HW4

Andrew Risse

USC ID: 5987-0295-24

1a. macOS High Sierra (Version 10.13.3), I have had these files since late 2010, so about 7.5 years.

1b. I found 275240 files.

1c. Average size 1.36MB.

1d. Standard Deviation 46.5MB.

1e. Median 6.39KB.

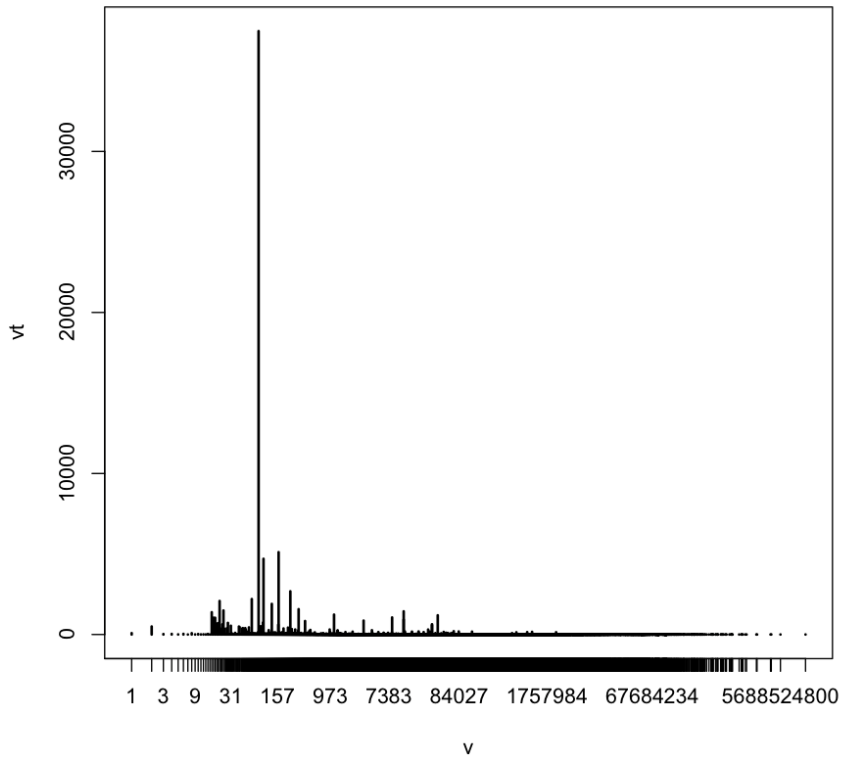
1f. There are more small files.

1g. Mode is 81B.It occurs 37,470 times.

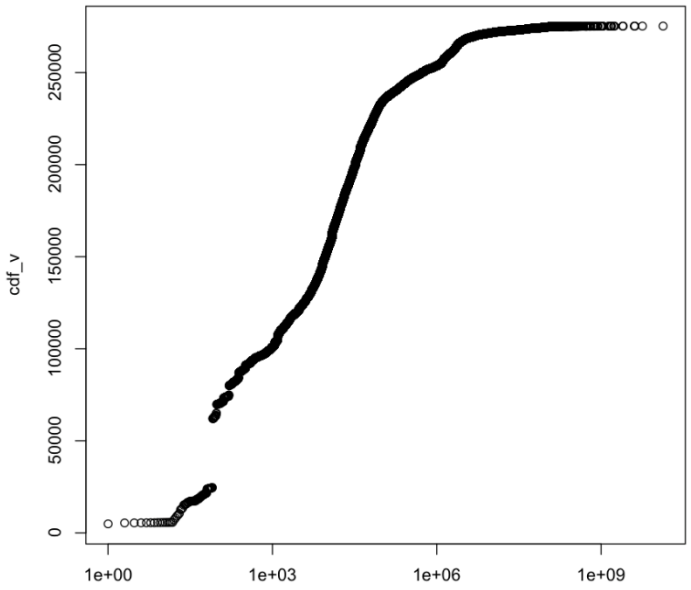
1h. Yes it is significant, it is 13.8% of files.

1i. Most of the 81B files are in ./Library/Application Support/CE/Debug

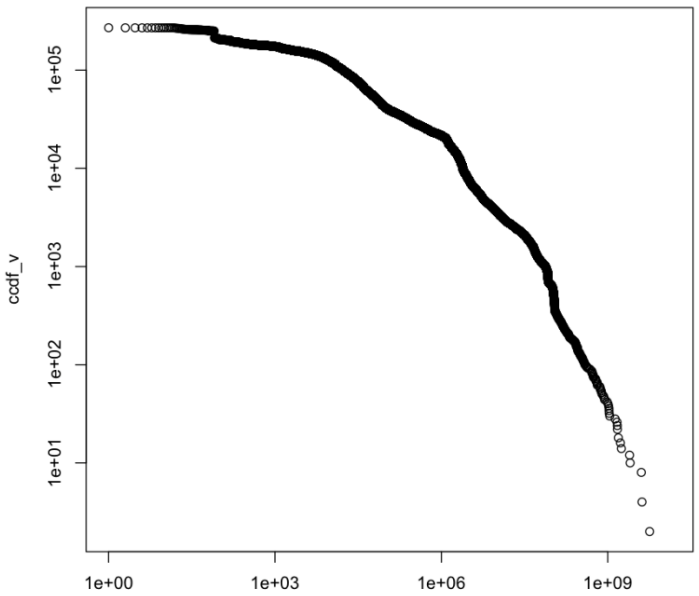
1j. PDF



1k. CDF



1l. CCDF



1m. A heavy tailed distribution in a CCDF would be approximately linear.

