de la matriu kernel estigui entre 0 i 1, on 0 indica que no hi ha cap similitud i 1 indica que la similitud és màxima (és el cas de la diagonal). En aquest cas la matriu seria:

```
> sk <- stringdot(type="spectrum", length=3, normalized=TRUE)
> sk

String kernel function.  Type =  spectrum
 Hyperparameters : sub-sequence/string length =  3
 Normalized

> kernelMatrix(sk,x)

An object of class "kernelMatrix"
           [,1]        [,2]       [,3]        [,4]        [,5]
[1,] 1.00000000 0.012335343 0.00000000 0.013600722 0.020419861
[2,] 0.01233534 1.000000000 0.02449163 0.008891787 0.026699914
[3,] 0.00000000 0.024491632 1.00000000 0.036005365 0.033786098
[4,] 0.01360072 0.008891787 0.03600537 1.000000000 0.007359709
[5,] 0.02041986 0.026699914 0.03378610 0.007359709 1.000000000
```

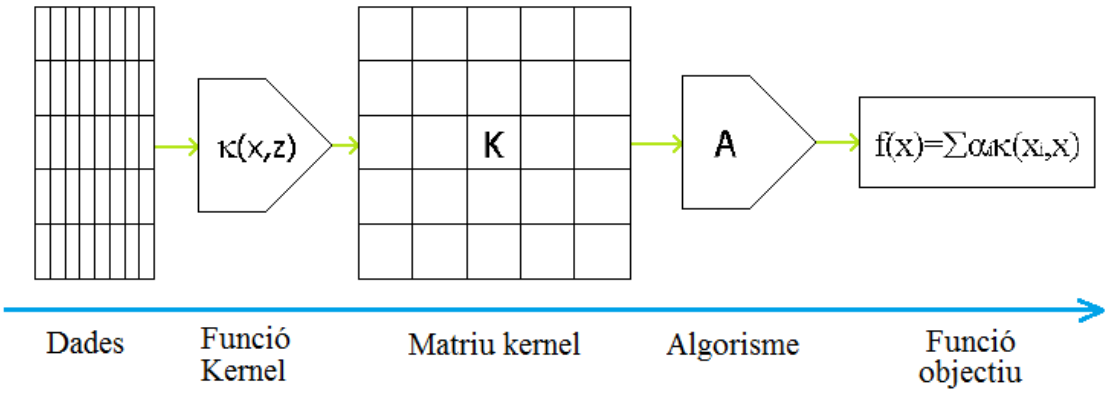
3.2.4 Mètodes kernel

Els mètodes kernel es poden definir com aquells per als quals les dades a ser analitzades només entren en l'algorisme mitjançant la funció kernel, o en altres paraules, els algorismes que prenen com a dades d'entrada la matriu kernel de similitud definida per un kernel. El resultat que dóna lloc a aquesta afirmació es coneix com el *kernel trick*. Hi ha molts algorismes que poden usar kernels, alguns dels quals són les màquines de vectors suport (SVM), el *clustering*, l'anàlisi de components principals (PCA), l'anàlisi de correlacions canòniques (CCA), la *ridge regression*, etc.

Kernel trick

El *kernel trick* és un principi general que sorgeix d'entendre els kernels com a productes escalars. El que diu és que qualsevol algorisme per a dades vectorials que pugui ser expressat només en termes de productes escalars entre vectors es pot realitzar implícitament en el *feature space* associat a qualsevol kernel, mitjançant la substitució de cada producte escalar pel resultat del kernel.

Figura 3.13: Esquema del *kernel trick* [4]



Aquest principi deriva grans conseqüències pràctiques. Per exemple:

- És un truc molt convenient per a transformar mètodes lineals en no lineals, simplement substituint el clàssic producte escalar per un kernel més general. D'aquesta manera els algorismes clàssics romanen iguals però es poden utilitzar amb dades no lineals sense cap cost computacional. L'operació que transforma un algorisme lineal en un mètode kernel més general s'anomena *kernelització*.
- La combinació del *kernel trick* amb kernels definits en dades no vectorials permet aplicar molts algorismes clàssics en pràcticament qualsevol tipus de dades, sempre i quan el kernel es pugui definir. Per exemple, es podria fer un PCA en un conjunt de seqüències de proteïnes.

Punts forts i punts dèbils

Algunes de les avantatges de treballar amb kernels es resumeixen a continuació:

- Es poden descobrir relacions no lineals entre les dades aplicant les tècniques clàssiques lineals.
- Es poden realitzar no només per a dades numèriques, sinó també per a seqüències de caràcters, texts, imatges i grafs.

[4] Imatge extreta de [41].

- Són mètodes computacionalment eficients, ja que no cal trobar la representació explícita de les dades en l'espai de característiques, que sol ser de gran dimensió, sinó que és suficient amb calcular els productes escalars.
- Proporcionen una manera natural d'integrar diferents fonts de dades.

Ara bé, com passa amb qualsevol tècnica, també hi ha certes limitacions associades a l'ús de mètodes kernel. Aquí es destaquen dos de bastant importants:

Tria del kernel En primer lloc, no hi ha una manera concreta ni de triar el millor kernel ni de triar de forma adequada els seus paràmetres. És una decisió que depèn tant de les dades que es tinguin com de l'objectiu de l'anàlisi i que requereix anar provant un darrera l'altre fins a trobar bons resultats.

Variables originals Un altre gran desavantatge es troba en la interpretació i la representació de les variables originals. Les variables originals queden ocultes dins del kernel, i és molt difícil mesurar la seva importància. Per exemple, en problemes de classificació, és difícil saber quines són les variables que més influeixen en aquesta separació, mentre que amb altres tècniques com els arbres de decisió això queda solucionat, ja que la seva interpretació és directa.

Capítol 4

TÈCNIQUES D'INTEGRACIÓ: KERNEL PCA

La integració de dades es pot definir com l'ús de múltiples fonts d'informació per tal de proporcionar una millor comprensió d'un sistema / situació / associació / etc. En el context de les dades òmiques, la integració de dades suposa un repte més complex donat la gran quantitat d'informació que es genera contínuament. En aquest capítol es presenta una breu introducció a diferents tècniques d'integració, i posteriorment s'explica en profunditat el Kernel PCA juntament amb un conjunt de procediments per a millorar la seva interpretació i amb detalls sobre la seva implementació en R.

4.1 Introducció a les tècniques d'integració de dades

El desenvolupament de les tecnologies per a mesurar dades òmiques i el posterior ús de les dades que es produeixen ha revolucionat la biologia molecular i ha motivat el desenvolupament de nous i sofisticats algorismes en el camp de la bioinformàtica. No obstant, és relativament recent la investigació de la possibilitat d'analitzar dades òmiques de més d'un tipus de forma simultània. Agafar dos tipus de dades òmiques i analizar-les de forma combinada segur que ens proporciona més informació que no pas si s'analitzen individualment. Una possibilitat que es deriva d'aquesta integració de diferents fonts de dades consisteix en combinar dades d'expressió transcriptòmica i proteòmica, cosa que permetria aprendre sobre els complexos mecanismes de regulació els quals, quan funcionen correctament, consoliden el comportament cel·lular, però que quan fallen poden ser responsables de malalties com el càncer.

Existeixen diversos mètodes estadístics multivariants que permeten combinar diferents matrius de dades. A continuació es presenten de forma breu els trets principals d'alguns dels mètodes existents.

Anàlisi de correlacions canòniques

El mètode estadístic "estàndard" per a combinar dues fonts de dades és el *Canonical Correlation Analysis* (CCA). L'anàlisi de correlacions canòniques identifica i quantifica associacions entre dos conjunts de variables buscant combinacions lineals de les variables originals amb correlació màxima. Un cop que troba el primer parell de combinacions lineals amb correlació màxima, els següents parells s'escullen de tal forma que són ortogonals als altres parells ja identificats. Els parells de combinacions lineals són anomenats **variables canòniques** i les seves correlacions són les **correlacions canòniques**. Aquestes correlacions canòniques mesuren la força de l'associació entre els dos conjunts de variables. Un cop realitzat el CCA es poden projectar els dos conjunts de dades en un espai de dimensió reduïda en el qual tenen correlació màxima. Aquesta tècnica es desenvolupa en el paquet de R `CCA`, juntament amb resultats gràfics per a millorar la seva interpretació.

Un obstacle del clàssic CCA és que assumeix que $p < n$ i $q < n$, on p i q són el nombre de variables en cada conjunt i n el total d'individus. Ara bé, en el cas de les dades òmiques per a les quals es vol aplicar aquesta tècnica això no es compleix: es tenen conjunts de milers de variables per a desenes de mostres. Una manera de fer front a aquest problema consisteix en la inclusió d'una etapa de regularització en el procediment, donant peu a l'extensió del CCA anomenada *Regularized CCA* (rCCA). Aquesta tècnica es desenvolupa en el paquet de R `mixOmics`, juntament amb altres tècniques estadístiques d'integració per analitzar conjunts de dades de grans dimensions.

Si bé amb l'anàlisi de correlacions canòniques regularitzat es solventa l'anterior problema, una de les seves limitacions és que només es pot utilitzar per a combinar dues fonts de dades. En l'anàlisi de dades òmiques, però, existeixen molts conjunts de dades (genòmiques, metabolòmiques, clíniques, etc). Com a resposta a aquesta limitació sorgeix l'anàlisi de correlacions canòniques regularitzat generalitzat (RGCCA), que és una generalització del rCCA per a tres o més blocs de variables, i que té com a objectiu estudiar les relacions entre aquests blocs de variables. Aquesta tècnica es pot trobar al paquet de R `RGCCA`.

Donat que la base d'aquesta tècnica és trobar correlacions entre els blocs de variables, només es poden aplicar aquests mètodes quan es tenen dades numèriques, cosa que no sempre es compleix quan es tracta de dades òmiques. Per exemple, les proteïnes són seqüències de caràcters. Una opció és aplicar el *kernel trick* al CCA clàssic, obtenint l'extensió no lineal Kernel CCA (KCCA). A més de permetre treballar amb diferents tipus de dades, el KCCA té el potencial de trobar relacions no lineals entre els dos conjunts de dades.

Anàlisi factorial múltiple

L'anàlisi factorial múltiple (Escofier and Pagès, 1992) és una tècnica estadística multivariant que permet analitzar individus que estan descrits per diversos blocs de variables, les quals poden ser de diferents tipus (contínues o categòriques). D'aquesta manera es pot comparar l'estructura dels individus induïda pels diversos grups de variables.

L'anàlisi factorial múltiple es realitza es dos passos:

Anàlisi per separat En un primer pas, es realitza un anàlisi de components principals (PCA) en cada subconjunt de dades, per després normalitzar els conjunts dividint tots els seus elements per l'invers del primer valor propi obtingut en el PCA. En aquest pas s'està balancejant la influència de cada bloc de variables, per evitar que un sol bloc contribueixi per ell mateix a la construcció dels primers eixos i domini l'anàlisi comú.

Anàlisi global En un segon pas, els subconjunts normalitzats es concatenen per a formar una única matriu, sobre la qual es realitza un PCA. Un cop es té, es poden projectar els subconjunts individuals en l'anàlisi global per a estudiar les associacions i les discrepàncies.

L'anàlisi factorial múltiple és una tècnica atractiva per diverses raons: permet combinar no només dos sinó múltiples conjunts de dades, els diversos blocs de variables poden ser de diferents tipus, tant de variables contínues com categòriques, i també permet afegir grups de dades suplementaris. El que cal destacar és que les relacions que permet trobar són de tipus lineal, cosa que pot ser una limitació en l'anàlisi de dades òmiques. Aquesta tècnica juntament amb diversos resultats gràfics es pot trobar al paquet de R `FactoMineR`.

Kernel PCA

El Kernel PCA és l'extensió no lineal de l'anàlisi de components principals, que permet la integració de dades des del punt de vista de reducció de la dimensió, i ho fa a través de l'ús de funcions kernel específiques. Aquestes funcions kernel es construeixen com combinacions de kernels vàlids. Una de les grans avantatges de treballar amb kernels és que es poden analitzar dades de tot tipus, no només numèriques. Així doncs, amb el Kernel PCA es pot realitzar un PCA de cadenes de caràcters, de texts, d'imatges, etc. A més, és una tècnica que funciona bé quan les dades són de gran dimensionalitat i quan les relacions entre les dades són no lineals, que són les sitacions habituals quan es treballa amb dades òmiques.

Ara bé, la gran limitació de la tècnica és que es perd la interpretació de les variables originals, que queden amagades dins del kernel. En aquest sentit es destaca el conjunt de

procediments desenvolupats per Reverter et al. (2014) [37] per a millorar la interpretació del Kernel PCA afegint informació suplementària a les representacions dels individus de la tècnica. En els següents apartats s'explica en profunditat les característiques del Kernel PCA i d'aquest conjunt de procediments, juntament amb la seva implementació en R i la seva aplicació a uns conjunts de dades simulades.

4.2 Kernel PCA

El Kernel PCA és una extensió de l'anàlisi de components principals que permet dur a terme una reducció de la dimensió de forma no lineal utilitzant tècniques de mètodes kernel. Això fa que aquesta tècnica sigui més realista que el simple PCA quan la relació entre les variables és no lineal, com per exemple en la interacció entre els gens. La idea del mètode es basa en calcular el PCA clàssic però un espai de característiques de gran dimensió, relacionat amb l'espai de les observacions per una transformació no lineal donada per una certa funció kernel. Aquest mètode es descriu a Scholkopf et al. (1998) [40].

En un primer apartat s'explica la metodologia del PCA estàndard per després en un segon apartat generalitzar la tècnica al cas no lineal. Un cop detallat el procediment del Kernel PCA, aquest s'aplica a unes dades simulades per veure la seva potència, i després s'explica com es pot utilitzar per a integrar informació de diverses fonts de dades. A continuació es presenta un nou procediment per a millorar la interpretabilitat de la tècnica afegint informació sobre les variables originals.

4.2.1 Anàlisi de components principals clàssic (PCA)

L'anàlisi de components principals es pot definir com una tècnica de reducció de la dimensió i de visualització de la informació. La idea central de l'anàlisi de components principals (PCA) és reduir la dimensionalitat d'un conjunt de dades que consta d'un gran nombre de variables interrelacionades, i al mateix temps conservar la màxima variabilitat possible present al conjunt de dades. Això s'aconsegueix mitjançant la transformació a un nou conjunt de variables, les anomenades **components principals** (PCs), que són combinacions lineals de les variables originals, que no estan correlacionades entre elles i que estan ordenades de tal manera que les primeres components són les que conserven la major part de la variació present en totes les variables originals.

El procediment per a realitzar aquesta tècnica es detalla a continuació:

Dades de partida Es parteix d'un conjunt de n individus que es volen analitzar ($\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$), on cada objecte $\mathbf{x}_i$ és un element de l'espai d'entrada $\mathcal{X}$, i primer

suposarem que $\mathcal{X} = \mathbb{R}^p$. Cal que les dades estiguin centrades, és a dir, cal que $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$. En cas de no tenir-les centrades, el primer pas és centrar-les restant a cada component de $\mathbf{x}_i$ la mitjana de la variable corresponent:

$$\sum_{i=1}^{n} x_{ij} - \bar{x}_j \quad j = 1, \ldots, p$$

Matriu de covariàncies El següent pas és calcular la matriu de covariàncies de les dades, que serà una matriu $\mathbf{C}$ de mida $p \times p$ definida per:

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{x}_i = \mathbf{X}^T \mathbf{X}$$

Primera component principal Amb les dades centrades i amb la matriu de covariàncies calculada, ja es pot passar a calcular les components principals. La primera component principal s'expressa com:

$$\mathbf{z}_1 = \mathbf{v}_1^T \mathbf{X}$$

on els coeficients del vector $\mathbf{v}_1$ s'escullen de tal forma que:

- $Var(\mathbf{z}_1) = Var(\mathbf{v}_1^T \mathbf{X}) = \mathbf{v}_1^T \mathbf{C} \mathbf{v}_1$ sigui màxima.
- El vector $\mathbf{v}_1$ tingui norma 1 ($\| \mathbf{v}_1 \| = 1$), per tal de tenir solució única i per a que la direcció sigui el factor important.

Així doncs, $\mathbf{z}_1$ apunta en la direcció i sentit de màxima dispersió. El vector $\mathbf{v}_1$ s'obté com a solució del següent problema d'optimització, que es pot resoldre mitjançant la tècnica dels multiplicadors de Lagrange.

$$\begin{aligned} &\text{màx.} && \mathbf{v}_1^T \mathbf{C} \mathbf{v}_1 \\ &\text{subjecte a} && \mathbf{v}_1^T \mathbf{v}_1 = 1. \end{aligned}$$

Aquest problema es redueix a resoldre el següent sistema d'equacions:

$$\mathbf{C}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

On:

- L'escalar λ_1 és el valor propi més gran de la matriu de covariàncies $\mathbf{C}$. I aquest valor propi és justament la variància de la primera component principal.
- El vector $\mathbf{v}_1$ és el vector propi del sistema d'equacions.

Segona component principal La segona component principal es defineix com:

$$\mathbf{z}_2 = \mathbf{v}_2^T \mathbf{X}$$

on els coeficients del vector $\mathbf{v}_2$ s'escullen de tal forma que:

- $Var(\mathbf{z}_2) = Var(\mathbf{v}_2^T \mathbf{X}) = \mathbf{v}_2^T \mathbf{C} \mathbf{v}_2$ sigui màxima.
- El vector $\mathbf{v}_2$ tingui norma 1 ($\| \mathbf{v}_2 \| = 1$).
- $\langle \mathbf{v}_2, \mathbf{v}_1 \rangle = 0$. Es demana que $\mathbf{v}_2$ sigui ortogonal a $\mathbf{v}_1$ ja que així la segona component principal està incorrelacionada amb la primera.

I per trobar $\mathbf{v}_2$, cal resoldre:

$$\mathbf{C}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

On:

- L'escalar λ_2 és el segon valor propi més gran de la matriu de covariàncies de $\mathbf{C}$, i es correspon amb la variància de la segona component principal.
- El vector $\mathbf{v}_2$ és el vector propi del sistema d'equacions associat a λ_2, i marca la direcció de la segona component principal.

La tercera component es defineix de forma anàloga a la segona, i així successivament. Es poden trobar fins a p components principals (sempre que $p > n$), però el que s'espera és que la major part de la variabilitat de X es conservi en les m primeres components principals, on $m \ll p$.

En termes matricials En termes més generals, el que es vol es trobar la matriu $\mathbf{V}$ de mida $p \times p$ tal que $\mathbf{Z} = \mathbf{X}\mathbf{V}$, de manera que les columnes de $\mathbf{Z}$ siguin les components principals i les columnes de $\mathbf{V}$ siguin els vectors propis. Cal solucionar les equacions:

$$\mathbf{C}\mathbf{V} = \Lambda \mathbf{V}$$

On:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{bmatrix} = \begin{bmatrix} Var(z_1) & 0 & \cdots & 0 \\ 0 & Var(z_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Var(z_p) \end{bmatrix}$$

Si en comptes de centrar les dades, aquestes s'estandarditzen, aleshores els valors propis $\lambda_1, \lambda_2, \ldots, \lambda_p$ s'obtenen de la diagonalització de la matriu de correlacions de $\mathbf{X}$, i no pas de la matriu de covariàncies.

4.2.2 Procediment del Kernel PCA

Tal i com s'ha vist en l'apartat anterior, el PCA es redueix a diagonalitzar la matriu de covariàncies del conjunt de dades, sempre que aquestes estiguin centrades. El procediment del Kernel PCA és anàleg al del PCA clàssic, però la matriu a diagonalitzar és la matriu kernel $\mathbf{K}$ que conté els productes escalars en $\mathcal{F}$ de tots els individus (veure apartat 3.2.1).

Dades de partida Es parteix del conjunt de n individus a analitzar en l'espai de característiques $\mathcal{F}$, és a dir, es té $\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_n)$, on cada objecte $\phi(\mathbf{x}_i)$ és un element de l'espai $\mathcal{F}$, de gran dimensionalitat.

Dades centrades Cal que les dades estiguin centrades. Així doncs, es defineix la mitjana com:

$$\bar{\phi} = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i)$$

De forma que els següents punts ja estan centrats:

$$\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \bar{\phi}$$

Matriu kernel de les dades centrades Es defineix $\tilde{K}_{ij} = \langle\tilde{\phi}(\mathbf{x}_i), \tilde{\phi}(\mathbf{x}_j)\rangle$ com l'element ij de la matriu kernel de les dades centrades. Ara bé, les dades centrades $\tilde{\phi}(\mathbf{x}_i)$ no es tenen, ja que generalment $\phi(\mathbf{x})$ és desconegut, pel que la matriu $\tilde{\mathbf{K}}$ no es pot calcular de forma explícita. No obstant, sí que es pot expressar en termes de la seva homòloga no centrada $\mathbf{K}$, on $K_{ij} = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$.

$$\begin{aligned}\tilde{K}_{ij} &= \langle\tilde{\phi}(\mathbf{x}_i), \tilde{\phi}(\mathbf{x}_j)\rangle = \langle\phi(\mathbf{x}_i) - \bar{\phi}, \phi(\mathbf{x}_j) - \bar{\phi}\rangle \\ &= \left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{l=1}^{n}\phi(\mathbf{x}_l)\right)\left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_{l=1}^{n}\phi(\mathbf{x}_l)\right)^T \\ &= K_{ij} - \frac{1}{n}\sum_{l=1}^{n}K_{il} - \frac{1}{n}\sum_{l=1}^{n}K_{jl} + \frac{1}{n^2}\sum_{l,t=1}^{n}K_{lt}\end{aligned}$$

I es pot escriure en forma matricial i més compacta, utilitzant el vector $\mathbf{1}_n = (1, \ldots, 1)^T$, com:

$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\mathbf{1}_n^T - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\mathbf{K} + \frac{1}{n^2}(\mathbf{1}_n^T\mathbf{K}\mathbf{1}_n)\mathbf{1}_n\mathbf{1}_n^T$$

Matriu de covariàncies El següent pas és calcular la matriu de covariàncies de les dades centrades en $\mathcal{F}$, que s'expressa com:

$$\tilde{\mathbf{C}} = \frac{1}{n}\sum_{j=1}^{n}\tilde{\phi}(\mathbf{x}_j)^T\tilde{\phi}(\mathbf{x}_j)$$

Diagonalització de la matriu de covariàncies Amb les dades centrades i amb la matriu de covariàncies calculada, el problema de trobar les components principals passa per resoldre:

$$\tilde{\mathbf{C}}\tilde{\mathbf{V}} = \tilde{\lambda}\tilde{\mathbf{V}}$$

On:

- Els escalars $\tilde{\lambda}$ són els valors propis que satisfan el sistema d'equacions.
- Els vectors $\tilde{\mathbf{V}}$ són els vectors propis del sistema d'equacions.

Es pot demostrar que els vectors propis es poden expressar com a combinacions lineals de les dades $\tilde{\phi}(\mathbf{x}_i)$ com:

$$\tilde{\mathbf{V}} = \sum_{i=1}^{n} \tilde{\alpha}_i \tilde{\phi}(\mathbf{x}_i)$$

De tal forma que trobar els vectors propis equival a trobar els coeficients $\tilde{\alpha}_i$.

Valors i vectors propis Per trobar les solucions de $\tilde{\mathbf{C}}\tilde{\mathbf{V}} = \tilde{\lambda}\tilde{\mathbf{V}}$ es pot resoldre:

$$\tilde{\mathbf{K}}\tilde{\alpha} = n\tilde{\lambda}\tilde{\alpha}$$

On:

- n es correspon amb el nombre d'individus / observacions.
- $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \ldots \geq \tilde{\lambda}_n$ són els valors propis de $\tilde{\mathbf{K}}$, on $\tilde{\lambda}_r$ és l'últim valor propi diferent de 0.
- $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_n$ són els vectors propis. Un cop calculats, cal normalitzar els $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_r$ ($\tilde{\lambda}_k \langle \tilde{\alpha}^k, \tilde{\alpha}^k \rangle = 1$).

Coordenades dels punts en les components principals Un cop calculades les components principals, les coordenades dels punts es troben calculant la seva projecció en aquestes noves variables. Si es considera un punt nou $\mathbf{y}$, la seva projecció $\tilde{\phi}(\mathbf{y})$ en la component principal k-èssima ($k = 1, \ldots, r$) s'expressa com:

$$\langle \tilde{\mathbf{V}}^k, \tilde{\phi}(\mathbf{y}) \rangle = \sum_{j=1}^{n} \tilde{\alpha}_j^k \tilde{k}(\mathbf{x}_j, \mathbf{y})$$

Sent $\mathbf{Z} = (k(\mathbf{y}, \mathbf{x}_i))_{n \times 1}$ i $\tilde{\mathbf{V}}$ la matriu $n \times r$ que té per columnes els vectors propis $\tilde{\mathbf{V}}^1, \ldots, \tilde{\mathbf{V}}^r$, aquesta projecció en les r components s'expressa com:

$$\left(\langle \tilde{\mathbf{V}}^k, \tilde{\phi}(\mathbf{y}) \rangle \right)_{1 \times r} = \left(\mathbf{Z}^T - \frac{1}{n} \mathbf{1}_n^T \mathbf{K} \right) \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T \right) \tilde{\mathbf{V}} \tag{4.1}$$

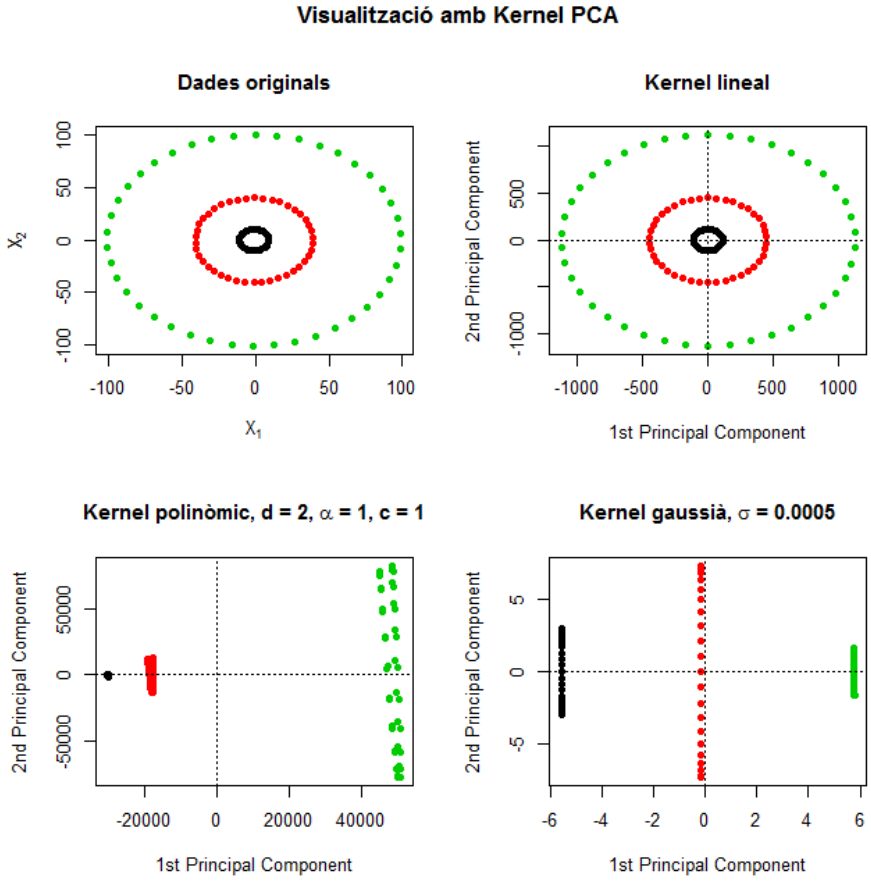
Un cop explicat el procediment, es pot veure com l'algorisme del KPCA es pot resumir a grans trets en cinc passos:

- Calcular la matriu kernel $\mathbf{K}$, on $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \quad i, j = 1, \ldots, n$
- Centrar la matriu kernel, és a dir, trobar $\tilde{\mathbf{K}}$.
- Diagonalitzar la matriu kernel centrada per a trobar els valors i els vectors propis (λ_i i α_k).
- Transformar els vectors propis per a que tinguin norma 1.
- Calcular les projeccions dels punts en les noves components.

Visualització amb Kernel PCA

Per tal d'entendre la capacitat del Kernel PCA, a continuació es mostra la seva aplicació a un conjunt de dades simulat. En concret, s'han generat dues variables X_1 i X_2 de forma que la representació dels individus en el pla formi tres cercles de diferent radi ($r_1 = 10$, $r_2 = 40$, $r_3 = 100$), incloent una mica de soroll. A continuació s'ha aplicat el Kernel PCA usant tres kernels diferents: el kernel lineal, el polinòmic (de grau 2) i el gaussià (amb $\sigma = 0.0005$), i s'ha representat la projecció dels individus en el primer pla factorial per a les tres anàlisis. Els resultats es mostren en la següent figura:

Figura 4.1: Visualització amb Kernel PCA segons diversos kernels



S'observa com el PCA clàssic, donat que és una tècnica lineal, és incapaç de separar els tres grups d'individus. En canvi, gràcies a l'ús dels kernels, els PCA en l'espai de característiques induïts tant pel kernel polinòmic com pel gaussià sí que troben una representació dels individus de forma que són linealment separables. Aquest és un exemple de la potència d'aquesta tècnica.

Cal notar, però, que encara que s'hagin marcat els punts de diferents colors donat que la classe de cadascun és sap a priori, el Kernel PCA és una tècnica d'aprenentatge no supervisat. Ni el PCA estàndard ni el Kernel PCA prenen les etiquetes de les classes com a *input*, donat que la situació general és que siguin desconegudes.

