Empirical Statistical Analysis of the Generalized Hirsch Index

BY

Ha Luu

An essay submitted to the
Graduate School-NewBrunswick
Rutgers, The State University of New Jersey
In partial fullfillment of the requirements
For the degree of
Master of Science
Graduate Program in Mathematics
Written under the direction of
Dr. Doron Zeilberger
And approved by:

New Brunswick, New Jersey
October 2014

Contents

Acknowledgements

I would like to thank Dr. Zeilberger for leading me to this interesting project and teaching me the valuable skills of studying math using Maple.

Abstract

In this paper, we will discuss how to use Maple programming to empirically investigate the asymptotic normality of the generalized Hirsch citation index (alias size of Durfeee rectangle) with respect to the uniform distribution on the sample space of integer partitions of n.

All the results of this article were obtained using Maple programs, and is available at: http://math.rutgers.edu/ hbl15/em15/HIRSCHgData.txt

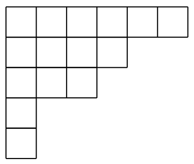
Chapter 1

Introduction

1.1 Informal Overview

1.1.1 The general h-index

The Ferrers diagram corresponding to a partition λ is a graphical representation of λ. To construct the Ferrers diagram for $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$, simply place a row of λ_{i+1} left justified blocks on top of λ_i blocks, for each $i = 1, 2, \ldots, k-1$. For example, the Ferrers diagram for the partition $\lambda = (6, 4, 3, 1, 1)$ is :



The Hirsch index (h-index), $h(\lambda)$ is the largest λ_i such that the $\lambda_i \times \lambda_i$ square fit inside the Ferrers diagram, in this case $h(\lambda) = 3$. And for **the general h-index**, $h(\lambda, a, b)$ is the largest λ_i such that the $(\lambda_i) \times (a\lambda_i + b)$ rectangle fit inside the Ferrers diagram.

1.1.2 Probability Review

In probability theory, a **sample space** is the set of all possible outcomes for an experiment. For example, if you toss a coin three times, the sample space for this experiment is S_3 ={HHH, HHT, HTH, THH, HTT, THT, TTH, TTT}, or shortly $\{H,T\}^3$

Random variable, $X : S \rightarrow R$, is the function that maps each event of in the sample space (S) into a real value number that represent at feature of interested, for example mapping HHT into the number of heads : $X(HHT) = 2$

Probability distribution is a function that map measurable subset of the possible outcomes in the sample space to a probability of occurrence. Continue from coin tossing example above, if the probability of head is p ($p \in [0,1]$) then the probability of getting HHH is: $p_{HHH} = p^3$. The general probability distribution for a single event, s,in the coin tossing experiment is : $p_s = p^{NumberOfHeads}(1-p)^{NumberOfTails}$. ($\sum_{s \in S} p_s = 1$). We can also combine single events with similar features and get a probability distribution for this "compound event" , such as the probability of getting exactly 2 heads out of 3 tosses: $Pr(s \in S|X(s) = 2) = \sum_{s \in S|X(s)=2} p_s = \binom{3}{2} p^2(1-p)$. The general probability distribution formula for getting exactly k heads out of n tosses is :

$$Pr(s \in S|X(s) = k) = \sum_{s \in S|X(s)=k} p_s = \binom{n}{k} p^k (1-p)^{n-k}$$

.

Probability generating function, f(t), is a power series of t with valued from probability distribution as coefficient :

$$f(t) = \sum_{r \in R} Pr(X(s) = r)t^r = \sum_{s \in S} p_s t^{X(s)}$$

Expectation, or **mean**, or **first moment** of a random variable X is the weighted average of all possible value that X can take. Each value of X is being weighted by the probability of that value occurring: $Pr(s \in S|X(s) = k) = \sum_{s\in S|X(s)=k} p_s$. Taking previous example, the expected number of heads if you toss a coin n times is : $\mu = E(X) = \sum_{r\in R} Pr(X(s) = r)r = \sum_{s\in S} p^s X(s)$. Expectation plays important role in characterizing the probability distribution, such as the location of shift of the distribution.

