In [72]:
```python
# Packages
# from matplotlib import (pyplot as plt, lines)
# import seaborn as sns
import numpy as np
import os
import pandas as pd

```

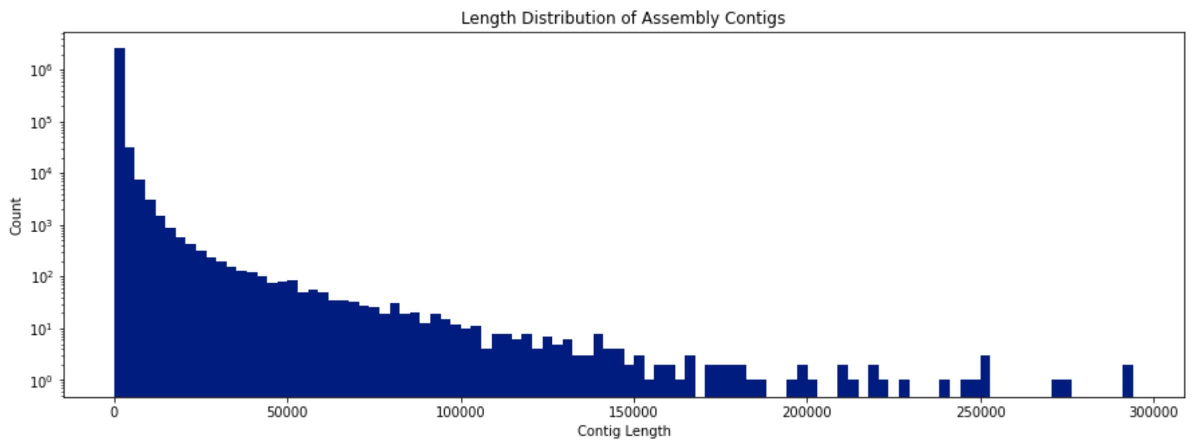
In [63]:
```python
# Globals
BASE_DIR = "/home/josh/PycharmProjects/eces-450/tutorial/data/algae


# Given files
print("Assembly Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,
print("\nRead Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,
```

```
Assembly Dir:
simple.contig.fa

Read Dir:
CSJP002C_R1.fastq
CSJP002A_R2.fastq
CSJP002B_R2.fastq
CSJP002A_R1.fastq
CSJP002B_R1.fastq
CSJP002C_R2.fastq
```

In [73]:
```python
# Read Assembly File
fn = os.path.join(BASE_DIR, 'assembly', 'simple.contig.fa')
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contig_lengths
contig_lengths = []
contigs = []
i = 0
for line in lines:
    if line[0] == '>':
        contigs.append(line[1:])  # grab record contig id
    elif i < 1000:
        contig_lengths.append(len(line))  # grab record sequence ler
    if i<2:
        print(line[0:250], end='')  # print the first record
        i+=1

# Plot the sequence lengths
plt.style.use('seaborn-dark-palette')
fig = plt.figure(figsize=(15, 5))
plt.hist(contig_lengths, bins=100, log=True)
plt.title("Length Distribution of Assembly Contigs")
plt.xlabel("Contig Length")
plt.ylabel("Count")
```

```
>contig-65_0
TGGCAGGCGATGTCGGAGCGCAAACTCCGCCGCCAGCGCGTCATCTTCTACATCTCACGTGGGTTGGAGA
ACGCCCCGACAAAGTTCCAGACCACCCCCGCCCGTCGCGACTACTACAACACGCTGCTCGACCAGCTTCA
GACCGAGTTCGAGCACGTTCATCGGCTGCTGCTGGAAATCTTCGGTTCAGGCGGCGCGCGCGTGTTGCCG
ATGACGGATGCCGACCACTTCCGGCACTACAAACGGTTTC
```

Out[73]:
```
Text(0, 0.5, 'Count')
```



```
bwa index

Usage:   bwa index [options]

Options: -a STR    BWT construction algorithm: bwtsw, is or rb2
[auto]
         -p STR    prefix of the index [same as fasta name]
         -b INT    block size for the bwtsw algorithm (effectiv
e with -a bwtsw) [10000000]
         -6        index files named as .64.* instead of .*

Source: bwa man pages
```

```
A bit of googling and I found:
.amb is text file, to record appearance of N (or other non-ATG
C) in the ref fasta.
.ann is text file, to record ref sequences, name, length, etc.
.bwt is binary, the Burrows-Wheeler transformed sequence.
.pac is binary, packaged sequence (four base pairs encode one b
yte).
.sa is binary, suffix array index.
```

Source: [http://seqanswers.com/forums/showthread.php?t=25553 (http://seqanswers.com/forums/showthread.php?t=25553)](http://seqanswers.com/forums/showthread.php?t=25553)

In [74]:

```
# After running bwa index on the assembly file, several new files a
print("Assembly Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,
```

```
Assembly Dir:
simple.contig.fa.amb
simple.contig.fa.ann
simple.contig.fa.pac
simple.contig.fa.sa
simple.contig.fa.bwt
simple.contig.fa
```

```
bwa mem

The  BWA-MEM  algorithm performs local alignment. It may produce
multiple primary alignments for different part of a query sequence.
This is a crucial feature for long sequences. However, some tools
such as Picard's markDuplicates does not work with split
alignments. One may consider to use option -M to flag shorter split
hits as secondary.

