Proplyd distribution

September 19, 2014

```
In [1]: %matplotlib inline
```

Calculations of distributions of proplyd distances and sizes and angles. These translate into distributions of β.

0.1 Data sources

We have the list of proplyds from Henney & Arthur (1998) that we used in Fig 12 of Tsamis et al (2013). This has distances and sizes.

We also have the Ricci et al (2008) catalog, which has distances but not sizes.

```
In [308]: from astropy.table import Table
          import numpy as np
          import seaborn as sns
          from scipy import stats
```

Now use seaborn to get nicer graphs.

```
In [359]: sns.set_palette("deep", desat=0.6)
          sns.set_context("talk", rc={"figure.figsize": (12, 8)})
```

0.1.1 Henney & Arthur (1998) sample

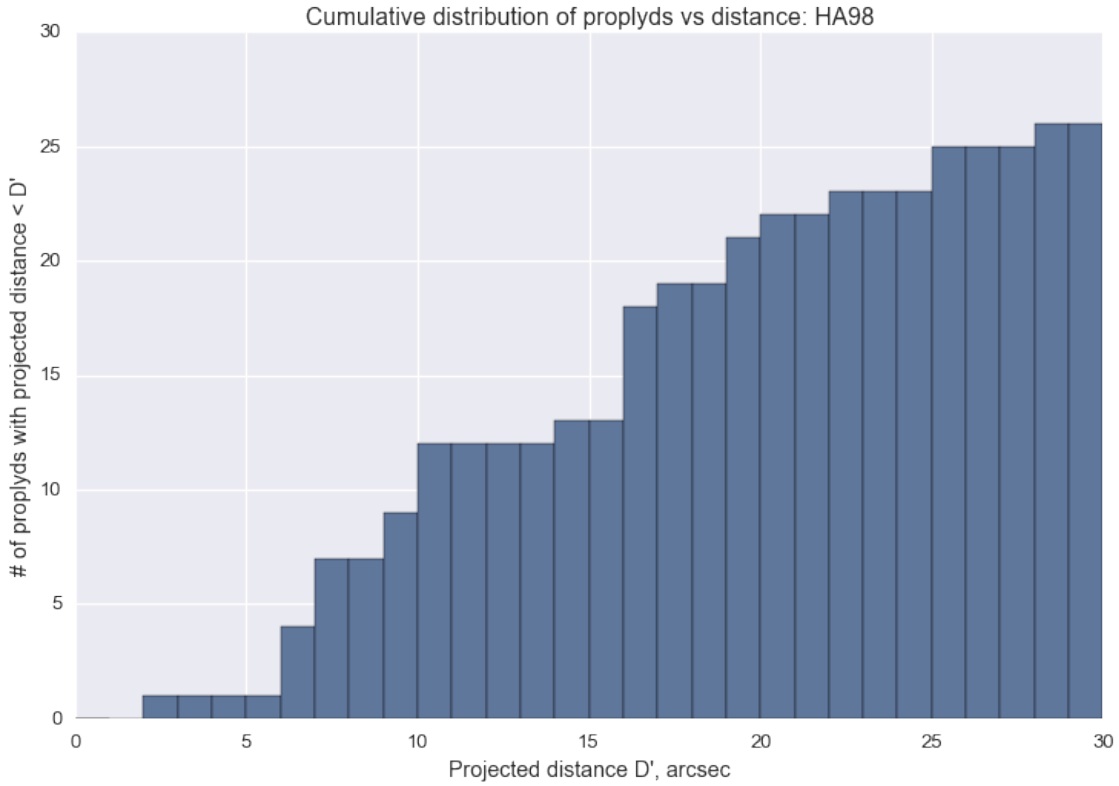
```
In [3]: datafile = "Proplyd-Datasets/vicente-vs-HA98.dat"
        data = Table.read(datafile, format='ascii')
```

```
In [4]: data
```

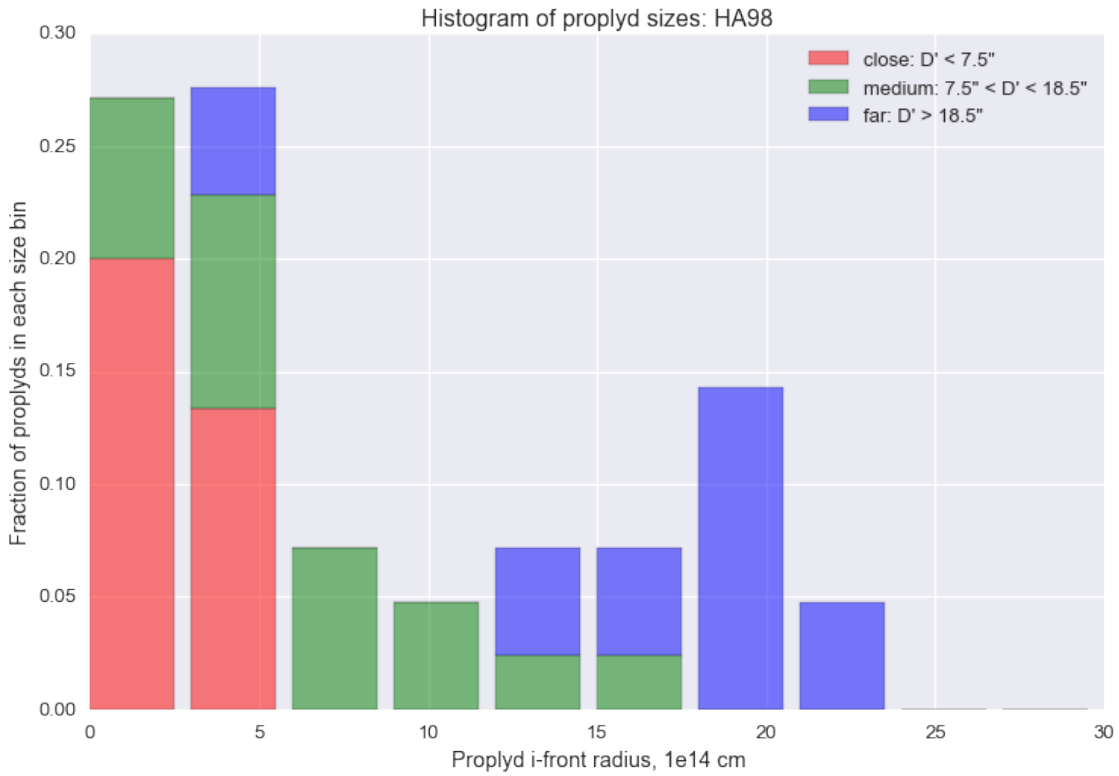
```
Out[4]: <Table rows=28 names=('ID','d','r14','VicR14')>
        array([('163-323', 2.14, 2.2, 19.4), ('161-324', 6.05, 3.5, 25.8),
               ('168-328', 6.64, 2.8, 21.0), ('163-317', 6.91, 5.0, 27.1),
               ('166-316', 7.01, 2.5, 14.8), ('161-328', 7.74, 9.1, 20.0),
               ('167-317', 7.83, 7.9, 35.8), ('158-323', 9.42, 6.3, 30.6),
               ('158-326', 9.6, 11.3, 31.6), ('161-314', 10.24, 5.3, 17.1),
               ('158-327', 10.6, 16.6, 35.5), ('157-323', 10.97, 2.5, 18.4),
               ('171-334', 14.29, 4.7, 22.9), ('170-337', 16.2, 12.2, 36.8),
               ('176-325', 16.38, 6.9, 23.5), ('169-338', 16.47, 2.8, 16.1),
               ('154-324', 16.63, 3.2, 10.3), ('153-321', 16.97, 1.2, 8.4),
               ('159-338', 17.2, 5.0, 13.5), ('171-340', 19.11, 23.3, 23.2),
               ('152-319', 19.16, 18.2, 14.8), ('155-338', 20.48, 17.0, 29.7),
               ('173-341', 22.48, 4.1, 18.1), ('180-331', 25.12, 12.2, 29.3),
               ('177-341', 25.84, 20.4, 51.6), ('159-350', 28.35, 20.1, 44.5),
               ('182-413', 56.7, 36.9, 48.7), ('244-440', 142.3, 104.0, 180.6)],
              dtype=[('ID', '<U7'), ('d', '<f8'), ('r14', '<f8'), ('VicR14', '<f8')])
```

```
In [5]: from matplotlib import pyplot as plt
```

```
In [360]: _ = plt.hist(data["d"], bins=30, range=(0, 30), cumulative=True)
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of proplyds with projected distance < D\'')
          _ = plt.title('Cumulative distribution of proplyds vs distance: HA98')
```

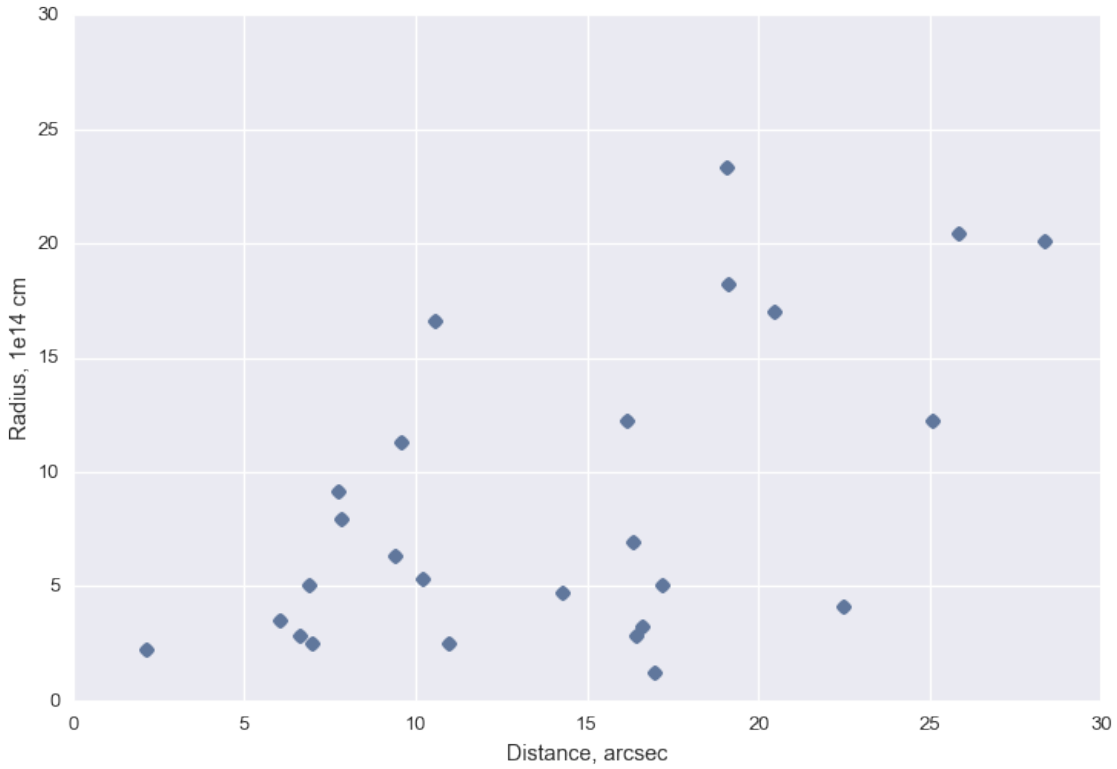



```
In [361]: m1 = data["d"] <= 7.5
          m2 = np.abs(data["d"] - 13.0) <= 5.5
          m3 = np.abs(data["d"] - 25.0) <= 6.5
          h1, edges = np.histogram(data["r14"][m1], bins=10, range=(0, 30), density=True)
          h2, edges = np.histogram(data["r14"][m2], bins=10, range=(0, 30), density=True)
          h3, edges = np.histogram(data["r14"][m3], bins=10, range=(0, 30), density=True)
          plt.bar(edges[:-1], h1, 2.5, color='r', alpha=0.5, label='close: D\' < 7.5"')
          plt.bar(edges[:-1], h2, 2.5, h1, color='g', alpha=0.5, label='medium: 7.5" < D\' < 18.5"')
          plt.bar(edges[:-1], h3, 2.5, h1 + h2, color='b', alpha=0.5, label='far: D\' > 18.5"')
          _ = plt.xlabel('Proplyd i-front radius, 1e14 cm')
          _ = plt.ylabel('Fraction of proplyds in each size bin')
          _ = plt.title('Histogram of proplyd sizes: HA98')
          _ = plt.legend(loc='upper right')
```




```
In [362]: _ = plt.plot(data["d"], data["r14"], 'o')
          _ = plt.xlim(0.0, 30.0)
          _ = plt.ylim(0.0, 30.0)
          plt.xlabel("Distance, arcsec")
          plt.ylabel("Radius, 1e14 cm")
```

```
Out[362]: <matplotlib.text.Text at 0x119f77710>
```



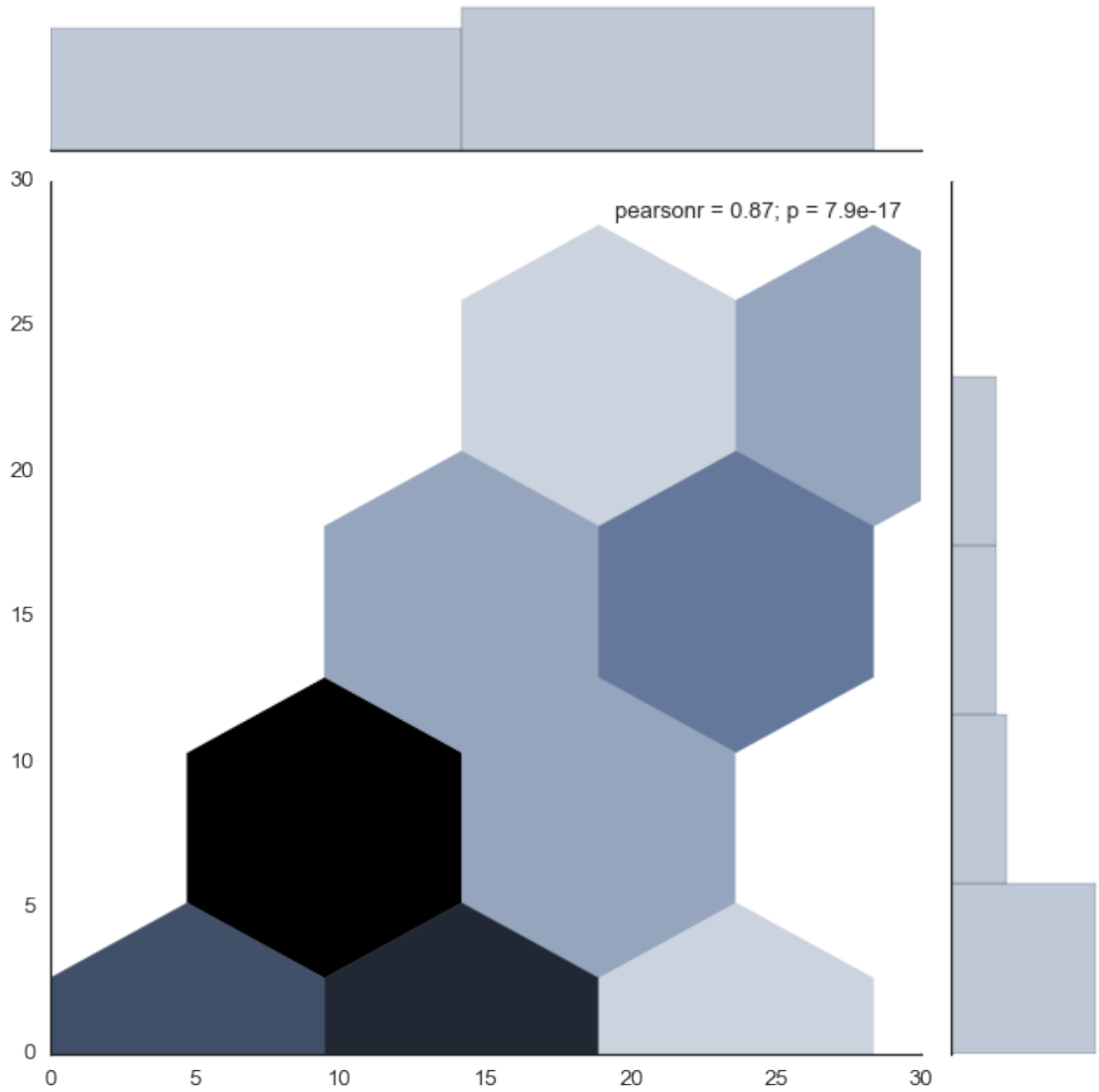
We need to take account of binaries

Now we will try out some of the seaborn tools for visualizing data distributions. See this tutorial.

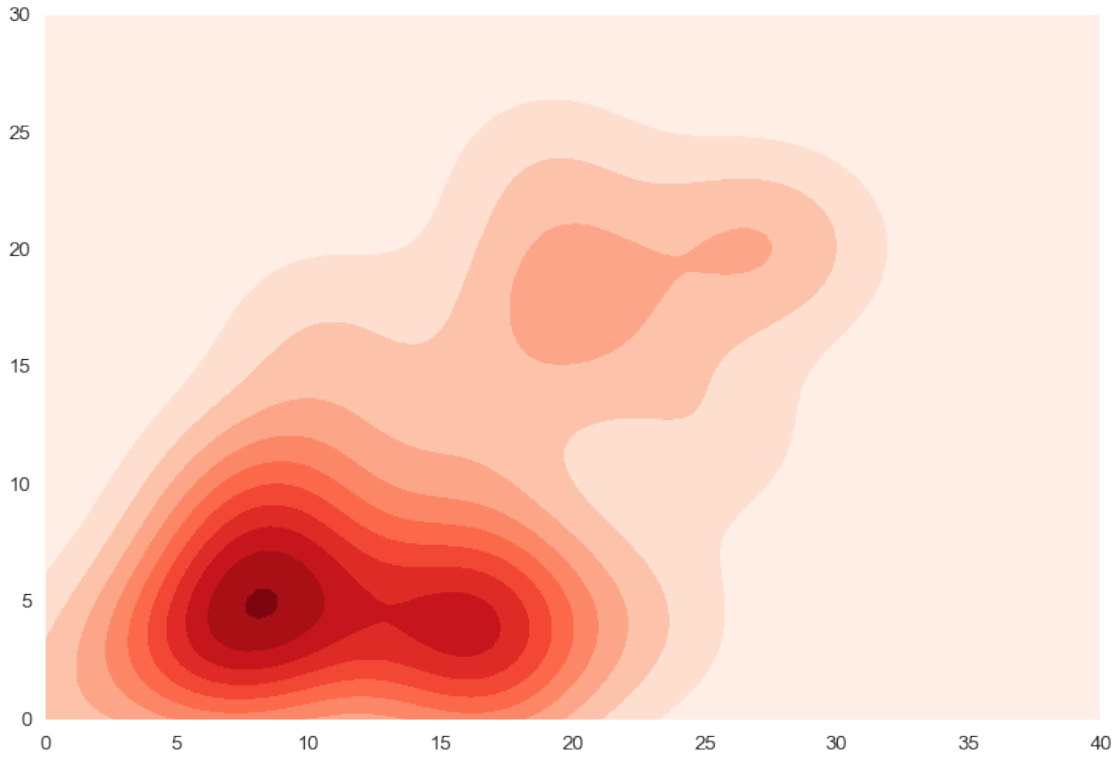
In the section on the Ricci data it occured to me that we need to mirror the data about zero because the radius and separation are positive-definite.

```
In [926]: def mirrored(dataset):
              """Expand a positive-definite dataset (1d sequence) to include its negative mirror set"""
              assert np.all(dataset >= 0.0)
              return np.r_[dataset, -dataset]

          def doubled(dataset):
              """Double up a dataset (1d sequence)"""
              return np.r_[dataset, dataset]
```

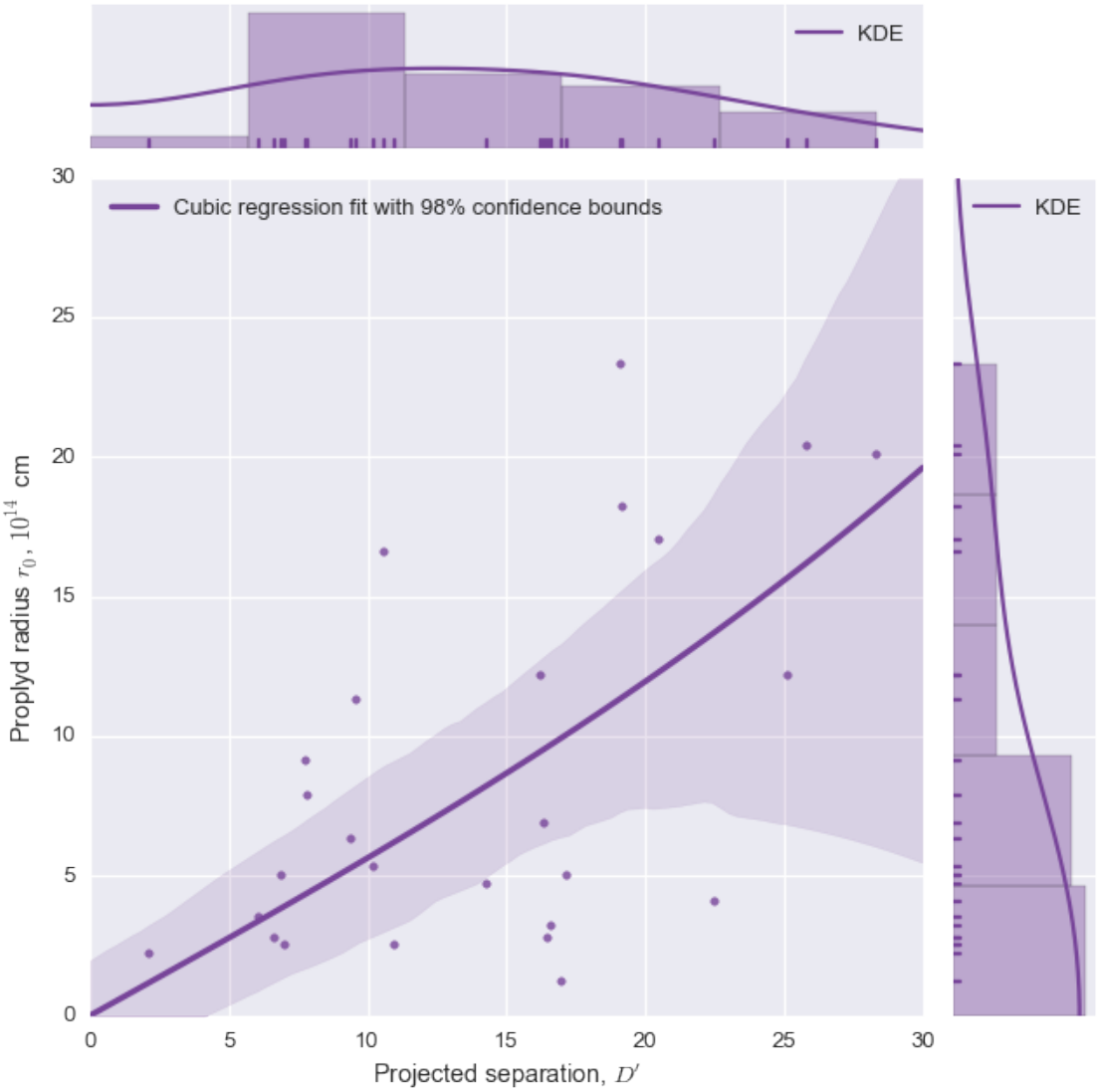
```
In [425]: m30 = data["d"] <= 30.0
          with sns.axes_style("white"):
              sns.jointplot(mirrored(data["d"][m30]), mirrored(data["r14"][m30]),
                            kind="hex", size=9, xlim=(0, 30), ylim=(0, 30))
```




```
In [446]: sns.kdeplot(mirrored(data["d"]), mirrored(data["r14"]),
                      clip=((0, 40), (0, 30)),
                      bw=3.0, shade=True, cmap="Reds",
                      )
          None
```



