CE 3354 Engineering Hydrology
Exercise Set 7

Exercises

1. Figure 1 The following data represent gage height and annual peak discharge for some gaging station in Oklahoma. The stage is in feet and the discharge is in cubic feet per second. The data are sequential from 1923 through 1971.

   Use the data to:

   (a) Plot year versus stage (x-axis is year).

   (b) Plot year versus discharge (x-axis is year).

   (c) Plot the discharge versus stage.

   (d) Using the Weibull plotting position formula, determine the distribution parameters that fit the data for a log-normal distribution.

   (e) Using the Weibull plotting position formula, determine the distribution parameters that fit the data for a Gumbell distribution.

   (f) Using the Weibull plotting position formula, determine the distribution parameters that fit the data for a Gamma distribution.

   (g) Estimate the discharge associated with a 25-percent chance exceedence probability (i.e. the value that is equal to or exceeded with a 1 in 4 chance).

   (h) A resident claims that in the early 1900?s a flood corresponding to a stage of 30 feet occurred at the gage location. Estimate the exceedence probability (return period) of the flow associated with this event.

Stage	Discharge	Stage	Discharge
23.0	200,000	18.50	114,000
11.8	42,000	14.93	70,200
6.4	11,300	15.30	70,700
10.4	32,400	17.60	92,800
18.7	108,000	21.45	135,000
15.0	73,000	10.48	25,800
15.3	76,500	8.80	17,500
12.1	47,800	9.07	18,700
9.5	28,200	12.71	36,300
10.6	33,700	14.64	49,200
9.3	25,700	21.41	120,000
6.4	11,700	14.86	56,800
16.0	77,800	14.65	54,800
9.9	26,600	21.62	158,000
13.0	47,500	21.22	165,000
16.44	75,600	17.83	103,000
8.48	19,200	8.76	19,700
10.26	27,800	9.00	21,100
13.59	51,000	22.60	171,000
18.54	94,000	6.74	10,400
18.12	97,200	12.54	42,000
22.82	179,000	14.10	52,800
19.55	124,000	16.42	77,000
19.48	110,000	18.33	101,000
		8.14	17,100

Figure 1: Data from Oklahoma Gaging Station

Solution(s) attached next page

Problem1-WS

August 1, 2025

0.1 Problem 1

Figure 1 The following data represent gage height and annual peak discharge for some gaging station in Oklahoma. The stage is in feet and the discharge is in cubic feet per second. The data are sequential from 1923 through 1971. Use the data to: 1. Plot year versus stage (x-axis is year). 2. Plot year versus discharge (x-axis is year). 3. Plot the discharge versus stage. 4. Using the Weibull plotting position formula, determine the distribution parame- ters that fit the data for a log-normal distribution. 5. Using the Weibull plotting position formula, determine the distribution parame- ters that fit the data for a Gumbell distribution. 6. Using the Weibull plotting position formula, determine the distribution parame- ters that fit the data for a Gamma distribution. 7. Estimate the discharge associated with a 25-percent chance exceedence probability (i.e. the value that is equal to or exceeded with a 1 in 4 chance). 8. A resident claims that in the early 1900?s a flood corresponding to a stage of 30 feet occurred at the gage location. Estimate the exceedence probability (return period) of the flow associated with this event.

0.2 Solution(s) using ENGR-1330 methods

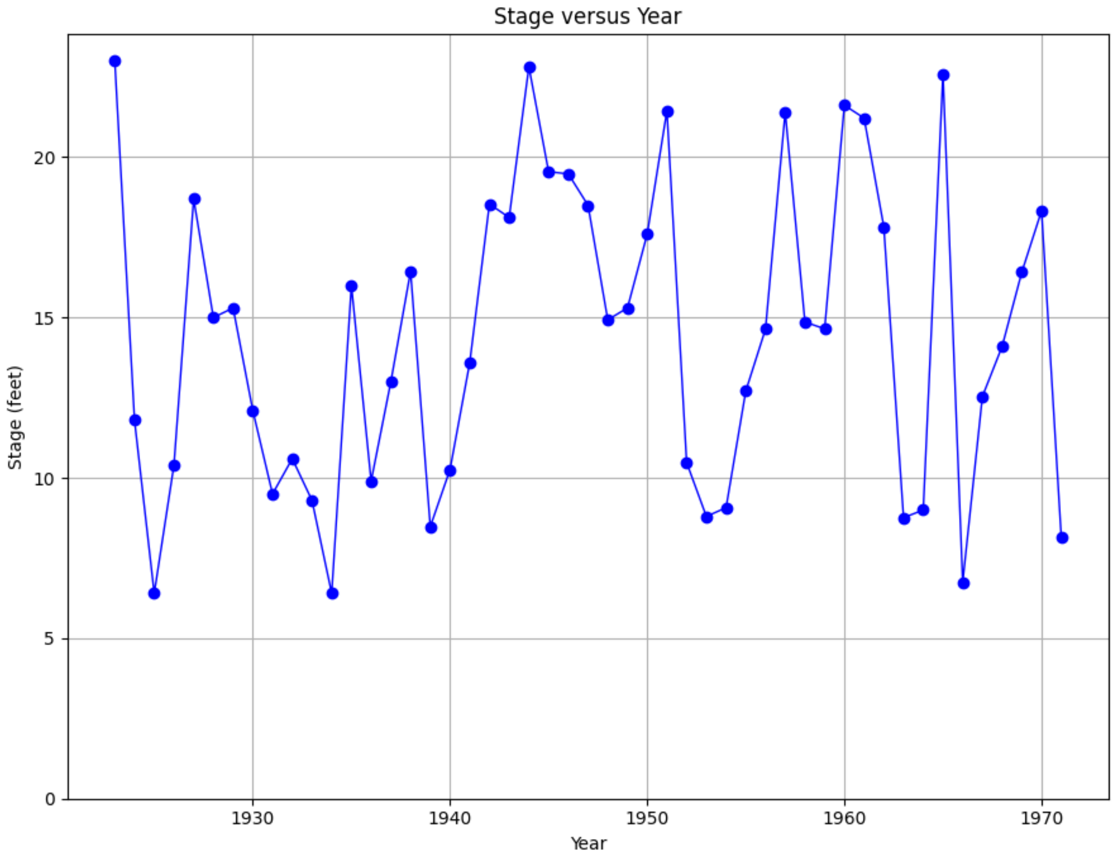
[235]:
```
#==== Import Libraries ====#
import matplotlib.pyplot # the python plotting library
import math        # import math package
import numpy       # import numpy package
import pandas      # import pandas package
import scipy.stats # import scipy stats package
#==== Prototype Functions ====#
def weibull_sorted(sample_length):
    # generate weibull plotting positions - sample is assumed already sorted␣
 ↪(small to large)
    weibull_pp = [] # built a relative frequency approximation to probability,␣
 ↪assume each pick is equally likely
    for i in range(0,sample_length,1):
        weibull_pp.append((i+1)/(sample_length+1))
    return weibull_pp

def loggit(x):  # A prototype function to log transform x
    return(math.log(x))

def antiloggit(logx):  # A prototype function to transformed log(x)
    return(math.exp(logx))
```

```python
def normdist(x,mu,sigma): # A prototype function to return density from normal␣
 ↪distribution(s)
    argument = (x - mu)/(math.sqrt(2.0)*sigma)
    normdist = (1.0 + math.erf(argument))/2.0
    return normdist

def ev1dist(x,alpha,beta):
    argument = (x - alpha)/beta
    constant = 1.0/beta
    ev1dist = math.exp(-1.0*math.exp(-1.0*argument))
    return ev1dist

def gammacdf(x,tau,alpha,beta): # Gamma Cumulative Density function - with␣
 ↪three parameter to one parameter convert
    xhat = x-tau
    lamda = 1.0/beta
    gammacdf = 1.0 - scipy.stats.gamma.cdf(lamda*xhat, alpha)
    return gammacdf

#==== Input Data ====#
database =[[1923,23,200000],
[1924,11.8,42000],
[1925,6.4,11300],
[1926,10.4,32400],
[1927,18.7,108000],
[1928,15,73000],
[1929,15.3,76500],
[1930,12.1,47800],
[1931,9.5,28200],
[1932,10.6,33700],
[1933,9.3,25700],
[1934,6.4,11700],
[1935,16,77800],
[1936,9.9,26600],
[1937,13,47500],
[1938,16.44,75600],
[1939,8.48,19200],
[1940,10.26,27800],
[1941,13.59,51000],
[1942,18.54,94000],
[1943,18.12,97200],
[1944,22.82,179000],
[1945,19.55,124000],
[1946,19.48,110000],
[1947,18.5,114000],
[1948,14.93,70200],
```

