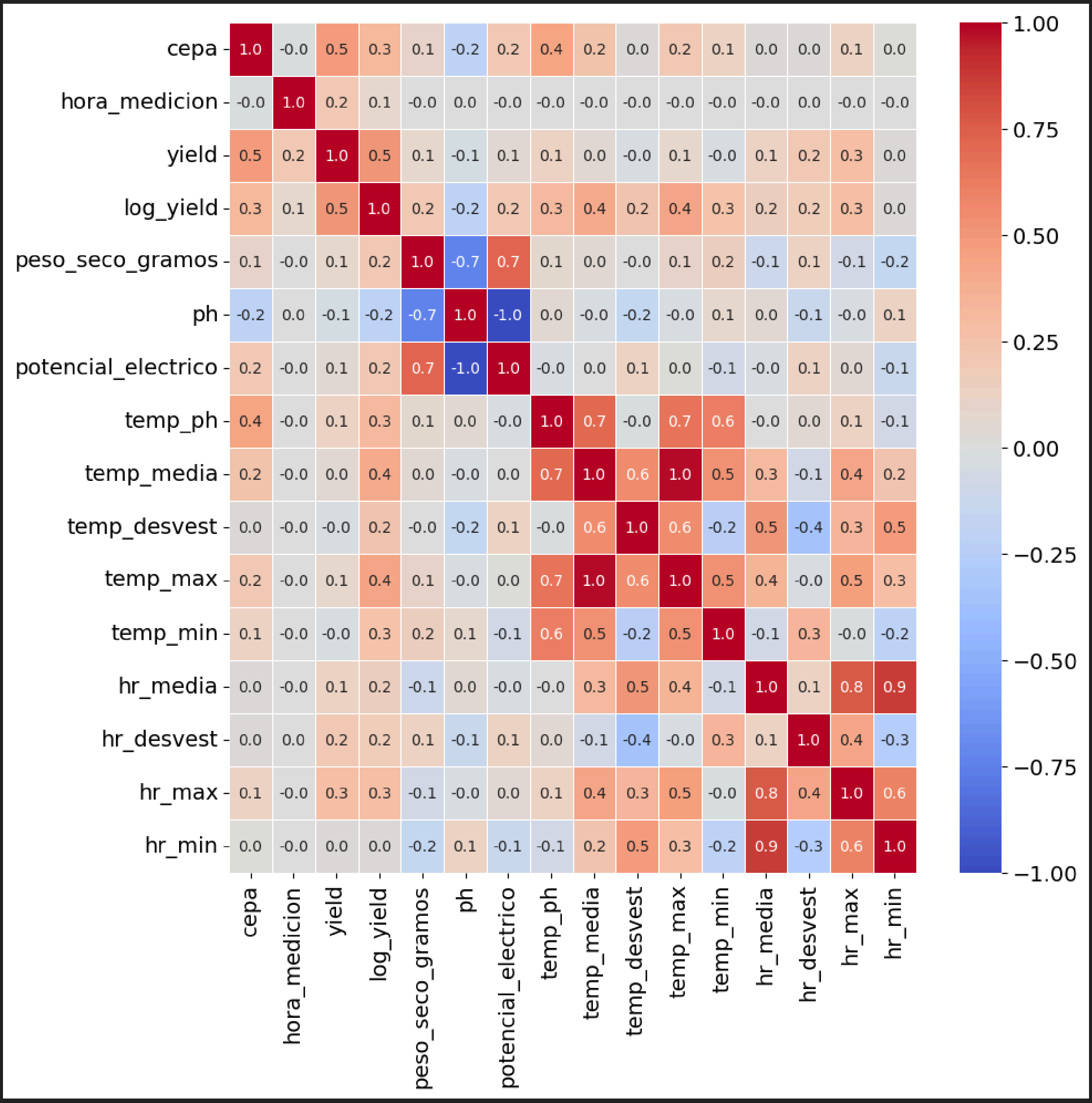
A boy doing his PhD collected a lot of data. For instance, he got a large piece of data that looks something like this:

```
df.dropna().head()
```