```

Bash script to iteratively generate bams

```
#!/bin/bash
#### Create a map from the reads to the newly indexed assembly-
file
#### This took 10.5 hours to complete on proteus

BASE_DIR="./"
samples=(2A 2B 2C)

for sample in ${samples[@]}
    do
        echo CSJP00${sample}_R1.fastq
        bwa mem ${BASE_DIR}assembly/simple.contig.fa ${BASE_DI
R}reads/CSJP00${sample}_R1.fastq ${BASE_DIR}reads/CSJP00${sampl
e}_R2.fastq | samtools view -b -o ${BASE_DIR}mapped/$sample.bam
done
```

In [4]:
```python

print("\nMapped:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,
```

```
Mapped:
2C.bam
2A.bam
2B.bam
```

```
# Sort bams with samtools
<pre>
Usage: samtools sort [options...] [in.bam]
Description:

    Sort  alignments  by leftmost coordinates, or by read name when
-n is used.  An appropriate @HD-SO sort order header tag will be
added or an existing one updated if necessary.

    The sorted output is written to standard output by default, or
to the specified file (out.bam) when -o is used.  This  command
will  also  create  temporary files tmpprefix.%d.bam as needed when
the entire alignment data cannot fit into memory (as controlled via
the -m option).

Options:
  -l INT     Set compression level, from 0 (uncompressed) to 9
(best)
  -m INT     Set maximum memory per thread; suffix K/M/G recognized
[768M]
  -n         Sort by read name
  -t TAG     Sort by value of TAG. Uses position as secondary index
(or read name if -n is set)
  -o FILE    Write final output to FILE rather than standard output
  -T PREFIX  Write temporary files to PREFIX.nnnn.bam
  --no-PG    do not add a PG line
      --input-fmt-option OPT[=VAL]
```

```
               Specify a single input file format option in the
form
               of OPTION or OPTION=VALUE
  -O, --output-fmt FORMAT[,OPT[=VAL]]...
               Specify output format (SAM, BAM, CRAM)
      --output-fmt-option OPT[=VAL]
               Specify a single output file format option in the
form
               of OPTION or OPTION=VALUE
      --reference FILE
               Reference sequence FASTA FILE [null]
  -@, --threads INT
               Number of additional threads to use [0]
      --verbosity INT
               Set level of verbosity


-------------------------------------------------------------------
-----------------------------------
samtools sort -o ./sorted/2A.sorted.bam ./mapped/2A.bam
samtools sort -o ./sorted/2B.sorted.bam ./mapped/2B.bam
samtools sort -o ./sorted/2C.sorted.bam ./mapped/2C.bam
-------------------------------------------------------------------
-----------------------------------
</pre>
```

In [48]:

```
# Sort the bam files for rapid processing, can also be run on proteu

print("\nSorted:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,'s
```

```
Sorted:
2B.sorted.bam
2C.sorted.bam
2A.sorted.bam
```

```
[js3973@proteusa01 Tutorial6_data]$ ls -al mapped
total 23896668
drwxrwsr-x 2 js3973 rosenclassGrp       4096 May 18 10:35 .
drwxrwsr-x 5 sk3389 rosenclassGrp       4096 May 18 10:32 ..
-rw-r--r-- 1 js3973 rosenclassGrp 5866752241 May 18 03:34 2A.ba
m
-rw-r--r-- 1 js3973 rosenclassGrp 6771359549 May 18 07:16 2B.ba
m
-rw-r--r-- 1 js3973 rosenclassGrp 5346570250 May 18 10:13 2C.ba
m
-rw-r--r-- 1 js3973 rosenclassGrp  581851433 May 18 10:37 2C.so
rted.bam
-rw-r--r-- 1 js3973 rosenclassGrp  361791128 May 18 10:26 2C.so
rted.bam.tmp.0000.bam
-rw-r--r-- 1 js3973 rosenclassGrp  361769335 May 18 10:27 2C.so
rted.bam.tmp.0001.bam
```

```
-rw-r--r-- 1 js3973 rosenclassGrp  363248908 May 18 10:27 2C.so
rted.bam.tmp.0002.bam
-rw-r--r-- 1 js3973 rosenclassGrp  364979550 May 18 10:28 2C.so
rted.bam.tmp.0003.bam
-rw-r--r-- 1 js3973 rosenclassGrp  362689520 May 18 10:28 2C.so
rted.bam.tmp.0004.bam
-rw-r--r-- 1 js3973 rosenclassGrp  360639096 May 18 10:29 2C.so
rted.bam.tmp.0005.bam
-rw-r--r-- 1 js3973 rosenclassGrp  363002478 May 18 10:30 2C.so
rted.bam.tmp.0006.bam
-rw-r--r-- 1 js3973 rosenclassGrp  363244902 May 18 10:30 2C.so
rted.bam.tmp.0007.bam
-rw-r--r-- 1 js3973 rosenclassGrp  355685132 May 18 10:31 2C.so
rted.bam.tmp.0008.bam
-rw-r--r-- 1 js3973 rosenclassGrp  357044755 May 18 10:32 2C.so
rted.bam.tmp.0009.bam
-rw-r--r-- 1 js3973 rosenclassGrp  359320772 May 18 10:32 2C.so
rted.bam.tmp.0010.bam
-rw-r--r-- 1 js3973 rosenclassGrp  358477722 May 18 10:33 2C.so
rted.bam.tmp.0011.bam
-rw-r--r-- 1 js3973 rosenclassGrp  355302152 May 18 10:33 2C.so
rted.bam.tmp.0012.bam
-rw-r--r-- 1 js3973 rosenclassGrp  358575023 May 18 10:34 2C.so
rted.bam.tmp.0013.bam
-rw-r--r-- 1 js3973 rosenclassGrp  360891810 May 18 10:35 2C.so
rted.bam.tmp.0014.bam
```

