In [41]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import defaultdict
plt.style.use(['fivethirtyeight'])
```

In [42]:
```
pop_const = 314
video_df = pd.read_csv('videodata.txt', delim_whitespace=True)
```

In [43]:
```
video_df.head()
```

Out[43]:

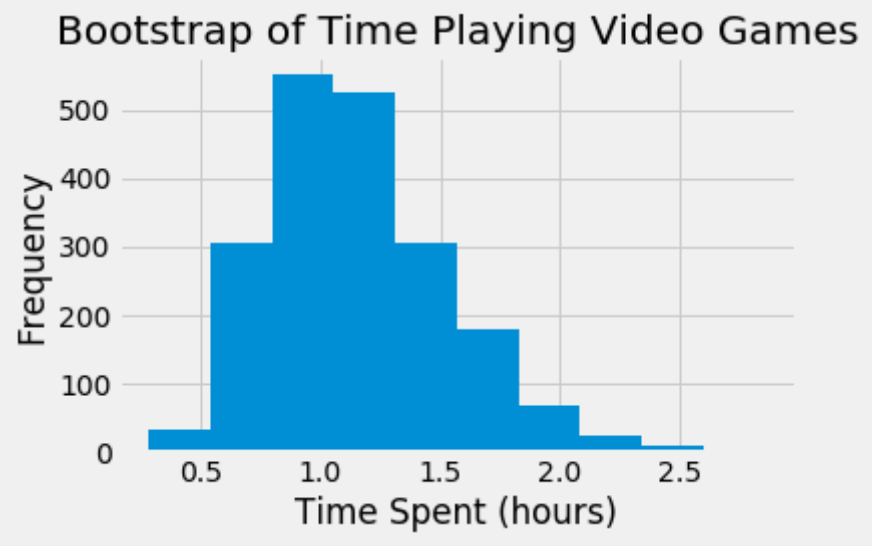
	time	like	where	freq	busy	educ	sex	age	home	math	work	own	cdrom	email	grade
0	2.0	3	3	2	0	1	0	19	1	0	10	1	0	1	4
1	0.0	3	3	3	0	0	0	18	1	1	0	1	1	1	2
2	0.0	3	1	3	0	0	1	19	1	0	0	1	0	1	3
3	0.5	3	3	3	0	1	0	19	1	0	0	1	0	1	3
4	0.0	3	3	4	0	1	0	19	1	1	0	0	0	1	3

(Scenario 4?) Bootstrap of Time Spent Playing Video Games

In [44]:
```
groups = video_df.groupby('time').groups
proportions = {time: int(np.floor((len(g) * 1.0 / 91) * pop_const)) for time
weights = [proportions[t] for t in video_df.time.values[:]]
boot_pop = video_df.groupby('time').apply(lambda g: g.sample(n=proportions[g
```

In [45]:
```
bootstrap_means = [np.mean(boot_pop.sample(91, replace=True).time.values[:])
```

In [46]:
```
plt.hist(bootstrap_means)

plt.title('Bootstrap of Time Playing Video Games')
plt.xlabel('Time Spent (hours)')
plt.ylabel('Frequency')
plt.tight_layout()

plt.savefig('boostrapped_time_spent', dpi=420)
plt.show()
```

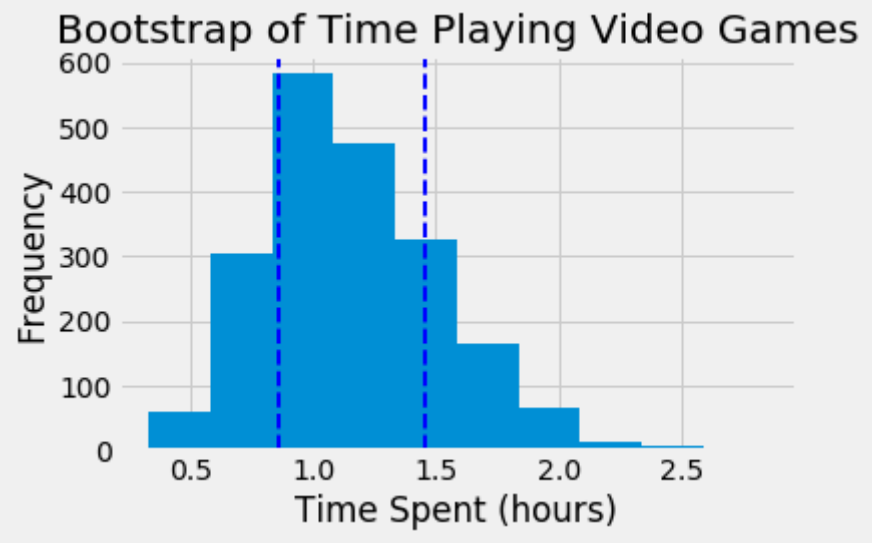


In [47]:
```
mean = np.mean(bootstrap_means)
s = 1/(91-1)*np.sum([(x - mean)**2 for x in bootstrap_means])


