

UNIVERSITAT DE BARCELONA



INSTITUT DE CIÈNCIES DEL COSMOS

FACULTAT DE FÍSICA

---

# Memòries de Pràctiques d'Empresa

---

*Supervisors:*

Marcel BERNET MARTÍ,

[mbernet@fqa.ub.edu](mailto:mbernet@fqa.ub.edu)

Albert GARCIA GUTIERREZ,

[albertgarcia@icc.ub.edu](mailto:albertgarcia@icc.ub.edu)

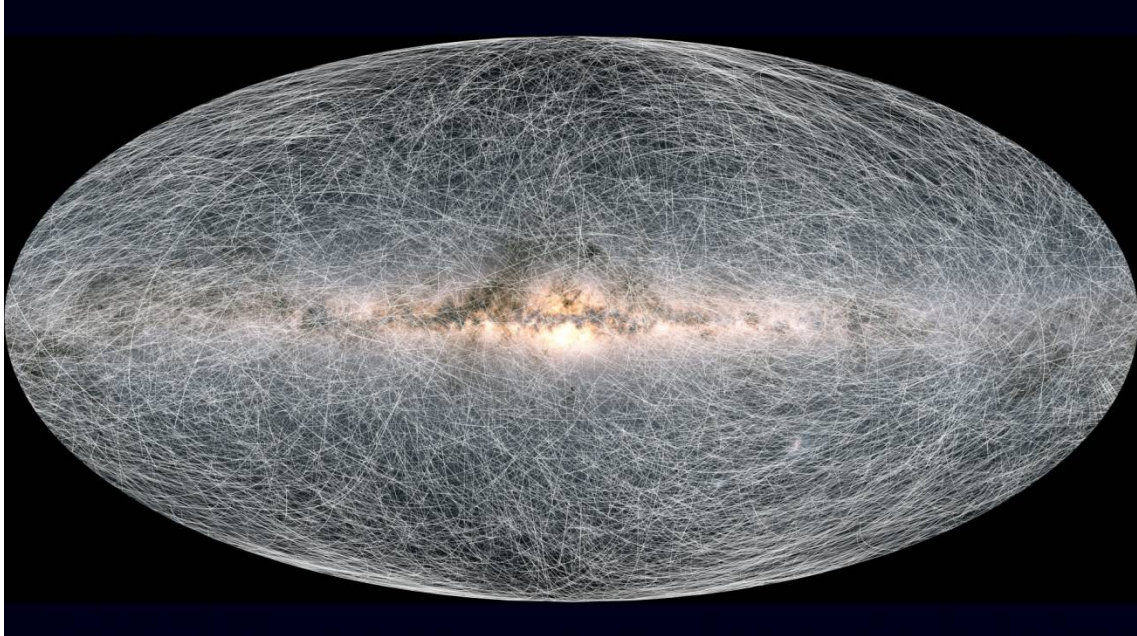
*Autor:*

Jan NOGUÉ GÓMEZ

25 de juliol de 2021

## Abstract

Les pràctiques d'empresa les he dut a terme a la divisió tecnològica del Institut de Ciències del Cosmos (ICCUB), al Parc Científic de Barcelona. El projecte consistia en processament, anàlisi i visualització de dades del catàleg de Gaia Early Data Release 3 (EDR3). La Fig.1 representa les velocitats tangencials de les estrelles del catàleg de Gaia amb ratlles blanques, simulant la direcció de propagació. L'objectiu final del projecte és aconseguir una animació d'aquesta figura per aconseguir una idea més clara de l'evolució de l'estructura de la Via Làctia.



**Figura 1:** Representació de les estrelles de la Via Làctia amb la seva velocitat tangencial.

## 1 Introducció

Gaia és un observatori espacial de l'Agència Espacial Europea (ESA), llançada el 2013 i, com a mínim, funcional fins el 2022. Malgrat tot, les dades indiquen que Gaia seguirà operativa fins el novembre de 2024 ja que els seus detectors no s'estan degradant a la velocitat esperada i gasta menys combustible del previst.

Gaia està dissenyada per l'astrometria: mesurar posicions, distàncies i velocitats d'estrelles amb una precisió sense precedents. La missió té com a objectiu obtenir el catàleg 3D més complet que s'ha fet mai de la Via Làctia. Aquest catàleg s'espera que obtingui dades astromètriques d'aproximadament un bil·lió d'objectes astronòmics (majoritàriament estrelles, però també mil·lions d'exoplanetes similars al tamany de Júpiter més enllà del Sistema Solar, 500.000 quàsars i desenes de milers d'asteroides i cometes).

Per fer-ho, l'observatori espacial està dissenyat per monitoritzar cada un dels cossos unes 70 vegades els primers cinc anys, una tècnica emprada per aconseguir un gran volum de dades en poc temps que servirà per obtenir la precisió desitjada.

Les mesures espectromètriques ens donaran informació vital de les característiques físiques de les estrelles observades, caracteritzant la lluminositat, temperatura efectiva, gravetat i composició química. Aquestes mesures donaran peu a dades que serviran per analitzar preguntes relacionades amb l'origen, estructura i evolució de la nostra galàxia [2].

## 1.1 Sensors i instruments

Gaia conté tres sensors principals, cada un enfocat a mesurar diferents característiques dels cossos celestes:

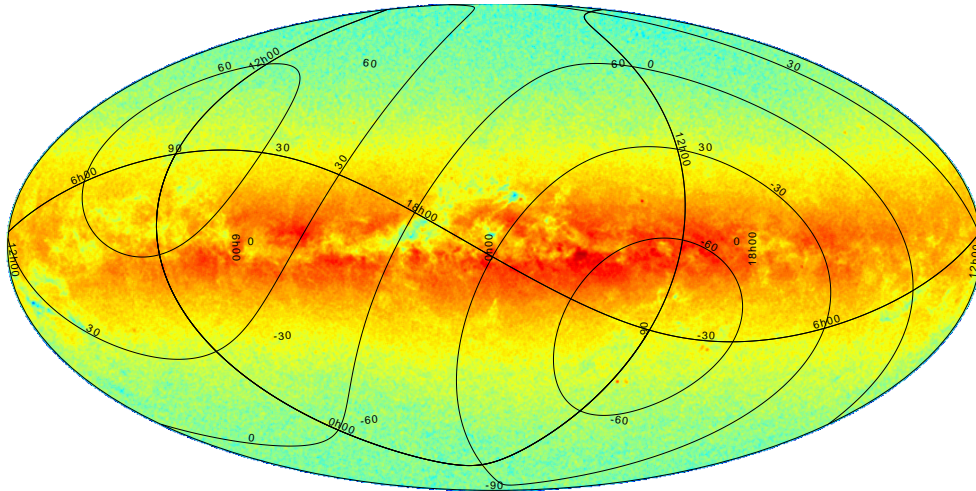
1. El sensor astromètric (Astro) determina amb precisió les posicions de totes les estrelles més brillants de magnitud 20 mitjançant la posició angular. Utilitzant les mesures en el període de cinc anys, serà possible determinar la paral·laxi, i per tant la distància, i la direcció i magnitud de la velocitat de propagació projectada al pla del cel.
2. L'instrument fotomètric (BP/RP) adquireix la lluminositat de les estrelles per al voltant dels 320-1000 nanòmetres de banda espectral, de totes les estrelles més brillants de magnitud 20. Els fotòmetres vermell i blau serveixen per determinar característiques estel·lars com la temperatura, massa, edat i composició elemental dels cossos celestes.
3. L'espectòmetre de velocitat radial (RVS) obté les velocitats radials dels cossos celestes amb una precisió entre 1 km/s i 30 km/s. [2]

## 2 Obtenció i visualització de dades

Les dades del catàleg EDR3 [5] les vaig obtenir de la pàgina web de Gaia archive [1]. Per descarregar un nombre elevat de dades i amb uns criteris concrets, vaig utilitzar el llenguatge ADQL [8]. Per visualitzar les dades durant tot el projecte, vaig utilitzar el software especialitzat en astrometria, TOPCAT [10] i la llibreria de Python Matplotlib [6].

## 3 Primers passos del projecte

Les primeres setmanes consistien amb familiaritzar-me amb el meu entorn de treball. La meua primera tasca era descarregar de la pàgina web de Gaia archive les coordenades equatorials amb les seves velocitats ( $\alpha$ ,  $\delta$ ,  $\mu_{\alpha*}$ ,  $\mu_{\delta}$ ), de les estrelles que tinguessin una velocitat radial no nul·la i amb una magnitud aparent mitjana en la banda G inferior a 13 (A.1). Obtinc aproximadament uns 5 milions d'estrelles. Represento aquestes estrelles en coordenades galàctiques i amb la projecció aitoff (Fig.2).