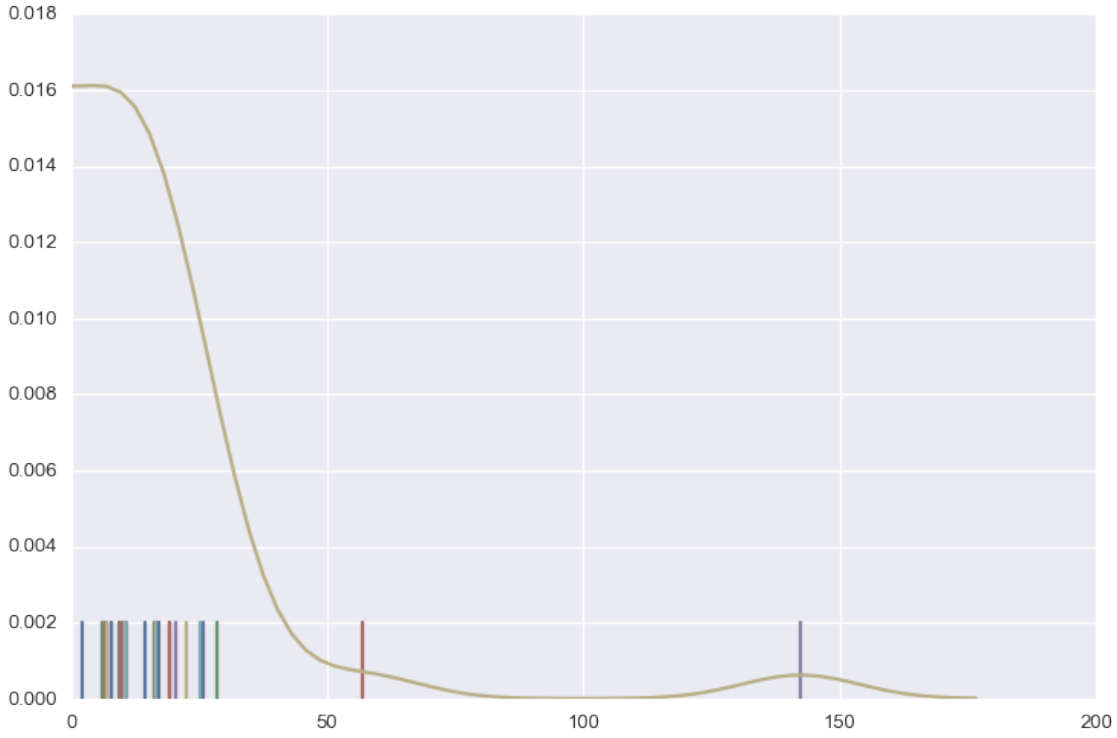

```
In [503]: color = "#774499"
          g = sns.JointGrid(mirrored(data["d"][m30]), mirrored(data["r14"][m30]), size=9, xlim=(0, 30),
          # g.plot(sns.regplot, sns.distplot, stats.pearsonr);
          g.plot_marginals(sns.distplot, color=color, bins=10, rug=True, kde_kws=dict(bw=5.0, label="KD
          g.plot_joint(sns.regplot, color=color, ci=98, order=3, label="Cubic regression fit with 98% c
          g.ax_joint.legend(loc="upper left")
          g.ax_joint.set_xlabel("Projected separation, $D'$")
          g.ax_joint.set_ylabel("Proplyd radius $r_0$, $10^{14}$ cm")
          g.ax_marg_x.legend(loc="upper right")
          g.ax_marg_y.legend(loc="best")
          g.fig.tight_layout()
          g.fig.savefig("proplyd-joint-separation-size.pdf");
          #g.annotate(stats.pearsonr);
```




```
In [494]: stats.scoreatpercentile??
```

```
In [401]: sns.rugplot(data["d"], height=0.002)
          sns.kdeplot(mirrored(data["d"]), shade=True)
          plt.xlim(0.0, None)
```

```
Out[401]: (0.0, 200.0)
```

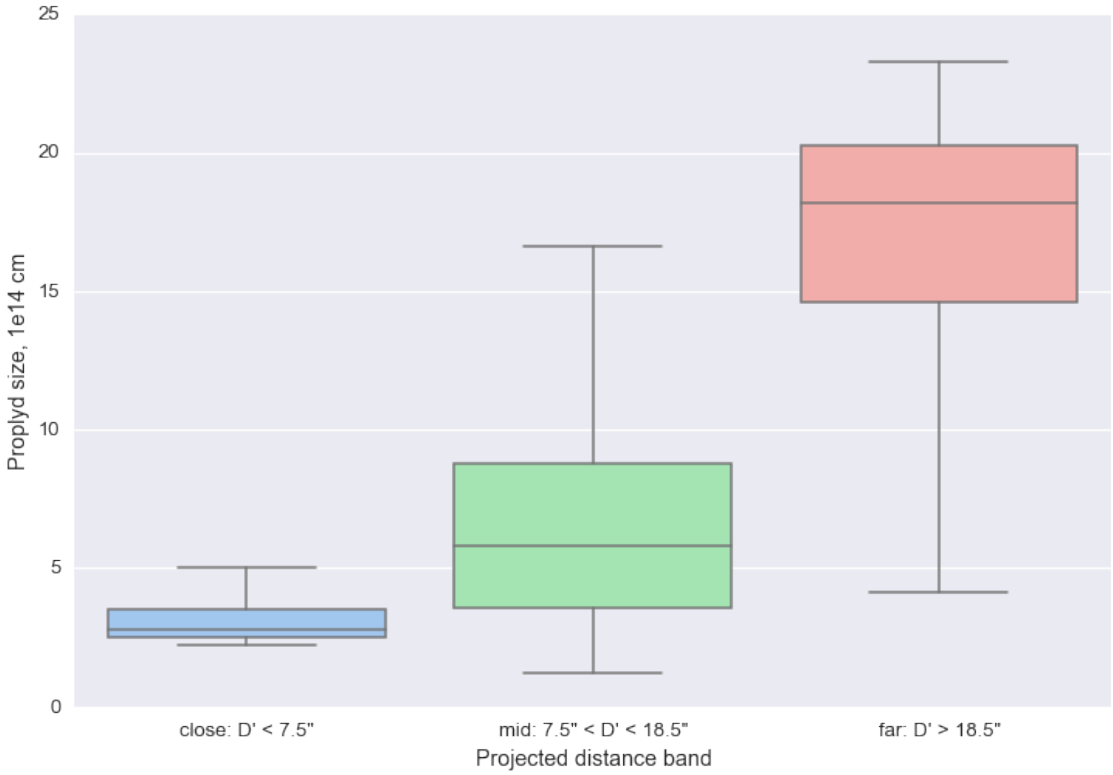


Now we will compare the sizes for different intervals of D'

```
In [407]: datalist = [data['r14'][m1], data['r14'][m2], data['r14'][m3]]
          mirrored_datalist = [mirrored(x) for x in datalist]
          namelist = ['close: D\' < 7.5"', 'mid: 7.5" < D\' < 18.5"', 'far: D\' > 18.5"']
```

The simplest option is `boxplot`, which shows the median and quartiles. By setting `whis=np.inf` we make the whiskers extend out to the full range of the data.

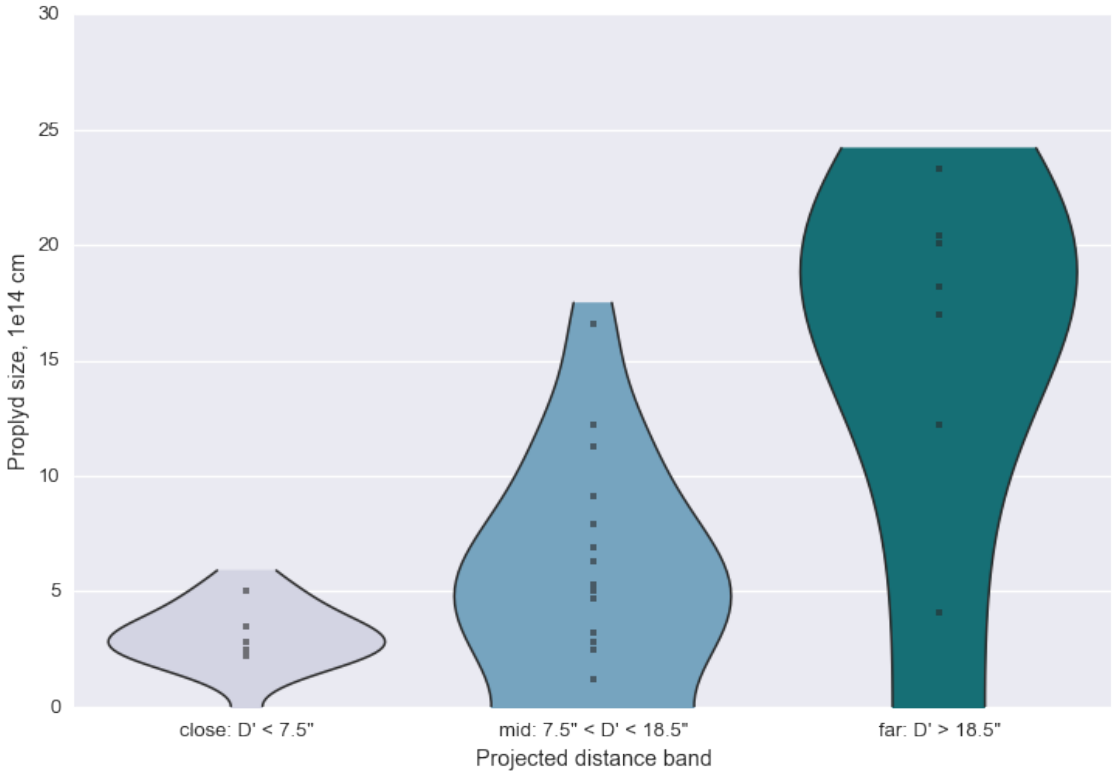
```
In [408]: sns.boxplot(datalist, names=namelist, whis=np.inf, color='pastel')
          plt.ylabel("Proplyd size, 1e14 cm")
          plt.xlabel("Projected distance band")
          None
```



Alternatively we can use `violinplot`, which shows the Kernel Density Estimate (KDE) of the values in each category. By setting `inner="points"` it shows the actual data points instead of the quartiles.

I am a little uncomfortable of this way of presenting the data, since we are forcing the distance to be categorical, when really it is continuous.

```
In [421]: sns.violinplot(mirrored_datalist,
                         names=namelist,
                         inner="points", bw=0.3, color="PuBuGn")
          plt.ylim(0.0, None)
          plt.ylabel("Proplyd size, 1e14 cm")
          plt.xlabel("Projected distance band")
          None
```



In [422]: sns.jointplot?

0.1.2 Ricci et al (2008) sample

In [9]:
```
rdata = Table.read("Proplyd-Datasets/ricci-2008.dat", format='ascii')
rdata.remove_row(0) # First row is rubbish
rdata
```

Out[9]:
```
<Table rows=219 names=('Name','RAJ2000','DEJ2000','[OW94]','Type','Note')>
masked_array(data = [('4364-146', '05 34 36.44', '-05 21 45.95', '0', 'j', 'J')
 ('4466-324', '05 34 46.59', '-05 23 24.19', '0', 'j', 'J,B')
 ('4468-605', '05 34 46.76', '-05 26 04.79', '0', 'i', 'J')
 ('4538-311', '05 34 53.79', '-05 23 10.73', '0', 'rn', '0')
 ('4596-400', '05 34 59.56', '-05 24 00.19', '4596-400', 'i', '0')
 ('4582-635', '05 34 58.16', '-05 26 35.13', '0', 'i', '0')
 ('005-514', '05 35 00.47', '-05 25 14.34', '005-514', 'i', '0')
 ('006-439', '05 35 00.58', '-05 24 38.79', '0', 'j', 'J')
 ('016-149', '05 35 01.60', '-05 21 49.35', '0', 'rn', '0')
 ('038-627', '05 35 04.19', '-05 26 27.89', '038-627', 'i', '0')
 ('044-527', '05 35 04.42', '-05 25 27.40', '044-527', 'i', '0')
 ('046-3838', '05 35 04.61', '-05 38 38.00', '0', 'rn', '0')
 ('046-245', '05 35 04.63', '-05 22 44.85', '0', 'i', '0')
 ('049-143', '05 35 04.94', '-05 21 42.99', '0', 'i', '0')
 ('051-3541', '05 35 05.05', '-05 35 40.84', '0', 'rn', 'EO')
 ('053-717', '05 35 05.40', '-05 27 16.99', '0', 'd', '0')
 ('057-419', '05 35 05.73', '-05 24 18.55', '057-419', 'i', '0')
```

```
('061-401', '05 35 06.09', '-05 24 00.60', '061-401', 'i', '0')
('064-3335', '05 35 06.44', '-05 33 35.25', '0', 'i', '0')
('066-3251', '05 35 06.57', '-05 32 51.49', '0', 'i', '0')
('066-652', '05 35 06.59', '-05 26 51.99', '066-652', 'i', 'B')
('069-601', '05 35 06.91', '-05 26 00.60', '069-601', 'i', '0')
('072-135', '05 35 07.21', '-05 21 34.43', '072-135', 'i', 'EO')
('073-227', '05 35 07.27', '-05 22 26.56', '073-227', 'i', '0')
('078-3658', '05 35 07.84', '-05 36 58.15', '0', 'j', 'J,B')
('090-326', '05 35 09.02', '-05 23 26.20', '0', 'd', '0')
('093-822', '05 35 09.59', '-05 28 22.92', '093-822', 'i', '0')
('099-339', '05 35 09.89', '-05 23 38.50', '0', 'i', 'J')
('102-233', '05 35 10.13', '-05 22 32.74', '102-233', 'i', '0')
('102-021', '05 35 10.19', '-05 20 20.99', '102-021', 'i', '0')
('102-322', '05 35 10.20', '-05 23 21.56', '102-322', 'i', '0')
('106-417', '05 35 10.54', '-05 24 16.70', '106-417', 'i', '0')
('106-156', '05 35 10.58', '-05 21 56.24', '106-156', 'i', '0')
('109-246', '05 35 10.90', '-05 22 46.36', '109-246', 'i', '0')
('109-327', '05 35 10.95', '-05 23 26.45', '109-327', 'i', 'J')
('110-3035', '05 35 10.98', '-05 30 35.23', '0', 'rn', 'J')
('114-426', '05 35 11.30', '-05 24 26.50', '114-426', 'd', 'EO,RN')
('117-421', '05 35 11.65', '-05 24 21.50', '117-421', 'i', '0')
('117-352', '05 35 11.73', '-05 23 51.70', '0', 'i', '0')
('119-340', '05 35 11.90', '-05 23 39.70', '119-340', 'i', '0')
('121-1925', '05 35 12.09', '-05 19 24.80', '121-1925', 'd', '0')
('121-434', '05 35 12.12', '-05 24 33.80', '121-434', 'i', '0')
('124-132', '05 35 12.38', '-05 21 31.39', '124-132', 'i', 'J,EO,B')
('131-046', '05 35 13.05', '-05 20 45.79', '131-046', 'i', '0')
('131-247', '05 35 13.11', '-05 22 47.11', '131-247', 'i', 'J')
('132-1832', '05 35 13.24', '-05 18 32.95', '0', 'd', 'EO')
('132-042', '05 35 13.24', '-05 20 41.94', '132-042', 'i', 'J,EO')
('133-353', '05 35 13.30', '-05 23 53.05', '133-353', 'i', 'B')
('135-220', '05 35 13.51', '-05 22 19.49', '135-220', 'i', '0')
('138-207', '05 35 13.78', '-05 22 07.39', '138-207', 'i', '0')
('139-320', '05 35 13.92', '-05 23 20.16', '0', 'i', '0')
('141-520', '05 35 14.05', '-05 25 20.50', '141-520', 'i', 'FO')
('140-1952', '05 35 14.05', '-05 19 51.90', '140-1952', 'd', 'FO')
('142-301', '05 35 14.15', '-05 23 00.91', '142-301', 'i', '0')
('143-425', '05 35 14.27', '-05 24 24.55', '143-425', 'i', '0')
('144-522', '05 35 14.34', '-05 25 22.30', '144-522', 'i', '0')
('146-201', '05 35 14.61', '-05 22 00.94', '0', 'i', '0')
('147-323', '05 35 14.72', '-05 23 23.01', '0', 'i', '0')
('148-305', '05 35 14.80', '-05 23 04.76', '148-305', 'i', 'B')
('149-329', '05 35 14.92', '-05 23 29.05', '149-329', 'i', '0')
('150-147', '05 35 15.00', '-05 21 47.34', '0', 'i', '0')
('150-231', '05 35 15.02', '-05 22 31.11', '150-231', 'i', 'B')
('152-319', '05 35 15.20', '-05 23 18.81', '152-319', 'i', '0')
('152-738', '05 35 15.21', '-05 27 37.85', '152-738', 'i', '0')
('154-324', '05 35 15.35', '-05 23 24.11', '154-324', 'i', 'B')
('154-225', '05 35 15.37', '-05 22 25.35', '154-225', 'i', 'B')
('154-240', '05 35 15.38', '-05 22 39.85', '0', 'i', '0')
('155-338', '05 35 15.51', '-05 23 37.45', '155-338', 'i', '0')
('157-323', '05 35 15.72', '-05 23 22.59', '157-323', 'i', 'B')
('158-323', '05 35 15.83', '-05 23 22.59', '158-323', 'i', 'B')
('158-327', '05 35 15.79', '-05 23 26.51', '158-327', 'i', 'B')
```

```
('158-326', '05 35 15.81', '-05 23 25.51', '158-326', 'i', 'B')
('159-338', '05 35 15.90', '-05 23 38.00', '159-338', 'i', 'B')
('159-418', '05 35 15.90', '-05 24 17.85', '159-418', 'i', 'O')
('159-221', '05 35 15.93', '-05 22 21.05', '159-221', 'd', 'O')
('159-350', '05 35 15.96', '-05 23 50.30', '159-350', 'i', 'B')
('160-353', '05 35 16.01', '-05 23 53.00', '160-353', 'i', 'J')
('161-324', '05 35 16.05', '-05 23 24.35', '161-324', 'i', 'O')
('161-328', '05 35 16.07', '-05 23 27.81', '161-328', 'i', 'O')
('161-314', '05 35 16.10', '-05 23 14.05', '161-314', 'i', 'O')
('162-133', '05 35 16.19', '-05 21 32.39', '162-133', 'i', 'O')
('163-026', '05 35 16.31', '-05 20 25.24', 'O', 'd', 'B')
('163-210', '05 35 16.27', '-05 22 10.45', '163-210', 'i', 'B')
('163-317', '05 35 16.27', '-05 23 16.51', '163-317', 'i', 'O')
('163-222', '05 35 16.30', '-05 22 21.50', '163-222', 'i', 'B,FO')
('163-249', '05 35 16.33', '-05 22 49.01', '163-249', 'i', 'B')
('164-511', '05 35 16.35', '-05 25 09.60', '164-511', 'i', 'O')
('165-235', '05 35 16.48', '-05 22 35.16', '165-235', 'i', 'O')
('165-254', '05 35 16.54', '-05 22 53.70', 'O', 'd', 'RN')
('166-316', '05 35 16.61', '-05 23 16.19', '166-315', 'i', 'O')
('166-519', '05 35 16.57', '-05 25 17.74', '166-519', 'd', 'O')
('166-406', '05 35 16.57', '-05 24 06.00', '166-406', 'i', 'O')
('166-250', '05 35 16.59', '-05 22 50.36', '166-250', 'i', 'O')
('167-231', '05 35 16.73', '-05 22 31.30', '167-231', 'd', 'FO')
('167-317', '05 35 16.74', '-05 23 16.51', '167-317', 'i', 'O')
('168-235', '05 35 16.81', '-05 22 34.71', '168-235', 'i', 'O')
('168-328', '05 35 16.77', '-05 23 28.06', '168-328', 'i', 'B')
('168-326', '05 35 16.83', '-05 23 25.91', '168-326', 'i', 'B')
('169-338', '05 35 16.88', '-05 23 38.10', '169-338', 'i', 'B')
('170-301', '05 35 16.95', '-05 23 00.91', '170-301', 'i', 'O')
('170-249', '05 35 16.96', '-05 22 48.51', '170-249', 'i', 'B')
('170-337', '05 35 16.97', '-05 23 37.15', '170-337', 'i', 'B')
('171-340', '05 35 17.04', '-05 23 39.75', '171-340', 'i', 'B')
('171-434', '05 35 17.11', '-05 24 34.40', '171-434', 'i', 'B')
('172-028', '05 35 17.22', '-05 20 27.84', '172-028', 'd', 'O')
('173-341', '05 35 17.32', '-05 23 41.40', '173-341', 'i', 'B')
('173-236', '05 35 17.34', '-05 22 35.81', '173-236', 'i', 'O')
('174-305', '05 35 17.37', '-05 23 04.86', '174-305', 'i', 'O')
('174-414', '05 35 17.38', '-05 24 13.90', '174-414', 'i', 'O')
('175-251', '05 35 17.47', '-05 22 51.26', '175-251', 'i', 'B')
('175-355', '05 35 17.54', '-05 23 55.05', '175-355', 'i', 'O')
('175-543', '05 35 17.54', '-05 25 42.89', '175-543', 'i', 'J,EO,B')
('176-325', '05 35 17.55', '-05 23 24.96', '176-325', 'i', 'O')
('176-252', '05 35 17.64', '-05 22 51.66', '176-252', 'i', 'O')
('177-341W', '05 35 17.66', '-05 23 41.00', '177-341', 'i', 'B')
('177-454', '05 35 17.69', '-05 24 54.10', '177-454', 'i', 'O')
('177-541', '05 35 17.71', '-05 25 40.76', 'O', 'i', 'EO,B')
('177-444', '05 35 17.73', '-05 24 43.75', '177-444', 'i', 'O')
('177-341E', '05 35 17.73', '-05 23 41.10', '177-341', 'i', 'B')
('178-441', '05 35 17.81', '-05 24 41.05', 'O', 'i', 'O')
('178-258', '05 35 17.84', '-05 22 58.15', 'O', 'i', 'O')
('179-056', '05 35 17.92', '-05 20 55.44', '179-056', 'i', 'B')
('179-354', '05 35 17.96', '-05 23 53.50', '179-354', 'i', 'O')
('180-331', '05 35 18.03', '-05 23 30.80', '180-331', 'i', 'B')
('181-247', '05 35 18.08', '-05 22 47.10', '181-247', 'i', 'B')
```

```
('181-825', '05 35 18.10', '-05 28 25.04', '0', 'i', 'RN')
('182-316', '05 35 18.19', '-05 23 31.55', '182-316', 'd', 'B')
('182-413', '05 35 18.22', '-05 24 13.45', '182-413', 'i', '0')
('183-439', '05 35 18.28', '-05 24 38.85', '183-439', 'i', 'B')
('183-419', '05 35 18.31', '-05 24 18.85', '183-419', 'i', '0')
('183-405', '05 35 18.33', '-05 24 04.85', '183-405', 'd', 'FO,B')
('184-427', '05 35 18.35', '-05 24 26.85', '184-427', 'i', 'B')
('184-520', '05 35 18.44', '-05 25 19.29', '184-520', 'i', '0')
('187-314', '05 35 18.68', '-05 23 14.01', '187-314', 'i', 'B')
('189-329', '05 35 18.87', '-05 23 28.85', '189-329', 'i', '0')
('190-251', '05 35 19.03', '-05 22 50.65', '0', 'i', 'B')
('191-232', '05 35 19.13', '-05 22 31.20', '0', 'd', '0')
('191-350', '05 35 19.06', '-05 23 49.50', '191-350', 'i', 'J')
('193-1659', '05 35 19.25', '-05 16 58.69', '0', 'rn', '0')
('197-427', '05 35 19.65', '-05 24 26.70', '197-427', 'i', 'RN')
('198-222', '05 35 19.82', '-05 22 21.55', '198-222', 'i', '0')
('198-448', '05 35 19.83', '-05 24 47.95', '198-448', 'i', '0')
('199-1508', '05 35 19.89', '-05 15 08.25', '0', 'i', 'B')
('200-106', '05 35 20.04', '-05 21 05.99', '200-106', 'i', '0')
('201-534', '05 35 20.14', '-05 25 33.84', '201-534', 'i', '0')
('202-228', '05 35 20.15', '-05 22 28.30', '202-228', 'i', 'B')
('203-504', '05 35 20.26', '-05 25 04.05', '203-504', 'i', 'B')
('203-506', '05 35 20.32', '-05 25 05.55', '0', 'd', 'RN')
('205-330', '05 35 20.45', '-05 23 29.96', '205-330', 'i', 'B')
('205-052', '05 35 20.52', '-05 20 52.05', '205-052', 'i', '0')
('205-421', '05 35 20.53', '-05 24 21.00', '205-421', 'i', 'FO')
('206-446', '05 35 20.62', '-05 24 46.45', '206-446', 'i', '0')
('208-122', '05 35 20.83', '-05 21 21.45', '208-122', 'rn', '0')
('209-151', '05 35 21.00', '-05 21 52.30', '209-151', 'i', 'B')
('210-225', '05 35 21.03', '-05 22 25.20', '0', 'i', '0')
('212-557', '05 35 21.15', '-05 25 57.04', '212-557', 'i', '0')
('212-400', '05 35 21.19', '-05 24 00.20', '0', 'i', '0')
('212-260', '05 35 21.24', '-05 22 59.51', '212-260', 'i', '0')
('213-533', '05 35 21.28', '-05 25 33.11', '0', 'i', 'B')
('213-346', '05 35 21.30', '-05 23 46.10', '0', 'i', 'B')
('215-652', '05 35 21.45', '-05 26 52.40', '0', 'i', '0')
('215-317', '05 35 21.49', '-05 23 16.71', '215-317', 'i', '0')
('215-106', '05 35 21.55', '-05 21 05.60', '215-106', 'i', 'J')
('216-541', '05 35 21.60', '-05 25 40.70', '0', 'i', '0')
('216-715', '05 35 21.62', '-05 27 14.65', '216-715', 'i', '0')
('218-339', '05 35 21.77', '-05 23 39.30', '218-339', 'i', '0')
('218-354', '05 35 21.79', '-05 23 53.90', '218-354', 'd', 'EO,B')
('218-529', '05 35 21.82', '-05 25 28.46', '0', 'i', 'B')
('218-306', '05 35 21.84', '-05 23 06.46', '0', 'i', 'B')
('221-433', '05 35 22.08', '-05 24 32.95', '221-433', 'i', '0')
('223-414', '05 35 22.31', '-05 24 14.25', '223-414', 'i', 'B')
('224-728', '05 35 22.37', '-05 27 28.40', '224-728', 'i', '0')
('228-548', '05 35 22.83', '-05 25 47.69', '228-548', 'i', '0')
('230-536', '05 35 23.02', '-05 25 36.29', '0', 'd', '0')
('231-460', '05 35 23.05', '-05 24 59.86', '231-460', 'i', '0')
('231-502', '05 35 23.16', '-05 25 02.19', '231-502', 'i', 'B')
('231-838', '05 35 23.10', '-05 28 37.34', '231-838', 'i', 'J')
('232-453', '05 35 23.22', '-05 24 52.79', '232-453', 'i', '0')
('234-853', '05 35 23.40', '-05 28 53.19', '0', 'i', '0')
```

```
('236-527', '05 35 23.59', '-05 25 26.54', '236-527', 'i', '0')
('237-627', '05 35 23.66', '-05 26 27.15', '237-627', 'i', '0')
('238-334', '05 35 23.80', '-05 23 34.30', '238-334', 'i', 'B')
('239-334', '05 35 23.86', '-05 23 34.05', '0', 'd', 'EO,B')
('239-510', '05 35 23.98', '-05 25 09.94', '239-510', 'i', '0')
('240-314', '05 35 24.02', '-05 23 13.85', '240-314', 'i', 'B')
('242-519', '05 35 24.22', '-05 25 18.79', '242-519', 'i', '0')
('244-440', '05 35 24.38', '-05 24 39.74', '244-440', 'i', '0')
('245-632', '05 35 24.45', '-05 26 31.55', '245-632', 'i', 'B')
('245-1910', '05 35 24.48', '-05 19 09.84', '0', 'i', 'RN,B')
('245-502', '05 35 24.51', '-05 25 01.59', '245-502', 'i', '0')
('247-436', '05 35 24.69', '-05 24 35.74', '247-436', 'i', 'J')
('250-439', '05 35 25.02', '-05 24 38.49', '250-439', 'i', '0')
('252-457', '05 35 25.21', '-05 24 57.34', '252-457', 'i', '0')
('253-1536', '05 35 25.30', '-05 15 35.54', '0', 'd', 'J,RN,B')
('254-412', '05 35 25.37', '-05 24 11.50', '254-412', 'i', '0')
('255-512', '05 35 25.52', '-05 25 11.84', '0', 'i', '0')
('262-521', '05 35 26.18', '-05 25 20.49', '262-521', 'i', '0')
('264-532', '05 35 26.42', '-05 25 31.69', '264-532', 'i', '0')
('266-558', '05 35 26.62', '-05 25 57.84', '0', 'i', '0')
('280-931', '05 35 27.96', '-05 29 31.15', '0', 'i', '0')
('280-1720', '05 35 28.05', '-05 17 20.33', '0', 'd', 'FO')
('281-306', '05 35 28.13', '-05 23 06.45', '0', 'd', 'FO')
('282-458', '05 35 28.20', '-05 24 58.19', '282-458', 'i', 'J')
('282-614', '05 35 28.20', '-05 26 14.20', '0', 'i', '0')
('284-439', '05 35 28.40', '-05 24 38.69', '0', 'i', '0')
('294-757', '05 35 29.43', '-05 37 56.60', '0', 'rn', '0')
('294-606', '05 35 29.48', '-05 26 06.63', '0', 'd', 'RN')
('297-025', '05 35 29.67', '-05 30 24.75', '0', 'i', '0')
('304-539', '05 35 30.41', '-05 25 38.63', '304-539', 'i', '0')
('307-1807', '05 35 30.70', '-05 18 07.24', '307-1807', 'i', '0')
('314-816', '05 35 31.40', '-05 28 16.48', '0', 'i', '0')
('321-602', '05 35 32.10', '-05 26 01.94', '321-602', 'd', 'EO')
('332-405', '05 35 33.19', '-05 24 04.74', '0', 'd', '0')
('332-1605', '05 35 33.20', '-05 16 05.38', '332-1605', 'i', '0')
('346-1553', '05 35 34.62', '-05 15 52.92', '0', 'd', 'FO')
('347-1535', '05 35 34.67', '-05 15 34.88', '347-1535', 'd', 'J')
('351-3349', '05 35 35.13', '-05 33 49.18', '0', 'i', 'J')
('353-130', '05 35 35.32', '-05 21 29.59', '0', 'j', 'J,B')
('473-245', '05 35 47.34', '-05 22 44.82', '0', 'd', 'EO')],
           mask = [(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)],
      fill_value = ('N/A', 'N/A', 'N/A', 'N/A', 'N/', 'N/A'),
           dtype = [('Name', '<U8'), ('RAJ2000', '<U11'), ('DEJ2000', '<U12'), ('[OW94]', '<U8
```