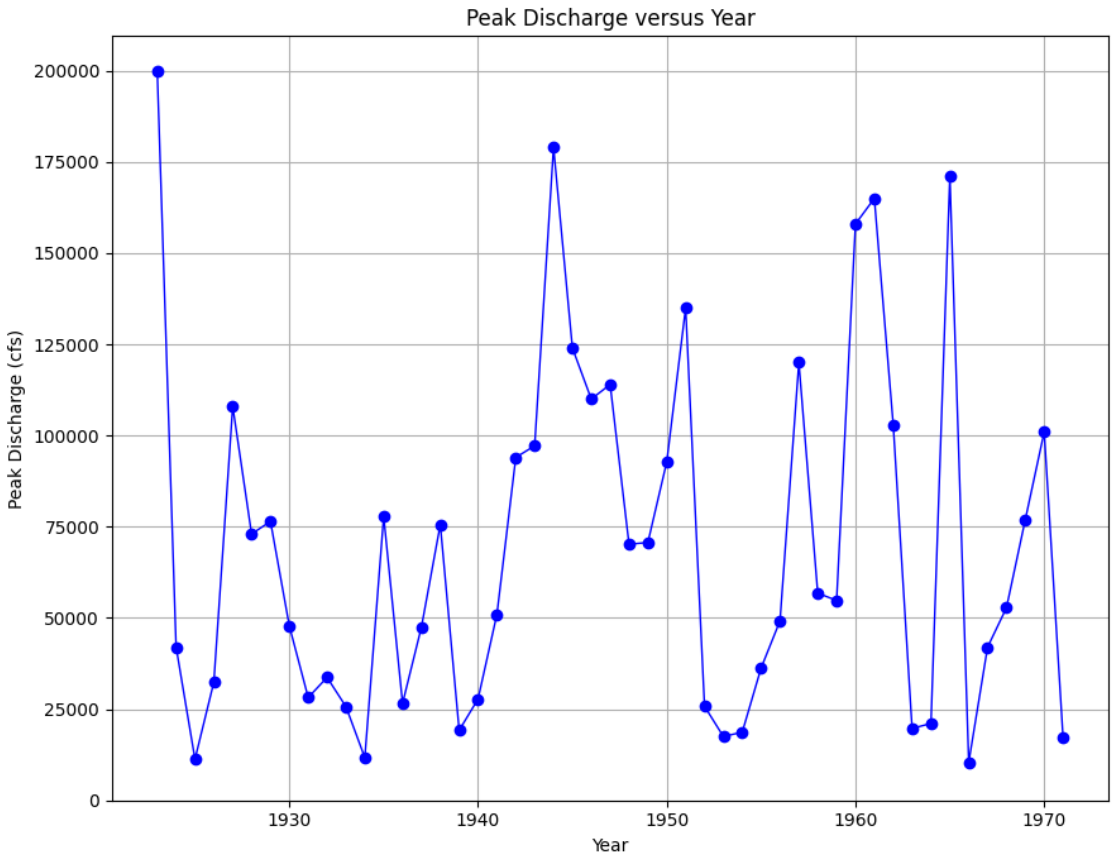
```
[1949,15.3,70700],
[1950,17.6,92800],
[1951,21.45,135000],
[1952,10.48,25800],
[1953,8.8,17500],
[1954,9.07,18700],
[1955,12.71,36300],
[1956,14.64,49200],
[1957,21.41,120000],
[1958,14.86,56800],
[1959,14.65,54800],
[1960,21.62,158000],
[1961,21.22,165000],
[1962,17.83,103000],
[1963,8.76,19700],
[1964,9,21100],
[1965,22.6,171000],
[1966,6.74,10400],
[1967,12.54,42000],
[1968,14.1,52800],
[1969,16.42,77000],
[1970,18.33,101000],
[1971,8.14,17100],
]
# extract annual peaks and stage
howmanyrows = len(database)
years=[0 for i in range(howmanyrows)]
stage=[0 for i in range(howmanyrows)]
peaks=[0 for i in range(howmanyrows)]
for i in range(howmanyrows):
    years[i]=database[i][0], #extract first entry each row of list database
    stage[i]=database[i][1] #extract second entry each row of list database
    peaks[i]=database[i][2] #extract third entry each row of list database
peaks_copy = list(peaks) # Copy the peaks list for making a rating curve later␣
 ↪on
```

[236]:
```
myfigure = matplotlib.pyplot.figure(figsize = (9,7)) # generate a object from␣
 ↪the figure class, set aspect ratio
matplotlib.pyplot.plot(years, stage ,color ='blue',marker='o',linewidth=1)
matplotlib.pyplot.xlabel("Year")
matplotlib.pyplot.ylabel("Stage (feet)")
mytitle = "Stage versus Year"
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.grid()  # Adjust rotation as needed
matplotlib.pyplot.ylim(bottom=0)# Set y-axis to start at zero and auto-scale␣
 ↪upper bound
matplotlib.pyplot.tight_layout()  # Prevent label/title clipping
```

```
matplotlib.pyplot.show()
```



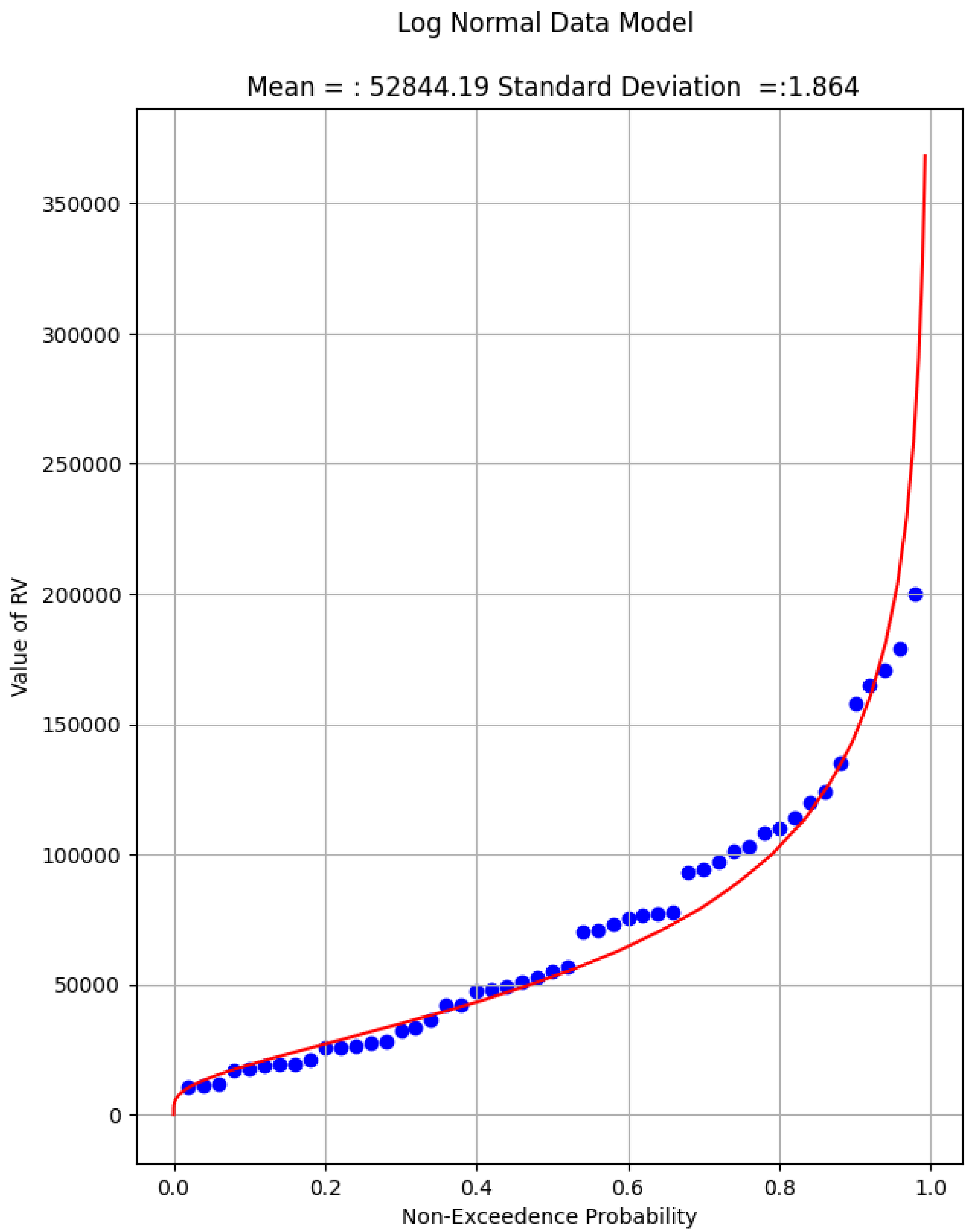
[237]:
```
myfigure = matplotlib.pyplot.figure(figsize = (9,7)) # generate a object from␣
↪the figure class, set aspect ratio
matplotlib.pyplot.plot(years, peaks ,color ='blue',marker='o',linewidth=1)
matplotlib.pyplot.xlabel("Year")
matplotlib.pyplot.ylabel("Peak Discharge (cfs)")
mytitle = "Peak Discharge versus Year"
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.grid()  # Adjust rotation as needed
matplotlib.pyplot.ylim(bottom=0)# Set y-axis to start at zero and auto-scale␣
↪upper bound
matplotlib.pyplot.tight_layout()  # Prevent label/title clipping
matplotlib.pyplot.show()
```



[238]:
```
# generate plotting positions
sample_length = len(peaks)
weibull_pp = weibull_sorted(sample_length)
peaks.sort() #sort in place
```

[239]:
```
# Fit and plot lognormal
peaks_array=pandas.Series(peaks)
logsample = peaks_array.apply(loggit).tolist() # put the peaks into a list
sample_mean = numpy.array(logsample).mean()
sample_variance = numpy.array(logsample).std()**2
logsample.sort() # sort the logsample in place!
mu = sample_mean # Fitted Model in Log Space
sigma = math.sqrt(sample_variance)
x = []; ycdf = []
xlow = 1; xhigh = 1.05*max(logsample) ; howMany = 100
xstep = (xhigh - xlow)/howMany
for i in range(0,howMany+1,1):
    x.append(antiloggit(xlow + i*xstep))
    yvalue = normdist(xlow + i*xstep,mu,sigma)
    ycdf.append(yvalue)
```

```
# Now plot the sample values and plotting position
peaks.sort() #sort in place
myfigure = matplotlib.pyplot.figure(figsize = (7,9)) # generate a object from␣
↪the figure class, set aspect ratio
matplotlib.pyplot.scatter(weibull_pp, peaks ,color ='blue')
matplotlib.pyplot.plot(ycdf, x, color ='red')
matplotlib.pyplot.xlabel("Non-Exceedence Probability")
matplotlib.pyplot.ylabel("Value of RV")
mytitle = "Log Normal Data Model \n \n Mean = : " +␣
↪str(round(antiloggit(sample_mean),2))+ " Standard Deviation  =:" +␣
↪str(round(antiloggit(sample_variance),3))
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.grid()
matplotlib.pyplot.show()
print(mu,sigma)
```




```
10.875102965825535 0.7890880443300708
```