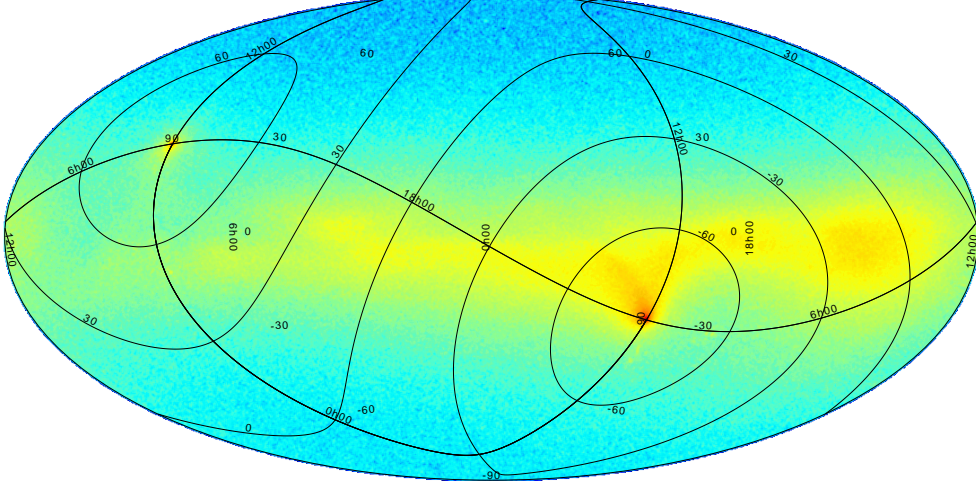


**Figura 2:** Projecció aitoff del mapa de densitat obtingut amb A.1 en coordenades galàctiques. Les línies negres representen els eixos de les coordenades equatorials.

Amb el mateix TOPCAT, propago les coordenades celestes de totes les estrelles fent servir una aproximació lineal,

$$\begin{aligned}\alpha &= \alpha_0 + \mu_{\alpha\star}t \\ \delta &= \delta_0 + \mu_{\delta}t\end{aligned}\tag{1}$$

Amb  $\mu_{\alpha\star} = \mu_{\alpha} \cos \delta$ . A mesura que propago, les estrelles s'apropen als pols situats a  $\pm 90^\circ$  i desapareixen de la representació, ja que els pols no estan definits en la propagació lineal (Fig. 3). He de trobar alguna manera d'evitar que les estrelles interactuin amb els pols a mesura que les propago.



**Figura 3:** Projectió aitoff en coordenades equatorials propagades  $t = 10^7$  anys.

Per evitar els pols a  $\pm 90^\circ$  penso en canviar de coordenades i fer la propagació en coordenades galàctiques, que tenen els pols al nord/sud i són evitables per la majoria d'estrelles que es propaguen en la direcció del pla galàctic.

El catàleg de Gaia EDR3 té les coordenades galàctiques  $(l, b)$  de les estrelles però no té les velocitats  $(\mu_{l\star} = \mu_l \cos b, \mu_b)$ . Afortunadament, utilitzant matrius de rotació, es poden calcular les velocitats en coordenades galàctiques a partir de les coordenades i velocitats equatorials [9]. Implemento un programa amb Python que calculi aquestes matrius de rotació i calculi les velocitats en coordenades galàctiques (B.1).

També trobo una llibreria de Python anomenada Astropy [3],[4] que realitza aquestes matrius de rotació de manera directa i amb menys línies de codi.



## 4 Desenvolupament de l'algoritme de propagació

A partir d'aquest moment treballarem desenvolupant un algoritme amb Python. L'objectiu es obtenir un fitxer.csv amb les posicions de cada estrella per a cada temps. Propago linealment en coordenades equatorials i obtinc un resultat similar al de la Fig.3, però les estrelles ara desapareixen als pols del nord/sud. Sí que he aconseguit que es perdin menys estrelles perquè ara els pols no estan al pla de propagació galàctic però en segueixo perdent per a temps llargs. El següent pas és assumir que les estrelles acabaran interectuant amb les pols i, per tant, l'algoritme ha d'incorporar correccions a les propagacions de les coordenades equatorials al voltant dels mateixos. Els angles  $l, b$  estan definits de la següent manera:

$$l \in [0^\circ, 360^\circ], \quad b \in [-90^\circ, 90^\circ] \quad (2)$$

Propagant indefinidament, les coordenades acabaran surtint d'aquest rang, TOPCAT no sabrà com representar-les perquè els angles no estaran definits i les estrelles desapareixeran. Per la correcció de l'angle  $l$ , utilitzo l'operació mòdul (%) de Python ja que, per exemple, l'angle  $l = 361^\circ$  és l'angle  $l = 1^\circ$  (dóna la volta).

La correcció de l'angle  $b$  no és tant trivial ja que l'angle  $b = 91^\circ$  no és l'angle  $b = -89^\circ$ . De fet, necessitem mínim dues cartes per representar tots els punts d'una esfera<sup>1</sup>. Per solucionar aquesta propagació, en un entorn proper als pols  $b = 90^\circ$  i  $b = -90^\circ$ , s'ha de canviar a coordenades equatorials i propagar en aquestes mateixes coordenades. Una vegada obtingudes aquestes noves posicions propagades en coordenades equatorials  $(\alpha_f, \delta_f)$ , hem de convertir-les en galàctiques de nou  $(l_f, b_f)$ , obtenint així les coordenades galàctiques propagades a prop dels pols.

Com a resum del nostre algoritme, hem de calcular les velocitats en coordenades galàctiques a partir de les posicions i velocitats en equatorials i propagar en coordenades galàctiques però corregint amb un canvi de coordenades quan la coordenada propagada estigui aprop de  $b = \pm 90^\circ$ .

Tots aquests càlculs requereixen un cost computacional elevat. Sorgeix la necessitat d'utilitzar més d'una CPU (l'ordinador del despatx del Parc Científic de Barcelona en té vuit) per la realització dels càlculs i posteriors correccions. És amb aquest motiu que investigo sobre el multiprocessing de Python [7]. El multiprocessing consisteix en dividir un fitxer en un objecte anomenat *dataframe*, assignar-li una CPU i realitzar els càlculs. Una vegada s'han realitzat tots els càlculs, ajunta totes les *dataframe* en un sol fitxer. El multiprocessing resulta senzill si les operacions que s'han de fer en un *dataframe* no depenen dels resultats de les altres, ideal per el nostre cas particular, on propago cada estrella independentment de la posició de les altres. Utilitzant el multiprocessing, les velocitats en coordenades galàctiques es calculen de manera molt més ràpida i senzilla (B.2).

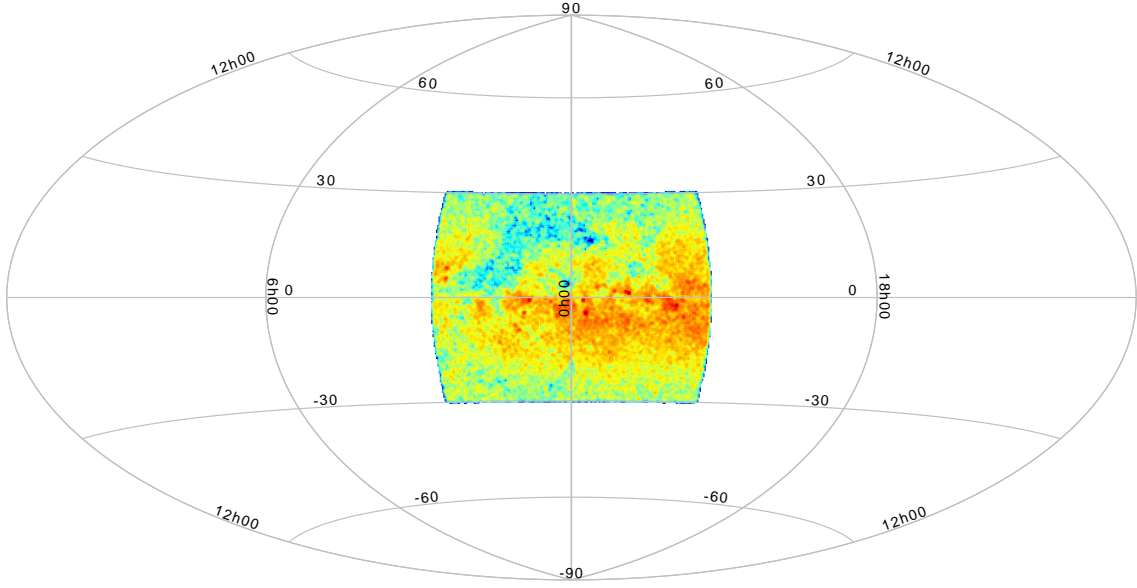
A l'hora de propagar, he de fer passos petits de temps per què així el programa pot anar corregint les coordenades a cada pas. Si faig servir passos més grans de  $t = 10^4$  anys, en l'aproximació lineal (Eq.1), a l'hora de canviar de coordenades equatorials a galàctiques alguns angles surten fora del rang en el que estan definits i la llibreria Astropy dóna error.

