[천체물리] 중간고사 대체과제

저자	
상태	Done
과목	Physics Python
태그	Paper exam
날짜	@2023/12/14
상위 항목	천체물리학

> **A.** 외계행성 목록을 웹 사이트에서 .csv 파일로 다운받아 두가지 다이어그램을 작성하여 제출
> (행성 발견 방법 별로 구분하여)

Catalogue of Exoplanets

Sortable and filterable catalogue of the exoplanet discovered so far.

https://exoplanet.eu/catalog/

해당 사이트에 접속하여 `Status`에 체크 되어있는 'Confirmed'를 해제하고, csv 형식의 데이터 파일을 다운로드 받는다. 파이썬 프로젝트 파일의 상대 경로 상 같은 폴더에 csv파일을 옮긴 후, 필요한 모듈과 함께 데이터를 불러온다.

```
# 모듈 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plot

#데이터 불러오기
df = pd.read_csv('./exoplanet.eu_catalog.csv')
df.head()
```

1. **Star mass vs Planet mass**
   (start_mass, mass)

불러온 외계 행성 데이터에서 확인된(Confirmed) 행성을 필터링하고`planet_status`, 행성 발견 방법 `detection_type` 별로 구분하여 태양 질량으로 나타낸 항성질량 `star_mass`과 목성질량으로 나타낸 행성질량 `mass`의 산점도를 로그스케일로 플로팅 한다.

```
# 확인된 행성 데이터 프레임
df_confirmed = df[(df['planet_status']=='Confirmed')]

# 발견 방법 목록
detection_type = df_confirmed['detection_type'].unique()

plot.figure(1, figsize=(10,8))
plot.title('Star Mass vs Planet Mass',fontsize = 35 ,fontweight='bold')
```

```
# 발견 방법 별 산점도 플롯
for index,type_name in enumerate(detection_type):
    plot.scatter(df_confirmed['star_mass'][df_confirmed['detection_type']==type_name],
                 df_confirmed['mass'][df_confirmed['detection_type']==type_name],
                 s=3,label=type_name)

# Mass Ratio
plot.plot([0.01, 10],[0.001,1], '-',label='q=0.0001',alpha = 0.5)
plot.plot([0.01, 10],[0.01,10], '-',label='q=0.001',alpha = 0.5)
plot.plot([0.01, 10],[0.1,100], '-',label='q=0.01',alpha = 0.5)
plot.plot([0.01, 10],[1,1000], '-',label='q=0.1',alpha = 0.5)

plot.xscale('log')
plot.yscale('log')

plot.xlabel('Star Mass(M$_\odot$)',fontsize = 20)
plot.ylabel('Planet Mass(M$_J$)',fontsize = 20)

plot.xlim(0.01,10)
plot.ylim(0.0004,100)

plot.grid(True, which="both", ls="-",alpha = 0.5)
plot.legend(loc=4,fontsize=7, frameon=True)

plot.show()
```