Integració de diverses fonts d'informació amb Kernel PCA

Els mètodes d'integració de dades basats en kernels consisteix en dos passos. Primer es tria el kernel adequat per a cada un dels conjunts a integrar. Segon, els kernels de les diferents fonts de dades es combinen per a donar una representació completa de totes les dades disponibles. Aquesta combinació de kernels és possible ja que operacions algebraiques bàsiques com la suma i el producte conserven les propietats de les matrius kernel i per tant donen lloc a kernels vàlids (veure l'apartat 3.2.2). Per exemple, sent a i b dos constants, els següents kernels són vàlids:

$$k(\mathbf{x}, \mathbf{x}') = \begin{cases} ak_1(\mathbf{x}, \mathbf{x}') \\ k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\ ak_1(\mathbf{x}, \mathbf{x}') + bk_2(\mathbf{x}, \mathbf{x}') \end{cases}$$

Així doncs, per exemple, siguin k_1 i k_2 dos kernels definits respectivament a $\mathcal{X}_1 \times \mathcal{X}_1$ i $\mathcal{X}_2 \times \mathcal{X}_2$, la seva suma directa:

$$(k_1 \oplus k_2)(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x'}_1, \mathbf{x'}_2) = k_1(\mathbf{x}_1, \mathbf{x'}_1) + k_2(\mathbf{x}_2, \mathbf{x'}_2)$$

és un kernel en $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$, on $\mathbf{x}_1, \mathbf{x'}_1 \in \mathcal{X}_1$ i $\mathbf{x}_2, \mathbf{x'}_2 \in \mathcal{X}_2$. Això permet integrar dos conjunts de dades diferents $\mathbf{X}_1$, $\mathbf{X}_2$ amb la possibilitat d'utilitzar un kernel diferent per a cadascun. Amb el Kernel PCA es realitza la integració de dades des del punt de vista de la reducció d'informació.

4.2.3 Millora de la interpretabilitat del Kernel PCA

Un dels grans incovenients dels mètodes kernel és la interpretació dels resultats, donat que les variables originals queden amagades dins del kernel ja que $\phi(\mathbf{x})$ no es coneix. A continuació s'explica un conjunt de procediments desenvolupats per Reverter et al. (2014)

[37] per a millorar la interpretabilitat del Kernel PCA afegint informació suplementària a les representacions dels individus de la tècnica, en un intent d'extendre el *biplot* del PCA lineal. En concret, es desenvolupen quatre aspectes:

- **Representació de les variables originals**
- **Representació de combinacions lineals de les variables originals**
- **Integració de diferents fonts d'informació i representació de les variables originals**
- **Descobrir la interpretació de les variables originals**

Cal comentar, però, que de moment aquest procediment només està desenvolupat per a tractar amb kernels del tipus gaussià, sent aquesta una limitació. A continuació es detallen els quatre aspectes del procediment.

Representació de les variables originals

Es millora la interpretabilitat del Kernel PCA afegint al gràfic resultant d'aplicar la tècnica una representació de les variables originals del conjunt de dades. En particular, per a cada variable d'entrada es representa la direcció de màxim creixement a nivell local en cada individu de la mostra. Això permet identificar variables que es comportin de forma similar en tots els individus, i en particular que apuntin a grups d'individus amb major variabilitat respecte a una variable, és a dir, amb valors superiors de les variables analitzades.

Es considera que les observacions són realitzacions del vector aleatori $\mathbf{X} = (X_1, \dots, X_p)$, i es vol representar la prominència de la variable d'entrada X_k en el Kernel PCA. S'escull un conjunt de punts de la forma $\mathbf{y} = \mathbf{a} + s\mathbf{e}_k \in \mathbb{R}^p$, on $\mathbf{e}_k = (0, \dots, 1, \dots, 0) \in \mathbb{R}^p$, $s \in \mathbb{R}$, on la k-èssima component és igual a 1 mentre que les altres són 0. El següent pas és representar aquests punts en el subespai definit pels vectors propis de la matriu de covariàncies $\tilde{C}$ de les variables, calculant les coordenades en les components principals mitjançant les projeccions $\tilde{\phi}(\mathbf{y})$. La corba s'expressa, segons el vist a 4.1, com:

$$\sigma(s)^k_{1\times r} = \left(\mathbf{Z}_s^T - \frac{1}{n}\mathbf{1}_n^T\mathbf{K}\right)\left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\right)\tilde{\mathbf{V}}, \tag{4.2}$$

on $\mathbf{Z}_s^T = (k(\mathbf{y}, \mathbf{x}_i))_{n\times 1}$ i $\tilde{\mathbf{V}}$ és la matriu $n \times r$ que té per columnes els vectors propis.

Per a representar la direcció de màxim creixement de $\sigma^k(s)$ respecte a la variable X_k cal projectar el vector tangent a $s = 0$. En forma matricial es té:

$$\left.\frac{d\sigma^k}{ds}\right|_{s=0} = \left.\frac{d\mathbf{Z}_s^T}{ds}\right|_{s=0}\left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\right)\tilde{\mathbf{V}}, \tag{4.3}$$

On:

$$\left.\frac{d\mathbf{Z}_s^T}{ds}\right|_{s=0} = \left(\left.\frac{d\mathbf{Z}_s^1}{ds}\right|_{s=0}, \ldots, \left.\frac{d\mathbf{Z}_s^n}{ds}\right|_{s=0}\right)^T,$$

I aquestes derivades de $\mathbf{Z}_s$ depenents del kernel usat, es poden escriure usant la regla de la cadena com:

$$\left.\frac{d\mathbf{Z}_s^i}{ds}\right|_{s=0} = \left.\frac{\partial k(\mathbf{y}, \mathbf{x}_i)}{\partial y_k}\right|_{\mathbf{y}=\mathbf{a}} \tag{4.4}$$

A continuació es mostra els resultats particulars per al kernel gaussià:

Kernel gaussià La funció kernel és:

$$k(\mathbf{y}, \mathbf{x}_i) = exp(-\sigma \parallel \mathbf{y} - \mathbf{x}_i \parallel^2) = exp\left(-\sigma \sum_{j=1}^{p} (y_j - x_{ij})^2\right)$$

Per al conjunt de punts de la forma $\mathbf{y} = \mathbf{a} + s\mathbf{e}_k \in \mathbb{R}^p$ es té:

$$\left.\frac{d\mathbf{Z}_s^i}{ds}\right|_{s=0} = \left.\frac{\partial k(\mathbf{y}, \mathbf{x}_i)}{\partial y_k}\right|_{\mathbf{y}=\mathbf{a}} = -2\sigma k(\mathbf{a}, \mathbf{x}_i)(a_k - x_{ik})$$

I en concret si $\mathbf{a} = \mathbf{x}_\beta$ (punt del conjunt de *training*), aleshores:

$$\left.\frac{d\mathbf{Z}_s^i}{ds}\right|_{s=0} = -2\sigma k(\mathbf{x}_\beta, \mathbf{x}_i)(x_{\beta k} - x_{ik})$$

Aquests resultats, doncs, permeten per a cada variable representar la seva direcció de màxim creixement en l'espai, projectada sobre el pla generat pel kernel. Pot ser que la llargada de les fletxes per a diverses variables sigui diferent, i això és degut a com està situada cada variable en l'espai i com de bé es projecta sobre el pla. Si interessa alguna variable en concret es pot intentar "deformar" el pla resultant del Kernel per tal d'aconseguir una millor representació. A més, cal puntualitzar que els autors d'aquests procediments estan investigant com calcular una mesura per tal d'avaluar aquesta bondat de la representació.

Representació de combinacions lineals de les variables originals

Una extensió natural del procediment anterior és representar combinacions lineals de les variables originals. Es suposa que es vol representar la combinació lineal $X_{k_1} + X_{k_2} + \cdots + X_{k_l}$ on $k_1, k_2, \ldots, k_l \in \{1, 2, \ldots, p\}$, amb $k_i \neq k_j$, $i, j = 1, \ldots, l$. Aleshores (4.4) és:

$$\left.\frac{d\mathbf{Z}_s^i}{ds}\right|_{s=0} = \sum_{I=1}^{l} \left.\frac{\partial k(\mathbf{y}, \mathbf{x}_i)}{\partial y_{k_I}}\right|_{\mathbf{y}=\mathbf{a}}$$

I mitjançant la projecció del vector tangent a $s = 0$ tenint en compte la definició (4.3) es pot representar qualsevol combinació lineal de les variables originals.

Integració de diferents fonts d'informació i representació de les variables originals

Amb l'ús dels kernels es poden combinar fonts de dades heterogènies gràcies a que operacions com la suma o el producte conserven les propietats dels kernels.

Així doncs, per exemple, siguin dos conjunts de dades diferents $\mathbf{X}_1$, $\mathbf{X}_2$ i el kernel K donat per $k_1 \oplus k_2$, definit com la suma directa de dos kernels k_1 i k_2. Primer es duu a terme la reducció de dimensió de les dades senceres $(\mathbf{x}_{1i}, \mathbf{x}_{2i}), i = \{1, \dots, n\}$ aplicant el Kernel PCA amb el kernel K. Després es troben les coordenades d'un punt de test $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ de forma anàloga a quan només es tenia un conjunt de dades (4.1) però ara $\mathbf{Z} = (K(\mathbf{y}_1, \mathbf{x}_{1i}, \mathbf{y}_2, \mathbf{x}_{2i}))_{n\times 1}$. Al integrar dos conjunts de dades es pot representar qualsevol variable d'entrada de qualsevol dels dos conjunts. Suposem, per exemple, que es vol representar la variable X_k^l on l indica a quin dels dos conjunts de dades pertany $(l = 1, 2)$. Aleshores (4.4) és:

$$\left.\frac{d\mathbf{Z}_s^i}{ds}\right|_{s=0} = \left.\frac{\partial K_l(\mathbf{y}_l, \mathbf{x}_{li})}{\partial y_{lk}}\right|_{\mathbf{y}_l=\mathbf{a}_l}$$

I mitjançant la projecció del vector tangent a $s = 0$ tenint en compte la definició (4.3) es poden representar variables originals pertanyents a qualsevol dels dos conjunts de dades a sobre de la representació del Kernel PCA de forma simultània. Així doncs, amb la integració de múltiples conjunts de dades diferents i la representació simulatània dels individus i de les variables es pot millorar el coneixement que proporciona la tècnica del Kernel PCA.

Descobrir la interpretació de les variables originals

Amb els procediments anteriors es poden representar les variables originals com a vectors la direcció dels quals és la direcció de màxim creixement d'una variable en un cert punt, i en particular, en els punts de la mostra. De manera inversa, es pot fixar una direcció en el pla, donada per un vector $\mathbf{w}$, i buscar quines variables originals tenen una representació en el Kernel PCA correlacionada amb aquesta direcció fixada. Per exemple, si després de fer el Kernel PCA les dades s'han separat en dos grups, una direcció interessant pot ser la que vagi del centroide d'un grup a l'altre centroide, ja que d'aquesta manera es podrà saber quines variables són les que influeixen en la separació dels dos clústers.

Primer cal seleccionar el vector $\mathbf{w}$ que marca la direcció a fixar. Després, es denota per $\mathbf{w}_i$ el vector paral·lel a $\mathbf{w}$ lligat a la projecció donada pel Kernel PCA d'un punt de

la mostra $\mathbf{x}_i, i = 1, \ldots, n$. Per a qualsevol variable X_k, la seva representació és (4.3):

$$\left.\frac{d\sigma^k}{ds}\right|_{s=0} = \left.\frac{d\mathbf{Z}_s^T}{ds}\right|_{s=0} \left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\right)\tilde{\mathbf{V}}$$

Així doncs, per a cada punt de la mostra es tenen dos vectors, un corresponent a la direcció $\mathbf{w}_i$ i un altre corresponent a la representació de la variable X_k, $\left(\left.\frac{d\sigma^k}{ds}\right|_{s=0}\right)_{\mathbf{x}_i}$. Es pot, doncs, mesurar la força de la correlació entre X_k i $\mathbf{w}$ fent la mitjana del cosinus dels angles entre cada parell de vectors per a cada individu, cosa que s'expressa com:

$$R_k = \frac{1}{n}\sum_{i=1}^{n} cos\left(\mathbf{w}_i, \left(\left.\frac{d\sigma^k}{ds}\right|_{s=0}\right)_{\mathbf{x}_i}\right)$$

Per últim, s'ordenen totes les variables respecte a R_k, i es poden seleccionar aquelles amb majors i menors valors. D'aquesta manera, per a cada grup d'indivudus, es poden trobar les variables amb més altes o baixes correlacions.

Una altra manera d'aplicar aquest procediment és, en comptes de triar una direcció $\mathbf{w}$, seleccionar la direcció d'una de les variables d'entrada. Si es sap que aquesta és important per a la interpretació, duent a terme aquest procediment es poden trobar altres variables originals tals que la seva representació en el pla del Kernel PCA estigui correlacionada amb la variable triada. I si s'estan utilizant conjunts de dades diferents per a la integració, es pot triar una variable d'un conjunt i veure quines variables de l'altre conjunt estan correlacionades amb ella. Aquest aspecte del procediment desenvolupat per Reverter et al. (2014) és el que té més utilitat pràctica de cara a un investigador, ja que li proporciona una eina exploratòria per tal d'extreure les variables més importants en la seva anàlisi i així construir hipòtesis, que després haurà de contrastar utilitzant tècniques més específiques.

4.3 Kernel PCA en R

Un cop explicat el procediment teòric del Kernel PCA, en aquest apartat s'explica com utilitzar aquesta tècnica amb R a partir d'unes dades d'exemple. Un cop vist això, es passa a aplicar els diferents procediments vistos per a millorar la interpretació d'aquest mètode.

Aplicació del Kernel PCA

La tècnica del Kernel PCA està implementada en el paquet *kernlab* de R mitjançant la funció `kpca`. Aquesta té la següent crida:

```
## S4 method for signature 'matrix'
```

```
kpca(x, kernel = "rbfdot", kpar = list(sigma = 0.1),
  features = 0, th = 1e-4, na.action = na.omit, ...)
```

A continuació s'expliquen els arguments principals de la funció:

x Les dades a analitzar, que poden entrar en la funció com una matriu indexada per files, com una fórmula describint el model, com una matriu kernel de la forma `kernelMatrix` o com una llista de vectors de caràcters.

data El `data frame` opcional que conté les variables en el model i que cal especificar si les dades `x` s'expressen com una fórmula.

kernel La funció kernel utilitzada en l'anàlisi. Les implementades al paquet *kernlab* són les funcions kernel més populars, però també es pot utilitzar una funció definida per l'usuari.

- `rbfdot` Funció kernel de base radial gaussiana.
- `polydot` Funció kernel polinomial.
- `vanilladot` Funció kernel lineal.
- `tanhdot` Funció kernel tangent hiperbòlica.
- `laplacedot` Funció kernel laplaciana.
- `besseldot` Funció kernel *Bessel*.
- `anovadot` Funció kernel de base radial ANOVA.
- `splinedot` Funció kernel *Spline*.

kpar La llista dels hiperparàmetres de cada funció kernel en particular. A continuació es mostren els paràmetres vàlids per als kernels implementats en el paquet, i en cas d'haver definit un kernel propi els seus paràmetres s'han de passar també a través de `kpar`.

- `sigma` Es correspon amb la inversa de l'amplada del kernel per als kernels `rbfdot` i `laplacedot`.
- `degree, scale, offset` El grau del polinomi, l'escala i la constant del kernel `polydot`.
- `scale, offset` L'escala i la constant del kernel `tanhdot`.
- `sigma, order, degree` Paràmentres per al kernel `besseldot`.
- `sigma, degree` Paràmentres per al kernel `anovadot`.

features Nombre de components principals a retornar per la funció. Per defecte està fixat a 0, indicant que es retornaran totes les components principals.

th Valor dels valors propis a partir del qual les components principals s'ignoren.

La funció retorna un objecte del tipus S4 que conté els vectors de les components principals i els valors propis de la següent forma:

pcv Matriu amb els vectors de les components principals.

eig Valors propis corresponents a les components principals.

rotated Les coordenades dels punts en les components principals.

xmatrix Matriu de les dades originals.

Per tal d'assimilar com funciona la tècnica del Kernel PCA, a continuació s'aplica a un petit exemple de joguina amb dades simulades. En concret es tracta d'un conjunt de dades que té 18 punts i 6 variables, o més formalment, es té un conjunt X de $n = 18$ individus on $X = (\mathbf{x}_1, ..., \mathbf{x}_{18})$ de forma que cada $\mathbf{x_i}$ és un element del conjunt $\mathcal{X} = \mathbb{R}^6$. Així doncs, cada individu està representat per un vector de sis variables numèriques. Les coordenades dels punts s'han seleccionat per tal de distingir clarament tres grups de la següent forma:

- El grup 1 té 6 punts tals que la suma de les coordenades en X_1 i X_2 és igual a 15 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals la suma de les coordenades en X_3, X_4 i X_5 és 0, mentre que aquesta suma val 6 per als altres 3 punts. Així doncs, les coordenades dels punts del primer grup són:

	x1	x2	x3	x4	x5
1	7.5	7.5	1	0	-1
2	6.0	9.0	0	-1	1
3	9.0	6.0	-1	1	0
4	7.5	7.5	3	2	1
5	6.0	9.0	2	1	3
6	9.0	6.0	1	3	2

- El grup 2 té 6 punts tals que la suma de les coordenades en X_3, X_4 i X_5 és 0 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals la suma de les coordenades en X_1 i X_2 és igual a 0, i per als altres 3 punts val -4. Les coordenades dels punts del segon grup són:

	x1	x2	x3	x4	x5
7	0	0	1	0	-1
8	1	-1	0	-1	1
9	-1	1	-1	1	0
10	-2	-2	1	0	-1
11	-1	-3	0	-1	1
12	-3	-1	-1	1	0

- Per últim, el grup 3 té 6 punts tals que la suma de les coordenades en X_1 i X_2 és igual a 0 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals la suma de les coordenades en X_3, X_4 i X_5 és igual a 15, i per als altres 3 punts val 24. Els punts del tercer grup són:

```
   x1 x2  x3  x4  x5
13  1 -1 6.0 4.5 4.5
14  1 -1 4.5 6.0 4.5
15 -1  1 4.5 4.5 6.0
16  0  0 9.0 7.5 7.5
17  1 -1 7.5 9.0 7.5
18 -1  1 7.5 7.5 9.0
```

A totes les coordenades dels punts s'aplica una perturbació afegint un soroll gaussià dèbil, per tal d'introduir una petita variabilitat dins de cada grup. Per últim, la variable X_6 s'assigna independentment del grup de forma aleatòria com una normal de mitjana 0 i desviació estàndard 1. Aquesta configuració dels punts és tal que al reduir la dimensió només caldran les primeres components per descobrir els tres grups. A continuació es té una mostra de les dades amb les que es treballarà:

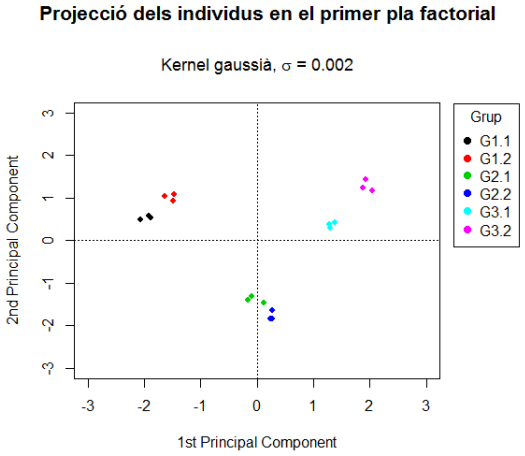
i	x1	x2	x3	x4	x5	x6
1	7.22	7.85	1.28	-0.11	-0.50	-1.29
⋮	⋮	⋮	⋮	⋮	⋮	
4	7.54	7.39	2.81	2.29	1.51	-0.61
⋮	⋮	⋮	⋮	⋮	⋮	
7	0.23	-0.31	0.37	0.19	-0.91	-1.69
⋮	⋮	⋮	⋮	⋮	⋮	
10	-2.22	-1.92	0.44	-0.51	-0.81	-1.14
⋮	⋮	⋮	⋮	⋮	⋮	
13	1.20	-0.79	6.39	4.72	4.39	-0.65
⋮	⋮	⋮	⋮	⋮	⋮	
16	0.89	0.44	8.99	8.53	7.72	-0.61
⋮	⋮	⋮	⋮	⋮	⋮	

Un cop descrites les dades, a continuació s'aplica la tècnica del Kernel PCA amb la funció `kpca` del paquet *kernlab* de R. S'escull el kernel gaussià fixant $\sigma = 0.002$, i es demana que es retornin les dues primeres components principals.

```
> kpc <- kpca(x, kernel="rbfdot", kpar=list(sigma=0.002), features=2)
```

Mitjançant `pcv(kpc)` es poden veure els dos vectors propis i amb `eig(kpc)` els dos valors propis. Les coordenades de les dades projectades en les components principals s'obtenen fent `rotated(kpc)`. Per últim, es poden representar els punts projectats en el pla format per les dues primeres components principals:

Figura 4.2: Kernel PCA amb el kernel gaussià



S'observa com els tres grups queden perfectament separats. El primer grup (negre i vermell) es troba a la part negativa del primer eix, el segon grup (verd i blau fosc) es troba a la part central del gràfic, i, per últim, el tercer grup (rosa i blau cel) es situa en la part positiva de les dues components principals. Així doncs:

Primera dimensió La primera dimensió permet separat els tres grups, situant el grup 1 a l'esquerra, el grup 2 al centre i el grup 3 a la dreta. De fet, aquesta dimensió també permet diferenciar els dos subgrups dins de cada grup.

Segona dimensió La segona dimensió separa el grup 2 dels altres dos grups, situant-lo en la part inferior mentre que els grups 1 i 3 es troben en la part superior.

A continuació es torna a aplicar la tècnica del Kernel PCA i es representen els individus en el primer pla factorial, però utilitzant kernels diferents per veure com canvien els resultats. Es prova amb els següents kernels:

Gràfic (a) La primera gràfica és el resultat de l'aplicació del kernel lineal (`vanilladot`). Com que aquest kernel no té cap paràmetre, a l'argument `kpar` se li ha assignat una llista buida. El resultat és equivalent a si s'hagués aplicat la tècnica clàssica del PCA.

```
> kpc_lineal <- kpca(x, kernel="vanilladot", kpar=list(), features=2)
```

Gràfic (b) El segon gràfic es correspon amb l'ús d'un kernel polinomial. En aquest cas s'ha fixat el grau del polinomi a 3, l'escala a 1 i l'offset a 0.

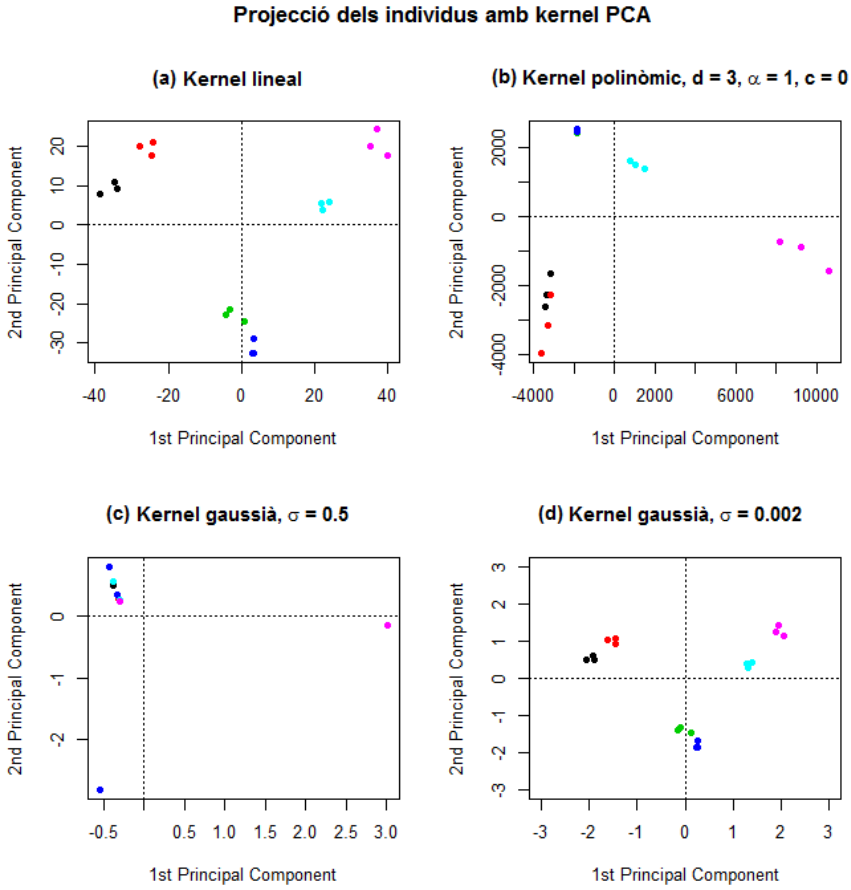
```
> kpc_poly <- kpca(x, kernel="polydot", kpar = list(degree=3,
+                  scale=1, offset=0), features=2)
```

Gràfic (c) El tercer gràfic es correspon amb l'ús d'un kernel gaussià fixant $\sigma = 0.5$.

```
> kpc_rbf <- kpca(x, kernel="rbfdot", kpar=list(sigma=0.5), features=2)
```

Gràfic (d) Per últim, el quart gràfic és el corresponent al kernel gaussià amb $\sigma = 0.002$, l'ús del qual ja s'ha vist que permet separar els grups de les dades.

Figura 4.3: Kernel PCA amb diversos kernels



D'aquestes gràfiques es destaquen alguns comentaris:

Kernel lineal Amb l'anàlisi de components principals lineal ja es troba una bona separació dels grups. La representació dels individus és molt semblant a la trobada amb el kernel gaussià ($\sigma = 0.002$). Cal dir que en aquest cas es tracta d'un exemple especialment pensat per a que els punts es separin bé, pel que amb una tècnica lineal ja es troba la separació.

Kernel polinòmic S'observa com s'ha obtingut una representació dels individus totalment diferent. Amb aquest kernel els individus ja no queden ben separats per grups. El primer grup sí es diferencia de la resta, però sembla com una part dels punts del grup 3 (blau cel) quedi més a prop del grup 2 que de la resta de punts del grup 3 (rosa).

Kernel gaussià El kernel gaussià s'ha utilitzat amb dos paràmentres diferents, i es pot observar la gran diferència entre les dues representacions. Amb $\sigma = 0.5$ els individus no queden gens diferenciats per grups, cosa que sí que passa amb $\sigma = 0.002$.

Valors de les projeccions Observant els rangs de valors dels eixos també es veuen diferències entre els kernels. Amb el kernel gaussià les coordenades dels punts en el primer eix prenen valors petits (entre -3 i 3 aproximadament). En canvi, amb el kernel lineal els valors d'aquestes coordenades oscil·len entre -40 i 40, i en el cas del kernel polinòmic el rang passa a estar entre -4 000 i 10 000.

La conclusió més important a extreure d'aquests resultats és que l'ús d'un kernel o un altre i el valor del paràmetre fa variar molt els resultats, pel que és una decisió crítica a prendre. No cal perdre de vista ni les dades ni l'objectiu de l'anàlisi.

Millora de la interpretabilitat del Kernel PCA amb R

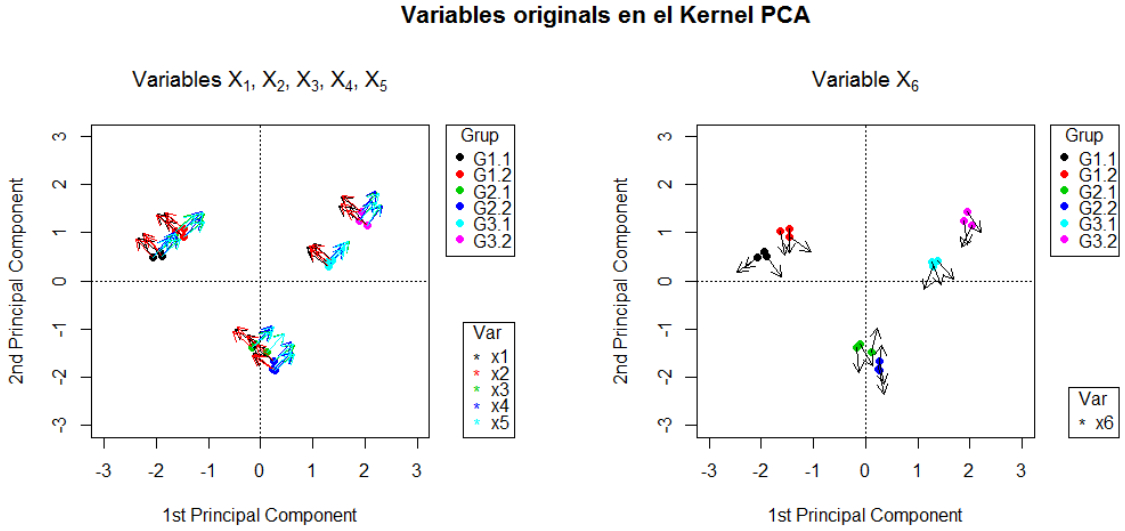
El conjunt de procediments desenvolupats per Reverter et al. (2014) per a millorar la interpretabilitat del Kernel PCA es pot trobar al paquet *KPCAvis* de R. Qüestions sobre la implementació d'aquests mètodes es descriuen de forma més detallada al capítol 6. En aquest apartat es mostren els resultats dels procediments aplicats sobre el conjunt de dades simulat, per tal d'entendre com funcionen de forma més visual i d'entendre la millora que aporten a la tècnica del Kernel PCA.

Per a totes les representacions que es mostren a continuació s'ha utilitzat el kernel gaussià amb $\sigma = 0.002$, que ja s'ha vist anteriorment que dóna lloc a una molt bona separació dels individus en els tres grups definits.

Representació de les variables originals

En primer lloc es mostra la representació de les variables originals $X_1, \ldots, X_6$ del conjunt de dades. En particular, les variables es representen per vectors que indiquen la direcció de màxim creixement a nivell local, apuntant cap els grups d'individus que tenen valors superiors en cada variable. En la figura de l'esquerra es representen les variables X_1 a X_5 i en el gràfic de la dreta es té la variable X_6.