Per tant, per calcular la propagació per  $10^5$  anys, el programa ha de fer 9 iteracions. Per calcular la propagació per  $10^6$  anys, n'ha de fer 90, i per calcular la propagació per  $10^7$  anys, n'ha de fer 900. Tantes iteracions, malgrat el multiprocessing, té un cost computacional elevat així que per comprovar que l'algoritme funciona com hauria, calculo la propagació del centre galàctic, amb menys estrelles, abans d'aplicar-ho a tota la bòveda celeste. També incorpore un criteri a la Query per tenir les estrelles amb un error inferior al 5% a les velocitats (A.2) per obtenir més precisió.

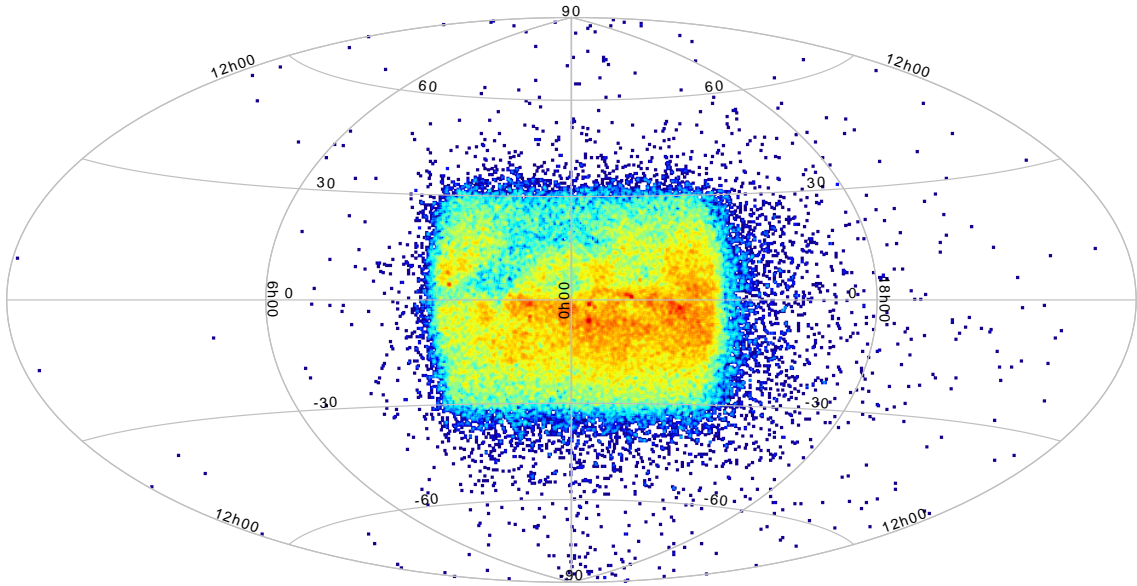
---

<sup>1</sup>Una esfera ( $S^2 \subset \mathbb{R}^3$ ) és una varietat analítica de dimensió 2. Una manera de fer-ne una carta per representar cada punt seria prendre coordenades esfèriques, amb els angles polars  $0 \leq \theta \leq \pi$  i azimutal  $0 \leq \phi \leq 2\pi$ . D'aquesta manera, tots els punts queden representats en un pla  $\mathbb{R}^2$  amb aquests angles de coordenades. És cert, però tenim que el meridià de GW (el de referència) té dos angles azimutals  $\varphi = 0$  i  $\varphi = 2\pi$ . Cal fer  $0 \leq \theta \leq \pi$  perquè la relació sigui 1 a 1. Igualment, els pols, caracteritzats amb  $\theta = 0$  o  $\theta = \pi$  tenen infinits  $\varphi$  que hi corresponen. L'única manera d'arreglar-ho és eliminant aquests punts conflictius de la carta. Aleshores, sí que es podria considerar l'aplicació de les coordenades esfèriques un homeomorfisme, en tant que tots els punts tenen una aplicació 1 a 1. Però, aleshores, no tindríem tota l'esfera coberta amb oberts, caldria parxegar-la amb algun altre obert que contingui els punts que hem deixat sense representar, és a dir, necessitem una altra carta. De fet, es pot demostrar que és impossible cobrir una esfera amb una sola carta, és segur que mínim calen dues.

Obtinc els següents resultats.

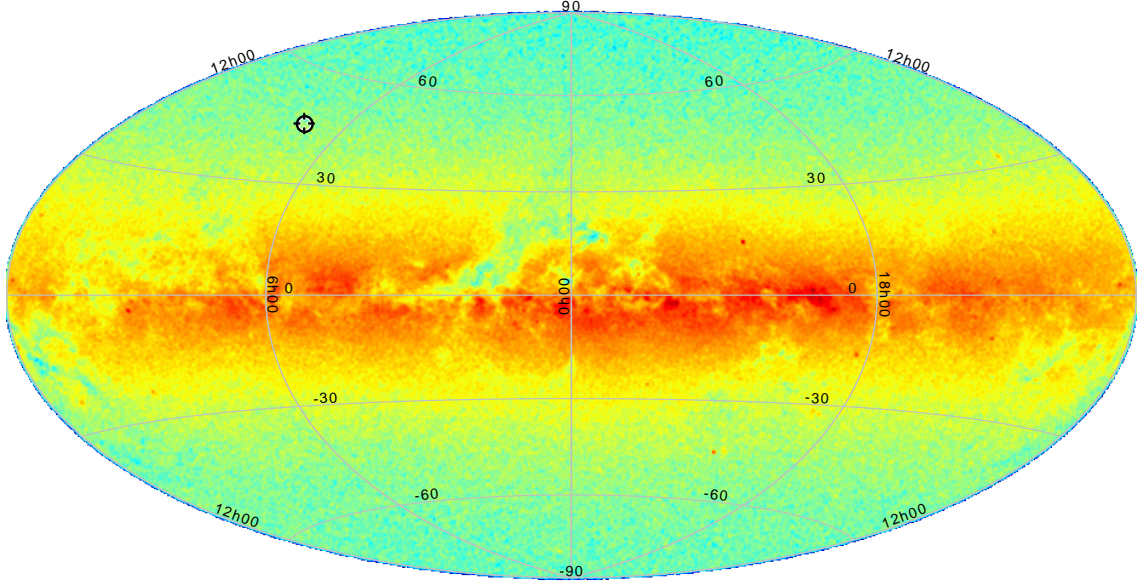


**Figura 4:** Mapa de densitat del centre galàctic sense propagar.

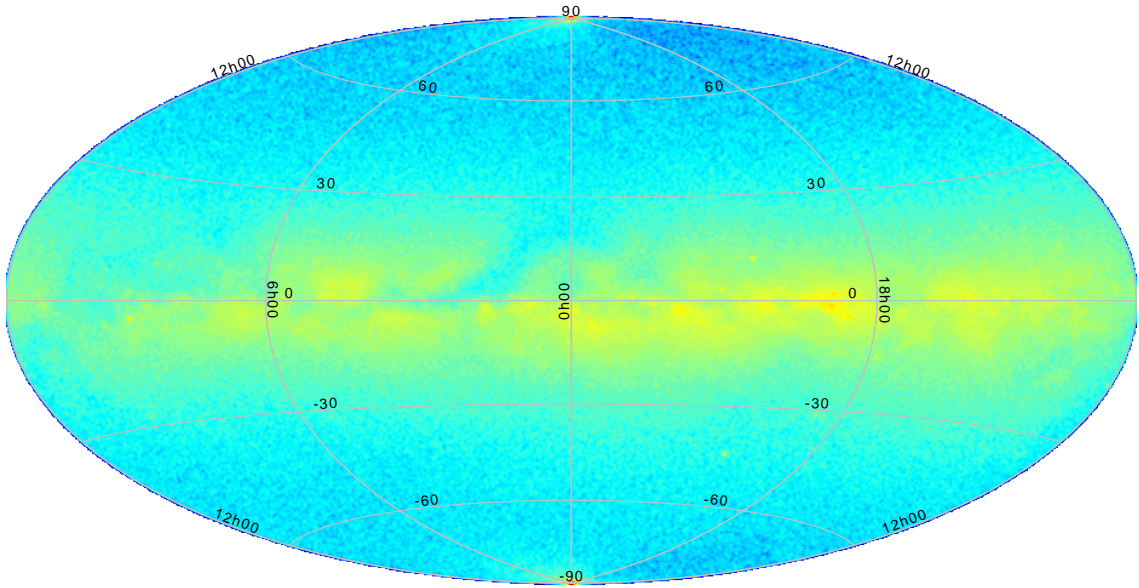


**Figura 5:** Mapa de densitat del centre galàctic amb una propagació de  $t = 10^6$  anys.

Els resultats són prometedors per el centre galàctic però queda comprovar com actuen les estrelles al voltant dels pols. Per la propagació de tota la bòveda celeste, utilitzo la Query [A.1](#) amb l'addició d'un error en les velocitats inferior al 5%. Surten uns tres milions d'estrelles i obtinc els següents resultats.