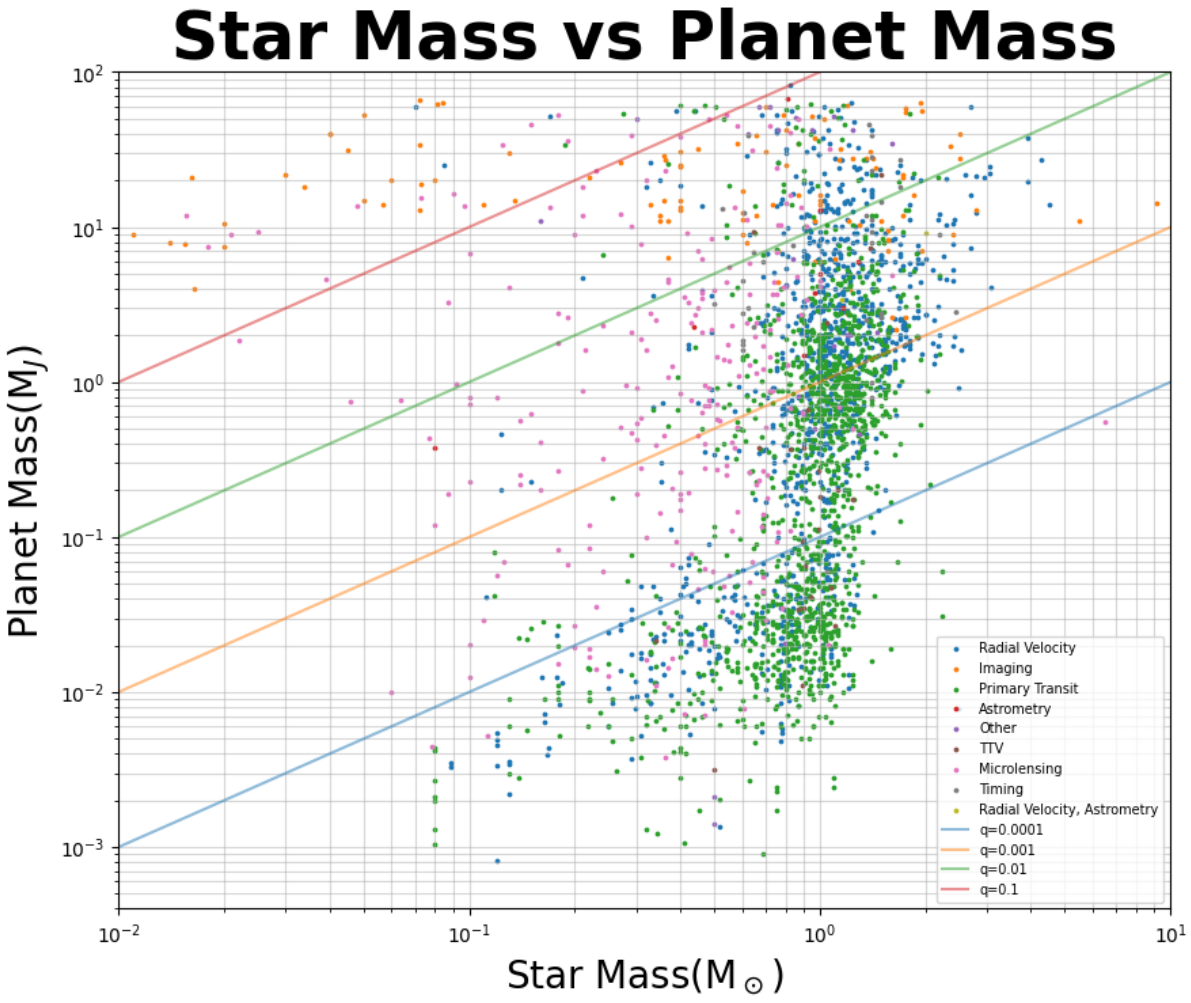

Figure 1. 행성 발견 방법 별 항성 대 행성 질량 그래프. 선은 비율이 q인 지점을 의미한다.

2. **Planet semi-major-axis vs Planet orbit period**
   (orbit_period,semi_major)

1번과 같은 방법으로 확인된 외계행성 데이터로 행성의 공전주기 `obital_period`와 장반경 `semi_major_axis`의 산점도를 플로팅한다.

```
plot.figure(2,figsize=(10,8))

plot.title('Planet Semi-Major-Axis vs Planet Orbital Period',fontsize = 20 ,fontweight='bold')

# 발견 방법 별 산점도 플로팅
for index,type_name in enumerate(detection_type):
    plot.scatter(df_confirmed['orbital_period'][df_confirmed['detection_type']==type_name],
                 df_confirmed['semi_major_axis'][df_confirmed['detection_type']==type_name],
                 s=3)

plot.xscale('log')
plot.yscale('log')

plot.xlabel('Planet Orbital Period [Day]',fontsize = 20)
plot.ylabel('Plaent Semi-Major-Axis [AU]',fontsize = 20)

plot.grid(True, which="both", ls="-",alpha = 0.5)
plot.legend(detection_type,loc=4,fontsize=10, frameon=True)

plot.show()
```

