Statistics of Standard Young Tableaux Using the Greene-Nijenhuis-Wilf Algorithm

Aurora Hiveley and Lucy Martinez

Spring 2026

Abstract

The standard Young tableau is a well-studied combinatorial object with many numerical properties of interest to researchers. However, as the size of the tableau shape increases, the sample space of tableaux increases even more rapidly, making statistical calculations prohibitively expensive both with regard to computational memory and runtime. To circumvent this, we use the Greene-Nijenhuis-Wilf algorithm to generate uniformly random tableaux and bootstrap sampling to produce sample spaces that approximate the whole set. In particular, we calculate row sums, column sums, and the lengths of the "backwards hooks." We compute empirical estimates of those statistics from the sampled tableaux and use bootstrap resampling to estimate standard errors of these estimators.

Introduction

This paper is primarily concerned with a specific combinatorial object: the standard Young tableau. A Young diagram is a grid of left-justified boxes such that each row's length is at most equal to the length of the row above it. Such a diagram may also be referred to as a Ferrers diagram when the boxes are replaced by circular dots. A Young tableau is obtained from a Young diagram by populating the boxes with numbers. A standard Young tableau results from filling the entries of a Young diagram with n total boxes with the numbers $\{1, 2, \ldots, n\}$ such that (1) each entry is unique, (2) the rows are increasing from left to right, and (3) the columns are increasing from top to bottom. An example is shown in Figure 1. We may refer to the underlying Young diagram for a given standard Young tableau as its "shape," typically encoded as a list of weakly decreasing row lengths. We denote the shape of a Young tableau by $L = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ where λ_i is the number of boxes in row i.

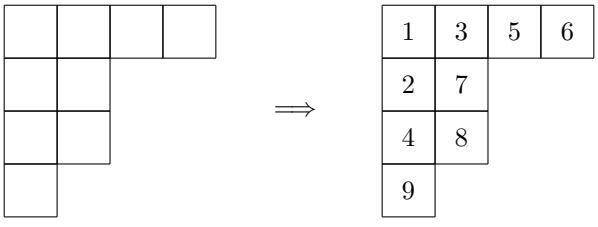

Figure 1: An example Young diagram on $n = 9$ (left) and the same Young diagram filled in to form a standard Young tableau (right) of shape $L = (3, 2, 2, 1)$.

In general, the number of standard Young tableau with n cells is enumerated by A00085 in the OEIS [3]. Note that the number of Young diagrams with n boxes is equivalent to the number of partitions of n, which are enumerated by A000041 in the OEIS [3]. Each partition of n produces a tableau shape by sorting the partition's pieces in weakly decreasing order, and letting the length of row i of the tableau be equal to the size of piece i of the partition. Then to count the total number of standard Young tableau with n entries as in OEIS A00085, we must determine how many standard Young tableau may be obtained from filling in the boxes of a shape L.

Perhaps the best-known method for calculating the number of standard Young tableaux of a shape L is with the hook length formula, originally developed by Frame-Robinson-Thrall in 1954. The hook H_{ij} of a cell (i,j) is the set of cells (a,b) such that either $a=i$ and $b \geq j$ or $a \geq i$ and $b=j$. The hook length h_{ij} is the number of cells in H_{ij}. Then the number of standard Young tableau with shape L is equal to $n!/\prod_{(i,j)\in L} h_{ij}$. For example, in Figure 1, the hook length of cell $(2,1)$ is 4 since the hook includes cells $(2,1),(2,2),(3,1)$ and $(4,1)$.

1 Greene-Nijenhuis-Wilf Algorithm

A much shorter proof of an alternative way of counting the number of standard Young tableau with shape L was later developed by Greene-Nijenhuis-Wilf in 1979 using probabilistic methods [4]. Our work employs the algorithm discussed in [4, Section 2], which proceeds as follows.

