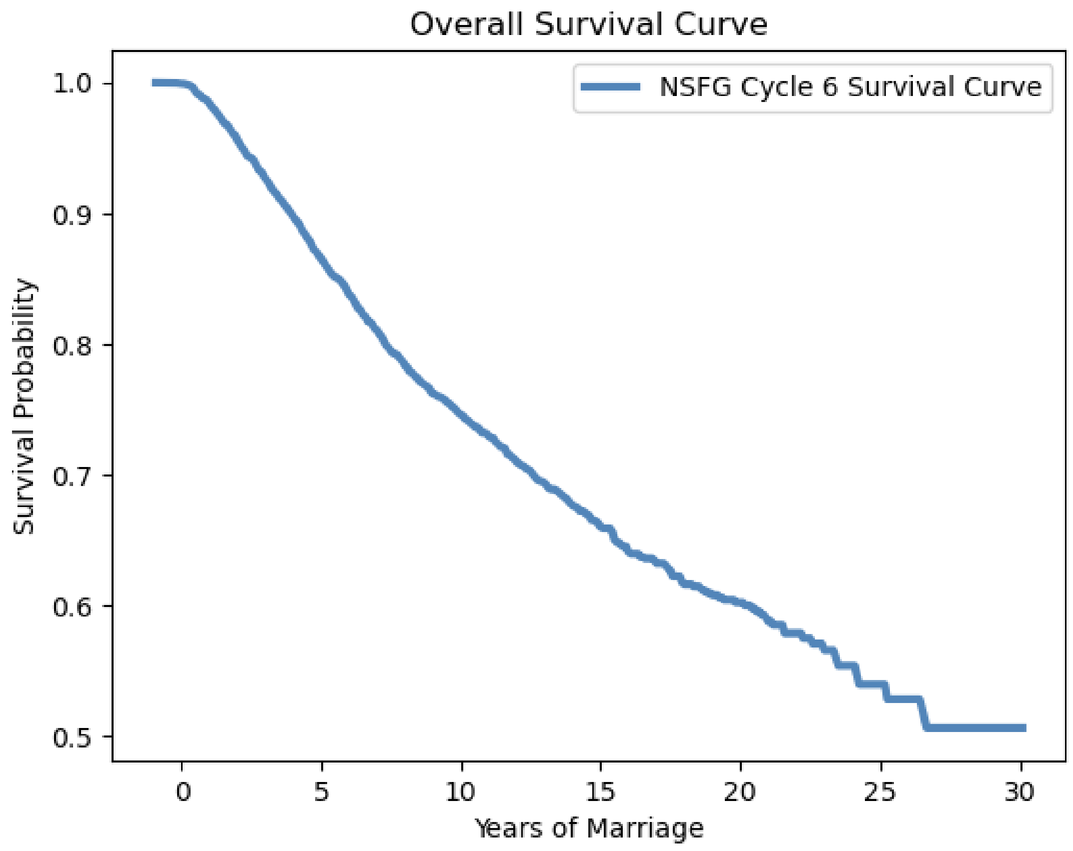
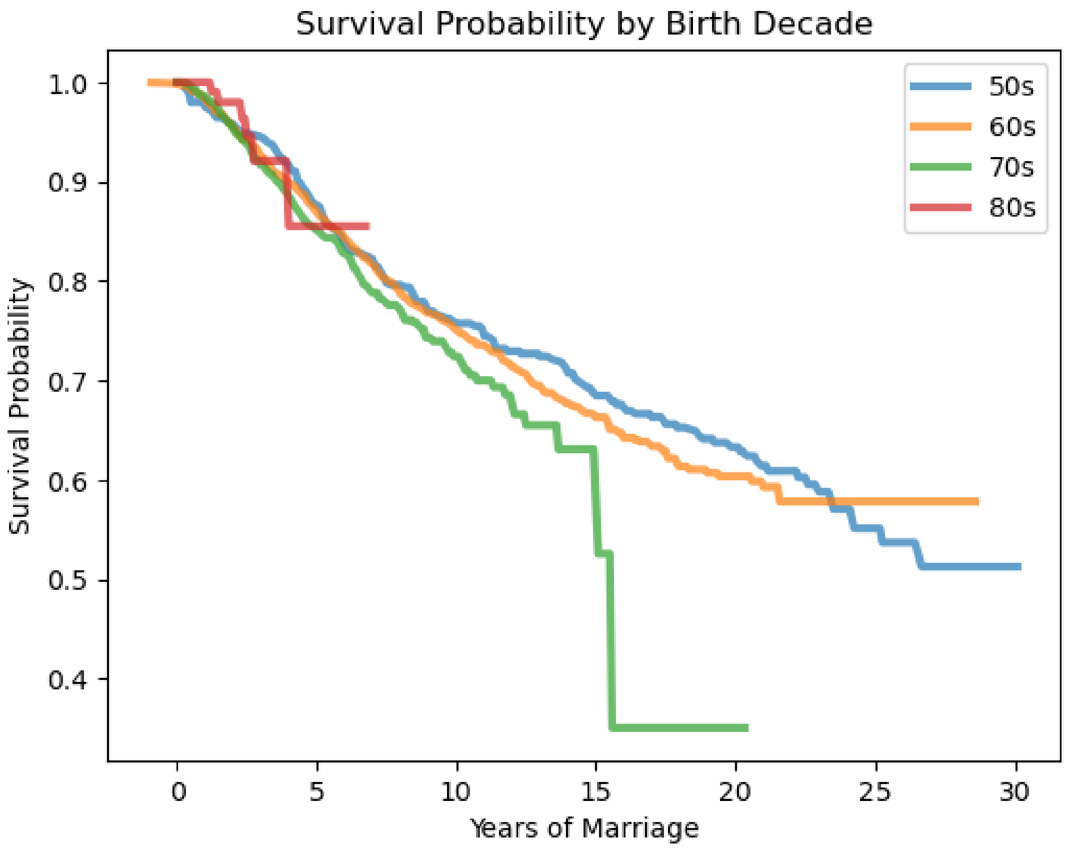
Bilenkin530Week11_Exercises_11.2

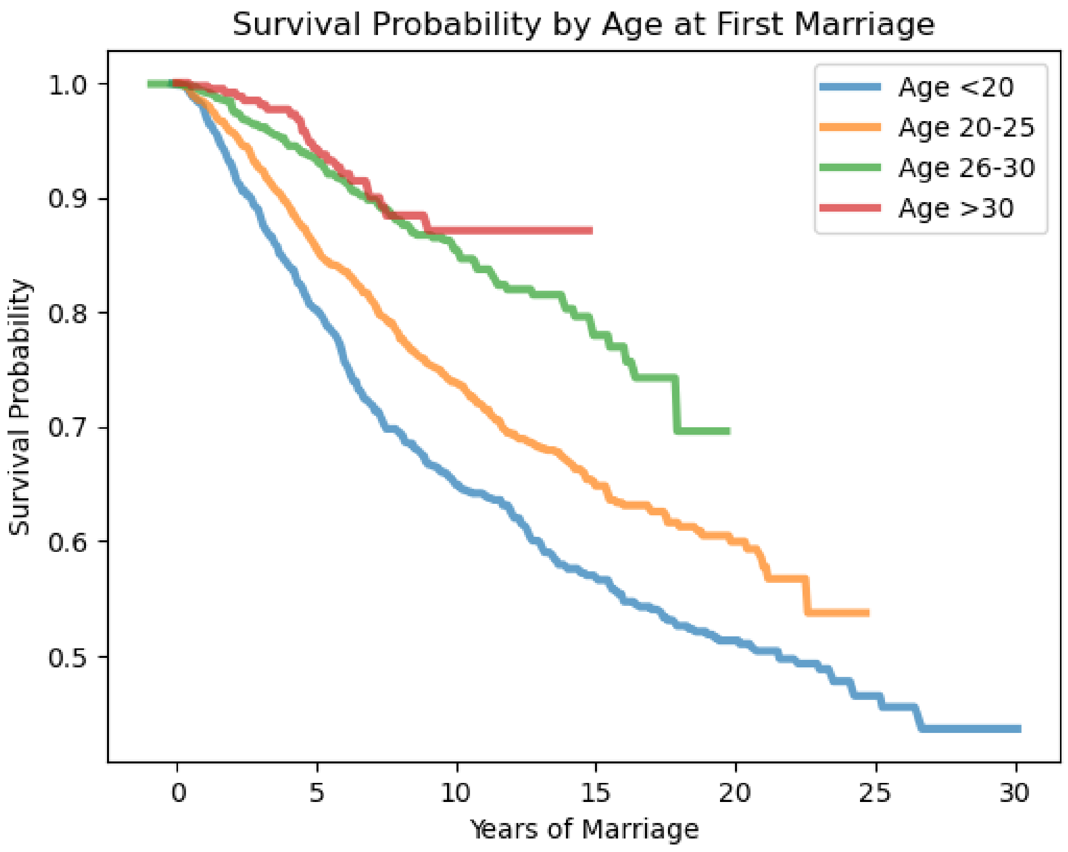
February 23, 2025

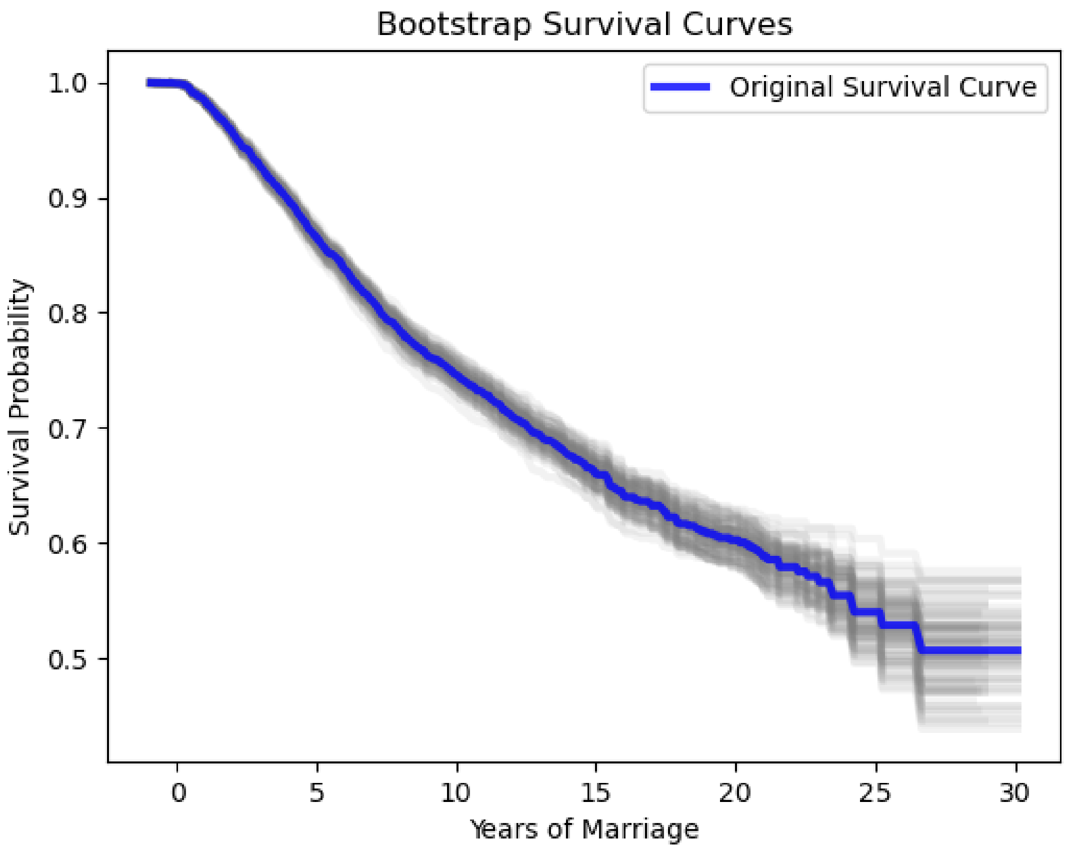
Chapter 13 (Pages 180-181: Exercise 13-1)

[88]:
```
# Downloading necessary files
from os.path import basename, exists
import urllib.request

def download(url):
    filename = basename(url)
    if not exists(filename):
        local, _ = urllib.request.urlretrieve(url, filename)
        print("Downloaded " + local)

urls = [
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/survival.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.
 ↪dct",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dat.
 ↪gz",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2006_2010_FemRespSetup.dct",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪2006_2010_FemResp.dat.gz"
]

for url in urls:
    download(url)

# Importing necessary libraries and reading data
import pandas as pd
import numpy as np
import nsfg
import survival
import thinkstats2
import thinkplot
import matplotlib.pyplot as plt
import seaborn as sns

response_6 = survival.ReadFemResp2002()
response_7 = survival.ReadFemResp2010()

response_6['cmmarrhx'] = response_6['cmmarrhx'].replace({9997: np.nan, 9998: np.
 ↪nan, 9999: np.nan})
response_7['cmmarrhx'] = response_7['cmmarrhx'].replace({9997: np.nan, 9998: np.