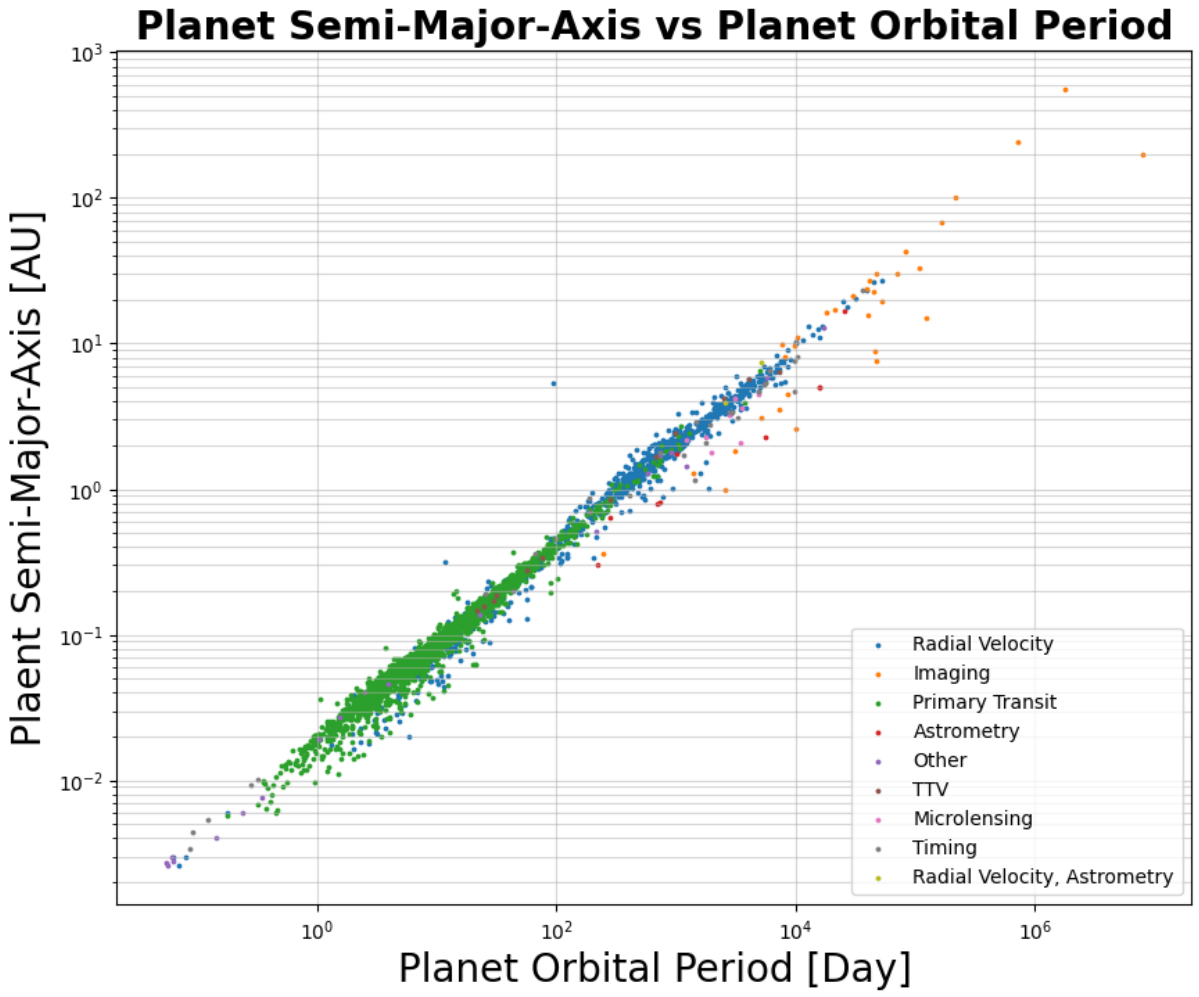

Figure 2. 행성 발견 방법 별 공전 주기 대 장반경 그래프.

