DOCUMENTATION TECHNIQUE DU CODE

Hein Léo

I. CREATION DU CATALOGUE

II. FONCTIONS D'AFFICHAGE DES DONNEES DES PRINCIPAUX PARAMETRES

III. FONCTIONS DE FILTRAGE DES DONNEES

IV. FONCTIONS D'AFFICHAGES SPATIAUX DES VITESSES ET MOUVEMENTS PROPRES

V. FONCTION D'AFFICHAGE DES CARTES

VI. FONCTION POUR LA COMPARAISON QUANTITATIVE

I. CREATION DU CATALOGUE

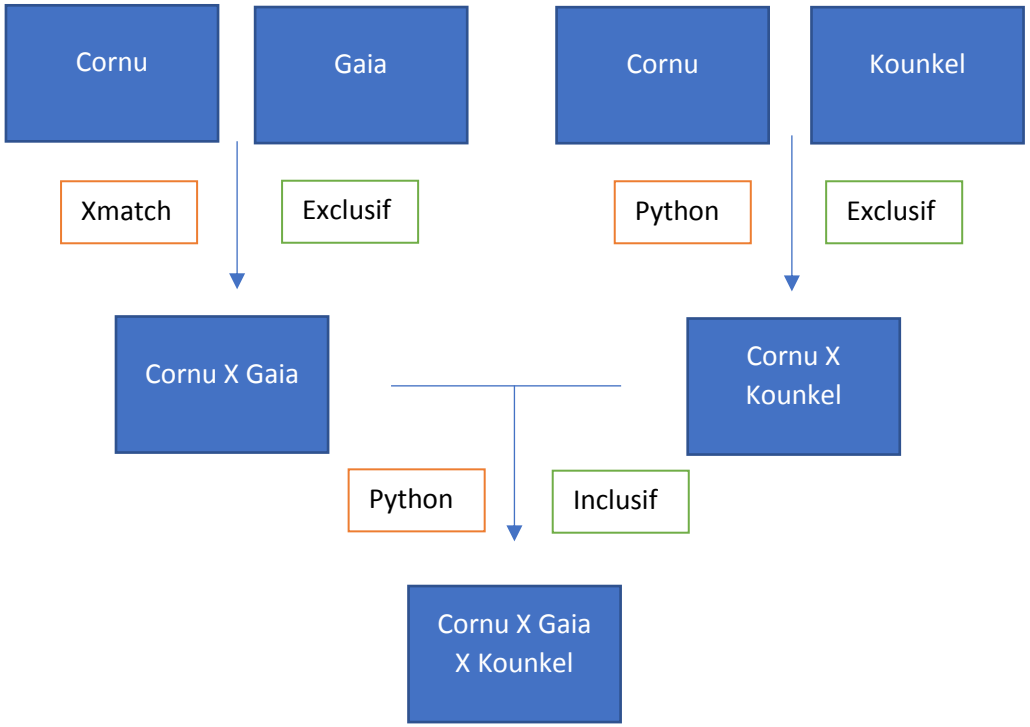
Fonction *Catalog* : Cette fonction permet de créer un catalogue de YSO personnalisé selon différents paramètres et basé sur les résultats de l'article de David Cornu paru en 2021. Le catalogue a pour but d'obtenir les données issues des relevés Apogée et Gaia pour les objets du catalogue de Cornu.

Pour ce faire, le catalogue de Cornu va être crossmatché en parallèle avec le catalogue de Gaia EDR3 et avec un des deux catalogues Apogée à disposition, à savoir le catalogue de Kounkel et celui de Jonsson (Apogée DR2). Dans les deux catalogues résultants, il reste uniquement les objets communs entre les deux catalogues de départ, pour chacun des deux crossmatchs (crossmatch exclusif). Ensuite ces deux catalogues intermédiaires vont être crossmatchés ensemble afin d'obtenir les objets présents dans les 3 catalogues de départ, ainsi que leurs données associées, dans un catalogue appelé final. Toutefois, le catalogue final est complété par les objets des deux catalogues intermédiaires restants (crossmatch inclusif). C'est-à-dire que le catalogue final comprendra les objets communs aux 3 catalogues suivants : Cornu, Gaia, Apogée ; mais aussi ceux uniquement en commun entre Cornu et Apogée, puis entre Cornu et Gaia. Il est aussi possible d'utiliser les quatre catalogues de départ et dans ce cas, le catalogue de Cornu va être crossmatché en parallèle avec les deux catalogues d'Apogée, puis ces deux résultats vont être crossmatchés pour avoir un unique catalogue intermédiaire Cornu-Apogée. Par ailleurs, certains des crossmatchs sont réalisés via l'outil Xmatch du CDS et sont importés sur Python alors que d'autres sont directement réalisé sur Python (à un arc second près) pour avoir plus de liberté sur les paramètres à choisir (certains ne sont pas accessibles via Xmatch). En effet, les catalogue Cornu X Gaia et Cornu X Johnsson sont réalisés via Xmatch, tous les autres sont réalisés via Python. Un schéma récapitulatif des différents cas de figure est proposé ci -dessous. Pour chacun des catalogues de départ, il faut définir une liste des paramètres que l'on souhaite avoir dans notre catalogue final. Il est aussi possible de limiter la localisation des objets et l'on peut définir la probabilité minimale acceptée pour définir un YSO de classe I ou de classe II, sinon cela signifie que l'on utilise la classification faite par Cornu. Par ailleurs, il est possible de ne pas sélectionner les étoiles doubles, les étoiles qui ont eu des changements de nomenclature ou les deux (entre Gaia DR2 et Gaia DR3). Si les étoiles doubles sont sélectionnées, elles sont divisées en 2 objets distincts dans le catalogue avec les paramètres justement associés pour chacune. Le catalogue final est présenté sous forme de fichier texte sous le nom *res_cornuXapoXgaia* et peut être mis à jour dans le dossier de travail pour chaque appel de la fonction *Catalog* grâce à une ligne de code supplémentaire. Les séparateurs sont des tabulations. Certains paramètres sont ajoutés automatiquement même s'ils ne sont pas présents dans les listes de paramètres indiquées, notamment les coordonnées RA DEC, la région ou encore la distance probable si l'on choisit d'y inclure les données sur les parallaxes.

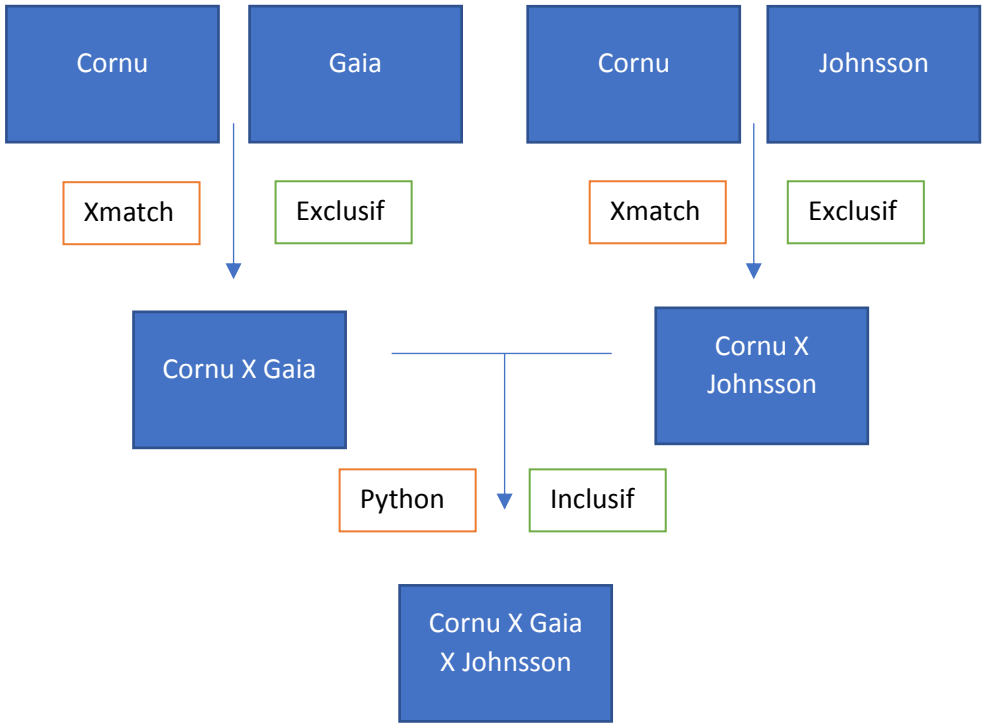
Extrait d'un catalogue final :