	cepa	sp	sp_detalle	fecha_inicio	bloque	hora_medicion	yield	log_yield	peso_seco_gramos	ph	potencial_electrico	temp_ph	temp_media
12	85	Metarhizium anisopliae	sensu stricto	2023-08-14	3	48	1.108333e+08	8.044670	2.197	4.75	163.0	20.0	21.134437
13	85	Metarhizium anisopliae	sensu stricto	2023-08-14	3	72	2.933333e+08	8.467361	2.197	4.75	163.0	20.0	21.134437
14	85	Metarhizium anisopliae	sensu stricto	2023-08-14	3	96	2.816667e+08	8.449735	2.197	4.75	163.0	20.0	21.134437
15	85	Metarhizium anisopliae	sensu stricto	2023-08-14	3	48	1.105833e+08	8.043690	2.246	4.80	160.0	20.0	21.134437
16	85	Metarhizium anisopliae	sensu stricto	2023-08-14	3	72	2.300000e+08	8.361728	2.246	4.80	160.0	20.0	21.134437

He wants to know if there is some Pearson correlation between some of the variables. He decides to do…

```
corr = df[["cepa", "hora_medicion", "yield", "log_yield", "peso_seco_gramos",
         "ph", "potencial_electrico", "temp_ph", "temp_media", "temp_desvest",
         "temp_max", "temp_min", "hr_media", "hr_desvest", "hr_max", "hr_min"]].corr()
plt.figure(figsize=(10, 10))
sns.heatmap(corr, vmin=-1, vmax=1,
          annot=True, fmt='.1f',
          cmap='coolwarm', linewidths= 0.5,
          #annot_kws={"size": 7}
         )
plt.tight_layout()
plt.show()
```