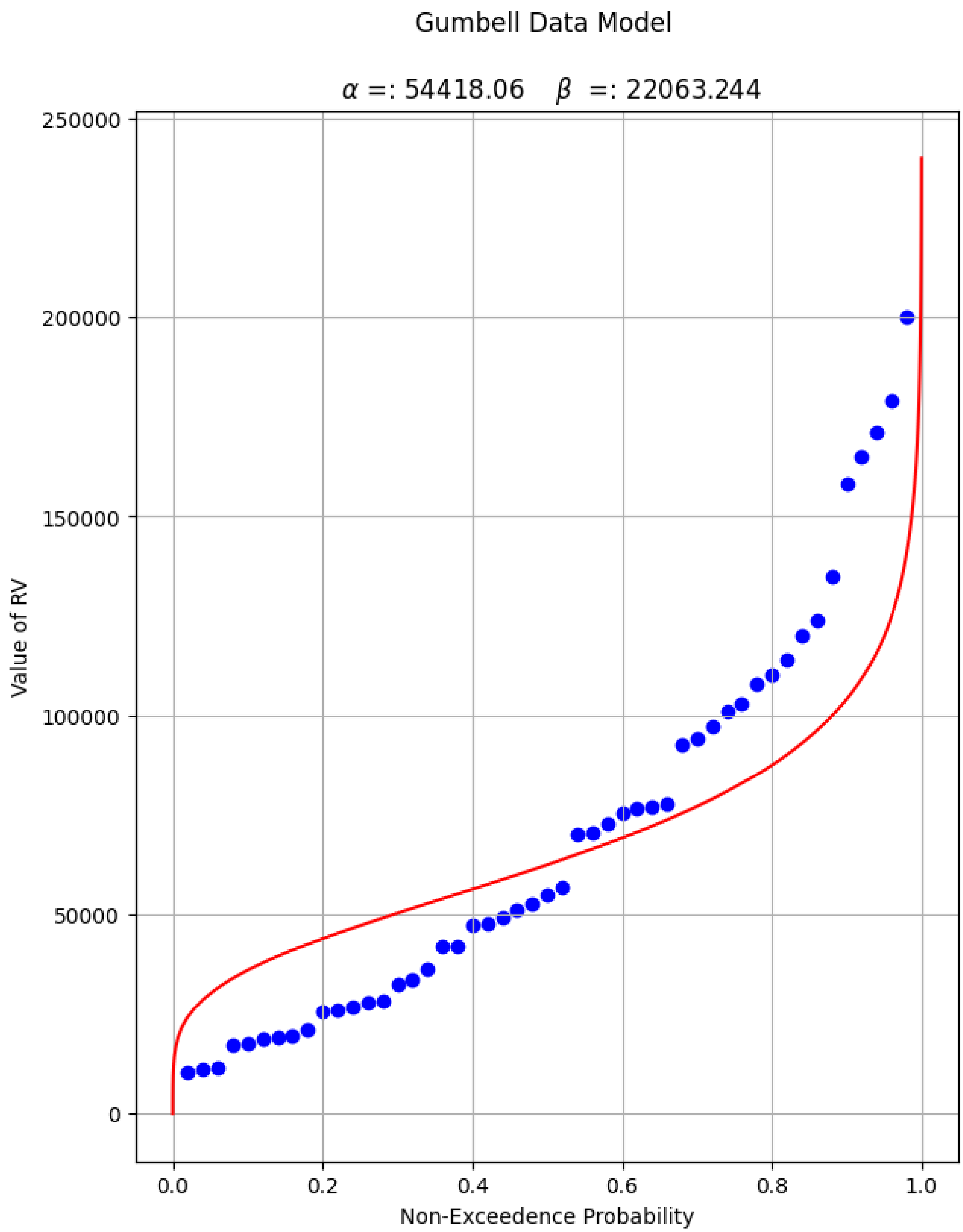
[240]:
```
from scipy.optimize import newton

def f(x):
```

```
    mu = 10.875102965825535
    sigma = 0.7890880443300708
    quantile = 0.75
    argument = (loggit(x) - mu)/(math.sqrt(2.0)*sigma)
    normdist = (1.0 + math.erf(argument))/2.0
    return normdist - quantile

print("Log-Normal Fit \n 0.25 AEP (4-Year ARI) :", round(newton(f,␣
↪20000),2),"cfs")
```

```
Log-Normal Fit
 0.25 AEP (4-Year ARI) : 89979.29 cfs
```

[241]:

```
# Fit and plot gumbell
peaks_array=pandas.Series(peaks)
sample_mean = numpy.array(peaks_array).mean()
sample_variance = numpy.array(peaks_array).std()**2
alpha_mom = sample_mean*math.sqrt(6)/math.pi
beta_mom = math.sqrt(sample_variance)*0.45
mu = sample_mean # Fitted Model
sigma = math.sqrt(sample_variance)
x = []; ycdf = []
xlow = 0; xhigh = 1.2*max(peaks) ; howMany = 100
xstep = (xhigh - xlow)/howMany
for i in range(0,howMany+1,1):
    x.append(xlow + i*xstep)
    yvalue = ev1dist(xlow + i*xstep,alpha_mom,beta_mom)
    ycdf.append(yvalue)
# Now plot the sample values and plotting position
peaks.sort() #sort in place
myfigure = matplotlib.pyplot.figure(figsize = (7,9)) # generate a object from␣
↪the figure class, set aspect ratio
matplotlib.pyplot.scatter(weibull_pp, peaks ,color ='blue')
matplotlib.pyplot.plot(ycdf, x, color ='red')
matplotlib.pyplot.xlabel("Non-Exceedence Probability")
matplotlib.pyplot.ylabel("Value of RV")
mytitle = "Gumbell Data Model \n \n " + r"$\alpha$ =: " +␣
↪str(round((alpha_mom),2))+ r"    $\beta$  =: " + str(round((beta_mom),3))
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.grid()
matplotlib.pyplot.show()
print(alpha_mom,beta_mom)
```




```
54418.063072225494 22063.243768147015
```

[242]:
```python
from scipy.optimize import newton
```

```
def f(x):
    alpha = 54418.063072225494
    beta = 22063.243768147015
    quantile = 0.75
    argument = (x - alpha)/beta
    constant = 1.0/beta
    ev1dist = math.exp(-1.0*math.exp(-1.0*argument))
    return ev1dist - quantile

print("Gumbell Fit \n 0.25 AEP (4-Year ARI) :", round(newton(f, 70000),2),"cfs")
```

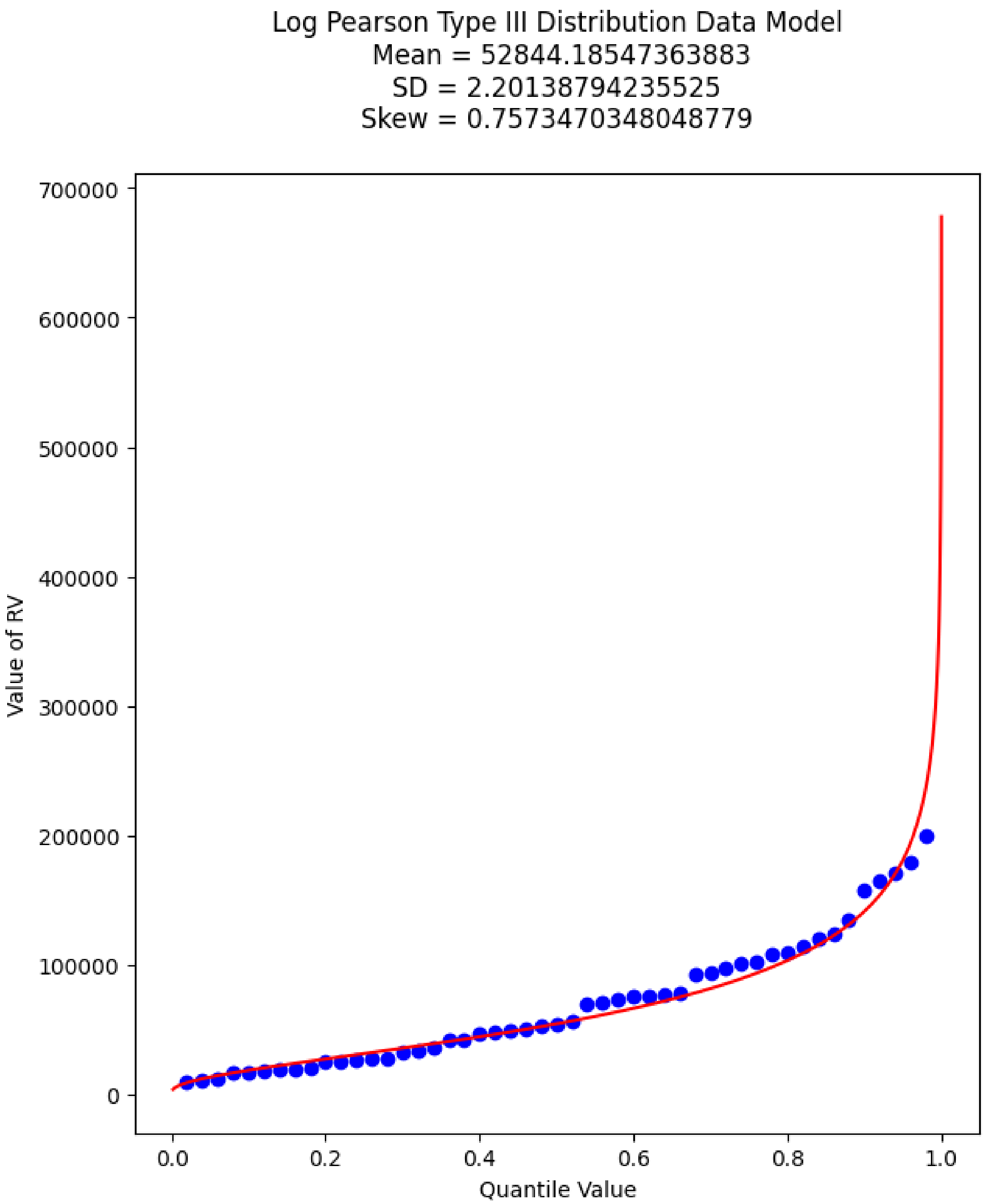
```
Gumbell Fit
 0.25 AEP (4-Year ARI) : 81906.64 cfs
```