pmdec	DEC	parallax	source_id	pmdec_error	parallax_error	RA	pmra	P(CII)	P(CI)	Pred	pmra_error	ProbDist	MinDist	MaxDist	2MASS	Gaia	RVmean	e_RVmean	Av	e_Av	Teff	e_Teff	logg	e_logg	Region	er
	-1.6339374		3216657314719233920		85.3766562		0.9999230	0.0000030	1			2M05413041-0138025		28.2727000	0.5816810		3819.3500000		42.6185000	4.8516200	0.2097110		L1630S			
	-1.9039090				85.4133072		0.9687200	0.0146080	1		2M05413919-0154140		27.2492000	0.2692860			4325.9800000		12.8737000	4.0651200	0.0490253		L1630S			
	-1.8247870				85.3249893		0.9881880	0.0013320	1		2M05411799-0149292		27.6048000	0.2781170			4138.1000000		15.0249000	4.2545300	0.0520230		L1630S			
0.4360000	-2.1254728	2.5495000	3216531283200675328	0.0800000	0.0998000		85.3759977			1.2820000	0.9997510	0.0000960	1	0.0930000		392.2337713	377.4581965	408.2132506							L1630S	
-1.5220000	-2.0894170	2.8109000	3216534581735569024	0.1170000	0.1748000		85.1559877			2.5100000	0.9998040	0.0001920	1	0.2090000		355.7579423	334.9298322	379.3482797							L1630S	
-1.6120000	-2.0467608	2.0053000	3216625871264375936	0.1940000	0.2411000		85.4372773			0.0690000	0.9999550	0.0000060	1	0.2040000		498.6785020	445.1566952	566.8291577							L1630S	
0.1620000	-2.0314243	2.8238000	3216626249221501056	0.3460000	0.4414000		85.4830756			0.1910000	0.9997130	0.0000880	1	0.3250000		354.1327289	306.2599534	419.7447952							L1630S	
	-2.0231517		3216626180503591936		85.4394853		0.9847690	0.0063300	1							L1630S										
	-1.9964849		3216626695899705344		85.4577292		0.0265900	0.9631230	0							L1630S										
-1.3510000	-1.9957537	3.0056000	3216626665833224576	0.3020000	0.3459000		85.4831846			0.1740000	0.9722010	0.0193750	1	0.2820000		332.7122704	298.3738624	375.9822536							L1630S	
-0.5970000	-1.9832462	2.6007000	3216627662265231616	0.2160000	0.2742000		85.3929184			-0.4540000	0.9792680	0.0106670	1	0.2390000		384.5118622	347.8381857	429.8302171							L1630S	
0.5320000	-1.9831212	2.0544000	3216627421747057664	0.8350000	1.0043000		85.4331450			-1.8040000	0.9860420	0.0069950	1	0.8230000		486.7601246	326.9362801	952.2902581							L1630S	
	-1.9815647		3216628212021030400		85.4864605		0.9956280	0.0019800	1							L1630S										
-1.1620000	-1.9802173	2.4915000	3216627623611063936	0.2630000	0.3529000		85.4243681			-0.1570000	0.9998140	0.0000600	1	0.2730000		401.3646398	351.5679932	467.5956233							L1630S	
-0.9270000	-1.9677820	3.1241000	3216627554892430976	0.3270000	0.4068000		85.4609999			0.0490000	0.9744640	0.0024440	1	0.3230000		320.0921865	283.2139115	368.0123652							L1630S	
-2.7870000	-1.9590274	1.9831000	3216628448246372992	0.8340000	1.0320000		85.5724171			-1.4800000	0.9999610	0.0000380	1	0.7730000		504.2610055	331.6639581	1051.4141520							L1630S	
-0.9620000	-1.9591463	2.9985000	3216627761050041856	0.2490000	0.2997000		85.4400980			0.1660000	0.9947480	0.0025620	1	0.2480000		333.5000834	303.1956825	370.5350526							L1630S	

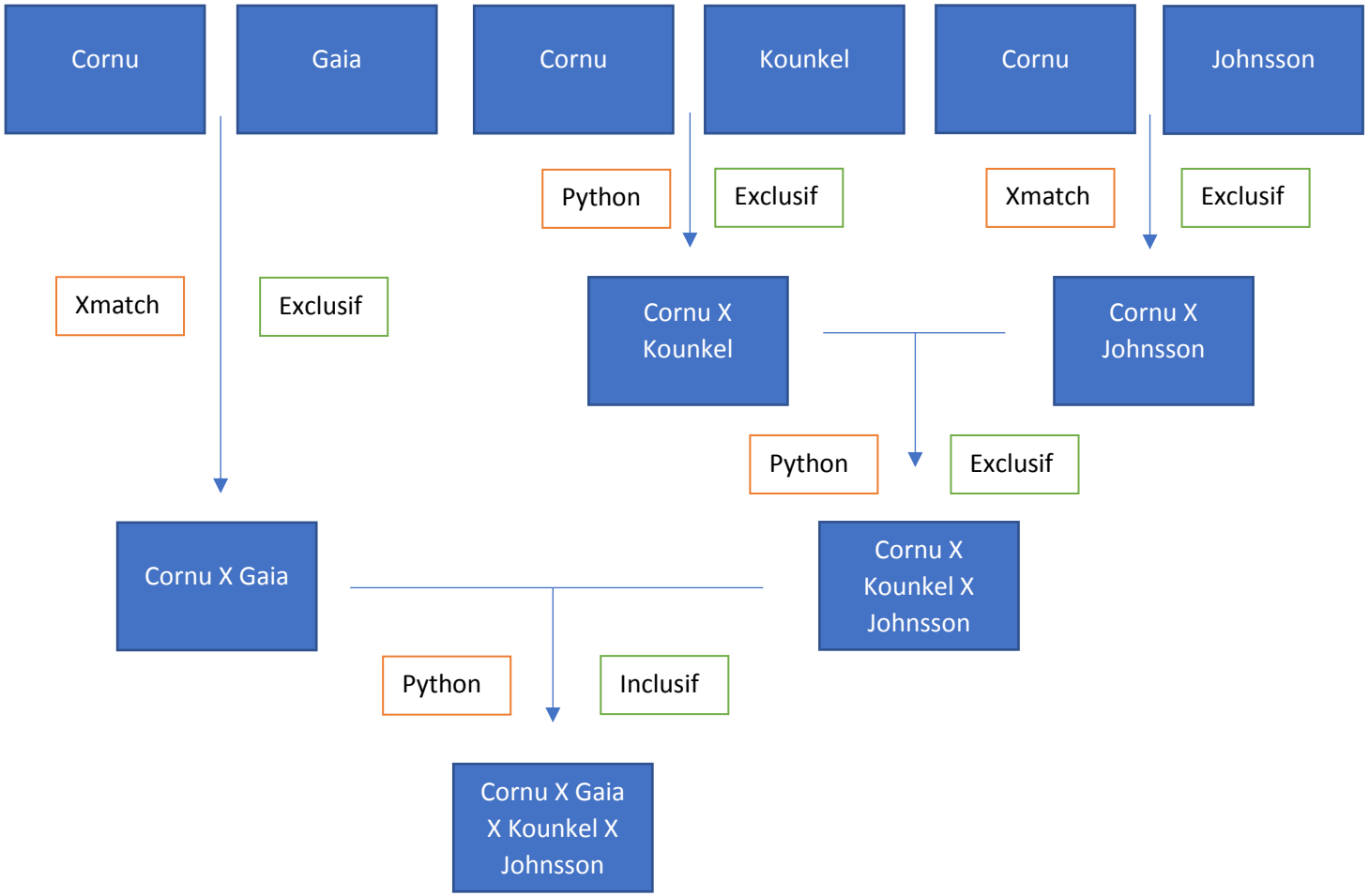
Cas de figure 1 : Crossmatch de Cornu avec Gaia et Kounkel.



Cas de figure 2 : Crossmatch de Cornu avec Gaia et Johnsson.



Cas de figure 3 : Crossmatch de Cornu avec Gaia, Kounkel et Johnsson.