Figura 4.4: Kernel PCA amb representació de les variables originals



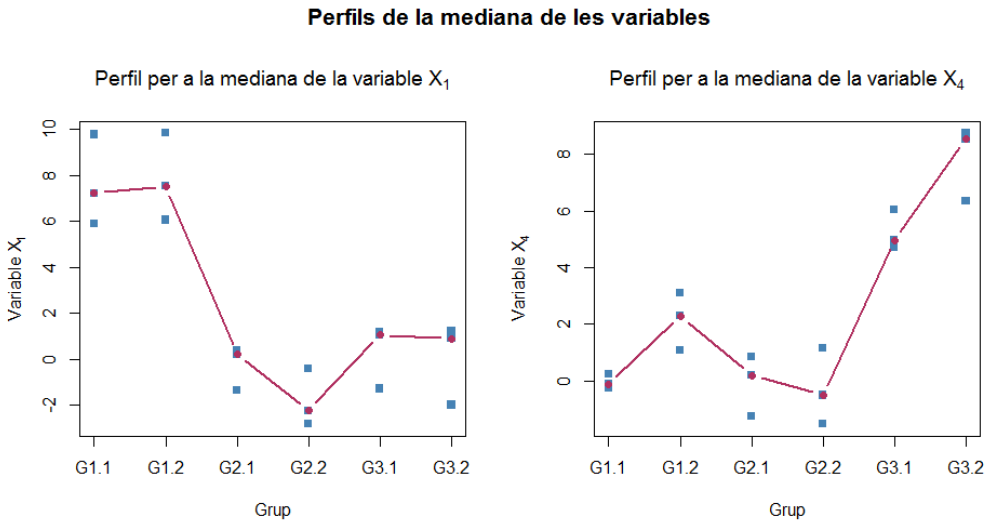
D'aquesta representació de les variables originals s'observa:

- En la figura de l'esquerra les variables X_1 i X_2 apunten cap al grup 1, mentre que les variables X_3, X_4 i X_5 apunten cap al grup 3, que és el que s'espera donada la construcció dels punts. Es recorda que les coordenades per a cadascun dels punts del grup 1 per a les variables X_1 i X_2 sumen 15, mentre que per a la resta de punts sumen 0 o -4. I en el cas de les variables X_3, X_4 i X_5, les coordenades per a cadascun dels punts del grup 3 sumen 15 o 24, mentre que per a la resta de punts sumen 6 o 0.

- En la figura de la dreta està representada la variable X_6, que s'ha construït com una normal aleatòria i de forma independent als grups d'individus. És per això que en el gràfic s'observa com la representació no apunta cap a ningun grup en particular, sinó que per a individus del mateix grup apunta cap a direccions diferents.

Per tal de corroborar els resultats anteriors, el paquet *KPCAvis* permet la representació dels perfils de les medianes de les variables agrupades segons els grups definits en els individus. Per exemple, els perfils per a les variables X_1 i X_4 són:

Figura 4.5: Perfils de la mediana de les variables X_1 i X_4

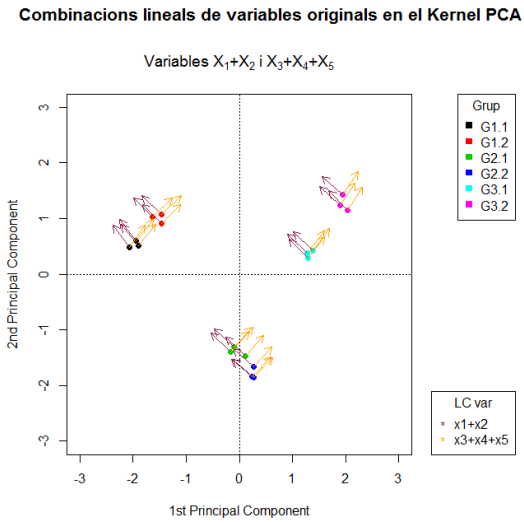


S'observa com la mediana de la variable X_1 és major en el grup 1 respecte als altres dos grups, mentre que per a la variable X_4 la mediana és major en el grup 3 que en la resta. Això confirma el que s'ha vist amb la representació de les variables.

Representació de combinacions lineals de les variables originals

En el següent pas natural és representar en comptes d'una variable una combinació lineal de les variables originals. En aquest cas, es representen les varibles $X_1 + X_2$ i $X_3 + X_4 + X_5$. Es tindrà en cada individu un vector que apunti a la direcció de màxim creixement de cadascuna de les dues combinacions lineals. El resultat es mostra en el següent gràfic.

Figura 4.6: Kernel PCA amb representació de combinacions lineals de variables originals



Es pot observar com en cada punt els vectors apunten cap els grups amb valors més

alts de les combinacions lineals representades. Així doncs, els vectors que representen la variable $X_1 + X_2$ apunten cap al grup 1 (negre i vermell), mentre que els vectors que representen la variable $X_3 + X_4 + X_5$ apunten cap al grup 3 (rosa i blau cel).

Integració de dades i representació de les variables originals

La següent característica del paquet *KPCAvis* que es posa en pràctica és la integració de diferents fonts d'informació juntament amb la representació de les variables originals, que poden pertànyer a qualsevol de les fonts. Per fer-ho, es crea un nou exemple de joguina amb dades simulades, de forma semblant al conjunt $\mathbf{X}$. Aquest conjunt de dades $\mathbf{Y}$ té 18 punts i 6 variables, igual que el primer. Les coordenades dels punts s'han seleccionat per tal de distingir els mateixos grups d'individus, però ara les relacions entre les coordenades no són combinacions lineals, sinó que són productes. Es té:

- El grup 1 té 6 punts tals que el producte de les coordenades en Y_1 i Y_2 és igual a 2 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals el producte de les coordenades en Y_3, Y_4 i Y_5 és 1, mentre que aquest producte val 3 per als altres 3 punts.

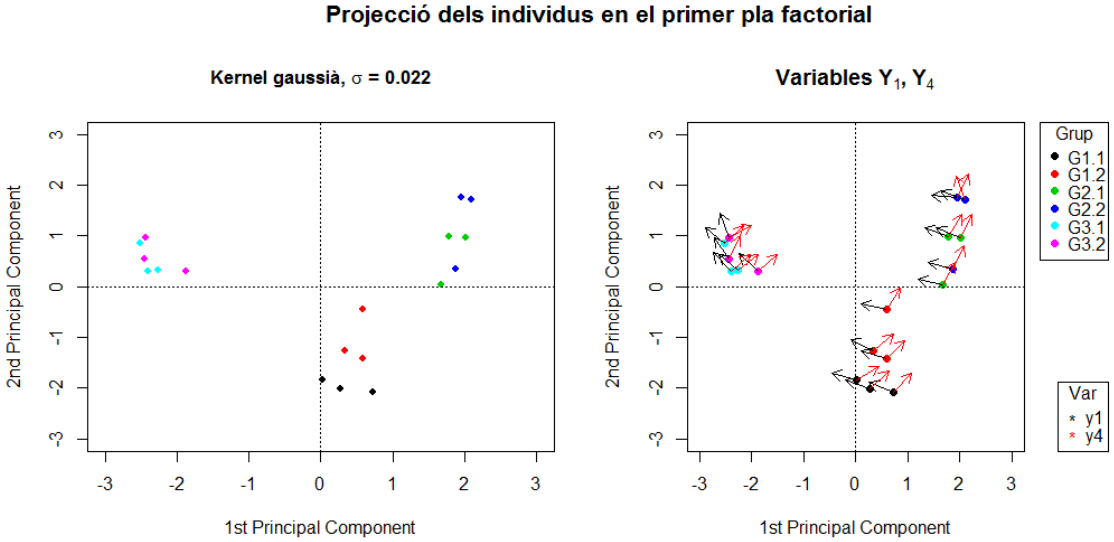
- El grup 2 té 6 punts tals que el producte de les coordenades en Y_1 i Y_2 és igual a 1 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals el producte de les coordenades en Y_3, Y_4 i Y_5 és 36, mentre que aquest producte val 3 per als altres 48 punts.

- Per últim, el grup 3 té 6 punts tals que el producte de les coordenades en Y_1 i Y_2 és igual a 20 per a cada punt. A més, en aquest grup, hi ha 3 punts pels quals el producte de les coordenades en Y_3, Y_4 i Y_5 és igual a 2, i per als altres 3 punts val 4.

Igual que per al $\mathbf{X}$, a totes les coordenades dels punts s'aplica una perturbació afegint un soroll gaussià dèbil, per tal d'introduir una petita variabilitat dins de cada grup. Per últim, la variable Y_6 s'assigna independentment del grup de forma aleatòria com una normal de mitjana 0 i desviació estàndard 1.

Tal i com estan creats els punts, el que s'espera quan es representin les variables és que les fletxes de les variables Y_1 i Y_2 apuntin al grup 3 i les fletxes de les variables Y_3, Y_4 i Y_5 apuntin cap el grup 2. Un cop realitzat el Kernel PCA, en els següents dos gràfics es mostra, per un costat, la representació dels punts amb un kernel gaussià ($\sigma = 0.022$), i per una altra banda la representació de les variables Y_1 i Y_4:

Figura 4.7: Kernel PCA amb representació de les variables originals

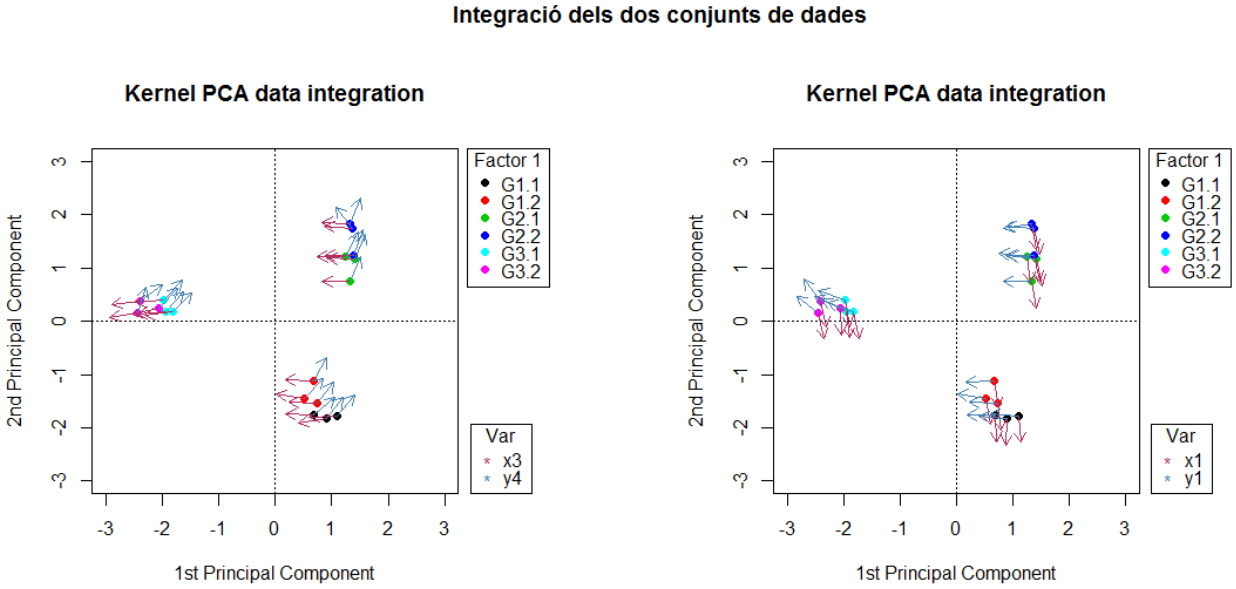


S'observa:

- Els tres grups d'individus queden clarament separats per la primera dimensió, igual que per a l'anterior conjunt. Però les posicions són diferents: el grup 3 queda a l'esquerra, el grup 1 al centre, i el grup 2 a la dreta. A més, hi ha més variabilitat en la representació dels punts dins de cada grup, estan més separats.
- Tal i com era d'esperar, la variable Y_1 apunta cap al grup 3 (a l'esquerra), mentre que la variable Y_4 apunta cap al grup 2 (a la dreta).

El que permet el Kernel PCA és integrar els dos conjunts $\mathbf{X}$ i $\mathbf{Y}$, i mitjançant els procediments del paquet *KPCAvis*, es poden representar les variables de qualsevol dels dos conjunts. Per a cada conjunt es trien les mateixes σ que abans (0.002 per al conjunt $\mathbf{X}$ i 0.022 per al conjunt $\mathbf{Y}$). La representació dels individus en el primer pla factorial juntament amb algunes variables es mostra en els següents dos gràfics.

Figura 4.8: Integració de dades amb Kernel PCA i representació de les variables originals



S'observa:

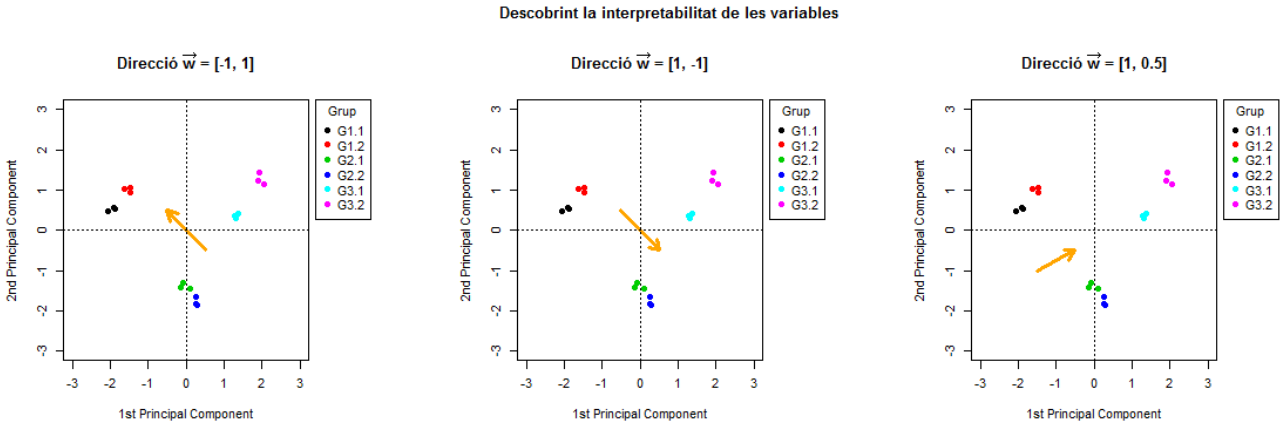
- Els tres grups d'individus queden clarament separats, i en les mateixes posicions que quan es feia el Kernel PCA només del conjunt $\mathbf{Y}$. És a dir, el grup 3 queda a l'esquerra, el grup 1 al centre, i el grup 2 a la dreta. A més, dins de cada grup els punts queden més junts, sense la variabilitat en la representació del conjunt $\mathbf{Y}$, sinó més semblants a quan es feia el Kernel PCA només del conjunt $\mathbf{X}$.
- En el gràfic de l'esquerra s'ha representat la variable X_3 del conjunt $\mathbf{X}$ juntament amb la variable Y_4 del conjunt $\mathbf{Y}$. Les direccions de les variables es mantenen un cop feta la integració. Així doncs, mentre que X_3 apunta cap al grup 3, Y_4 ho fa cap al grup 2.
- En el gràfic de la dreta s'han representat les variables X_1 i Y_1. La primera apunta cap al grup 1, mentre que la segona ho fa cap al grup 3, les mateixes direccions que quan es fa el Kernel PCA per separat per a cada conjunt.

Descobrir la interpretació de les variables originals

Per últim, en comptes de representar una de les variables originals, es fixa una direcció $\mathbf{w}$ en el pla i es busquen les variables originals correlacionades amb aquesta direcció. Aquest procediment és el que té més utilitat pràctica, ja que ajuda a l'investigador a trobar quines variables són les que sembla que diferencien els grups d'individus i així generar noves hipòtesis de treball futur. El resultat és una mesura de la força de la correlació entre les variables originals i la direcció fixada.

Per a il·lustrar el procediment, es fixen en el primer pla factorial tres direccions, cada una apuntant a un dels tres grups definits en les dades. Aquestes són:

Figura 4.9: Kernel PCA i interpretació de les variables originals



Els resultats obtinguts són:

Gràfic 1: Direcció $\vec{w}$ = [-1, 1] Aquesta direcció apunta cap al grup 1. La mesura de la força de la correlació entre aquesta direcció i les variables és:

	Var	Mean	Sd
1	x1	0.9955	0.0050
2	x2	0.9971	0.0041
3	x3	0.0554	0.0852
4	x4	0.1067	0.0840
5	x5	0.0364	0.0866
6	x6	-0.5534	0.5297

Tal i com és d'esperar, les variables X_1 i X_2 són les que estan correlacionades amb la direcció marcada, indicant que el grup 1 presenta valors més alts en aquestes dues variables.

Gràfic 2: Direcció $\vec{w}$ = [1, -1] Aquesta direcció apunta cap al grup 2 i en sentit contrari al grup 1. La mesura de la força de la correlació entre aquesta direcció i les variables és:

	Var	Mean	Sd
1	x1	-0.9955	0.0050
2	x2	-0.9971	0.0041
3	x3	-0.0554	0.0852
4	x4	-0.1067	0.0840
5	x5	-0.0364	0.0866
6	x6	0.5534	0.5297

La direcció marcada és inversa a la primera direcció, pel que els resultats obtinguts són els mateixos però amb el signe invers. El que està indicant és que les variables X_1 i X_2 estan correlacionades negativament amb la direcció marcada, indicant que el grup 1 presenta valors més alts en aquestes dues variables.

Gràfic 3: Direcció $\vec{\mathbf{w}}$ = [1, 0.5] Aquesta direcció apunta cap al grup 3. La mesura de la força de la correlació entre aquesta direcció i les variables és:

	Var	Mean	Sd
1	x1	-0.3529	0.0837
2	x2	-0.3314	0.0718
3	x3	0.9264	0.0324
4	x4	0.9063	0.0343
5	x5	0.9331	0.0331
6	x6	-0.1917	0.4912

Per últim, les variables X_3, X_4 i X_5 són les que estan correlacionades amb la direcció $\mathbf{w}$, indicant que el grup 3 presenta valors més alts en aquestes tres variables, fet que es correspon totalment a com s'han construït els individus.

Aplicant el Kernel PCA i els procediments de visualització de variables als dos exemples de joguina s'han pogut comprovar les característiques i propietats d'aquestes tècniques explicades en l'apartat anterior.

Capítol 5

APLICACIÓ DEL KERNEL PCA A DADES ÒMIQUES

En el capítol anterior s'ha vist en detall la tècnica del Kernel PCA i s'ha aplicat a conjunts de dades simulades. En aquest capítol s'apliquen els mètodes vistos a dos conjunts de dades òmiques. En el primer cas, les dades provenen d'un estudi nutrigenòmic realitzat en ratolins, pel que es disposa d'un bloc de dades metabolòmiques i d'un altre bloc de dades transcriptòmiques. En el segon cas, les dades analitzades combinen l'expressió gènica i la hibridació genòmica comparativa mesurades en nens amb glioma.

5.1 *Nutrimouse dataset*

Les dades procedeixen d'un estudi de nutrigenòmica en ratolins (Martin *et al.*, 2007) [27] en el qual es van considerar els efectes de cinc dietes amb composicions d'àcids grassos contrastades sobre els lípids hepàtics i l'expressió gènica hepàtica en ratolins. L'estudi va proporcionar dos conjunts de variables mesurades en quaranta ratolins:

Transcriptòmic Expressions de 120 gens mesurats en cèl·lules del fetge, seleccionats (entre uns 30 000) com potencialment rellevants en el context de l'estudi de la nutrició. Aquestes expressions van ser mesurades per un *macrochip* de niló amb etiquetatge radioactiu.

Lipídic Concentracions (en percentatge) de 21 àcids grassos hepàtics mesurats per cromatografia de gasos.

Les unitats biològiques (els ratolins) van ser classificades en funció de dos factors de disseny experimental:

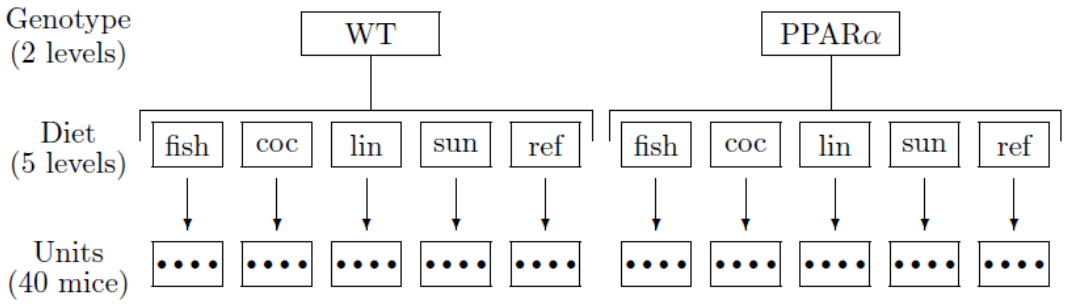
Genotip Factor de dos nivells, els quals són el genotip *wild-type* (WT) i el genotip PPARα -/- (PPAR).

Dieta Factor de cinc nivells. Els olis utilitzats per a la preparació de les dietes experimentals van ser:

- Olis de blat de moro i de colza (50/50) per a la dieta de referència (REF).
- Oli hidrogenat de coco per a una dieta d'àcids grassos saturats (COC).
- Oli de gira-sol per a una dieta rica en àcids grassos Omega 6 (SUN).
- Oli de llinosa per a una dieta rica en Omega 3 (LIN).
- Olis de blat de moro, de colza i de peix enriquits (43/43/14) per a la dieta de peix (FISH).

El disseny experimental de l'estudi es resumeix en la següent figura:

Figura 5.1: Disseny experimental per a l'estudi *nutrimouse*



El conjunt de dades *nutrimouse* va ser proporcionat per Pascal Martin, del Laboratori de toxicologia i farmacologia de l'Institut Nacional d'Investigació Agronòmica de França. Aquest conjunt de dades es pot trobar al paquet `mixOmics` de R (veure [5]). És un *dataset* molt utilitzat en la literatura per a il·lustrar diverses tècniques d'integració, com l'anàlisi factorial múltiple (MFA) o l'anàlisi de correlacions canòniques regularitzat (rCCA).

En aquest estudi no es sabia a priori si els canvis en l'expressió gènica implicaven canvis en les concentracions d'àcids grassos o viceversa. L'objectiu era generar hipòtesis interessants sobre els camins dels factors de transcripció que potencialment enllacen els àcids grassos hepàtics i l'expressió gènica.

5.1.1 Anàlisi de l'expressió gènica

El primer *dataset* a analitzar és el de les expressions dels gens. Aquest està format per les expressions de 120 gens en 40 ratolins, pel que formalment es té un conjunt X de $n = 40$ individus on $X = (\mathbf{x}_1, ..., \mathbf{x}_{40})$, de forma que cada $\mathbf{x_i}$ és un element del conjunt $\mathcal{X} = \mathbb{R}^{120}$. Així doncs, cada individu està representat per un vector de 120 variables numèriques.

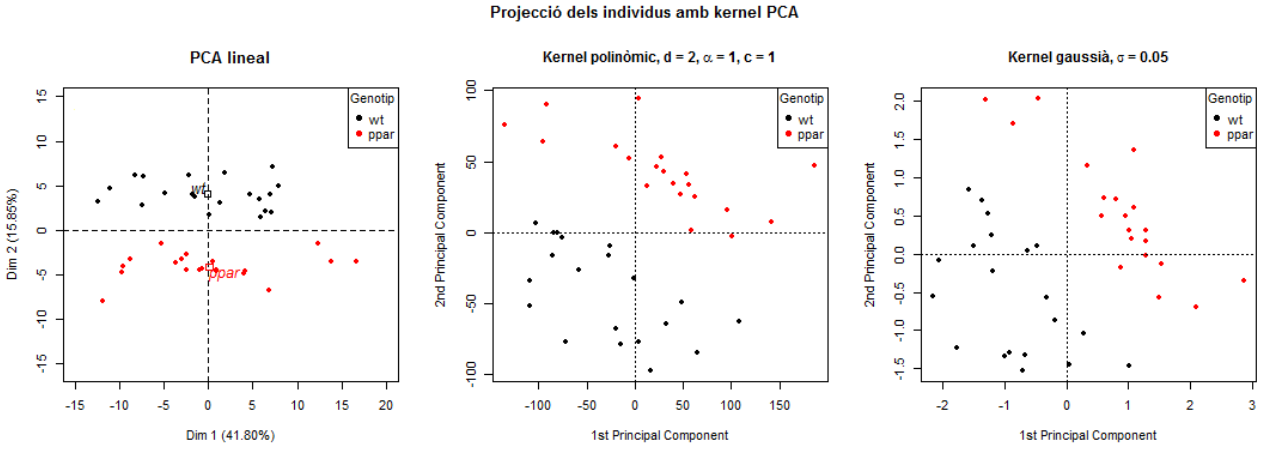
Per tal d'analitzar aquest conjunt de dades, primer s'aplica el Kernel PCA fent diverses proves per tal de triar el valor dels paràmetres adequat. Posteriorment, s'aplica el procediment de Reverter et al. (2014) per a representar alguns dels gens disponibles.

1. Aplicació del Kernel PCA

Primer s'ha aplicat la tècnica del Kernel PCA utilitzant diversos kernels, per veure com canvien les representacions dels individus. En concret, s'ha utilitzat el kernel lineal (PCA clàssic), el kernel polinòmic (grau = 2, escala = 1, *offset* = 1) i el kernel gaussià ($\sigma = 0.05$). El disseny de l'experiment permet dividir els individus segons dos condicions: segons el genotip del ratolí i segons la dieta. Per tal de visualitzar millor els resultats, es representen els individus amb colors diferents segons cadascun dels dos factors.

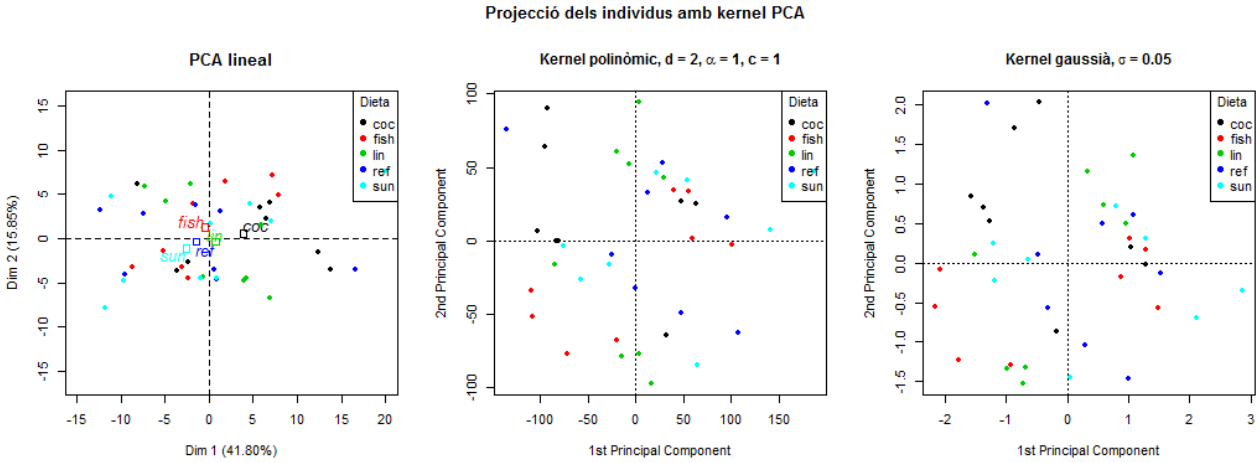
Representació dels individus segons el factor genotip Tal i com es mostra en el gràfic, amb els tres kernels s'aconsegueix una separació lineal dels individus en funció del seu genotip. En el PCA lineal, és la segona dimensió la que separa els grups, quedant el del genotip WT a la part superior i el del genotip PPAR a sota. En el Kernel PCA amb kernel polinòmic la disposició dels punts segons el factor és la inversa. I amb el Kernel PCA amb kernel gaussià, els ratolins amb genotip PPAR queden en la part triangular superior del gràfic mentre que els del genotip WT queden en la part oposada, de forma semblant al kernel polinòmic.

Figura 5.2: *Nutrimouse genes*: Representació dels individus segons el genotip



Representació dels individus segons el factor dieta Al contrari que amb el genotip, cap dels tres kernels aconsegueix una separació clara dels individus en funció de la dieta. No hi ha cap representació dels individus que estigui agrupada per a alguna dieta, tal i com s'observa en la figura 5.3. S'ha provat canviant el grau del kernel polinòmic i canviant la σ del kernel gaussià, però només s'aconsegueix separar els individus per genotip.

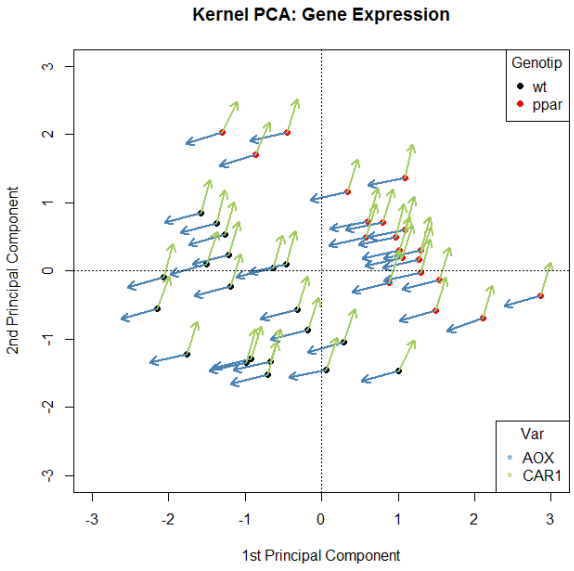
Figura 5.3: *Nutrimouse genes*: Representació dels individus segons la dieta



2. Representació de les variables originals (gens)

Un cop aplicat el Kernel PCA i triat el valor de la σ del kernel gaussià ($\sigma = 0.05$), s'ha aplicat el procediment de Reverter et al. (2014) per a representar les variables originals, en aquest cas els gens. En concret es representen dos gens: *AOX* (en color blau) i *CAR1* (en color verd).