1n. Yes.

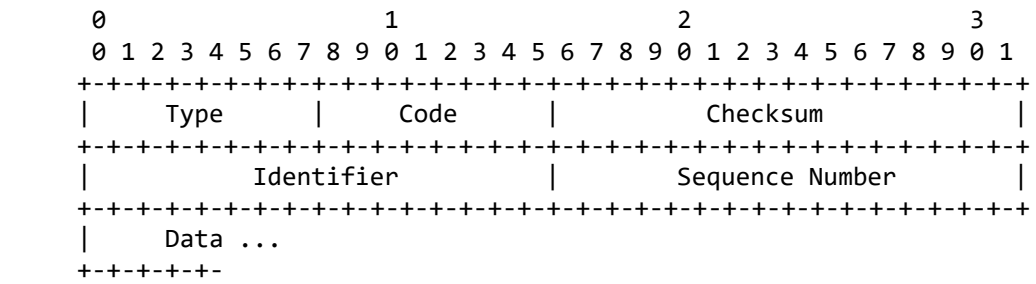
1o. The CDF graph shows heavy tailed distribution because of the approximately linear behavior over significant range (left and right).

1p. I am much less of a pack rat.

2a. RFC 792

2b.

Echo or Echo Reply Message



2c. macOS High Sierra, Version 10.13.3

2d.

```
/*
* Structure of an icmp header.
*/
struct icmp {
        u_char  icmp_type;              /* type of message, see below */
        u_char  icmp_code;              /* type sub code */
        u_short icmp_cksum;             /* ones complement cksum of struct */
        union {
                u_char ih_pptr;                 /* ICMP_PARAMPROB */
                struct in_addr ih_gwaddr;       /* ICMP_REDIRECT */
                struct ih_idseq {
                        n_short icd_id;
                        n_short icd_seq;
        }
         ih_idseq;
        int ih_void;
```

```
                /* ICMP_UNREACH_NEEDFRAG -- Path MTU Discovery (RFC1191) */
                struct ih_pmtu {
                        n_short ipm_void;
                        n_short ipm_nextmtu;
                } ih_pmtu;

                struct ih_rtradv {
                        u_char irt_num_addrs;
                        u_char irt_wpa;
                        u_int16_t irt_lifetime;
                } ih_rtradv;
        } icmp_hun;
```

2e. Yes, and no. It also just has an “int” and just “short”s and it doesn’t define how big those are

2f. It specifies u_int16_t ->16 bit unsigned integer

2g. Linux Fedora 27

2h.

```
struct icmphdr
{
 u_int8_t type;          /* message type */
 u_int8_t code;          /* type sub-code */
 u_int16_t checksum;
 union
 {
  struct
  {
   u_int16_t    id;
   u_int16_t    sequence;
  } echo;                       /* echo datagram */
  u_int32_t     gateway;        /* gateway address */
  struct
  {
```

```
    u_int16_t    __unused;
    u_int16_t    mtu;
   } frag;                          /* path mtu discovery */
  } un;
};
```

2i. This approach specifies the size of each int.

2j. Entire program to test my code included, referenced: https://codereview.stackexchange.com/questions/149717/implementation-of-c-standard-library-function-ntohl

```
#include <stdio.h>
#include <sys/types.h>
#include <string.h>

struct decoded_icmp
{
        unsigned int type,
        code,
        checksum,
        id,
        seqno;
};

void demarshall(unsigned char bytes[], struct decoded_icmp *out)
{
        out->type = (u_int16_t)bytes[0]<<0;
        out->code = (u_int16_t)bytes[1]<<0;
        out->checksum = ((u_int16_t) bytes[3]<<0) | ((u_int16_t) bytes[2]<<8);
        out->id = ((u_int16_t) bytes[5]<<0) | ((u_int16_t) bytes[4]<<8);
        out->seqno = ((u_int16_t) bytes[7]<<0) | ((u_int16_t) bytes[6]<<8);
}
```

```
int main()
{
	struct decoded_icmp out;
	unsigned char bytes[] = { 1, 2, 3, 69, 6, 120, 9, 171 };
	demarshall(bytes, &out);
	printf("%08x, %08x, %08x, %08x, %08x\n", out.type, out.code, out.checksum, out.id, out.seqno);
	return 0;
}
```