[ ]:

[243]:

```
# Fit and plot Log-Pearson Type III (Gamma)
logsample = peaks_array.apply(loggit).tolist() # put the peaks into a list
sample_mean  = numpy.array(logsample).mean()
sample_stdev = numpy.array(logsample).std()
sample_skew  = scipy.stats.skew(logsample)
sample_alpha = 4.0/(sample_skew**2)
sample_beta  = numpy.sign(sample_skew)*math.sqrt(sample_stdev**2/sample_alpha)
sample_tau   = sample_mean - sample_alpha*sample_beta
#==== Build Plot Data ====
x = []; ycdf = []
xlow = (0.9*min(logsample)); xhigh = (1.1*max(logsample)) ; howMany = 100
xstep = (xhigh - xlow)/howMany
for i in range(0,howMany+1,1):
    x.append(xlow + i*xstep)
    yvalue = gammacdf(xlow + i*xstep,sample_tau,sample_alpha,sample_beta)
    ycdf.append(yvalue)
#=== Reverse Transform x ===
for i in range(len(x)):
    x[i] = antiloggit(x[i])
#=== Plot Result(s) ===
peaks.sort()
myfigure = matplotlib.pyplot.figure(figsize = (7,8)) # generate a object from
 ↪the figure class, set aspect ratio
matplotlib.pyplot.scatter(weibull_pp, peaks ,color ='blue')
matplotlib.pyplot.plot(ycdf, x, color ='red')
matplotlib.pyplot.xlabel("Quantile Value")
matplotlib.pyplot.ylabel("Value of RV")
mytitle = "Log Pearson Type III Distribution Data Model\n "
mytitle += "Mean = " + str(antiloggit(sample_mean)) + "\n"
mytitle += "SD = " + str(antiloggit(sample_stdev)) + "\n"
mytitle += "Skew = " + str(antiloggit(sample_skew)) + "\n"
```

```
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.show()
print(sample_tau,sample_alpha,sample_beta)
```




```
16.55334964841315 51.78185270767057 -0.10965707841014505
```

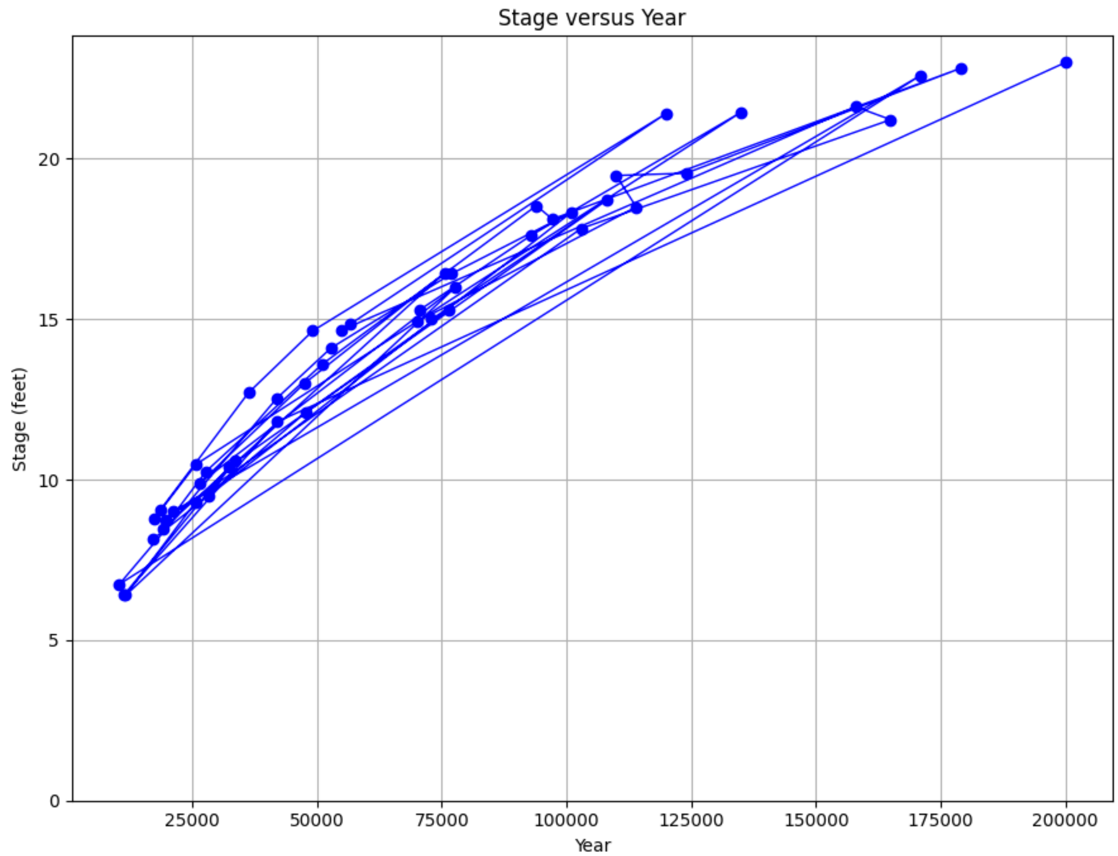
[244]:

```
from scipy.optimize import newton

def f(x):
    sample_tau = 16.55334964841315
    sample_alpha = 51.78185270767057
    sample_beta = -0.10965707841014505
    quantile = 0.75
    argument = loggit(x)
    gammavalue = gammacdf(argument,sample_tau,sample_alpha,sample_beta)
    return gammavalue - quantile

print("Log-Pearson (Gamma) Fit \n 0.25 AEP (4-Year ARI) :", round(newton(f,␣
↪70000),2),"cfs")
```

```
Log-Pearson (Gamma) Fit
 0.25 AEP (4-Year ARI) : 91615.5 cfs
```

[245]:

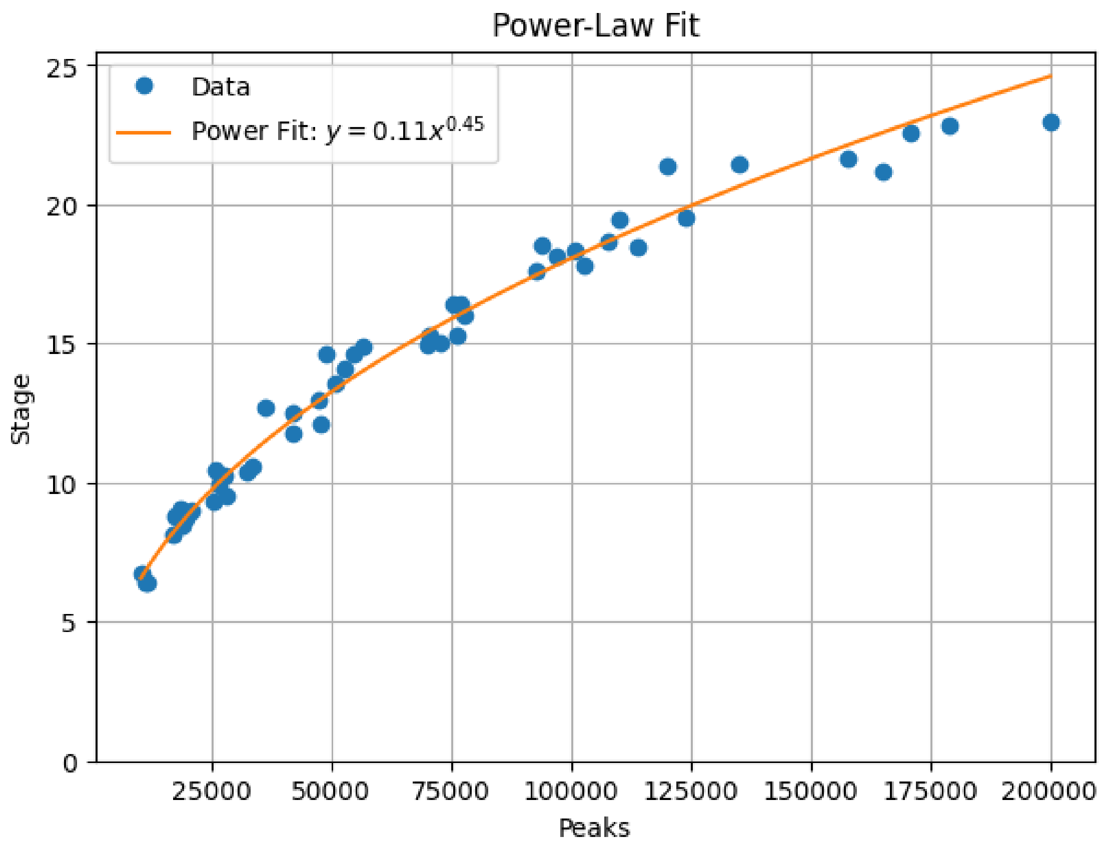
```
# Plot Stage versus Q
myfigure = matplotlib.pyplot.figure(figsize = (9,7)) # generate a object from␣
↪the figure class, set aspect ratio
matplotlib.pyplot.plot(peaks_copy, stage ,color ='blue',marker='o',linewidth=1)
matplotlib.pyplot.xlabel("Year")
matplotlib.pyplot.ylabel("Stage (feet)")
mytitle = "Stage versus Year"
matplotlib.pyplot.title(mytitle)
matplotlib.pyplot.grid()  # Adjust rotation as needed
matplotlib.pyplot.ylim(bottom=0)# Set y-axis to start at zero and auto-scale␣
↪upper bound
matplotlib.pyplot.tight_layout()  # Prevent label/title clipping
matplotlib.pyplot.show()
```