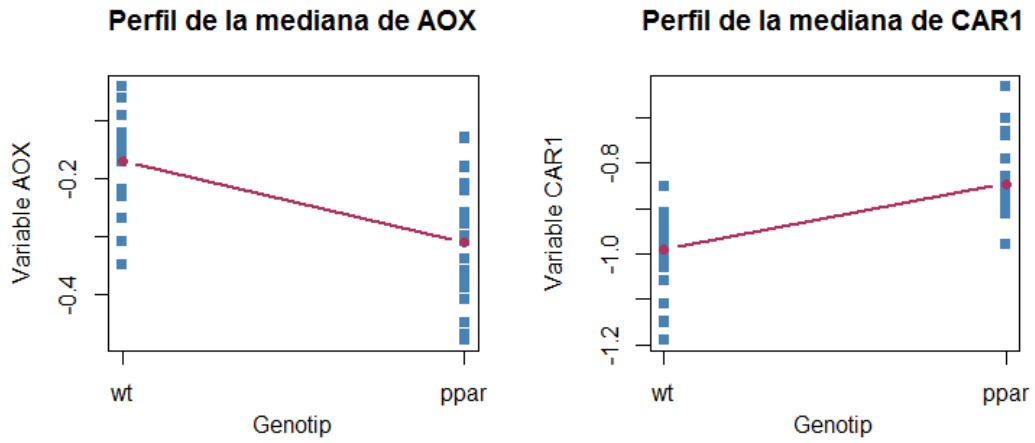
Figura 5.4: *Nutrimouse*: Representació dels gens *AOX* i *CAR1*



Els vectors representats indiquen la direcció de màxim creixement de l'expressió gènica de cada gen en cada un dels ratolins de la mostra. S'observa com per al gen *AOX* les fletxes apunten cap al grup de ratolins amb genotip WT, indicant que en els ratolins amb aquest genotip el gen *AOX* s'expressa més que en el grup amb el genotip PPAR. Per contra, per al gen *CAR1* les fletxes apunten cap al grup de ratolins amb genotip PPAR,

indicant que l'expressió d'aquest gen és més alta si es té el genotip PPAR. Per tal de corroborar aquests resultats es mostra el perfil de la mediana per als dos gens segons el genotip:

Figura 5.5: *Nutrimouse*: Perfil de la mediana dels gens *AOX* i *CAR1*



5.1.2 Anàlisi de les dades d'àcids grassos

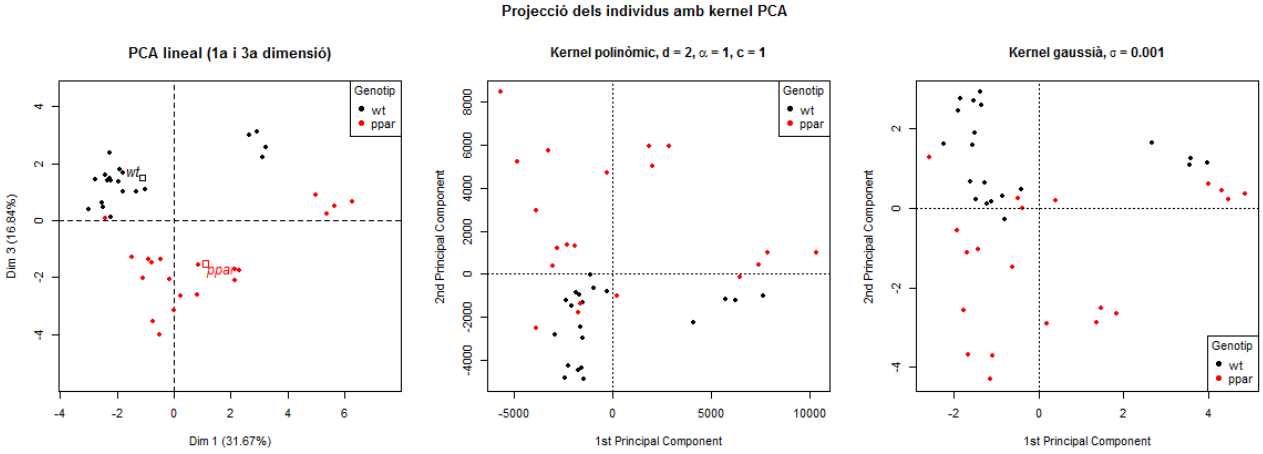
El segon *dataset* a analitzar és el que conté dades metabolòmiques. Aquest està format per les concentracions de 21 àcids grassos hepàtics en els mateixos 40 ratolins, pel que formalment es té un conjunt Y de $n = 40$ individus on $Y = (\mathbf{y}_1, ..., \mathbf{y}_{40})$, de forma que cada $\mathbf{y_i}$ és un element del conjunt $\mathcal{Y} = \mathbb{R}^{21}$. Seguint el mateix esquema que per al conjunt genòmic, primer s'aplica el Kernel PCA i després es representen alguns àcids grassos.

1. Aplicació del Kernel PCA

S'ha utilitzat el kernel lineal (PCA clàssic), el kernel polinòmic (grau = 2, escala = 1, *offset* = 1) i el kernel gaussià ($\sigma = 0.001$), donant com a resultat segons cadascun dels dos factors les següents representacions dels ratolins.

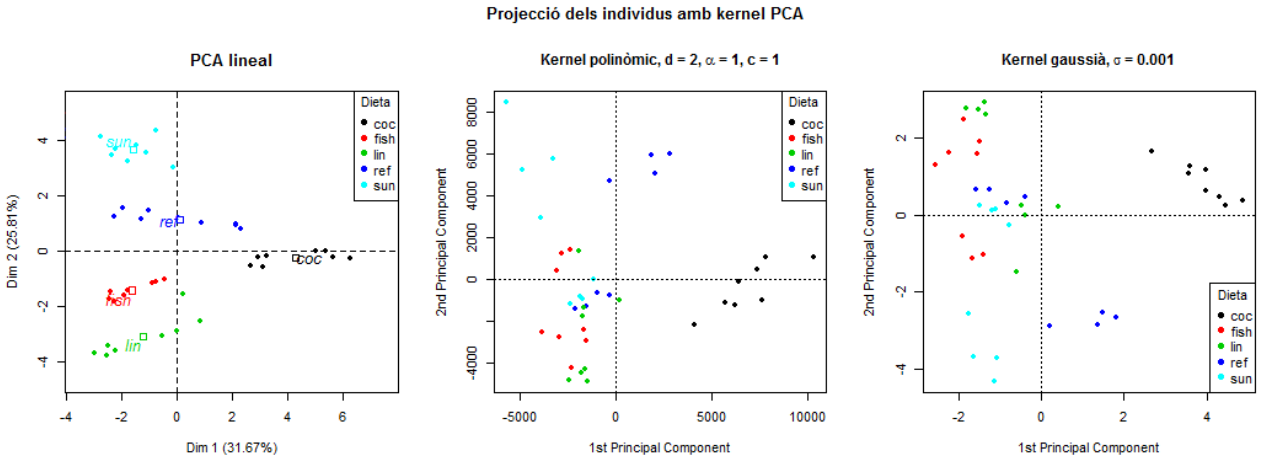
Representació dels individus segons el genotip Amb les dades dels àcids grassos també s'aconsegueix separar als individus segons el genotip, tot i que no de forma tan clara com amb les dades genòmiques. El PCA clàssic mostra els individus representats en la primera i tercera dimensió, ja que s'ha trobat que la separació és més clara que no pas en el primer pla factorial. En el Kernel PCA amb kernel polinòmic la separació dels punts és menys clara, quedant alguns ratolins amb genotip PPAR barrejats en el grup del genotip WT. I amb el Kernel PCA amb kernel gaussià, la segona dimensió és la que separa bastant clarament els dos grups, quedant el del genotip WT a la part superior i el del genotip PPAR a sota.

Figura 5.6: *Nutrimouse fatty acids*: Representació dels individus segons el genotip



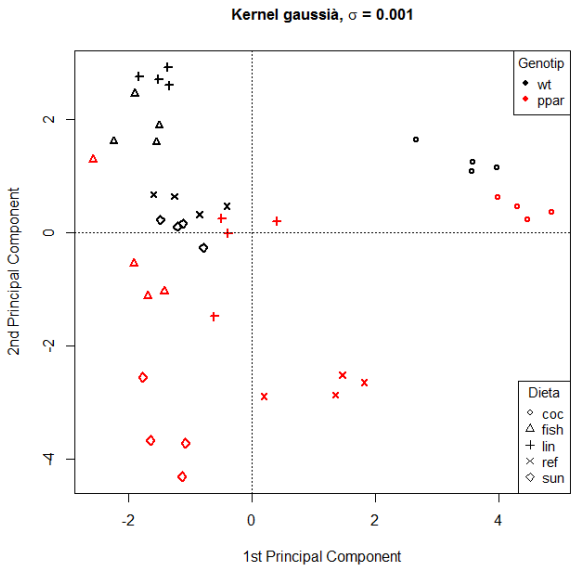
Representació dels individus segons el factor dieta Al contrari que amb el conjunt genòmic, amb les dades d'àcids grassos sí s'aconsegueix trobar grups d'individus segons la dieta. El PCA clàssic és el que permet una separació més clara, quedant cinc grups ben definits. Amb el Kernel PCA amb kernels polinòmic i gaussià el que s'aconsegueix és separar els ratolins amb dieta COC (a la dreta) de la resta.

Figura 5.7: *Nutrimouse fatty acids*: Representació dels individus segons la dieta



Per últim, es mostra en la figura 5.8 la representació dels individus amb Kernel PCA i escollint el kernel gaussià ($\sigma = 0.001$) diferenciant als ratolins segons els dos factors a la vegada (per color segons el genotip i per símbol segons la dieta).

Figura 5.8: *Nutrimouse fatty acids*: Representació dels individus segons el genotip i la dieta

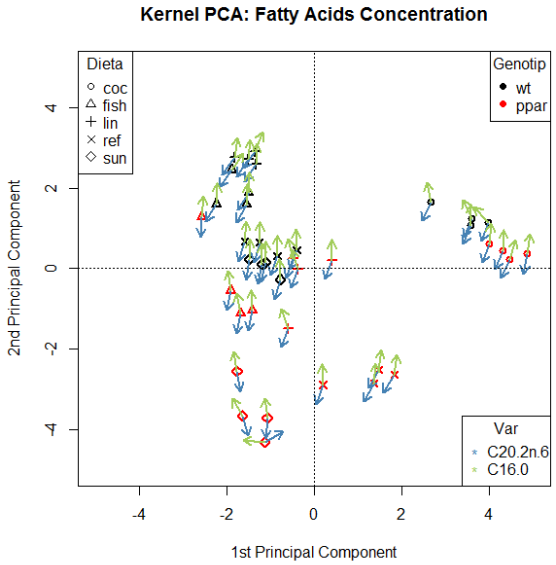


Amb aquesta representació es pot donar noms als dos primers factors. S'observa com la primera dimensió separa la dieta COC de la resta de dietes, mentre que amb la segona dimensió els ratolins queden bastant diferenciats segons el tipus de genotip. És en aquest gràfic on s'afegeix la representació de les variables originals.

2. Representació de les variables originals (àcids grassos)

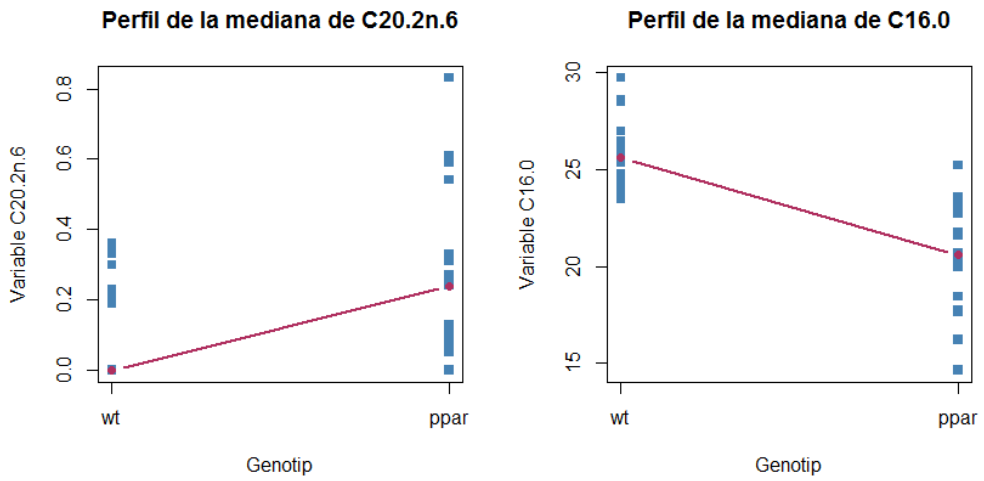
En el cas del conjunt metabolòmic, en primer lloc s'han representat dos àcids grassos: el *C20.2ω.6* (en color blau) i el *C16.0* (en color verd).

Figura 5.9: *Nutrimouse*: Representació dels àcids *C20.2ω.6* i *C16.0*



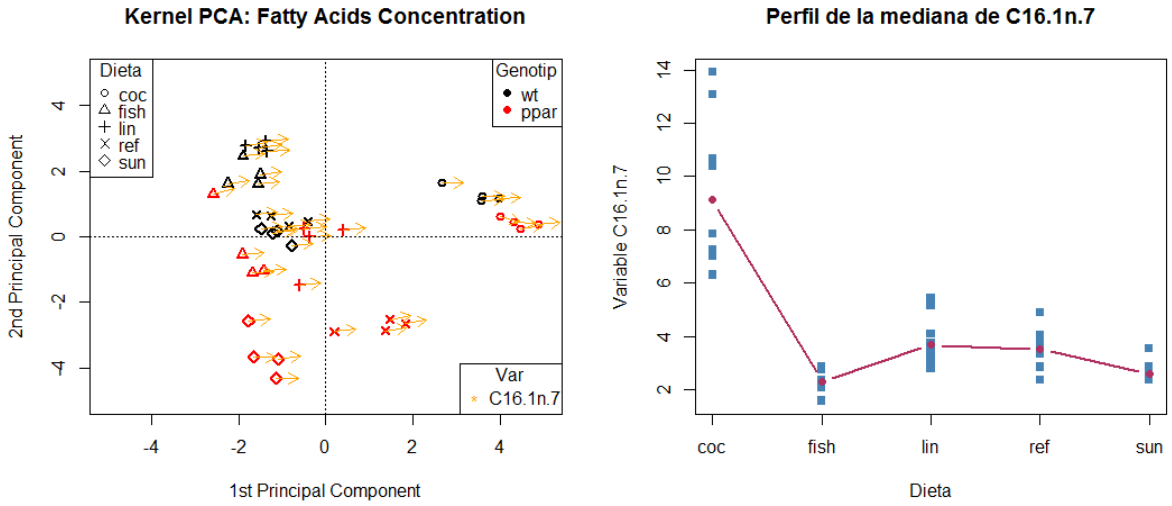
Els vectors representats indiquen la direcció de màxim creixement de la concentració dels dos àcids grassos en cada un dels ratolins de la mostra. S'observa com per al *C20.2ω.6* les fletxes apunten cap al grup de ratolins amb genotip PPAR, indicant que els ratolins amb aquest genotip tenen concentracions més altes d'aques àcid. També es veu com la concentració de *C16.0* és més alta en els ratolins amb genotip WT. Els perfils de les medianes per als dos àcids grassos estan en concordança amb la representació del Kernel PCA.

Figura 5.10: *Nutrimouse*: Perfil de la mediana dels àcids *C20.2ω.6* i *C16.0*



Per últim es mostra la representació de l'àcid *C16.1ω.7* juntament amb el perfil de la seva mediana segons la dieta. S'observa com les fletxes apunten cap al grup de ratolins amb la dieta COC, indicant que els ratolins amb aquesta dieta tenen una concentració més alta d'aquest àcid gras, cosa que es corrobora amb el perfil de la mediana.

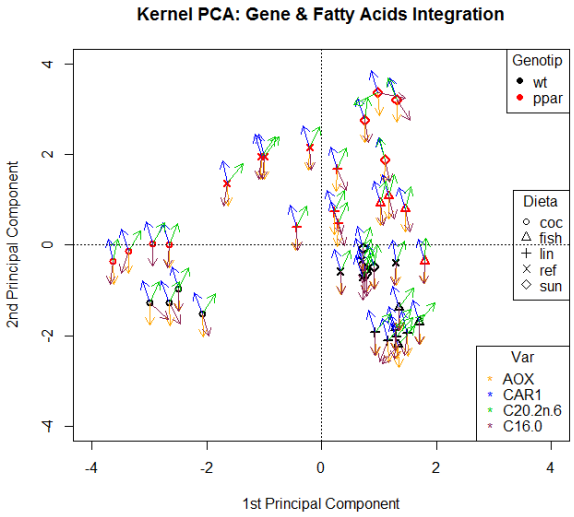
Figura 5.11: *Nutrimouse*: Representació de l'àcid *C16.1ω.7* i perfil de la seva mediana



5.1.3 Integració de les dades transcriptòmiques i metabolòmiques

Un cop estudiats per separat els dos conjunts de dades, es procedeix a la seva integració amb el Kernel PCA. Per a l'expressió gènica s'escull el kernel gaussià amb $\sigma = 0.05$ i per a les concetracions d'àcids grassos el kernel gaussià amb $\sigma = 0.001$. Es mostra la representació dels individus donada pel Kernel PCA afegint les variables dels gens *AOX* i *CAR1* i dels àcids grassos *C20.2ω.6* i *C16.0*.

Figura 5.12: *Nutrimouse data integration*: Representació dels individus amb Kernel PCA

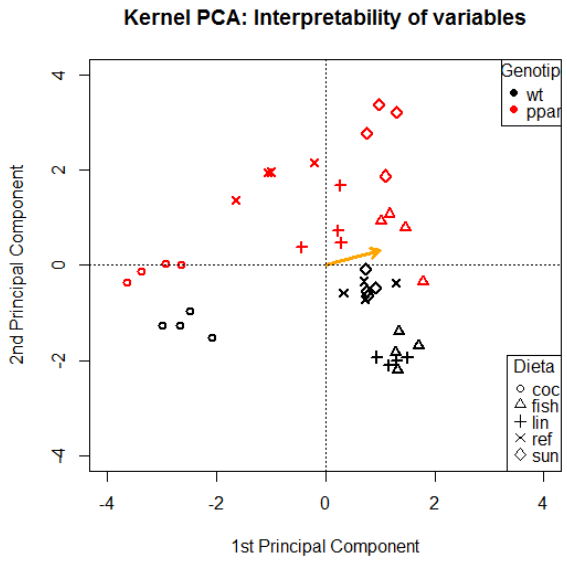


El fet d'analitzar els dos conjunts simultàneament permet obtenir una millor representació dels individus, ja que aquests es separen clarament per genotip i a més també es forma el grup de la dieta COC. La primera dimensió separa la dieta COC de la resta de dietes, situant-la a l'esquerra (en l'anàlisi dels àcids per separat quedava a la dreta), i amb la segona dimensió els ratolins queden ben diferenciats segons el genotip. Pel que fa a la representació de les variables, s'observa com el gen *CAR1* i l'àcid gras *C20.2ω.6* (fletxes blaves i verdes) apunten cap als ratolins amb genotip WT, mentre que les fletxes del gen *AOX* i l'àcid gras *C16.0* (taronges i liles) apunten cap als ratolins amb genotip PPAR. Les direccions trobades en els anàlisis individuals es mantenen en la integració.

5.1.4 Descobrint la interpretació de les variables

L'últim dels procediments de Reverter et al. (2014) que s'aplica és el que permet fixar una direcció en el pla i buscar les variables originals correlacionades amb aquesta direcció. La direcció escollida (que es mostra en la figura 5.13 amb el vector taronja) permet trobar les variables que estan més correlacionades amb aquesta direcció i per tant menys expressades en els ratolins amb dieta COC (ja que apunta en sentit contrari al grup COC).

Figura 5.13: *Nutrimouse*: Interpretació de les variables



El resultat que proporciona el procediment és la força de la correlació entre les variables originals (tant gens com àcids grassos) i la direcció fixada. Es mostren les tres variables amb una major correlació positiva i negativa per als dos conjunts de dades.

Taula 5.1: *Nutrimouse*: Correlacions entre les variables i una certa direcció

Gene	Mean	Sd
ACC2	-0.9986	0.0013
i.FABP	-0.9978	0.0036
COX1	-0.9951	0.0067
⋮	⋮	⋮
CYP3A11	0.1999	0.0305
SPI1.1	0.4352	0.0400
GSTpi2	0.6562	0.0349

Acid	Mean	Sd
C16.1ω.7	-0.9530	0.0874
C18.1ω.7	-0.9314	0.2628
C14.0	-0.9292	0.1182
⋮	⋮	⋮
C20.4ω.6	0.6997	0.3664
C22.5ω.6	0.7025	0.5070
C18.3ω.3	0.7196	0.5012

Els gens *ACC2*, *i.FABP* i *COX1* tenen una correlació negativa molt forta amb la direcció marcada, pel que es conclou que són gens amb alta expressió en els ratolins amb dieta COC. De forma anàloga, com que els àcids grassos *C16.1ω.7*, *C18.1ω.7* i *C14.0* estan també correlacionats negativament amb la direcció fixada, les seves concentracions en els ratolins amb aquesta dieta són majors. Es destaca que l'àcid *C16.1ω.7* ja s'havia representat en la figura 5.11, on apuntava cap al grup de la dieta COC, pel que els resultats són coherents en els dos procediments.

5.2 *Pediatric glioma data*

El segon conjunt de dades a analitzar està format per dades transcriptòmiques i dades d'alteracions del genoma mesurades en un conjunt de nens amb glioma, un tipus de tumor

cerebral. Les mostres de tumor es van obtenir de 53 nens amb diagnòstic recent de glioma pediàtric d'alt grau (*pediatric high-grade glioma - pHGG*) de l'hospital universitari Necker Enfants Malades, a París. Es tenen dos conjunts de variables:

Transcriptòmiques Expressions de 15702 gens, obtingudes amb *microarrays*.

Alteracions del genoma Conté els desequilibris de 1229 segments de cromosomes, obtinguts amb la tècnica de la hibridació genòmica comparativa (*Comparative genomic hybridization - CGH*).

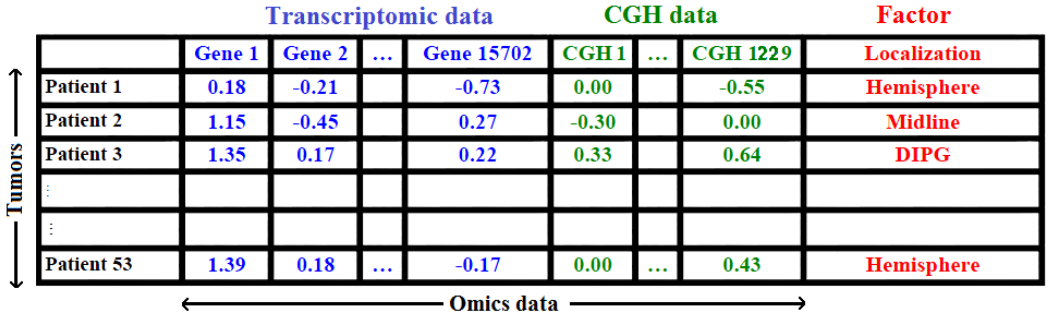
Totes les variables estan estandarditzades, de forma que tenen mitjana 0 i variància 1. A més, els 53 tumors cerebrals estan dividits en 3 localitzacions: supratentorial (HEMI), nucli central (MIDL) i tronc cerebral (DIPG). Les freqüències absolutes i relatives per a aquest factor es mostren en la següent taula:

Taula 5.2: Freqüències absolutes i relatives del factor localització

	n	%
HEMI	20	37.74
DIPG	22	41.51
MIDL	11	20.75
Total	53	100.00

Així doncs, es disposa d'un factor que classifica les observacions més dos blocs d'informació a integrar. Les dades, que es poden descarregar de `http://biodev.cea.fr/sgcca/` en forma de paquet de R, es resumeixen en la següent figura:

Figura 5.14: *Pediatric high-grade glioma*: dades disponibles



Background: Estudis sobre el pHGG i la seva ubicació

Els tumors cerebrals són els tumors sòlids més freqüents en nens i tenen la major taxa de mortalitat de tots els càncers pediàtrics. Malgrat els avenços en teràpies, els nens amb pHGG tenen una supervivència global del voltant del 20% als 5 anys. En funció de la seva ubicació, el glioma presenta característiques diferents en quant al seu aspecte radiològic i histològic, i el seu pronòstic.

Les dades a analitzar ja han estat utilitzades en els estudis de Puget et al. (2012) [35] i Tenenhaus et al. (2014) [44]. La hipòtesis plantejada és que si els gliomes tenen diferents orígens genètics i diferents vies oncogèniques en funció de la seva ubicació, els processos biològics implicats en el desenvolupament del tumor poden ser diferents d'un lloc a un altre. Amb una anàlisi integrada dels desequilibris de l'expressió gènica i cromosòmica es podrien identificar amb més precisió els processos biològics i els gens que diferencien aquests gliomes per a les diferents ubicacions. Això, en última instància, podria justificar un tractament específic per a cada ubicació del tumor.

En l'estudi de Tenenhaus et al. (2014) apliquen una extensió de l'anàlisi de correlacions canòniques regularitzat generalitzat (RGCCA) adreçat a l'aspecte de la selecció de les variables originals significatives. En els anàlisis que venen a continuació, les variables representades s'han escollit del subgrup seleccionat en aquest article.

5.2.1 Anàlisi de l'expressió gènica

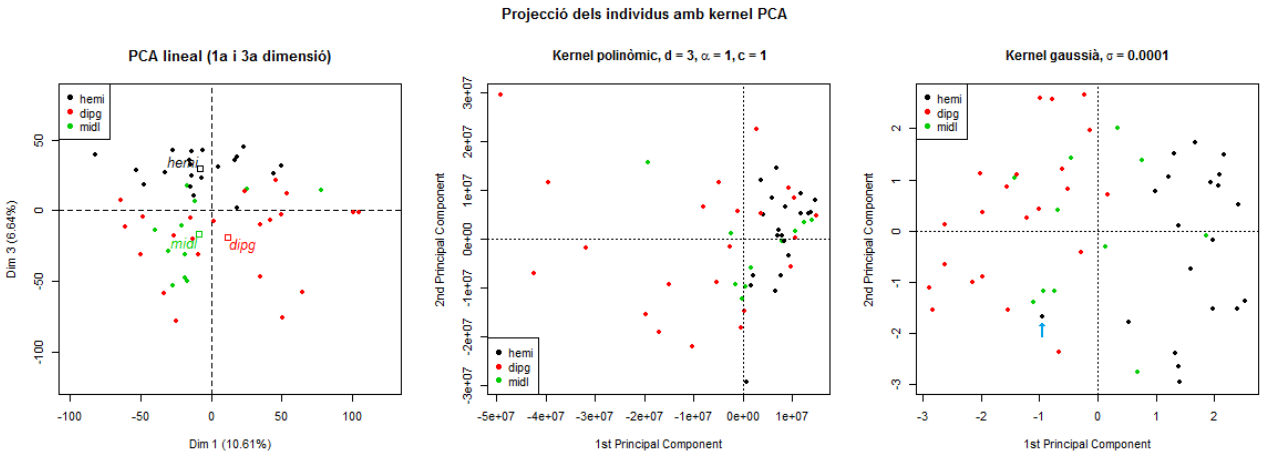
El primer *dataset* a analitzar és el de les expressions dels gens. Aquest està format per les expressions de 15702 gens en els 53 tumors, pel que formalment es té un conjunt X de $n = 53$ individus on $X = (\mathbf{x}_1, ..., \mathbf{x}_{53})$, de forma que cada $\mathbf{x_i}$ és un element del conjunt $\mathcal{X} = \mathbb{R}^{15702}$. Cada tumor, doncs, està representat per un vector de 15702 variables numèriques, que com ja s'ha indicat estan estandarditzades.

L'estudi de l'expressió gènica permet detectar gens que s'expressen diferencialment en condicions diverses. En el nostre cas, les condicions són les diferents ubicacions dels gliomes. Així doncs, es poden trobar gens que estan sobreexpressats o subexpressats en les mostres amb el tumor en una certa localització respecte als gliomes en les altres localitzacions.

1. Aplicació del Kernel PCA

El primer pas ha estat aplicar la tècnica del Kernel PCA utilitzant diversos kernels i diferents paràmetres, per buscar la millor representació dels tumors segons la seva localització. Els tres kernels provats són el kernel lineal, el polinòmic i el gaussià. Després de les diverses proves, s'ha triat un kernel polinòmic de grau 3 (escala = 1, *offset* = 1) i un kernel gaussià amb $\sigma = 0.0001$. La representació dels individus amb Kernel PCA segons el factor localització per als diferents kernels es mostra en la següent figura.

Figura 5.15: *Glioma genes*: Representació dels individus segons la localització



S'observa:

PCA lineal El primer a destacar és que la representació dels tumors es mostra en la primera i la tercera dimensió, ja que s'ha observat que en aquest pla la separació és més clara que no pas en el primer pla factorial. Pel que fa a la primera dimensió, sembla que separi els tumors localitzats al nucli central (MIDL) de la resta, posicionant-los a l'esquerra. I en relació a la segona dimensió separa els tumors amb localització supratentorial (HEMI) de la resta, situant-los en la part superior. Però aquesta separacions no són del tot netes, sinó que hi ha barrejats alguns tumors d'altres localitzacions.

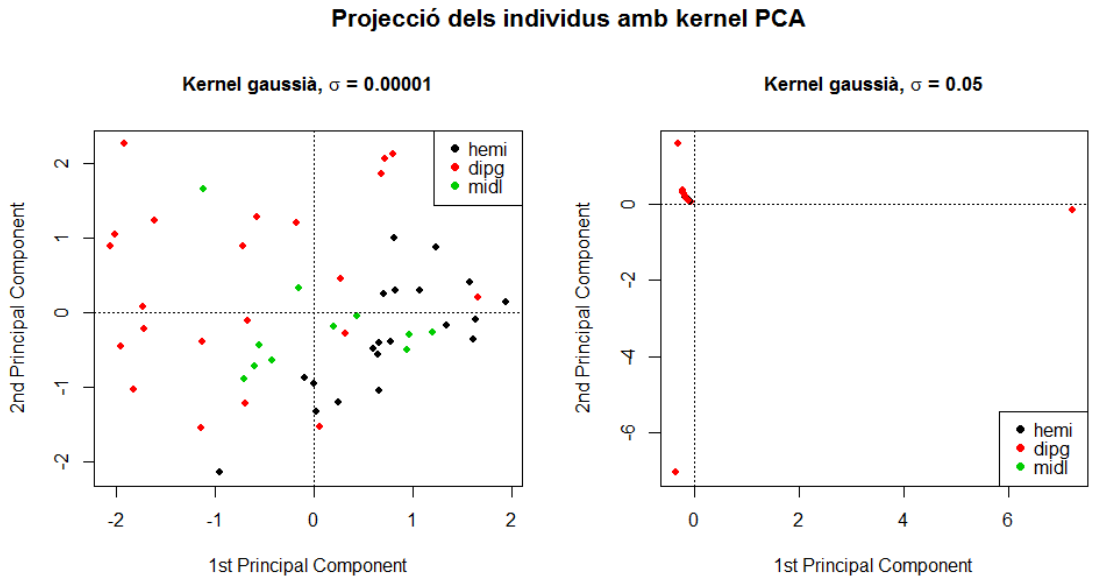
Kernel polinòmic El kernel polinòmic concentra la major part dels tumors en la part positiva del primer eix. Si bé tots els tumors amb localització supratentorial (HEMI) es troben en aquesta part del gràfic, estan barrejats amb tumors de les altres dues localitzacions. Aquest kernel és el que dóna lloc a una pitjor separació de les mostres.