Bright proplyds have `i` in the `Type` column.

```
In [10]: m = rdata['Type'] == 'i'
         md = rdata['Type'] == 'd'
         mj = rdata['Type'] == 'j'
         mr = rdata['Type'] == 'rn'
```

```
In [11]: rdata[m]
```

```
Out[11]: <Table rows=178 names=('Name','RAJ2000','DEJ2000','[OW94]','Type','Note')>
         masked_array(data = [('4468-605', '05 34 46.76', '-05 26 04.79', '0', 'i', 'J')
          ('4596-400', '05 34 59.56', '-05 24 00.19', '4596-400', 'i', '0')
          ('4582-635', '05 34 58.16', '-05 26 35.13', '0', 'i', '0')
          ('005-514', '05 35 00.47', '-05 25 14.34', '005-514', 'i', '0')
          ('038-627', '05 35 04.19', '-05 26 27.89', '038-627', 'i', '0')
          ('044-527', '05 35 04.42', '-05 25 27.40', '044-527', 'i', '0')
          ('046-245', '05 35 04.63', '-05 22 44.85', '0', 'i', '0')
          ('049-143', '05 35 04.94', '-05 21 42.99', '0', 'i', '0')
          ('057-419', '05 35 05.73', '-05 24 18.55', '057-419', 'i', '0')
          ('061-401', '05 35 06.09', '-05 24 00.60', '061-401', 'i', '0')
          ('064-3335', '05 35 06.44', '-05 33 35.25', '0', 'i', '0')
          ('066-3251', '05 35 06.57', '-05 32 51.49', '0', 'i', '0')
          ('066-652', '05 35 06.59', '-05 26 51.99', '066-652', 'i', 'B')
          ('069-601', '05 35 06.91', '-05 26 00.60', '069-601', 'i', '0')
          ('072-135', '05 35 07.21', '-05 21 34.43', '072-135', 'i', 'EO')
          ('073-227', '05 35 07.27', '-05 22 26.56', '073-227', 'i', '0')
          ('093-822', '05 35 09.59', '-05 28 22.92', '093-822', 'i', '0')
          ('099-339', '05 35 09.89', '-05 23 38.50', '0', 'i', 'J')
          ('102-233', '05 35 10.13', '-05 22 32.74', '102-233', 'i', '0')
          ('102-021', '05 35 10.19', '-05 20 20.99', '102-021', 'i', '0')
          ('102-322', '05 35 10.20', '-05 23 21.56', '102-322', 'i', '0')
          ('106-417', '05 35 10.54', '-05 24 16.70', '106-417', 'i', '0')
          ('106-156', '05 35 10.58', '-05 21 56.24', '106-156', 'i', '0')
          ('109-246', '05 35 10.90', '-05 22 46.36', '109-246', 'i', '0')
          ('109-327', '05 35 10.95', '-05 23 26.45', '109-327', 'i', 'J')
          ('117-421', '05 35 11.65', '-05 24 21.50', '117-421', 'i', '0')
          ('117-352', '05 35 11.73', '-05 23 51.70', '0', 'i', '0')
          ('119-340', '05 35 11.90', '-05 23 39.70', '119-340', 'i', '0')
          ('121-434', '05 35 12.12', '-05 24 33.80', '121-434', 'i', '0')
          ('124-132', '05 35 12.38', '-05 21 31.39', '124-132', 'i', 'J,EO,B')
          ('131-046', '05 35 13.05', '-05 20 45.79', '131-046', 'i', '0')
          ('131-247', '05 35 13.11', '-05 22 47.11', '131-247', 'i', 'J')
          ('132-042', '05 35 13.24', '-05 20 41.94', '132-042', 'i', 'J,EO')
          ('133-353', '05 35 13.30', '-05 23 53.05', '133-353', 'i', 'B')
          ('135-220', '05 35 13.51', '-05 22 19.49', '135-220', 'i', '0')
          ('138-207', '05 35 13.78', '-05 22 07.39', '138-207', 'i', '0')
          ('139-320', '05 35 13.92', '-05 23 20.16', '0', 'i', '0')
          ('141-520', '05 35 14.05', '-05 25 20.50', '141-520', 'i', 'FO')
          ('142-301', '05 35 14.15', '-05 23 00.91', '142-301', 'i', '0')
          ('143-425', '05 35 14.27', '-05 24 24.55', '143-425', 'i', '0')
          ('144-522', '05 35 14.34', '-05 25 22.30', '144-522', 'i', '0')
          ('146-201', '05 35 14.61', '-05 22 00.94', '0', 'i', '0')
          ('147-323', '05 35 14.72', '-05 23 23.01', '0', 'i', '0')
          ('148-305', '05 35 14.80', '-05 23 04.76', '148-305', 'i', 'B')
          ('149-329', '05 35 14.92', '-05 23 29.05', '149-329', 'i', '0')
          ('150-147', '05 35 15.00', '-05 21 47.34', '0', 'i', '0')
          ('150-231', '05 35 15.02', '-05 22 31.11', '150-231', 'i', 'B')
          ('152-319', '05 35 15.20', '-05 23 18.81', '152-319', 'i', '0')
          ('152-738', '05 35 15.21', '-05 27 37.85', '152-738', 'i', '0')
          ('154-324', '05 35 15.35', '-05 23 24.11', '154-324', 'i', 'B')
          ('154-225', '05 35 15.37', '-05 22 25.35', '154-225', 'i', 'B')
          ('154-240', '05 35 15.38', '-05 22 39.85', '0', 'i', '0')
          ('155-338', '05 35 15.51', '-05 23 37.45', '155-338', 'i', '0')
```

```
('157-323', '05 35 15.72', '-05 23 22.59', '157-323', 'i', 'B')
('158-323', '05 35 15.83', '-05 23 22.59', '158-323', 'i', 'B')
('158-327', '05 35 15.79', '-05 23 26.51', '158-327', 'i', 'B')
('158-326', '05 35 15.81', '-05 23 25.51', '158-326', 'i', 'B')
('159-338', '05 35 15.90', '-05 23 38.00', '159-338', 'i', 'B')
('159-418', '05 35 15.90', '-05 24 17.85', '159-418', 'i', '0')
('159-350', '05 35 15.96', '-05 23 50.30', '159-350', 'i', 'B')
('160-353', '05 35 16.01', '-05 23 53.00', '160-353', 'i', 'J')
('161-324', '05 35 16.05', '-05 23 24.35', '161-324', 'i', '0')
('161-328', '05 35 16.07', '-05 23 27.81', '161-328', 'i', '0')
('161-314', '05 35 16.10', '-05 23 14.05', '161-314', 'i', '0')
('162-133', '05 35 16.19', '-05 21 32.39', '162-133', 'i', '0')
('163-210', '05 35 16.27', '-05 22 10.45', '163-210', 'i', 'B')
('163-317', '05 35 16.27', '-05 23 16.51', '163-317', 'i', '0')
('163-222', '05 35 16.30', '-05 22 21.50', '163-222', 'i', 'B,FO')
('163-249', '05 35 16.33', '-05 22 49.01', '163-249', 'i', 'B')
('164-511', '05 35 16.35', '-05 25 09.60', '164-511', 'i', '0')
('165-235', '05 35 16.48', '-05 22 35.16', '165-235', 'i', '0')
('166-316', '05 35 16.61', '-05 23 16.19', '166-315', 'i', '0')
('166-406', '05 35 16.57', '-05 24 06.00', '166-406', 'i', '0')
('166-250', '05 35 16.59', '-05 22 50.36', '166-250', 'i', '0')
('167-317', '05 35 16.74', '-05 23 16.51', '167-317', 'i', '0')
('168-235', '05 35 16.81', '-05 22 34.71', '168-235', 'i', '0')
('168-328', '05 35 16.77', '-05 23 28.06', '168-328', 'i', 'B')
('168-326', '05 35 16.83', '-05 23 25.91', '168-326', 'i', 'B')
('169-338', '05 35 16.88', '-05 23 38.10', '169-338', 'i', 'B')
('170-301', '05 35 16.95', '-05 23 00.91', '170-301', 'i', '0')
('170-249', '05 35 16.96', '-05 22 48.51', '170-249', 'i', 'B')
('170-337', '05 35 16.97', '-05 23 37.15', '170-337', 'i', 'B')
('171-340', '05 35 17.04', '-05 23 39.75', '171-340', 'i', 'B')
('171-434', '05 35 17.11', '-05 24 34.40', '171-434', 'i', 'B')
('173-341', '05 35 17.32', '-05 23 41.40', '173-341', 'i', 'B')
('173-236', '05 35 17.34', '-05 22 35.81', '173-236', 'i', '0')
('174-305', '05 35 17.37', '-05 23 04.86', '174-305', 'i', '0')
('174-414', '05 35 17.38', '-05 24 13.90', '174-414', 'i', '0')
('175-251', '05 35 17.47', '-05 22 51.26', '175-251', 'i', 'B')
('175-355', '05 35 17.54', '-05 23 55.05', '175-355', 'i', '0')
('175-543', '05 35 17.54', '-05 25 42.89', '175-543', 'i', 'J,EO,B')
('176-325', '05 35 17.55', '-05 23 24.96', '176-325', 'i', '0')
('176-252', '05 35 17.64', '-05 22 51.66', '176-252', 'i', '0')
('177-341W', '05 35 17.66', '-05 23 41.00', '177-341', 'i', 'B')
('177-454', '05 35 17.69', '-05 24 54.10', '177-454', 'i', '0')
('177-541', '05 35 17.71', '-05 25 40.76', '0', 'i', 'EO,B')
('177-444', '05 35 17.73', '-05 24 43.75', '177-444', 'i', '0')
('177-341E', '05 35 17.73', '-05 23 41.10', '177-341', 'i', 'B')
('178-441', '05 35 17.81', '-05 24 41.05', '0', 'i', '0')
('178-258', '05 35 17.84', '-05 22 58.15', '0', 'i', '0')
('179-056', '05 35 17.92', '-05 20 55.44', '179-056', 'i', 'B')
('179-354', '05 35 17.96', '-05 23 53.50', '179-354', 'i', '0')
('180-331', '05 35 18.03', '-05 23 30.80', '180-331', 'i', 'B')
('181-247', '05 35 18.08', '-05 22 47.10', '181-247', 'i', 'B')
('181-825', '05 35 18.10', '-05 28 25.04', '0', 'i', 'RN')
('182-413', '05 35 18.22', '-05 24 13.45', '182-413', 'i', '0')
('183-439', '05 35 18.28', '-05 24 38.85', '183-439', 'i', 'B')
```

```
('183-419', '05 35 18.31', '-05 24 18.85', '183-419', 'i', '0')
('184-427', '05 35 18.35', '-05 24 26.85', '184-427', 'i', 'B')
('184-520', '05 35 18.44', '-05 25 19.29', '184-520', 'i', '0')
('187-314', '05 35 18.68', '-05 23 14.01', '187-314', 'i', 'B')
('189-329', '05 35 18.87', '-05 23 28.85', '189-329', 'i', '0')
('190-251', '05 35 19.03', '-05 22 50.65', '0', 'i', 'B')
('191-350', '05 35 19.06', '-05 23 49.50', '191-350', 'i', 'J')
('197-427', '05 35 19.65', '-05 24 26.70', '197-427', 'i', 'RN')
('198-222', '05 35 19.82', '-05 22 21.55', '198-222', 'i', '0')
('198-448', '05 35 19.83', '-05 24 47.95', '198-448', 'i', '0')
('199-1508', '05 35 19.89', '-05 15 08.25', '0', 'i', 'B')
('200-106', '05 35 20.04', '-05 21 05.99', '200-106', 'i', '0')
('201-534', '05 35 20.14', '-05 25 33.84', '201-534', 'i', '0')
('202-228', '05 35 20.15', '-05 22 28.30', '202-228', 'i', 'B')
('203-504', '05 35 20.26', '-05 25 04.05', '203-504', 'i', 'B')
('205-330', '05 35 20.45', '-05 23 29.96', '205-330', 'i', 'B')
('205-052', '05 35 20.52', '-05 20 52.05', '205-052', 'i', '0')
('205-421', '05 35 20.53', '-05 24 21.00', '205-421', 'i', 'FO')
('206-446', '05 35 20.62', '-05 24 46.45', '206-446', 'i', '0')
('209-151', '05 35 21.00', '-05 21 52.30', '209-151', 'i', 'B')
('210-225', '05 35 21.03', '-05 22 25.20', '0', 'i', '0')
('212-557', '05 35 21.15', '-05 25 57.04', '212-557', 'i', '0')
('212-400', '05 35 21.19', '-05 24 00.20', '0', 'i', '0')
('212-260', '05 35 21.24', '-05 22 59.51', '212-260', 'i', '0')
('213-533', '05 35 21.28', '-05 25 33.11', '0', 'i', 'B')
('213-346', '05 35 21.30', '-05 23 46.10', '0', 'i', 'B')
('215-652', '05 35 21.45', '-05 26 52.40', '0', 'i', '0')
('215-317', '05 35 21.49', '-05 23 16.71', '215-317', 'i', '0')
('215-106', '05 35 21.55', '-05 21 05.60', '215-106', 'i', 'J')
('216-541', '05 35 21.60', '-05 25 40.70', '0', 'i', '0')
('216-715', '05 35 21.62', '-05 27 14.65', '216-715', 'i', '0')
('218-339', '05 35 21.77', '-05 23 39.30', '218-339', 'i', '0')
('218-529', '05 35 21.82', '-05 25 28.46', '0', 'i', 'B')
('218-306', '05 35 21.84', '-05 23 06.46', '0', 'i', 'B')
('221-433', '05 35 22.08', '-05 24 32.95', '221-433', 'i', '0')
('223-414', '05 35 22.31', '-05 24 14.25', '223-414', 'i', 'B')
('224-728', '05 35 22.37', '-05 27 28.40', '224-728', 'i', '0')
('228-548', '05 35 22.83', '-05 25 47.69', '228-548', 'i', '0')
('231-460', '05 35 23.05', '-05 24 59.86', '231-460', 'i', '0')
('231-502', '05 35 23.16', '-05 25 02.19', '231-502', 'i', 'B')
('231-838', '05 35 23.10', '-05 28 37.34', '231-838', 'i', 'J')
('232-453', '05 35 23.22', '-05 24 52.79', '232-453', 'i', '0')
('234-853', '05 35 23.40', '-05 28 53.19', '0', 'i', '0')
('236-527', '05 35 23.59', '-05 25 26.54', '236-527', 'i', '0')
('237-627', '05 35 23.66', '-05 26 27.15', '237-627', 'i', '0')
('238-334', '05 35 23.80', '-05 23 34.30', '238-334', 'i', 'B')
('239-510', '05 35 23.98', '-05 25 09.94', '239-510', 'i', '0')
('240-314', '05 35 24.02', '-05 23 13.85', '240-314', 'i', 'B')
('242-519', '05 35 24.22', '-05 25 18.79', '242-519', 'i', '0')
('244-440', '05 35 24.38', '-05 24 39.74', '244-440', 'i', '0')
('245-632', '05 35 24.45', '-05 26 31.55', '245-632', 'i', 'B')
('245-1910', '05 35 24.48', '-05 19 09.84', '0', 'i', 'RN,B')
('245-502', '05 35 24.51', '-05 25 01.59', '245-502', 'i', '0')
('247-436', '05 35 24.69', '-05 24 35.74', '247-436', 'i', 'J')
```

```
('250-439', '05 35 25.02', '-05 24 38.49', '250-439', 'i', '0')
('252-457', '05 35 25.21', '-05 24 57.34', '252-457', 'i', '0')
('254-412', '05 35 25.37', '-05 24 11.50', '254-412', 'i', '0')
('255-512', '05 35 25.52', '-05 25 11.84', '0', 'i', '0')
('262-521', '05 35 26.18', '-05 25 20.49', '262-521', 'i', '0')
('264-532', '05 35 26.42', '-05 25 31.69', '264-532', 'i', '0')
('266-558', '05 35 26.62', '-05 25 57.84', '0', 'i', '0')
('280-931', '05 35 27.96', '-05 29 31.15', '0', 'i', '0')
('282-458', '05 35 28.20', '-05 24 58.19', '282-458', 'i', 'J')
('282-614', '05 35 28.20', '-05 26 14.20', '0', 'i', '0')
('284-439', '05 35 28.40', '-05 24 38.69', '0', 'i', '0')
('297-025', '05 35 29.67', '-05 30 24.75', '0', 'i', '0')
('304-539', '05 35 30.41', '-05 25 38.63', '304-539', 'i', '0')
('307-1807', '05 35 30.70', '-05 18 07.24', '307-1807', 'i', '0')
('314-816', '05 35 31.40', '-05 28 16.48', '0', 'i', '0')
('332-1605', '05 35 33.20', '-05 16 05.38', '332-1605', 'i', '0')
('351-3349', '05 35 35.13', '-05 33 49.18', '0', 'i', 'J')],
             mask = [(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
(False, False, False, False, False, False)
```

```
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)
          (False, False, False, False, False, False)],
               fill_value = ('N/A', 'N/A', 'N/A', 'N/A', 'N/', 'N/A'),
                    dtype = [('Name', '<U8'), ('RAJ2000', '<U11'), ('DEJ2000', '<U12'), ('[OW94]', '<U8
```

```
In [12]: print("Number of bright proplyds:", len(rdata[m]))
```

```
Number of bright proplyds: 178
```

```
In [13]: from astropy import coordinates as coord
         import astropy.units as u
```