Start with a tableau shape L (or equivalently, a Young/Ferrers diagram). Choose a cell (i,j) uniformly at random from the shape L. Note that any individual cell is chosen with probability $1/n$. Next, choose another cell $(i',j') \neq (i,j)$ where $(i',j') \in H_{ij}$. Once again, this cell is chosen uniformly at random with probability $1/(h_{ij}-1)$. This process is repeated until a cell (i^*,j^*) is chosen such that $H_{i^*j^*} = \{(i^*,j^*)\}$. In other words, (i^*,j^*) is a corner cell. This cell becomes the terminal cell of one trial of the algorithm, and we fill it in with entry n. We repeat for the shape $L - \{(i^*,j^*)\}$, and continue recursively until each cell is filled and a standard Young tableau has been produced. In this sense, the algorithm produces an entire standard Young tableau uniformly at random. In fact, the algorithm's recursivity is much faster and less computationally expensive than the naive approach: generating the set of *all* standard Young tableaux of shape L, and selecting one uniformly at random from the full set. This algorithm is implemented in our Maple package `proj3.txt` as the procedure `RandSYT(L)`, which is discussed in further detail in Section 3. For an example of how the algorithm works, see Example 1.

Example 1. In this example, we begin with a Young diagram on $n=9$ with shape $L=(4,2,2,1)$.

The algorithm starts by selecting a random cell with probability $1/9$. Then, it follows the next steps:

1. Suppose that the algorithm selects the cell $(3,1)$. Since $(3,1)$ is not a corner cell, the algorithm must select a cell within the hook of $(3,1)$. The options within the hook of $(3,1)$ are $(3,2)$ and $(4,1)$. Suppose that the algorithm selects cell $(4,1)$. Since cell $(4,1)$ is a corner cell, then the algorithm stops and labels this cell with 9. We show this process in Figure 2.

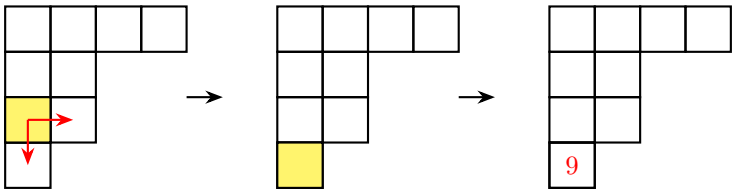

Figure 2: From left to right, each diagram shows the process until the algorithm reaches a corner cell. The yellow cells represent the cells that the algorithm selects at the current step, and the red arrows show the cells within the hook that could be chosen in the next step.

2. Now that the algorithm found a corner cell which is a terminal cell, the process is repeated on a smaller Young diagram. The new shape of the Young diagram is $L = (4, 2, 2)$. The algorithm then selects a random cell with probability $1/8$. Suppose that it selects the cell $(1, 2)$. Since $(1, 2)$ is not a corner cell, the algorithm must select a cell withing the hook $(1, 2)$. For our example, the algorithm selects cell $(2, 2)$ which is not a corner cell, so it must look for a cell within its hook. Since it only has one option, the terminal cell is $(3, 2)$ and this cell is labeled with 8. We show this process in Figure 3.

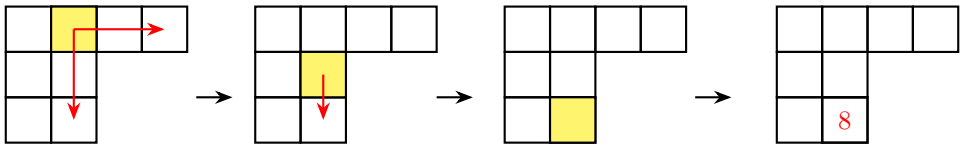

Figure 3: From left to right, each diagram shows the process until the algorithm reaches a corner cell. The yellow cells represent the cells that the algorithm selects at the current step, and the red arrows show the cells within the hook that could be chosen in the next step.

3. For the third step of the recursion, the shape of the Young diagram changes from $L = (4, 2, 2)$ to $L = (4, 2, 1)$. The algorithm then selects a random cell with probability $1/7$. Suppose that it selects the cell $(1, 2)$. Since it is not a corner cell, the algorithm must choose a cell within its hook: the options are $(1, 3), (1, 4)$ or $(2, 2)$. Suppose that cell $(2, 2)$ is selected, and since it is a corner cell, the algorithm stops and labels the cell with 7. We show this process in Figure 4.

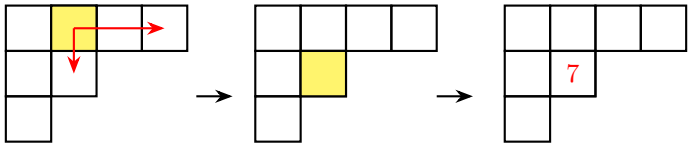

Figure 4: From left to right, each diagram shows the process until the algorithm reaches a corner cell. The yellow cells represent the cells that the algorithm selects at the current step, and the red arrows show the cells within the hook that could be chosen in the next step.