Kernel gaussià La representació dels tumors queda marcada per la primera dimensió, que situa els localitzats al tronc cerebral (DIPG) a l'esquerra, els del nucli central (MIDL) al centre, i els que tenen localització supratentorial (HEMI) a la dreta. S'observa com els tumors MIDL són els que queden més barrejats, confonent-se amb la resta, però cal tenir en compte que és la classe amb menys mostres, pel que també és més difícil de distingir. Per últim, es destaca un tumor HEMI (senyalat al gràfic amb una fletxeta blava), ja que és l'únic d'aquesta classe que no es situa a la dreta de la primera dimensió, pel que pot ser una observació atípica.

Per últim, es destaca com varia la representació dels tumors amb el kernel gaussià en funció del valor del paràmetre. Tot i que la σ escollida ha estat 0.0001, s'ha observat que una σ dins del rang [0.000001, 0.0001] dóna lloc a representacions dels tumors molt

semblants, mentre que les σ majors concentren casi tots els punts al voltant del (0,0) i uns altres pocs en extrems del gràfic. Aquest comportament es mostra en la següent figura, on estan representats els tumors escollint una σ de 0.00001 a l'esquerra i una σ de 0.05 a la dreta.

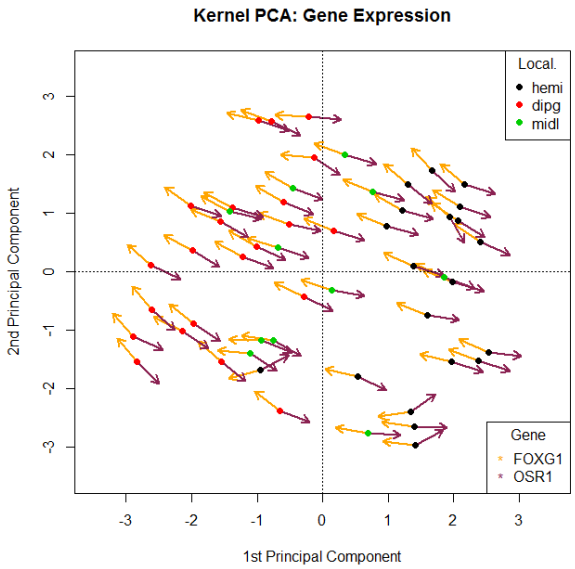
Figura 5.16: Variació en la representació dels tumors segons la σ del kernel gaussià



2. Representació de les variables originals (gens)

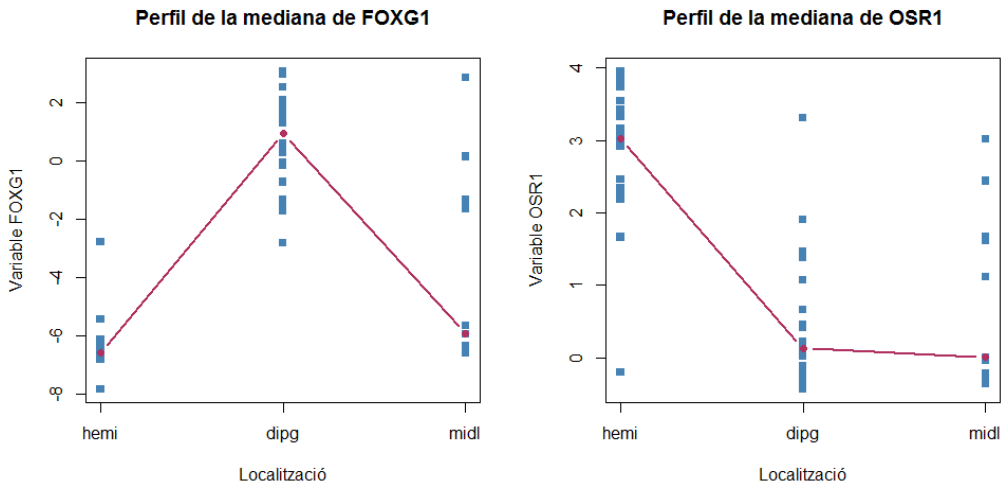
Un cop aplicat el Kernel PCA i triat el valor de la σ del kernel gaussià ($\sigma = 0.0001$), s'ha aplicat el procediment de Reverter et al. (2014) per a representar les variables originals, en aquest cas els gens. En concret es representen dos gens: el *FOXG1* (en color taronja) i el *OSR1* (en color lila).

Figura 5.17: *Glioma dataset*: Representació dels gens *FOXG1* i *OSR1*



Els vectors indiquen la direcció de màxim creixement de l'expressió gènica de cada gen en cada un dels tumors de la mostra. S'observa com per al gen *FOXG1* les fletxes apunten cap al grup de tumors localitzats al tronc cerebral (DIPG), indicant que aquest gen es sobreexpressa en aquesta situació. Per contra, per al gen *OSR1* les fletxes apunten cap al grup de tumors amb localització supratentorial (HEMI), on aquest gen està sobreexpressat respecte a la resta de localitzacions. Per tal de comprovar aquests resultats es mostra el perfil de la mediana per als dos gens segons la localització:

Figura 5.18: *Glioma dataset*: Perfil de la mediana dels gens *FOXG1* i *OSR1*



5.2.2 Anàlisi de les dades d'alteracions del genoma

El segon *dataset* a analitzar és el que conté les dades d'alteracions del genoma. Aquest està format pels desequilibris de 1229 segments de cromosomes en els mateixos 53 tumors, pel que formalment es té un conjunt Y de $n = 53$ individus on $Y = (\mathbf{y}_1, ..., \mathbf{y}_{53})$, de forma que cada $\mathbf{y_i}$ és un element del conjunt $\mathcal{Y} = \mathbb{R}^{1229}$. En aquest cas, doncs, cada tumor està representat per un vector de 1229 variables numèriques, que estan estandarditzades.

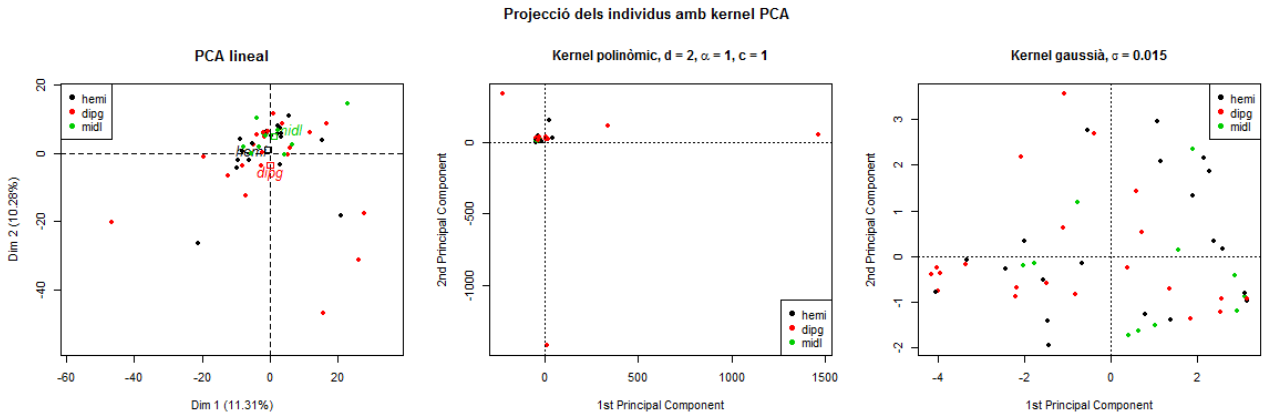
La tècnica de la hibridació genòmica comparativa permet identificar i analitzar alteracions genètiques del tipus guany o pèrdua de material genètic en una mostra d'ADN en comparació amb una mostra de referència. En el context del nostre estudi, aquesta tècnica permet explorar els 46 cromosomes humans en un únic experiment per a estudiar possibles duplicacions o supressions de segments d'ADN en les mostres amb glioma.

Seguint el mateix esquema que per al conjunt genòmic, primer s'aplica el Kernel PCA i després es representen alguns dels desequilibris.

1. Aplicació del Kernel PCA

S'ha utilitzat el kernel lineal (PCA clàssic), el kernel polinòmic (grau = 2, escala = 1, *offset* = 1) i el kernel gaussià ($\sigma = 0.015$), donant com a resultats les següents representacions dels tumors.

Figura 5.19: *Glioma CGH*: Representació dels individus segons la localització



PCA lineal La representació dels tumors es mostra en el primer pla factorial, i s'observa com casi tots els tumors queden agrupats al centre del gràfic mentre que uns pocs queden dispersats pel pla. El PCA no aconsegueix separar els tumors per localització.

Kernel polinòmic El kernel polinòmic accentua l'efecte del kernel lineal, ja que es veu com concentra la major part dels tumors al centre mentre que unes poques mostres

estan disperses i molt allunyades del grup central. Amb el kernel polinòmic tampoc s'aconsegueix una bona representació dels tumors.

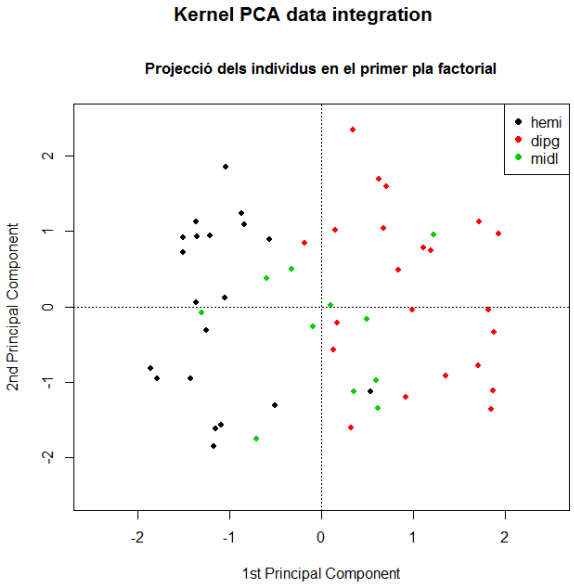
Kernel gaussià Amb el kernel gaussià s'aconsegueix una millor representació dels tumors en el sentit que aquests queden distribuïts per tot el pla, sense les concentracions dels altres gràfics. Però en quant a la separació per localització, els tumors no queden ben dividits: en els quatre quadrants hi ha punts de les tres localitzacions possibles. S'ha provat amb diversos valors per a la σ del kernel gaussià, i s'ha trobat que les σ en el rang de valors [0.01, 0.02] donen lloc a representacions molt semblant a la mostrada, mentre que fora d'aquest rang es dóna el mateix efecte que amb els altres punts.

Així doncs, amb les dades de les alteracions del genoma no s'ha pogut trobar una representació dels tumors separada en funció de la localització del glioma. És per això que la representació de les variables originals d'aquest conjunt es farà un cop s'hagin integrat les dues fonts d'informació.

5.2.3 Integració de les dades transcriptòmiques i dades d'alteracions del genoma

Un cop estudiats per separat els dos conjunts de dades, es procedeix a la seva integració amb el Kernel PCA. Per a l'expressió gènica s'escull el kernel gaussià amb $\sigma = 0.0001$ i per a les alteracions del genoma el kernel gaussià amb $\sigma = 0.001$. S'ha canviat el valor de la σ per al segon conjunt de dades ja que amb aquest nou valor s'aconsegueix una millor representació un cop feta la integració. Els tumors tenint en compte les dues fonts d'informació per a la realització del Kernel PCA queden representats tal i com es mostra en la següent figura:

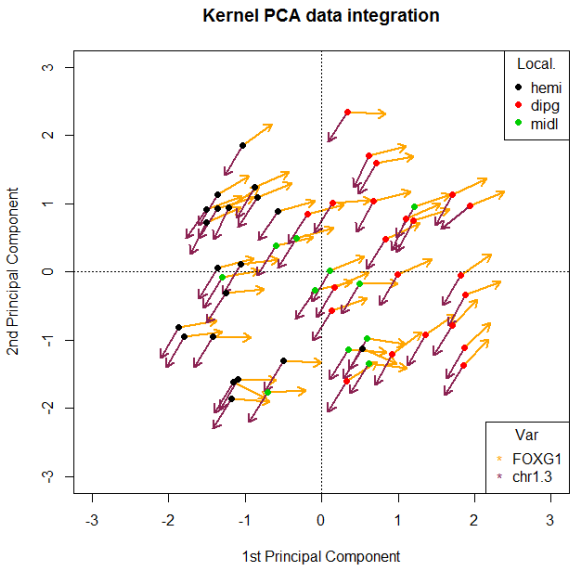
Figura 5.20: *Glioma data integration*: Representació dels individus amb Kernel PCA



S'observa com el resultat és molt semblant a l'obtingut utilitzant només les dades d'expressions dels gens, però rotat 180º. Els tumors localitzats al tronc cerebral (DIPG) queden situats a la dreta, mentre que abans ho estaven a l'esquerra. Els del nucli central (MIDL) es troben al centre, i els que tenen localització supratentorial (HEMI) a l'esquerra, mentre que abans estaven a la dreta. La primera dimensió és la que segueix marcant la separació.

Un cop trobada una representació adequada dels tumors, es passa a representar les variables originals. Donat que s'han integrat les dues fonts d'informació, es poden representar tant gens com segments de DNA desequilibrats. En la següent figura es mostra el gen *FOXG1*, ja representat anteriorment, juntament amb el segment 3 del cromosoma 1.

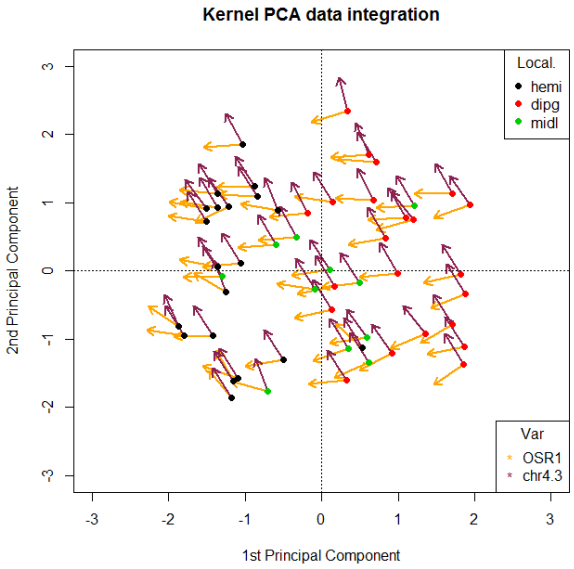
Figura 5.21: *Glioma data integration*: Variables dels dos conjunts (I)



S'observa com per al gen *FOXG1* les fletxes apunten cap al grup de tumors localitzats al tronc cerebral (DIPG), igual que quan es representava utilitzant únicament una de les fonts d'informació. I pel que fa al segment de DNA representat, s'observa com les fletxes apunten en la mateixa direcció i sentit per a tots els tumors. Sembla que apuntin cap als tumors amb localització supratentorial (HEMI), tot i que no és del tot clar.

Per finalitzar, es mostra el gen *OSR1* juntament amb el segment 3 del cromosoma 4.

Figura 5.22: *Glioma data integration*: Variables dels dos conjunts (II)



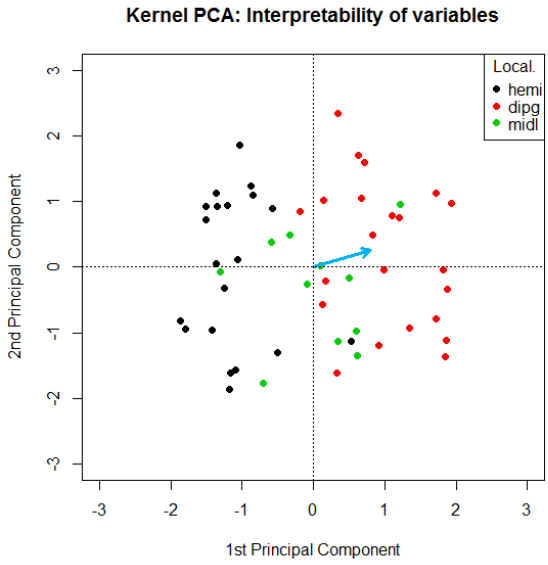
Tal i com era d'esperar, per al gen *OSR1* les fletxes apunten cap al grup de tumors amb localització supratentorial (HEMI), com ja passava abans. Així doncs, les direccions

trobades en els anàlisis individuals es mantenen en la integració. I pel que fa al segment de DNA representat, es torna a veure les fletxes apunten en la mateixa direcció i sentit per a tots els tumors, indicant que la variable es comporta igual per a totes les mostres.

5.2.4 Descobrint la interpretació de les variables

L'últim dels procediments que s'ha aplicat és el que permet fixar una direcció en el pla i buscar les variables originals que tenen una representació en el Kernel PCA correlacionada amb aquesta direcció. La direcció escollida, que es mostra en la figura 5.23 amb el vector blau, s'ha fixat de forma que apunta cap al grup de tumors localitzats al tronc cerebral (DIPG).

Figura 5.23: *Glioma dataset*: Interpretació de les variables



El resultat que proporciona el procediment és la força de la correlació entre les variables originals, tant gens com segments de DNA amb desequilibris, i la direcció fixada. Es mostren les tres variables amb una major correlació positiva i negativa per als dos conjunts de dades.

Taula 5.3: *Glioma dataset*: Correlacions entre les variables i una certa direcció

Gene	Mean	Sd
MGST1	-0.9378	0.1327
FAM89A	-0.9295	0.1619
IRX1	-0.9204	0.1855
⋮	⋮	⋮
DLX1	0.9484	0.0967
DLX2	0.9486	0.0972
FOXG1	0.9566	0.0632

Segment	Mean	Sd
chr9.19	-0.9968	0.0029
chr3.13	-0.9958	0.0094
chr2.55	-0.9950	0.0036
⋮	⋮	⋮
chr7.25	0.9964	0.0091
chr12.30	0.9965	0.0082
chr12.26	0.9966	0.0076

Els gens *MGST1*, *FAM89A* i *IRX1*, com que tenen una correlació negativa molt forta amb la direcció marcada, són gens subexpressats en les mostres de tumor localitzats al tronc cerebral (DIPG). Pel contrari, els gens *DLX1*, *DLX2* i *FOXG1* es sobreexpressen en els tumors d'aquesta localització. Es destaca que el gen *FOXG1* ja s'havia representat en la figura 5.17, on s'observa com la seva representació està en concordança amb la correlació trobada.

En quant als segments d'ADN, s'observa com els segments 19 del cromosoma 9, 13 del cromosoma 3 i 55 del cromosoma 2 estan correlacionats negativament amb la direcció fixada. Això indica que la quantitat de material genètic en aquests segments és menor en les mostres de tumor localitzats al tronc cerebral (DIPG). Per contra, els segments 25 del cromosoma 7, i 26 i 30 del cromosoma 12 tenen una correlació positiva amb la direcció marcada, indicant que en les mostres de tumor DIPG són segments de DNA amb major quantitat de material genètic que en les altres localitzacions.

5.3 Alguns comentaris

Els resultats anteriors mostren com tant la tècnica del Kernel PCA com els procediments de Reverter et al. s'han aplicat satisfactòriament a dades òmiques. En el cas de l'estudi nutrigenòmic, ha estat relativament fàcil fixar el paràmetre del kernel per tal de trobar una bona separació dels individus. Cal remarcar que es tracta d'un conjunt més senzill, amb un nombre de variables no gaire gran, i que a més amb l'aplicació de tècniques lineals ja s'aconsegueixen bons resultats. La dificultat en l'anàlisi s'ha incrementat amb el conjunt de dades de glioma. El nombre de variables ja és molt significatiu, i la tria del paràmetre del kernel gaussià ha estat més complicada. També la representació dels individus intentant separar-los pel factor no és tan clara com en el cas del conjunt *nutrimouse*. En tots dos casos, però, un cop representats els individus, gràcies als procediments de visualització s'han pogut representar les variables d'entrada, tant quan s'ha analitzat un sol bloc de variables com quan s'han integrat. Aquesta visualització de les variables ha permès millorar la interpretació de les variables en relació als grups d'individus, pel que s'ha incrementat el coneixement dels processos biològics.

Capítol 6

IMPLEMENTACIÓ I DIVULGACIÓ DEL KERNEL PCA

En aquest capítol es cobreix l'aspecte de la implementació i divulgació del Kernel PCA amb visualització de les variables originals. En un primer lloc es presenta la creació d'un paquet de R per a que qualsevol usuari d'aquest software pugui utilitzar les funcions necessàries per a l'aplicació de les tècniques vistes. En un segon lloc es presenta la creació d'aplicacions web interactives amb Shiny, per a que qualsevol usuari pugui aplicar les tècniques encara que no tingui o no sàpiga utilitzar R.

6.1 Paquet de R *KPCAvis*

En els capítols anteriors s'ha presentat i aplicat la tècnica del Kernel PCA juntament amb un conjunt de procediments per a millorar la interpretació d'aquest mètode a partir d'una certa visualització de les variables originals. En quant a la implementació de les tècniques, si bé el Kernel PCA es troba al paquet de R *kernlab* i qualsevol usuari de R el pot descarregar, aquest no és el cas per als procediments de visualització de les variables originals. És per això que, com a part d'aquest treball, s'ha creat un nou paquet de R amb les funcions necessàries per implementar i compartir el conjunt de procediments desenvolupats per Reverter et al. (2014) [37].

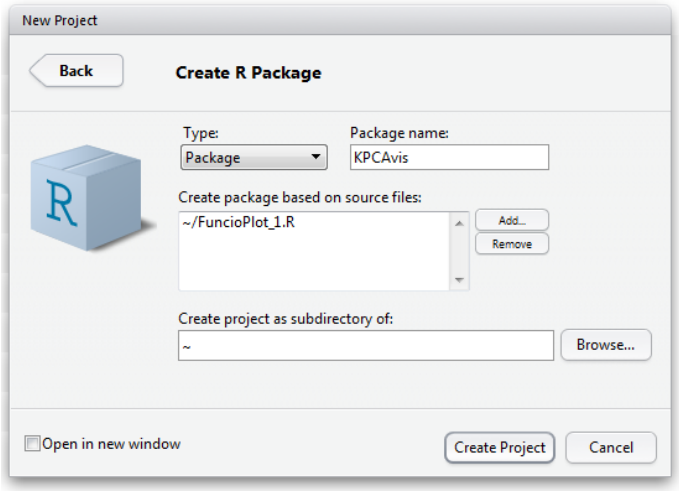
En un primer subapartat s'explica breument com crear paquets de R amb RStudio, i en els següents subapartats es detalla tant l'estructura del paquet construït com les ajudes creades per a facilitar el seu ús.

6.1.1 Creació d'un paquet de R amb RStudio

Els paquets de R són una forma ideal d'emmagatzemar i distribuir codi R i dades per a que altres usuaris el puguin utilitzar. RStudio inclou una gran varietat d'eines que fan que el desenvolupament de paquets de R sigui més fàcil i més productiu. Es poden trobar molts tutorials a Internet sobre com crear paquets amb RStudio. L'objectiu d'aquest apartat és només comentar de forma breu els principals passos a seguir per a la construcció d'un paquet, extrets de [33].

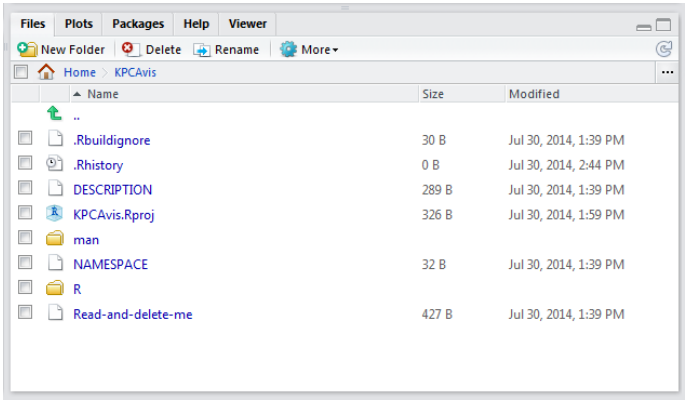
Crear un nou paquet Per crear un nou paquet cal utilitzar la comanda *Create Project* (*File > New project...*) i seleccionar l'opció *New Directory*. En la següent pantalla cal especificar com a tipus de projecte "*Package*" i s'obtindrà la següent finestra:

Figura 6.1: Creació d'un paquet de R



Si es tenen *scripts* de R que es volen utilitzar com a base per al nou paquet es poden especificar aquí i seran inclosos en el nou paquet.
Fent clic a "*Create Project*" RStudio crea una carpeta al directori base amb diversos arxius.

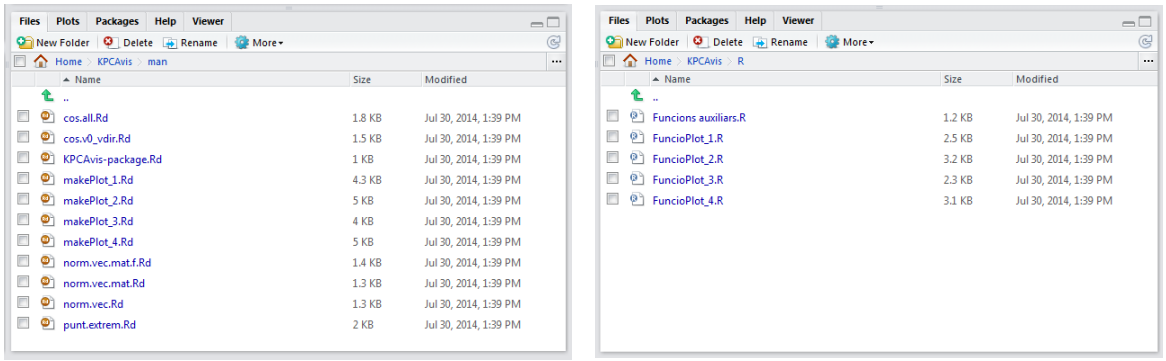
Figura 6.2: Creació d'un paquet de R: Arxius necessaris



D'aquests, cal destacar:

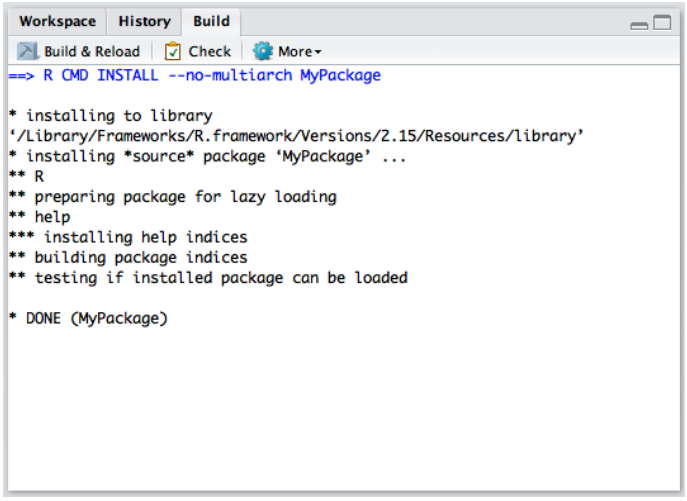
- Arxiu **DESCRIPTION**: Conté la informació bàsica del paquet, incloent el títol, l'autor, una petita descripció del que fa, la data, etc.
- Arxiu **NAMESPACE**: Indica, entre d'altres coses, quines funcions del paquet seran exportades, és a dir, les que els usuaris podran veure i utilitzar.
- Carpeta **man**: Contindrà els arxius del tipus *.Rd* amb les ajudes per a cadascuna de les funcions del paquet. Aquesta ajuda és el que es veurà quan es faci el *help* de la funció en la consola de R. Si en la creació del paquet s'han inclòs *scripts* de R, RStudio crea automàticament arxius *.Rd* per a les funcions en els *scripts*, amb l'esquema que caldrà omplir amb la informació concreta.
- Carpeta **R**: Contindrà els *scripts* de R amb les funcions que implementen les tècniques desitjades. El més usual és que cada *script* es correspongui amb una determinada funció, però també es poden tenir vàries funcions en un mateix *script*. Si en la creació del paquet s'han inclòs *scripts* de R, RStudio els col·loca directament en aquesta carpeta.

Figura 6.3: Creació d'un paquet de R: Arxius *.Rd* i *.R*



Construir el paquet Un cop s'hagin afegit els *scripts* de R i s'hagin completat tant les ajudes de les funcions com l'arxiu *DESCRIPTION*, només queda construir el paquet. Això es realitza usualment amb la comanda *Build and Reload* en el panell *Build*.

Figura 6.4: Construcció d'un paquet de R: comanda *Build and Reload*



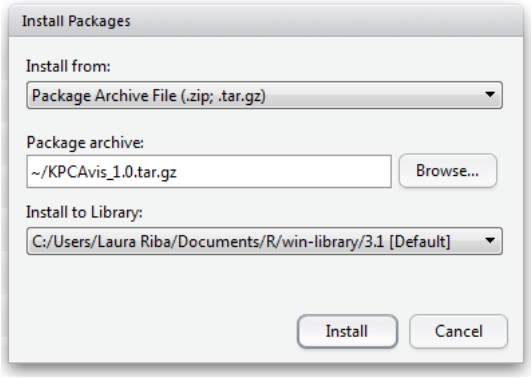
Aquesta comanda realitza diversos passos en seqüència per assegurar un resultat net i correcte:

- Descarrega qualsevol versió existent del paquet.
- Construeix i instal·la el paquet utilitzant *R CMD INSTALL*.
- Reinicia la sessió subjacent de R per assegurar un ambient net per tornar a carregar el paquet.
- Torna a carregar el paquet en la nova sessió de R mitjançant l'execució de la funció `library`.

També és molt útil la comanda *Check*, que comprova que el paquet sigui correcte fent diverses proves, mirant que la documentació estigui per a cada funció exportada i que sigui correcta, carregant el codi R, executant els exemples, etc. D'aquesta manera pots assegurar que un cop construeixis el paquet, aquest es pugui utilitzar correctament. Cal destacar que si vols posar el teu paquet al CRAN, cal que passi tots els tests sense cap error ni cap *warning*.