# Dependence biased std
s

# Dependence fixed variance
((s**2)*(len(boot_pop.time) - 91))/(len(boot_pop)*91)

sd = np.sqrt(((s**2)*(len(boot_pop.time) - 91))/(len(boot_pop)*91))

c_hi = np.mean(bootstrap_means) + sd*1.96

c_lo = np.mean(bootstrap_means) - sd*1.96

print("(%f,%d)" %(c_hi, c_lo))
```

In [24]:
```
#(len(boot_pop.time)* 91)
```

Out[24]: 28119

In [32]:
```
plt.hist(bootstrap_means)

plt.title('Bootstrap of Time Playing Video Games')
plt.xlabel('Time Spent (hours)')
plt.ylabel('Frequency')
plt.tight_layout()

line_kwargs = {'linewidth' : 2, 'linestyle' :'dashed'}
plt.axvline(**line_kwargs, x=c_lo, color='blue', label='Mean Proportion')
plt.axvline(**line_kwargs, x=c_hi, color='blue', label='Mean Proportion')

plt.savefig('boostrapped_time_spent', dpi=420)
plt.show()
```



In [33]:
```
c_lo
```

Out[33]:
```
0.86141302664518404
```

In [34]:
```
c_hi
```

Out[34]:
```
1.4526122480800909
```

In [ ]:
```
print("point estimate", np.mean(bootstrap_means))
stderr = np.sqrt(np.mean(bootstrap_means) * (1- np.mean(bootstrap_means))) /
CI_lb = (np.mean(bootstrap_means) - (1.96*stderr))
CI_up = (np.mean(bootstrap_means) + (1.96*stderr))
IE = (CI_lb, CI_up)
print("interval estimate:", IE)
```

(Scenario 1) Bootstrapped Proportions

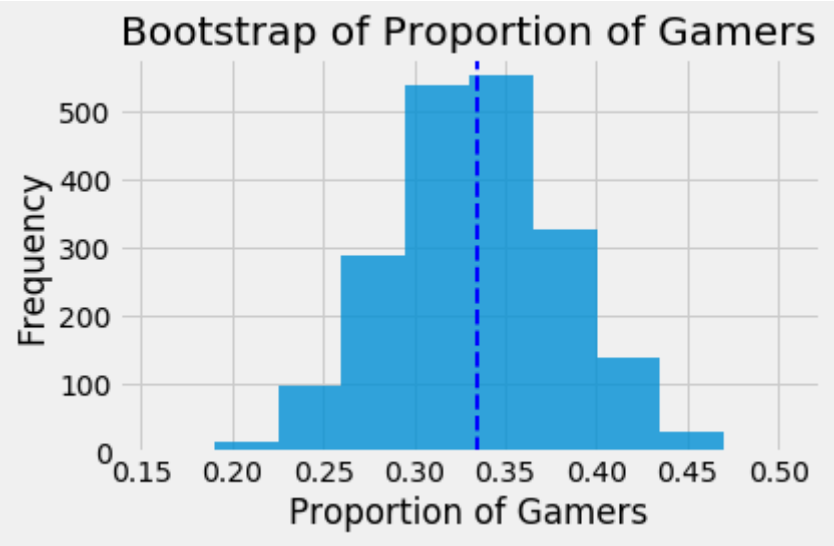
In [63]:
```
video_df.loc[ (video_df.time >=1), 'time' ] = 1.0
```

In [64]:
```
groups = video_df.groupby('time').groups
```

In [65]:
```
groups = video_df.groupby('time').groups
proportions = {time: int(np.floor((len(g) * 1.0 / 91) * pop_const)) for time
weights = [proportions[t] for t in video_df.time.values[:]]
boot_pop = video_df.groupby('time').apply(lambda g: g.sample(n=proportions[g
```

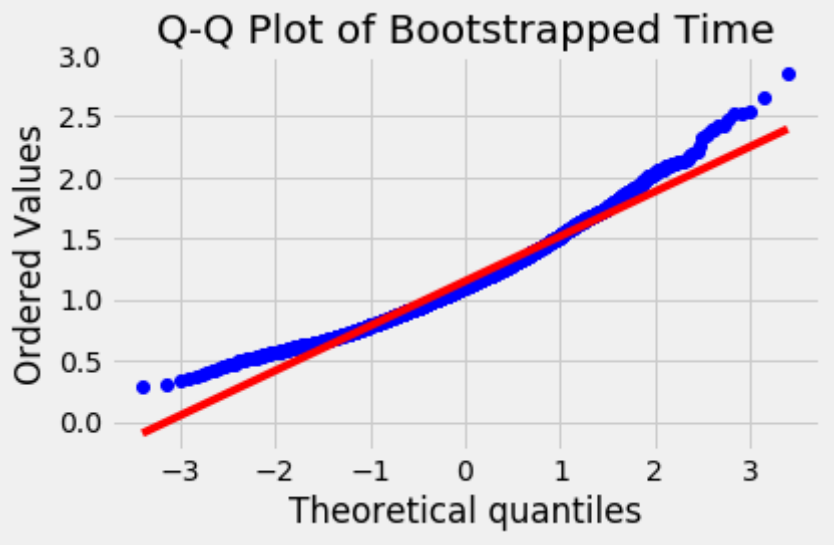
In [66]:
```
bootstrap_means = [np.mean(boot_pop.sample(91, replace=True).time.values[:])
```

In [71]:
```
plt.hist(bootstrap_means, alpha=0.8)
line_kwargs = {'linewidth' : 2, 'linestyle' :'dashed'}
plt.axvline(**line_kwargs, x=np.mean(bootstrap_means), color='blue', label='