Variance, or **second moment** of a random variable X describes the spread of the possible values of X. For a random variable X with expectation $\mu = E[X]$, the variance of X is

$$\sigma^2 = E[(X-\mu)^2] = \sum_{s\in S} p_s(X(s)-\mu)^2$$

Higher moments is defined by the formula :

$$m_r(X) = \sum_{s\in S} p_s(X(s)-\mu)^r$$

Moments are values that help understanding the characters of a distribution such as its skewness, kurtosis... These values plays central role in our method of analyzing the general h-index because of their special pattern when it comes to normal distribution.

Normal distribution of a random variable X, with mean μ and variance σ^2 has a probability distribution satisfies :

$$Pr(a \leq x \leq b) = \frac{1}{\sigma\sqrt{2\pi}} \int_a^b e^{-\dfrac{(x-\mu)^2}{2\sigma^2}} dx$$

When $\sigma = 1$ and $\mu = 0$, then this distribution is called **standard normal distribution**. All normal distribution can be transformed into standard normal distribution by centralized and normalized the random variable by the formula :

$$Z_n = \frac{X_n - \mu}{\sigma}$$

As mentioned above, the sequence of moments (starting from order 1 moment) of a normal distribution when calculated has the pattern: $\{\mu, \mu^2+\sigma^2, \mu^3+2\mu\sigma, \mu^4+6\mu^2\sigma^2+3\sigma^4, \mu^5+10\mu^3\sigma^2+15\mu\sigma^4, \mu^6+15\mu^4\sigma^2+45\mu^2\sigma^4+15\sigma^6, \mu^7+21\mu^5\sigma^2+105\mu^3\sigma^4+105\mu\sigma^6, \mu^8+28\mu^6\sigma^2+210\mu^4\sigma^4+420\mu^2\sigma^6+105\sigma^8, ...\}$

Thus, for a standard normal distribution, its moment sequence (starting from order 1 moment) is always: {0,1,0,3,0,15,0,105,...} (subtituting $\sigma = 1$ and $\mu = 0$ into the sequence above). We used this characteristics to empirically prove whether the general h-index for a Durfee rectangle is asymptotically normal, by using the **refined central limit theorem** in *The Automatic Central Limit Theorems Generator* by Dr. Doron Zeilberger [1].

The Central Limit Theorem : Let $X_1, X_2, ...$ be a sequence of independent and identically distributed random variable with common mean μ and variance σ^2. If $S_n = X_1 + X_2 + ... + X_n$ then the distribution of $\dfrac{S_n - n\mu}{\sigma\sqrt{n}}$ converges to the standard normal distribution, that is for every fixed β :

$$\lim_{n\to\infty} P\{\frac{S_n - n\mu}{\sigma\sqrt{n}} < \beta\} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\beta} e^{-\dfrac{x^2}{2}}\,dx = \mathcal{N}(\beta)$$

Before getting to the refined central limit theorem, one other set of moments that play crucial role in our analyzing method is **factorial moments** :

$$f_r(X) = \sum_{s\in S} p_s(X(s)-\mu)^{(r)}$$

$$= \sum_{s\in S} p_s(X(s)-\mu)(X(s)-\mu-1)...(X(s)-\mu-r+1)$$

Factorial moments are important because they are very easy to calculate from the probability generating function f(t). Supposed we had done our centralization to our random variables and we have mean 0 for all of them, then we can get f_r by differentiating f(t) r times and substitute t = 1:

$$f_r = \frac{d^r f(t)}{dt^r}|_{t=1}$$

With Maple, we can do a Maclaurin expansion around t = 0 for f(1+t):

$$f(1+t) = \sum_{r=0}^{\infty} f_r \frac{t^r}{r!}$$

And we can get the ordinary moments from factorial moments by the connection formula ([GKP],p.250):

$$m_r = \sum_{k=1}^{r} S(r,k) f_r$$

where S(r,k) is the Stirling Numbers of the Second kind, defined as:

$$S(r,k) = kS(r-1,k) + S(r-1,k-1)$$

with S(1,1) = 1 and S(1,k) = 0 for k$\neq$ 1

And without going to much further details, in [1], there is also a nice recurrence formula that connect $f_r(n+1)$ and $f_r(n)$:

$$f_r(n+1) - f_r(n) = \sum_{s=2}^{r} \binom{r}{s} F_s f_{r-s}(n)$$

where F_s is the factorial moments of the single event, $f_r(0) = 0$ and $f_1(n) = 0$

The Refined Central Limit Theorem [1] : Let $\{X_k\}$ be a sequence of mutually independent and identically distributed random variables. Suppose that $\mu = 0$ and $\sigma^2 = 1$ and all the first 2s moments, $M_1 = 0, M_2 = 1, M_3, M_4, ..., M_{2s}$ are finite. Let $S_n = X_1 + X_2 + ... + X_n$, and m_{2r} be the 2r-th moment of S_n, then for even s, then :

$$m_{2r} = \frac{(2r)!}{(2^r r!)(1 + O(\frac{1}{n^s}))}$$

if the first 2s moments of X are the same as the first 2 moments of the Standard Normal Distribution.