Create Depth Matrix:

First generate depth matrix from sorted bam files using jgi_summarize_bam_contig_depths (included with metabat2)

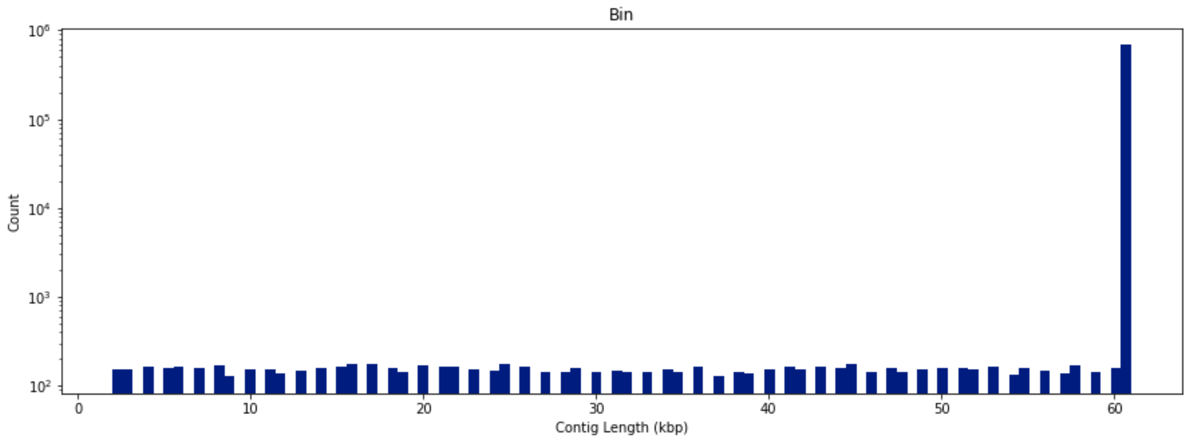
In [86]:
```python
# Read Depth Matrix
fn = os.path.join(BASE_DIR, 'depth', 'depth_matrix.tab')
with open(fn, 'r') as fh:
    df = pd.read_csv(fh, delimiter='\t')
df.head(50)  # Show
```

22	65_22	182600	23.3662	0.013412	0.020377	13.125900
23	contig-65_23	182193	14.7668	0.017331	0.025937	14.648100
24	contig-65_24	182140	16.1361	0.015869	0.018644	4.641220
25	contig-65_25	177867	15.1804	0.013932	0.023602	9.764390
26	contig-65_26	176673	34.4763	0.044357	0.060298	0.824199
27	contig-65_27	176345	24.1709	0.023406	0.028539	13.847500
28	contig-65_28	174764	15.0156	0.014123	0.021368	14.886200

```
Input to metabat2:
sorted bam files
depth matrix

Output:
130 bins
```

In [85]:
```python
# Visualize Contig lengths in largest bin
fn = os.path.join(BASE_DIR, 'bins',"bin.10.fa")
# fn = os.path.join(BASE_DIR, 'bins',"bin.22.fa")
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contig_lengths
contig_lengths = []
for line in lines:
    if not line[0] == '>':
        contig_lengths.append(len(line))

# Plot the sequence lengths
plt.style.use('seaborn-dark-palette')
fig = plt.figure(figsize=(15, 5))
plt.hist(contig_lengths, bins=100, log=True)
plt.title("Bin")
plt.xlabel("Contig Length (kbp)")
plt.ylabel("Count")
```

Out[85]:
```
Text(0, 0.5, 'Count')
```



In [80]:
```python
# Binned data
fn = os.path.join(BASE_DIR, 'bins',"bin.10.fa")
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contigs
contigs = []
for line in lines:
    if line[0] == '>':
        contigs.append(line[1:].strip())

```