> **B.** 위 목록에서 모성으로부터 나오는 행성표면 단위면적 $(1\mathrm{m}^2)$당 도달하는 에너지를 계산하여 지구에 도달하는 에너지와 비슷한 행성들을 5개 찾아보시오.
> (semi_major, star_teff, star_radius)

슈테판-볼츠만 법칙에 의해서 반지름이 r, 표면 온도가 T인 모성으로부터 a만큼 떨어진 행성의 단위면적 당 도달하는 에너지 E를 구할 수 있다.

$$E = \frac{4\pi r^2 \sigma T^4}{4\pi a^2}$$

$$= \left(\frac{r}{a}\right)^2 \sigma T^4$$

이 식을 통해 지구 위치$(1\mathrm{AU})$에서 단위면적당 태양의 에너지를 구할 수 있다. 이를 태양 상수 S라고 하며, 태양의 크기 $R_\odot$, 온도 $T_\odot$에 대해 다음과 같다.

$$S = \left(\frac{R_\odot}{1\mathrm{AU}}\right)^2 \sigma T_\odot^2$$

태양의 지름을 천문단위로 나타내면 약 $0.004652\mathrm{AU}$, 태양의 온도는 약 $6000\mathrm{K}$이다. 슈테판-볼츠만 상수 $5.673 \times 10^{-8}\mathrm{W/m^2 \cdot K^4}$를 대입하면 S는 약 $1591\mathrm{W/m^2}$이다.

`flux_ratio`를 구하기 위한 식은 다음과 같다.

$$F_{\text{ratio}} = \frac{E}{S}$$

$$= \frac{\left(\frac{r}{a}\right)^2 \cancel{\sigma} T^4}{\left(\frac{R_\odot}{1\mathrm{AU}}\right)^2 \cancel{\sigma} T_\odot^4}$$

$$= \frac{1}{a^2} \cdot \left(\frac{r}{R_\odot}\right)^2 \cdot \left(\frac{T}{T_\odot}\right)^4$$

데이터에서 항성의 반지름 r은 태양반경 $R_\odot$ 단위이므로 정리하면

$$F_{\text{ratio}} = \left(\frac{r}{a}\right)^2 \cdot \left(\frac{T}{T_\odot}\right)^4$$

필요한 상수를 선언하고 태양 상수를 구한 후, 데이터에서 행성의 장반경 `semi_major_axis`, 항성의 온도 `star_teff`와 반지름 `star_radius`을 통해 태양상수와 단위면적당 에너지의 비율 `flux_ratio`를 구해 허용치 내에 있는 항성 목록을 내림차순으로 정리하여 상위 5개의 목록을 가져온다. 허용치는 10%로 설정하였다.

```
#필요한 상수
o = 5.673e-8 #W/m^2
Sol_rad = 0.004652 # AU
Sol_teff = 6000 # K

# 태양상수 계산과 허용비율
S = (Sol_rad**2)*o*Sol_teff**4
percent = 10

# 태양상수, 허용치(허용비율) 출력
print('Solar Constant :',round(S),u"\u00B1",round(S * 0.01 * percent),f'W/m^2 ({percent}%)')

df['flux_ratio'] = ((df['star_radius']/df['semi_major_axis'])**2)*(df['star_teff']/Sol_teff)**4

# 범위에 해당하는 데이터 프레임 저장
df_filterd = df_confirmed[(df['flux_ratio'] > (1-0.01*percent)) & (df['flux_ratio'] < (1+0.01*percent))]

# 필요한 데이터만 가져와서 새로 저장
EarthLikePlanet = df_filterd[['name','semi_major_axis','star_teff','star_radius','flux_ratio']]
# flux ratio 내림차순 정렬
EarthLikePlanet.sort_values('flux_ratio',ascending=False).head()
```

	name	semi_major_axis	star_teff	star_radius	flux_ratio
7575	Ross 508 b	0.053	3071.0	0.211	1.087746
1290	HD 216520 c	0.528	5103.0	0.760	1.084068
1463	HD 4203 b	1.164	5701.0	1.33	1.064134

	name	semi_major_axis	star_teff	star_radius	flux_ratio
6471	Kepler-61 b	0.270	4017.0	0.620	1.059396
515	GJ 273 b	0.091	3382.0	0.293	1.046506