1.2 Historical Motivation

This work is inspired by Dr. Zeilberger's paper " A Quick Empirical Reproof of the Asymptotic Normality of the Hirsch Citation Index" ([2]), which successfully used number crunching method for reproving the asymptotic normality of the h-index (a.k.a Durfee square).

The beauty of this method is that, while it's almost impossible to figure out the closed **general** form for the all factorial moments in term of both n and r, Maple is very efficient in getting the explicit the first s terms, no matter why s is. This is because we have constructs the math around using **recurrence and Taylor**, both of which Maple is very fast in calculate a **specific number of terms** that we want.

On this paper, we want to expand on the original program so that we can analyze a more general case of the h-index, and investigate the asymptotic normality of other Durfee rectangle with respect to the uniform distribution on the sample space of integer partition n.

Chapter 2

Methodology and Discussion

2.1 Methodology

We started with the simplest case of Durfee rectangle : the largest rectangle with sides k and k+b for a specific b. In order to analyze its asymptotic normality, we start with its generating function:

The Durfee rectangle contributes a total of k(k+b) to the total size n. The generating function for the choice of an arm with no more than k row is:

$$P_{\leq k}(q) = \frac{1}{(1-q)(1-q^2)...(1-q^k)}$$

The generating function for the choice of a leg with largest part not larger than k+b is:

$$P_{\leq (k+b)}(q) = \frac{1}{(1-q)(1-q^2)...(1-q^{k+b})}$$

Thus the grand generating function is :

$$F(q,t) = \sum_{k=0}^{\infty} \frac{q^{k(k+b)}t^k}{(1-q)(1-q^2)...(1-q^k)(1-q)(1-q^2)...(1-q^{k+b})}$$

If we can rewrite F(q,t) into this form :

$$F(q,t) = \sum_{k=0}^{\infty} C_k(t) q^k$$

From this, if we can calculate all moments for each $C_i(t)$ and prove that for these moments are equal to 0 for k odd and $\frac{k!}{(k/2)!2^{k/2}}$, then with the **refined central limit theorem**, we can proved that F(q,t) is asymptotically normal.

However, since F(q,t) formula is a very complicated, to perform a rigorous proof about the asymptotic normally is potentially very hard. Thus, we take advantage of the computability power of computer to give an empirical analysis on the asymptotic normality of the general h-index:

First step: we only try to compute the first couple thousand terms of $C_k(t)$, say K^2 terms. This is done by consider only first K terms in the sum

$$F_K(q,t) = \sum_{k=0}^{K} \frac{q^{k(k+b)} t^k}{(1-q)(1-q^2)...(1-q^k)(1-q)(1-q^2)...(1-q^{k+b})}$$

and then use Taylor expansion on $F_K(q,t)$ up to q^{K^2} :

$$F'(0,t)q^0 + F''(0,t)q^1 + F'''(0,t)\frac{q^2}{2!} + ... + F^{(k^2+1)}(0,t)\frac{q^{k^2}}{k!}$$

$$\Rightarrow C_i(t) = F^{(i)}(0,t), i \in [0, K^2]$$

This step is done by the SidraG function in the code section.

Second step: Transform the combinatorial generating function $C_i(t)$ into the probability generating function $P_i(t)$ by diving each $C_i(t)$ with $C_i(1)$, under the uniform distribution.

$$P_i(t) = \frac{C_i(t)}{C_i(1)}$$

We have the sum:

$$\frac{C_0(t)}{C_i(1)}q^0 + \frac{C_1(t)}{C_i(1)}q^1 + ... + \frac{C_{K^2}(t)}{C_{K^2}(1)}q^{K^2}$$

This step is done by function PGFsG.