**Figura 6:** Mapa de densitat amb projecció aitoff de la bòveda celeste en coordenades galàctiques.



**Figura 7:** Mapa de densitat amb projecció aitoff de la bòveda celeste en coordenades galàctiques amb una propagació de  $t = 2 \cdot 10^6$  anys.

De la Fig. 7 veiem que les estrelles s'acomulen als pols. Aparentment no té sentit ja que amb una propagació lineal les estrelles s'haurien de propagar indefinidament sense concentrar-se en cap lloc concret. Procedim a repassar l'algoritme de correcció quan  $|b| \geq 85^\circ$  per determinar per què les estrelles segueixen acumulant-se al pols. La propagació en coordenades galàctiques es fa de la següent manera:

$$\begin{aligned} new_l &= l + \mu_l t \\ new_b &= b + \mu_b t \end{aligned} \tag{3}$$



Si  $|new_b| \geq 85^\circ$ , transformem les coordenades inicials  $(l, b)$  a coordenades equatorials  $(\alpha, \delta)$  i propago en coordenades equatorials.

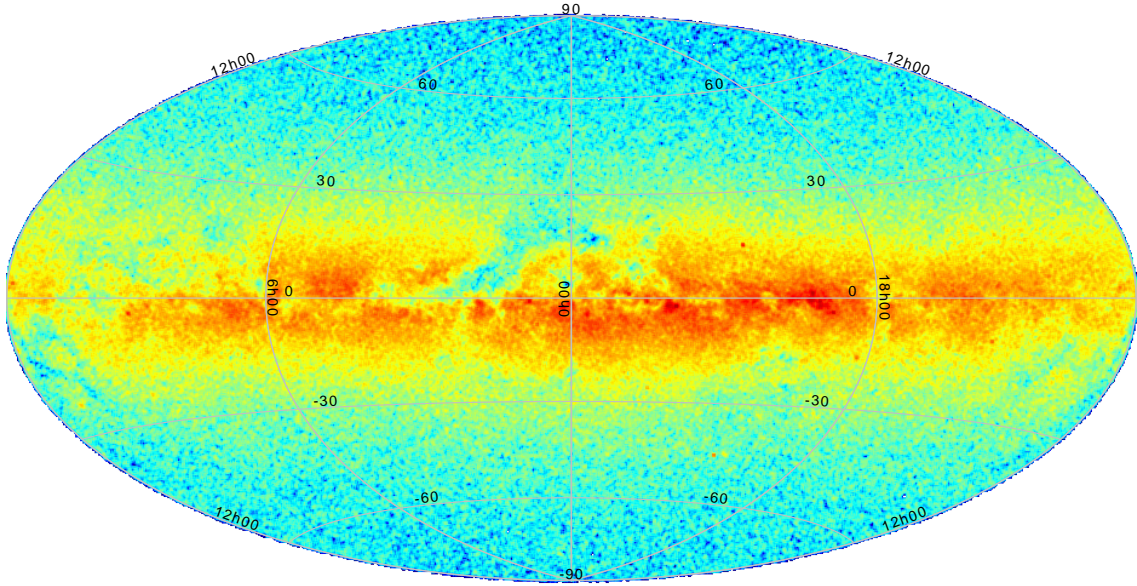
$$\begin{aligned} new_\alpha &= \alpha + \mu_{\alpha\star}t \\ new_\delta &= \delta + \mu_\delta t \end{aligned} \tag{4}$$

Amb les transformacions definides a la classe SkyCoord de la llibreria Astropy. Ara podem calcular  $(new_b, new_l)$  a partir de  $(new_\alpha, new_\delta, \mu_{\alpha\star}, \mu_\delta)$ . Una vegada obtinguda  $(new_l, new_b)$  les puc tornar a propagar amb les seves velocitats inicials  $(\mu_{l\star}, \mu_b)$  que considero constants durant tota la propagació.

M'adono que aquesta aproximació no és vàlida i és el motiu per el qual les estrelles s'acumulen als pols. Si considero les velocitats constants, encara que transformi les coordenades quan s'apropen als pols, propagaré les noves coordenades amb la mateixa velocitat utilitzada abans de transformar-les i les coordenades faran una especie de oscil·lació harmònica al voltant dels pols i no sortiran d'allà.

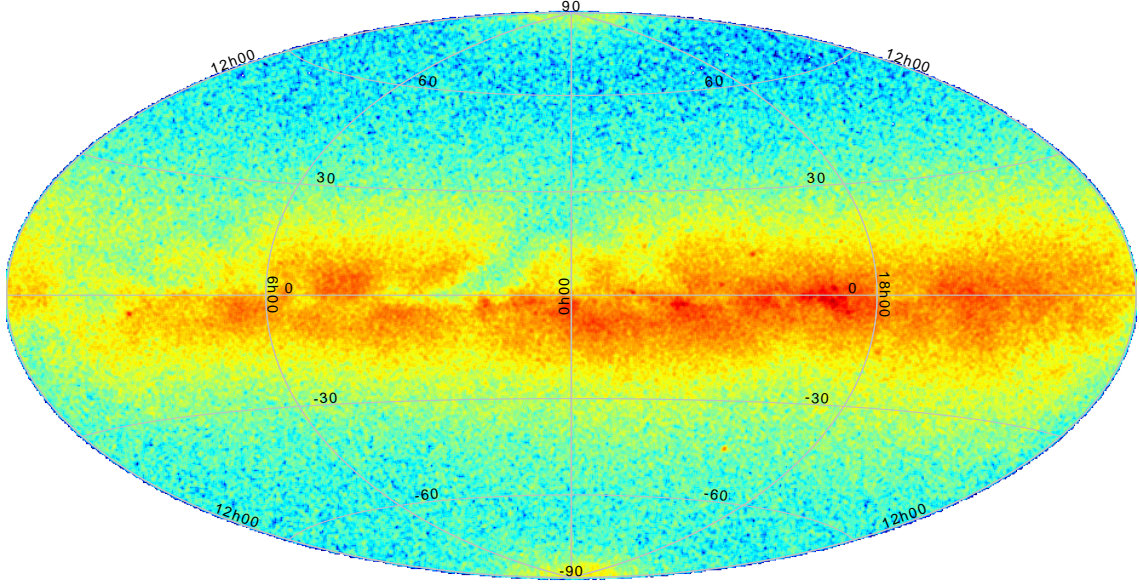
Per solucionar aquest problema, he de millorar l'algoritme incorporant la transformació de les velocitats, també inclosa amb la classe SkyCoord. D'aquesta manera, aprop dels pols les velocitats canviaran i les estrelles seguiran la seva trajectòria sense acumular-se. Així, una vegada tinguem  $(new_l, new_b)$ , les propagarem amb les velocitats transformades  $(\mu'_{l\star}, \mu'_b)$  enlloc de les inicials  $(\mu_{l\star}, \mu_b)$ . L'algoritme millorat és el de la secció B.3.

Per agilitzar l'execució de l'algoritme i poder analitzar els resultats amb més rapidesa, passo a treballar amb una mostra d'un mil·lió d'estrelles ben distribuïdes per la bòveda celeste per obtenir resultats representatius de manera més immediata (Query A.3).



**Figura 8:** Mapa de densitat amb projecció aitoff de la bòveda celeste de  $10^6$  estrelles en coordenades galàctiques.