▼ 실제 출력결과

```
Solar Constant : 1591 ± 159 W/m^2 (10%)
/var/folders/t3/cmpm9p3n4sl01hqyv6l_d7n80000gn/T/ipykernel_66688/5
  df_filterd = df_confirmed[(df['flux_ratio'] > (1-0.01*percent))
```

	name	semi_major_axis	star_teff	star_radius	flux_ratio
7575	Ross 508 b	0.053	3071.0	0.211	1.087746
1290	HD 216520 c	0.528	5103.0	0.760	1.084068
1463	HD 4203 b	1.164	5701.0	1.330	1.064134
6471	Kepler-61 b	0.270	4017.0	0.620	1.059396
515	GJ 273 b	0.091	3382.0	0.293	1.046506

> **C.** $0.1 \sim 1.5M_{\odot}$범위를 가지는 항성의 Habitable Zone를 그리고 태양계와 Gliese 581, 그리고 TRAPPIST-1 시스템을 표시하시오.
> (가로 flux on planet 범위는 각자 알아서 설정)

항성의 광도를 $L = 4\pi r\sigma T^4$로 쓸 수 있으므로, 행성이 L의 광도를 갖는 항성으로부터 받는 단위 면적당 에너지는 다음과 같이 쓸 수 있다.

$$E = \frac{L}{4\pi a^2}$$

태양광도를 $L_{\odot}$라고 할 때 $L, L_{\odot}$를 통해 표현하는 F_{ratio}는

$$F_{\text{ratio}} = \frac{L}{\cancel{4\pi}a^2}\left(\frac{L_{\odot}}{\cancel{4\pi}R^2}\right)^{-1}$$

$$= \left(\frac{R}{a}\right)^2 \cdot \frac{L}{L_{\odot}}$$

B번에서 구한 F_{ratio}는 지구를 기준으로 하였으므로 생명이 존재할 수 있는 지대의 값을 1로 볼 수 있다. 따라서 항성으로부터 단위 면적당 받는 에너지가 지구와 같은 행성의 위치 a(장반경)에 대해서 위 식을 정리하면

$$a = \sqrt{R^2 \cdot \frac{L}{L_{\odot}}} \ (\text{when } F_{\text{ratio}} = 1)$$

또한 항성의 질량 M에 따른 질량-광도 관계는 다음과 같다. (태양질량 $= M_{\odot}$)

$$\frac{L}{L_\odot} \approx 0.23 \left(\frac{M}{M_\odot}\right)^{2.3} (\text{where } M < 0.43 M_\odot)$$

$$\frac{L}{L_\odot} = \left(\frac{M}{M_\odot}\right)^{4} (\text{where } 0.43 M_\odot < M < 2 M_\odot)$$

F_{ratio}를 주어진 항성의 질량 범위 $0.1 \sim 1.5 M_\odot$를 두 구간으로 나누어 다시 쓰면

$$a_1 = \sqrt{R^2 \cdot 0.23 \left(\frac{M}{M_\odot}\right)^{2.3}} (\text{where } M < 0.43 M_\odot)$$

$$a_2 = \sqrt{R^2 \cdot \left(\frac{M}{M_\odot}\right)^{4}} (\text{where } 0.43 M_\odot < M < 1.5 M_\odot)$$

그리고 태양계의 생명 가능지대 범위를 $0.5\text{AU} < R < 1.5\text{AU}$라고 할 때, 범위 양 끝의 값을 R에 대입하여 질량이 M인 항성의 생명가능지대를 구할 수 있다.

이 생명가능지대를 태양계, Gliese 581, TRAPPIST-1 시스템과 비교하기 위해 각각의 데이터 프레임을 만든다. csv파일은 외계 행성에 대한 데이터이므로 태양계는 직접 값을 넣어 만들고, 나머지는 모항성의 이름 `star_name`이 각각 GJ 581, TRAPPIST-1인 행성을 필터링하여 만든다.