Third step: Let $a_i = E_i(f)$ be the expectation of the random variable f(x) such that

$$C_i(t) = \sum_{x \in X_i} t^{f(x)}$$

then we know :

$$a_i = P_i'(1)$$

In this step we centralized the distribution ($\mu = 0$) by diving each $P_i(t)$ with t^{a_i}:

$$Q_i(t) = \frac{P_i(t)}{t^{a_i}}$$

$\Rightarrow$ we have the sum :

$$\frac{C_0(t)}{C_i(1)t^{a_0}}q^0 + \frac{C_1(t)}{C_i(1)t^{a_1}}q^1 + ... + \frac{C_{K^2}(t)}{C_{K^2}(1)t^{a_{K^2}}}q^{K^2}$$

We get the sequence of moments for each of the $Q_i(t)$:

$$m_k(i) = (t\frac{d}{dt})^k Q_n(t)|_{t=1}$$

Then we standardized these moments:

$$\alpha_k(i) = \frac{m_k(i)}{m_2(i)^{k/2}}$$

This step is done by CGFsG, CGFsTg, AveAndMoms, Alpha functions.

Fourth step: Now we estimate α_k when $i \to \infty$ by using $L = [\alpha_k(1), \alpha_k(2), \alpha_k(3)....]$, which are sequence of real number we already calculate, and try to estimate a function of α_k by setting:

$$\alpha_k = \sum(a[i]/n^i, i = 0..K)$$

and solved for a[i] by creating the K equations:

$$a[0] + \frac{a[1]}{(K+1-1)^1} + \frac{a[2]}{(K+2-1)^2} + ..\frac{a[2]}{(K+j-1)^2} = L[j]$$

Then we compare the sequence of α_k with the sequence of moments of the Standard Normal Distribution.

2.2 Discussion

For this work, one of the biggest mistake that we stumbled on was the generating function for the Durfee rectangle. Originally, we thought for any integer a and b, the generating function for the Durfee regtangle with the side k and a*k + b is:

$$\sum_{k=0}^{\infty} \frac{q^{k(ak+b)}t^k}{(1-q)(1-q^2)...(1-q^k)(1-q)(1-q^2)...(1-q^{ak+b})}$$

which seems pretty good initially. However, when we are trying the asymptotic normality with the case $a \neq 1$, we can see that the moments do not match those of Standard Normal Distribution. A close look at the generating function, we can clearly see that it is in fact not correct. Thus, to find the correct generating function for all a will be the next step of this work.

Besides, we are also starting to take step toward the Multivariate Normal Distribution. As in our code, we have started working on finding moments for the probability generating function in the variable a and y.

All our code and result can be found at math.rutgers.edu/ hbl15/em15

Chapter 3

Code

- SidraG(K,q,t,a,b) : the polynomial of degree K(a*K+b) in q and K in t such that the coefficient of $q^n \times t^k$ is the exact number of partitions of n whose size of the largest Durfee rectangle has the form k(ak+b)

  Caution: Since we have not found the correct general formula for the generating function of all Durfee rectangle, this function for now is **only correct if a = 1**. However, we want to keep it in this form in case we are able to find the correct generating function for all case of a. Since, the rest of the program will work if

```
SidraG:=proc(K,q,t,a,b) local lu, k, j:
option remember:
lu:=add(q^(k*(a*k+b))*t^k/mul(1-q^j,j=1..a*k+b),k=1..K):
lu:=taylor(lu,q=0,K*(a*K+b)+1):
add(coeff(lu,q,j)*q^j,j=0..K*(a*K+b)):
end:
```

- PGFsG(K,t,a,b): the list of polynomials of size K(a*K+b), whose i-th entry is the probability generating function for the random variable "size of Durfee rectangle" defined on integer partitions of i. Try: PGFsG(20,t,1,0);

```
PGFsG:=proc(K,t,a,b) local lu,q,i:
lu:=SidraG(K,q,t,a,b):
[seq(sort(coeff(lu,q,i)/subs(t=1,coeff(lu,q,i))),i=1..K*(a*K+b))]:
end:
```

- CGFsG(K,t,a,b): the list of polynomials of size K^2, whose i-th entry is the combinatorial generating function for the random variable "size of Durfee sequare" defined on integer partitions of i. Try: CGFsG(20,t,1,0);

```
CGFsG:=proc(K,t,a,b) local lu,q,i:
lu:=SidraG(K,q,t,a,b):
[seq(sort(coeff(lu,q,i)),i=1..K*(a*K+b))]:
end:
```