plt.title('Bootstrap of Proportion of Gamers')
plt.xlabel('Proportion of Gamers')
plt.ylabel('Frequency')
plt.tight_layout()

plt.savefig('boostrapped_proportion', dpi=420)
plt.show()
```



Part 5

In [48]:
```
from scipy.stats import probplot
probplot(bootstrap_means, plot=plt)

plt.title('Q-Q Plot of Bootstrapped Time')
plt.tight_layout()
plt.savefig('qq_boostrapped_time_spent', dpi=420)

plt.show()
```



In [52]:
```
from scipy.stats import skew
skew(bootstrap_means)
```

Out[52]:
```
0.7074124299397307
```

In [49]:
```
from scipy.stats import kurtosis
kurtosis(bootstrap_means, fisher=False)
```

Out[49]:
```
3.5592567232309076
```

(Scenario 6)

Just for fun, further investigate the grade that students expect in the course. How does it match the standard distribution used in grade assignment (20% A's, 30% B's, 40% C's and 10% D's or lower)?

If the nonrespondents were failing students who no longer bothered to come to the discussion section, would this change the picture ?

videodata variable meanings

- Time : # of hours played in the week prior to survey
- Like to play : 1=never played, 2=very much, 3=somewhat, 4=not really, 5=not at all.
- Where play : 1=arcade, 2=home system, 3=home computer, 4=arcade and either home computer of system, 5= home computer and system, 6=all three.
- How often : 1=daily, 2=weekly, 3=monthly, 4=semesterly.
- Play if busy : 1=yes, 0=no.

- Playing educational : 1=yes, 0=no.
- Sex : 1=male, 0=female.
- Age : Students age in years.
- Computer at home : 1=yes, 0=no.
- Hate math: 1=yes, 0=no.
- Work: # of hours worked the week prior to the survey.
- Own PC : 1=yes, 0=no.
- PS has CD-Rom: 1=yes, 0=no.