Per últim, en el panell *Build*, fent clic a *More* > *Build Source Package*, RStudio crea el paquet com un arxiu comprimit. Per poder realitzar aquest pas cal tenir instal·lada l'eina *Rtools*, que es pot descarregar a `http://cran.rstudio.com/bin/windows/Rtools/`. Aquest arxiu comprimit ja es pot compartir amb altres usuaris, que el podran instal·lar a R tal i com es mostra a continuació:

Figura 6.5: Instal·lació del paquet propi en R



Duent a terme aquests passos, s'ha creat el paquet de R *KPCAvis* amb la implementació dels procediments per a millorar la interpretació del Kernel PCA. A continuació s'explica l'estructura del paquet així com les seves ajudes.

6.1.2 Funcions del paquet *KPCAvis*

El paquet *KPCAvis* conté quatre funcions principals per realitzar les representacions gràfiques comentades en els capítols anteriors, on cada funció cobreix un dels quatre aspectes que es desenvolupen en el procediment:

- **Representació de les variables originals**: Funció *makePlot_1*.
- **Integració de diferents fonts d'informació i representació de les variables originals**: Funció *makePlot_2*.
- **Representació de combinacions lineals de les variables originals**: Funció *makePlot_3*.
- **Descobrir la interpretació de les variables originals**: Funció *makePlot_4*.

La resta de funcions són funcions internes que són cridades directament per les quatre funcions principals, pel que no són directament executades per l'usuari. A continuació es detallen aquestes funcions principals:

Funció `makePlot_1` La primera de les funcions permet realitzar el Kernel PCA sobre el *dataset* indicat i representar la projecció dels individus en el primer pla factorial. A cada representació d'un individu s'afegeix la fletxa que equival a la direcció de màxim creixement a nivell local per a la o les variables seleccionades. A més, es poden indicar fins a dos factors per tal de diferenciar grups d'individus, amb colors i símbols dels punts diferents per a cada grup. Per últim, aquesta funció també retorna els gràfics amb els perfils de les medianes de les variables seleccionades per grups d'individus, segons el primer factor introduït. La crida a aquesta funció és:

```
makePlot_1(data, factor_1, factor_2 = NULL, CC = 0.05, v0,
  title1 = "Kernel PCA", xl = c(-3, 3), yl = c(-3, 3))
```

data Les dades a analitzar, que han d'entrar en la funció com una *dataset* on les columnes representen les variables i les files representen els individus o observacions.

factor_1 Factor per a classificar les observacions en grups, segons colors diferents. A més, aquest primer factor és el que s'utilitzarà en els gràfics dels perfils de les medianes de les variables. És un argument obligatori.

factor_2 Factor per a classificar les observacions en grups, segons símbols diferents. Aquest és opcional.

CC Valor del paràmetre σ del kernel gaussià.

v0 Vector que conté els noms de les variables d'interès del *dataset* que es representaran per a cada individu. Ha de ser un vector de tipus caràcter.

Funció makePlot_2 Aquesta funció permet la mateixa visualització que l'anterior però ara integrant dos conjunts de dades diferents. El resultat és el *plot* de la projecció dels individus en el primer pla factorial segons el Kernel PCA, on el kernel utilitzat és la suma directa dels kernels per als dos conjunts de dades. Igual que abans, a cada representació d'un individu s'afegeix la fletxa que apunta a la direcció de màxim creixement a nivell local per a les variables seleccionades, amb l'afegit que aquestes variables poden ser de qualsevol dels dos *datasets*. També es poden indicar fins a dos factors per tal de diferenciar grups d'individus, i en aquest cas no es retornen els gràfics dels perfils de les medianes. La crida a aquesta funció és:

```
makePlot_2(data1, data2, factor_1, factor_2 = NULL, CC = c(0.05, 0.05),
  v01, v02, title1 = "Kernel PCA data integration", xl = c(-4, 4),
  yl = c(-4, 4))
```

data1 Primer *dataset*, amb les variables en columnes i els individus en files.

data2 Segon *dataset*, amb les variables en columnes i els individus en files.

CC Vector de dues posicions, cadascuna referent al valor del paràmetre σ del kernel gaussià per als *datasets* indicats, respectivament.

v01 Vector que conté els noms de les variables que es volen representar del primer conjunt de dades.

v02 Vector que conté els noms de les variables que es volen representar del segon conjunt de dades.

Funció makePlot_3 Aquesta funció permet la mateixa visualització que les dues anteriors, però ara no es representa una variable concreta sinó una combinació lineal de

variables d'interès. Es pot tant representar només un conjunt de dades com realitzar la integració de dos conjunts. En aquest últim cas, les variables d'interès han de pertànyer al primer *dataset* que s'indiqui. També es poden especificar fins a dos factors per tal de diferenciar grups d'individus. La crida a aquesta funció és:

```
makePlot_3(data1, data2 = NULL, factor_1, factor_2 = NULL, CC = 0.05,
  v0, title1 = "Kernel PCA", xl = c(-4, 4), yl = c(-4, 4))
```

`data1` Les dades a analitzar, que han d'entrar en la funció com una *dataset* on les columnes representen les variables i les files representen els individus o observacions.

`data2` Segon *dataset* per integrar amb el primer. En cas de ser `NULL`, només es representarà el conjunt `data1`, sense integració.

`CC` Si només es representa un conjunt de dades (`data2 = NULL`), `CC` és la σ del kernel gaussià. Si es representen dos conjunts, `CC` ha de ser un vector amb els dos valors del paràmetre σ per a cada kernel.

`v0` Vector que conté els noms de les variables d'interès per a les quals es representarà la seva combinació lineal. Aquestes variables han de pertànyer al dataset 1, i s'han d'especificar en un vector de caràcters.

Funció `makePlot_4` Per últim, aquesta funció permet, donada una direcció en el pla, buscar quines variables originals tenen una representació en el Kernel PCA correlacionada amb aquesta direcció fixada. Igual que en el cas anterior, es pot representar només un conjunt de dades o realitzar la integració de dos conjunts. La funció retorna com a resultat un fitxer de text (o dos, si es realitza integració) amb les correlacions (mitjana i desviació típica) entre les variables del conjunt de dades i la direcció determinada. La crida a aquesta funció és:

```
makePlot_4(data1, data2 = NULL, factor_1, factor_2 = NULL, CC = 0.05,
  v01, v02 = NULL, v_dir = c(1, 1), title1 = "Kernel PCA",
  xl = c(-4, 4), yl = c(-4, 4), print = FALSE)
```

`v01` Vector que conté els noms de les variables del primer conjunt de dades per a les quals es calcularà la correlació amb la direcció fixada.

`v02` En cas d'especificar dos conjunts de dades per integrar-los, `v02` és un vector que conté els noms de les variables del segon conjunt de dades per a les quals es calcularà la correlació amb la direcció fixada.

`v_dir` Vector director que marca la direcció amb la qual es buscaran les variables correlacionades en el Kernel PCA.

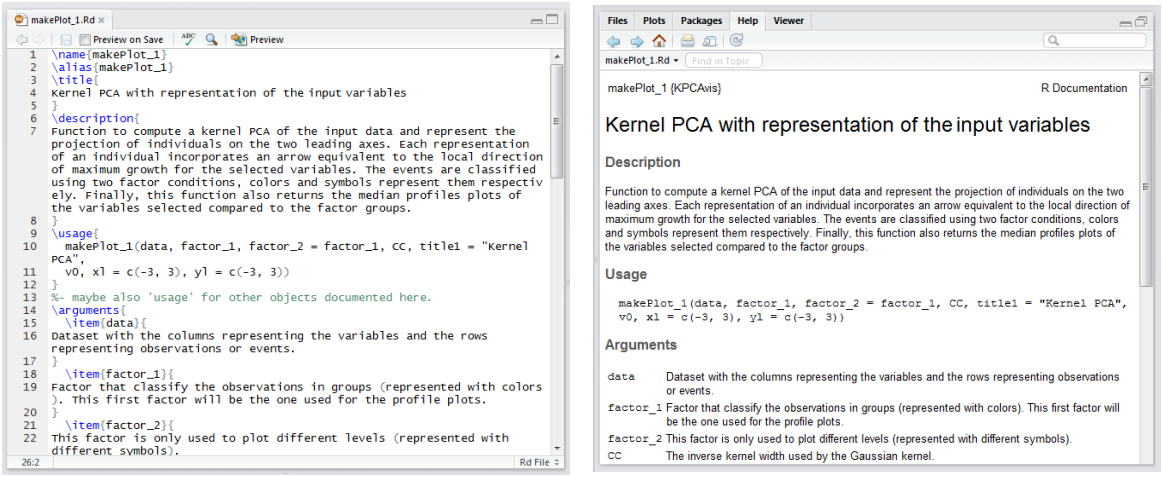
print Valor lògic, per defecte fixat a fals. En cas de ser vertader, els resultats de les correlacions entre les variables indicades i la direcció seleccionada es guarden en fitxers de text i també es mostren per pantalla. En cas de ser fals, els resultats es guarden en els fitxers de text però no es mostren per pantalla.

6.1.3 Ajudes de les funcions del paquet *KPCAvis*

Un dels requisits bàsics dels paquets de R és que totes les funcions, objectes i conjunts de dades han de tenir la documentació completa. RStudio inclou diverses eines que faciliten la creació d'aquests documents, de manera que es poden crear fàcilment clicant a *File > New File > Rd file*, i a més també es poden previsualitzar en format HTML. També permet usar el paquet `Roxygen`, el qual facilita escriure la documentació, ja que la sintaxi dels arxius *.Rd* és semblant al llenguatge Latex, que és una mica complicada al principi.

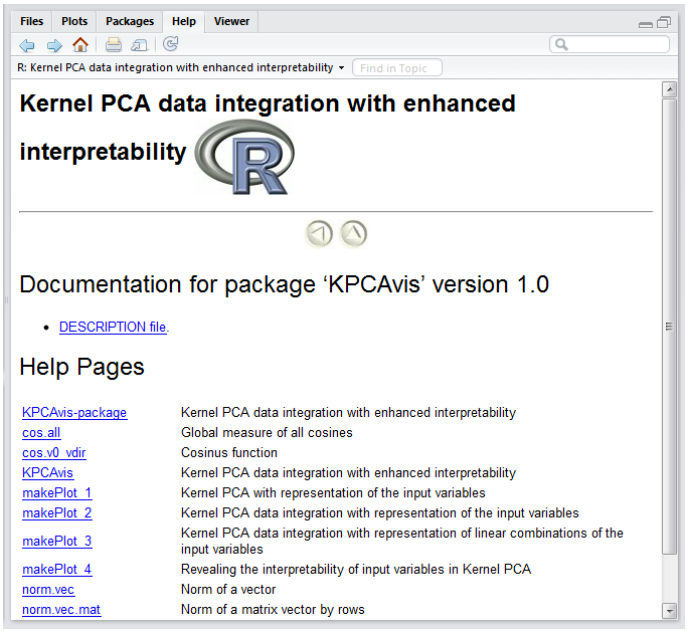
Per exemple, per a la funció `makePlot_1`, l'arxiu *.Rd* i la seva visualització en HTML es mostra en les següents imatges:

Figura 6.6: Ajudes a les funcions: Arxiu *.Rd* i visualització en HTML



Per al paquet *KPCAvis* s'han completat les ajudes de totes les funcions, de manera que quan es faci el *help* del paquet s'obtindrà:

Figura 6.7: Ajudes a les funcions del paquet *KPCAvis*



Així doncs, per a cada funció del paquet es pot accedir a la seva ajuda corresponent, facilitant així l'ús del paquet. Per aquest paquet s'han exportat totes les funcions, cosa que implica que s'ha d'escriure l'ajuda per a totes elles. Una altra opció podria ser exportar només les quatre funcions que ofereixen les representacions. En aquest cas només caldria escriure la documentació per a aquests quatre casos.

6.2 Aplicacions web interactives amb Shiny

El *software* R s'ha consolidat com un dels entorns estadístics més potents, oferint excel·lents característiques per a manipular, analitzar i representar dades. Però a més, també és una plataforma òptima per a la redacció d'informes, gràcies a eines com `Sweave` i `knitr`, que es centren principalment en la generació de documents PDF i HTML. Ara bé, aquests són documents estàtics, de manera que per al lector és impossible modificar qualsevol dels paràmetres utilitzats o repetir els anàlisis en dades pròpies. Com a resposta a aquesta demanda, sorgeix el paquet `Shiny`, que facilita als usuaris de R crear aplicacions web interactives proporcionant una manera ràpida i flexible de compartir els projectes fets en R.

En un primer subapartat s'explica què és Shiny i què permet crear, i en el següent subapartat es detallen les aplicacions web creades per a facilitar l'ús del Kernel PCA amb visualització de les variables originals.

6.2.1 El paquet Shiny

Shiny és un paquet de R creat per RStudio que permet construir aplicacions web interactives directament des de R de forma fàcil, i sense necessitat de saber HTML o JavaScript. Les aplicacions creades es poden executar de forma local o bé poden funcionar des d'un servidor, cosa que facilita compartir-les amb qualsevol persona. En aquest apartat només es destaquen algunes de les principals característiques de Shiny, pel que si es vol conèixer molt més sobre aquest projecte es pot consultar la web principal `http://shiny.rstudio.com/`.

Estructura d'una aplicació Shiny

Qualsevol aplicació Shiny ha de tenir com a mínim dos components: el fitxer `ui.R` i el fitxer `server.R`. Aquests dos fitxers han d'estar continguts en un mateix directori juntament amb altres documents que puguin ser necessaris per a que l'aplicació funcioni, com per exemple conjunts de dades o imatges.

- **`ui.R`**: Aquest fitxer conté les instruccions que defineixen la interfície d'usuari de l'aplicació, i per tant controla el seu disseny i el seu aspecte.
- **`server.R`**: Aquest fitxer conté les instruccions que l'ordinador necessita per a construir l'aplicació, és a dir, les instruccions que realitzaran els càlculs i els gràfics necessaris.

A continuació es mostren dos fitxers d'exemple que permeten crear una aplicació que traça un histograma d'un conjunt de dades de R amb un nombre configurable d'intervals.

Figura 6.8: Exemples dels fitxers `ui.R` i `server.R`

ui.R

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:", min = 1,
                  max = 50, value = 30) ),
    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

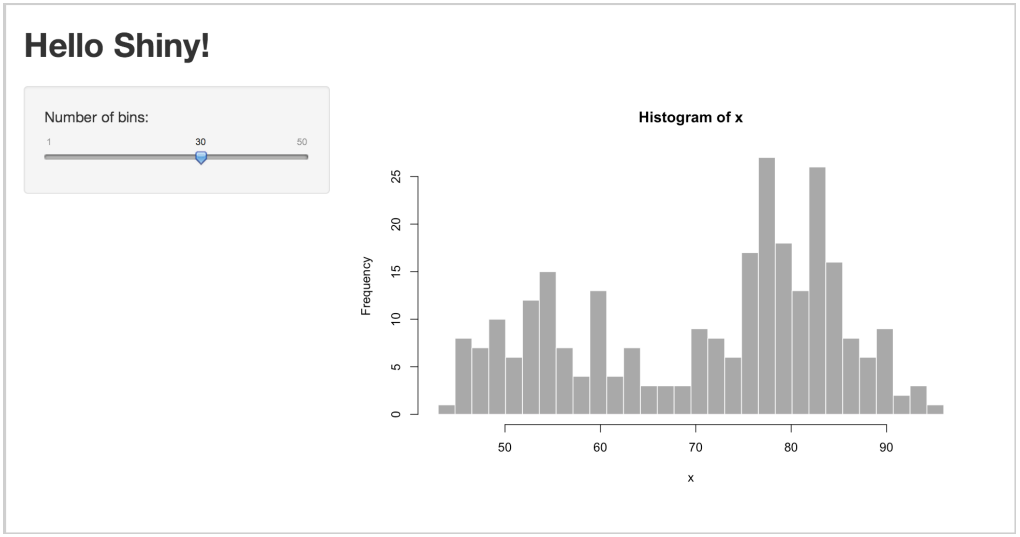
server.R

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute
  #     automatically when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Execució d'una aplicació Shiny

Es pot crear una aplicació Shiny creant un nou directori i guardant els fitxers `ui.R` i `server.R` a dins. Cada aplicació necessita el seu propi i únic directori. Si la carpeta on s'han guardat els fitxers s'anomena "exemple", per executar la App des de R (cal tenir carregat el paquet `shiny`) només cal executar la funció `runApp("exemple")`. Un cop executat, automàticament s'inicia el navegador i es mostra l'aplicació. Per exemple, per als fitxers de la figura 6.8 s'obtindria una pàgina web amb un aspecte similar a la següent imatge:

Figura 6.9: Exemple d'aplicació Shiny



En aquesta aplicació, tal i com s'observa, l'usuari pot modificar el nombre d'intervals de l'histograma i el gràfic anirà canviant automàticament segons el valor definit.

Disseny de les aplicacions

El paquet Shiny té incorporades funcions que permeten:

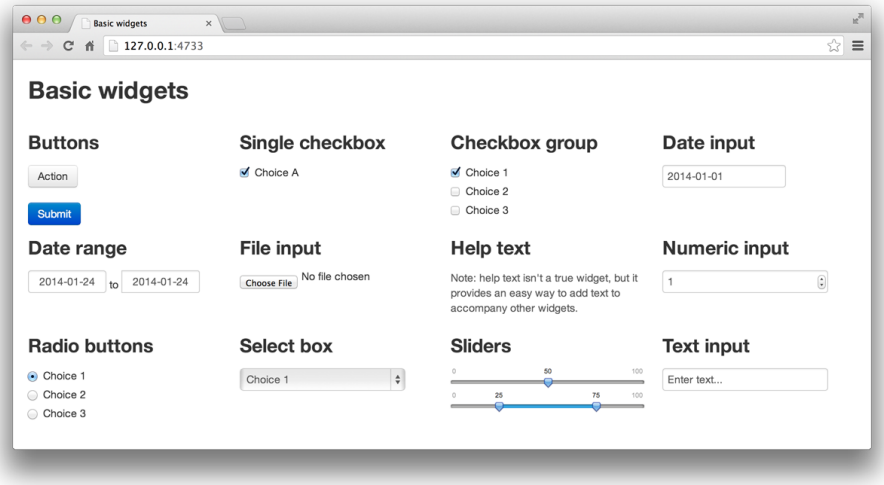
- Crear *widgets* d'entrada com barres lliscants, entrades numèriques i llistes desplegables.
- Incloure sortides en forma de gràfic, de taula o de text, que reaccionen i s'actualitzen cada cop que l'usuari canvia el valor dels *widgets* d'entrada.
- Distribuir els *widgets* d'entrada i les sortides en graelles, en pestanyes o en llistes de navegació.

1. *Widgets* d'entrada

Els *widgets* són elements web amb els quals poden interactuar els usuaris, sent el mecanisme amb el que els usuaris es comuniquen amb l'aplicació. Shiny té incorporades

funcions per a crear *widgets* de diversos tipus, que es mostren en la següent imatge:

Figura 6.10: *Widgets* d'entrada de Shiny



A la Galeria de *Widgets* es troben plantilles que es poden utilitzar per afegir ràpidament *widgets* a les aplicacions Shiny.

2. Sortides

Una de les principals característiques de Shiny és la programació reactiva, que permet que les sortides canviïn automàticament cada cop que l'usuari modifica el valor del *widget* d'entrada. Shiny té incorporades funcions específiques que s'han de col·locar tant a l'arxiu `ui.R` com al `server.R` per a crear sortides reactives, de les que es destaquen:

Taula 6.1: Funcions per a crear sortides reactives

`server.R`	`ui.R`	Crea...
renderImage	imageOutput	Imatge
renderPlot	plotOutput	Gràfic
renderTable	tableOutput	Taula
renderText	textOutput	Text
renderText	htmlOutput	HTML
renderText	verbatimTextOutput	Text

3. Distribució dels elements

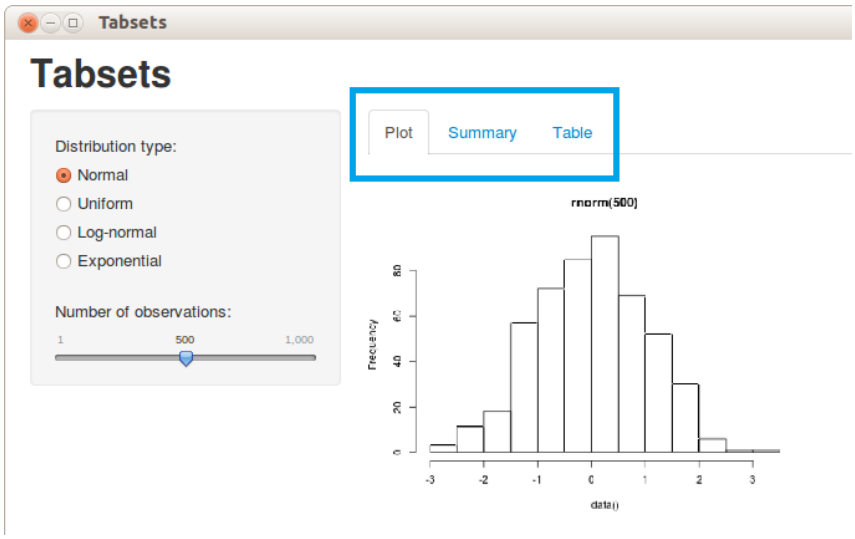
El paquet Shiny permet posicionar els components de les aplicacions de diferents formes:

- El disseny per defecte és el més simple, i inclou una barra lateral per als *inputs* i una àmplia zona principal per a les sortides. Aquest és el disseny de la imatge 6.9.

- Si es vol personalitzar el disseny, Shiny disposa del disseny en graella, en el qual l'usuari decideix el nombre de files i columnes de la graella i què vol posicionar a cada casella.

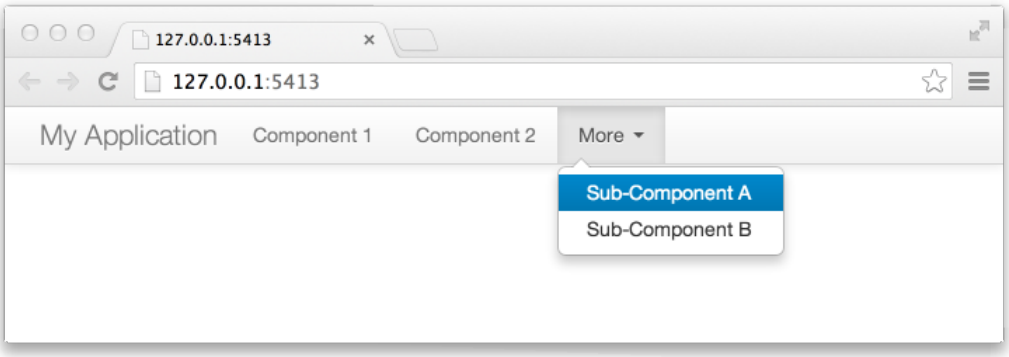
- Es pot segmentar el disseny posicionant les sortides en diferents pestanyes o en forma de llista, tal i com mostra la següent imatge.

Figura 6.11: Navegació amb pestanyes



- Es poden crear aplicacions més complexes amb múltiples sub-components, cadascun amb el seu panell lateral o amb diverses pestanyes.

Figura 6.12: Aplicacions amb barres de navegació



Compartir les aplicacions

Quan es tracta de compartir aplicacions Shiny, es tenen dues opcions bàsiques:

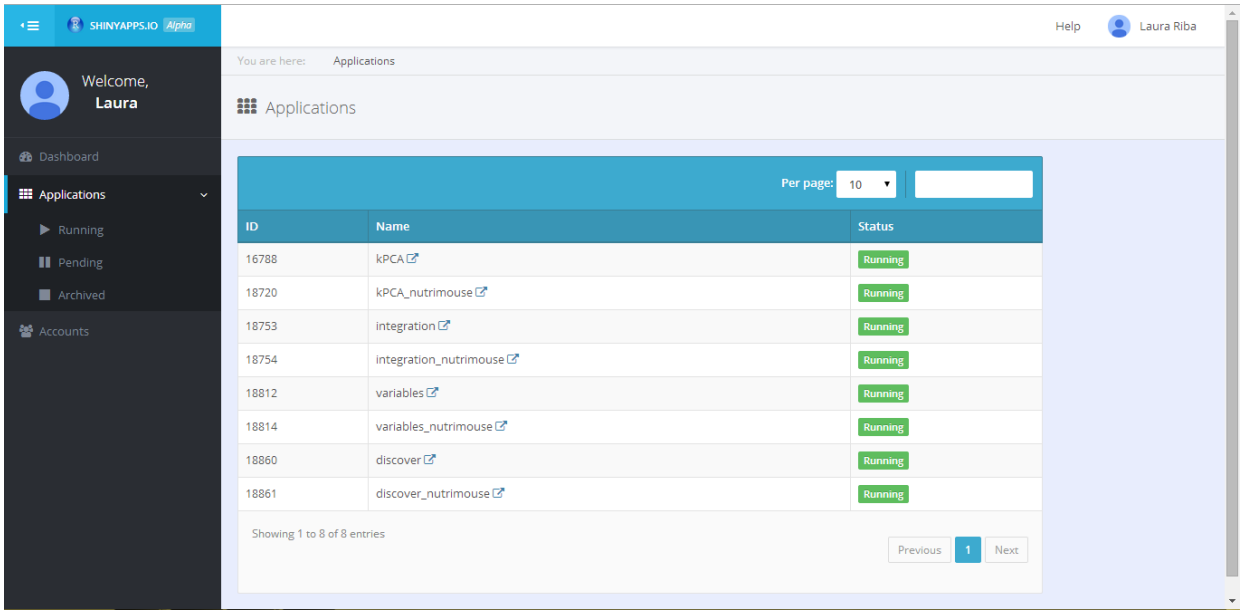
- **Compartir la App com dos arxius: `server.R` i `ui.R`** Aquesta és la forma més senzilla de compartir una aplicació, però només funciona si els usuaris tenen R en el seu ordinador (i saben com usar-lo). Els usuaris poden utilitzar aquests fitxers per

llançar l'aplicació des de la seva pròpia sessió de R utilitzant la funció `runApp()`, tal i com ja s'ha explicat.

- **Compartir la App com una pàgina web** Aquesta és la manera més potent de compartir una aplicació, ja que no cal que els usuaris tinguin R instal·lat, ni que sàpiguen què és. Els usuaris poden navegar per la teva aplicació a través d'internet amb un navegador web, ja sigui amb un ordinador, amb una tableta o amb un telèfon mòbil.

RStudio ofereix diverses opcions per a allotjar les teves aplicacions Shiny en una pàgina web, sent la més senzilla (i la utilitzada en aquest treball) a través de *ShinyApps.io*. *ShinyApps.io* et permet pujar la teva aplicació directament des de la sessió de R a un servidor allotjat per RStudio, i et proporciona control complet sobre la teva aplicació a través d'eines d'administració de servidor. En la imatge 6.13 es mostra una captura de pantalla d'aquesta web. Per saber més es pot visitar `https://www.shinyapps.io/`.

Figura 6.13: Captura de pantalla de *ShinyApps.io*



6.2.2 Aplicacions creades

Per tal de facilitar l'ús de la tècnica del Kernel PCA juntament amb la visualització de les variables originals, s'han creat diverses aplicacions web, que s'expliquen a continuació. Totes elles estan disponibles en pàgines web en dues versions: per un costat es troben aplicades al conjunt de dades *Nutrimouse* (veure secció 5.1), i per una altra banda estan dissenyades de forma que els usuaris poden pujar els seus arxius de dades propis per tal d'aplicar els diversos procediments.

En cas de voler utilitzar dades pròpies, cal que els arxius segueixin una sèrie de característiques:

Conjunts de dades Cada conjunt de dades ha d'estar en un arxiu de text tipus `.txt`, que ha de tenir les observacions per files i les variables per columnes. La primera fila de l'arxiu cal que contingui el nom de les variables. Les dades han d'estar separades per espais en blanc (un o més espais, tabuladors, ...), i el separador decimal ha de ser un punt.

Factors Si es tenen condicions (factors) que identifiquin les observacions, aquests han d'estar continguts en un arxiu del mateix tipus que les dades. Cada columna fa referència a una condició, i el nom de les columnes és obligatori. L'arxiu pot tenir tants factors com es vulgui, però a l'hora de representar les observacions es podran triar dos com a màxim.

A continuació es detallen les característiques de les quatre aplicacions web creades.

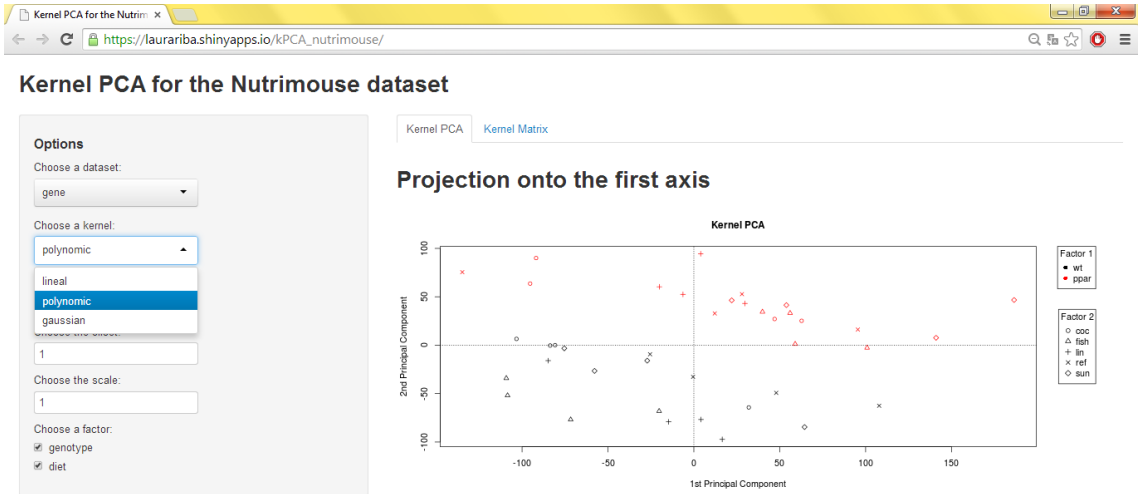
Shiny App 1: Kernel PCA

La primera de les *apps* creades permet aplicar la tècnica del Kernel PCA a un sol conjunt de dades numèriques, i representar les observacions en el primer pla factorial. A més, en una segona pestanya, es mostra el `summary` de la matriu kernel així com l'histograma dels seus valors i una representació de la matriu. Té les següents característiques:

- Es pot triar el tipus de kernel dessitjat i els seus paràmetres. Les possibilitats són el kernel lineal, el polinòmic i el gaussià.
- Les observacions es poden identificar amb un màxim de dues condicions (dos factors). Si no hi ha cap factor, tots els punts es representen igual, si es tria un factor es diferencia amb el color dels punts i si es trien dos factors, s'utilitza el color i el símbol dels punts per a diferenciar les observacions.