Try out the new coordinates interface.

```
In [14]: c = coord.SkyCoord(ra="05 35 33.20", dec="-05 16 05.38", unit=("hourangle", "deg"))
         c
```

```
Out[14]: <SkyCoord (ICRS): ra=83.88833333333332 deg, dec=-5.268161111111111 deg>
```

Add a column with an astropy `SkyCoord` instance for each object

```
In [15]: rdata["coord"] = coord.SkyCoord(ra=rdata["RAJ2000"], dec=rdata["DEJ2000"], unit=("hourangle",
```

Find the coordinates of θ^1 Ori C:

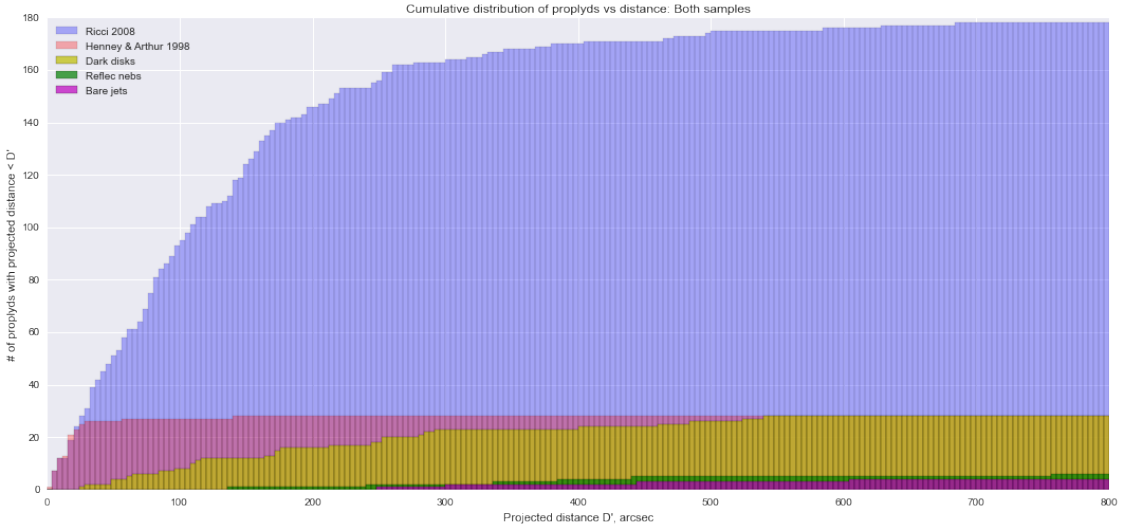
```
In [16]: c0 = coord.get_icrs_coordinates("tet01 ori c")
         c0

Out[16]: <SkyCoord (ICRS): ra=83.818598977 deg, dec=-5.389680154 deg>

In [17]: c0.separation(c).arcsec

Out[17]: 503.84327222379954
```

Add another column with the projected separation in arcsec:

```
In [18]: rdata['Dprime'] = [c0.separation(c).arcsec for c in rdata["coord"]]
```

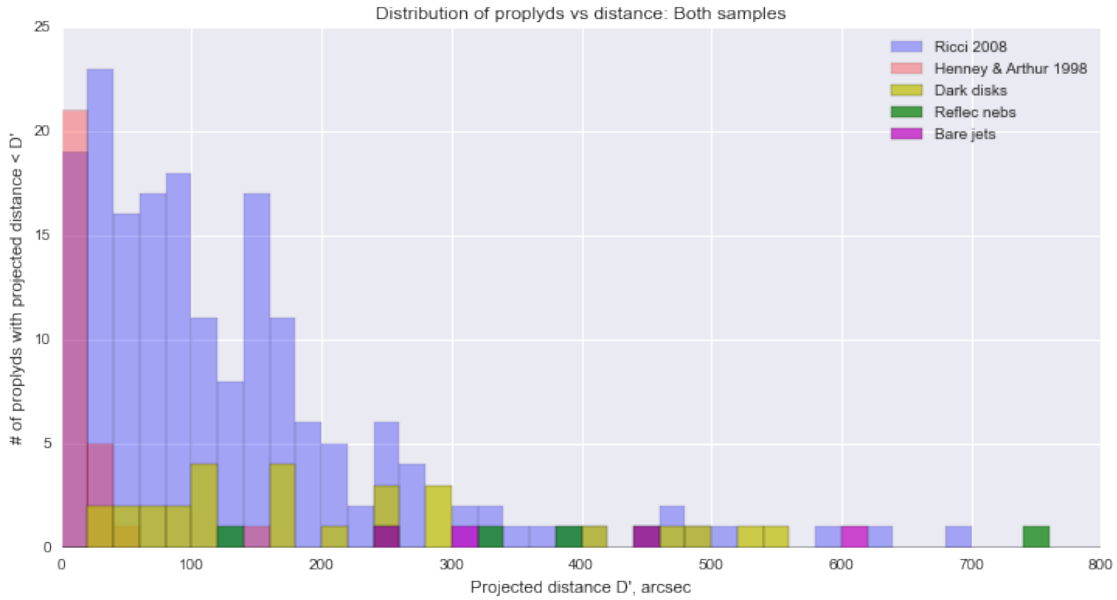
Make a cumulative histogram to compare with the HA98 sample:

```
In [19]: rdata['Dprime'][m].max(), rdata['Dprime'].max()

Out[19]: (685.54604284372397, 932.10782801757205)

In [257]: rmax = 800.0
          nbins = rmax//4
          _ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=True, color='b', alpl
          _ = plt.hist(data["d"], bins=nbins, range=(0, rmax), cumulative=True, color='r', alpha=0.3, la
          _ = plt.hist(rdata["Dprime"][md], bins=nbins, range=(0, rmax), cumulative=True, color='y', alp
          _ = plt.hist(rdata["Dprime"][mr], bins=nbins, range=(0, rmax), cumulative=True, color='g', alp
          _ = plt.hist(rdata["Dprime"][mj], bins=nbins, range=(0, rmax), cumulative=True, color='m', alp
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of proplyds with projected distance < D\'')
          _ = plt.title('Cumulative distribution of proplyds vs distance: Both samples')
          plt.legend(loc='upper left')
          plt.gcf().set_size_inches(18, 8)
          plt.gcf().savefig("proplyd-distance-distribution.pdf")
```



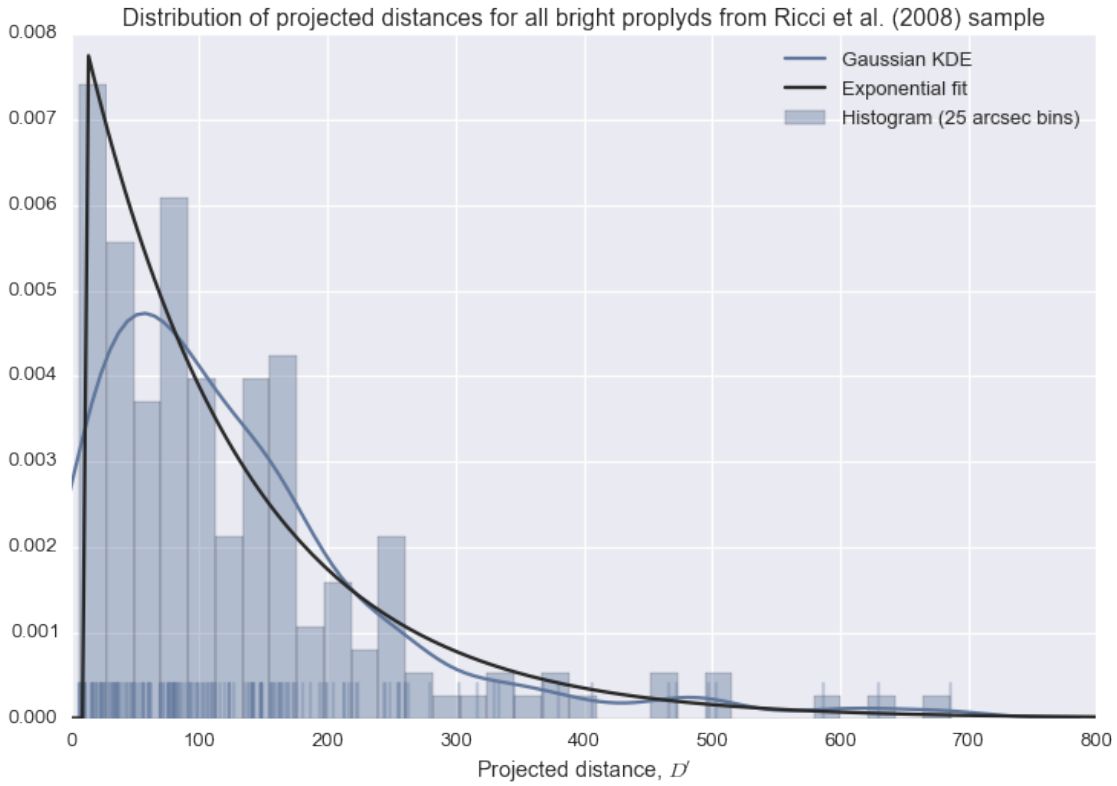
Try it again, but non-cumulative. If the cumulative distribution is linear in D' then the non-cumulative one should be constant.

```
In [256]: rmax = 800.0
          nbins = rmax//20
          _ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=False, color='b', al
          _ = plt.hist(data["d"], bins=nbins, range=(0, rmax), cumulative=False, color='r', alpha=0.3,
          _ = plt.hist(rdata["Dprime"][md], bins=nbins, range=(0, rmax), cumulative=False, color='y', a
          _ = plt.hist(rdata["Dprime"][mr], bins=nbins, range=(0, rmax), cumulative=False, color='g', a
          _ = plt.hist(rdata["Dprime"][mj], bins=nbins, range=(0, rmax), cumulative=False, color='m', a
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of proplyds with projected distance < D\'')
          _ = plt.title('Distribution of proplyds vs distance: Both samples')
          plt.legend(loc='upper right')
          plt.gcf().set_size_inches(12, 6)
```



The ultimate tool for plotting a distribution is `distplot`. This can show the following if we turn on all the options: 1. A rugplot of the individual data points 2. A histogram of the distribution 3. The KDE of the distribution 4. A fit to the distribution of any function from `scipy.stats`

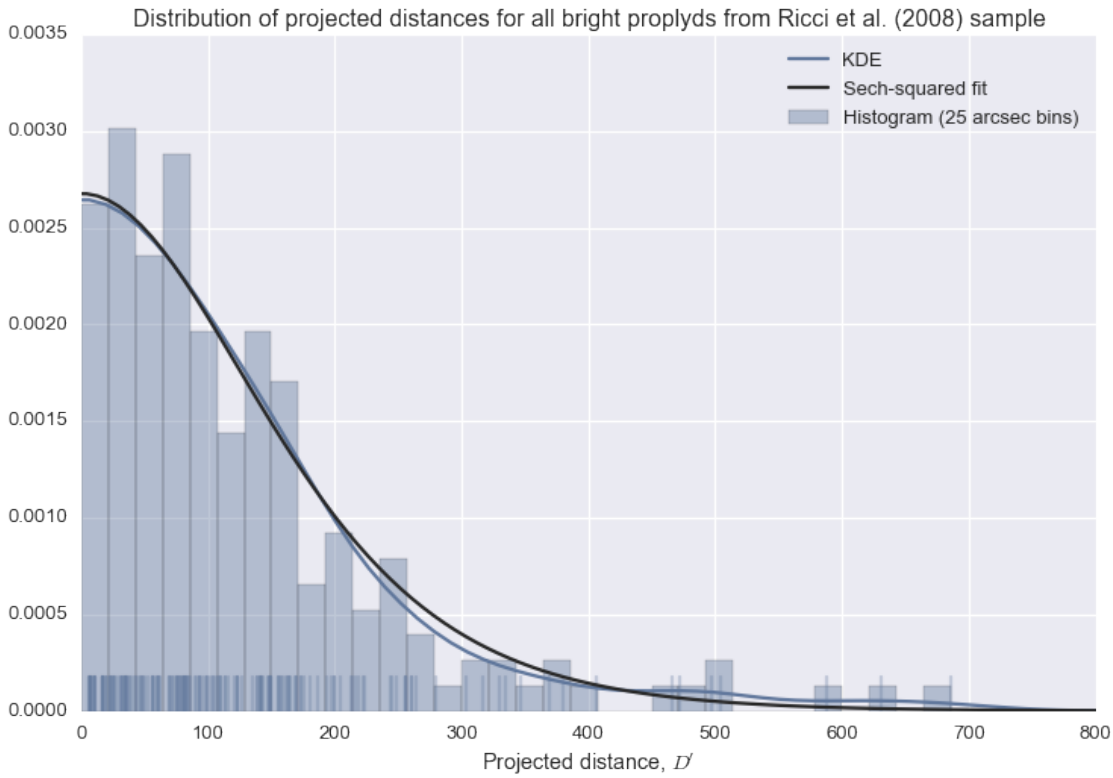
```
In [387]: sns.distplot(rdata['Dprime'][m],
                       rug=True,
                       fit=stats.expon,
                       kde_kws={"clip": (0.0, 800.0), "kernel": "gau", "label": "Gaussian KDE"},
                       rug_kws={"alpha": 0.3},
                       fit_kws={"label": "Exponential fit"},
                       bins=800//25,
                       label="Histogram (25 arcsec bins)", axlabel=r"Projected distance, $D'$")
          plt.legend()
          plt.xlim(0.0, 800.0)
          plt.title("Distribution of projected distances for all bright proplyds from Ricci et al. (200
          None
```



What I don't like about this is that the KDE is affected by the fact that there are no points with $D' < 0$. That is why it drops towards zero. To fix this, we could reflect the dataset about zero and add a negative mirrored dataset.

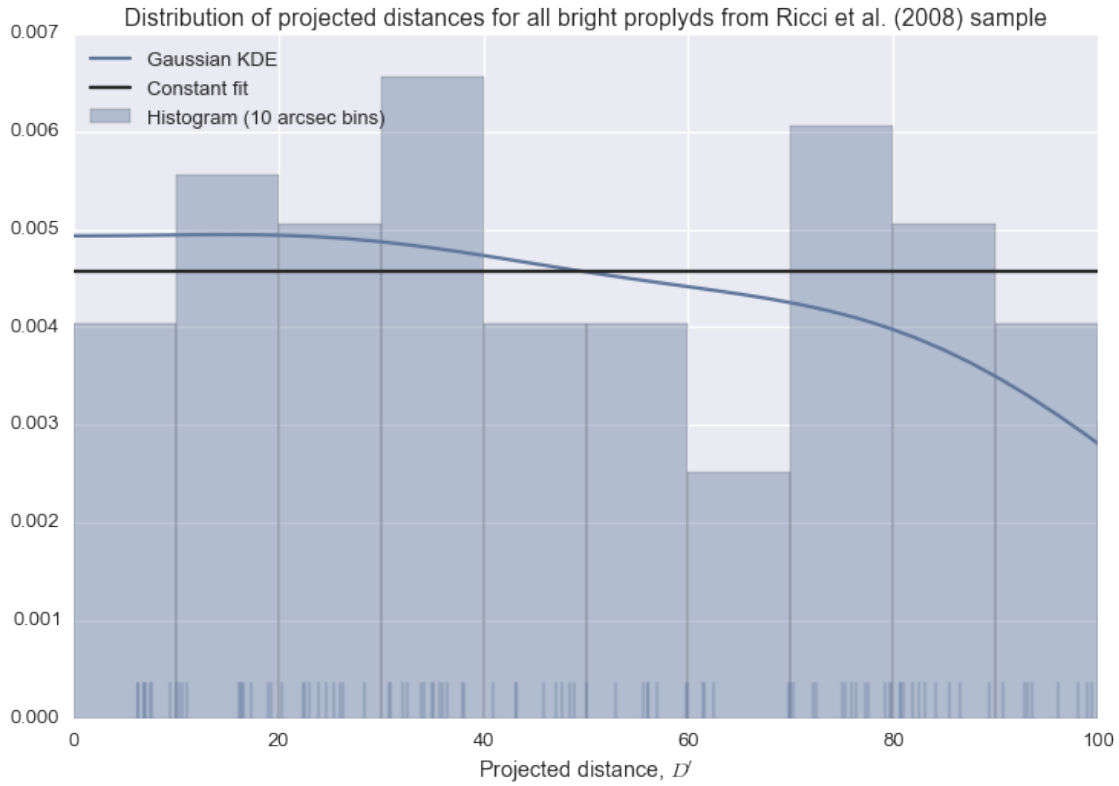
Note that `stats.laplace` is actually a 2-sided explonential. And it doesn't fit very well. On the other hand, the KDE now looks much nicer.

```
In [546]: sns.distplot(np.r_[rdata['Dprime'][m], -rdata['Dprime'][m]],
                       rug=True,
                       fit=stats.logistic,
                       kde_kws={"kernel": "gau", "label": "KDE"},
                       rug_kws={"alpha": 0.3},
                       fit_kws={"label": "Sech-squared fit"},
                       bins=2*800//25,
                       label="Histogram (25 arcsec bins)", axlabel=r"Projected distance, $D'$")
          plt.legend()
          plt.xlim(0.0, 800.0)
          plt.title("Distribution of projected distances for all bright proplyds from Ricci et al. (2008
          None
```



So now, we will try the same trick on the joint distribution of D' and r_0. This is in a previous section. Zooming in on the central part, which is relevant to the bowshocks.

```
In [531]: Dmax = 110.0
          Dmaxx = 800.0
          mzoom = rdata['Dprime'] < Dmax
          dataset = mirrored(np.array(rdata['Dprime'][m & mzoom]))
          sns.distplot(dataset,
                       rug=True,
                       fit=stats.uniform,
                       kde_kws={"kernel": "gau", "label": "Gaussian KDE"},
                       rug_kws={"alpha": 0.3},
                       fit_kws={"label": "Constant fit"},
                       hist_kws={"range": (-Dmaxx, Dmaxx)},
                       bins=2*Dmaxx//10,
                       label="Histogram (10 arcsec bins)", axlabel=r"Projected distance, $D'$")
          plt.legend(loc="upper left")
          plt.xlim(0.0, 100.0)
          plt.title("Distribution of projected distances for all bright proplyds from Ricci et al. (2008
```



So this suggests that at scales $< 100''$ the proplyd distribution is uniform (implying density $\sim D^{-2}$. Whereas at larger scales it does fall off.

```
In [532]: sns.kdeplot?
```

0.1.3 What about the stars?

```
In [21]: sdata = Table.read("Proplyd-Datasets/Robberto2013/table5.dat", format='ascii')

In [22]: sdata.colnames

Out[22]: ['Seq',
          'oncacs',
          'RAh',
          'RAm',
          'RAs',
          'DE-',
          'DEd',
          'DEm',
          'DEs',
          'visit',
          'CCD',
          'x435',
          'x555',
          'x658',
          'x775',
          'x850',
```

```
'y435',
'y555',
'y658',
'y775',
'y850',
'F435mag',
'e_F435mag',
'f_F435mag',
'F555mag',
'e_F555mag',
'f_F555mag',
'F658mag',
'e_F658mag',
'f_F658mag',
'F775mag',
'e_F775mag',
'f_F775mag',
'F850mag',
'e_F850mag',
'f_F850mag',
'rad435',
'rad555',
'rad658',
'rad775',
'rad850',
'csky435',
'csky555',
'csky658',
'csky775',
'csky850',
'e_csky435',
'e_csky555',
'e_csky658',
'e_csky775',
'e_csky850',
'isky435',
'isky555',
'isky658',
'isky775',
'isky850',
'osky435',
'osky555',
'osky658',
'osky775',
'osky850',
'max435',
'max555',
'max658',
'max775',
'max850',
'pixsat435',
'pixsat555',
'pixsat658',
'pixsat775',
```

```
'pixsat850',
'radsat435',
'radsat555',
'radsat658',
'radsat775',
'radsat850',
'type',
'strip',
'xstrip',
'ystrip',
'F435smag',
'F555smag',
'F658smag',
'F775smag',
'F850smag',
'e_F435smag',
'e_F555smag',
'e_F658smag',
'e_F775smag',
'e_F850smag',
'rad435s',
'rad555s',
'rad658s',
'rad775s',
'rad850s',
'sky435s',
'sky555s',
'sky658s',
'sky775s',
'sky850s',
'dsky435s',
'dksy555s',
'dsky658s',
'dksy775s',
'dsky850s',
'Date',
'time']
```

We first need to restrict the table to unique sources, since it contains multiple observations of the same source. We can do this by comparing the first two columns:

```
In [23]: m_uniq = sdata['Seq'] == sdata['oncacs']
```

Then winnowing down the table:

```
In [24]: sdata = sdata[m_uniq]
```

```
In [24]:
```

```
In [24]:
```

Description of the `type` column from notes to Table 5:

```
Source type, from visual inspection: (0) not measured; (1) detected in at least one filter and unresolv
```

```
In [25]: source_types = ['not measured', 'single', 'close double', 'wide double',
                         '???', 'disk', 'ionized', 'galaxy', 'herbig haro']
         count_table = Table(names=['Code', 'Type', 'Number'],
                             dtype=[int, 'S12', int])
         for itype, label in enumerate(source_types):
             count_table.add_row([itype, label, np.sum(sdata['type'] == itype)])
         count_table

Out[25]: <Table rows=9 names=('Code','Type','Number')>
         array([(0, b'not measured', 2), (1, b'single', 2912),
                (2, b'close double', 47), (3, b'wide double', 154), (4, b'???', 0),
                (5, b'disk', 36), (6, b'ionized', 184), (7, b'galaxy', 59),
                (8, b'herbig haro', 5)],
               dtype=[('Code', '<i8'), ('Type', 'S12'), ('Number', '<i8')])
```

The `ionized` type is "*proplyd or with other evidence of photoionization*" and there are 184 of them, compared with 178 in the Ricci catalog. This is close enough for me.