- CGFsTg(K,t,a,b): the list of polynomials of size K, whose i-th entry is the combinatorial generating function for the random variable "size of Durfee rectangle" defined on integer partitions of i*(a*i+b). Try: CGFsTg(20,t,1,0);

```
CGFsTg:=proc(K,t,a,b) local lu,q,i:
lu:=SidraG(K,q,t,a,b):
[seq(sort(coeff(lu,q,i*(a*i+b))),i=1..K)]:
end:
```

- QkN(k,N,q): the truncation up to q^N, of the infinite product $((1-q^(k+1))*(1-q^(k+2)))^2 * \ldots$. Try: QkN(0,100,q);

Note: This is a function borrowed from [4]. It's necessary for CGFsRTg function.

```
QkN:=proc(k,N,q) local gu,i:
if N<0 then
```

```
        RETURN(0):
    fi:
    gu:=taylor(mul((1-q^i)^2,i=k+1..trunc(N/2)+1),q=0,N+1):
    add(coeff(gu,q,i)*q^i,i=0..N):
    end:
```

- RHS(q,t,N): the truncation to order q^N in q of $Sum(q^(k^2) * t^k * ((1 - q^(k+1)) * (1 - q^(k+2) * ...)^2$, try: RHS(q,t,100);

  Note: This is a function borrowed from [4]. It's necessary for CGFsRTg function.

```
    RHS:=proc(q,t,N) local gu,k:
    gu:=0:
    for k from 0 to trunc(sqrt(N))+1 do
            gu:=expand(gu+t^k*q^(k^2)*QkN(k,N-k^2,q)):
    od:
    end:
```

- CGFsRTg(K,t,a,b): the list of polynomials of size K, whose i-th entry is the combinatorial generating function for the random variable "size of Durfee rectangle" defined on integer partitions of i*(i*a+b), using the recurrence approach. Same output as CGFsTg(K,t) but hopefully faster. Try CGFsRTg(20,t,1,0);

```
    CGFsRTg:=proc(K,t,a,b) local q,N,L,R,i,j,T:
    N:=K*(a*K+b):
    L:=QkN(0,2*N+1,q):
    R:=RHS(q,t,2*N+1):
    T[0]:=1:
    for i from 1 to N do
            T[i]:=coeff(R,q,i)-add(coeff(L,q,j)*T[i-j],j=1..i):
```

```
od:
[seq(T[i*(a*i+b)],i=1..K)]:
end:
```

- CGFsTpcG(K,t,a,b): Like CGFsTpc(K,t,a,b) but for K¡=100 using the precomputed values, hence much faster. K must be ¡=100. Try: CGFsTpcG(20,t,1,1);

```
CGFsTpcG:=proc(K,t,a,b) local H50:
if K>50 then
      print(‘K must be <=50, try: CGFsTg(K,t,a,b), but allow a
         lot of time!‘):
fi:
H50:=DS(a,b,t):
[op(1..K,H50)]:
end:
```

- AveAndMoms(f,x,N): Given a probability generating function f(x) (a polynomial, rational function and possibly beyond) returns a list whose first entry is the average (alias expectation, alias mean) followed by the variance, the third moment (about the mean) and so on, until the N-th moment (about the mean). If f(1) is not 1, than it first normalizes it by dividing by f(1) (if it is not 0) .For example, try: $AveAndMoms(((1+x)/2)^{100}, x, 4)$;

  Note: This is a function borrowed from [3]. It’s necessary for MamarG function.

```
AveAndMoms:=proc(f,x,N) local mu,F,memu1,gu,i:
mu:=simplify(subs(x=1,f)):
if mu=0 then
      print(f, ‘is neither a prob. generating function nor can
         it be made so‘):
```

```
        RETURN(FAIL):
fi:
F:=f/mu:
memu1:=simplify(subs(x=1,diff(F,x))):
gu:=[memu1]:
F:=F/x^memu1:
F:=x*diff(F,x):
for i from 2 to N do
        F:=x*diff(F,x):
        gu:=[op(gu),simplify(subs(x=1,F))]:
od:
gu:
end:
```

- Alpha(f,x,N): Given a probability generating function f(x) (a polynomial, rational function and possibly beyond) returns a list, of length N, whose (i) First entry is the average, (ii): Second entry is the variance, for i=3...N, the i-th entry is the so-called alpha-coefficients that is the i-th moment about the mean divided by the variance to the power i/2 (For example, i=3 is the skewness and i=4 is the Kurtosis). If f(1) is not 1, than it first normalizes it by dividing by f(1) (if it is not 0) . For example, try: $Alpha(((1+x)/2)^{100}, x, 4)$;