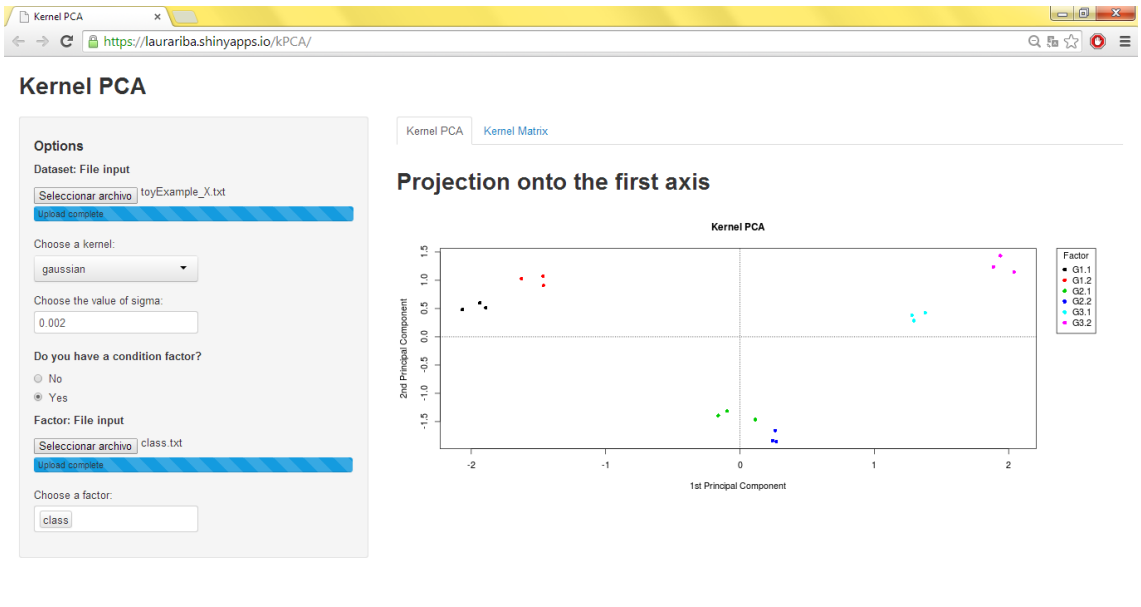
En concret, per al conjunt de dades *Nutrimouse*, es pot triar si es vol aplicar la tècnica a les dades transcriptòmiques o a les metabolòmiques; i en quant a les condicions, es pot escollir si representar els ratolins segons el tipus de genotip i / o el tipus de dieta. Executant aquesta aplicació s'obté una pàgina web amb un aspecte similar a la següent imatge:

Figura 6.14: Shiny App 1: Kernel PCA per al conjunt *Nutrimouse*



En la següent imatge es mostra l'aspecte que tindria la pàgina web si s'executa l'aplicació que permet pujar els arxius de dades propis. En concret, les dades utilitzades són les corresponents a l'exemple de joguina utilitzat per a il·lustrar la tècnica del Kernel PCA al capítol 4 (veure 4.3).

Figura 6.15: Shiny App 1: Kernel PCA per a un conjunt de dades de l'usuari



Aquesta primera aplicació es troba disponible en les següents adreces web:

- Per al conjunt *Nutrimouse*: `http://laurariba.shinyapps.io/kPCA_nutrimouse`
- Per a dades de l'usuari: `http://laurariba.shinyapps.io/kPCA`

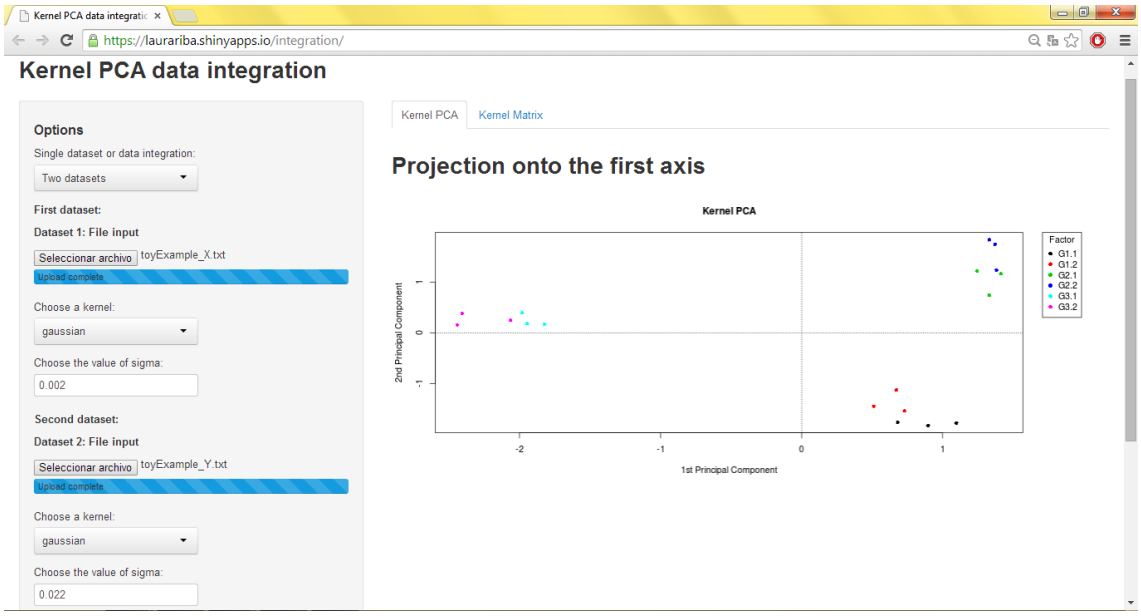
Shiny App 2: Kernel PCA data integration

La segona de les *apps* creades és una extensió de la primera, que incorpora la característica de la integració de diverses fonts de dades. Així doncs, aquesta permet aplicar la tècnica del Kernel PCA bé a un sol conjunt de dades numèriques o bé integrant dos conjunts de dades. El resultat, igual que abans, és la representació de les observacions en el primer pla factorial. També inclou, en una segona pestanya, el `summary` de la matriu kernel així com l'histograma dels seus valors i una representació de la matriu. Té les següents característiques:

- Es pot escollir si aplicar la tècnica a un conjunt de dades per separat o bé integrant dos conjunts.
- Es pot triar el tipus de kernel dessitjat i els seus paràmetres. En cas d'integrar dos conjunts de dades, cal triar el kernel i els paràmetres per a cadascun dels dos. Els kernels possibles tornen a ser el lineal, el polinòmic i el gaussià.
- Igual que en la primera *app*, les observacions es poden identificar amb un màxim de dues condicions (dos factors), mitjançant colors i símbols diferents.

En la següent imatge es mostra l'aspecte que tindria la pàgina web si s'executa l'aplicació que permet pujar els arxius de dades propis. Igual que abans, les dades utilitzades són les corresponents als dos exemples de joguina utilitzats per a il·lustrar la integració de dades amb Kernel PCA al capítol 4 (veure 4.3).

Figura 6.16: Shiny App 2: Kernel PCA amb integració per a un conjunt de dades de l'usuari



Aquesta segona aplicació es troba disponible en les següents adreces web:

- Versió per al conjunt *Nutrimouse*: `http://laurariba.shinyapps.io/integration_nutrimouse`

- Versió per a dades de l'usuari: `http://laurariba.shinyapps.io/integration`

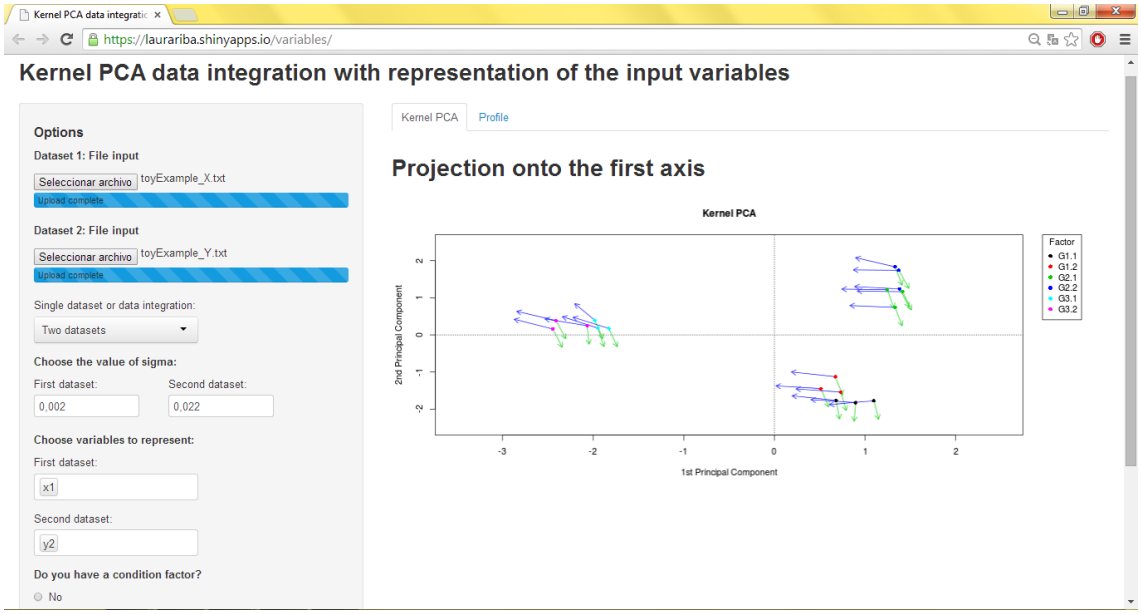
Shiny App 3: Kernel PCA data integration with representation of the input variables

La tercera de les *apps* creades parteix de la segona aplicació i incorpora els procediments de Reverter et al. (2014) per a representar les variables d'entrada. Així doncs, aquesta aplicació permet realitzar la tècnica del Kernel PCA bé a un sol conjunt de dades numèriques o bé integrant dos conjunts de dades i representar les variables de qualssevol dels conjunts. El resultat, igual que abans, és la representació de les observacions en el primer pla factorial. Té les següents característiques:

- Es pot escollir si aplicar la tècnica a un conjunt de dades per separat o bé integrant dos conjunts.
- En aquest cas no cal triar el tipus de kernel, ja que els procediments per a visualitzar les variables només estan definits per al kernel gaussià. El que sí cal triar és el valor de la σ del kernel. En cas d'integrar dos conjunts de dades, cal triar els paràmetres per a cadascun dels dos kernels.
- Igual que en les anteriors *apps*, les observacions es poden identificar amb un màxim de dues condicions (dos factors), mitjançant colors i símbols diferents.
- En una segona pestanya es troba la representació dels perfils de les medianes de les variables triades agrupades segons els grups d'individus definits pel factor, en cas que n'hi hagi. Aquesta característica només està disponible quan es representa un sol conjunt de dades.

En la següent imatge es mostra l'aspecte que tindria la pàgina web si s'executa l'aplicació que permet pujar els arxius de dades propis. Igual que abans, les dades utilitzades són les corresponents als dos exemples de joguina utilitzats per a il·lustrar al capítol 4 el Kernel PCA.

Figura 6.17: Shiny App 3: Kernel PCA amb integració i representació de les variables



Aquesta tercera aplicació es troba disponible en les següents adreces web:

- Versió per al conjunt *Nutrimouse*: `http://laurariba.shinyapps.io/variables_nutrimouse`
- Versió per a dades de l'usuari: `http://laurariba.shinyapps.io/variables`

Shiny App 4: Kernel PCA with enhanced interpretability

L'última de les *apps* creades permet aplicar el Kernel PCA a un conjunt de dades de l'usuari i incorpora el procediment de Reverter et al. (2014) per a trobar variables d'entrada correlacionades amb una certa de direcció fixada. Aquesta aplicació té les següents característiques:

- Cal escollir la la σ del kernel gaussià del Kernel PCA.
- Es pot triar la direcció amb la qual es buscaran variables amb representació correlacionada. Cal fixar les coordenades x i y del vector director.
- Igual que en les anteriors *apps*, les observacions es poden identificar amb un màxim de dues condicions (dos factors), mitjançant colors i símbols diferents.
- Els resultats de la *app* es divideixen en tres pestanyes:
  - En la primera pestanya es troba la representació de les observacions en el primer pla factorial juntament amb la fletxa que assenyala la direcció triada.
  - En una segona pestanya es mostra una taula amb la correlació (mitjana i desviació tipus) entre les variables i la direcció. Es pot triar si mostrar la correlació per a totes les variables, per a les més correlacionades tant positivament com

negativament (l'usuari escull quantes mostrar), per a les variables amb correlació superior a un cert llindar que també pot fixar l'usuari, o bé es poden triar una a una les variables d'interès.

– En l'última pestanya es troba un gràfic on es representa la correlació per a totes les variables, ordenades de menor a major. Es representa tant la mitjana com la desviació tipus.

En les següents imatges es mostra l'aspecte que tindria la pàgina web si s'executa l'aplicació que permet pujar els arxius de dades propis, utilitzant l'exemple de joguina del capítol 4. La primera imatge mostra la primera pestanya de l'aplicació i la segona imatge la tercera pestanya.

Figura 6.18: Shiny App 4: Kernel PCA amb millora en la interpretació de les variables (I)

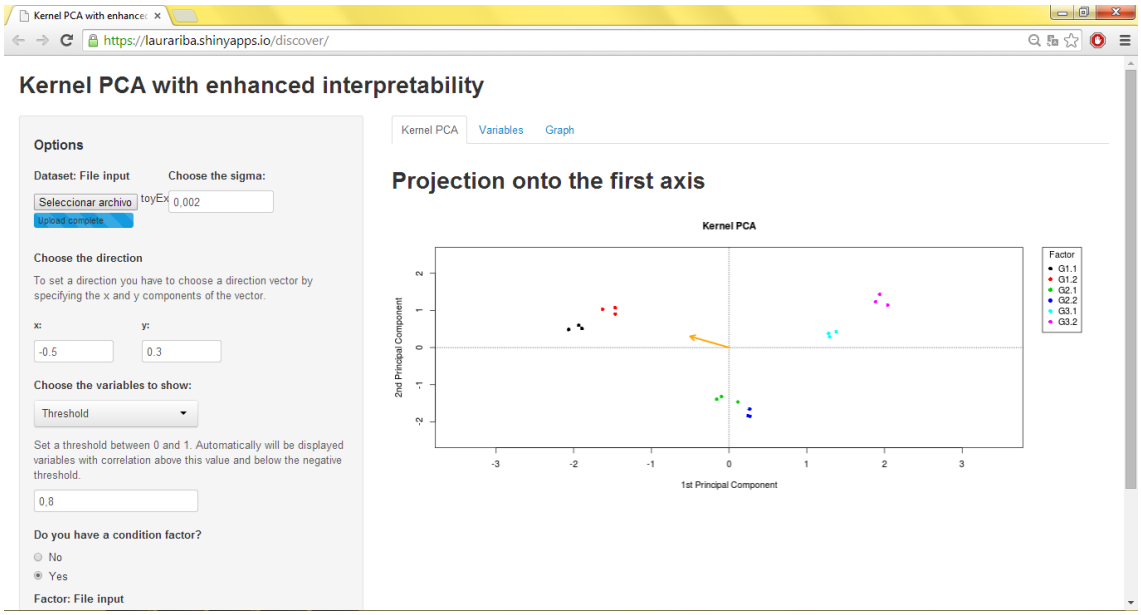
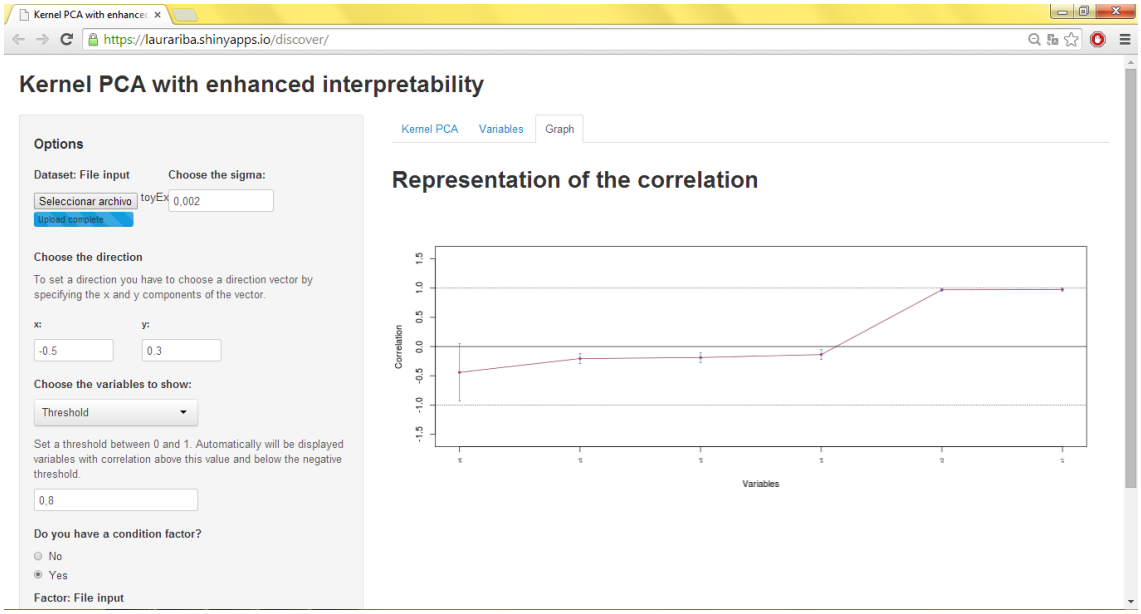


Figura 6.19: Shiny App 4: Kernel PCA amb millora en la interpretació de les variables (II)



Aquesta quarta aplicació es troba disponible en les següents adreces web:

- Versió per al conjunt *Nutrimouse*: `http://laurariba.shinyapps.io/discover_nutrimouse`
- Versió per a dades de l'usuari: `http://laurariba.shinyapps.io/discover`

Capítol 7

CONCLUSIONS

En aquest capítol es resumeixen les conclusions més importants sorgides amb el desenvolupament d'aquest projecte, es comenten diverses línies de continuació del treball realitzat i es presenta la valoració personal del projecte.

7.1 Resultats obtinguts

El present treball s'ha realitzat amb l'objectiu general d'obtenir una visió global de l'aplicació de l'estadística al camp de la bioinformàtica, en concret a l'anàlisi de dades òmiques. Amb aquest objectiu present, s'han desenvolupat els capítols dos i tres d'aquesta memòria. En el segon capítol s'han presentat els principals tipus de dades òmiques, quines són les tecnologies d'alt rendiment associades i quins són els reptes que presenta el seu anàlisi, inclòs el de la integració de dades òmiques. En el tercer capítol s'ha donat una visió general de la disciplina del *Machine Learning*, que és la que dóna lloc als algorismes i tècniques necessàries per a analitzar les dades òmiques. A més, s'ha parat especial atenció als mètodes kernel, ja que el seu ús permet integrar diverses fonts de dades de forma natural. D'aquests capítols s'han pogut extreure diverses conclusions:

- Les dades òmiques són un tipus de dades particular caracteritzat per constar de conjunts amb milers de variables però habitualment amb relativament poques observacions, sent dades heterogènies i amb bastant de soroll. Això dificulta el seu anàlisi amb les tècniques estadístiques clàssiques i requereix mètodes més sofisticats que sorgeixen de camps com l'aprenentatge automàtic.

- L'aprenentatge automàtic es pot pensar com una estadística "no paramètrica" on s'apliquen algorismes que aprenen a partir de dades d'exemple, cosa que fa que sigui adaptable a situacions molt diferents i a més amb molt bones propietats. Amb l'estudi del camp del *Machine Learning* s'ha descobert un món molt ampli i que actualment està a l'ordre del dia. Tot i que en aquest projecte s'ha vist aplicat a un

tipus de dades molt concret, els algorismes de *Machine Learning* s'utilitzen en un munt d'aplicacions que usem cada dia.

- Moltes de les tècniques clàssiques d'Estadística encaixen en les tasques d'aprenentatge automàtic. Per exemple, la regressió lineal o la regressió logística són exemples d'aprenentatge supervisat mentre que l'anàlisi de components principals és un cas d'aprenentatge no supervisat. S'ha destacat com entre aquestes disciplines es comparteixen tècniques i conceptes, però que utilitzen llenguatges diferents.

- Els mètodes kernels permeten extreure relacions no lineals de forma òptima en dades de gran dimensionalitat, i que a més poden ser de diferents tipus (numèriques, caràcters, text, grafs, ...). Això els converteix en una metodologia molt potent tant per a reduir la dimensió com per a integrar múltiples conjunts de dades. També s'ha destacat com no hi ha cap regla ni per a la tria del kernel adequat ni a l'hora d'escollir els seus paràmetres, sinó que es requereix anar provant, sent un procés bastant pesat.

Un cop posat en context tant el tipus de dades a treballar com la disciplina que proporciona les tècniques a utilitzar, l'objectiu que s'ha perseguit és conèixer i aplicar mètodes que permetin tant la integració de diferents fonts de dades òmiques com la visualització de les variables originals, dos dels principals reptes actuals en Bioinformàtica. Com a resposta a aquest objectiu s'han desenvolupat els capítols quatre i cinc d'aquesta memòria. En el quart capítol s'han comentat algunes tècniques estadístiques multivariants per a la integració de dades, i s'ha explicat en profunditat el Kernel PCA, que ha estat la tècnica d'integració triada. Juntament amb el Kernel PCA s'han detallat un conjunt de procediments per a poder visualitzar les variables originals. Aquesta tècnica s'ha aplicat satisfactòriament a dos conjunts de dades òmiques, resultats que es troben en el cinquè capítol. Amb aquests capítols s'ha pogut arribar a les següents conclusions:

- El Kernel PCA s'ha utilitzat com un mètode d'integració des del punt de vista de reducció de la dimensió. S'ha pogut comprovar com els mètodes basats en kernels proporcionen mitjançant la combinació de kernels una manera natural i potent d'integrar la informació.

- Si bé amb el Kernel PCA s'obtenen generalment bones representacions dels individus, les variables originals queden amagades dins del kernel, fent difícil veure quines són les variables rellevants. El problema de la visualització de les variables originals és comú en els mètodes basats en kernels, i és de moment un tema no resolt en el que es segueix investigant. El conjunt de procediments que proposa Reverter et al. (2014) és una primera aproximació de caire exploratori a aquesta qüestió. Aquests procediments encara estant sota investigació per tal de poder-los estendre a kernels

a més del gaussià i per trobar mesures que millorin la informació que proporciona la seva forma de representar les variables.

- La tècnica del Kernel PCA s'ha aplicat satisfactòriament a dos conjunts de dades diferents. S'ha observat com el conjunt *Nutrimouse* és més simple i amb menys variables, pel que ha resultat més fàcil obtenir una bona representació dels individus. Per contra, les dades de glioma són un conjunt amb moltes més variables, fet que ha dificultat fixar els paràmetres del kernel i s'ha arribat a una separació no tan clara de les mostres.
- Tant la integració dels diferents tipus de dades òmiques com els diferents procediments aplicats per a visualitzar les variables originals han portat a una millor comprensió dels processos biològics, demostrant la utilitat i la potència dels mètodes vistos.

Finalment, l'últim objectiu específic marcat ha estat proporcionar la tècnica del Kernel PCA amb visualització de les variables originals d'una forma accessible tant per als usuaris de R com per als no usuaris d'aquest programari. Aquest objectiu s'ha assolit mitjançant el desenvolupament d'un paquet de R i la creació d'aplicacions web interactives amb el paquet Shiny, processos detallats al capítol 6. Per una banda, d'aquesta etapa es destaca com es poden crear paquets de R fàcilment gràcies a RStudio i com aquests paquets són una forma perfecta per a compartir codi R i conjunts de dades amb altres usuaris. D'altra banda es ressalta l'eina tant potent que és Shiny per a divulgar coneixements, com amb molt poc temps es poden crear grans aplicacions, i com aquesta eina s'ha utilitzat de forma satisfactòria per a facilitar l'ús del Kernel PCA amb visualització de les variables originals.

7.2 Possibles extensions del treball realitzat

Un cop acabat el projecte i revisant fins a on s'ha arribat, sorgeixen algunes extensions que es podrien realitzar a partir d'aquest treball:

Comparar els resultats amb altres tècniques Tot i que des del començament es va determinar que la tècnica principal del treball seria el Kernel PCA, i que el que s'intentaria era abordar el problema de la representació de les variables, una altra tasca plantejada a l'inici del projecte va ser comparar els resultats obtinguts amb altres tècniques estadístiques multivariants. De fet, es va arribar a estudiar les tècniques de l'anàlisi factorial múltiple (MFA) i de l'anàlisi de correlacions canòniques regularitzat (rCCA) i es van aplicar al conjunt de dades *Nutrimouse*. Es va decidir, però, no continuar amb aquesta investigació i no incloure aquestes anàlisis en el treball per evitar sobrepassar l'extensió marcada per al TFG.

Paràmetres del kernel Com ja s'ha comentat, la tria dels paràmetres per als kernels manca d'un procés específic que faciliti aquesta elecció. Una possible via de continuació del treball és investigar si s'està treballant en aquest aspecte i quins són els principals resultats trobats actualment.

Kernel per a *strings* Una de les grans avantatges de l'ús de kernels exposada en el treball és la possibilitat de treballar fàcilment amb dades de cadenes de caràcters. Així doncs, una extensió natural del projecte és aplicar el Kernel PCA en conjunts de dades que incorporin un bloc de dades *string*, com per exemple amb seqüències d'ADN o de proteïnes.

Integració de més de dos conjunts de dades S'ha explicat com els mètodes kernel proporcionen una manera natural d'integrar diferents fonts de dades, però s'han utilitzat per a combinar només dos fonts de dades òmiques. Per això sorgeix la idea d'estendre el treball afegint l'aplicació del Kernel PCA a un cas pràctic amb tres o més conjunts de dades òmiques.

Aplicacions Shiny Si bé s'ha assolit l'objectiu de crear aplicacions web interactives amb Shiny, les aplicacions creades podrien ser la base d'altres de més sofisticades i que incloguin altres dels procediments vistos per a visualitzar les variables originals. També es podria explorar l'ús del Shiny per a explicar el funcionament de les funcions kernel i dels mètodes kernel de forma més entenedora.

7.3 Valoració personal

La realització d'un treball d'aquesta magnitud ha estat tot un repte personal. Si bé a l'inici del projecte no ho veia del tot clar, veient fins a on s'ha arribat finalment fa que estigui satisfeta amb el resultat obtingut.

De totes les coses que m'ha aportat el TFG, la que més destacaria ha estat la gran quantitat i varietat de coneixements que m'ha permès assolir. Aquests coneixements van des del tipus de dades utilitzades i les tècniques aplicades, fins a nous paquets de R. Mirant enrere, recordo el que entenia o creia entendre quan llegia per primer cop documentació sobre kernels i sobre el Kernel PCA, i com ara quan torno a llegir aquests documents me n'adono de tot el que he après al llarg del projecte, cosa que valoro molt. Un dels coneixements més interessants assolits amb la realització d'aquest projecte ha estat precisament conèixer Shiny, una eina molt potent i que de forma molt fàcil et permet crear aplicacions senzilles per a il·lustrar de forma clara l'ús de moltes tècniques estadístiques.

Un altre aspecte a destacar és com de duradora ha estat la fase de documentació, no només degut a que calia conèixer molts conceptes nous, sinó també degut a la gran quantitat d'informació disponible en llibres, articles i Internet, ja que el tema tractat al treball és innovador i està d'actualitat. Aquesta etapa de documentació no consistia només en seguir un sol material, com les transparències d'una assignatura, sinó d'entendre conceptes diferents a partir de molts recursos, cosa que ha dificultat i allargat la tasca d'aprenentatge.

D'altra banda ressaltaria la gran dedicació en temps que ha requerit aquest projecte, cosa que fa que sigui molt difícil de compaginar amb la realització d'altres activitats, no només amb assignatures sinó en especial amb unes pràctiques empresarials.

Per últim, vull mostrar el meu agraïment a l'Esteban Vegas, qui m'ha assessorat en totes les fases de la realització del treball i sense l'ajuda del qual aquest projecte no hagués estat possible. I també a la meva família, per la seva paciència i suport aquests últims mesos.

Bibliografia

[1] Alpaydin, E. (2010). *Introduction to machine learning* (2a ed.). MIT Press.

[2] Breiman, L. (2001). *Statistical Modeling: The Two Cultures*. Statistical Science, vol. 16, nº 3, p. 199-231, 2001.

[3] Chapelle, O., Zien, A. i Scholkopf, B. (2006). *Semi-supervised learning*. MIT Press.

[4] Cortes, C. i Vapnik, V. (1995). *Support-vector networks*. Machine learning 20.3 (1995): 273-297.

[5] Déjean, S., González, I. i Lê Cao, K.-A. (2013). *Package "mixOmics"*. Disponible a `http://cran.r-project.org/web/packages/mixOmics/mixOmics.pdf`.

[6] De Keersmaecker, S., Thijs, I., Vanderleyden, J. i Marchal, K. (2006). *Integration of omics data: how well does it work for bacteria?*. Molecular Microbiology, vol. 62, Issue 5, p. 1239–1250, 2006.

[7] Domingos, P. (2012). *A Few Useful Things to Know about Machine Learning*. Communications of the ACM, 55 (10), 78-87, 2012.

[8] French, A. i Chess, S. (2005). *Canonical Correlation and Principal Components Analysis*. Department of Biology, San Francisco State University, California, USA.

[9] Fukumizu, K. (2008). *Methods with Kernels*. Institute of Statistical Mathematics, Graduate University for Advanced Studies, Japan.

[10] Ge, H., Walhout, A. i Vidal, M. (2003). *Integrating "omic" information: a bridge between genomics and systems biology*. TRENDS in Genetics, vol. 19, nº 10, 2003.

[11] Gomez-Cabrero, D., Abugessaisa, I., Maier, D., Teschendorff, A., Merkenschlager, M., Gisel, A., Ballestar, E., Bongcam-Rudloff, E., Conesa, A. i Tegnér, J. (2014). *Data integration in the era of omics: current and future challenges*. BMC Systems Biology 2014 8(Suppl 2):I1.