Paramètres d'entrée :

Param_Cornu	Cf liste des paramètres Cornu possibles	Liste des paramètres de Cornu voulus
Param_Gaia	Cf liste des paramètres Gaia possibles	Liste des paramètres de Gaia EDR3 voulus
Param_Johnsson	Cf liste des paramètres Johnsson possibles	Liste des paramètres de Johnsson voulus
Param_Kounkel	Cf liste des paramètres Kounkel possibles	Liste des paramètres de Kounkel voulus
ProbThresh	Entier entre 0 et 1 inclus.	Seuil de probabilité minimal accepté pour qu'un objet soit considéré un YSO de classe I ou de classe II. 0 pour utiliser la classification de l'article.
Loc	[0 à 360, 0 à 360, -90 à 90, -90 à 90]	Liste de quatre coordonnées galactiques représentant la localisation voulue Dans l'ordre : [RAmin, RAmax, DECmin, DECmax] en degrés
RemoveType	None, 'Double', 'NomChange', 'Both'	Type d'étoile que l'on ne veut pas prendre en compte dans notre catalogue
Apo	'Kounkel', 'Johnsson', 'Both'	Catalogue Apogée sur lequel on veut se baser

Exemple de paramètres d'entrée et appel de la fonction :

```
param_cornu = ['Pred', 'P(CI)', 'P(CII)']
param_johnsson = ['ID', 'HRV', 'errHRV']
param_gaia = ['source_id', 'parallax', 'parallax_error', 'pmra', 'pmdec', 'pmra_error', 'pmdec_error']
param_kounkel = ['2MASS', 'Gaia', 'RVmean', 'e_RVmean', 'Av', 'e_Av']
ProbThresh = 0.95
Loc = [85, 86, -3, -0.8]
Remove = 'Double'
Apogee = 'Kounkel'

Tab = catalog(param_cornu, param_gaia, param_johnsson, param_kounkel, ProbThresh, Loc, Remove, Apogee)
Tab.to_csv('res_cornuXapoXgaia.txt',  header=True,  decimal='.',  index=False,  sep='\t',  float_format='%.7f')
```

Dans cet exemple, le catalogue sera créé à partir de Gaia et Kounkel, avec des objets situés dans L1630S uniquement, ayant une probabilité de 0.95 ou plus. Les étoiles doubles ne seront pas prises en compte et les paramètres du catalogue final seront la concaténation des paramètres des listes *param_gaia, param_cornu et param_kounkel*, ainsi que les ajouts automatiques comme la position par exemple.

II. FONCTIONS D'AFFICHAGE DES DONNEES DES PRINCIPAUX PARAMETRES

Fonction *distance_graph* : affiche les distances probables des YSO du catalogue donné en entrée dans l'ordre croissant ainsi que les barres d'erreurs associées, basées sur les données de parallaxes.

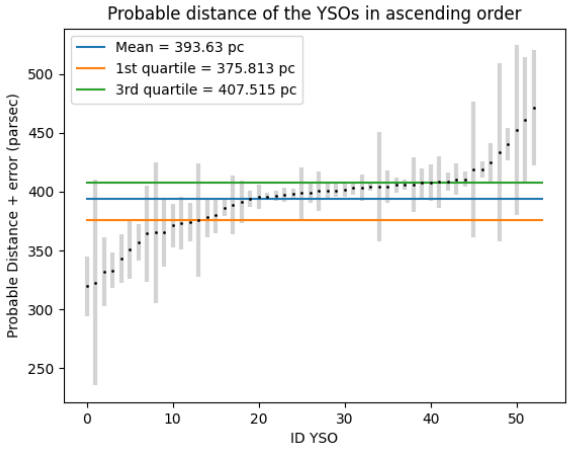
df	Dataframe	Tableau de données à utiliser

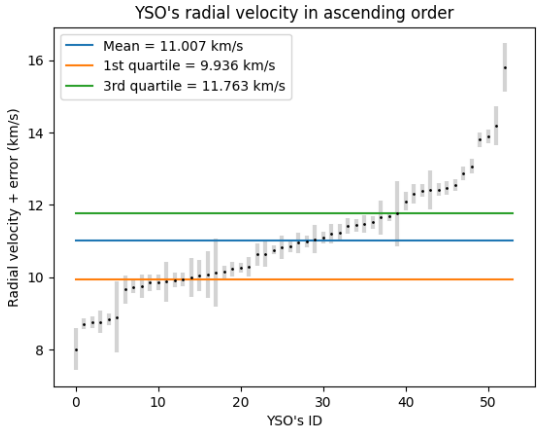
```
distance_graph(df=Tab)
```

Fonction *velocity_graph* : affiche les vitesses stipulées des YSO du catalogue donné en entrée dans l'ordre croissant ainsi que les barres d'erreurs associées.

df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom du paramètre vitesse radiale à afficher
velerr	String : 'e_RVmean', 'errHRV'…	Nom de l'erreur du paramètre vitesse radiale à afficher
type	'Normal', 'Relative'	Type d'affichage, vitesse normale ou vitesse relative au mouvement radial d'ensemble

```
velocity_graph(df=Tab, velocity='RVmean', velerr='e_RVmean', type=0)
```





Fonction *pm_graph* : affiche les mouvements propres selon RA et DEC dans l'ordre croissant ainsi que les barres d'erreurs associées des objets du catalogue donné en entrée, sous forme de deux graphiques distincts.

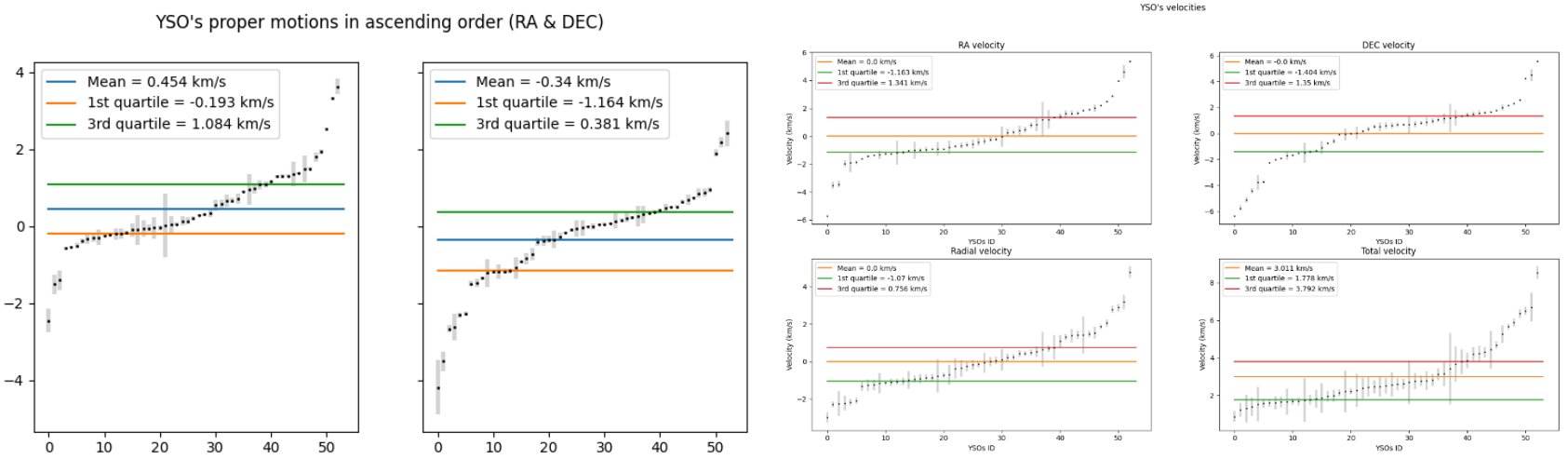
df	Dataframe	Tableau de données à utiliser
abs	0 ou 1	Affichage en valeur normale (0) ou absolue (1)

```
pm_graph(df=Tab,  abs=0)
```

Fonction *velocity_graphs* : affiche respectivement selon quatre graphiques distincts, les vitesses selon l'ascension droite, les vitesses selon la déclinaison, les vitesses radiales et les vitesses totales des YSO du catalogue courant, dans l'ordre croissant. Une barre horizontale représentant la moyenne de chaque paramètre sur le catalogue courant est aussi affichée sur le graphique associé.