- Have email : 1=yes, 0=no.
- Grade expected : 4=A,3=B,2=C,1=D,0=F.

In [40]:
```
max(video_df.age)
```

Out[40]: 33

In [38]:
```
max(video_df.work)
```

Out[38]: 99

In [72]:
```
# Reload the data
pop_const = 314
video_df = pd.read_csv('videodata.txt', delim_whitespace=True)
video_df.head()
```

Out[72]:

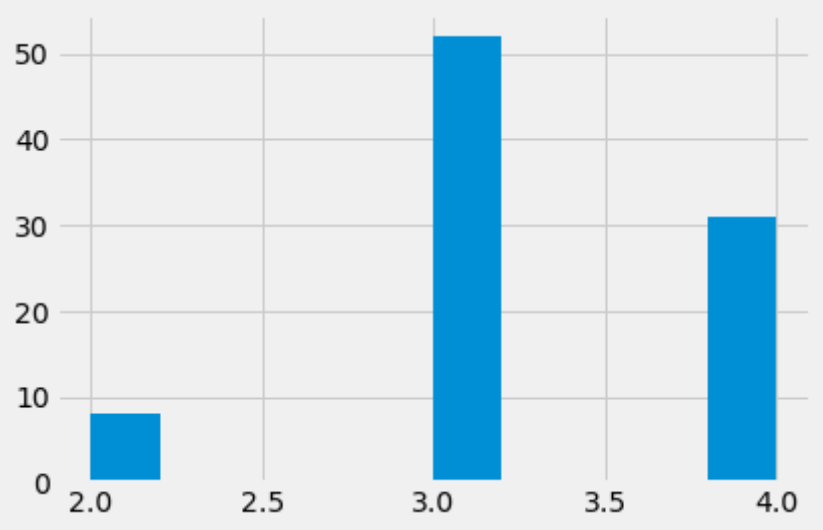
	time	like	where	freq	busy	educ	sex	age	home	math	work	own	cdrom	email	grade
0	2.0	3	3	2	0	1	0	19	1	0	10	1	0	1	4
1	0.0	3	3	3	0	0	0	18	1	1	0	1	1	1	2
2	0.0	3	1	3	0	0	1	19	1	0	0	1	0	1	3
3	0.5	3	3	3	0	1	0	19	1	0	0	1	0	1	3
4	0.0	3	3	4	0	1	0	19	1	1	0	0	0	1	3

In [ ]:
```
```

In [73]:
```
list(set(video_df.grade))
```

Out[73]: [2, 3, 4]

In [83]:
```
plt.hist(video_df.grade)
plt.show()
```



In [88]:
```
from collections import Counter
Counter(video_df.grade)
```

Out[88]:
```
Counter({2: 8, 3: 52, 4: 31})
```

In [76]:
```
.1*len(video_df)
```

Out[76]:
```
9.1
```

In [77]:
```
A = 2/9
B = 3/9
C = 4/9
```

In [84]:
```
31/91
```

Out[84]:
```
0.34065934065934067
```

In [ ]:

In [143]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

plt.style.use(['fivethirtyeight'])
```

Load data

In [144]:
```
pop_const = 314
video_df = pd.read_csv('videodata.txt', delim_whitespace=True)
```

Scenario 1 (point estimate, interval estimate [gamer/non gamer])

In [145]:
```
v_df_nongamer = video_df[video_df['time'] == 0.0]
v_df_gamer = video_df[video_df['time'] != 0.0]

xbar = len(v_df_gamer) / len(video_df)

stderr = np.sqrt(xbar * (1 - xbar)) / np.sqrt(len(video_df) - 1)
stderr *= np.sqrt(pop_const - len(video_df)) / np.sqrt(pop_const)
ie_gamer = (xbar - (1.96 * stderr), xbar + (1.96 * stderr))
```