Next job is to winnow out types 0, 7, and 8, which we don't want.

```
In [26]: sdata = sdata[(sdata['type'] > 0) & (sdata['type'] < 7)]
```

And now create a mask to select out the `ionized` sources (mainly proplyds).

```
In [27]: m_rob_ionized = sdata['type'] == 6
```

Now we combine the 6 columns used for coordinates into a single RA-Dec string:

```
In [28]: sdata["coordstring"] = ["{} {} {} -{} {} {}".format(*args)
                                 for args in zip(*[sdata[col] for col in ["RAh", "RAm", "RAs", "DEd", "DEm",

In [29]: sdata["coordstring"]

Out[29]: <MaskedColumn name='coordstring' unit=None format=None description=None>
         masked_BaseColumn(data = ['5 34 9.889 -5 22 26.26' '5 34 10.25 -5 21 52.29'
          '5 34 11.111 -5 22 54.46' ..., '5 35 52.472 -5 38 7.85'
          '5 35 53.093 -5 30 10.38' '5 35 53.504 -5 39 32.88'],
                           mask = [False False False ..., False False False],
                     fill_value = N/A)
```

Then that string can be used to initialize an `astropy.coordinates.SkyCoord` instance for each source.

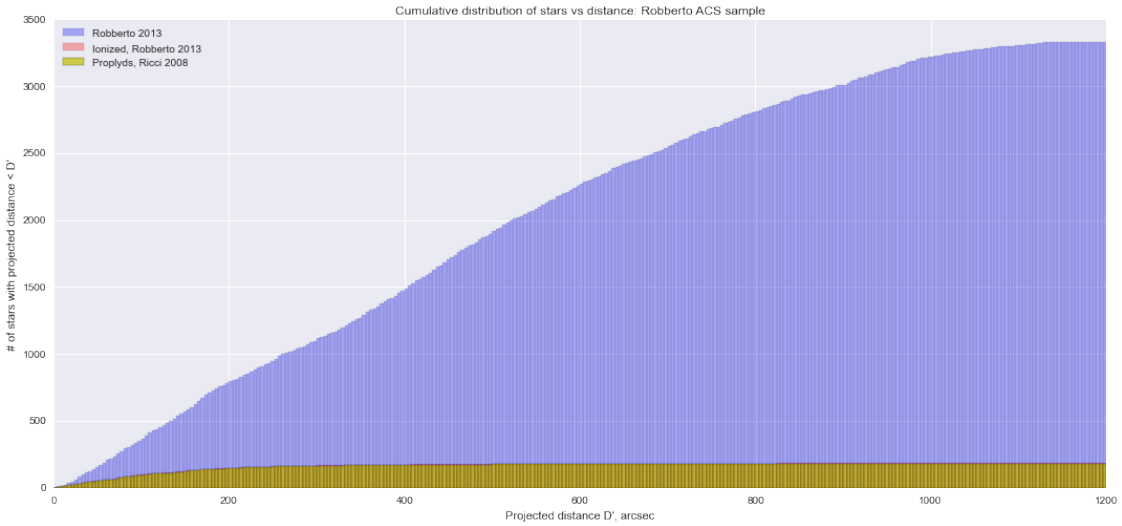
Warning: The following cell takes a couple of minutes to run.

```
In [30]: sdata["coord"] = coord.SkyCoord(sdata["coordstring"], unit=("hourangle", "deg"))

In [31]: sdata["coord"]

Out[31]: <MaskedColumn name='coord' unit=None format=None description=None>
         masked_BaseColumn(data = [<SkyCoord (ICRS): ra=83.54120416666666 deg, dec=-5.373961111111111 de
          <SkyCoord (ICRS): ra=83.54270833333332 deg, dec=-5.3645249999999995 deg>
          <SkyCoord (ICRS): ra=83.54629583333332 deg, dec=-5.381794444444444 deg>
          ...,
          <SkyCoord (ICRS): ra=83.96863333333332 deg, dec=-5.635513888888888 deg>
          <SkyCoord (ICRS): ra=83.97122083333332 deg, dec=-5.502883333333333 deg>
          <SkyCoord (ICRS): ra=83.97293333333332 deg, dec=-5.659133333333334 deg>],
                           mask = [False False False ..., False False False],
                     fill_value = ?)
```

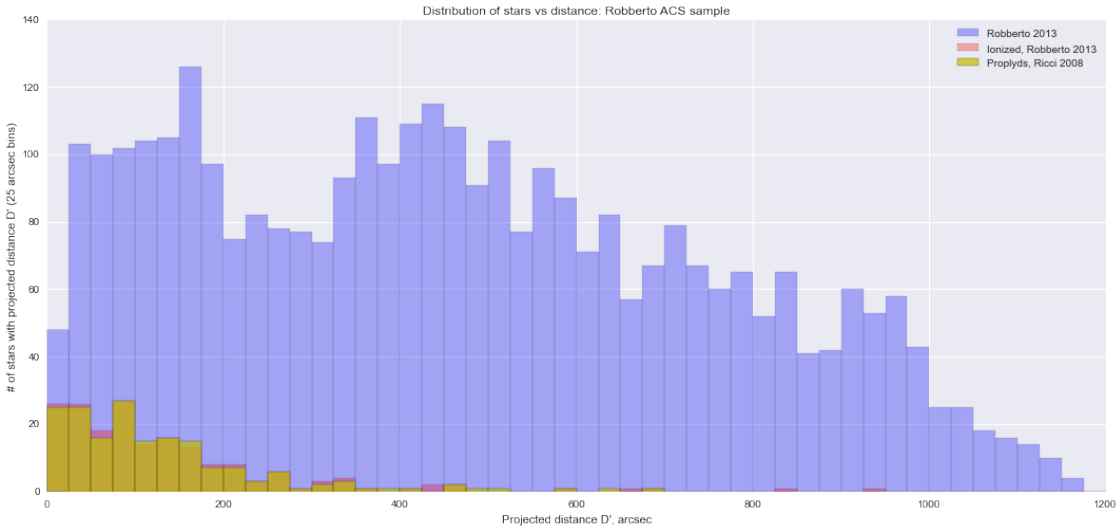
And that in turn can be used to find the separation of each source from θ^1 C.

```
In [32]: sdata['Dprime'] = [c0.separation(c).arcsec for c in sdata["coord"]]

In [33]: len(sdata['Dprime'])

Out[33]: 3333
```

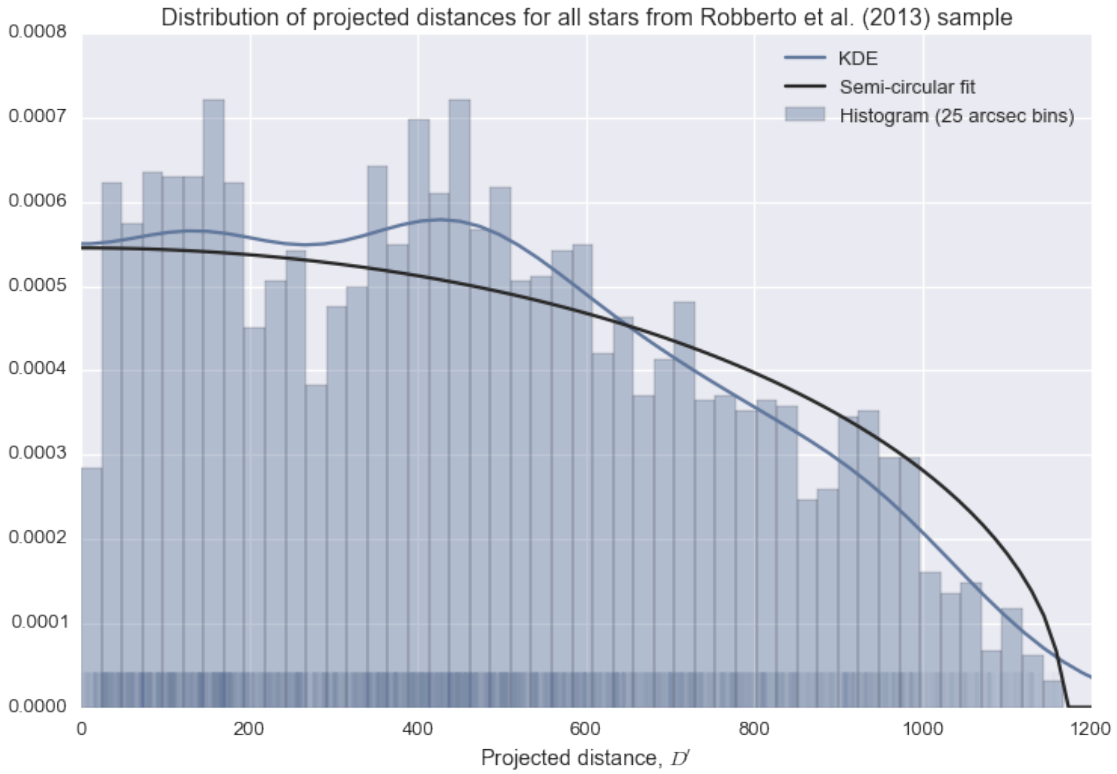
```
In [258]: rmax = 1200.0
          nbins = rmax//4
          _ = plt.hist(sdata["Dprime"], bins=nbins, range=(0, rmax), cumulative=True, color='b', alpha=
          _ = plt.hist(sdata["Dprime"][m_rob_ionized], bins=nbins, range=(0, rmax), cumulative=True, co
          _ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=True, color='y', alp
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of stars with projected distance < D\'')
          _ = plt.title('Cumulative distribution of stars vs distance: Robberto ACS sample')
          plt.legend(loc='upper left')
          plt.gcf().set_size_inches(18, 8)
```




```
In [259]: rmax = 1200.0
          binsize = 25
          nbins = rmax//binsize
          _ = plt.hist(sdata["Dprime"], bins=nbins, range=(0, rmax), cumulative=False, color='b', alpha=
          _ = plt.hist(sdata["Dprime"][m_rob_ionized], bins=nbins, range=(0, rmax), cumulative=False, c
          _ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=False, color='y', al
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of stars with projected distance D\' (25 arcsec bins)')
          _ = plt.title('Distribution of stars vs distance: Robberto ACS sample')
          plt.legend(loc='upper right')
          plt.gcf().set_size_inches(18, 8)
```

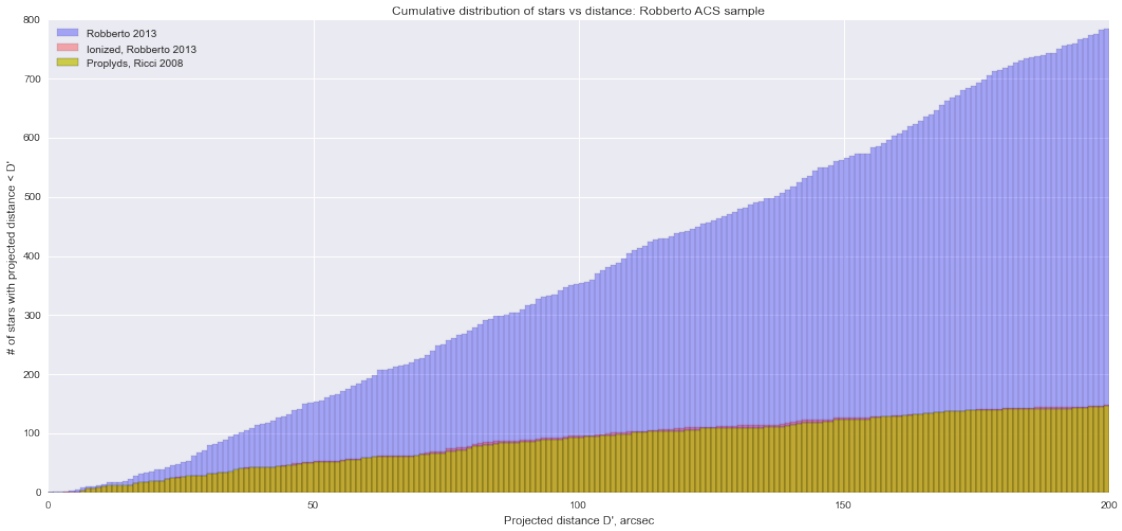


In [554]:
```
rmax = 1200.0
sns.distplot(mirrored(sdata['Dprime']),
             rug=True,
             fit=stats.semicircular,
             kde_kws={"kernel": "gau", "label": "KDE"},
             rug_kws={"alpha": 0.03},
             fit_kws={"label": "Semi-circular fit"},
             bins=2*rmax//25,
             label="Histogram (25 arcsec bins)", axlabel=r"Projected distance, $D'$")
plt.legend()
plt.xlim(0.0, 1200.0)
plt.title("Distribution of projected distances for all stars from Robberto et al. (2013) sampl
None
```



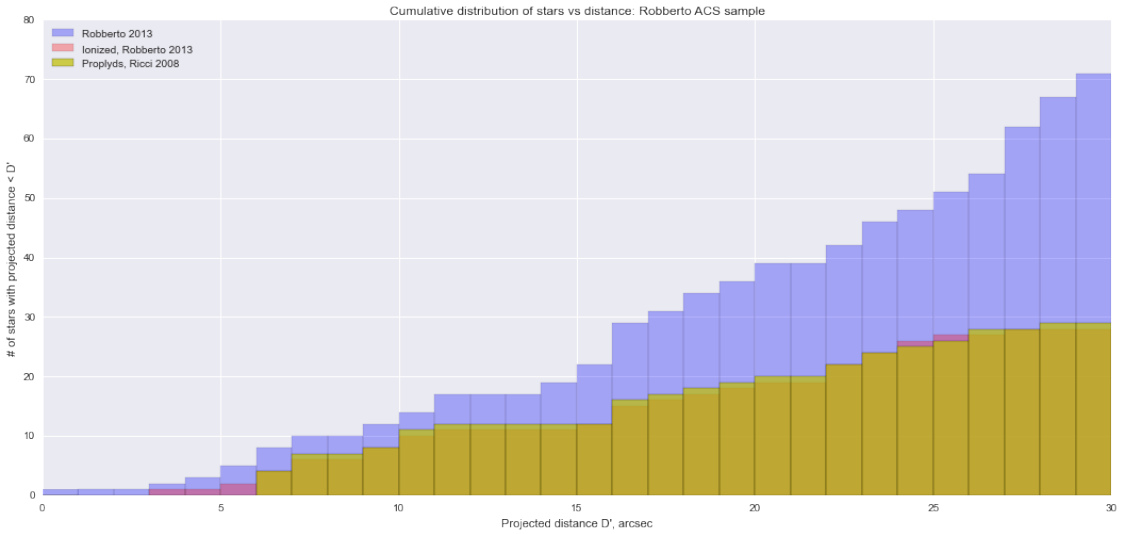
Zoom in on inner 200”

```
In [260]: rmax = 200.0
          nbins = rmax
          _ = plt.hist(sdata["Dprime"], bins=nbins, range=(0, rmax), cumulative=True, color='b', alpha=
          _ = plt.hist(sdata["Dprime"][m_rob_ionized], bins=nbins, range=(0, rmax), cumulative=True, co
          _ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=True, color='y', alp
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of stars with projected distance < D\'')
          _ = plt.title('Cumulative distribution of stars vs distance: Robberto ACS sample')
          plt.legend(loc='upper left')
          plt.gcf().set_size_inches(18, 8)
```



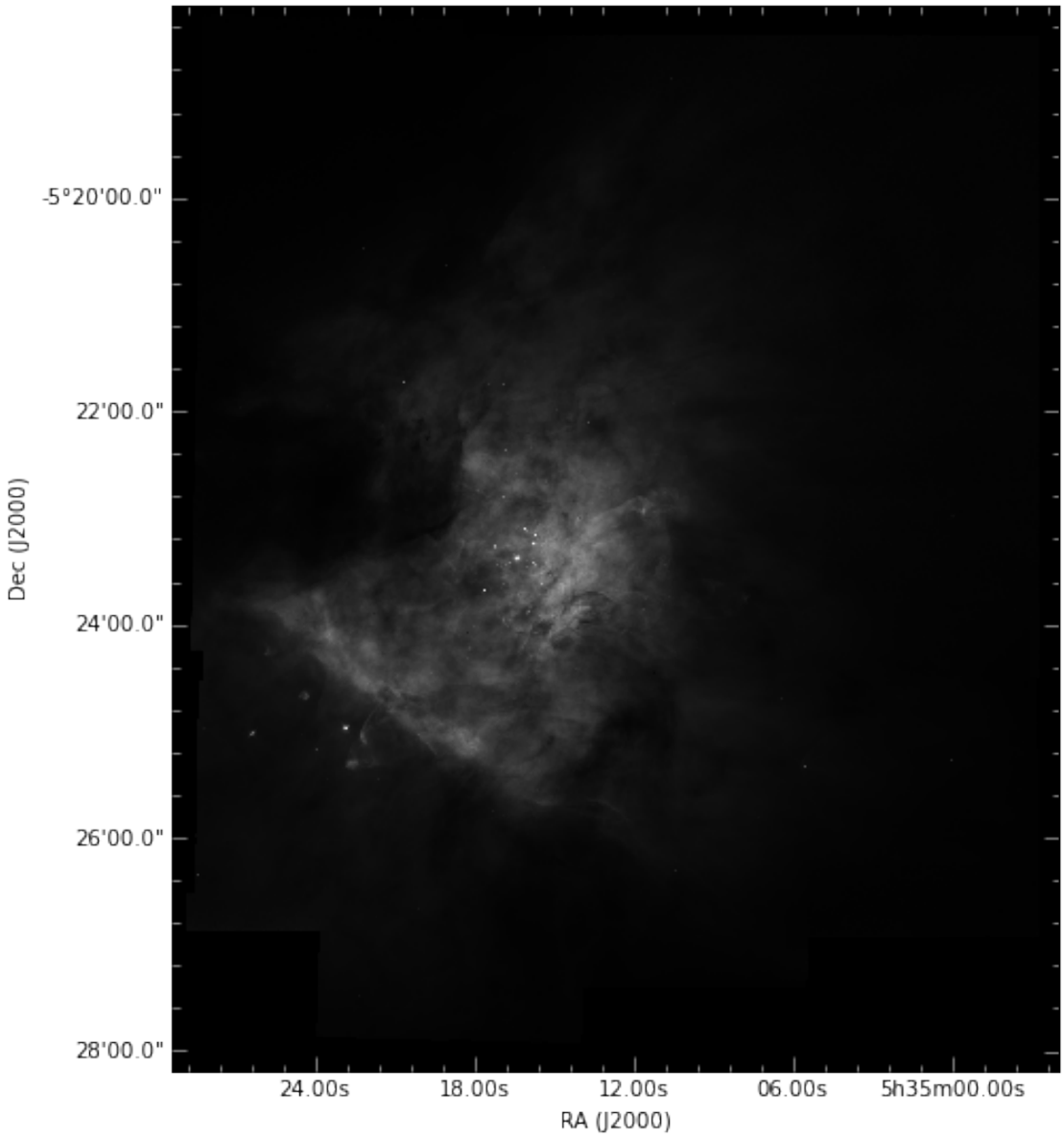
Zoom in on inner 30”

In [261]:
```
rmax = 30.0
nbins = rmax
_ = plt.hist(sdata["Dprime"], bins=nbins, range=(0, rmax), cumulative=True, color='b', alpha=
_ = plt.hist(sdata["Dprime"][m_rob_ionized], bins=nbins, range=(0, rmax), cumulative=True, co
_ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=True, color='y', alp
_ = plt.xlabel('Projected distance D\', arcsec')
_ = plt.ylabel('# of stars with projected distance < D\'')
_ = plt.title('Cumulative distribution of stars vs distance: Robberto ACS sample')
plt.legend(loc='upper left')
plt.gcf().set_size_inches(18, 8)
```