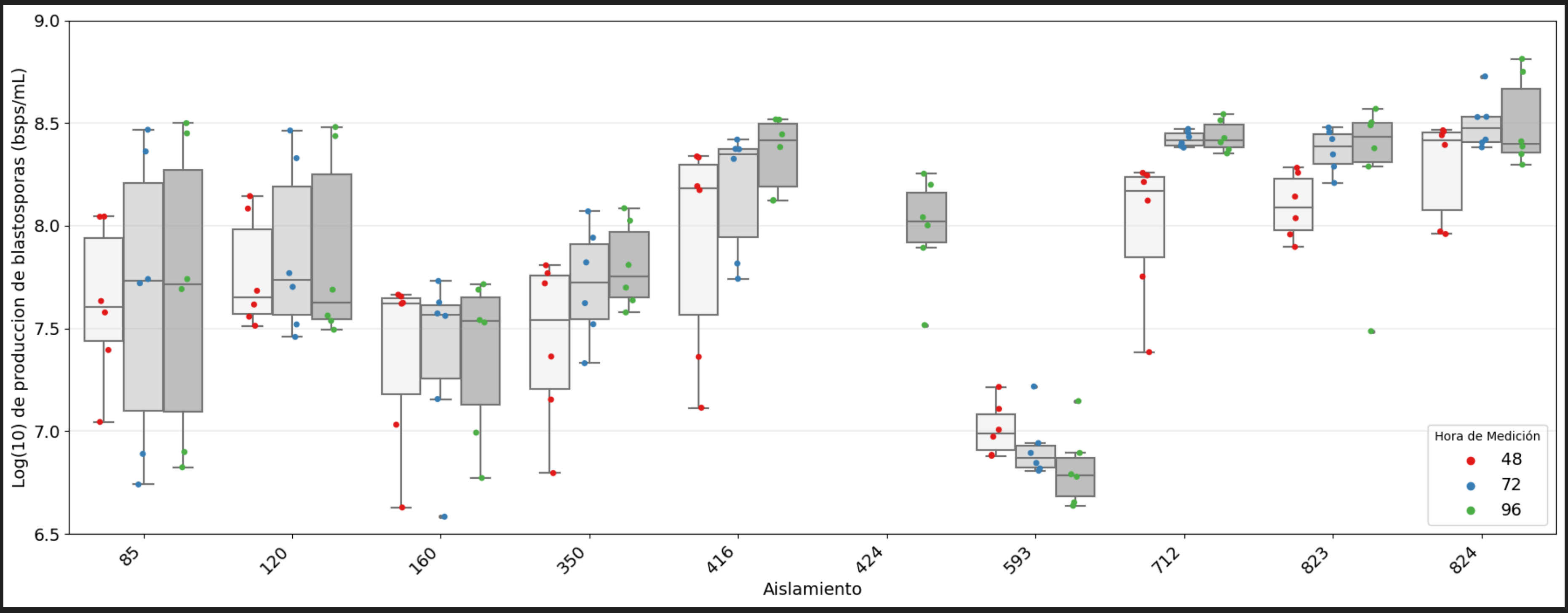


He is very interested in the variable "yield" which he previously normalized by the logarithm of base 10. So he makes some plots…

```python
plt.figure(figsize=(18, 7))
boxp = sns.boxplot(data=df_para_plots1, x='cepa', y='log_yield', hue='hora_medicion',
                   #showmeans=True,
            flierprops={"marker": "."},
             #color="0.9",
            palette=colors
           )
strip = sns.stripplot(x="cepa", y="log_yield", data=df_para_plots1, hue="hora_medicion",
             palette="Set1",
              dodge=True, jitter=True
             )

handles, labels = strip.get_legend_handles_labels()
plt.legend(handles[3:], labels[3:], title="Hora de Medición", loc="lower right")

plt.xlabel('Aislamiento')
plt.ylabel('Log(10) de produccion de blastosporas (bsps/mL)')
plt.xticks(rotation=45, ha='right')
plt.yticks(ticks=(1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9))
plt.ylim(6.5, 9)
plt.grid(True, linestyle='-', linewidth=.2, ydata=None)
plt.tight_layout()
plt.show()
```



He would like to know if there are (statistically significant) differences between the different fermentation times for each fungal isolate, so…

```python
#Obtener las cepas únicas
cepas_unicas = df['cepa'].unique()
#Iterar sobre cada cepa única
for cepa in cepas_unicas:
    #Filtrar el DataFrame para la cepa actual y seleccionar solo las filas con valores de log_yield no NaN
    df_cepa = df[(df['cepa'] == cepa) & (~df['log_yield'].isna())]

    #Crear listas de log_yield para cada valor de hora_medicion
    log_yield_48 = df_cepa[df_cepa['hora_medicion'] == 48]['log_yield']
    log_yield_72 = df_cepa[df_cepa['hora_medicion'] == 72]['log_yield']
    log_yield_96 = df_cepa[df_cepa['hora_medicion'] == 96]['log_yield']

    #Realiza ANOVA
    anova_result = f_oneway(log_yield_48, log_yield_72, log_yield_96)

    #Print resultados
    print(f"ANOVA para cepa {cepa}:")
    print("F:", anova_result.statistic)
    print("P valor:", anova_result.pvalue)
    print("\n")
```

and found that the only fungal isolate that showed significant differences is 712:

```
ANOVA para cepa 712:
F: 8.4859342859308
P valor: 0.0034270804553824647
```

The ANOVA procedure tells us that there is at least one group that differs from the rest, but not which group or how it differs. For this purpose, we performed a robust post hoc analysis such as the HSD Tukey test:

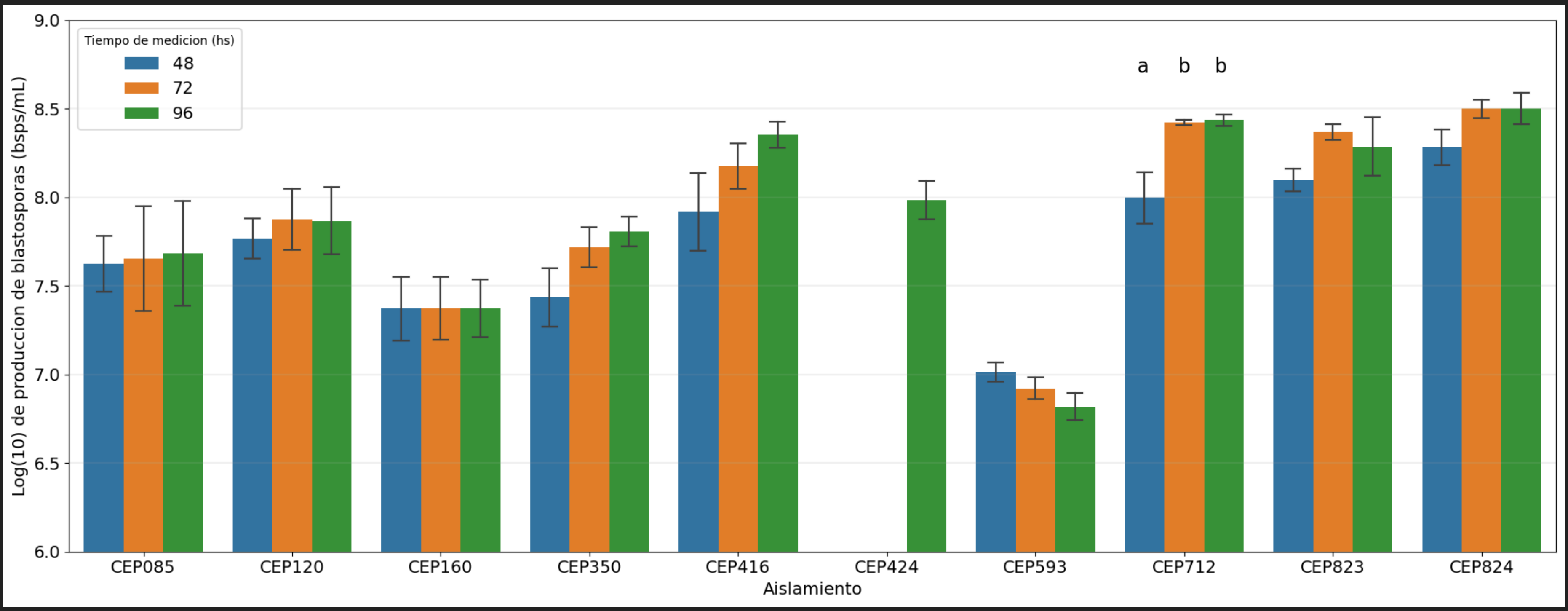
```python
#Filtrar el DataFrame para la cepa 712 y seleccionar solo las filas con valores de log_yield no NaN
df_cepa_712 = df[(df['cepa'] == 712) & (~df['log_yield'].isna())]

#Realizar la prueba de Tukey como análisis post hoc
tukey_result = pairwise_tukeyhsd(df_cepa_712['log_yield'], df_cepa_712['hora_medicion'])
#Imprimo resultados
print(tukey_result)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================
group1 group2 meandiff p-adj   lower  upper  reject
---------------------------------------------------
    48     72   0.4259 0.0084  0.1107 0.7411   True
    48     96   0.4397 0.0066  0.1245 0.7549   True
    72     96   0.0138 0.9929 -0.3014  0.329  False
---------------------------------------------------
```

And now that we know how it differs we can use the CLD method to present it graphically in another plot type:

```python
plt.figure(figsize=(18, 7))
sns.barplot(data=df_para_plots1, x='cepa', y='log_yield', hue='hora_medicion', errorbar="se",
            capsize=0.1, errwidth=1.5,)
plt.text(7.25, 8.7, 'b', fontsize=16, ha='center')
plt.text(7, 8.7, 'b', fontsize=16, ha='center')
plt.text(6.72, 8.7, 'a', fontsize=16, ha='center')
plt.xlabel('Aislamiento')
plt.ylabel('Log(10) de produccion de blastosporas (bsps/mL)')
plt.xticks(rotation=0, ha='center')
plt.yticks(ticks=(1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9))
plt.ylim(6, 9)
plt.legend(title='Tiempo de medicion (hs)',
           #bbox_to_anchor=(1.05, 1),
           loc='upper left',)
plt.grid(True, linestyle='-', linewidth=.2, ydata=None)
nombres_cepas = ['CEP{}'.format(num) for num in df_para_plots1['cepa'].unique()]
plt.gca().set_xticklabels(["CEP085", "CEP120", "CEP160", "CEP350", "CEP416",
                           "CEP424", "CEP593", "CEP712", "CEP823", "CEP824"])
plt.tight_layout()
plt.show()
```



You can use “lifelines” library to do some survival analysys:

```
from lifelines import KaplanMeierFitter
from lifelines import WeibullFitter
from lifelines.statistics import multivariate_logrank_test
from lifelines.statistics import logrank_test
from lifelines.statistics import pairwise_logrank_test
```

When your data is structured like this:

```
df.head()
```

	tiempo	status	sexo	replica	cepa	fecha
0	12	1	m	1.0	CEP085	2023-11-21
1	12	1	f	1.0	CEP085	2023-11-21
2	14	1	m	1.0	CEP085	2023-11-21
3	14	1	f	1.0	CEP085	2023-11-21
4	14	1	f	1.0	CEP085	2023-11-21

You can make your own color palette!