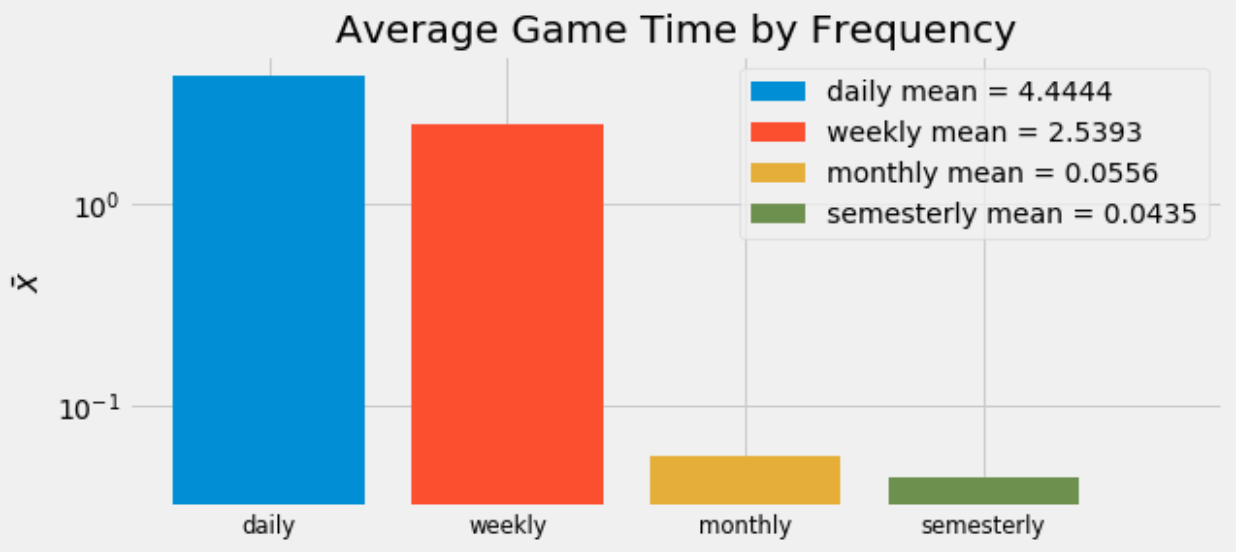
In [171]:
```
print("point estimate:", xbar)
print("interval estimate:", ie_gamer)
```

```
point estimate: 0.37362637362637363
interval estimate: (0.28939809634670244, 0.4578546509060448)
```

In [ ]:
```

```

In [62]:
```
video_daily_df = video_df[video_df['freq'] == 1]
mean_vid_daily = np.mean(video_daily_df['time'])
std_vid_daily = np.std(video_daily_df['time'])

video_weekly_df = video_df[video_df['freq'] == 2]
mean_vid_weekly = np.mean(video_weekly_df['time'])
std_vid_weekly = np.std(video_weekly_df['time'])

video_monthly_df = video_df[video_df['freq'] == 3]
mean_vid_monthly = np.mean(video_monthly_df['time'])
std_vid_monthly = np.std(video_monthly_df['time'])

video_semesterly_df = video_df[video_df['freq'] == 4]
mean_vid_semesterly = np.mean(video_semesterly_df['time'])
std_vid_semesterly = np.std(video_semesterly_df['time'])
```

In [212]:
```
plt.figure(figsize=(9,4))

means = [mean_vid_daily, mean_vid_weekly, mean_vid_monthly, mean_vid_semeste
labels = ['daily', 'weekly', 'monthly', 'semesterly']

for i in range(4):
    plt.bar([i],
            height=[means[i]],
            label='{} mean = {:.4f}'.format(labels[i], means[i]),
            log=True,)
            #yerr=[std_vid_daily, std_vid_weekly, std_vid_monthly, std_vid_s

plt.xticks(x, labels, size='small')
plt.ylabel(r'$\bar{x}$')
plt.title('Average Game Time by Frequency')
plt.legend()
plt.show()
```



In [ ]:
```
```

Scenario 3 (normality assumption incoming)

In [174]:
```
xbar_time = np.mean(video_df['time'])
std_time = np.std(video_df['time'])
err_time = xbar_time / np.sqrt(len(video_df['time']))
ie_time = (xbar_time - err_time, xbar_time + err_time)

print("point estimate:", xbar_time)
print("interval estimate:", ie_time)
```