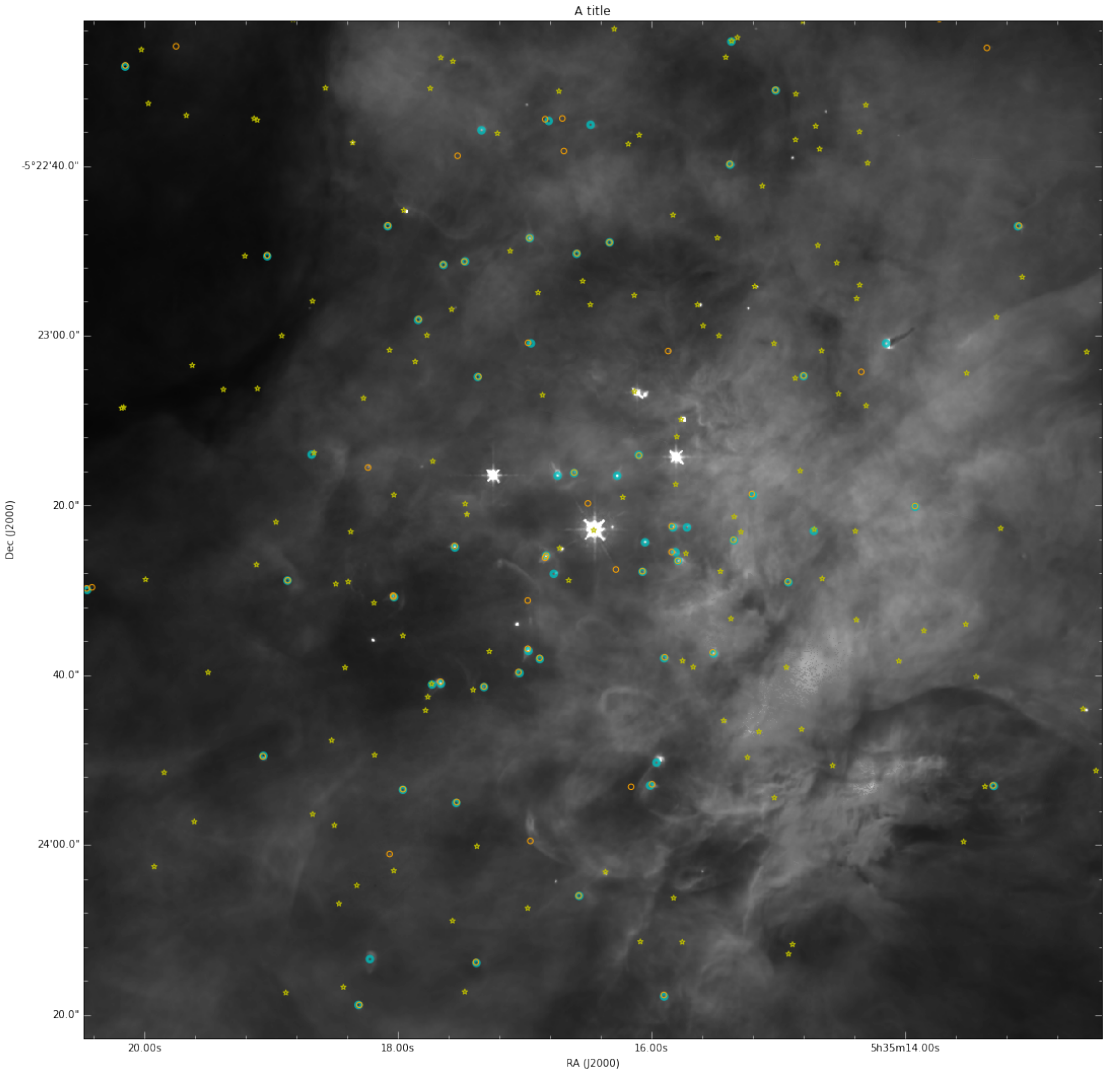
0.2 Maps of different source samples

```
In [51]: rob_ra = [c.ra.deg for c in sdata['coord']]
         rob_dec = [c.dec.deg for c in sdata['coord']]

In [56]: rob_ra = np.array(rob_ra)
         rob_dec = np.array(rob_dec)

In [74]: ricci_ra = np.array([c.ra.deg for c in rdata['coord']])
         ricci_dec = np.array([c.dec.deg for c in rdata['coord']])

In [38]: import aplpy
         import os

In [41]: from imp import reload

In [39]: aplpy.__version__

Out[39]: '0.9.13.dev310'

In [40]: hstdir = os.path.expanduser("~/Dropbox/JorgeBowshocks/HST")

In [105]: fig = aplpy.FITSFigure(os.path.join(hstdir, "mosaicf656-fixw-align.fits"))
          fig.show_grayscale(vmin=0.0, vmax=5.e3)

WARNING: Cannot determine equinox. Assuming J2000. [aplpy.wcs_util]
WARNING:astropy:Cannot determine equinox. Assuming J2000.
WARNING: Cannot determine equinox. Assuming J2000. [aplpy.wcs_util]
WARNING:astropy:Cannot determine equinox. Assuming J2000.
```



In [108]:
```
fig._figure.set_size_inches(18, 18)
fig.show_markers(ricci_ra[m], ricci_dec[m], layer='Ricci proplyds', edgecolor='c', lw=3, alpha
fig.show_markers(rob_ra[m_rob_ionized], rob_dec[m_rob_ionized], layer='Robb proplyds', edgecol
fig.show_markers(rob_ra[~m_rob_ionized], rob_dec[~m_rob_ionized], layer='Robb non-proplyds',
fig.recenter(c0.ra.deg, c0.dec.deg, radius=60/3600)
ax = fig._figure.axes[0]
ax.set_title('A title')
fig._figure
```

Out[108]:



So the conclusion of this section is that the Ricci proplyd catalog (shown by blue circles on the image above) is more reliable than the Robberto `ionized` catalog (shown by orange circles) for the close-in proplyds. Robberto is missing a lot of obvious proplyds, and furthermore has a few sources that don't seem to correspond to anything, although some might be bowshocks or jet knots.

The Ricci catalog is only missing a very few, such as the one very close to th1c.

0.3 Monte Carlo simulations

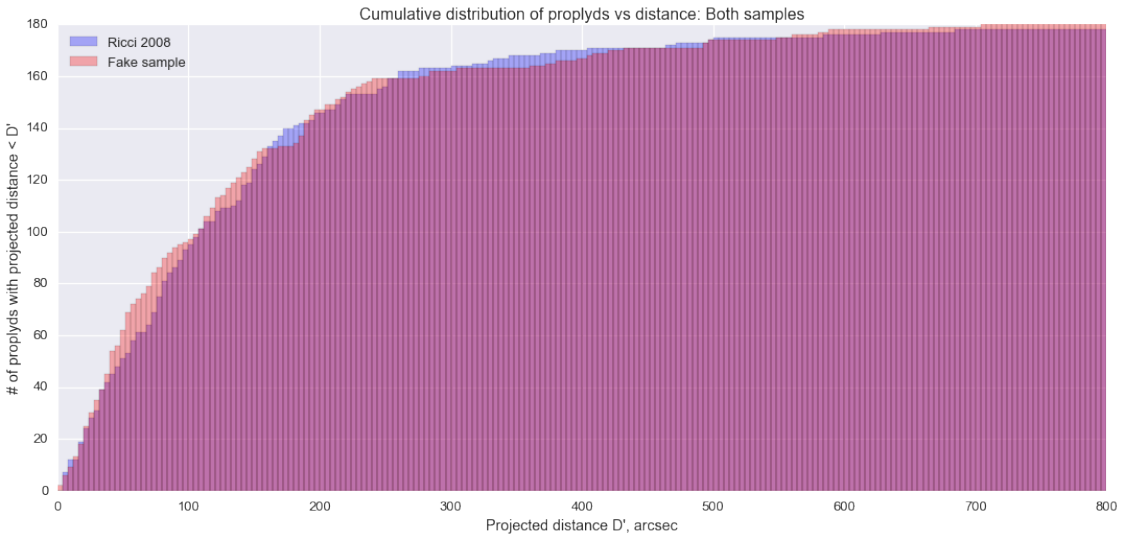
Assume power law distribution in radius.

```
In [562]: class SourcePopulation(object):

              def __init__(self, N, Dmin=0.0, Dmax=800.0, m=2.6):
```

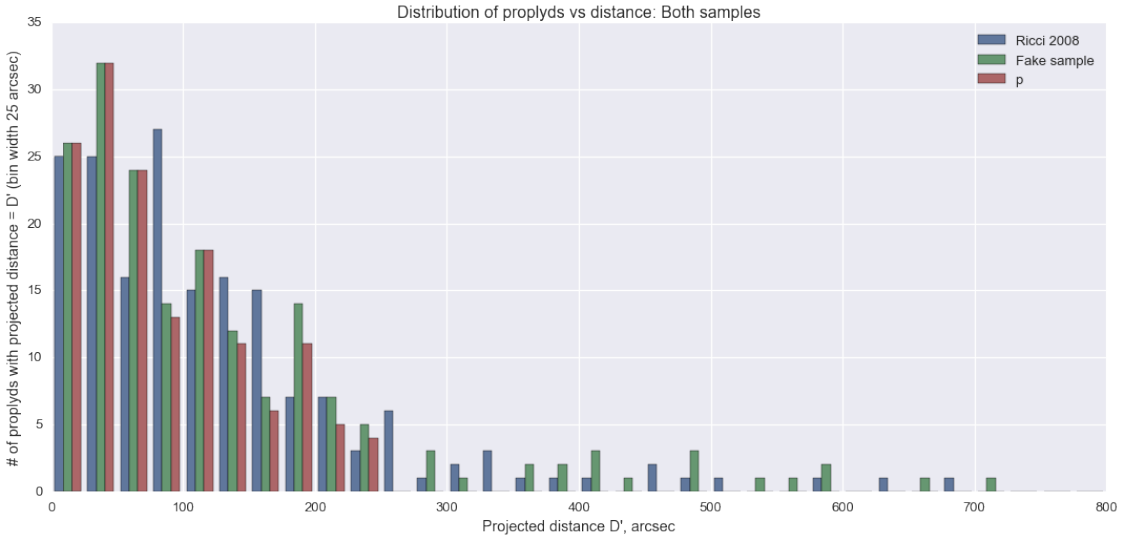
```
        """Return `N` samples of sources drawn from a radial distribution with density ~ D^-m
        Maximum and minimum radii can be specified with `Dmin` and `Dmax`.
        If `Dmin = 0` then `m < 3` is required."""
        assert(m < 3)

        self.N = N
        self.m = m
        self.Dmin = Dmin
        self.Dmax = Dmax

        # Radial distances from center
        xi = np.random.random_sample((N,))
        x0 = Dmin/Dmax
        self.D = Dmax*((1 - x0**(3.0 - m))*xi + x0**(3.0 - m))**(1.0/(3.0 - m))
        # Angle cosine with LOS
        xi = np.random.random_sample((N,))
        self.mu = 2*xi - 1.0
        # Position angle
        xi = np.random.random_sample((N,))
        self.PA = 360.0*xi

        # Inclination to plane of sky inc = [-90:90]
        self.inc = np.degrees(np.arcsin(self.mu))

        # Projected distance
        self.Dprime = self.D*np.sqrt(1.0 - self.mu**2)

        # Plane-of-sky coords
        self.Dec = self.Dprime*np.cos(np.radians(self.PA))
        self.RA = self.Dprime*np.sin(np.radians(self.PA))
```

In [662]:
```
p = SourcePopulation(150, m=2.1, Dmin=5.0, Dmax=250.0)
p2 = SourcePopulation(30, m=2.5, Dmin=150.0, Dmax=800.0)
Dprimes = np.r_[p.Dprime, p2.Dprime]
```

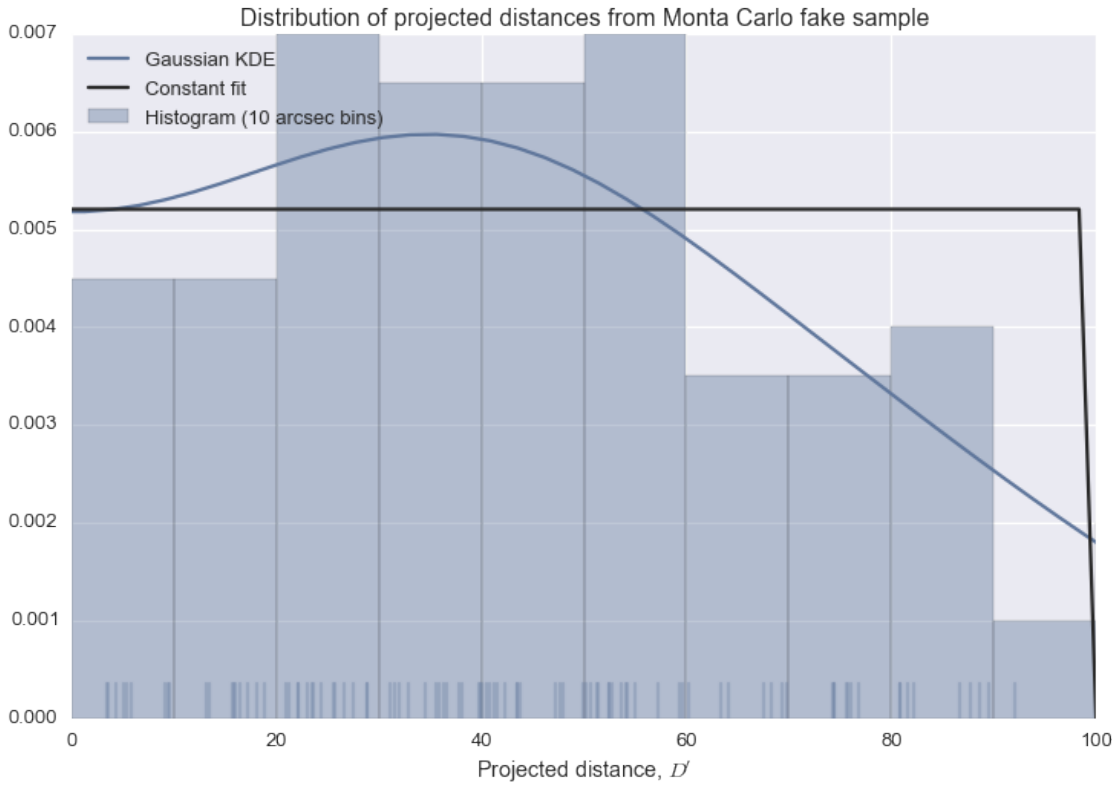
In [663]:
```
rmax = 800.0
nbins = rmax//4
_ = plt.hist(rdata["Dprime"][m], bins=nbins, range=(0, rmax), cumulative=True, color='b', alp
_ = plt.hist(Dprimes, bins=nbins, range=(0, rmax), cumulative=True, color='r', alpha=0.3, lab
# _ = plt.hist(p.Dprime, bins=nbins, range=(0, rmax), cumulative=True, color='g', alpha=0.3,
# _ = plt.hist(p2.Dprime, bins=nbins, range=(0, rmax), cumulative=True, color='y', alpha=0.8,
_ = plt.xlabel('Projected distance D\', arcsec')
_ = plt.ylabel('# of proplyds with projected distance < D\'')
_ = plt.title('Cumulative distribution of proplyds vs distance: Both samples')
plt.legend(loc='upper left')
plt.gcf().set_size_inches(18, 8)
```




```
In [250]: import seaborn

In [251]: cmap = seaborn.palettes.dark_palette('red', as_cmap=True)

In [664]: rmax = 800.0
          nbins = rmax//25
          datasets = [rdata["Dprime"][m], Dprimes, p.Dprime, p2.Dprime]
          colors = ['b', 'y', 'g', 'y']
          labels = ['Ricci 2008', 'Fake sample', 'p', 'p2']
          _ = plt.hist(datasets[:3], bins=nbins, range=(0, rmax), cumulative=False, color=None, label=la
          _ = plt.xlabel('Projected distance D\', arcsec')
          _ = plt.ylabel('# of proplyds with projected distance = D\' (bin width 25 arcsec)')
          _ = plt.title('Distribution of proplyds vs distance: Both samples')
          plt.legend(loc='upper right')
          plt.gcf().set_size_inches(18, 8)
```



```
In [665]: Dmax = 110.0
          Dmaxx = 800.0
          mzoom = Dprimes < Dmax
          dataset = mirrored(Dprimes[mzoom])
          sns.distplot(dataset,
                       rug=True,
                       fit=stats.uniform,
                       kde_kws={"kernel": "gau", "label": "Gaussian KDE"},
                       rug_kws={"alpha": 0.3},
                       fit_kws={"label": "Constant fit"},
                       hist_kws={"range": (-Dmaxx, Dmaxx)},
                       bins=2*Dmaxx//10,
                       label="Histogram (10 arcsec bins)", axlabel=r"Projected distance, $D'$")
          plt.legend(loc="upper left")
          plt.xlim(0.0, 100.0)
          plt.title("Distribution of projected distances from Monta Carlo fake sample");
```

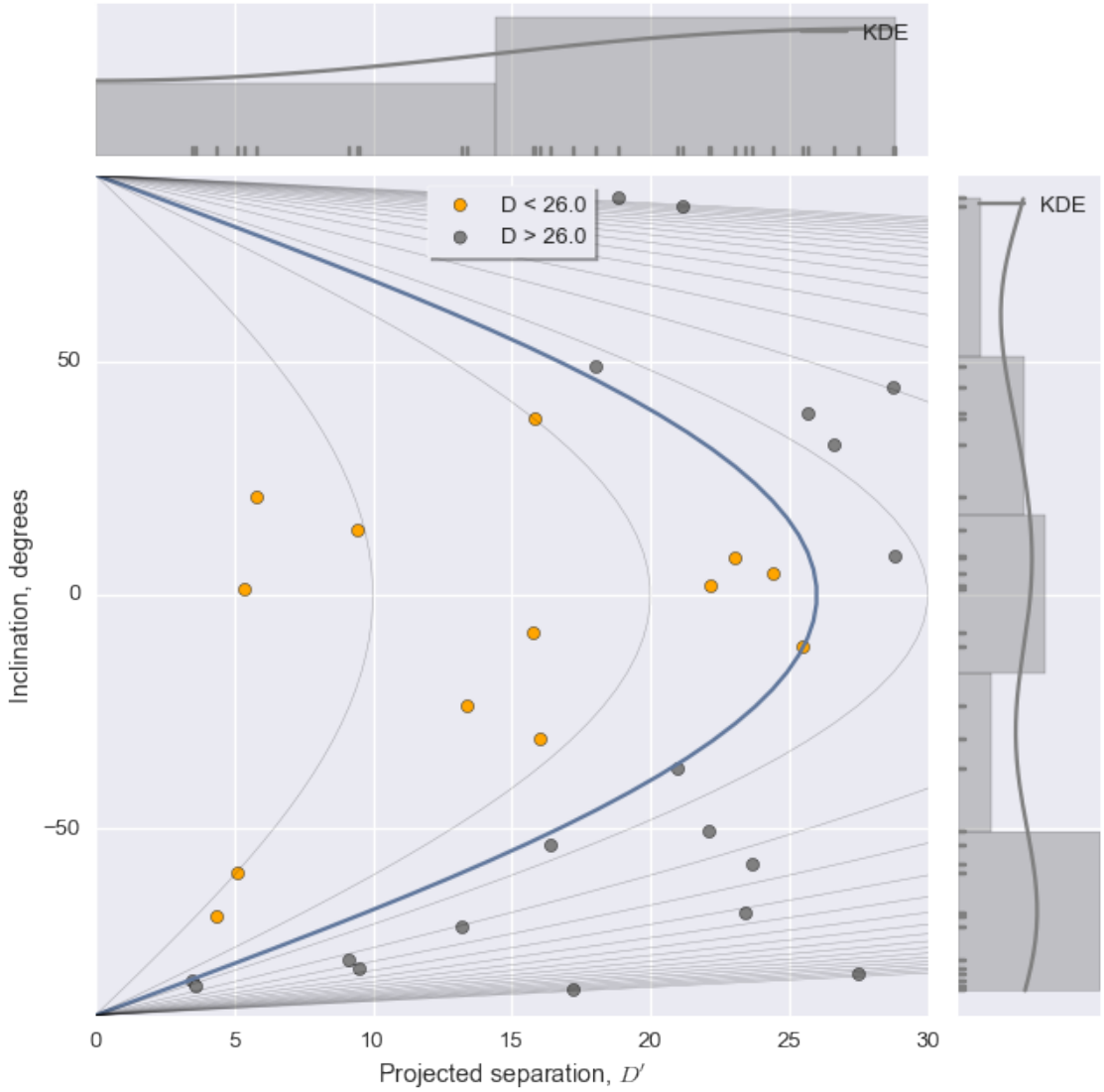


0.3.1 Compare the projected separation and the inclination for the inner 30 arcsec of the MC sample

Assume that the inner wind shock lies at about $D = 26''$. Only proplyds that are physically clser than this will have Type I bowshocks. These are shown by orange points in the figure. Physically more distant

proplyds are shown by gray points. We also show contours of physical separation every $10''$ from $D = 10''$ to $D = 200''$.

```
In [674]: color = ".5"
          m30 = p.Dprime <= 30.0
          Dmax = 26.0
          mwind = p.D <= Dmax
          g = sns.JointGrid(mirrored(p.Dprime[m30]), np.r_[p.inc[m30], p.inc[m30]], size=9, xlim=(0, 30)
          g.plot_marginals(sns.distplot, color=color, rug=True,
                           kde_kws=dict(label="KDE", bw="silverman", cut=0))
          #g.plot_joint(sns.regplot, color=color, fit_reg=False, scatter_kws=dict(s=50))
          g.ax_joint.scatter(p.Dprime[mwind], p.inc[mwind], marker='o', c="orange", s=50, label="D < {}"
          g.ax_joint.scatter(p.Dprime[~mwind], p.inc[~mwind], marker='o', c=color, s=50, label="D > {}"
          inc_pts = np.linspace(-90.0, 90.0)
          Dprime_pts = Dmax*np.cos(np.radians(inc_pts))
          g.ax_joint.plot(Dprime_pts, inc_pts, '-')
          for D in np.arange(10, 200, 10):
              g.ax_joint.plot(Dprime_pts*D/Dmax, inc_pts, '-', color='k', lw=0.5, alpha=0.3)
          g.ax_joint.legend(loc="upper center", frameon=True, shadow=True)
          g.ax_joint.set_xlabel("Projected separation, $D'$")
          g.ax_joint.set_ylabel("Inclination, degrees")
          g.ax_marg_x.legend(loc="upper right")
          g.ax_marg_y.legend(loc="best")
          g.fig.tight_layout()
          #g.fig.savefig("proplyd-joint-separation-size.pdf");
```