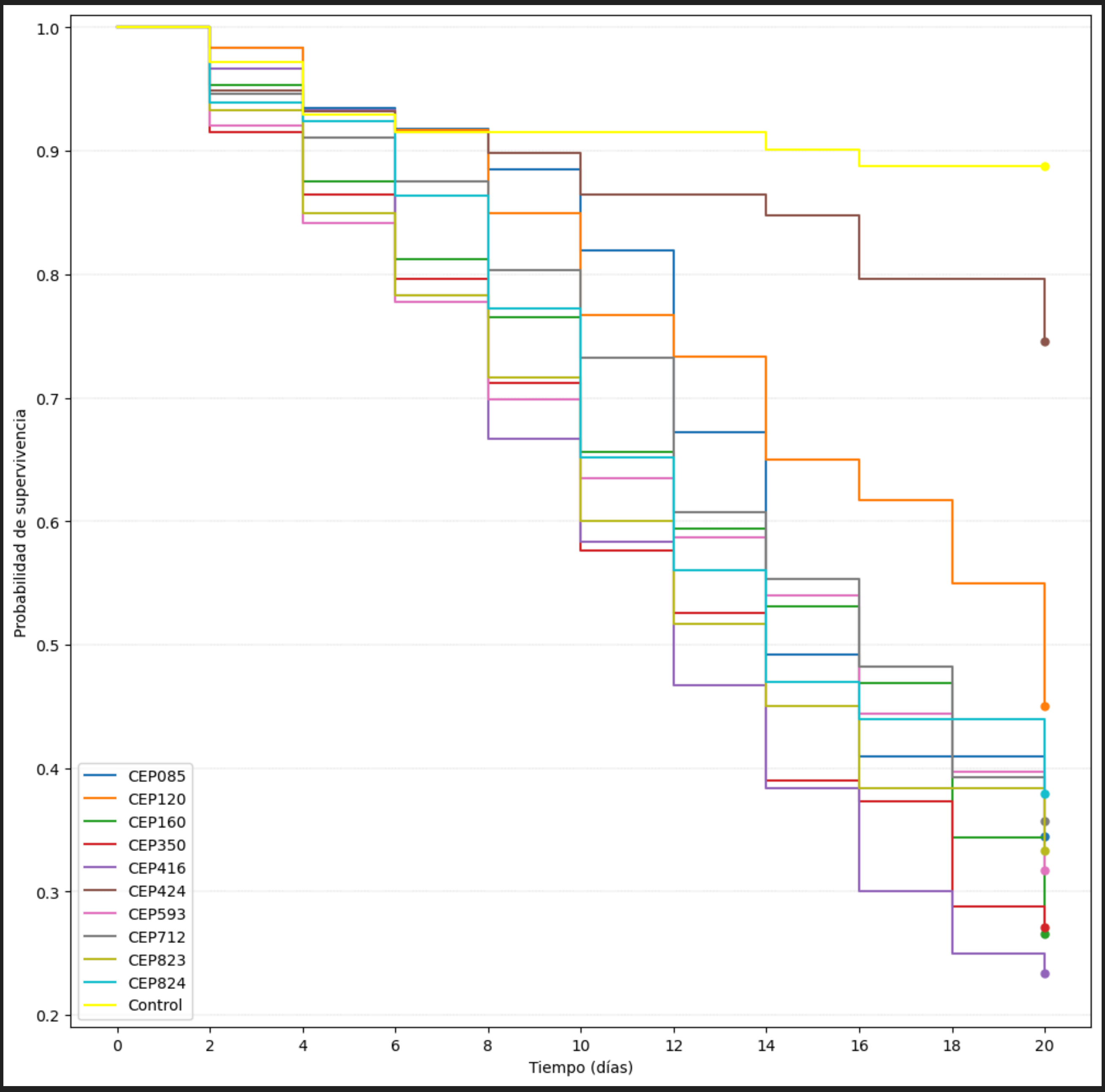
```
paleta = sns.color_palette( n_colors=11)
amarillo=(1,1,0)
paleta[-1] = amarillo
paleta
```