[12] Graffelman, J. (2013). *Canonical Correlation Analysis*. Department of Statistics and Operations Research, Universitat Politècnica de Catalunya, Barcelona, Spain.

[13] Hastie, T., Tibshirani, R. i Friedman, J. (2009). *The Elements of Statistical Learning* (2a ed.). Springer. New York, USA.

[14] Hofmann, T., Schoelkopf, B. i Smola, A. (2008). *Kernel methods in Machine Learning*. The Annals of Statistics, 2008, vol. 36, nº 3, p. 1171–1220.

[15] Hu, J. (2013). *About Data Mining*. Consultat: 16/05/14 a `http://www.aboutdm.com/2013/04/history-of-machine-learning.html`.

[16] Instituto de Salud Carlos III. Madrid. Consultat: 12/05/14 a `http://infobiochip.isciii.es/marcos%20navegacion.htm`.

[17] James, G., Witten, D., Hastie, T. i Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer. New York, USA.

[18] Kaelbling, LP., Littman, ML. i Moore, AW. (1996). *Reinforcement Learning: A Survey*. Journal of Artificial Intelligence Research, vol. 4, p. 237-285, 2006.

[19] Karatzoglou, A., Smola, A. i Hornik, K. (2013). *kernlab – An S4 Package for Kernel Methods in R*. Disponible a `http://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf`.

[20] Karatzoglou, A., Smola, A. i Hornik, K. (2014). *Package "kernlab"*. Disponible a `http://cran.r-project.org/web/packages/kernlab/kernlab.pdf`.

[21] Lê, S., Josse, J. i Husson, F. (2008). *FactoMineR: An R Package for Multivariate Analysis*. Journal of Statistical Software, vol. 25, Issue 1, 2008.

[22] Lê Cao, K.-A. (2012). *Unravelling "omics" data with the mixOmics R package, Illustration on several studies*. General presentation on mixOmics.

[23] Lee, JK. (2010). *Road to Statistical Bioinformatics*. Statistical Bioinformatics, p.1–6, 2010. Wiley-Blackwell Publishing.

[24] Leslie, C., Eskin, E. i Noble, W. (2002). textitThe spectrum kernel: a string kernel for svm protein classification. Pacific Symposium on Biocomputing, 2002, p. 564-575.

[25] Mann, A. (2014). *That Computer Actually Got an F on the Turing Test*. Wired. Consultat: 23/06/14 a `http://www.wired.com/2014/06/turing-test-not-so-fast/`.

[26] Markowetz, F. (2003). *Canonical Correlation Analysis with Kernels*. Computational Diagnostics Group Seminar, Berlin.

[27] Martin, PG., Guillou, H., Lasserre, F., Déjean, S., Lan, A., Pascussi, JM., Sancristobal, M., Legrand, P., Besse, P., Pineau, T. (2007). *Novel aspects of PPARα-mediated*

regulation of lipid and xenobiotic metabolism revealed through a multrigenomic study. Hepatology 2007, 54:767-777.

[28] Mason, H. (2010). *Machine Learning: A Love Story.* Consultat: 19/06/14 a `http://www.infoq.com/presentations/Machine-Learning/`.

[29] Minsky, M. i Papert, S. (1969). *Perceptrons: an introduction to computational geometry* (1a ed.). MIT Press, Cambridge MA, ISBN 0-262-63022-2.

[30] Mitchell, T. (2006). *The Discipline of Machine Learning.* Pittsburgh, PA, USA.

[31] O'Connor, B. (2008). *Statistics vs. Machine Learning, fight!.* Consultat: 01/07/14 a `http://brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight/`.

[32] Pagès, J. (2004). *Multiple Factor Analysis: main features and application to sensory data.* Revista Colombiana de Estadística, vol. 27, p.1-26, 2004.

[33] Paulson, J. (2013). *Developing Packages with RStudio.* Consultat: 26/07/14 a `https://support.rstudio.com/hc/en-us/articles/200486488-Developing-Packages-with-RStudio`.

[34] Primrose, SB. i Twyman, RM. (2003). *Principles of Genome Analysis and Genomics* (3a ed.). Blackwell Publishing.

[35] Puget, S., Philippe, C., Bax, D. A., Job, B., Varlet, P., Junier, M.-P., Andreiuolo, F., Carvalho, D., Reis, R., Guerrini-Rousseau, L. i altres. (2012). *Mesenchymal transition and PDGFRA amplification/mutation are key distinct oncogenic events in pediatric diffuse intrinsic pontine gliomas.* PLoS ONE 7(2), e30313.

[36] Quinlan, J. (1986). *Induction of decision trees.* Machine learning 1.1 (1986): 81-106.

[37] Reverter, F., Vegas, E. i Oller, JM. *Kernel-PCA data integration with enhanced interpretability.* BMC Systems Biology, 2014, 8(Suppl 2):S6.

[38] Rosenblatt, F. (1958). *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review 65.6(1958):386.

[39] Schneider, M. i Orchard, S. (2011). *Omics Technologies, Data and Bioinformatics Principles.* Bioinformatics for Omics Data, vol. 719, p.3–30, 2011.

[40] Schoelkopf, B., Smola, A. i Mueller, KR. (1999). *Kernel principal component analysis.* Advances in kernel methods, MIT Press, Cambridge, MA, USA 327-352.

[41] Shawe-Taylor, J. i Cristianini, N. (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[42] Shulaev, V. (2006). *Metabolomics technology and bioinformatics*. Briefings in Bioinformatics, vol. 7, nº 2, p. 128-139, 2006.

[43] Sutton, R. i Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA. ISBN 0-262-19398-1.

[44] Tenenhaus, A., Philippe, C., Guillemot, V., Lê Cao, K.-A., Grill, J. i Frouin, V. (2014). *Variable selection for generalized canonical correlation analysis*. Biostatistics, 15(3), pp. 569-583, 2014.

[45] Tibshiriani, R (2008). Glossary. Consultat: 29/06/14 a `http://statweb.stanford.edu/~tibs/stat315a/glossary.pdf`.

[46] Vert, J.-P., Tsuda, K. i Schölkopf,B. (2004). *A primer on kernel methods*. Kernel Methods in Computational Biology. MIT Press, p.35-70, 2004.

[47] Wallace, M. (2006). *Talk to Eliza*. Consultat: 23/06/14 a `http://www.manifestation.com/neurotoys/eliza.php3`.

[48] Wang, Q. (2012). *Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models*. CoRR, 2012.

[49] Williams, C. (2006). *Machine Learning and Statistics: What's the Connection?* Belfast, UK.

Annex A

CODI R

A.1 Capítol 3: Kernels amb R

A.1.1 Kernels per a vectors

```
> # Generació de dades
> set.seed(29)
> x <- round(rnorm(5), 2)
> y <- round(rnorm(5), 2)
> xy = cbind(x, y)
> xy

> # Kernel lineal
> val = vanilladot()
> val
> kernelMatrix(val, xy)

> # Kernel polinòmic
> pol <- polydot(degree = 2)
> pol
> kernelMatrix(pol, xy)
> pol_bis <- polydot(degree = 1, offset=0)
> pol_bis
> kernelMatrix(pol_bis, xy)

> # Kernel gaussià
> rbf <- rbfdot(sigma = 2)
> rbf
> kernelMatrix(rbf, xy)
> rbf_bis <- rbfdot(sigma = 0.25)
```

```
> rbf_bis
> kernelMatrix(rbf_bis, xy)

> # Comprovació de les propietats de les matrius kernel
> round(eigen(kernelMatrix(val, xy))$values, 4)
> round(eigen(kernelMatrix(pol, xy))$values, 4)
> round(eigen(kernelMatrix(rbf, xy))$values, 4)

> # Comparació de les matrius kernel
> l = kernelMatrix(val, xy)
> g = kernelMatrix(rbf, xy)
> p = kernelMatrix(pol, xy)
> summary(as.numeric(l))
> summary(as.numeric(g))
> summary(as.numeric(p))
> library(plotrix)
> par(mfrow = c(1,3), oma = c(0,0,0,0))
> hist(l, main = expression(paste("Kernel lineal")), xlab = "",
+      ylab="", col = "steelblue")
> hist(p, main = "Kernel polinòmic (d=2,alpha=1,c=1)", xlab = "",
+      ylab="", col = "steelblue")
> hist(g, main = expression(paste("Kernel gaussià (",sigma,"=2)")),
+      xlab = "", ylab="", col = "steelblue")
> par(mfrow = c(1,1), oma = c(0,0,0,0))
> par(mfrow = c(1,3), oma = c(0,0,0,0))
> color2D.matplot(l, main = expression(paste("Kernel lineal")),
+                 show.legend=TRUE)
> color2D.matplot(p, main = expression(paste("Kernel polinòmic (d = 2, ",
+                 alpha," = 1, c = 1)")), show.legend=TRUE)
> color2D.matplot(g, main = expression(paste("Kernel gaussià (",sigma,
+                 " = 2)")), show.legend=TRUE)
> par(mfrow = c(1,1), oma = c(0,0,0,0))
```

A.1.2 Kernels per a strings

```
> # Spectrum kernel
> sk <- stringdot(type="spectrum", length=2, normalized=FALSE)
> sk
> kernelMatrix(sk, c('radar','abracadabra'))
```

```
> # Proteïnes: dades
> library(seqinr)
> protdata <- read.fasta("dataset1_0.txt",seqtype="AA",as.string=TRUE)
> annotation <- getName(protdata)
> x <- unlist(getSequence(protdata,as.string=TRUE),recursive=FALSE)
> x = x[c(5,1411,643,8,9)]
> x = setNames(x, annotation[c(5,1411,643,8,9)])

> # Spectrum kernel
> sk <- stringdot(type="spectrum", length=3, normalized=FALSE)
> sk
> kernelMatrix(sk,x)
> sk <- stringdot(type="spectrum", length=3, normalized=TRUE)
> sk
> kernelMatrix(sk,x)
```

A.2 Capítol 4: Kernel PCA amb R

A.2.1 Visualització amb Kernel PCA (Figura 4.1)

```
> ### Generació de dades
> i = seq(from=-314,to=314,by=15)
> set.seed(8976565)
> x1 = c(sin(i/100)*10, sin(i/100)*40, sin(i/100)*100) + rnorm(1)
> set.seed(92654782)
> x2 = c(cos(i/100)*10, cos(i/100)*40, cos(i/100)*100) + rnorm(1)
> x = as.matrix(cbind(x1,x2))
> class = as.factor(c(rep(1, 42), rep(2, 42), rep(3, 42)))
> windows()
> par(oma=c(0,0,2,0), mfrow=c(2,2))
> # Dades originals
> plot(x1, x2, col = as.integer(class), pch = 16, main = "Dades originals",
+      xlab = expression("X"[1]), ylab = expression("X"[2]),
+      cex=0.9, cex.main=1.1)
> # Kernel lineal
> plot(rotated(kpca(x, kernel="vanilladot", kpar=list())),
+      col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9,
+      cex.main=1.1, main = 'Kernel lineal')
```

```
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> #Kernel polinòmic
> plot(rotated(kpca(x, kernel="polydot", kpar=list(degree=2, offset=1))),
+      col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel polinòmic, d=2,',alpha,'=1,c=1')))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> #Kernel gaussià
> plot(rotated(kpca(x, kernel="rbfdot", kpar=list(sigma=0.0005))),
+      col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià,',sigma,'=',"0.0005")))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Visualització amb Kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0), mfrow=c(1,1))
```

A.2.2 Aplicació del kernel PCA

```
> # Lectura dades: CONJUNT X
> myData <- read.csv("toyExample1.csv")
> x <- as.matrix(myData)
> set.seed(123)
> noise <- matrix(rnorm((nrow(x)*ncol(x)),0,0.5),nrow=nrow(x),ncol=ncol(x))
> x<- x + noise
> set.seed(49)
> x <- cbind(x, sample(rnorm(nrow(x),0,1)))
> colnames(x)[6] <-"x6"
> class = as.factor(c(rep("G1.1", 3), rep("G1.2", 3), rep("G2.1", 3),
+                     rep("G2.2", 3), rep("G3.1", 3), rep("G3.2", 3)))

> # Kernel PCA amb un kernel gaussià
> kpc <- kpca(x, kernel="rbfdot", kpar=list(sigma=0.002), features=2)
> par(mar = c(5.1, 4.1, 4.1, 2.1), xpd = FALSE, mfrow = c(1,1))
> par(oma=c(0,0,2,0))
> plot(rotated(kpc),col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", xlim = c(-3,3), ylim = c(-3,3),
+      pch = 16, cex=0.9, cex.main=1.1, main = expression(paste('Kernel
```

```
+      gaussià,', sigma, ' = 0.002')))
> title("Projecció dels individus en el primer pla factorial", outer=T,
+       cex.main=1.3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> par(oma=c(0,0,0,0))

> # Kernel PCA amb diversos kernels
> kpc <- kpca(x, kernel="rbfdot", kpar=list(sigma=0.002), features=2)
> kpc_lineal <- kpca(x, kernel="vanilladot", kpar=list(), features=2)
> kpc_poly <- kpca(x, kernel="polydot", kpar = list(degree=3,
+                  scale=1, offset=0), features=2)
> kpc_rbf <- kpca(x, kernel="rbfdot", kpar=list(sigma=0.5), features=2)
> par(oma=c(0,0,2,0), mfrow=c(2,2))
> # Lineal
> plot(rotated(kpc_lineal),col=as.integer(class) ,pch = 16, cex=0.9,
+      xlab="1st Principal Component",ylab="2nd Principal Component",
+      cex.main=1.1, main = 'Kernel lineal')
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Polinòmic
> plot(rotated(kpc_poly),col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel polinòmic, d=3,',alpha,'=1,c=0')))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià 1
> plot(rotated(kpc_rbf),col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel gaussià,', sigma,'= 0.5'))))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià 2
> plot(rotated(kpc),col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià, ', sigma, ' = 0.002')))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
```

```
> par(oma=c(0,0,0,0),mfrow=c(1,1))
```

A.2.3 Millora en la interpretació del kernel PCA

```
> # Representació de les variables originals
> par(oma=c(0,0,2,0), mfrow=c(1,2))
> makePlot(x, class, CC=0.002, v0 = c("x1", "x2", "x3", "x4", "x5"),
+          title = expression("Variables X" [1]*", X" [2]*", X" [3]*",
+                             X" [4]*", X" [5]), )
> makePlot(x, class, CC=0.002, v0 = c("x6"),
+          title = expression("Variable X" [6]))
> title("Variables originals en el Kernel PCA", outer=T, cex.main=1.3)
> par(mfrow=c(1,1), oma=c(0,0,0,0))

> # Representació de combinacions lineals de les variables originals
> par(oma=c(0,0,2,0), mfrow=c(1,1))
> makePlot_3(x, class, CC=0.002, v01=c("x1","x2"), v02=c("x3","x4","x5"),
+           title1 = expression("Variables X" [1]*"+X" [2]*" i
+                               X" [3]*"+X" [4]*"+X" [5]))
> title("Combinacions lineals de variables originals en el Kernel PCA",
+       outer=T, cex.main=1.3)

> # Lectura de dades: CONJUNT Y
> myData2 <- read.table("generacio_Y.txt", header=TRUE, dec = ",")
> y <- as.matrix(myData2)
> set.seed(256)
> noise <- matrix(rnorm((nrow(y)*ncol(y)),0,0.5),nrow=nrow(y),ncol=ncol(y))
> y<- y + noise
> set.seed(49)
> y <- cbind(y, sample(rnorm(nrow(y),0,1)))
> colnames(y)[6] <-"y6"

> # Kernel PCA amb representació de les variables originals
> par(oma=c(0,0,2,0), mfrow=c(1,2))
> plot(rotated(kpca(y, kernel="rbfdot", kpar=list(sigma=0.022))),
+      col=as.integer(class),xlab="1st Principal Component",
+      ylab="2nd Principal Component", xlim = c(-3,3), ylim = c(-3,3),
+      pch = 16, cex=0.9, cex.main=1.1,
+      main = 'Kernel gaussià, sigma = 0.022')
> title("Projecció dels individus en el primer pla factorial",
```

```
+       outer=T, cex.main=1.3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> makePlot(y, class, CC=0.022, v0 = c("y1", "y4"),
+          title = expression(bold("Variables Y" [1]*", Y" [4])))
> par(oma=c(0,0,1,0), mfrow=c(1,1))

> # Integració de dades amb Kernel PCA i representació
> # de les variables originals
> par(oma=c(0,0,2,0), mfrow=c(1,2))
> makePlot_2(x, y, class, CC = c(0.002, 0.022), v01 = c("x3"),
+            v02 = c("y4"), xl = c(-3, 3), yl = c(-3, 3))
> makePlot_2(x, y, class, CC = c(0.002, 0.022), v01 = c("x1"),
+            v02 = c("y1"), xl = c(-3, 3), yl = c(-3, 3))
> title("Integració dels dos conjunts de dades", outer = T, cex =1.3)
> par(oma=c(0,0,0,0), mfrow=c(1,1))

> # Descobrir la interpretació de les variables originals
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> tab1 = makePlot_4(x, class, CC=0.002, v0=c("x1","x2","x3","x4","x5"),
+       title1 = expression(bold(paste("Direcció ",bar(w),"=[-1,1]"))),
+       v_dir = c(-1,1))
> tab2 = makePlot_4(x, class, CC=0.002, v0=c("x1","x2","x3","x4","x5"),
+       title1 = expression(bold(paste("Direcció ",bar(w),"=[-1,1]"))),
+       v_dir = c(1,-1))
> tab3 = makePlot_4(x, class, CC=0.002, v0=c("x1","x2","x3","x4","x5"),
+       title1 = expression(bold(paste("Direcció ",bar(w),"=[-1,1]"))),
+       v_dir = c(1,0.5))
> title("Descobrint la interpretabilitat de les variables", outer=T,
+       cex.main=1.3)
> par(oma=c(0,0,0,0), mfrow=c(1,1))

> # Resultats de les correlacions
> library(xtable)
> print(xtable(tab1, digits = 4))
> print(xtable(tab2, digits = 4))
> print(xtable(tab3, digits = 4))
```

A.3 Capítol 5: Nutrimouse dataset

```
> # Lectura dades
> library(mixOmics)
> data(nutrimouse)
> X <- nutrimouse$gene
> Y <- nutrimouse$lipid
> f1 <- nutrimouse$genotype
> f2 <- nutrimouse$diet
```

A.3.1 Gene expression data analysis

1. KERNEL PCA

```
> # PCA lineal
> nut.pca1 = PCA(cbind(X, f1), ncp=5, quali.sup=121)
> nut.pca2 = PCA(cbind(X, f2), ncp=5, quali.sup=121)
> # Kernel polinòmic
> kpc_pol = kpca(as.matrix(X), kernel= polydot, kpar=list(degree=2))
> # Kernel gaussià
> kpc_g = kpca(as.matrix(X), kernel= rbfdot, kpar=list(sigma=0.05))

> ### Gràfic 1
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(nut.pca1, choix ="ind", habillage=121,
+      label = "quali", title = "PCA lineal", cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(f1),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel polinòmic,
+      d = 2, ', alpha, ' = 1, c = 1'))))
> legend("topright", legend = levels(f1), pch = 16,
+        col = as.numeric(as.factor(levels(f1)))[order(levels(f1))],
+        title = "Genotip")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g), col=as.integer(f1),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
```

```
+      main = expression(bold(paste('Kernel gaussià, ', sigma, ' = 0.05'))))
> legend("topright", legend = levels(f1), pch = 16,
+        col = as.numeric(as.factor(levels(f1)))[order(levels(f1))],
+        title = "Genotip")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))

> ### Gràfic 2
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(nut.pca2, choix ="ind", habillage=121,
+      label = "quali", title = "PCA lineal", cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(f2),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel polinòmic, d = 2, ',
+      alpha, ' = 1, c = 1'))))
> legend("topright", legend = levels(f2), pch = 16,
+        col = as.numeric(as.factor(levels(f2)))[order(levels(f2))],
+        title = "Dieta")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g), col=as.integer(f2),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel gaussià, ', sigma, ' = 0.05'))))
> legend("topright", legend = levels(f2), pch = 16,
+        col = as.numeric(as.factor(levels(f2)))[order(levels(f2))],
+        title = "Dieta")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))
```

2. INPUT VARIABLES REPRESENTATION

```
> makePlot_1(X, f1, CC = 0.05, v0 = c("AOX", "CAR1"),
+            title = "Kernel PCA: Gene Expression")
```

A.3.2 Fatty acids concentration data analysis

1. KERNEL PCA

```
> # PCA lineal
> nut.pca1 = PCA(cbind(Y, f1), ncp=5, quali.sup=22)
> nut.pca2 = PCA(cbind(Y, f2), ncp=5, quali.sup=22)
> nut.pca = PCA(cbind(Y, f2), ncp=5, quali.sup=c(21,22))
> # Kernel polinòmic
> kpc_pol = kpca(as.matrix(Y), kernel= polydot, kpar=list(degree=2))
> # Kernel gaussià
> kpc_g = kpca(as.matrix(Y), kernel= rbfdot, kpar=list(sigma=0.001))

> ### Gràfic 1
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(nut.pca1, choix ="ind", habillage=22, axes = c(1,3),
+      label = "quali", title = "PCA lineal", cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(f1),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel polinòmic, d = 2, ',
+      alpha, ' = 1, c = 1'))))
> legend("topright", legend = levels(f1), pch = 16,
+        col = as.numeric(as.factor(levels(f1)))[order(levels(f1))],
+        title = "Genotip")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g), col=as.integer(f1),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel gaussià, ', sigma, ' = 0.05'))))
> legend("topright", legend = levels(f1), pch = 16,
+        col = as.numeric(as.factor(levels(f1)))[order(levels(f1))],
+        title = "Genotip")
```

```
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))

> ### Gràfic 2
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(nut.pca2, choix ="ind", habillage=22,
+      label = "quali", title = "PCA lineal", cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(f2),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel polinòmic, d = 2, ',
+      alpha, ' = 1, c = 1'))))
> legend("topright", legend = levels(f2), pch = 16,
+        col = as.numeric(as.factor(levels(f2)))[order(levels(f2))],
+        title = "Dieta")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g), col=as.integer(f2),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(bold(paste('Kernel gaussià, ', sigma, ' = 0.05'))))
> legend("topright", legend = levels(f2), pch = 16,
+        col = as.numeric(as.factor(levels(f2)))[order(levels(f2))],
+        title = "Dieta")
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))

> ### Gràfic 3
>
> plot(rotated(kpc_g),col=as.integer(f1),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = as.integer(f2), cex=0.9,
+      main = expression(paste('Kernel gaussià, ', sigma, ' = 0.001')))
> legend("topright", legend=levels(f1), pch=16,title="Genotip", cex = 0.9,
+        col=as.numeric(as.factor(levels(f1)))[order(levels(f1))])
```

```
> legend("bottomright", legend=levels(f2), title="Dieta", cex = 0.9,
+        pch=as.numeric(as.factor(levels(f2))))
> abline(h=0, lty=3)
> abline(v=0, lty=3)
```

2. INPUT VARIABLES REPRESENTATION

```
> makePlot_1(Y, f1, f2, CC = 0.001, v0 = c("C20.2n.6", "C16.0"), xl =c(-5,5),
+             title = "Kernel PCA: Fatty Acids Concentration",yl =c(-5,5))
> makePlot_1(Y, f1, f2, CC = 0.001, v0 = c("C16.1n.7"), xl =c(-5,5),
+             title = "Kernel PCA: Fatty Acids Concentration",yl =c(-5,5))
```

A.3.3 Gene expression and Fatty acids data integration

```
> makePlot_2(X, Y, f1, f2, CC = c(0.05, 0.001),
+            v01 = c("AOX", "CAR1"), v02 = c("C20.2n.6", "C16.0"),
+            title = "Kernel PCA: Gene & Fatty Acids Integration")
```

A.3.4 Revealing the interpretability of variables

```
> makePlot_4(X, Y, f1, f2, CC = c(0.05, 0.001),
+            v01 = colnames(X), v02 = colnames(Y),
+            title = "Kernel PCA: Interpretability of variables",
+            v_dir = c(1,0.3))
> gen = read.table("alin_1.txt", header = T)
> names(gen)[1] = "Gene"
> print(xtable(tail(gen,n=3), digits = 4))
> print(xtable(head(gen,n=3), digits = 4))
> FA = read.table("alin_2.txt", header = T)
> names(FA)[1] = "Acid"
> print(xtable(tail(FA,n=3), digits = 4))
> print(xtable(head(FA,n=3), digits = 4))
```

A.4 Capítol 5: Pediatric glioma data

```
> # Lectura dades
> library(gliomaData)
> # Blocs
> data(ge_cgh_locIGR)
> x = ge_cgh_locIGR$multiblocks$GE
```

```
> y = ge_cgh_locIGR$multiblocks$CGH
> dim(x)
> dim(y)
> ge = as.data.frame(x)
> cgh = as.data.frame(y)
> # Factor
> loc = as.factor(ge_cgh_locIGR$ylabel)
> levels(loc) = c("hemi", "dipg", "midl")
> # Annotacions
> data(GE_annot)
> data(CGH_annot)
> # Noms de les variables: CGH data
> t = NULL
> for(i in 1:23) t = c(t, 1:table(CGH_annot$Chr)[i])
> colnames(cgh) = paste(CGH_annot$Chrom, ".", t, sep = "")

> # Freqüències absolutes i relatives del factor localització
> library(Epi)
> print(stat.table(list(Localització=loc), list(n = count(),
+       "%" = percent(loc)), margin=T))
```

A.4.1 Gene expression data analysis

1. KERNEL PCA

```
> # PCA lineal
> ge.pca = PCA(cbind(ge, loc), ncp=5, quali.sup=15703, graph = FALSE)
> # Kernel polinòmic
> kpc_pol = kpca(x, kernel= polydot, kpar=list(degree=3))
> # Kernel gaussià
> kpc_g = kpca(x, kernel= rbfdot, kpar=list(sigma=0.0001))

> ### Representació dels tumors
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(ge.pca, choix ="ind",habillage=15703, label = "quali",
+      title = "PCA lineal", axes = c(1,3), cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
```

```
+      main = expression(paste('Kernel polinòmic, d=3,',alpha,'=1,c=1')))
> legend("topright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g),col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià, ', sigma, ' = 0.0001')))
> legend("topright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))

> ### Variació de les representacions en funció de la sigma
> par(oma=c(0,0,2,0), mfrow=c(1,2))
> plot(rotated(kpca(x, kernel= rbfdot, kpar=list(sigma=0.00001))),
+      col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià,', sigma,'=0.00001')))
> legend("topright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> plot(rotated(kpca(x, kernel= rbfdot, kpar=list(sigma=0.05))),
+      col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià,', sigma,'=0.05')))
> legend("bottomright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))
```

2. INPUT VARIABLES REPRESENTATION

```
> makePlot_1(ge, loc, CC = 0.0001, v0 = c("A_23_P205428", "A_23_P323272"),
+            title = "Kernel PCA: Gene Expression", xl = c(-3.5,3.5),
+            yl = c(-3.5,3.5))
```

A.4.2 Genome alteration data analysis

1. KERNEL PCA

```
> # PCA lineal
> cgh.pca = PCA(cbind(cgh, loc), ncp=5, quali.sup=1230, graph = FALSE)
> # Kernel polinòmic
> kpc_pol = kpca(y, kernel= polydot, kpar=list(degree=2))
> # Kernel gaussià
> kpc_g = kpca(y, kernel= rbfdot, kpar=list(sigma=0.015))

> ### Representació dels tumors
>
> par(oma=c(0,0,2,0), mfrow=c(1,3))
> # Lineal
> plot(cgh.pca, choix ="ind",habillage=1230, label = "quali",
+      title = "PCA lineal", cex = 1.25)
> # Polinòmic
> plot(rotated(kpc_pol), col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel polinòmic, d=2,',alpha,'=1,c=1')))
> legend("topright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> # Gaussià
> plot(rotated(kpc_g),col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main = expression(paste('Kernel gaussià, ', sigma, ' = 0.015')))
> legend("topright", legend = levels(loc), pch = 16, col = 1:3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> title("Projecció dels individus amb kernel PCA", outer=T, cex.main=1.3)
> par(oma=c(0,0,0,0),mfrow=c(1,1))
```

A.4.3 Gene expression and CGH data integration

1. KERNEL PCA

```
> kmatrixX1 = kernelMatrix(kernel = rbfdot(sigma = 0.0001), x)
> kmatrixX2 = kernelMatrix(kernel = rbfdot(sigma = 0.001), y)
> kmatrixX <- 0.5 * kmatrixX1 + 0.5 * kmatrixX2
```

```
> kmatrixX <- as.kernelMatrix(kmatrixX)
> kpc <- kpca(kmatrixX, features = 2)
> par(oma=c(0,0,2,0))
> plot(rotated(kpc),col=as.integer(loc),xlab="1st Principal Component",
+      ylab="2nd Principal Component", pch = 16, cex=0.9, cex.main=1.1,
+      main ="Projecció dels individus en el primer pla factorial",
+      xlim = c(-2.5,2.5), ylim = c(-2.5,2.5))
> legend("topright", legend = levels(loc), col = 1:3, pch = 16)
> title("Kernel PCA data integration", outer=T, cex.main=1.3)
> abline(h=0, lty=3)
> abline(v=0, lty=3)
> par(oma=c(0,0,0,0))
```

2. INPUT VARIABLES REPRESENTATION

```
> # Gràfic 1
> makePlot_2(ge, cgh, loc, CC = c(0.0001,0.001), v01="A_23_P205428",
+            v02 = "chr1.3", xl=c(-3,3), yl=c(-3,3))
> # Gràfic 2
> makePlot_2(ge, cgh, loc, CC = c(0.0001,0.001), v01="A_23_P323272",
+           v02 = "chr4.3", xl=c(-3,3), yl=c(-3,3))
```

A.4.4 Revealing the interpretability of variables

```
> makePlot_4(ge, cgh, loc, CC = c(0.0001, 0.001), v01 = colnames(ge),
+            v02 = colnames(cgh), v_dir = c(0.8,0.25),
+            title = "Kernel PCA: Interpretability of variables")
> gen = read.table("alin_1.txt", header = T)
> names(gen)[1] = "Gene"
> print(xtable(tail(gen,n=3), digits = 4))
> print(xtable(head(gen,n=3), digits = 4))
> DNA = read.table("alin_2.txt", header = T)
> names(DNA)[1] = "Segment"
> print(xtable(tail(DNA,n=3), digits = 4))
> print(xtable(head(DNA,n=3), digits = 4))
```