[246]:

```
# Generate a rating curve Power-law should work OK

# Convert lists to arrays
x = np.array(peaks_copy)
y = np.array(stage)

# Transform to log-log space
log_x = np.log(x)
log_y = np.log(y)

# Linear fit in log space
b, log_a = np.polyfit(log_x, log_y, deg=1)
a = np.exp(log_a)

# Power-law function: y = a * x^b
y_power = a * x_fit**b

# Plot
plt.figure(figsize=(7, 5))
plt.plot(x, y, 'o', label='Data')
```

```
plt.plot(x_fit, y_power, '-', label=fr'Power Fit: $y = {a:.2f}x^{{{b:.2f}}}$')
plt.xlabel("Peaks")
plt.ylabel("Stage")
plt.title("Power-Law Fit")
plt.legend()
plt.ylim(bottom=0)
plt.grid(True)
plt.show()
```



[247]:
```
# Use newtons method to find Q for a given Stage
from scipy.optimize import newton

def f(x):
    stage = 30.0 #reported stage from old-timer
    constant=0.11
    exponent=0.45
    f=constant*(x**exponent)-stage
    return f

print("Estimated Discharge for Stage :", round(newton(f, 300000),0),"cfs")
```

```
Estimated Discharge for Stage : 258661.0 cfs
```

[248]:
```python
from scipy.optimize import newton

def f(x):
    sample_tau = 16.55334964841315
    sample_alpha = 51.78185270767057
    sample_beta = -0.10965707841014505
    quantile = 0.985765
    argument = loggit(x)
    gammavalue = gammacdf(argument,sample_tau,sample_alpha,sample_beta)
    return gammavalue - quantile

print("Log-Pearson (Gamma) Fit \n 0.014 AEP (71.4-Year ARI) :", round(newton(f,␣
↪200000),2),"cfs")
```

```
Log-Pearson (Gamma) Fit
 0.014 AEP (71.4-Year ARI) : 258622.37 cfs
```

2. Use the Oklahoma data you just prepared and analyze using the Bulletin 17C procedure (using the HEC-SSP software tool - use station skew option).

Solution(s) below

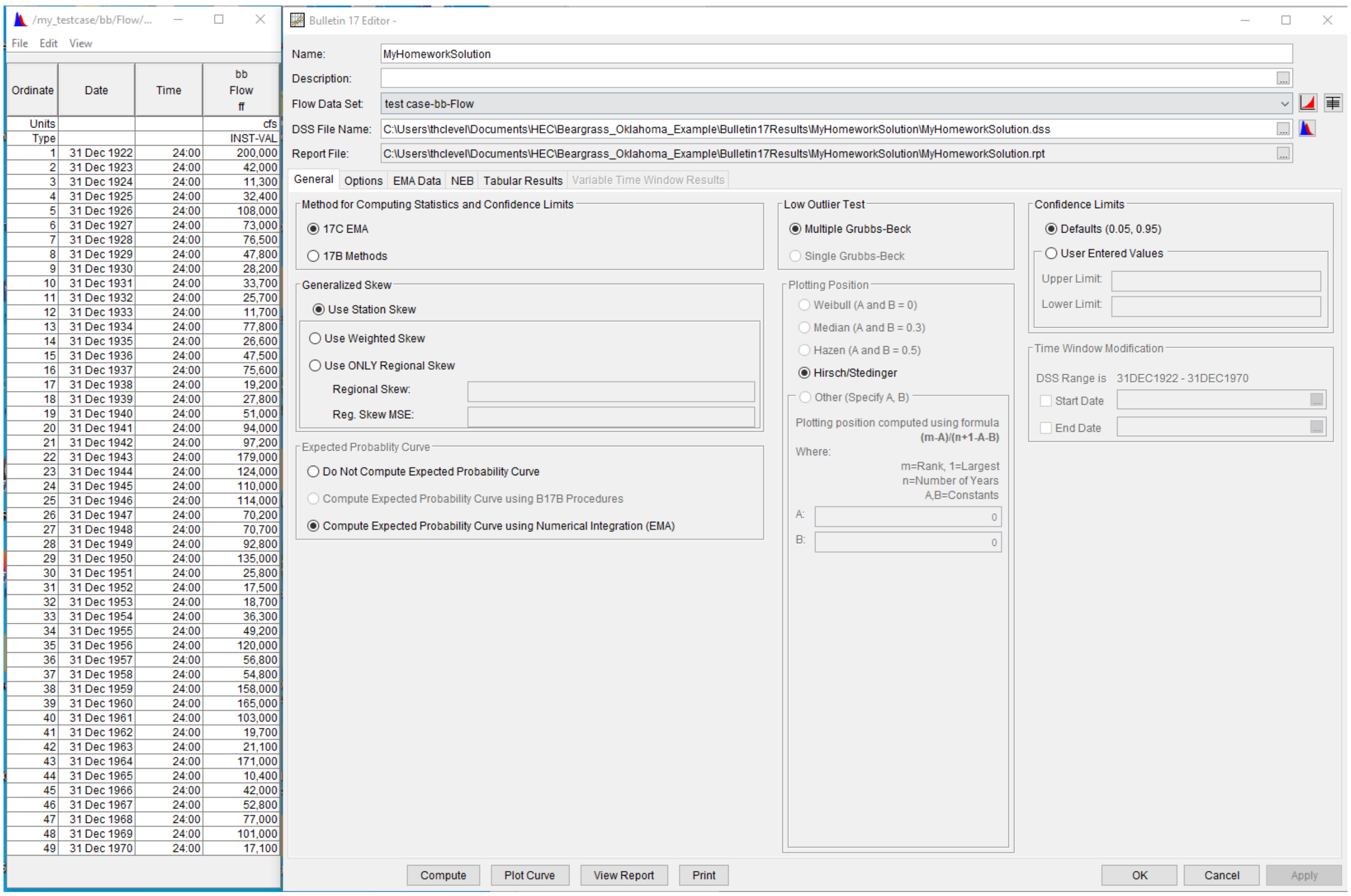


Figure 2: Interface and Data from Oklahoma Gaging Station

The 0.25 AEP (4-year ARI) value falls between 104,273 CFS at 0.20 AEP and 54,894 CFS at 0.50 AEP. Logarithmic interpolation produces an estimate of 89,189 CFS which is close to our homebrew result of 91,615 CFS for the same AEP value.

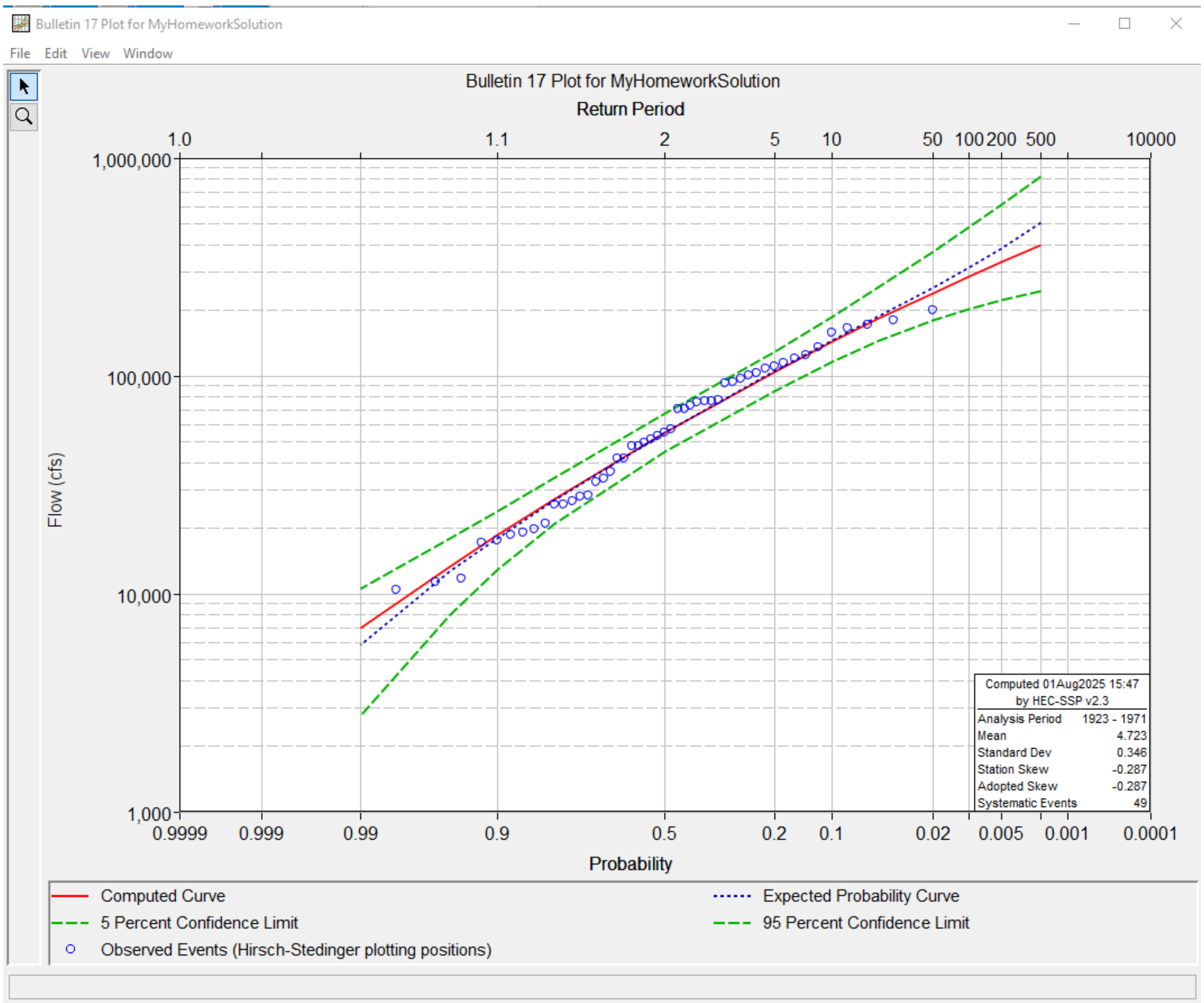


Figure 3: Probability Plot from Oklahoma Gaging Station

MyHomeworkSolution.rpt

File Edit Search Format

File: C:\Users\thclevel\Documents\HEC\Beargrass_Oklahoma_Example\Bulletin17Results\MyHomeworkSolution\MyHomeworkSolution.rpt