df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom du paramètre vitesse à afficher
velerr	String : 'e_RVmean', 'errHRV'…	Nom de l'erreur du paramètre vitesse à afficher
type	String : 'Normal', 'Relative'	Type d'affichage, vitesses normales ou vitesses relatives aux mouvements d'ensemble respectifs

```
velocity_graphs(df=Tab, velocity='RVmean', velerr='e_RVmean', type='Relative')
```



III. FONCTIONS DE FILTRAGE DES DONNEES

Fonction *position_filter* : permet de filtrer les données du catalogue donné en entrée selon la distance des objets. Le filtrage se fait selon un intervalle donnant une distance minimale à dépasser et une distance maximale à ne pas dépasser. On peut aussi ajouter ou non un filtrage sur l'erreur de associée.

df	Dataframe	Tableau de données à utiliser
MinDist	Float	Distance minimale acceptée
MaxDist	Float	Distance maximale acceptée
error	0 ou float	Filtre sur l'erreur ou non

```
Tab = position_filter(df=Tab, MinDist=300, MaxDist=490, error=100)
```

Fonction *velocity_filter* : permet de filtrer les données du catalogue donné en entrée selon leurs vitesses radiales. Selon le même principe.

df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom de la vitesse
velerr	String : 'e_RVmean', 'errHRV'…	Nom de l'erreur de la vitesse
vmin	Float	Vitesse minimale acceptée
vmax	Float	Vitesse maximale acceptée
error	0 ou float	Erreur acceptée

```
Tab = velocity_filter(df=Tab, velocity='RVmean', velerr='e_RVmean', vmin=7, vmax=20, error=2)
```

Fonction *pm_filter* : permet de filtrer les données du catalogue donné en entré selon leurs mouvements propres. Selon le même principe mais avec possibilité de passer d'une erreur absolue à une erreur relative, principalement pour écarter les erreurs d'orientation pour des valeurs proches de 0.

df	Dataframe	Tableau de données à utiliser
pmramin	Float	Nom de la vitesse
pmramax	Float	Nom de l'erreur de la vitesse
pmdecmin	Float	Vitesse minimale acceptée
pmdecmax	Float	Vitesse maximale acceptée
type	String : 'uniform' ou 'relative'	Erreur relative ou non
error	0 ou float	Erreur acceptée

```
Tab = pm_filter(df=Tab,  pmramin=-10,  pmramax=10,  pmdecmin=-10,  pmdecmax=10,  type='Relative',  error=0.9)
```

IV. FONCTIONS D'AFFICHAGES SPATIAUX DES VITESSES ET MOUVEMENTS PROPRES

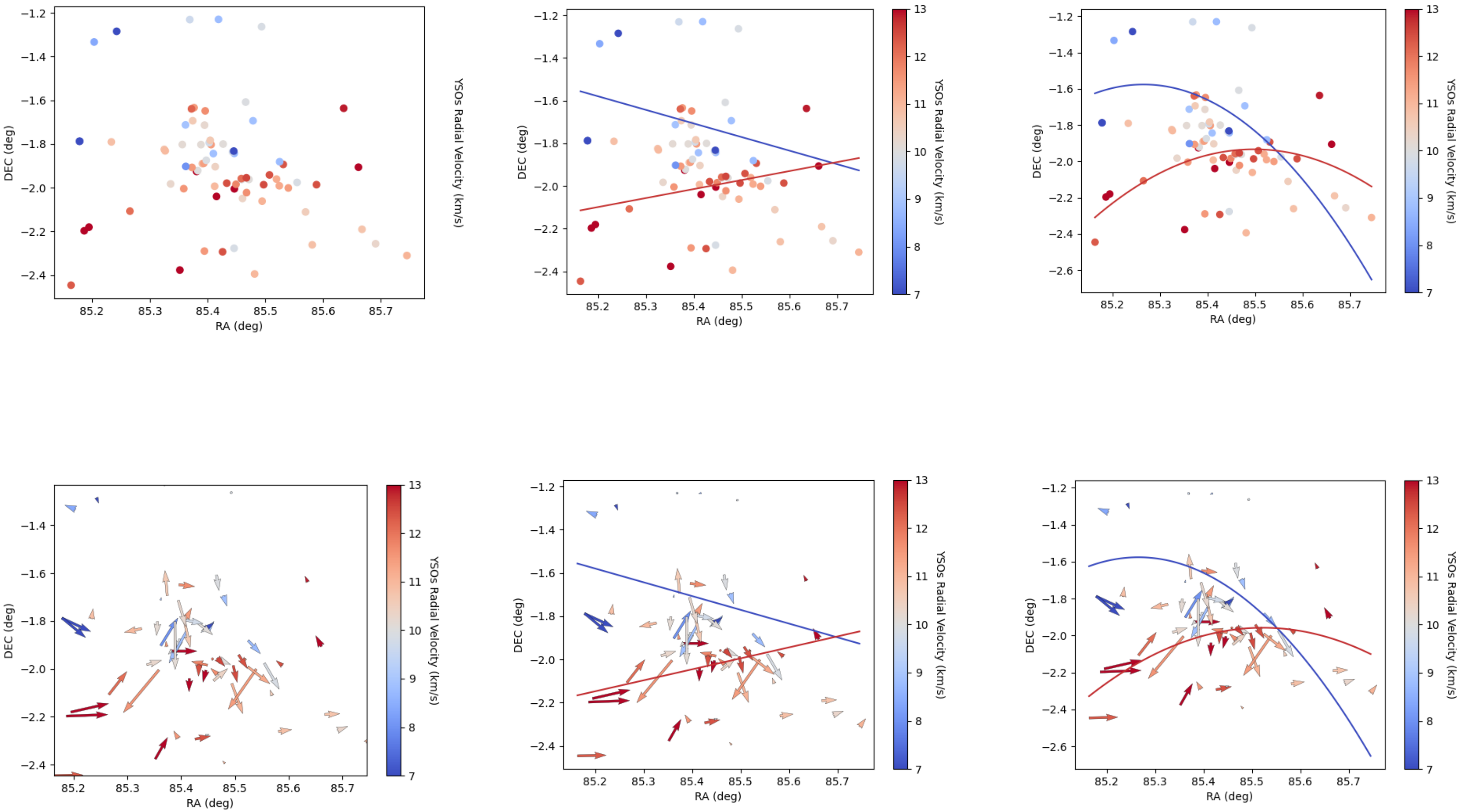
Fonction *velocity_2D* : permet d'afficher dans l'espace en deux dimensions (vue du ciel), les YSO conformément à leur position réelle, avec un code couleur correspondant à leurs vitesses radiales, sous forme de nuage de points. Il est possible de faire le même graphe avec en plus les données des mouvements propres de chaque YSO sous forme de nuage de vecteurs. Il est aussi possible d'ajouter au graphe deux lignes de tendance polynomiale de degré un ou deux pour essayer d'observer des positions différentes en fonction des vitesses radiales. Le principe est de créer deux échantillons de vitesses à partir du catalogue entier, un échantillon de vitesses 'hautes' défini par une vitesse faisant office de borne inférieure et un échantillon de vitesses 'basses', défini par une vitesse faisant office de borne supérieure. Par exemple avec *vinf* = 9 et *vsup* = 11, l'échantillon de vitesses 'basses' contiendra tous les objets ayant une vitesse inférieure à 9 km/s et celui de vitesses 'hautes' ceux ayant une vitesse supérieure à 11 km/s.

df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom de la vitesse
velerr	String : 'e_RVmean', 'errHRV'…	Nom de l'erreur de vitesse
pm	0 ou 1	Nuage de points (0) ou de vecteurs (1)
fit	0 ou 1	Avec un fitting (1) ou non (0)
order	1 ou 2	Ordre du fitting
vinf	Float	Vitesse max pour l'échantillon bas
vsup	Float	Vitesse min pour l'échantillon haut
vmin	Float	Vitesse min de l'échelle de couleur
vmax	Float	Vitesse max de l'échelle de couleur