 ↪nan, 9999: np.nan})

# Cleaning and filtering data
def CleanData(resp):
    resp = resp.copy()
    resp['cmdivorcx'] = resp['cmdivorcx'].replace({9998: np.nan, 9999: np.nan})
    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
    resp['age_at_marriage'] = (resp['cmmarrhx'] - resp['cmbirth']) / 12.0
    bins = [0, 20, 25, 30, np.inf]
    labels = ['<20', '20-25', '26-30', '>30']
    resp['age_group'] = pd.cut(resp['age_at_marriage'], bins=bins,␣
 ↪labels=labels, right=False)
    return resp

married_6 = CleanData(response_6).query("evrmarry == 1")
married_7 = CleanData(response_7).query("evrmarry == 1")

# Estimating survival functions
def EstimateSurvival(resp):
    complete = resp.loc[resp["notdivorced"] == 0, "duration"].dropna()
    ongoing = resp.loc[resp["notdivorced"] == 1, "durationsofar"].dropna()
    hf = survival.EstimateHazardFunction(complete, ongoing)
    sf = hf.MakeSurvival()
    return hf, sf

hf, sf = EstimateSurvival(married_6)

# Plotting functions

# Setting color palette
colors = sns.color_palette("tab10")

def PlotSurvivalCurve(sf, label="NSFG Cycle 6 Survival Curve"):
    thinkplot.PrePlot(1)
    thinkplot.Plot(sf.ts, sf.ss, label=label, alpha=0.7)
    plt.xlabel("Years of Marriage")
    plt.ylabel("Survival Probability")
    plt.title("Overall Survival Curve")
    plt.legend(loc='best')
    plt.show()

def CompareByDecade(resp):
    grouped = resp.groupby("decade")
    colors = sns.color_palette("tab10", n_colors=len(grouped))  # creating␣
 ↪different colors

    thinkplot.PrePlot(len(grouped))
    for i, (decade, group) in enumerate(grouped):
        _, sf = EstimateSurvival(group)
        thinkplot.Plot(sf.ts, sf.ss, label=f"{int(decade * 10)}s",␣
 ↪color=colors[i])

    plt.xlabel("Years of Marriage")
    plt.ylabel("Survival Probability")
    plt.title("Survival Probability by Birth Decade")
    plt.legend(loc='best')
    plt.show()

def CompareByAgeAtMarriage(resp):
    grouped = resp.groupby("age_group", observed=False)
    colors = sns.color_palette("tab10", n_colors=len(grouped))  # creating␣
 ↪different colors

    thinkplot.PrePlot(len(grouped))
    for i, (age_group, group) in enumerate(grouped):
        _, sf = EstimateSurvival(group)
        thinkplot.Plot(sf.ts, sf.ss, label=f"Age {age_group}", color=colors[i])

    thinkplot.Config(
        xlabel="Years of Marriage",
        ylabel="Survival Probability",
        title="Survival Probability by Age at First Marriage",
        legend=True
    )
    plt.show()

def BootstrapSurvival(resp, num_samples=100):
    thinkplot.PrePlot(num_samples)  # Set up for multiple plots

    for _ in range(num_samples):
        sample = resp.sample(frac=1, replace=True)  # Resample with replacement
        _, sf = EstimateSurvival(sample)
        thinkplot.Plot(sf.ts, sf.ss, color="gray", alpha=0.1)  # Light gray for␣
↪bootstrap curves

    # Plot the original survival curve in blue for comparison
    _, sf_original = EstimateSurvival(resp)
    thinkplot.Plot(sf_original.ts, sf_original.ss, label="Original Survival␣
↪Curve", color="blue", alpha=0.8)

    thinkplot.Config(
        xlabel="Years of Marriage",
        ylabel="Survival Probability",
        title="Bootstrap Survival Curves",
        legend=True
    )
    plt.show()

# Calling plotting functions
PlotSurvivalCurve(sf, label="NSFG Cycle 6 Survival Curve")
CompareByDecade(married_6)
CompareByAgeAtMarriage(married_6)
BootstrapSurvival(married_6, num_samples=100)
```




Survival Probability by Birth Decade
Survival Probability
Years of Marriage
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
50s
60s
70s
80s


Survival Probability by Age at First Marriage
Survival Probability
Years of Marriage
Age <20
Age 20-25
Age 26-30
Age >30
1.0
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
30


Bootstrap Survival Curves
Original Survival Curve
Survival Probability
Years of Marriage
1.0
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
30