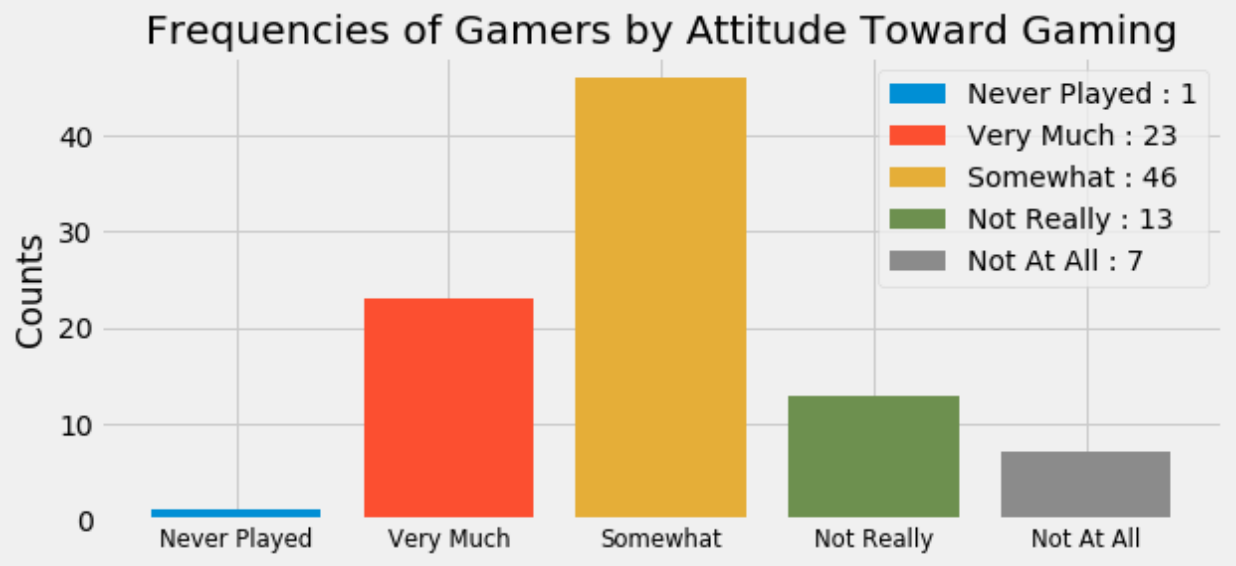
```
point estimate: 1.2428571428571429
interval estimate: (1.1125703131502758, 1.37314397256401)
```

In [ ]:
```
```

Scenario 4

In [182]:
```
num_np = len(video_df[video_df['like'] == 1])
num_vm = len(video_df[video_df['like'] == 2])
num_some = len(video_df[video_df['like'] == 3])
num_nr = len(video_df[video_df['like'] == 4])
num_naa = len(video_df[video_df['like'] == 5])
```

In [189]:
```
plt.figure(figsize=(9,4))

counts = [num_np, num_vm, num_some, num_nr, num_naa]
labels = ['Never Played', 'Very Much', 'Somewhat', 'Not Really', 'Not At All

for i in range(5):
    plt.bar([i],
            height=[counts[i]],
            label='{} : {}'.format(labels[i], counts[i]),
            log=False,)

plt.xticks(x, labels, size='small')
plt.ylabel('Counts')
plt.title('Frequencies of Gamers by Attitude Toward Gaming')
plt.legend()
plt.show()
```



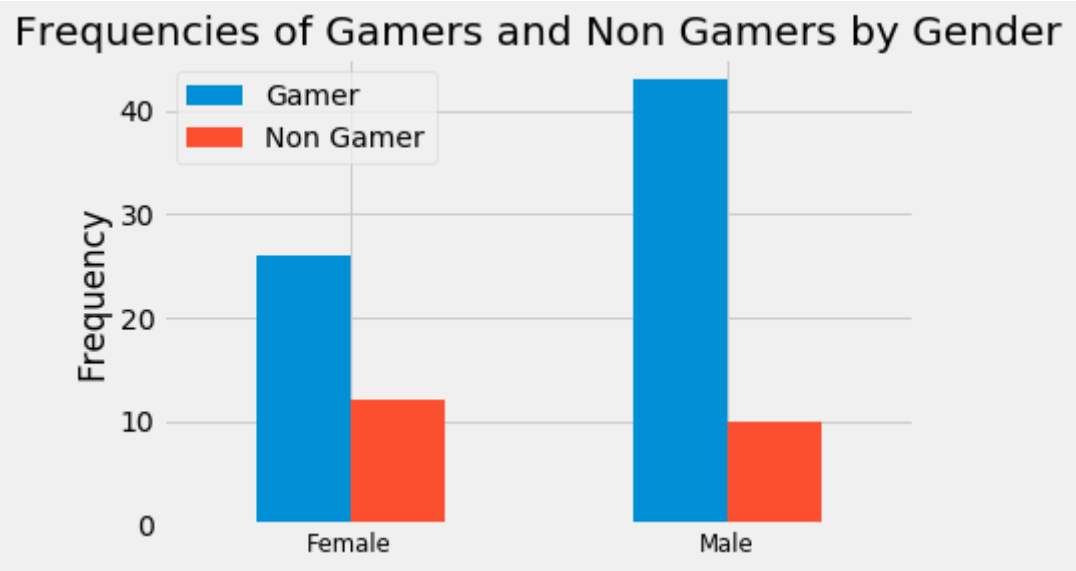
Scenario 5

In [ ]:

In [ ]:

In [ ]:

In [271]:
```
def isgamer(row):
    isg = int((row['like'] == 2) | (row['like'] == 3))

    if isg:
        return 'Gamer'
    return 'Non Gamer'

video_df['isgamer'] = video_df.apply(lambda row: isgamer(row), axis=1)

video_df[['isgamer', 'sex']].pivot_table(index='sex',
               columns='isgamer',
               aggfunc=len,
               fill_value=0).plot(kind='bar')

plt.xticks([0, 1], ['Female', 'Male'], size='small', rotation=0)
plt.xlabel('')
plt.ylabel('Frequency')
plt.title('Frequencies of Gamers and Non Gamers by Gender')
plt.legend()

plt.show()
```

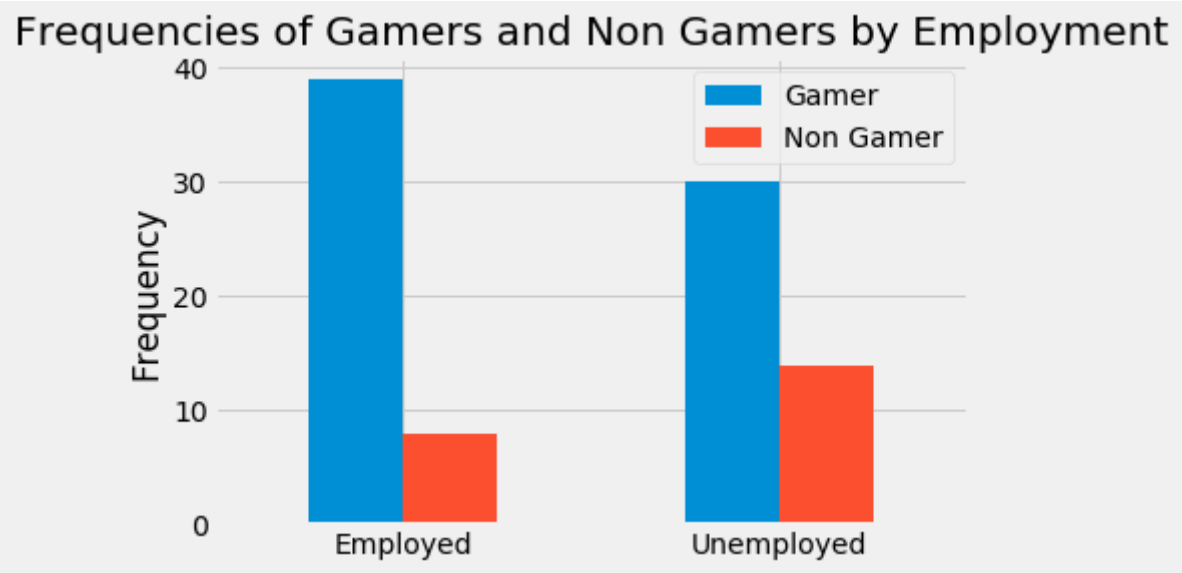


In [ ]:

In [ ]:

In [ ]:

In [272]:
```
def isemployed(row):
    ise = row['work'] != 0

    if ise:
        return 'Employed'
    return 'Unemployed'

video_df['employed'] = video_df.apply(lambda row: isemployed(row), axis=1)

video_df[['isgamer', 'employed']].pivot_table(index='employed',
               columns='isgamer',
               aggfunc=len,
               fill_value=0).plot(kind='bar')

plt.xticks(rotation=0)
plt.xlabel('')
plt.ylabel('Frequency')
plt.title('Frequencies of Gamers and Non Gamers by Employment')
plt.legend()

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
setwd("~/Desktop/workspace/math189/cs2/")
data <- read.table("videodata.txt", header=TRUE)

library(rpart)
library(rpart.plot)


data$dis_like<- rep(NA, dim(data)[1])

data = data[ which( data$work != 99) , ]
data = data[ which( data$educ != 99) , ]


for(i in 1:dim(data)[1]){
  like <- data[i, 'like']
  if(like==1 || like==4 || like==5 || like==3){
    data[i, 'dis_like'] = 0
  }else{
    data[i, 'dis_like'] = 1
  }
}

# Create a decision tree model
tree <- rpart(dis_like~educ+sex+age+home+math+work+own+cdrom+grade,
data=data, cp=.02)
# Visualize the decision tree with rpart.plot
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE)
title("Classification Decision Tree on Liking Video Games", line=2.5)
```