  Note: This is a function borrowed from [3]. It's necessary for MamarG function.

```
Alpha:=proc(f,x,N) local gu,i:
gu:=AveAndMoms(f,x,N):
if gu=FAIL then
         RETURN(gu):
 fi:
if gu[2]=0 then
```

```
                print(‘The variance is 0‘):
                 RETURN(FAIL):
                 fi:
        [gu[1],gu[2],seq(gu[i]/gu[2]^(i/2),i=3..N)]:
        end:
```

- AsyAnal(L,N,n): given a list of numbers L, of length k, say, whose i-th entry is f(N+i-1) for i from 1 to nops(L), conjectures an asymptotic approximate expression for f(m) in the form $a[0] + a[1]/n + ... + a[k]/n^{(k-1)}$. Try: $AsyAnal([seq(1 + 6/i + 11/i^2, i = 98..100)], 98, n)$;

  Note: This is a function borrowed from [4]. It's necessary for MamarG function.

```
        AsyAnal:=proc(L,N,n) local k,eq,var,X,a,i:
        k:=nops(L)-1:
        X:=add(a[i]/n^i,i=0..k):
        var:={seq(a[i],i=0..k)}:
        eq:={seq(subs(n=N+i-1,X)-L[i],i=1..k+1)}:
        var:=solve(eq,var):
        subs(var,X):
        end:
```

- MamarG(N,K,a,b): inputs a positive integer N and outputs an article EMPIRICALLY proving the asymptotic normality, using standardized moments up to N, and empirical asymptotics to order K. Try: MamarG(10,3,1,1);

```
MamarG:=proc(N,K,a,b) local i,lu,lu1,mu1,t,m,kak,R,vu1,j:
if not type(N,integer) and N<=2 then
 print(N, ‘should be an integer larger than 2‘):
 RETURN(FAIL):
```

```
fi:
if not (type(K,integer) and K<=10 and K>=1) then
 print(K, `must be an integer between 1 and 10`):
 RETURN(FAIL):
fi:
if not ([a,b]=[1,1] or [a,b]=[1,5] or [a,b]=[2,0] or [a,b]=[3,0] or
    [a,b]=[4,0]) then
 print(a,b,`can only be one of the pairs (1,1),
     (1,5),(2,0),(3,0),(4,0)`):
 RETURN(FAIL):
fi:
print(`Empirical Proof of the Asymptotic Normality of the Hirsch
    Citation Index `):
print(``):
print(`By Shalosh B. Ekhad & Ha Luu `):
print(`Consider the random variable: h-index (alias size of Durfee
    rectangle) defined on integer partitions of n, for large n`):
lu:=evalf(CGFsTpcG(50,t,a,b)):
for i from 1 to N do
 R[i]:=[]:
od:
for i from 50-K to 50 do
        lu1:=lu[i]:
        mu1:=evalf(Alpha(lu1,t,N)):
        print(`With `, i^2, ` citations, the average h-index is ` ,
            evalf(mu1[1],10)):
        print(`and when it is divided by the sqrt of `, i^2, ` namely
            by`, i, ` it is `, evalf(mu1[1]/i,10) ):
        print(``):
         print(`The variance is`, evalf(mu1[2],10)):
        print(`and when it is divided by the sqrt of`, i^2, `namely
           by`, i, ` it is `, evalf(mu1[2]/i,10) ):
        print(``):
        print(`The standardized moments, starting with the third are
           :`):
         lprint(evalf([op(3..nops(mu1),mu1)],10)):
        print(``):


        for j from 1 to N do
              R[j]:=[op(R[j]),mu1[j]]:
         od:
od:
print(``):
print(`------------------------------------------------------`):
print(``):
vu1:=R[1]:
vu1:=[seq(vu1[i]/(50-K+i-1),i=1..K+1)]:
print(`The estimated asymptotic expression for the average, divided by
   m=sqrt(n), to order`, K, `using the data from m=`, 50-K, `to `,
   50, `is `):
kak:=evalf(AsyAnal(vu1,50-K,m),50):
print(kak):
print(``):
print(`and in Maple input format: `):
print(``):
lprint(kak):
print(``):
print(`note that the leading term is not far from the theoretical
    limit‘, evalf(sqrt(6)*log(2)/Pi,10) ):
print(‘‘):
print(‘-----------------------------------------------------‘):
vu1:=R[2]:
vu1:=[seq(vu1[i]/(50-K+i-1),i=1..K+1)]:
print(‘The estimated asymptotic expression for the variance, divided
    by m=sqrt(n), to order‘, K, ‘using the data from m=‘, 50-K, ‘to ‘,
    50, ‘is ‘):
kak:=evalf(AsyAnal(vu1,50-K,m),10):
print(kak):
print(‘‘):
print(‘and in Maple input format: ‘):
print(‘‘):
lprint(kak):
print(‘‘):
print(‘-----------------------------------------------------):
for j from 3 to N do
        vu1:=R[j]:
        print(‘The estimated asymptotic expression for the‘ , j , ‘-th
            standardized moment, to order‘, K, ‘using the above data,
            is: ‘):
        kak:=evalf(AsyAnal(vu1,50-K,m),10):
        print(kak):
        print(‘‘):
        print(‘and in Maple input format: ‘):
        print(‘‘):
        lprint(kak):
        print(‘‘):
        if j mod 2=1 then
        print(‘Note that the leading term is close to that of the
           Normal Distribution, namely 0‘):
        else
        print(‘Note that the leading term is close to that of the
           Normal Distribution, namely‘, j!/(j/2)!/2^(j/2)):
        fi:
od:


print(‘‘):
print(‘-----------------------------------------------------‘):
print(‘‘):
print(‘This concludes this empirical, but VERY CONVINCING, proof of
   the asymptotic normality‘):
print(‘Note also that it (empirically) proves Rodney Canfields
   concentration measure since ‘):
print(‘The variance is proportional to the average, and since it is
   normal, it is concentrated around the mean. ‘):
end:
```