And fit the data to Kaplan-Meier curves:

```
kmf = KaplanMeierFitter()
contador=0
#Creo un grafico de supervivencia para cada tratamiento con un ciclo for
for cepa in df['cepa'].unique():#Iterar sobre cada cepa única en el dataframe
    cepa_data = df[df['cepa'] == cepa]#Filtrar el dataframe por la cepa
    kmf.fit(cepa_data['tiempo'], event_observed=cepa_data['status'], label=f'{cepa}')#Ajustar el modelo Kaplan-Meier para la cepo filtrada
    kmf.plot_survival_function(color=paleta[contador],
        show_censors=True, censor_styles={'ms': 5, 'marker': 'o'},
        #at_risk_counts=True,
        ci_alpha=0)#Graficar la curva de supervivencia sin intervalo de confianza visible

    #Obtener TL50
    tl50_time = kmf.median_survival_time_# Obtener el TL50
    ci_1 = kmf.confidence_interval_survival_function_
    print(f'TL50 de Cep{cepa}: {tl50_time} días')# Imprimir el TL50
    print(kmf.survival_function_)
    print(ci_1)
    contador += 1

plt.xlabel('Tiempo (días)')
plt.ylabel('Probabilidad de supervivencia')
plt.legend(loc='best')
plt.ylim([0.19, 1.01])
plt.yticks([.2,.3,.4,.5,.6,.7,.8,.9,1])
plt.xticks(range(0, int(max(df['tiempo']))+1, 2))#Mostrar el tiempo en el eje x de 2 en 2
plt.grid(True, linestyle='dashed', linewidth=.2, ydata=None)
plt.show()
```



You can also compare these curves pairwise using the log-rank test to see if they are significantly different:

```
#LOGRANK DE A PARES
resultados = pairwise_logrank_test(df['tiempo'], df['cepa'], df['status'],t_0=20)
resultados
```

t_0	20
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

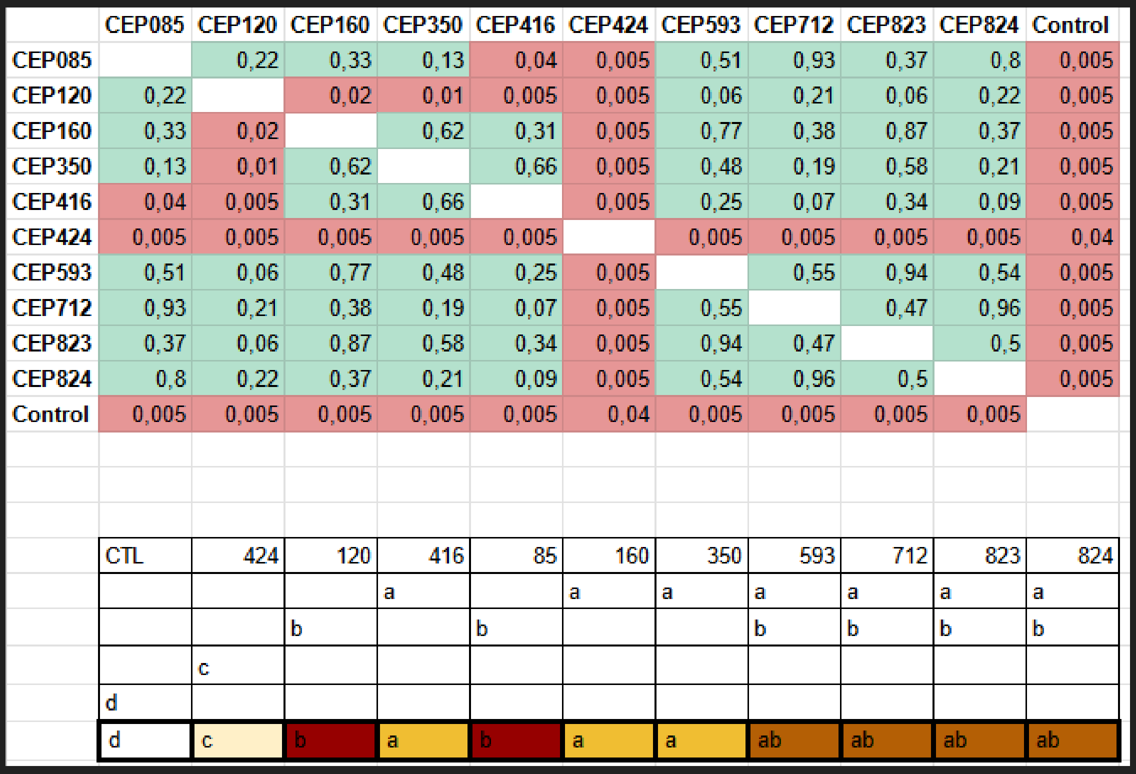
		test_statistic	**p**	**-log2(p)**
CEP085	**CEP120**	1.47	0.22	2.15
	CEP160	0.96	0.33	1.61
	CEP350	2.29	0.13	2.94
	CEP416	4.06	0.04	4.51
	CEP424	18.30	<0.005	15.69
	CEP593	0.44	0.51	0.98
	CEP712	0.01	0.93	0.10
	CEP823	0.81	0.37	1.44
	CEP824	0.06	0.80	0.32
	Control	39.30	<0.005	31.36
CEP120	**CEP160**	5.05	0.02	5.34
	CEP350	6.53	0.01	6.56