Liste des appels pour les 6 figures types ci-dessous :

```
Velocity, VelErr = 'RVmean', 'e_RVmean'

velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=0, order=0, vinf=0, vsup=0, vmin=7, vmax=13)
velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=1, order=1, vinf=9, vsup=11, vmin=7, vmax=13)
velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=1, order=2, vinf=9, vsup=11, vmin=7, vmax=13)
velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=0, order=0, vinf=0, vsup=0, vmin=7, vmax=13)
velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=1, order=1, vinf=9, vsup=11, vmin=7, vmax=13)
velocity_2D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=1, order=2, vinf=9, vsup=11, vmin=7, vmax=13)
```


DEC (deg)
RA (deg)
YSOs Radial Velocity (km/s)

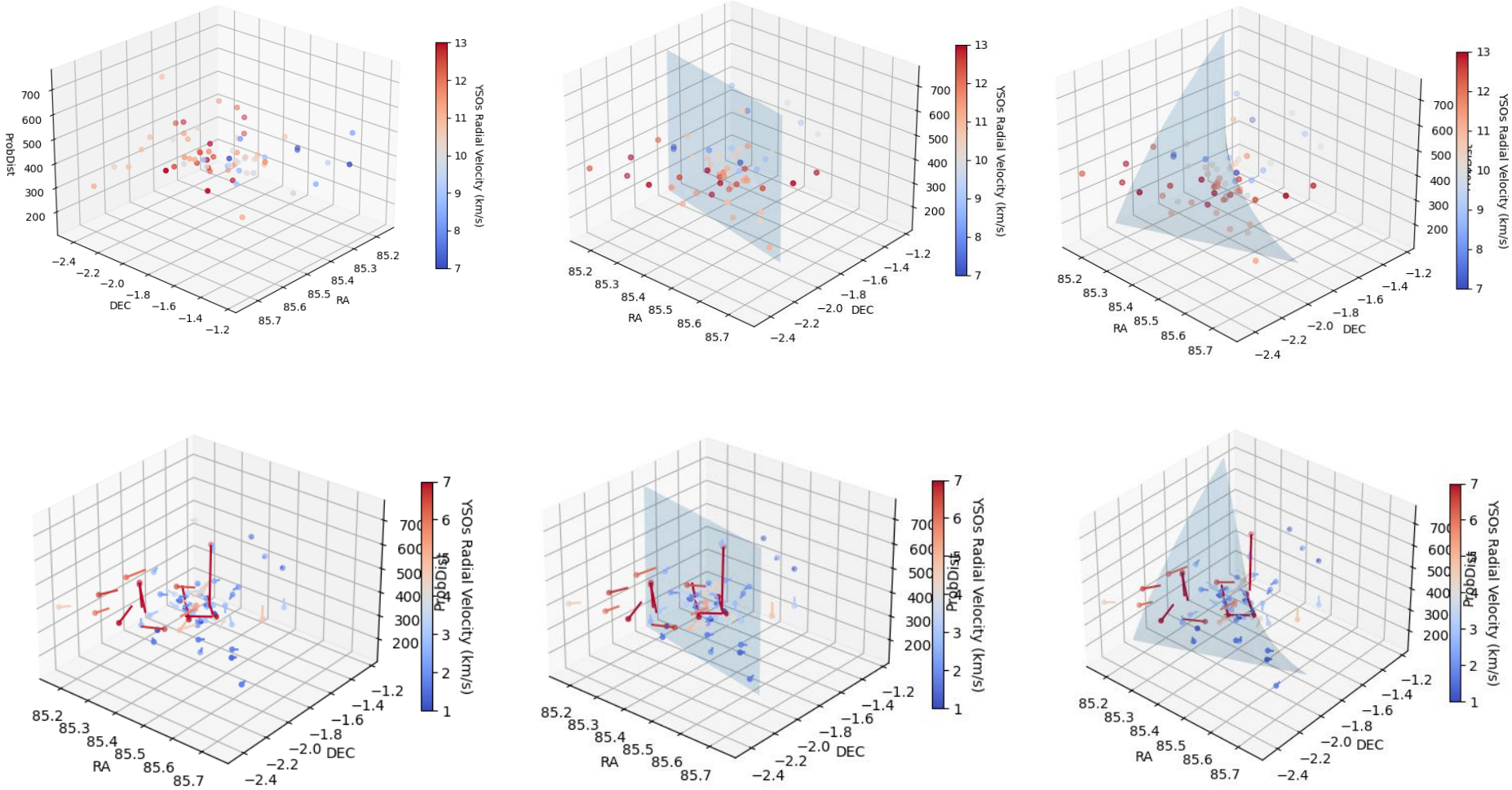
Fonction *velocity_3D* : même principe en trois dimensions. Pour le nuage de vecteurs, le code couleur change par rapport au nuage de points car toutes les vitesses sont cette fois relatives au mouvement moyen du nuage. Ainsi les couleurs représentent cette fois les vitesses totales relatives des YSO.

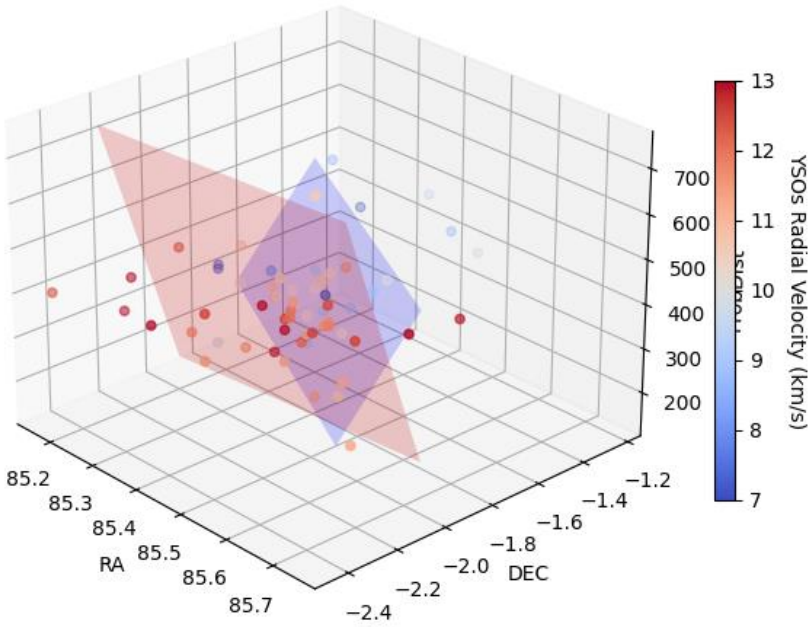
df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom de la vitesse
velerr	String : 'e_RVmean', 'errHRV'…	Nom de l'erreur de vitesse
pm	0 ou 1	Nuage de points (0) ou de vecteurs (1)
fit	0, 1 ou 2	Avec un fitting simple (1), double (2) ou non (0)
order	1 ou 2	Ordre du ou des fitting
vinf	Float	Vitesse max pour l'échantillon bas
vsup	Float	Vitesse min pour l'échantillon haut
vmin	Float	Vitesse min de l'échelle de couleur
vmax	Float	Vitesse max de l'échelle de couleur

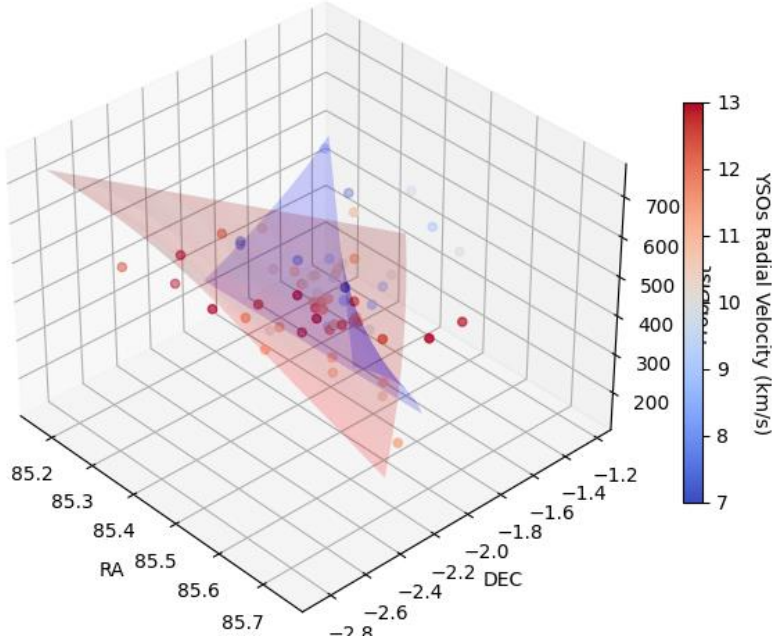
Liste des appels pour les 8 figures types ci-dessous :

```
Velocity, VelErr = 'RVmean', 'e_RVmean'

velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=0, order=0, vinf=0, vsup=0, vmin=7, vmax=13)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=1, order=1, vinf=0, vsup=0, vmin=7, vmax=13)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=1, order=2, vinf=0, vsup=0, vmin=7, vmax=13)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=0, order=0, vinf=0, vsup=0, vmin=1, vmax=7)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=1, order=1, vinf=0, vsup=0, vmin=1, vmax=7)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=1, fit=1, order=2, vinf=0, vsup=0, vmin=1, vmax=7)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=2, order=1, vinf=10, vsup=11, vmin=7, vmax=13)
velocity_3D(df=Tab, velocity=Velocity, velerr=VelErr, pm=0, fit=2, order=2, vinf=10, vsup=11, vmin=7, vmax=13)
```