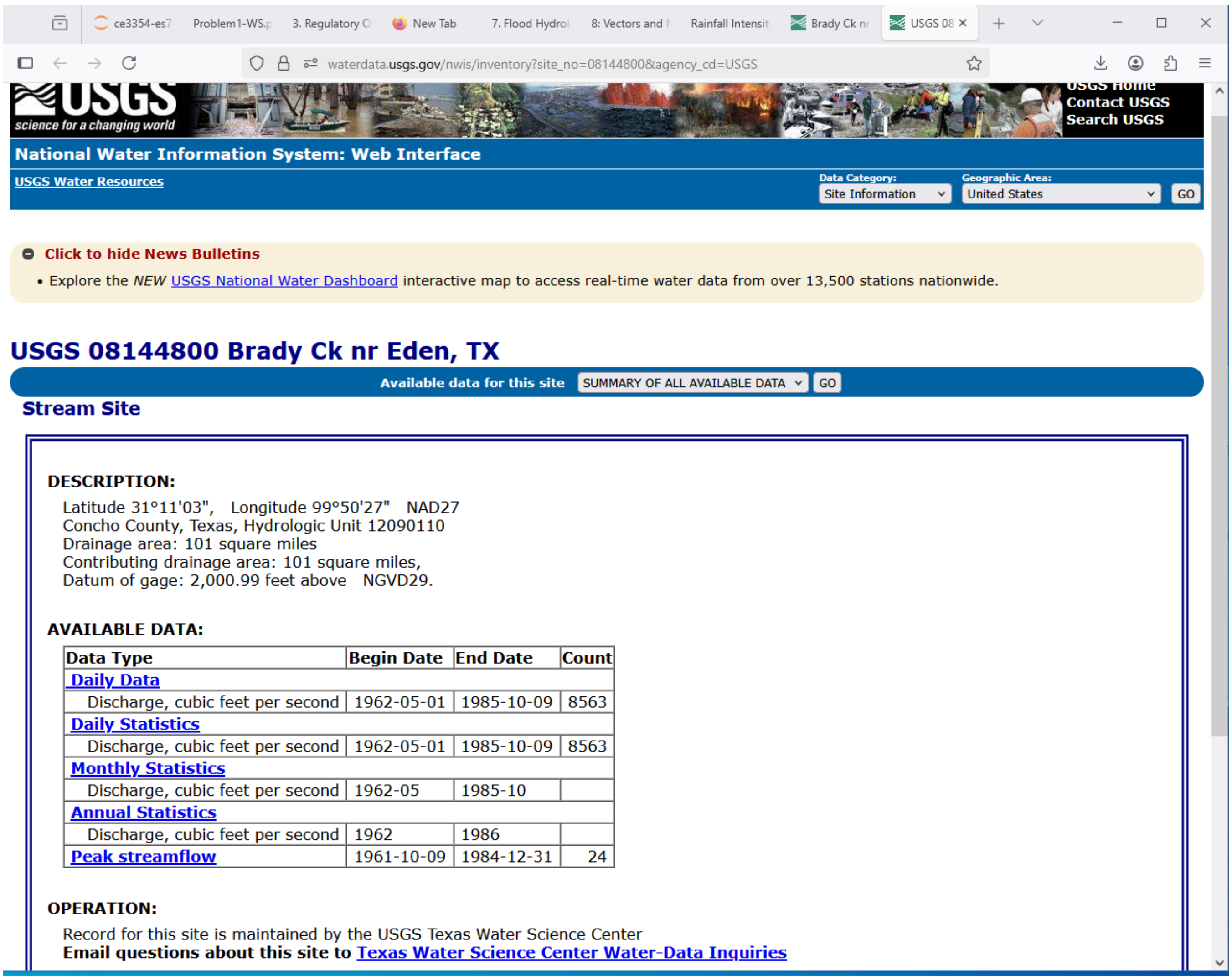
```
* Low outlier plotting positions are computed using Median parameters.



<< Frequency Curve >>
test case-bb-Flow
-----------------------------------------------------------------------------------------
|   Computed      Variance      Expected    |   Percent    |      Confidence Limits      |
|    Curve        Log(EMA)     Probability  |   Chance     |        0.05          0.95  |
|                 FLOW, cfs                 | Exceedance   |          FLOW, cfs          |
|-------------------------------------------|--------------|-----------------------------|
|     398,178.1     0.0228903     506,919.1 |        0.2   |      820,010.8     246,443.4 |
|     332,392.7     0.0158145     385,355.4 |        0.5   |      608,326.1     223,368.8 |
|     285,031.5     0.0114952     313,451.8 |        1.0   |      479,275.1     203,295.5 |
|     239,777.7     0.0080699     253,924.2 |        2.0   |      371,893.1     180,510.6 |
|     183,203.8     0.0048961     188,258.4 |        5.0   |      256,583.8     145,652.6 |
|     142,806.3     0.0034759     145,011.5 |       10.0   |      186,084.2     116,042.0 |
|     104,273.5     0.0028127     104,907.4 |       20.0   |      129,239.0      85,405.6 |
|      54,894.3     0.0028407      54,645.9 |       50.0   |       67,423.4      44,628.6 |
|      27,381.1     0.0038459      26,923.3 |       80.0   |       34,149.1      20,905.9 |
|      18,620.6     0.0057026      18,083.1 |       90.0   |       23,900.4      12,905.6 |
|      13,386.6     0.0088478      12,687.8 |       95.0   |       17,903.6       8,067.2 |
|       7,004.0     0.0219658       5,817.2 |       99.0   |       10,608.3       2,754.9 |
|-------------------------------------------|--------------|-----------------------------|
Non-monotonically increasing confidence limits were encountered.  Results may be inaccurate due to smoothing.


<< Multiple Grubbs-Beck Test P-Values >>
test case-bb-Flow
--------------------------------
|    Number Of    |  P-Values    |
|  Low Outliers   |              |
|-----------------|--------------|
|               1 |    6.884E-1  |
|               2 |    3.938E-1  |
|               3 |    1.480E-1  |
|               4 |    5.628E-1  |
|               5 |    3.589E-1  |
|               6 |    2.825E-1  |
|               7 |    1.527E-1  |
|               8 |    6.977E-2  |
|               9 |    4.990E-2  |
|              10 |    2.117E-1  |
|              11 |    9.460E-2  |
|              12 |    5.121E-2  |
|              13 |    3.164E-2  |
|              14 |    1.054E-2  |
|              15 |    3.356E-2  |
```

08/01/2025 15:47 PM   1:1..1:1   1:1

Figure 4: Probability Tabulation from Oklahoma Gaging Station

3. Locate USGS Station 08144800 Brady Creek near Eden, TX. and analyze the historical peaks using the Bulletin 17C procedures (use the PeakFQ software tool use station skew option). Determine the median discharge predicted for this station by PeakFQ. Also determine the discharge per square mile of contributing drainage area.[1]

   Solution(s) below



Data Type	Begin Date	End Date	Count
Daily Data			
Discharge, cubic feet per second	1962-05-01	1985-10-09	8563
Daily Statistics			
Discharge, cubic feet per second	1962-05-01	1985-10-09	8563
Monthly Statistics			
Discharge, cubic feet per second	1962-05	1985-10	
Annual Statistics			
Discharge, cubic feet per second	1962	1986	
Peak streamflow	1961-10-09	1984-12-31	24

Figure 5: NWIS Inventory Page for 08144800, choose peak streamflow link

[1] Download the annual peaks from NWIS in the tab-delimited format for use in peakFQ; The example in the instructor notes is this particular site.