- MamarGs(N,K,a,b): short version of MamarG inputs a positive integer N and outputs an article EMPIRICALLY proving the asymptotic normality, using standardized moments up to N, and empirical asymptotic to order K. Try: MamarGs(10,3,1,1);

  Note: This function is using precomputed data. In order to run this function correctly, you will need to download and read HIRSCHgData.txt beforehand.

```
MamarGs:=proc(N,K,a,b) local i,lu,lu1,mu1,t,m,kak,R,vu1,j:
if not type(N,integer) and N<=2 then
 print(N, ‘should be an integer larger than 2‘):
```

```
 RETURN(FAIL):
fi:
if not (type(K,integer) and K<=10 and K>=1) then
 print(K, `must be an integer between 1 and 10`):
 RETURN(FAIL):
fi:
if not ([a,b]=[1,1] or [a,b]=[1,5] or [a,b]=[2,0] or [a,b]=[3,0] or
    [a,b]=[4,0]) then
 print(a,b,`can only be one of the pairs (1,1),
     (1,5),(2,0),(3,0),(4,0)`):
 RETURN(FAIL):
fi:
lu:=evalf(CGFsTpcG(50,t,a,b)):
for i from 1 to N do
 R[i]:=[]:
od:
for i from 50-K to 50 do
        lu1:=lu[i]:
        mu1:=evalf(Alpha(lu1,t,N)):
        print(`With `, i^2, ` citations, the average h-index is ` ,
            evalf(mu1[1],10)):
        print(`and when it is divided by the sqrt of `, i^2, ` namely
            by`, i, ` it is `, evalf(mu1[1]/i,10) ):
        print(``):
        print(`The variance is`, evalf(mu1[2],10)):
        print(`and when it is divided by the sqrt of`, i^2, `namely
            by`, i, ` it is `, evalf(mu1[2]/i,10) ):
        print(``):
        print(`The standardized moments, starting with the third are
            :‘):
        lprint(evalf([op(3..nops(mu1),mu1)],10)):
        print(‘‘):
        for j from 1 to N do
                R[j]:=[op(R[j]),mu1[j]]:
         od:
od:
vu1:=R[1]:
vu1:=[seq(vu1[i]/(50-K+i-1),i=1..K+1)]:
print(‘The estimated asymptotic expression for the average, divided by
    m=sqrt(n), to order‘, K, ‘using the data from m=‘, 50-K, ‘to ‘,
    50, ‘is ‘):
kak:=evalf(AsyAnal(vu1,50-K,m),50):
print(kak):
print(‘‘):
vu1:=R[2]:
vu1:=[seq(vu1[i]/(50-K+i-1),i=1..K+1)]:
print(‘The estimated asymptotic expression for the variance, divided
    by m=sqrt(n), to order‘, K, ‘using the data from m=‘, 50-K, ‘to ‘,
    50, ‘is ‘):
kak:=evalf(AsyAnal(vu1,50-K,m),10):
print(kak):
print(‘‘):
for j from 3 to N do
        vu1:=R[j]:
        print(‘The estimated asymptotic expression for the‘ , j , ‘-th
            standardized moment, to order‘, K, ‘using the above data,
            is: ‘):
        kak:=evalf(AsyAnal(vu1,50-K,m),10):
```

```
        print(kak):
        print(``):
od:
end:
```