ProbDist
DEC
RA
YSOs Radial Velocity (km/s)


RA
DEC
YSOs Radial Velocity (km/s)


RA
DEC
YSOs Radial Velocity (km/s)

V. FONCTION D'AFFICHAGE DES CARTES

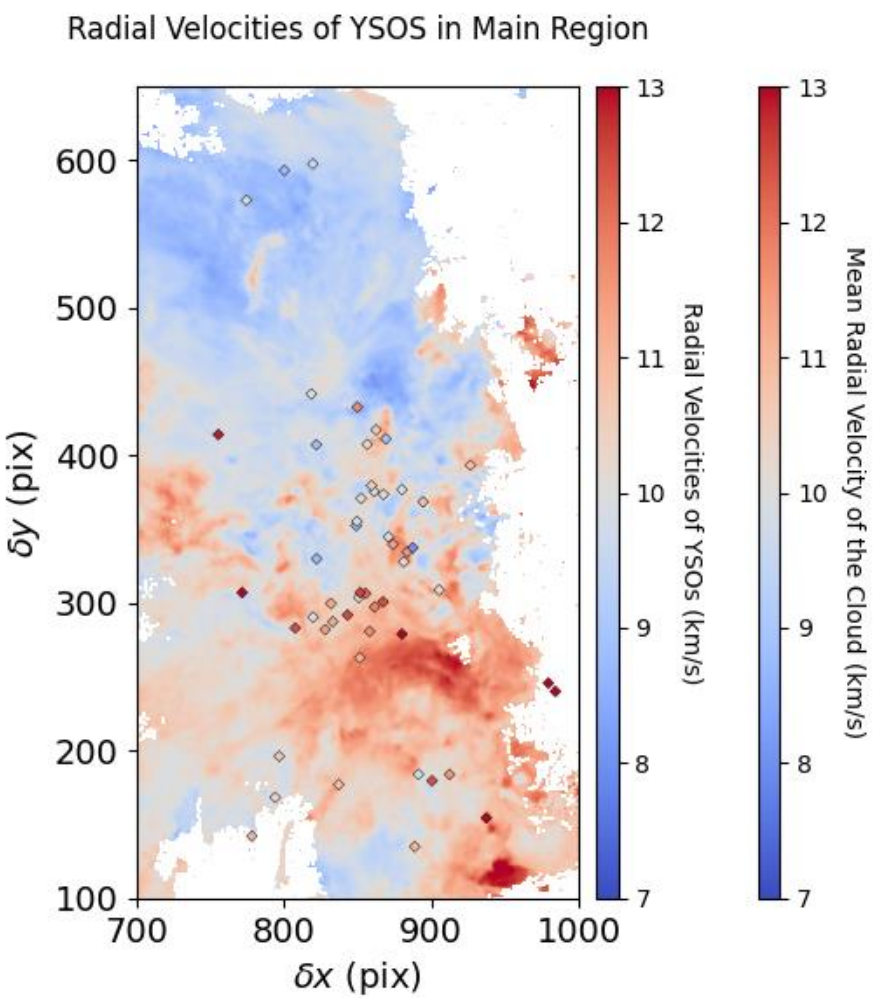
Fonction *mapsplot* : Affiche une multitude de cartes avec ajout sous forme de nuage de points ou de nuage de vecteurs les données des YSOs.

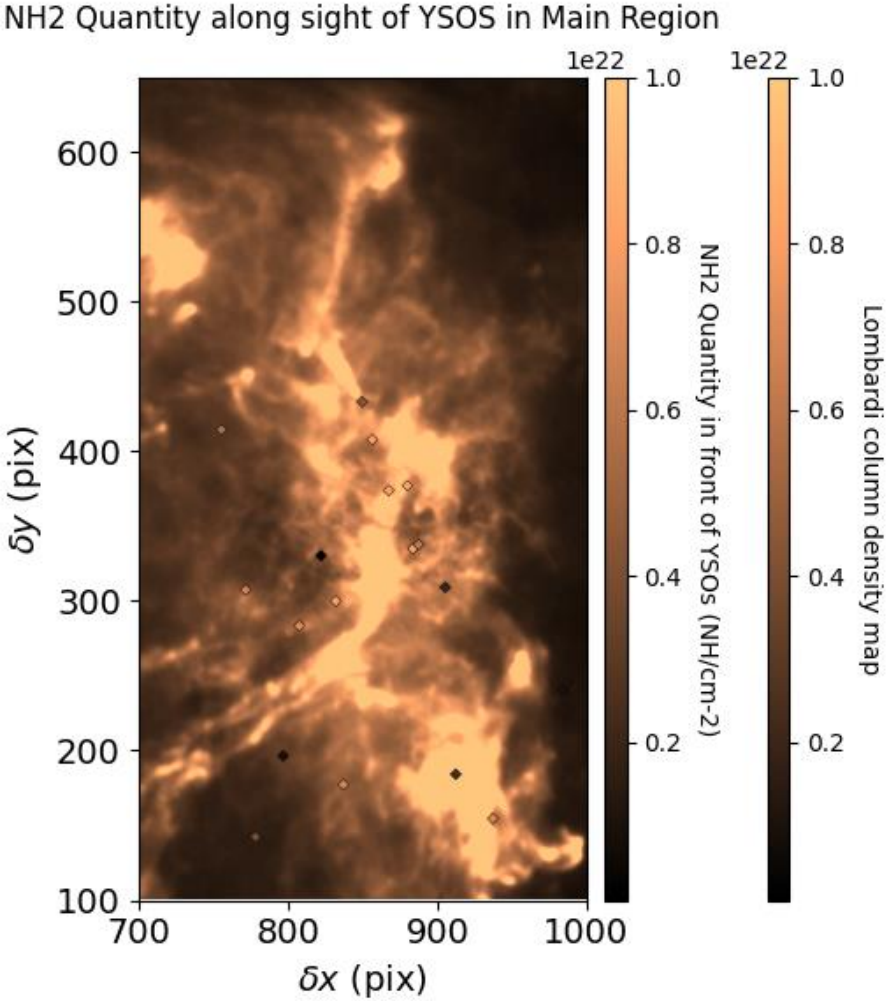
df	Dataframe	Tableau de données à utiliser
plottype	String : 'Scatter' ou 'Quiver'	Type de nuage de points (avec ou sans pm)
maptype	String : 'Integrated Intensity', 'Mean Velocity', 'Velocity Dispersion' ou 'Lombardi'	Type de carte (vitesse ou extinction)
region	String : 'Main', 'Cloak', 'B9' ou Orion'	Région de la carte
p	String : 'Distance', 'Radial Velocity' ou 'Extinction'	Paramètre d'affichage des YSOs
velocity	String : 'RVmean', 'HRV' …	Nom de la vitesse
zoom	0 ou 1	Zoom (1) ou non (0)
axis	[xmin, xmax, ymin, ymax]	Si zoom, délimitations des axes
vmin	Float	Borne inférieure de l'échelle de couleurs des YSOs
vmax	Float	Borne supérieure de l'échelle de couleurs des YSOs