TRAPPIST-1의 질량 `star_mass`가 문제에 주어진 범위보다 작은 $0.08 M_\odot$이므로, $0.01 \sim 1.5 M_\odot$의 값을 임의로 생성하여 생명가능지대 경계선을 계산하여 각 항성계와 함께 로그스케일로 플롯한다. 항성의 이름, 행성 번호나 이름과 같이 필요한 정보를 표기하고 x축과 y축의 범위를 제한하여 그래프를 만든 뒤 그 결과를 확인한다.

```
# Solar System
ss= [['Mercury',0.39,'Sun',1],
     ['Venus',0.72,'Sun',1],
     ['Earth',1,'Sun',1],
     ['Mars',1.52,'Sun',1],
     ['Jupiter',5.2,'Sun',1],
     ['Saturn',9.5,'Sun',1],
     ['Uranus',19.2,'Sun',1],
     ['Neptune',30.1,'Sun',1]]

SolarSystem = pd.DataFrame(data = ss,columns= ['name','semi_major_axis','star_name','star_mass'])

# Gliese 581 System
Gliese581system = df[(df['star_name']=='GJ 581')]

# TRAPPIST - 1 System
TRAPPIST1system = df[(df['star_name'])=='TRAPPIST-1']

# Flux on Planet 범위
R_min = 0.7 #AU
R_max = 1.5 #AU

# Star Mass 리스트 생성
a_1_mass = pd.DataFrame({'star_mass' : np.linspace(0.01,0.43,3000)})
a_2_mass = pd.DataFrame({'star_mass' : np.linspace(0.43,1.5,3000)})

a_1 = pd.DataFrame({'star_mass' : np.linspace(0.01,0.43,3000),
                    'a_min' : (0.23*(R_min**2)*(a_1_mass['star_mass']**2.3))**0.5,
                    'a_max' : (0.23*(R_max**2)*(a_1_mass['star_mass']**2.3))**0.5
                    })
```

```
a_2 = pd.DataFrame({'star_mass' : np.linspace(0.43,1.5,3000),
                    'a_min' : ((R_min**2)*a_2_mass['star_mass']**4)**0.5,
                    'a_max' : ((R_max**2)*a_2_mass['star_mass']**4)**0.5
                    })

plot.figure(3,figsize=(10,8))

plot.title('Habitable Zone of Star, (0.1~1.5M$_\odot$)',fontsize = 20 ,fontweight='bold')

# Plot Host Star Systems
# Solar System
plot.scatter(SolarSystem['semi_major_axis'],SolarSystem['star_mass'],s=800,label='Solar System')
# Planet Name
for i, name in enumerate(SolarSystem['name'][0:-2]):
    plot.text(SolarSystem.iloc[i]['semi_major_axis'],SolarSystem.iloc[i]['star_mass']-0.13,name,
              horizontalalignment='center',
              verticalalignment='top',
              fontsize = 8,
              fontweight = 'bold'
              )
# Gliese 581
plot.scatter(Gliese581system['semi_major_axis'],Gliese581system['star_mass'],s=500,label = 'Gliese 581')
# Planet Numer
for i, name in enumerate(Gliese581system['name']):
    plot.text(Gliese581system.iloc[i]['semi_major_axis'],Gliese581system.iloc[i]['star_mass'],name[-1],
              horizontalalignment='center',
              verticalalignment='center',
              color = 'white',
              fontweight = 'bold')
# TRAPPIST-1
plot.scatter(TRAPPIST1system['semi_major_axis'],TRAPPIST1system['star_mass'],s=200,label = 'TRAPPIST-1')
# Planet Number
for i, name in enumerate(TRAPPIST1system['name']):
    plot.text(TRAPPIST1system.iloc[i]['semi_major_axis'],TRAPPIST1system.iloc[i]['star_mass'],name[-1],
              horizontalalignment='center',
              verticalalignment='center',
              color = 'white',
              fontweight = 'bold')

# Habitable Zone Plot
plot.plot(a_1['a_max'],a_1['star_mass'],'C0',alpha = 0.5)
plot.plot(a_1['a_min'],a_1['star_mass'],'C0',alpha = 0.5)
plot.plot(a_2['a_max'],a_2['star_mass'],'C0',alpha = 0.5)
plot.plot(a_2['a_min'],a_2['star_mass'],'C0',alpha = 0.5)

plot.xlabel('Distance from Star (AU)',fontsize = 20)
plot.ylabel('Mass of Star (M$_\odot$)',fontsize = 20)

plot.xscale('log')
plot.yscale('log')

plot.xlim(0.01,10)
plot.ylim(0.05,1.5)

# y-axis value
s = SolarSystem['star_mass'].iloc[0]
g = Gliese581system['star_mass'].iloc[0]
t = TRAPPIST1system['star_mass'].iloc[0]
plot.yticks([s,g,t],[f'Sun\n{s}',f'Gliese 581\n{g}',f'TRAPPIST - 1\n{t}'])

plot.legend(loc=4,fontsize=10, frameon=True)
plot.grid(True, which="both", ls="-",alpha = 0.5)

plot.show()
```