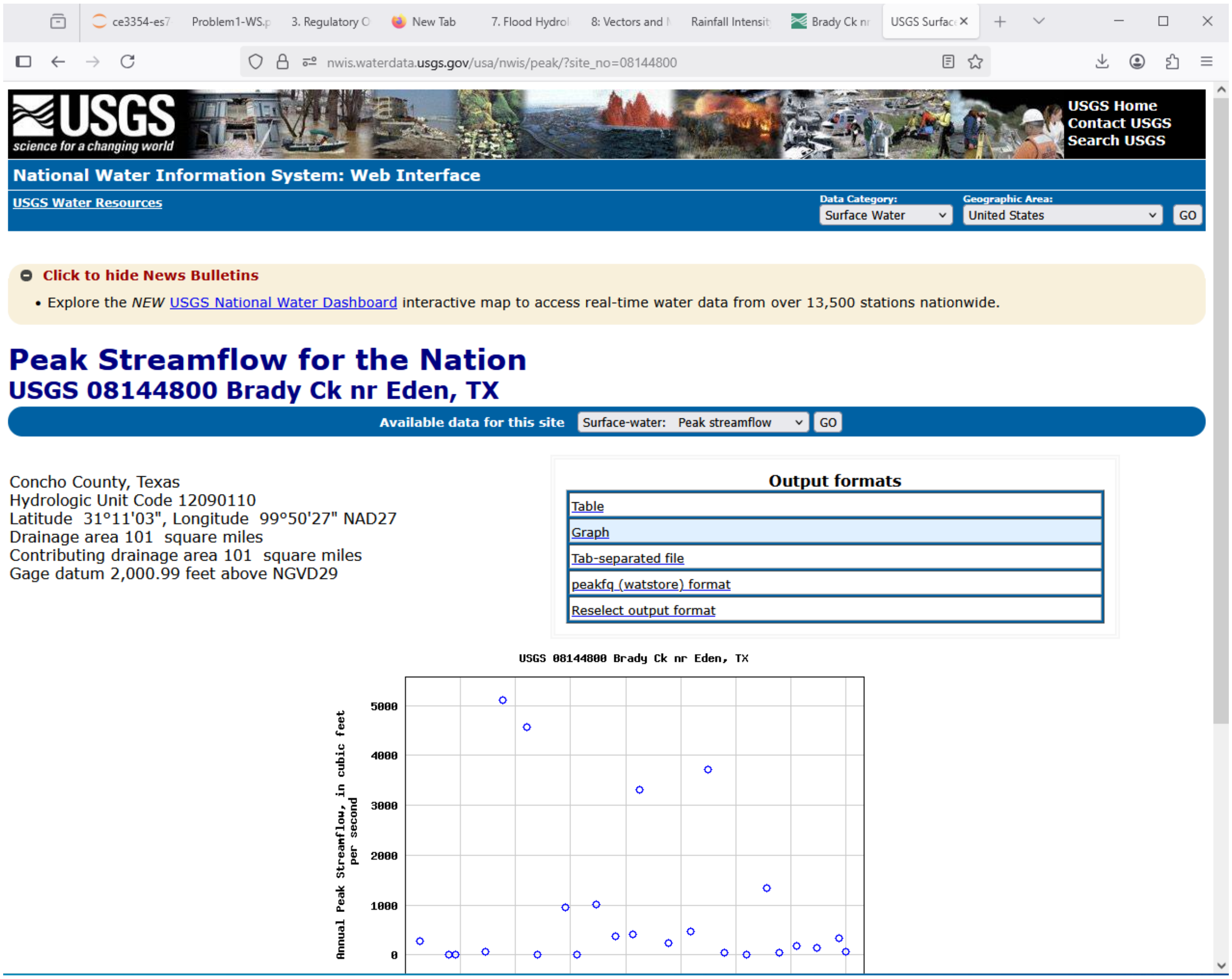
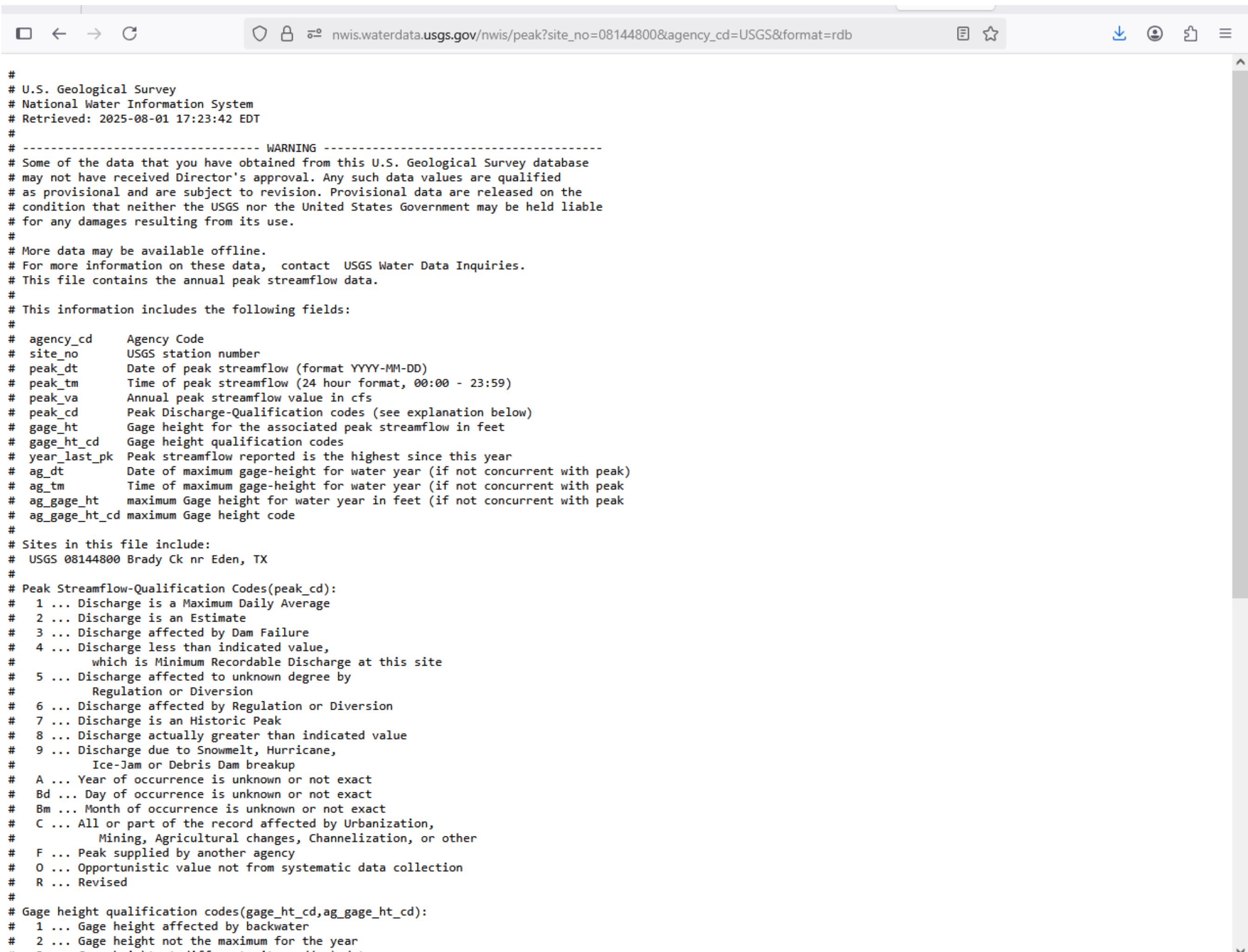


Figure 6: Download Page for 08144800 (Note drainage area is shown here; about 101 square miles.




```
#
# U.S. Geological Survey
# National Water Information System
# Retrieved: 2025-08-01 17:23:42 EDT
#
# ---------------------------------- WARNING ----------------------------------------
# Some of the data that you have obtained from this U.S. Geological Survey database
# may not have received Director's approval. Any such data values are qualified
# as provisional and are subject to revision. Provisional data are released on the
# condition that neither the USGS nor the United States Government may be held liable
# for any damages resulting from its use.
#
# More data may be available offline.
# For more information on these data,  contact  USGS Water Data Inquiries.
# This file contains the annual peak streamflow data.
#
# This information includes the following fields:
#
#  agency_cd     Agency Code
#  site_no       USGS station number
#  peak_dt       Date of peak streamflow (format YYYY-MM-DD)
#  peak_tm       Time of peak streamflow (24 hour format, 00:00 - 23:59)
#  peak_va       Annual peak streamflow value in cfs
#  peak_cd       Peak Discharge-Qualification codes (see explanation below)
#  gage_ht       Gage height for the associated peak streamflow in feet
#  gage_ht_cd    Gage height qualification codes
#  year_last_pk  Peak streamflow reported is the highest since this year
#  ag_dt         Date of maximum gage-height for water year (if not concurrent with peak)
#  ag_tm         Time of maximum gage-height for water year (if not concurrent with peak
#  ag_gage_ht    maximum Gage height for water year in feet (if not concurrent with peak
#  ag_gage_ht_cd maximum Gage height code
#
# Sites in this file include:
#  USGS 08144800 Brady Ck nr Eden, TX
#
# Peak Streamflow-Qualification Codes(peak_cd):
#   1 ... Discharge is a Maximum Daily Average
#   2 ... Discharge is an Estimate
#   3 ... Discharge affected by Dam Failure
#   4 ... Discharge less than indicated value,
#           which is Minimum Recordable Discharge at this site
#   5 ... Discharge affected to unknown degree by
#           Regulation or Diversion
#   6 ... Discharge affected by Regulation or Diversion
#   7 ... Discharge is an Historic Peak
#   8 ... Discharge actually greater than indicated value
#   9 ... Discharge due to Snowmelt, Hurricane,
#           Ice-Jam or Debris Dam breakup
#   A ... Year of occurrence is unknown or not exact
#   Bd ... Day of occurrence is unknown or not exact
#   Bm ... Month of occurrence is unknown or not exact
#   C ... All or part of the record affected by Urbanization,
#            Mining, Agricultural changes, Channelization, or other
#   F ... Peak supplied by another agency
#   O ... Opportunistic value not from systematic data collection
#   R ... Revised
#
# Gage height qualification codes(gage_ht_cd,ag_gage_ht_cd):
#   1 ... Gage height affected by backwater
#   2 ... Gage height not the maximum for the year
#   3 ... Gage height at different site and(or) datum
```