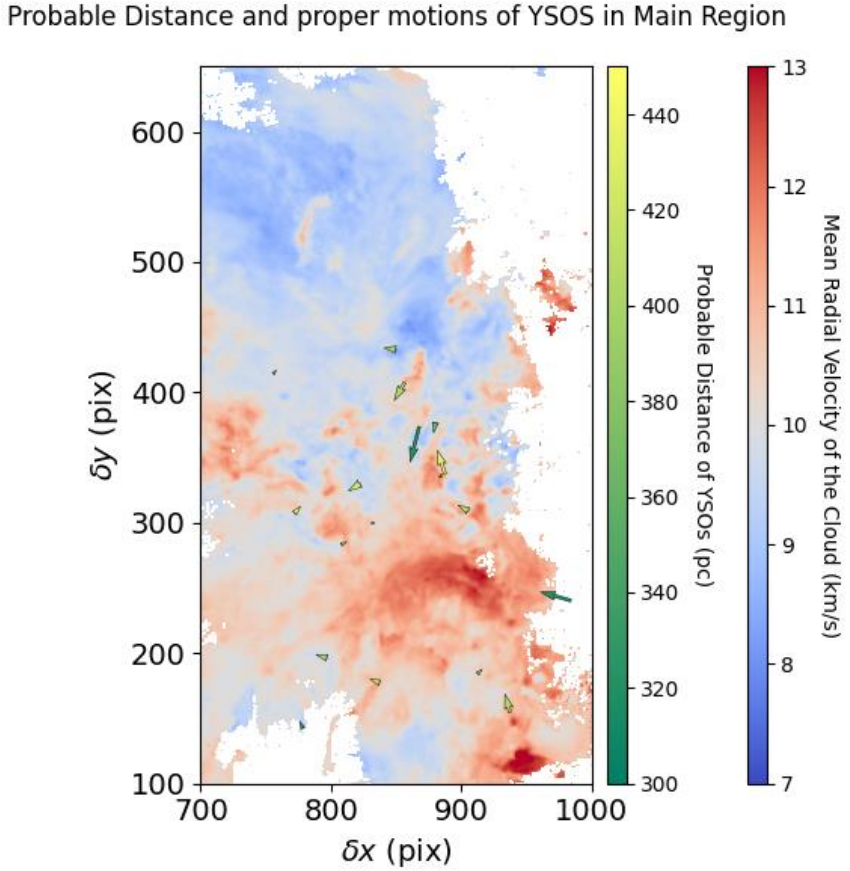
Liste des appels pour les 4 figures types ci-dessous :

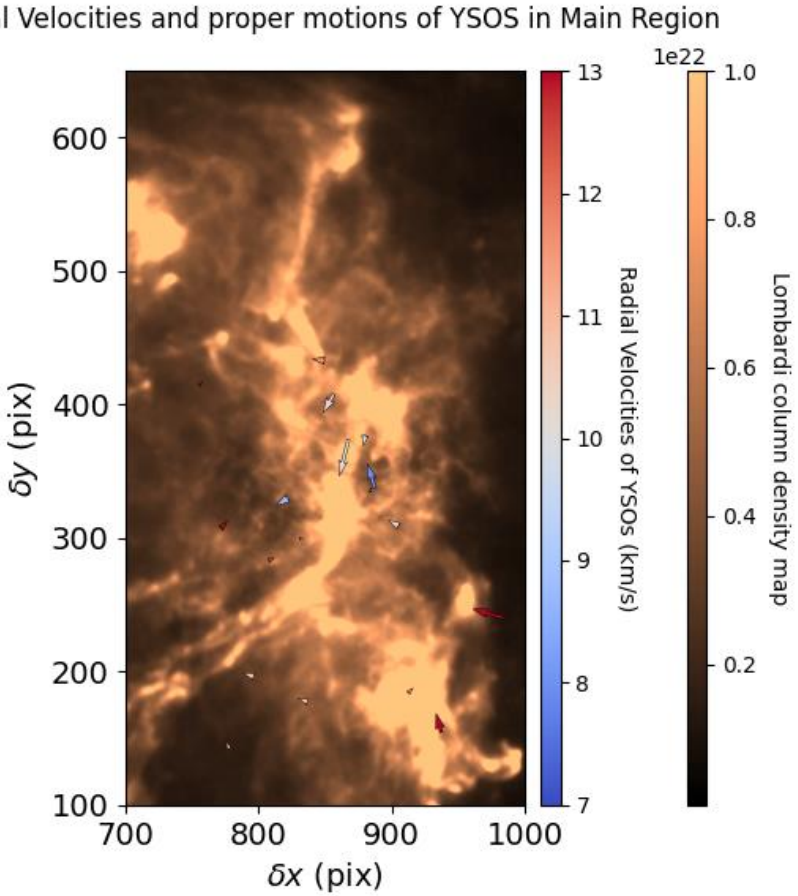
```
Velocity = 'RVmean'

mapsplot(df=Tab, plottype='Scatter', maptype='Mean Velocity', region='Main', p='Radial Velocity', velocity=Velocity,
zoom=1, axis=[700, 1000, 100, 650], vmin=7, vmax=13)
mapsplot(df=Tab, plottype='Scatter', maptype='Lombardi', region='Main', p='Extinction', velocity=Velocity, zoom=1,
axis=[700, 1000, 100, 650], vmin=10**20, vmax=10**22)
mapsplot(df=Tab, plottype='Quiver', maptype='Mean Velocity', region='Main', p='Distance', velocity=Velocity, zoom=1,
axis=[700, 1000, 100, 650], vmin=300, vmax=450)
mapsplot(df=Tab, plottype='Quiver', maptype='Lombardi', region='Main', p='Radial Velocity', velocity=Velocity,
zoom=1, axis=[700, 1000, 100, 650], vmin=7, vmax=13)
```


Radial Velocities of YSOS in Main Region
δy (pix)
δx (pix)
Radial Velocities of YSOs (km/s)
Mean Radial Velocity of the Cloud (km/s)


NH2 Quantity along sight of YSOS in Main Region
δy (pix)
δx (pix)
1e22
NH2 Quantity in front of YSOs (NH/cm-2)
Lombardi column density map


Probable Distance and proper motions of YSOS in Main Region
δy (pix)
δx (pix)
Probable Distance of YSOs (pc)
Mean Radial Velocity of the Cloud (km/s)


Radial Velocities and proper motions of YSOS in Main Region
1e22
δy (pix)
δx (pix)
Radial Velocities of YSOs (km/s)
Lombardi column density map