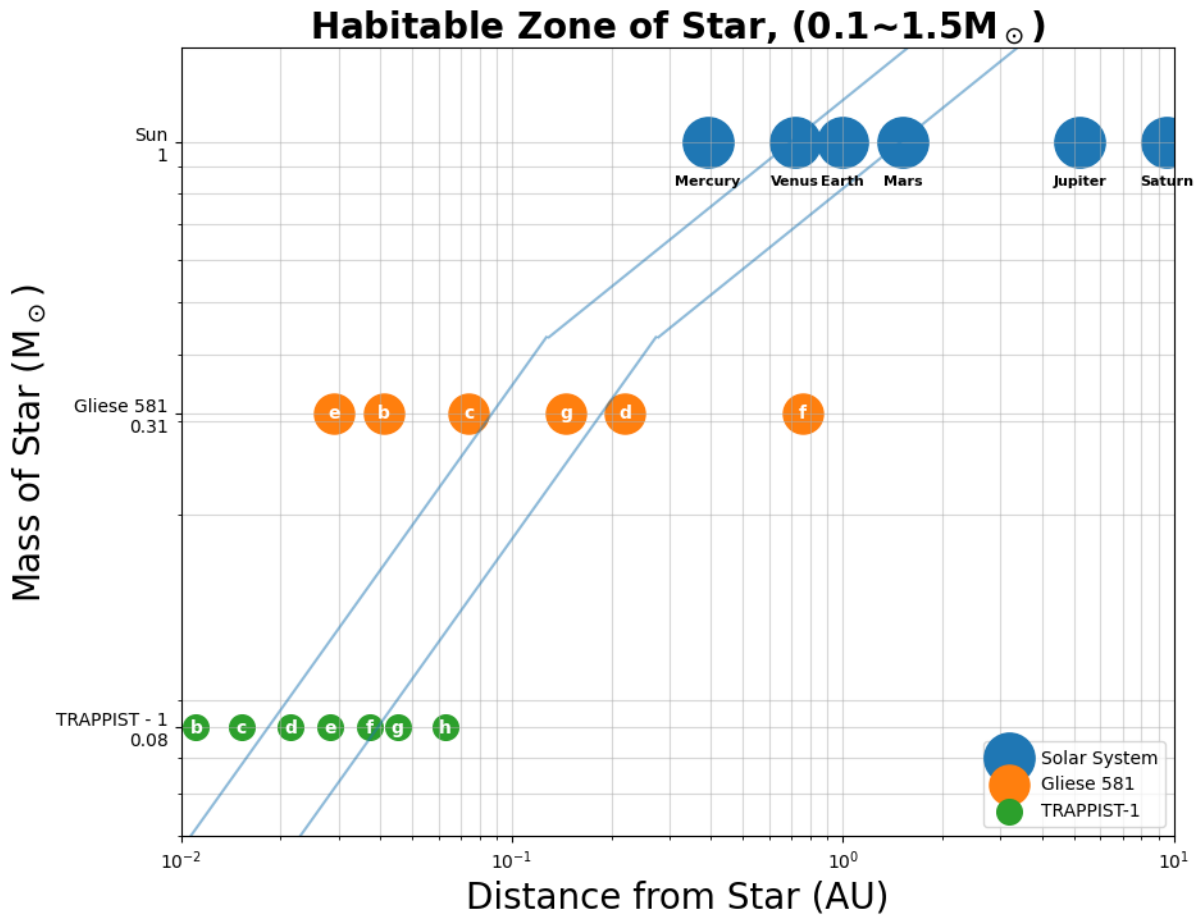

Figure 4. 항성 질량에 따른 생명가능지대 범위(선)과 각 항성계를 나타낸 그래프.