0.3.2 Distribution of sizes

Read in the size table that I converted from the DS9 regions.

```
In [874]: sizedata = Table.read("Proplyd-Datasets/proplyd-chords-D60.dat", format='ascii.commented_head
          sizedata[:4]
```

```
Out[874]: <Table rows=4 names=('ID','RA1','DEC1','RA2','DEC2')>
          array([('154-240', '5:35:15.394', '-5:22:39.61', '5:35:15.354', '-5:22:39.71'),
                 ('163-249', '5:35:16.340', '-5:22:48.79', '5:35:16.324', '-5:22:48.80'),
                 ('165-235S', '5:35:16.487', '-5:22:35.29', '5:35:16.458', '-5:22:35.22'),
                 ('165-235N', '5:35:16.488', '-5:22:34.97', '5:35:16.479', '-5:22:34.97')],
                dtype=[('ID', '<U8'), ('RA1', '<U11'), ('DEC1', '<U11'), ('RA2', '<U11'), ('DEC2', '<U1
```

Create `coord.SkyCoord` objects for the two ends of the chord that I measured at the back of each proplyd's head. Then use those to find the chord diameter in arcsec (`Chord` column), and then convert to a radius in units of 10^{14} cm (`r14` column).

```
In [875]: coords1 = coord.SkyCoord(ra=sizedata["RA1"], dec=sizedata["DEC1"], unit=("hourangle", "deg"))
          coords2 = coord.SkyCoord(ra=sizedata["RA2"], dec=sizedata["DEC2"], unit=("hourangle", "deg"))
          sizedata["Chord"] = [c1.separation(c2).arcsec for c1, c2 in zip(coords1, coords2)]
          sizedata["r14"] = 0.5 * sizedata["Chord"] * 430*u.AU.in_units(u.cm)/1.e14
          sizedata[:4]

Out[875]: <Table rows=4 names=('ID','RA1','DEC1','RA2','DEC2','Chord','r14')>
          array([ ('154-240', '5:35:15.394', '-5:22:39.61', '5:35:15.354', '-5:22:39.71', 0.60567146710
                  ('163-249', '5:35:16.340', '-5:22:48.79', '5:35:16.324', '-5:22:48.80', 0.239151817904
                  ('165-235S', '5:35:16.487', '-5:22:35.29', '5:35:16.458', '-5:22:35.22', 0.43870683708
                  ('165-235N', '5:35:16.488', '-5:22:34.97', '5:35:16.479', '-5:22:34.97', 0.13440609108
                 dtype=[('ID', '<U8'), ('RA1', '<U11'), ('DEC1', '<U11'), ('RA2', '<U11'), ('DEC2', '<U1
```

Take the average of the RA and DEC components of the two ends of the chord to get an average coordinate (`coords0`) for each source. **Note: This will not quite be the center of the proplyd, but it is close enough.**

```
In [876]: coords0 = coord.SkyCoord(ra=0.5*(coords1.ra + coords2.ra), dec=0.5*(coords1.dec + coords2.dec
          coords0[:4]

Out[876]: <SkyCoord (ICRS): (ra, dec) in deg
              [(83.81405833333332, -5.377683333333333),
               (83.81804999999999, -5.380220833333333),
               (83.81863541666667, -5.376459722222222),
               (83.81868124999998, -5.3763805555555555)]>
```

Remove the columns with the chord end coordinates because we don't need them any more.

```
In [877]: sizedata.remove_columns(["RA1", "RA2", "DEC1", "DEC2"])
          sizedata[:4]

Out[877]: <Table rows=4 names=('ID','Chord','r14')>
          array([('154-240', 0.6056714671056789, 19.48053979189225),
                 ('163-249', 0.23915181790427623, 7.691969587490479),
                 ('165-235S', 0.4387068370843108, 14.11036586820911),
                 ('165-235N', 0.13440609108383775, 4.322975982579263)],
                dtype=[('ID', '<U8'), ('Chord', '<f8'), ('r14', '<f8')])
```

Add some more columns:

- `RA` and `Dec` corresponding to `coords0`
- Projected distance `Dprime` from th1C (in arcsec)
- Position angle `PA` from th1C (in degrees).

```
In [878]: sizedata['RA'] = coords0.ra.to_string(unit=u.hourangle, sep=':')
          sizedata['Dec'] = coords0.dec.to_string(sep=':')
          sizedata['Dprime'] = c0.separation(coords0).arcsec
          sizedata['PA'] = c0.position_angle(coords0).deg
          sizedata[-4:]

Out[878]: <Table rows=4 names=('ID','Chord','r14','RA','Dec','Dprime','PA')>
          array([ ('163-323', 0.1011089261716131, 3.2520212137679168, '5:35:16.3125', '-5:23:22.6', 2.2
                  ('153-321', 0.08000000000590013, 2.573083376229769, '5:35:15.342', '-5:23:21.31', 16.8
                  ('167-231', 0.32854879080509697, 10.567292897982984, '5:35:16.728', '-5:22:31.18', 51.8
                  ('171-334', 0.20948633602138664, 6.737822609051992, '5:35:17.0625', '-5:23:34.115', 14
                 dtype=[('ID', '<U8'), ('Chord', '<f8'), ('r14', '<f8'), ('RA', 'O'), ('Dec', 'O'), ('Dp
```

Now we try and merge in the HA98 table.

```
In [747]: from astropy.table import join, Column
```

First naive attempt was a straight outer join on the ID column. This gave some objects that are in the HA98 table but not in the new data. Of these, 244-440 is OK, since it is outside our distance cut-off, but the others need fixing. 177-341 (not in Ricci catalog) is just a matter of the name, which should be 177-341W. There were also two really missing: 171-334 and 154-324 (my typo).

So we need to fix things up. First make the ID field in the HA98 table wider and fix the name of 177-341W.

This took much longer than it should have done. It turns out that you can't change the dtype of a column without making a new table.

```
In [798]: data = Table(data, names=data.colnames, dtype=['<U12', '<f8', '<f8', '<f8'])
          data['ID'] = Column(data['ID'], dtype='<U12')
          irow = np.argmax(data['ID'] == '177-341')
          data['ID'][irow] = '177-341W'
```

So now we can do the join properly:

```
In [917]: jdata = join(sizedata, data, join_type='outer', keys='ID')
```

And the only one missing is 244-440:

```
In [918]: jdata[~jdata["r14_2"].mask & jdata["r14_1"].mask]
```

```
Out[918]: <Table rows=1 names=('ID','Chord','r14_1','RA','Dec','Dprime','PA','d','r14_2','VicR14')>
          masked_array(data = [('244-440', --, --, --, --, --, --, 142.3, 104.0, 180.6)],
                       mask = [(False, True, True, True, True, True, True, False, False, False)],
                 fill_value = ('N/A', 1e+20, 1e+20, '?', '?', 1e+20, 1e+20, 1e+20, 1e+20, 1e+20),
                      dtype = [('ID', '<U12'), ('Chord', '<f8'), ('r14_1', '<f8'), ('RA', 'O'), ('Dec',
```

Now we tidy up some of the column names:

```
In [919]: jdata["r14_1"].name = "r14"
          jdata["r14_2"].name = "r14_HA"
          jdata["d"].name = "Dprime_HA"
```

Look at the rows that are in both datasets:

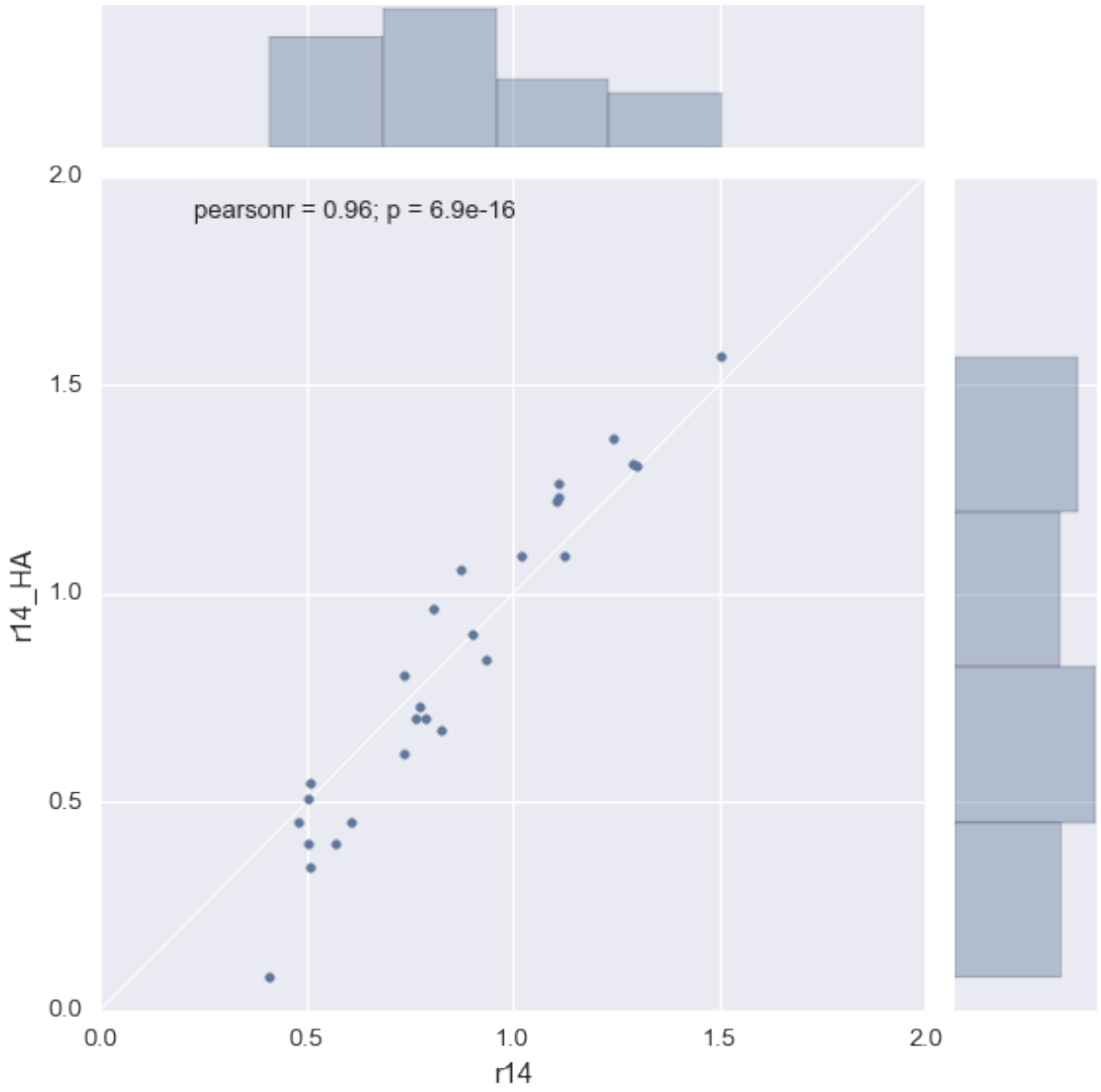
```
In [920]: overlap = ~jdata["r14"].mask & ~jdata["r14_HA"].mask
          jdata[overlap]
```

```
Out[920]: <Table rows=27 names=('ID','Chord','r14','RA','Dec','Dprime','PA','Dprime_HA','r14_HA','VicR14
          masked_array(data = [ ('152-319', 0.4021371270602539, 12.934154456590305, '5:35:15.197', '-5:2
           ('153-321', 0.08000000000590013, 2.573083376229769, '5:35:15.342', '-5:23:21.31', 16.8224369
           ('154-324', 0.09999999999592658, 3.216354219918985, '5:35:15.342', '-5:23:24.06', 16.7956685
           ('155-338', 0.4026223646938783, 12.949761417696655, '5:35:15.505', '-5:23:37.455', 20.453470
           ('157-323', 0.1000000000016516, 3.2163542201031214, '5:35:15.705', '-5:23:22.45', 11.3380074
           ('158-323', 0.1706546658700654, 5.488858547424075, '5:35:15.8045', '-5:23:22.455', 9.8529615
           ('158-326', 0.23232597535592958, 7.472426312632764, '5:35:15.8285', '-5:23:25.54', 9.8610873
           ('158-327', 0.4002366670401905, 12.873028930534636, '5:35:15.7755', '-5:23:26.585', 10.93625
           ('159-338', 0.19157551869142098, 6.161747280014195, '5:35:15.886', '-5:23:38.08', 17.5053947
           ('159-350', 0.625922271425744, 20.13187739123473, '5:35:15.9375', '-5:23:50.125', 28.3860230
           ('161-314', 0.18604946713139633, 5.9840098874612035, '5:35:16.095', '-5:23:14.055', 10.37556
```

```
('161-324', 0.10110892444353344, 3.252021158186754, '5:35:16.0435', '-5:23:24.31', 6.4438585
('161-328', 0.19956873670361883, 6.418837484867318, '5:35:16.0625', '-5:23:27.805', 7.776427
('163-317', 0.18169847973921033, 5.844066720858787, '5:35:16.27', '-5:23:16.325', 7.13645181
('163-323', 0.1011089261716131, 3.2520212137679168, '5:35:16.3125', '-5:23:22.6', 2.27242083
('166-316', 0.11587838074598988, 3.7270591892492524, '5:35:16.6065', '-5:23:15.955', 7.21563
('167-317', 0.24955691955806011, 8.026634513632452, '5:35:16.7455', '-5:23:16.245', 7.830071
('168-328', 0.1269980216773827, 4.084706229600512, '5:35:16.76', '-5:23:28.155', 6.908720522
('169-338', 0.09449034060536078, 3.03914405760014, '5:35:16.883', '-5:23:38.095', 16.4818823
('170-337', 0.4162404272647277, 13.38776654788322, '5:35:16.98', '-5:23:37.205', 16.2954814
('171-334', 0.20948633602138664, 6.737822609051992, '5:35:17.0625', '-5:23:34.115', 14.38340
('171-340', 0.5462728212701063, 17.57006893990726, '5:35:17.0555', '-5:23:39.805', 19.120982
('173-341', 0.16941402292211163, 5.448955075611812, '5:35:17.327', '-5:23:41.42', 22.6072175
('176-325', 0.2699999999970833, 8.684156394041189, '5:35:17.569', '-5:23:24.785', 16.6185851
('177-341W', 0.6072973409169007, 19.532833652832174, '5:35:17.684', '-5:23:41.095', 25.78757
('180-331', 0.3261921659047774, 10.491495493550804, '5:35:18.057', '-5:23:30.745', 25.069100
('182-413', 0.9989495547792666, 32.12975616131363, '5:35:18.2155', '-5:24:13.155', 56.701578
             mask = [(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)
(False, False, False, False, False, False, False, False, False, False)],
       fill_value = ('N/A', 1e+20, 1e+20, '?', '?', 1e+20, 1e+20, 1e+20, 1e+20, 1e+20),
            dtype = [('ID', '<U12'), ('Chord', '<f8'), ('r14', '<f8'), ('RA', 'O'), ('Dec', '
```

Check that the two estimates of the proplyd sizes are not too different.

```
In [895]: x, y = [np.log10(jdata[overlap][colname]) for colname in ['r14', 'r14_HA']]
          g = sns.jointplot(x, y, size=8)
          g.ax_joint.plot([0, 2], [0, 2], '-w', zorder=-100, lw=1)
          g.ax_joint.set_xlim(0.0, 2.0)
          g.ax_joint.set_ylim(0.0, 2.0);
```



So it doesn't look too bad. Here are the cases where HA98 got >30% larger sizes. These are mainly large sized.

```
In [884]: jdata[overlap][jdata[overlap]['r14_HA'] > 1.3*jdata[overlap]['r14']]['ID', 'r14', 'r14_HA']
```

```
Out[884]: <Table rows=5 names=('ID','r14','r14_HA')>
          masked_array(data = [('152-319', 12.934154456590305, 18.2)
           ('155-338', 12.949761417696655, 17.0) ('158-326', 7.472426312632764, 11.3)
           ('161-328', 6.418837484867318, 9.1) ('171-340', 17.57006893990726, 23.3)],
                        mask = [(False, False, False) (False, False, False) (False, False, False)
           (False, False, False) (False, False, False)],
                 fill_value = ('N/A', 1e+20, 1e+20),
                      dtype = [('ID', '<U12'), ('r14', '<f8'), ('r14_HA', '<f8')])
```

And here are the cases where HA98 sizes were more than 30% smaller. These are mainly small sized. In this case, I think that the current results are more reliable because in HA98 we did not remove the central stars and so overestimated the compactness of some of the small proplyds.

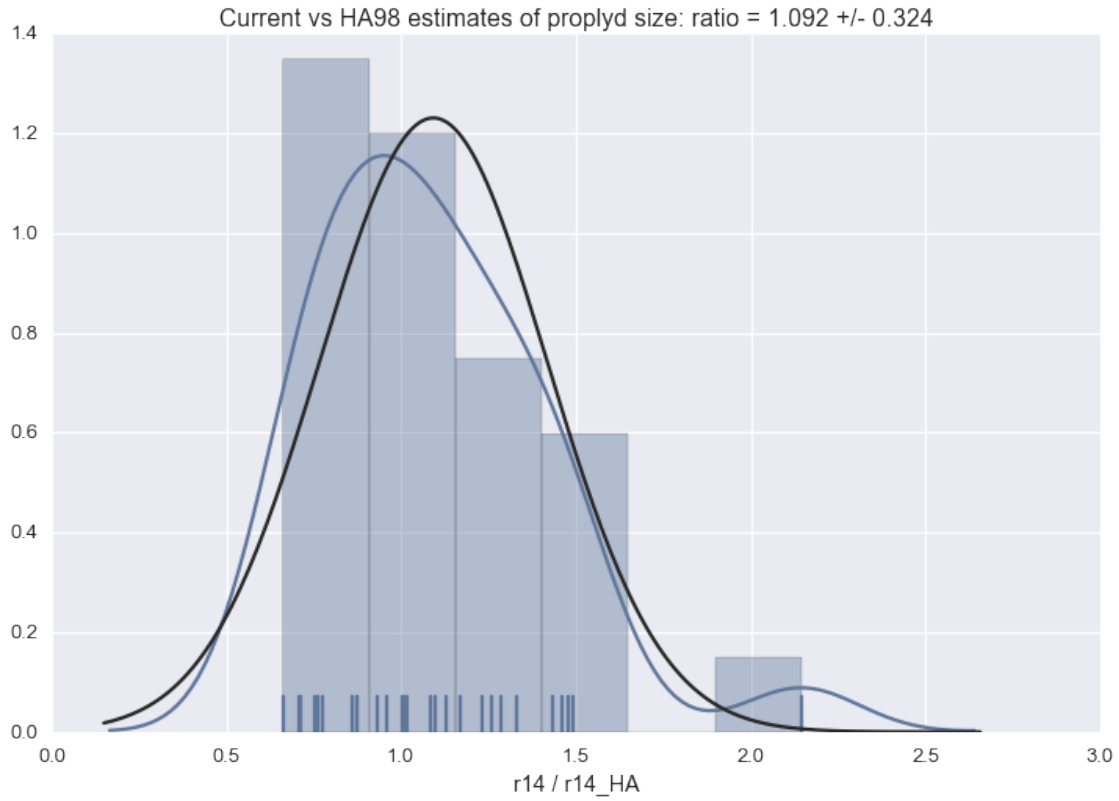
```
In [887]: jdata[overlap][jdata[overlap]['r14'] > 1.3*jdata[overlap]['r14_HA']]['ID', 'r14', 'r14_HA']
```

```
Out[887]: <Table rows=6 names=('ID','r14','r14_HA')>
          masked_array(data = [('153-321', 2.573083376229769, 1.2) ('163-323', 3.2520212137679168, 2.2)
           ('166-316', 3.7270591892492524, 2.5) ('168-328', 4.084706229600512, 2.8)
           ('171-334', 6.737822609051992, 4.7) ('173-341', 5.448955075611812, 4.1)],
                       mask = [(False, False, False) (False, False, False) (False, False, False)
           (False, False, False) (False, False, False) (False, False, False)],
                 fill_value = ('N/A', 1e+20, 1e+20),
                      dtype = [('ID', '<U12'), ('r14', '<f8'), ('r14_HA', '<f8')])
```

If we look at the distribution of ratios, then it seems we have agreement to a level of about 30%. This is not bad, given that the sizes range over an order of magnitude.