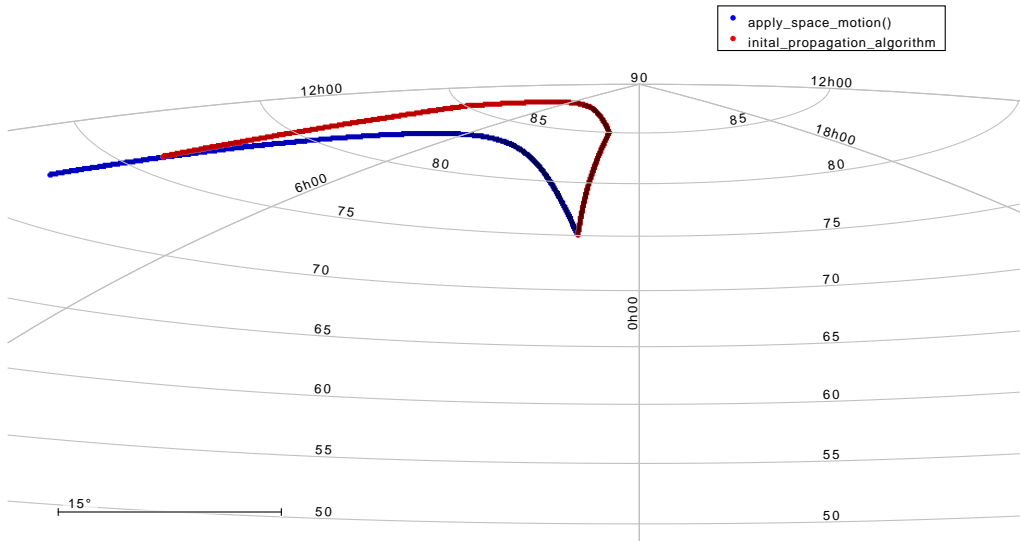




**Figura 9:** Mapa de densitat amb projecció aitoff de la bòveda celeste de  $10^6$  estrelles en coordenades galàctiques amb una propagació de  $t = 2 \cdot 10^6$  anys.

En la Fig.9 es segueix observant la tendència de les estrelles a acumular-se a  $b = \pm 90^\circ$ . L'algoritme de canvi de coordenades té sentit físic però sembla que propagar linealment i només corregint als pols no és una aproximació suficientment bona del moviment de les estrelles. Arribem a la conclusió que s'han de tenir en compte altres factors i anar corregint la trajectòria a cada pas de temps.

Llegint documentació de la llibreria Astropy, trobem dins de l'objecte SkyCoord el mètode *apply\_space\_motion()*. És un mètode que propaga linealment les coordenades celestes incloent correccions relativistes (algoritme de la secció B.4). Per mirar la diferència entre els dos algoritmes, propago una estrella amb els dos algoritmes desde les mateixes condicions inicials (Fig.10).

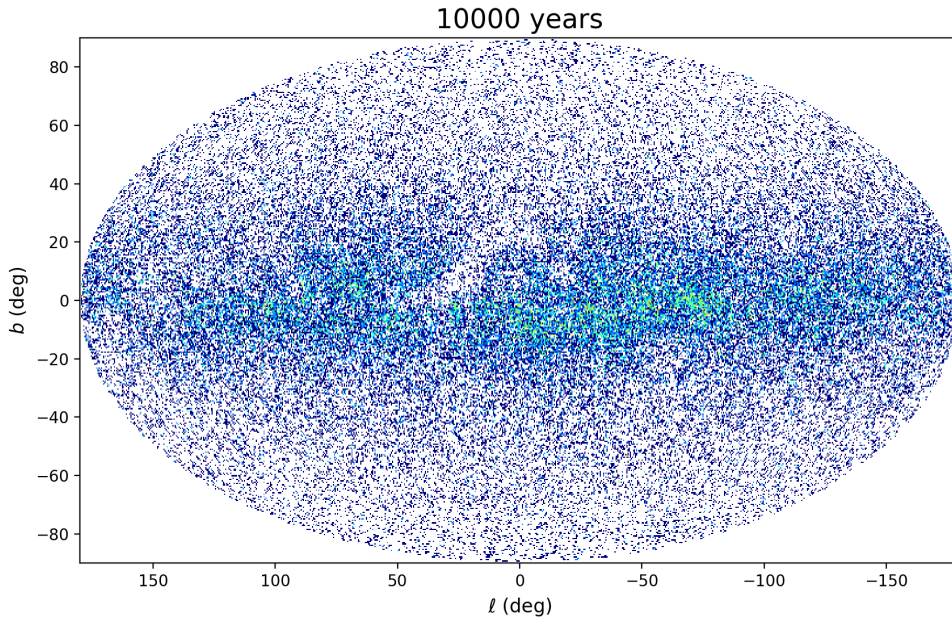


**Figura 10:** Comparació dels dos algoritmes de propagació per una mateixa estrella propagada 2 milions d'anys. En vermell la trajectòria de l'estrella per l'algoritme de la secció B.3. En blau la trajectòria de l'estrella amb l'algoritme de la secció B.4 amb el mètode *apply\_space\_motion()*.

S'observa a la Fig.10 com es propaga l'estrella a prop del pol per cada un dels algoritmes. A la trajectòria de l'estrella en color vermell s'aprecia clarament com l'estrella corregeix la seva trajectòria quan arriba a l'angle  $b = 85^\circ$ .

En canvi, amb el mètode *apply\_space\_motion()* s'observa una trajectòria menys brusca. Té sentit ja que el mètode corregeix les velocitats a cada pas de temps mentre que l'algoritme B.3 només ho fa aprop dels pols.

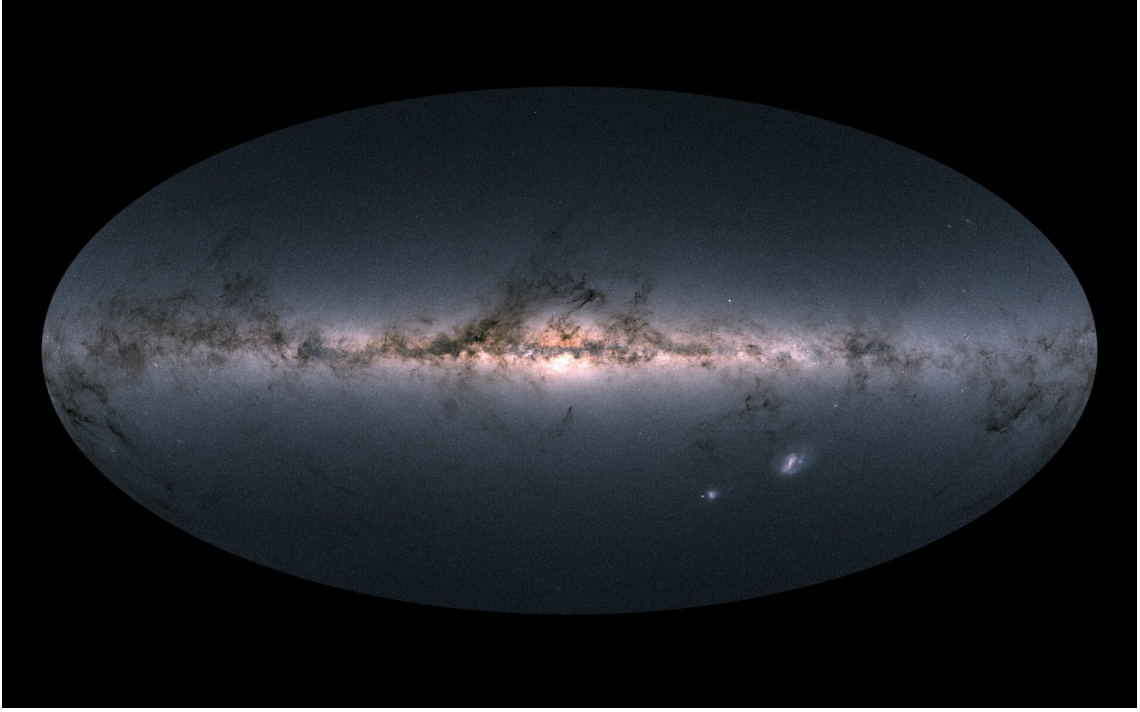
La principal diferència entre algoritmes de propagació és que el mètode *apply\_space\_motion()* és més precís i evita que les estrelles s'acomulin als pols. Per contra, propaga anant estrells per estrells i, gràcies a això, és molt més lent d'executar que l'algoritme de la secció B.3. És per això que per provar el nou algoritme utilitzo  $10^4$  estrelles amb les mateixes característiques que les obtingudes amb la query A.3. Els resultats són físicament correctes ja que les estrelles no s'acomulen enlloc i es propaguen elegantment (Fig.11).