And use the Benjamini-Hochberg method for false discovery rate:

```python
rejected, p_adjusted = fdrcorrection(p_values, alpha=0.05, method='indep', is_sorted=False)
#Convertir los resultados a un dataframe
adjusted_p_values_df = pd.DataFrame({
    "name" : names,
    'p_valor_original': p_values,
    'p_valor_ajustado': p_adjusted,
    'rechazar_hipotesis_nula': rejected
})
adjusted_p_values_df
```

```
              name  p_valor_original  p_valor_ajustado  rechazar_hipotesis_nula
0  (CEP085, CEP120)      2.248358e-01      3.747263e-01                    False
1  (CEP085, CEP160)      3.265381e-01      4.988776e-01                    False
2  (CEP085, CEP350)      1.300333e-01      2.554225e-01                    False
3  (CEP085, CEP416)      4.400920e-02      1.052394e-01                    False
4  (CEP085, CEP424)      1.888368e-05      6.924016e-05                     True
5  (CEP085, CEP593)      5.066439e-01      6.333049e-01                    False
6  (CEP085, CEP712)      9.312107e-01      9.549439e-01                    False
7  (CEP085, CEP823)      3.692478e-01      5.192231e-01                    False
```

Conditional formatting can be applied to the p-values to display them clearly, and the same CLD method can be applied to show significant differences between survival functions:



	CEP085	CEP120	CEP160	CEP350	CEP416	CEP424	CEP593	CEP712	CEP823	CEP824	Control
CEP085		0,22	0,33	0,13	0,04	0,005	0,51	0,93	0,37	0,8	0,005
CEP120	0,22		0,02	0,01	0,005	0,005	0,06	0,21	0,06	0,22	0,005
CEP160	0,33	0,02		0,62	0,31	0,005	0,77	0,38	0,87	0,37	0,005
CEP350	0,13	0,01	0,62		0,66	0,005	0,48	0,19	0,58	0,21	0,005
CEP416	0,04	0,005	0,31	0,66		0,005	0,25	0,07	0,34	0,09	0,005
CEP424	0,005	0,005	0,005	0,005	0,005		0,005	0,005	0,005	0,005	0,04
CEP593	0,51	0,06	0,77	0,48	0,25	0,005		0,55	0,94	0,54	0,005
CEP712	0,93	0,21	0,38	0,19	0,07	0,005	0,55		0,47	0,96	0,005
CEP823	0,37	0,06	0,87	0,58	0,34	0,005	0,94	0,47		0,5	0,005
CEP824	0,8	0,22	0,37	0,21	0,09	0,005	0,54	0,96	0,5		0,005
Control	0,005	0,005	0,005	0,005	0,005	0,04	0,005	0,005	0,005	0,005	

CTL	424	120	416	85	160	350	593	712	823	824
			a		a	a	a	a	a	a
		b		b			b	b	b	b
	c									
d										
d	c	b	a	b	a	a	ab	ab	ab	ab