- ID(f,x,k): Given the probability generating function f in the variable x, for some r.v. under some prob. distibution outputs the list [av,var, std. moms]

```
        ID:=proc(e,x,k) local i,av,M, f:
        f:=e/subs(x=1,e):
        av:=subs(x=1,diff(f,x)):
        M:=[av]:
        f:=f/x^av:
        f:=x*diff(f,x):
        for i from 2 to k do
                f:=x*diff(f,x):
                M:=[op(M),subs(x=1,f)]:
        od:
        M:
        normal([M[1],M[2],seq(M[i]/M[2]^(i/2),i=3..k)]):
end:
```

- NumLr(n,k): Given a number n, output A(n,k) such that A(n,k) is total number of ways to partition n, with the largest part is k, using the recurrence equation.

```
        NumLr:=proc(n,k) local i:
        option remember;
        if n<0 then
                return 0:
        elif n=k then
               return 1:
        else
               return add(NumLr(n-k,i),i=1..k):
        fi:
        end:
```

- NumLrs(n): give a number n, out put list of numbers such that the i-th entry is the number of way to partition n , with i to be the largest part.

```
NumLrs:=proc(n) local i:
[seq(NumLr(n,i),i=1..n)]:
end:
```

- GFLP(k,t): the first k terms in the sequence of polynomials $P_i(t)$:= Sum of $t^(largestpart(partition))$, over all partitions of the integer i, using the generating function, and taylor

```
GFLP:=proc(k,t) local i,lu,q:
lu:=expand(taylor(mul(1/(1-q^i*t),i=1..k),q=0,k+2)):
[seq(coeff(lu,q,i),i=1...k)]:
end:
```

- GFLPf(k,t): the first k terms in the sequence of polynomials $P_i(t)$:= Sum of $t^(largestpart(partition))$, over all partitions of the integer i, using the recurrence for A(n,k):=number of partitions of n with largest part k

```
GFLPf:=proc(k,t) local i,n:
option remember:
[seq(add(NumLr(n,i)*t^i,i=1..n),n=1..k)]:
end:
```

- ID2(f,x,y,k): Given the probability generating function f in the variable x,y for some r.v. under some prob. distribution outputs the list [av,var, std. moms]

```
ID2:=proc(e,x,y,k) local Mx,CPf,f1, f2, i, M, My, f:
f:=e/subs({x=1,y=1},e):
Mx:=subs({x=1,y=1},diff(f,x)):
My:=subs({x=1,y=1},diff(f,y)):
M:=[[Mx,My]]:
f:=f/(x^Mx*y^My):
f1:=x*diff(f,x):
f2:=y*diff(f,y):
for i from 2 to k do
       f1:=x*diff(f1,x):
       f2:=y*diff(f2,y):
       M:=[op(M),[subs({x=1,y=1},f1),subs({x=1,y=1},f2)]]:
od:
M:
#[M[1],M[2],seq([M[i][1]/M[2][1]^(i/2),M[i][2]/M[2][2]^(i/2)],
    i=3..k)]:
end:
```

Chapter 4

Reference

[1] D. Zeilberger, The Automatic Central Limit Theorems Generator (and Much More!)

[2] A Quick Empirical Reproof of the Asymptotic Normality of the Hirsch Citation Index

[3] http://www.math.rutgers.edu/ zeilberg/tokhniot/HISTABRUT

[4] http://www.math.rutgers.edu/ zeilberg/tokhniot/HIRSCH

[5] D. Zeilberger, HISTABRUT: A Maple Package for Symbol-Crunching in Probability theory,