**Figura 11:** Mapa de densitat de  $10^5$  estrelles en coordenades galàctiques amb una propagació de  $t = 10000$  anys.

## 5 Visualització de les dades

Una vegada trobat aquest algorisme de propagació que funciona, s'ha de buscar una altra manera de representar les dades que sigui visualment més maca de veure. L'objectiu és aconseguir una animació amb colors, com a la Fig.12.



**Figura 12:** Representació de la Via Làctia on hem transformat els índexs de color dels cossos celestes a escala RGB.

Per generar aquest tipus d'imatge, hem de dividir la bòveda celeste en una quadrícula on cada cel·la es diu *healpix*. Per obtenir els colors de la imatge, hem de descarregar de l'arxiu de Gaia estrelles amb diferents bandes de color (Query A.4).

Per determinar el color d'un healpix en un instant de temps, mirem quines estrelles hi ha i sumem tot el flux en les bandes BP, RP i G. Aquest flux es pot transformar en el que es coneix com *índex de color*. L'índex de color representa una propietat intrínseca de l'estel i s'obté restant magnituds aparents en diferents bandes de color. Per exemple, l'índex B-V s'obté

$$m_B - m_V = B - V = -2.5 \log_{10} f_B / f_V + cte \quad (5)$$

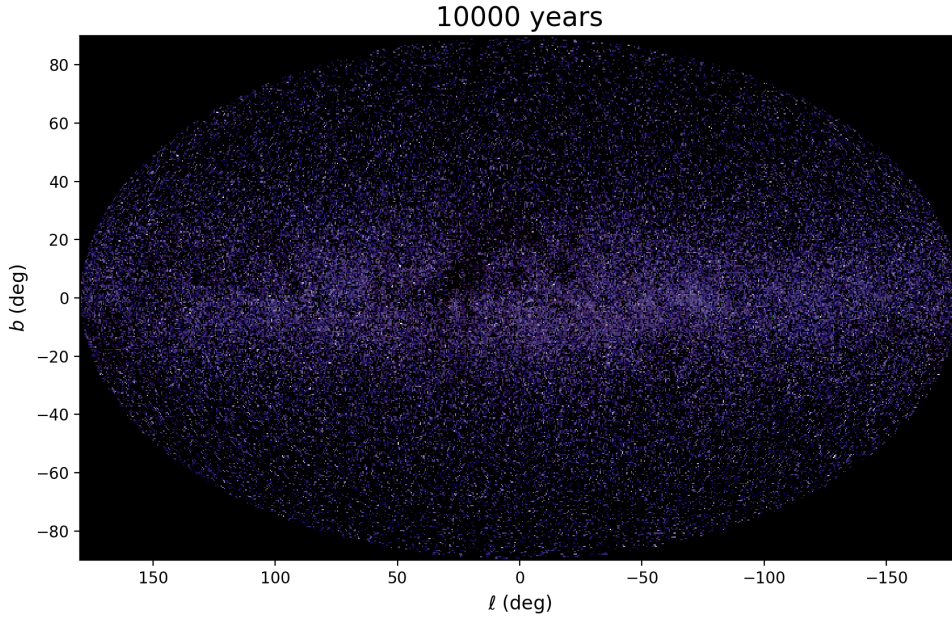
En el nostre cas, els índexs de color en les bandes RP, BP i G que s'obtenen fent les transformacions següents a partir dels fluxos extrets del catàleg de Gaia.

$$\begin{aligned} m_{RP} &= 24.7479 - 2.5 \log_{10} f_{RP} \\ m_G &= 25.6085 - 2.5 \log_{10} f_G \\ m_{BP} &= 25.3385 - 2.5 \log_{10} f_{BP} \end{aligned} \quad (6)$$

Aquestes magnituds donen entre -27 i -3. Per transformar aquesta magnitud a escala RGB (Red-Green-Blue) normalitzem aquests valors. Definim una nova escala on el -3 és el 0 i el -27 és el número 1. Transformant linealment segons aquesta nova escala tenim totes les magnituds per a cada healpix situada entre el 0 i 1. Per últim només queda fer la suma de les tres magnituds per a cada healpix i obtenim un color definit a l'escala RGB que sabem com representar.

Una vegada tenim definit cada healpix amb el seu color corresponent, utilitzem la llibreria Matplotlib [6] per obtenir un dibuix similar a la Fig.12 (codi B.5). Obtenim els següents resultats.





**Figura 13:** Mapa de color de  $10^5$  estrelles en coordenades galàctiques amb una propagació de  $t = 10000$  anys.

## 6 Futurs passos del projecte

Ara que hem trobat un algorisme que funciona i és capaç de generar gràfics visualment bonics, podem mirar de visualitzar altres tipus concrets de dades i obtenir resultats interessants. Les idees que proposem a continuació no s'han pogut dur a terme per cost computacional o falta de temps.

### 6.1 Propagació de la bòveda celeste 1000000 fonts

El catàleg de Gaia EDR3 conté coordenades celestes, velocitats i paral·laxis d'aproximadament 1.468 bilions de fonts, amb una limitació de magnitud  $3 \simeq G \simeq 21$ . El següent pas seria generar un mapa de color similar a la Fig.13 per més de  $10^6$  fonts. L'animació generada tindria més resolució i s'apreciaria amb més detall la propagació de les fonts. No hem pogut dur a terme aquesta idea degut a l'alt cost computacional que suposaria. Per fer una mapa com el descrit, necessitariem un ordinador amb més de 8 CPUS per obtenir resultats amb temps raonables.

### 6.2 Propagació de fonts particulars

Seria interessant mirar fonts de manera particular i amb més detall. Un exemple seria propagar el cúmul globular M4, el més aprop que tenim. Podríem comprovar com les estrelles que el formen es propaguen a velocitats similars al llarg del temps, o potser descobriríem que el cúmul es separarà al llarg dels anys. Altres fonts interessants per investigar podrien el cúmul obert de les Plèiades o propagar el centre galàctic per mirar l'evolució de l'entorn del forat negre Sagittarius A\*.

## 7 Agraïments

Aquest treball ha fet ús de dades de la missió *Gaia* (<https://www.cosmos.esa.int/gaia>) de la Agència Espacial Europea (ESA) processades per Data Processing and Analysis Consortium (DPAC, <https://www.cosmos.esa.int/web/gaia/dpac/consortium>). Financiació per DPAC ha estat proporcionada per institucions nacionals, especialment els països que participen en el *Gaia* Multilateral Agreement. També vull agrair el Marcel, l'Albert i el Jordi per la seva guia que m'han permès gaudir tant d'aquest projecte.



## A Scripts ADQL

### A.1 Velocitat radial diferent de zero i magnitud mitjana aparent en la banda G inferior a 13

```
SELECT dr2_radial_velocity, phot_g_mean_mag, source_id, ra,
       dec, l, b
FROM gaiaedr3.gaia_source
WHERE dr2_radial_velocity IS NOT NULL
AND phot_g_mean_mag < 13
```

### A.2 Estrelles del centre galàctic i error en les velocitats inferior al 5%

```
SELECT dr2_radial_velocity, phot_g_mean_mag, source_id, ra,
       dec, pmra, pmdec, l, b
FROM gaiaedr3.gaia_source
WHERE pmra_error/pmra < 0.05 AND pmdec_error/pmdec < 0.05
AND (l < 40 OR l > 320)
AND b > -30 AND b < 30
AND phot_g_mean_mag < 13
```

### A.3 $10^6$ estrelles ben distribuïdes per la bòveda celeste amb un error en les velocitats inferior al 5%

```
SELECT TOP 1000000 dr2_radial_velocity, phot_g_mean_mag,
                  source_id, ra, dec, pmra, pmdec, l, b, random_index
FROM gaiaedr3.gaia_source
WHERE pmra_error/pmra < 0.05 AND pmdec_error/pmdec < 0.05
AND phot_g_mean_mag < 12
ORDER BY random_index
```

### A.4 Query per generar una imatge amb healpix

```
SELECT TOP 100000 dr2_radial_velocity, phot_g_mean_mag,
                  source_id, ra, dec, pmra, pmdec, l, b, random_index,
                  phot_g_mean_flux, phot_bp_mean_flux, phot_rp_mean_flux
FROM gaiaedr3.gaia_source
WHERE pmra_error/pmra < 0.05 AND pmdec_error/pmdec < 0.05
AND phot_g_mean_mag < 12
ORDER BY random_index
```