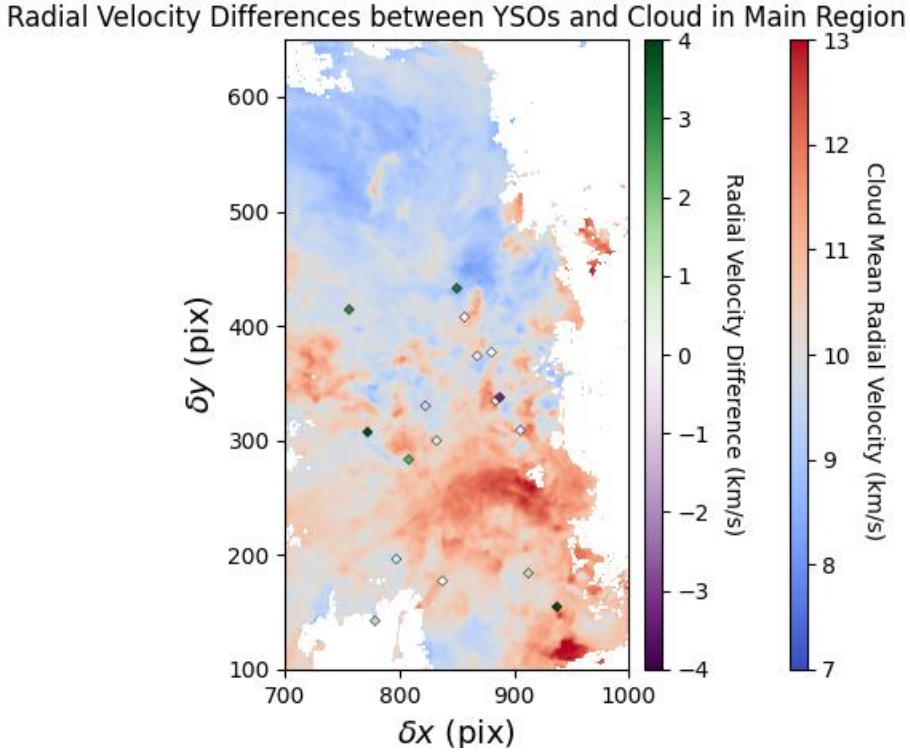
VI. FONCTION POUR LA COMPARAISON QUANTITATIVE

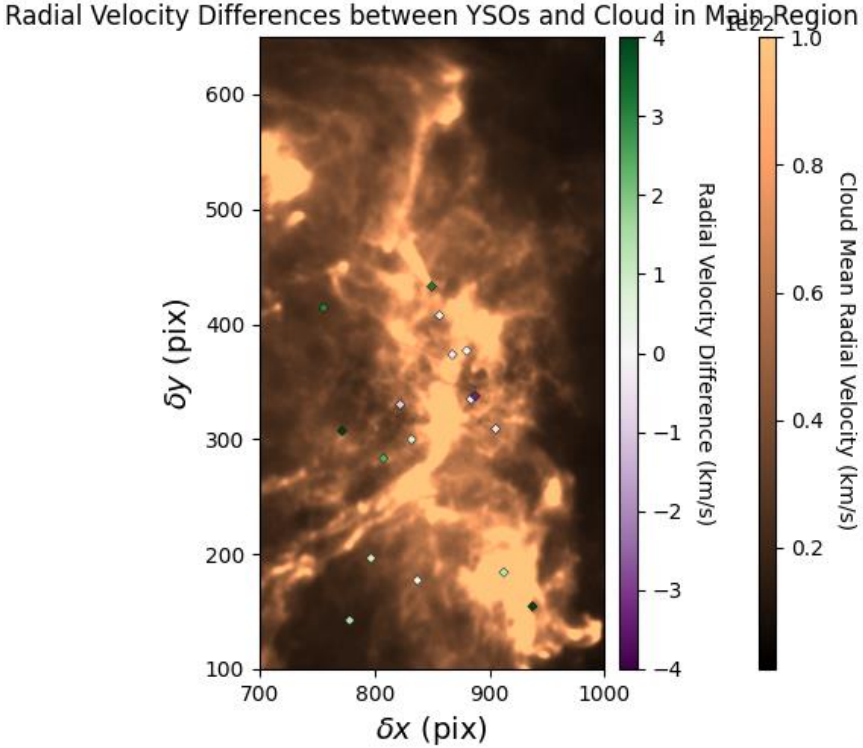
Fonction *comparison* : Affiche les graphiques et cartes avec ajout sous forme de nuage de points les différences de vitesses radiales observées entre les YSOs et le gaz au même point.

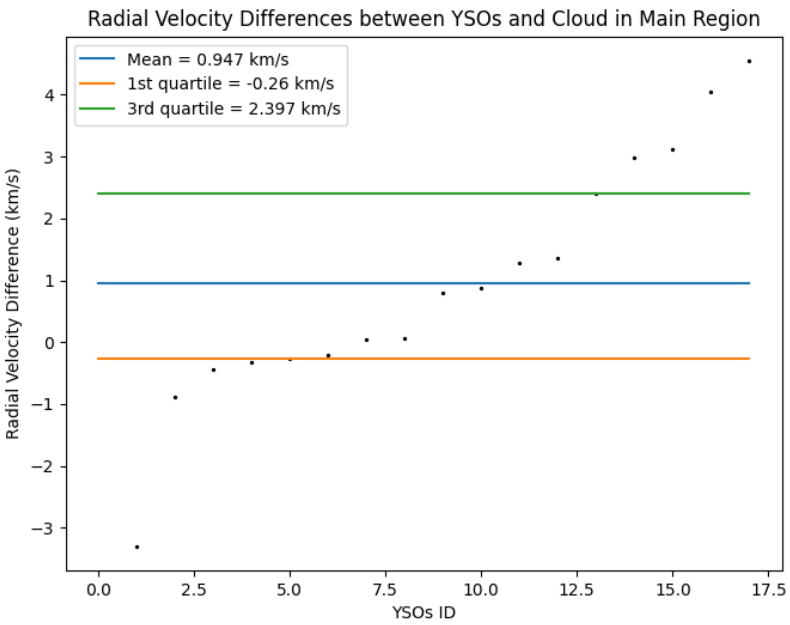
df	Dataframe	Tableau de données à utiliser
velocity	String : 'RVmean', 'HRV' …	Nom de la vitesse
maptype	String : 'Mean Velocity', 'Lombardi'	Type de carte (vitesse ou extinction)
region	String : 'Main', 'Cloak', 'B9' ou Orion'	Région de la carte
plottype	String : 'MapDiff', 'Graph' ou 'Correlation'	Paramètre d'affichage des YSOs
type	String : 'abs'	Pour travailler en valeurs absolues
vmin	Float	Borne inférieure de l'échelle de couleurs des YSOs
vmax	Float	Borne supérieure de l'échelle de couleurs des YSOs
mapplot	String : 'Lombardi'	Pour utiliser la carte de Lombardi
thresh	Float	Seuil de différences accepté pour l'affichage d'un échantillon 'extrême' des YSOs

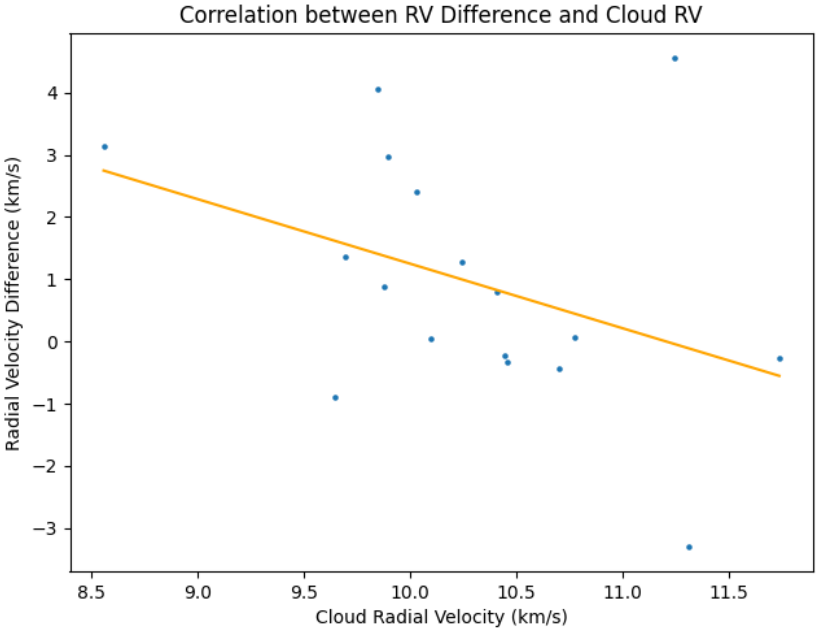
Liste des appels pour les 4 figures types ci-dessous :

```
comparison(df=Tab, velocity=Velocity, maptype='Mean Velocity', region='Main', plottype='MapDiff', type=0, vmin=-4, vmax=4, mapplot=0, thresh=0)
comparison(df=Tab, velocity=Velocity, maptype='Mean Velocity', region='Main', plottype='MapDiff', type=0, vmin=-4, vmax=4, mapplot='Lombardi', thresh=0)
comparison(df=Tab, velocity=Velocity, maptype='Mean Velocity', region='Main', plottype='Graph', type=0, vmin=0, vmax=0, mapplot=0, thresh=0)
comparison(df=Tab, velocity=Velocity, maptype='Mean Velocity', region='Main', plottype='Correlation', type=0, vmin=0, vmax=0, mapplot=0, thresh=0)
```


Radial Velocity Differences between YSOs and Cloud in Main Region
δy (pix)
δx (pix)
Radial Velocity Difference (km/s)
Cloud Mean Radial Velocity (km/s)


Radial Velocity Differences between YSOs and Cloud in Main Region
1e22
δy (pix)
δx (pix)
Radial Velocity Difference (km/s)
Cloud Mean Radial Velocity (km/s)


Radial Velocity Differences between YSOs and Cloud in Main Region
Mean = 0.947 km/s
1st quartile = -0.26 km/s
3rd quartile = 2.397 km/s
Radial Velocity Difference (km/s)
YSOs ID


Correlation between RV Difference and Cloud RV
Radial Velocity Difference (km/s)
Cloud Radial Velocity (km/s)