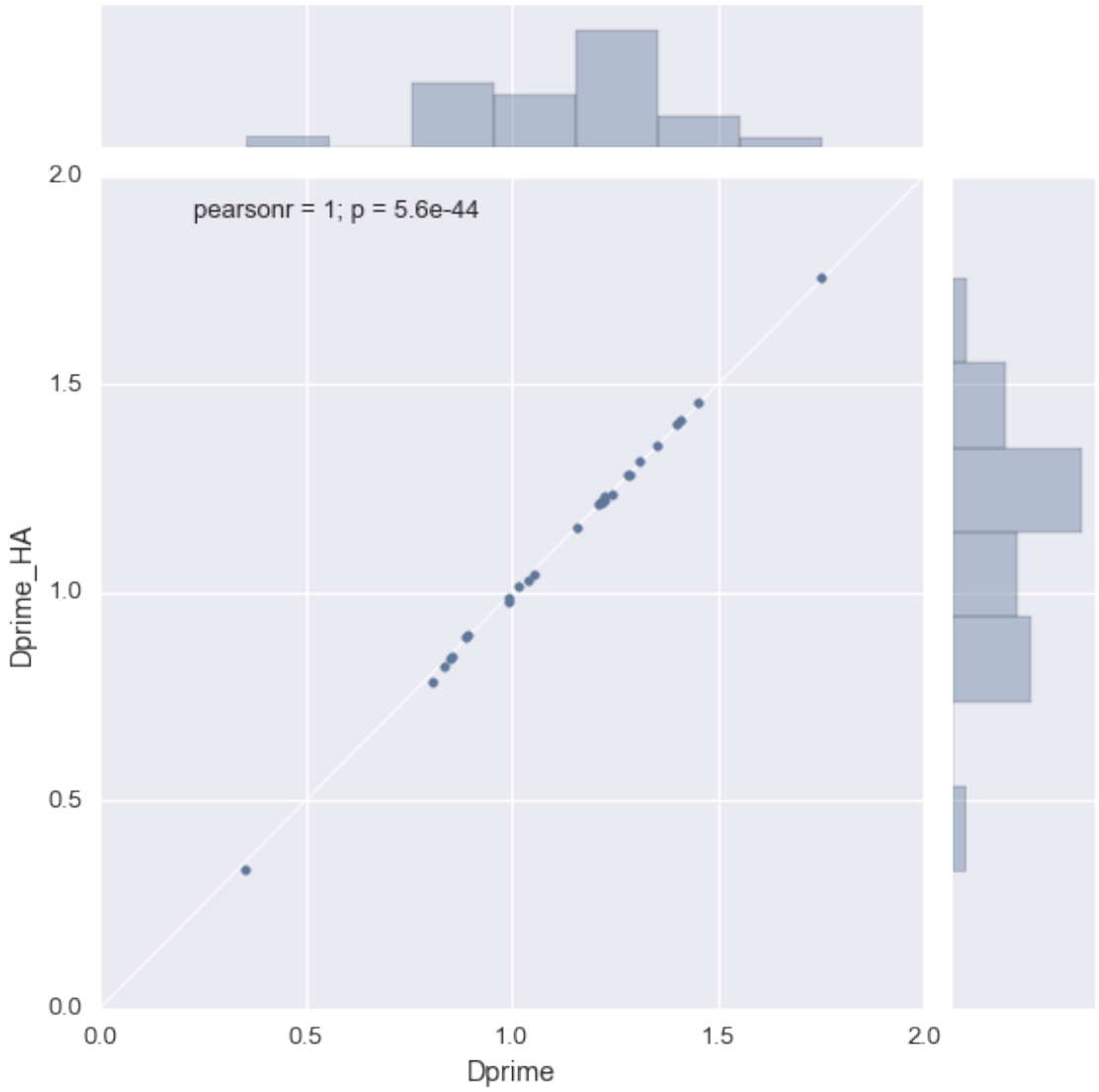
```
In [898]: ratio = jdata['r14']/jdata['r14_HA']
          sns.distplot(ratio[overlap], rug=True, fit=stats.norm, axlabel='r14 / r14_HA')
          plt.title('Current vs HA98 estimates of proplyd size:'
                    ' ratio = {:.3f} +/- {:.3f}'.format(ratio[overlap].mean(),
                                                        ratio[overlap].std()))
```

```
Out[898]: <matplotlib.text.Text at 0x1575874d0>
```



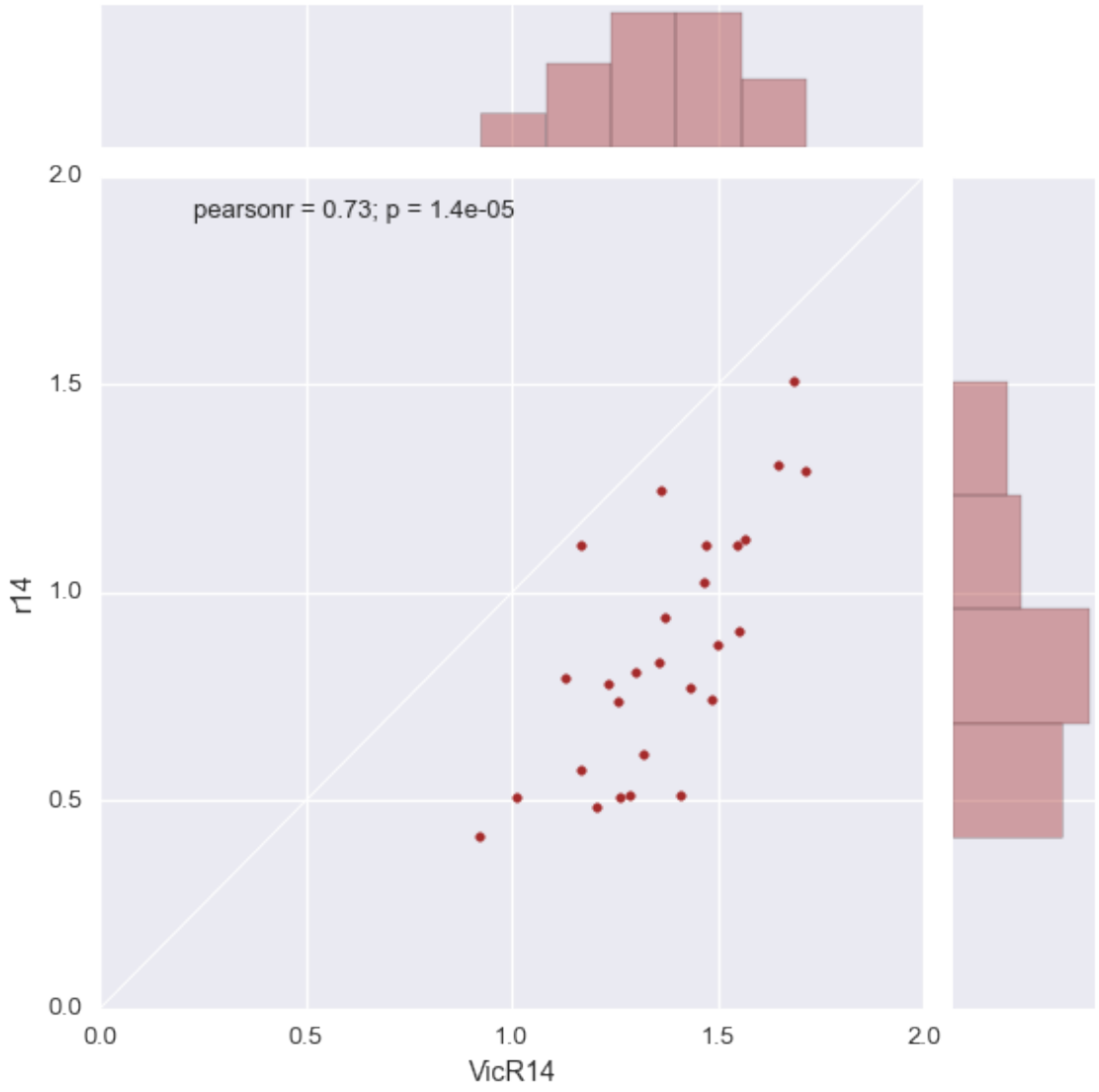
We can do the same thing for the separations. This shows a much tighter correlation, as one would expect

```
In [900]: x, y = [np.log10(jdata[overlap][colname]) for colname in ['Dprime', 'Dprime_HA']]
          g = sns.jointplot(x, y, size=8)
          g.ax_joint.plot([0, 2], [0, 2], '-w', zorder=-100, lw=1)
          g.ax_joint.set_xlim(0.0, 2.0)
          g.ax_joint.set_ylim(0.0, 2.0);
```



And just for fun, here is the same plot for the Vicente & Alves (2005) sizes.

```
In [896]: x, y = [np.log10(jdata[overlap][colname]) for colname in ['VicR14', 'r14']]
          g = sns.jointplot(x, y, color='brown', size=8)
          g.ax_joint.plot([0, 2], [0, 2], '-w', zorder=-100, lw=1)
          g.ax_joint.set_xlim(0.0, 2.0)
          g.ax_joint.set_ylim(0.0, 2.0);
```



Finally, we can go back to the size-distance relationship:

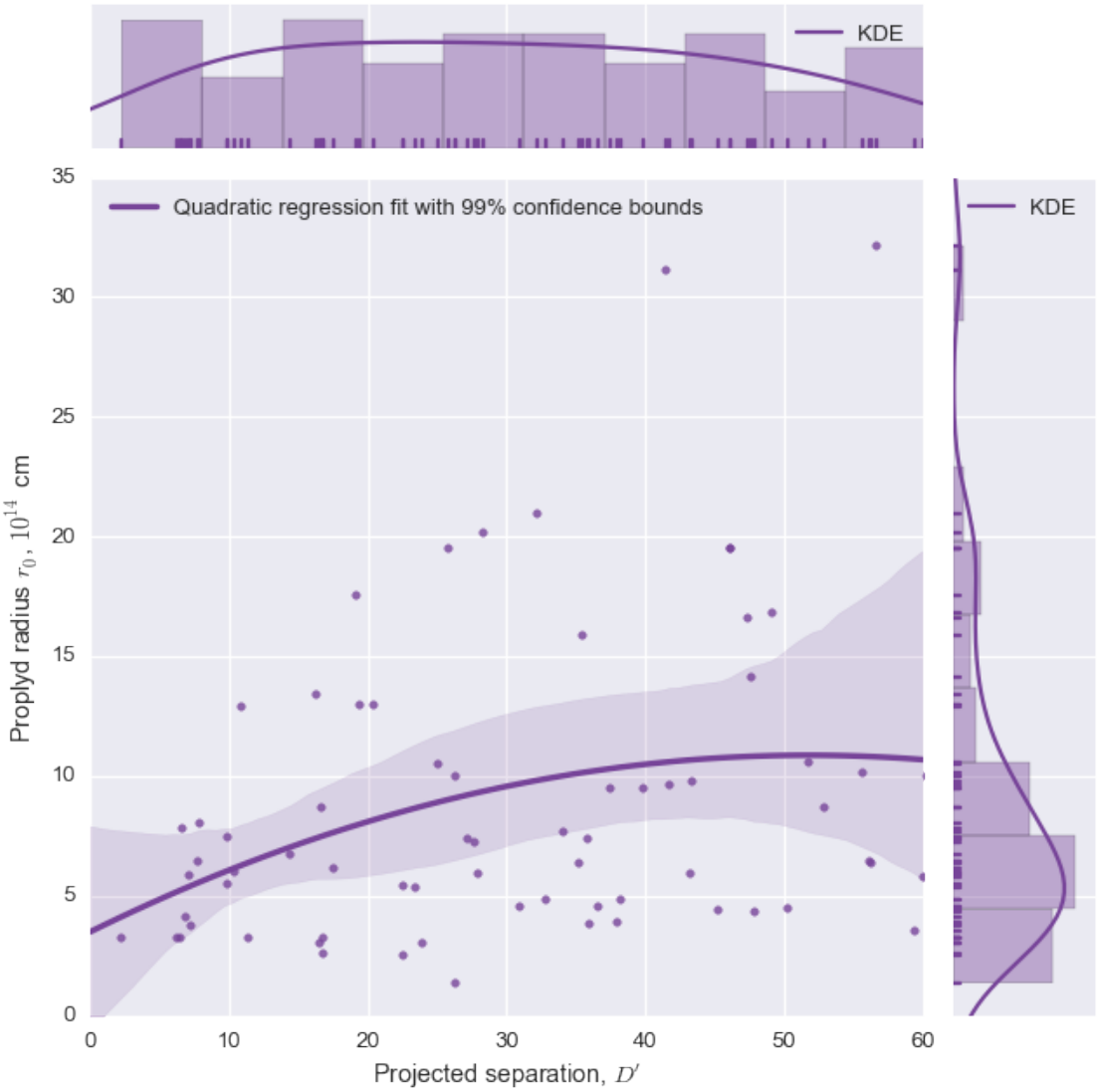
```
In [923]: jdata.filled(np.nan)
```

```
Out[923]: <Table rows=70 names=('ID','Chord','r14','RA','Dec','Dprime','PA','Dprime_HA','r14_HA','VicR14
          array([ ('133-353', 0.19956810372609624, 6.418817126068057, '5:35:13.2955', '-5:23:53.055', 5
                  ('139-320', 0.15074155191820304, 4.848382266489986, '5:35:13.9075', '-5:23:20.045', 38
                  ('142-301', 0.9679778688332578, 31.13359703336854, '5:35:14.1165', '-5:23:00.71', 41.45
                  ('147-323', 0.3103594930344309, 9.982260651538704, '5:35:14.7055', '-5:23:22.775', 26.1
                  ('148-305', 0.14187554655648482, 4.563220128888503, '5:35:14.79', '-5:23:04.565', 30.90
                  ('149-329', 0.0948264445804458, 3.049954351986543, '5:35:14.914', '-5:23:28.995', 23.94
                  ('150-231', 0.19822035630358525, 6.37546879496851, '5:35:15.0175', '-5:22:30.83', 56.31
                  ('152-319', 0.4021371270602539, 12.934154456590305, '5:35:15.197', '-5:23:18.66', 19.37
                  ('153-321', 0.08000000000590013, 2.573083376229769, '5:35:15.342', '-5:23:21.31', 16.81
```

```
('154-225', 0.1803214866471138, 5.799777745431342, '5:35:15.362', '-5:22:25.15', 59.998
('154-240', 0.6056714671056789, 19.48053979189225, '5:35:15.374', '-5:22:39.66', 46.153
('154-324', 0.09999999999592658, 3.216354219918985, '5:35:15.342', '-5:23:24.06', 16.79
('155-338', 0.4026223646938783, 12.949761417696655, '5:35:15.505', '-5:23:37.455', 20.4
('157-323', 0.1000000000016516, 3.2163542201031214, '5:35:15.705', '-5:23:22.45', 11.33
('158-323', 0.1706546658700654, 5.488858547424075, '5:35:15.8045', '-5:23:22.455', 9.85
('158-326', 0.23232597535592958, 7.472426312632764, '5:35:15.8285', '-5:23:25.54', 9.86
('158-327', 0.4002366670401905, 12.873028930534636, '5:35:15.7755', '-5:23:26.585', 10
('159-338', 0.19157551869142098, 6.161747280014195, '5:35:15.886', '-5:23:38.08', 17.50
('159-350', 0.625922271425744, 20.13187739123473, '5:35:15.9375', '-5:23:50.125', 28.38
('159-418', 0.3142365584879447, 10.106960809866894, '5:35:15.8965', '-5:24:17.87', 55.0
('160-350', 0.18361197284160802, 5.905611437008119, '5:35:15.958', '-5:23:49.7', 27.893
('161-314', 0.18604946713139633, 5.9840098874612035, '5:35:16.095', '-5:23:14.055', 10
('161-324', 0.10110892444353344, 3.252021158186754, '5:35:16.0435', '-5:23:24.31', 6.44
('161-328', 0.19956873670361883, 6.418837484867318, '5:35:16.0625', '-5:23:27.805', 7.1
('163-249', 0.23915181790427623, 7.691969587490479, '5:35:16.332', '-5:22:48.795', 34.1
('163-317', 0.18169847973921033, 5.844066720858787, '5:35:16.27', '-5:23:16.325', 7.130
('163-323', 0.1011089261716131, 3.2520212137679168, '5:35:16.3125', '-5:23:22.6', 2.271
('165-235N', 0.13440609108383775, 4.322975982579263, '5:35:16.4835', '-5:22:34.97', 47
('165-235S', 0.4387068370843108, 14.11036586820911, '5:35:16.4725', '-5:22:35.255', 47
('166-250', 0.1493390719216531, 4.803273541935594, '5:35:16.586', '-5:22:50.07', 32.829
('166-316', 0.11587838074598988, 3.7270591892492524, '5:35:16.6065', '-5:23:15.955', 7
('166-406', 0.18361065843219077, 5.905569160945359, '5:35:16.569', '-5:24:06.1', 43.279
('167-231', 0.32854879080509697, 10.567292897982984, '5:35:16.728', '-5:22:31.18', 51.8
('167-317', 0.24955691955806011, 8.026634513632452, '5:35:16.7455', '-5:23:16.245', 7.8
('168-326E', 0.2435727716841236, 7.834163120955061, '5:35:16.8425', '-5:23:26.36', 6.65
('168-326W', 0.1005342149006856, 3.2335364635523374, '5:35:16.8225', '-5:23:25.985', 6
('168-328', 0.1269980216773827, 4.084706229600512, '5:35:16.76', '-5:23:28.155', 6.908
('168-404', 0.3001709084357675, 9.654559680836229, '5:35:16.758', '-5:24:04.305', 41.68
('169-338', 0.09449034060536078, 3.03914405760014, '5:35:16.883', '-5:23:38.095', 16.48
('170-249', 0.4937316418995504, 15.880158499958343, '5:35:16.9675', '-5:22:48.135', 35
('170-301', 0.16698907149310396, 5.370960047990761, '5:35:16.9735', '-5:23:00.685', 23
('170-337', 0.4162404272647277, 13.38776654788322, '5:35:16.98', '-5:23:37.205', 16.295
('170-400', 0.2947976834595547, 9.48173773256103, '5:35:16.9535', '-5:23:59.55', 37.423
('171-334', 0.20948633602138664, 6.737822609051992, '5:35:17.0625', '-5:23:34.115', 14
('171-340', 0.5462728212701063, 17.57006893990726, '5:35:17.0555', '-5:23:39.805', 19.1
('173-236', 0.521743596411042, 16.78112218100719, '5:35:17.349', '-5:22:35.5', 49.1595
('173-341', 0.16941402292211163, 5.448955075611812, '5:35:17.327', '-5:23:41.42', 22.60
('174-305', 0.07789927861626998, 2.505516735162907, '5:35:17.368', '-5:23:04.715', 22.0
('174-400', 0.2947975908462822, 9.48173475379013, '5:35:17.3775', '-5:24:00.29', 39.850
('174-414', 0.26954302964271354, 8.66945860876404, '5:35:17.385', '-5:24:13.97', 52.940
('175-251', 0.19821860324606935, 6.375412410429117, '5:35:17.4775', '-5:22:51.03', 35.1
('175-355', 0.22822885465100226, 7.340648397939293, '5:35:17.5345', '-5:23:54.97', 35.8
('176-252', 0.11947118063428656, 3.8426163600744343, '5:35:17.641', '-5:22:51.48', 35.9
('176-325', 0.2699999999970833, 8.684156394041189, '5:35:17.569', '-5:23:24.785', 16.6
('177-341E', 0.042332549361707744, 1.3615647378500364, '5:35:17.736', '-5:23:41.065', 1
('177-341W', 0.6072973409169007, 19.532833652832174, '5:35:17.684', '-5:23:41.095', 25
('178-258', 0.6506124739054455, 20.9260017606295, '5:35:17.8295', '-5:22:57.93', 32.201
('179-354', 0.12011790445720019, 3.8634172890447855, '5:35:17.965', '-5:23:53.53', 37.9
('180-331', 0.3261921659047774, 10.491495493550804, '5:35:18.057', '-5:23:30.745', 25.0
('181-247', 0.30308283098129235, 9.748217424513806, '5:35:18.084', '-5:22:46.9', 43.333
('181-401', 0.13588072788920955, 4.3704055256992485, '5:35:18.0645', '-5:24:01.23', 45
('182-316', 0.2245153617232573, 7.221209311446509, '5:35:18.2485', '-5:23:15.48', 27.65
('182-316', 0.2283166074842956, 7.343470839896137, '5:35:18.187', '-5:23:31.455', 27.13
```

```
       ('182-413', 0.9989495547792666, 32.12975616131363, '5:35:18.2155', '-5:24:13.155', 56.
       ('189-329', 0.14079422461206662, 4.52844098489688, '5:35:18.8795', '-5:23:28.82', 36.5
       ('190-251', 0.13794221136562448, 4.4367101364885535, '5:35:19.0345', '-5:22:50.465', 5
       ('191-350', 0.5153676065060261, 16.576047760627247, '5:35:19.075', '-5:23:49.68', 47.3
       ('204-330', 0.10999999999952767, 3.537989642039808, '5:35:20.42', '-5:23:29.655', 59.4
       ('205-330', 0.31035949075157415, 9.982260578113944, '5:35:20.4725', '-5:23:29.725', 60
       ('244-440', nan, nan, nan, nan, nan, nan, 142.3, 104.0, 180.6)],
      dtype=[('ID', '<U12'), ('Chord', '<f8'), ('r14', '<f8'), ('RA', 'O'), ('Dec', 'O'), ('D
```

In [931]:
```
color = "#774499"
# g = sns.JointGrid(mirrored(jdata["Dprime"].filled(0.0)), doubled(jdata["r14"].filled(0.0)),
g = sns.JointGrid(jdata["Dprime"], jdata["r14"],
                  size=9, xlim=(0, 60), ylim=(0, 35))
# g.plot(sns.regplot, sns.distplot, stats.pearsonr);
g.plot_marginals(sns.distplot, color=color, bins=10, rug=True, kde_kws=dict(label="KDE"))
g.plot_joint(sns.regplot, color=color, ci=99, order=2, label="Quadratic regression fit with 9
g.ax_joint.legend(loc="upper left")
g.ax_joint.set_xlabel("Projected separation, $D'$")
g.ax_joint.set_ylabel("Proplyd radius $r_0$, $10^{14}$ cm")
g.ax_marg_x.legend(loc="upper right")
g.ax_marg_y.legend(loc="best")
g.fig.tight_layout()
g.fig.savefig("proplyd-joint-separation-size-new.pdf");
```




```
In []:

In []:

In []:

In []:

In [743]: data[data['ID'] == '177-341']['ID']

Out[743]: <Column name='ID' unit=None format=None description=None>
          array(['177-341'],
                dtype='<U7')

In [897]: sns.jointplot?

In []:
```

```
In []:

In [729]: sizedata[::10]

Out[729]: <Table rows=7 names=('ID','Chord','r14','RA','Dec','Dprime','PA')>
          array([ ('154-240', 0.6056714671056789, 19.48053979189225, '5:35:15.374', '-5:22:39.66', 46.15
                  ('174-305', 0.07789927861626998, 2.505516735162907, '5:35:17.368', '-5:23:04.715', 22.(
                  ('149-329', 0.0948264445804458, 3.049954351986543, '5:35:14.914', '-5:23:28.995', 23.94
                  ('177-341E', 0.042332549361707744, 1.3615647378500364, '5:35:17.736', '-5:23:41.065', 1
                  ('190-251', 0.13794221136562448, 4.4367101364885535, '5:35:19.0345', '-5:22:50.465', 50
                  ('158-326', 0.1726037583473396, 5.551548265569561, '5:35:15.824', '-5:23:25.565', 9.931
                  ('174-414', 0.26954302964271354, 8.66945860876404, '5:35:17.385', '-5:24:13.97', 52.940
                 dtype=[('ID', '<U8'), ('Chord', '<f8'), ('r14', '<f8'), ('RA', 'O'), ('Dec', 'O'), ('Dpr

In []:
```

```
In [719]: coord.SkyCoord.to_string?

In [720]: coords0[:4].to_string('hmsdms')

Out[720]: ['05h35m15.374s -05d22m39.66s',
           '05h35m16.332s -05d22m48.795s',
           '05h35m16.4725s -05d22m35.255s',
           '05h35m16.4835s -05d22m34.97s']

In [721]: coord.Angle.to_string?

In [723]: coords0[:4].ra.to_string(unit=u.hourangle)

Out[723]: array(['5h35m15.374s', '5h35m16.332s', '5h35m16.4725s', '5h35m16.4835s'], dtype=object)

In [588]: plt.scatter?

In [672]: sns.kdeplot?

In [79]: import matplotlib

In [583]: matplotlib.markers?

In [600]: plt.legend?

In [93]: f.axes[0] == fig._ax1

Out[93]: True

In [242]: r = np.random.RandomState(5)

In [243]: r.random_sample(10)

Out[243]: array([ 0.22199317,  0.87073231,  0.20671916,  0.91861091,  0.48841119,
                  0.61174386,  0.76590786,  0.51841799,  0.2968005 ,  0.18772123])

In [182]: np.r_[[1, 2, 3], 0, 0, [1, 2]]

Out[182]: array([1, 2, 3, 0, 0, 1, 2])

In [244]: import seaborn

In [249]: seaborn.palettes.dark_palette('red', as_cmap=True)

Out[249]: <matplotlib.colors.LinearSegmentedColormap at 0x111cb7490>

In []:
```