## B Scripts Python

### B.1 Càlcul de les velocitats en coordenades galàctiques amb matrius de rotació

```
import pandas as pd
import csv
import numpy as np
import math as m
#Pandas is a useful python library to analyze csv files

def equator_to_galac(row):
```

```

"""Constants used to transform to equatorial coordinates
to galactic coordinates"""
alpha_G = 192.85948#Degree
delta_G = 27.12825 #Degree
l_NGD = 122.93192#Degree
pmra = row['pmra']#mas/yr
pmrdec = row['pmdec']#mas/yr
ra = row['ra']
dec = row['dec']
"""Now define the constants from the matrix of rotation
"""
C_1 = np.sin(np.radians(delta_G))*np.cos(np.radians(dec))
      -np.cos(np.radians(delta_G))*np.sin(np.radians(dec))*
      np.cos(np.radians(ra-alpha_G))
C_2= np.cos(np.radians(delta_G))*np.sin(np.radians(ra-
      alpha_G))
cosb = m.sqrt(C_1**2 + C_2**2)

pm_l = (1.0/cosb)*(C_1*row['pmra']+C_2*row['pmdec'])#mas/
      yr
pm_b = (1.0/cosb)*(-C_2*row['pmra']+C_1*row['pmdec'])#mas
      /yr
pm = (pm_l,pm_b)

return pm

#Read the data from Gaia archive
data = pd.read_csv("gaia.csv")
# Preview the first 5 lines of the loaded data
test_manual = data.head(20)
#Calls the function equator_to_galac and adds two more
      columns to the csv file with the proper motions in
      galactic coordinates.
test_manual[['pm_l','pm_b']] = test_manual.apply(
      equator_to_galac,axis = 1,result_type = "expand")
test_manual.to_csv('new_gaia.csv',index = False)

```

## B.2 Càlcul de les velocitats en coordenades galàctiques amb Astropy

```

import pandas as pd
import numpy as np
from astropy.coordinates import SkyCoord
import astropy.units as u
from multiprocessing import Pool

def equator_to_galac(row):
#This function returns the galactic proper motions
calculated from equatorial positions and proper
motions
coord = SkyCoord(ra = row['ra']*u.degree,dec = row['dec']
      ]*u.degree,pm_ra_cosdec =row['pmra']*u.mas/u.yr,pm_dec
      = row['pmdec']*u.mas/u.yr)
gala = coord.galactic

return (gala.pm_l_cosb.value, gala.pm_b.value)

def apply(df):

```

```

#This function adds two new columns to the file.csv with
the galactic proper motions
df[['pm_l','pm_b']] = df.apply(equator_to_galac,axis = 1,
    result_type = "expand")

return df

def parallelize_dataframe(df,func, n_cores):
#Here is the multiprocessing
#This function divides the object dataframe (df) in
n_cores arrays and does the calculation with each CPU,
in this case
df_split = np.array_split(df,n_cores)
pool = Pool(n_cores)
df = pd.concat(pool.map(func,df_split))
pool.close()
pool.join()
return df

#Reads the file and applies all the calculations, in this
case I have used n_cores = 6 CPU
data = pd.read_csv("path_to_file/file.csv")
test = data
n_cores = 6
new_df = parallelize_dataframe(test,apply,n_cores)
new_df.to_csv('path_to_file/new_file.csv', index=False)

```

### B.3 Algoritme de propagació de coordenades galàctiques

```

import pandas as pd
import numpy as np
from astropy.coordinates import SkyCoord
import astropy.units as u
from multiprocessing import Pool

def check_poles(row,t):
#This function has as parameters row, which refers to a
row of a csv file and t which is the time used to
propagate

#Assigns proper motions in galactics from the csv file
pm_l = row['pm_l']
pm_b = row['pm_b']

#Propagates in galactics
new_l = (row['new_l'] + t*pm_l/(3600*1000))%360
new_b = row['new_b'] + t*pm_b/(3600*1000)

#Controls if the propagated b coordinate is around a pole
and thus shall be propagated in equatorials
if abs(new_b) >= 85:

#define the galactic coordinates object
coord_gal = SkyCoord(l = row['new_l']*u.degree,b = row['
    new_b']*u.degree,pm_l_cosb = row['pm_l']*u.mas/u.yr,

```

```

pm_b = row['pm_b']*u.mas/u.yr,frame='galactic')

#transform galactic coordinates to equatorials
ra = coord_gal.icrs.ra.value
dec = coord_gal.icrs.dec.value
pmra = coord_gal.icrs.pm_ra_cosdec.value
pmdec = coord_gal.icrs.pm_dec.value

#propagate in equatorials coordinates
new_ra = (ra+ t*pmra/(3600*1000))%360
new_dec = dec + t*pmdec/(3600*1000)

#define the equatorial coordinates object
coord_eq = SkyCoord(ra = new_ra*u.degree,dec = new_dec*u.
    degree, pm_ra_cosdec = pmra*u.mas/u.yr,
    pm_dec = pmdec*u.mas/u.yr, frame = 'icrs')

#transform to galactic coordinates
new_b = coord_eq.galactic.b.value
new_l = coord_eq.galactic.l.value
pm_l = coord_eq.galactic.pm_l_cosb.value
pm_b = coord_eq.galactic.pm_b.value

#returns galactic propagated coordinates and corrected
    velocities
return new_l,new_b, pm_l, pm_b

def apply(df):
#rewrites pm_l and pm_b in case it has been corrected
    near the poles
df[['new_l','new_b','pm_l','pm_b']] = df.apply(
    check_poles,axis = 1,result_type = "expand",args=(t,))
return df

def parallelize_dataframe(df,func, n_cores):
#Here is the multiprocessing
#This function divides the object dataframe (df) in
    n_cores arrays and does the calculation with each CPU,
    in this case
df_split = np.array_split(df,n_cores)
pool = Pool(n_cores)
df = pd.concat(pool.map(func,df_split))
pool.close()
pool.join()
return df

#Here begins the propagation: Reads the csv file for
    100000 years, does 91 steps of 10000 years and
    computes the csv file for 1 milion years.
t0 = 1E5#inital time
deltat = 1E4#step
n_cores = 6 #number of CPU's used
data_0 = pd.read_csv("path_to_file/file.csv")
for i in range(1,91):

    t_past = int(t0 + (i-1)*deltat)#time - 1 step
    print(t_past)

```



```

t = int(t0 + i*deltat)#current time

filename = str(t)
print(filename)
new_df = parallelize_dataframe(data_0,apply,n_cores)

if i%10== 0:#saves a file every 10 iterations
new_df.to_csv('path_to_file/'+filename+'.csv', index=
False)
print("Fitxer: "+filename+'.csv fet')
data_0 = new_df# inicialises data0 with the previous file
to calculate the next step

```

#### B.4 Algoritme de propagació amb el mètode *apply\_space\_motion()*

```

#The algorithm is the same as described in section B.3
but the function check_poles(row,t). T
#Note that with this method we need to specify that the
stars propagated are very far away (distance=200 kpc).
Indeed, linear propagation can only be assumed when
stars are at long distances and thus paralax can be
ignored.
def check_poles(row, t):
#Define the coordinate object read from the file
coord = SkyCoord(l=row['l']*u.deg, b=row['b']*u.deg,
pm_l_cosb=row['pm_l']*u.mas/u.yr, pm_b=row['pm_b']*u.
mas/u.yr, distance=200*u.kpc,frame='galactic')

#Apply the motion star by star
coord = coord.apply_space_motion(dt=t*u.yr)
#Calculates and returns both new coordinates and proper
motions
new_b = coord.galactic.b.value
new_l = coord.galactic.l.value
pm_l = coord.galactic.pm_l_cosb.value
pm_b = coord.galactic.pm_b.value
return new_l,new_b, pm_l, pm_b

```

#### B.5 Algoritme per generar un diagrama de densitat i de color amb healpix

```

import healpy
from astropy_healpix import HEALPix
import pandas as pd
import numpy as np
from astropy.coordinates import SkyCoord
import astropy.units as u
from multiprocessing import Pool
import matplotlib.pyplot as plt

def propagate(row,hpx,t):
coord = SkyCoord(l=row['l']*u.deg, b=row['b']*u.deg,
pm_l_cosb=row['pm_l']*u.mas/u.yr,
pm_b=row['pm_b']*u.mas/u.yr, distance=200*u.kpc,frame='
galactic')