Figure 7: Data file excerpt - verify non-empty file

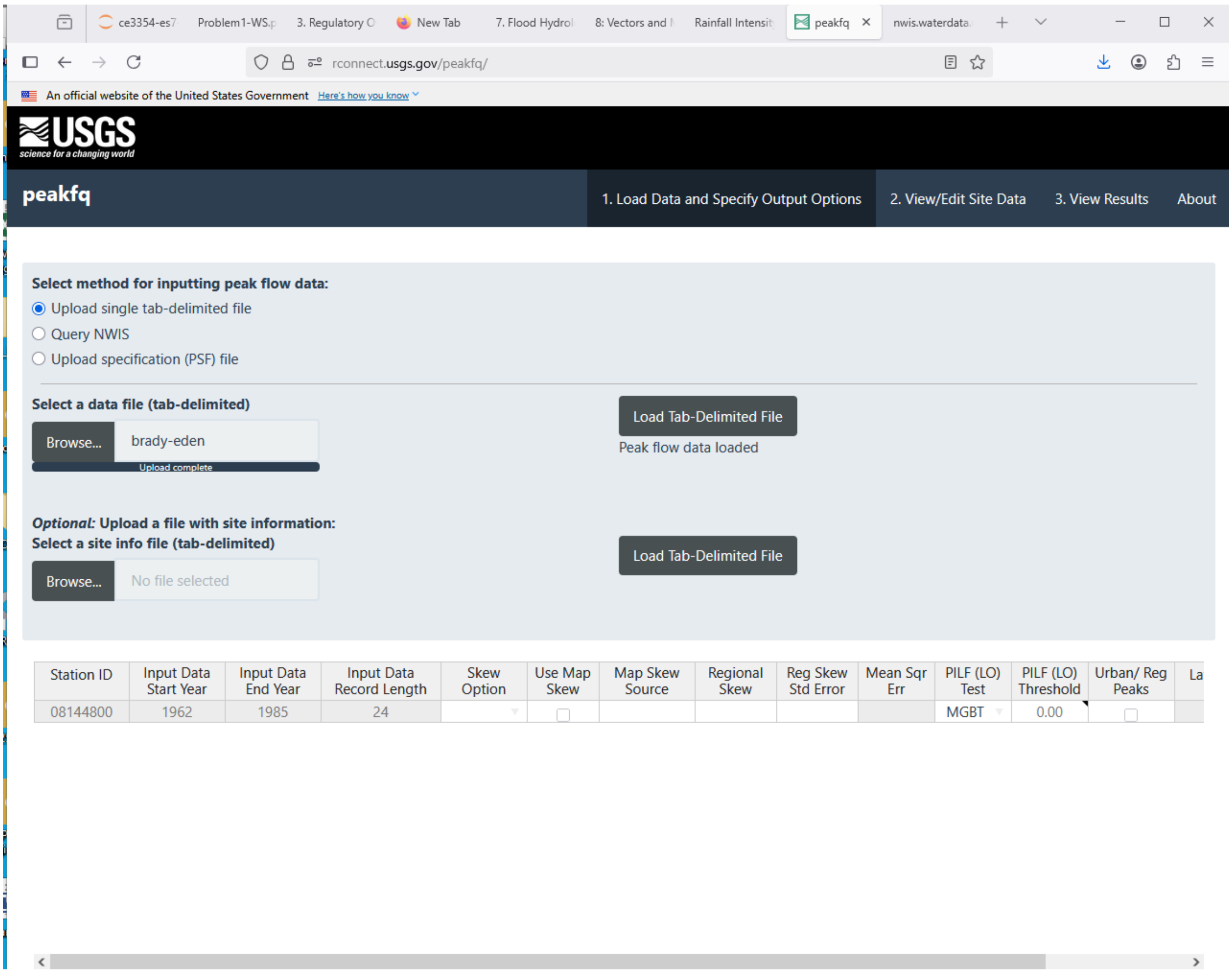


Figure 8: PeakFQ Landing Page - uploaded

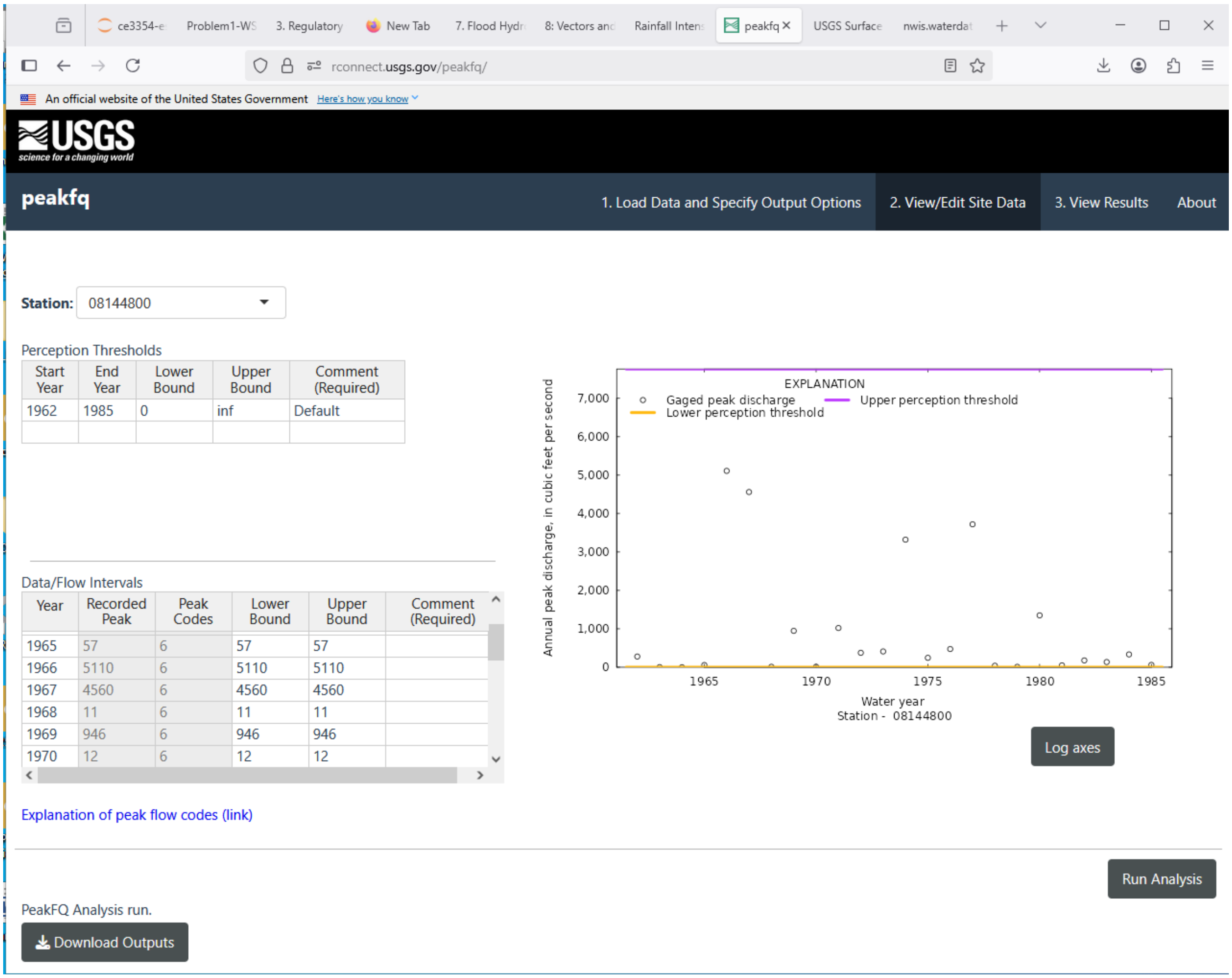


Figure 9: PeakFQ run completed. Had to select station skew and urban peaks

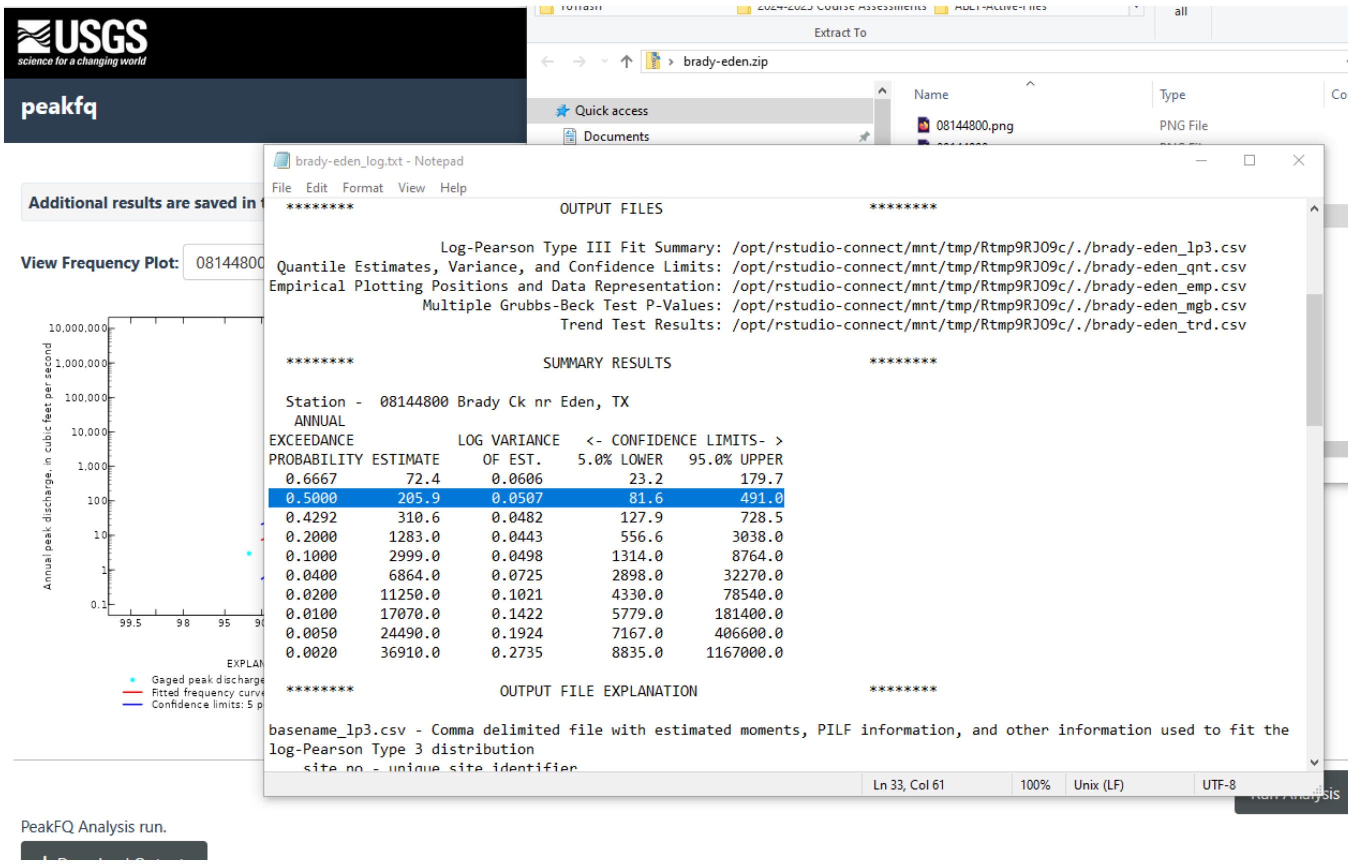


Figure 10: Tabulated output

From the tabulated output we can determine the discharge per square mile as

$$\frac{Q_{50\%}}{mi^2} = \frac{205.9\ cfs}{101\ mi^2} = 2.04 \frac{cfs}{mi^2} \tag{1}$$