Theodore Koby-Hercsky
DSC530-T302

10.2 Exercises

Exercises

Exercise: In NSFG Cycles 6 and 7, the variable `cmdivorcx` contains the date of divorce for the respondent's first marriage, if applicable, encoded in century-months.

Compute the duration of marriages that have ended in divorce, and the duration, so far, of marriages that are ongoing. Estimate the hazard and survival curve for the duration of marriage.

Use resampling to take into account sampling weights, and plot data from several resamples to visualize sampling error.

Consider dividing the respondents into groups by decade of birth, and possibly by age at first marriage.

In [68]:
```
def CleanData(resp):
    """Cleans respondent data.

    resp: DataFrame
    """
    resp.cmdivorcx.replace([9998, 9999], np.nan, inplace=True)

    resp['notdivorced'] = resp.cmdivorcx.isnull().astype(int)
    resp['duration'] = (resp.cmdivorcx - resp.cmmarrhx) / 12.0
    resp['durationsofar'] = (resp.cmintvw - resp.cmmarrhx) / 12.0

    month0 = pd.to_datetime('1899-12-15')
    dates = [month0 + pd.DateOffset(months=cm)
             for cm in resp.cmbirth]
    resp['decade'] = (pd.DatetimeIndex(dates).year - 1900) // 10
```

In [69]:
```
CleanData(resp6)
married6 = resp6[resp6.evrmarry==1]

CleanData(resp7)
married7 = resp7[resp7.evrmarry==1]
```

In [80]:
```
# First I will create a for loop that will have a range of 15 which
# I will be working off the for loop from the more data section
# above to help build my code. That uses the respondent data frames
# to help plot the curve.

def ResampleSurvivalCurve(resps):
    for _ in range(15):
        samples = [thinkstats2.ResampleRowsWeighted(resp)
                   for resp in resps]
        sample = pd.concat(samples, ignore_index=True)
        PlotDivorceCurveByDecade(sample, color='gray', alpha=0.1)
    thinkplot.Show(xlabel='years', axis=[0, 35, 0.2, 1.2])
```

In [85]:
```
# Next I will create another for loop and use the range function
# in my defined function that uses the resps data frames that will
# plot our curves that are for each birth by going off the code
# used in the more data section.

def ResampleSurvivalDecadeCurve(resps):
    for i in range(50):
        samples = [thinkstats2.ResampleRowsWeighted(resp)
                   for resp in resps]
        sample = pd.concat(samples, ignore_index=True)
        groups = sample.groupby('decade')
        if i == 0:
            survival.AddLabelsByDecade(groups, alpha=0.7)
        EstimateSurvivalByDecade(groups, alpha=0.1)
    thinkplot.Config(xlabel='By Year', ylabel='Not Divorced', axis=[0, 3
```

In [86]:
```
# Next I will define each group by a decade while plotting the curves
# By working off the more data section above for the EstimateMarriageSur
# to help build my code that groups by objects.
def EstimateSurvivalByDecade(groups, **options):
    thinkplot.PrePlot(len(groups))
    for name, group in groups:
        _, survival_function = EstimateCompleteandOngoing(group)
        thinkplot.Plot(survival_function, **options)
```

In [87]:
```
# As for the last defined we will use the resp data frame that will
# give us the survival and hazard function that the
# EstimateHazardFunction is defined above that the function uses
# Kaplan-Meier to estimate the hazard functionby going off a similar
# code above in the EstimateMarriageSurvival and create the
# hazard_function and survival_function

def EstimateCompleteandOngoing(resp):
    complete = resp[resp.notdivorced == 0].duration.dropna()
    ongoing = resp[resp.notdivorced == 1].durationsofar.dropna()
    hazard_function = survival.EstimateHazardFunction(complete, ongoing)
    survival_function = hazard_function.MakeSurvival()

    return hazard_function, survival_function
```

In [88]:
```
# Last we will create our plot from the ResampleSurvivalDecadeCurve
# that will use the married 6 and 7 that was already created above
ResampleSurvivalDecadeCurve([married6, married7])
```