```

```

coord = coord.apply_space_motion(dt=t*u.yr)
new_b = coord.galactic.b.value
new_l = coord.galactic.l.value
pm_l = coord.galactic.pm_l_cosb.value
pm_b = coord.galactic.pm_b.value

#Assigns to each pair of coordinates a healpix
h8 = hpx.lonlat_to_healpix(new_l*u.deg,new_b*u.deg)

return new_l,new_b, pm_l, pm_b,h8

def apply(df):
#modified apply function where a healpix8 column is added
to the file
df[['l','b','pm_l','pm_b','healpix8']] = df.apply(
    propagate,axis = 1,result_type = "expand",args=(hpx,t)
)

return df

def parallelize_dataframe(df,func, n_cores):
df_split = np.array_split(df,n_cores)
pool = Pool(n_cores)
df = pd.concat(pool.map(func,df_split))
pool.close()
pool.join()
return df

#Healpix characteristics, more level equals more
resolution
level=7
hpx = HEALPix(nside=4096/2**(12-level), order='nested')
num_hp=int(healpy.nside2npix(nside=4096/2**(12-level)))
area_hp=(4*np.pi/num_hp) #in sr
nside=4096/2**(12-level)

#Here begins the propagation: Reads the csv file for
100000 years, does 91 steps of 10000 years and
computes the csv file for 1 milion years.
t0 = 1E5
deltat = 1E4
n_cores = 6

data_0 = pd.read_csv("path_to_file/100000.csv")

for i in range(1,91):

    t_past = int(t0 + (i-1)*deltat)
    print(t_past)
    t = t0 + i*deltat

    filename = str(int(t))
    print(filename)
    new_df = parallelize_dataframe(data_0,apply,n_cores)

```

```

#HEALPIX BEGINS

healpixRVals = dict()
healpixGVals = dict()
healpixBVals = dict()
healpixCount = dict()

for index, row in new_df.iterrows():

    if index % 10000 == 0:
        print(str(index) + " of " + str(len(new_df)))

    healpix = row['healpix8']

    if healpix in healpixRVals:
        healpixRVals[healpix] += row['phot_rp_mean_flux']
    else:
        healpixRVals[healpix] = row['phot_rp_mean_flux']

    if healpix in healpixGVals:
        healpixGVals[healpix] += row['phot_g_mean_flux']
    else:
        healpixGVals[healpix] = row['phot_g_mean_flux']

    if healpix in healpixBVals:
        healpixBVals[healpix] += row['phot_bp_mean_flux']
    else:
        healpixBVals[healpix] = row['phot_bp_mean_flux']

    if healpix in healpixCount:
        healpixCount[healpix] += 1
    else:
        healpixCount[healpix] = 1

    r_list = []
    b_list = []
    g_list = []
    count_list = []

    for healpix_num in range(num_hp):

        if healpix_num in healpixRVals and not np.isnan(
            healpixRVals[healpix_num]):
            r_val = healpixRVals[healpix_num]
        else:
            r_val = 0

        if healpix_num in healpixGVals and not np.isnan(
            healpixGVals[healpix_num]):
            g_val = healpixGVals[healpix_num]
        else:
            g_val = 0

        if healpix_num in healpixBVals and not np.isnan(
            healpixBVals[healpix_num]):
            b_val = healpixBVals[healpix_num]

```

```

else:
    b_val = 0

if healpix_num in healpixCount and not np.isnan(
    healpixCount[healpix_num]):
    count_val = healpixCount[healpix_num]
else:
    count_val = 0

r_list.append(r_val)
g_list.append(g_val)
b_list.append(b_val)
count_list.append(count_val)

fonts=12

fig = plt.figure(figsize=(8,5))

ax = healpy.projaxes.HpxMollweideAxes(fig
    , [0.1,0.1,0.9,0.9], rot=(0,0.0,0.0), coord=['G'])

arrR=ax.projmap(np.array(r_list),nest=True,coord=['G'])
arrG=ax.projmap(np.array(g_list),nest=True,coord=['G'])
arrB=ax.projmap(np.array(b_list),nest=True,coord=['G'])
arrCount=ax.projmap(np.array(count_list),nest=True,coord
    =['G'])

#Transformations explained in equation (6)
arrR = 24.7479-2.5*np.log(arrR)
arrG = 25.6085-2.5*np.log(arrG)
arrB = 25.3385-2.5*np.log(arrB)

#Reescalte to obtain a value of magnitude between 0 and 1
(RGB scale)
arrR = -(arrR-np.min(arrR)*np.ones(np.shape(arrR)))/(np.
    max(arrR)-np.min(arrR))+np.ones(np.shape(arrR))
arrG = -(arrG-np.min(arrG)*np.ones(np.shape(arrG)))/(np.
    max(arrG)-np.min(arrG))+np.ones(np.shape(arrR))
arrB = -(arrB-np.min(arrB)*np.ones(np.shape(arrB)))/(np.
    max(arrB)-np.min(arrB))+np.ones(np.shape(arrR))

rgbArr = np.stack((arrR,arrG,arrB), axis=2, out=None)

## Color Map

plt.figure(figsize = (10,6))
fig = plt.imshow(rgbArr,extent=(180,-180,90,-90), aspect=
    'auto')

ax = plt.gca()
ax.set_title(str(int(t))+ ' years',fontsize=fonts+5)
ax.invert_yaxis()
ax.set_xlabel('$ \ell$ (deg)',fontsize=fonts)
ax.set_ylabel('$ b$ (deg)',fontsize=fonts)

```



```

plt.savefig('path_to_file/healpix_color_hp7'+str(int(t))+
            '.png',format='png',dpi=200)
plt.close()

## Density Map
plt.figure(figsize = (10,6))
fig = plt.imshow(np.log(arrCount),'jet',extent
                 =(180,-180,90,-90), aspect='auto')
ax = plt.gca()
ax.set_title(str(int(t))+ ' years' ,fontsize=fonsts+5)
ax.invert_yaxis()
ax.set_xlabel('$ \ell$ (deg)',fontsize=fonsts)
ax.set_ylabel('$b$ (deg)',fontsize=fonsts)

plt.savefig('path_to_file/healpix_count_hp7'+str(int(t))+
            '.png',format='png',dpi=200)
plt.close()

#Obtain final csv
new_df.to_csv('path_to_file/'+filename+'.csv', index=
             False)
print("Fitxer: "+filename+'.csv fet')

data_0 = new_df

```

## Referències

- [1] European Space Agency (ESA). *Gaia Archive*. <https://gea.esac.esa.int/archive/>.
- [2] European Space Agency (ESA). *Gaia Mission*. <https://sci.esa.int/web/gaia>.
- [3] Astropy Collaboration et al. “Astropy: A community Python package for astronomy”. A: 558, A33 (oct. de 2013), A33. DOI: [10.1051/0004-6361/201322068](https://doi.org/10.1051/0004-6361/201322068). arXiv: [1307.6212](https://arxiv.org/abs/1307.6212) [astro-ph.IM].
- [4] Astropy Collaboration et al. “The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package”. A: 156.3, 123 (set. de 2018), pàg. 123. DOI: [10.3847/1538-3881/aabc4f](https://doi.org/10.3847/1538-3881/aabc4f). arXiv: [1801.02634](https://arxiv.org/abs/1801.02634) [astro-ph.IM].
- [5] A. G. A. Brown et al. “Gaia Early Data Release 3”. A: *Astronomy Astrophysics* 650 (juny de 2021), pàg. C3. ISSN: 1432-0746. DOI: [10.1051/0004-6361/202039657e](https://doi.org/10.1051/0004-6361/202039657e). URL: <http://dx.doi.org/10.1051/0004-6361/202039657e>.
- [6] J. D. Hunter. “Matplotlib: A 2D graphics environment”. A: *Computing in Science & Engineering* 9.3 (2007), pàg. 90-95. DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- [7] M. M. McKerns, L Strand, T. Sullivan, A. Fang i M. A. Aivazis. “Building a framework for predictive science”. A: Proceedings of the 10th Python in Science Conference (2011). URL: <http://arxiv.org/pdf/1202.1056>.
- [8] Pedro Osuna et al. “IVOA Astronomical Data Query Language Version 2.00”. A: (). Ed. de Pedro Osuna i Iñaki Editors Ortiz. DOI: [10.5479/ads/bib/2008ivoa.spec.1030o](https://doi.org/10.5479/ads/bib/2008ivoa.spec.1030o). URL: <http://dx.doi.org/10.5479/ADS/bib/2008ivoa.spec.1030o>.
- [9] Radosław Poleski. “Transformation of the equatorial proper motion to the Galactic system”. A: (2018). arXiv: [1306.2945](https://arxiv.org/abs/1306.2945) [astro-ph.IM].
- [10] M. B. Taylor. “TOPCAT & STIL: Starlink Table/VOTable Processing Software”. A: *Astronomical Society of the Pacific Conference Series* 347 (des. de 2005). Ed. de P. Shopbell, M. Britton i R. Ebert, pàg. 29.