4. In the fourth step of the recursion, the shape of the Young diagram changes from $L = (4, 2, 1)$

to $L = (4, 1, 1)$. The algorithm then selects a random cell with probability $1/6$. Suppose that it selects cell $(1, 2)$, which is not a corner cell. In this case, the algorithm continues looking to the right of cell $(1, 2)$ since those are the only cells within the hook of $(1, 2)$ until it reaches the corner cell $(1, 4)$. The algorithm stops and labels the cell with 6. We show this process in Figure 5.

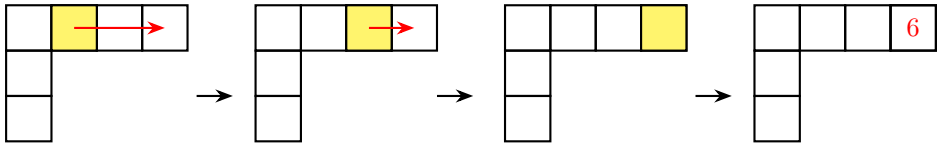

Figure 5: From left to right, each diagram shows the process until the algorithm reaches a corner cell. The yellow cells represent the cells that the algorithm selects at the current step, and the red arrows show the cells within the hook that could be chosen in the next step.

5. So far, the algorithm has labeled 4 cells. We show the current state of the Young tableau in Figure 6

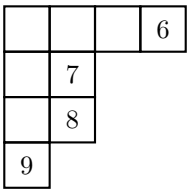

Figure 6: The current state of the Young diagram after 4 steps of the algorithm.

We omit the rest of the steps but Figure 1 shows a possible output of the algorithm which produces a Young tableau uniformly at random.

2 Bootstrapping

We describe the method we use to sample B many standard Young tableaux. The *bootstrapping method* was introduced by Efron in 1979 [2]. The bootstrap method is used for estimating a desired statistic by resampling with replacement. For example, if we are interested in calculating the average height of the population of the world, it is not be feasible to collect all the data. Instead, we can consider selecting N people from the world, recording their height, and then estimating the average height with the following process. Suppose that we record all the heights of N people in a list B, here $N = |B|$. To create *one sample*, we create a different list B_1 which is generated by sampling list B with replacement. It is important that the list B_1 is of length N. Of course, the list B_1 may contain repeated values from the list B because we sample with replacement from B. Once we create the first sample list B_1, we repeat and create lists $B_2, B_3, \ldots, B_N$ in the same manner, each of length N. Each list is called a *bootstrap*. For each bootstrap, we calculate the average height and get an estimated distribution of the average heights.

Going back to standard Young tableaux, it is computationally slow to generate many standard Young tableaux as n goes large with the Greene-Nijenhuis-Wilf algorithm. This is partially due to the fact that the algorithm undergoes n iterations and in each iteration, the algorithm searches for a corner cell by selecting the next cell within the hook of the current cell. So, if there are a large number of rows and columns, then the algorithm starts somewhere and repeats the process until it reaches a single corner cell. Instead, we use the bootstrapping method to generate many tableaux. We formally define the bootstrapping method and describe the process that we use to implement it in Maple.

Definition 2.1 (Bootstrapping Method [2, 5])**.** Let $X = \{x_1, x_2, \ldots, x_n\}$ be an independent and identically distributed sample from an unknown distribution F. Then the bootstrap procedure performs the following steps:

1. Generate B bootstrap samples $X^{*1}, X^{*2}, \ldots, X^{*B}$.
2. Each bootstrap sample X^{*b} is obtained by sampling n observations with replacement from the original data set X.
3. For each bootstrap sample X^{*b} compute a desired statistic and record it in a new list Y.
4. Use the distribution of the computed statistic across bootstrap samples to estimate the standard error of the original statistic computed from X.

In Maple, we first generate a sample list L of B many standard Young tableaux using the Greene-Nijenhuis-Wilf algorithm. This sample serves as an empirical approximation of the uniform distribution on Young tableaux of shape $m \times n$. We then apply bootstrap resampling to this empirical sample to estimate standard errors of the computed statistics. In Section 3, we provide more details of the implementation in Maple and in Section 4, we provide some observations about some statistics of interest.

3 Maple Procedures

Accompanying this paper is the Maple package `SYTSamplingProject.txt`. We describe the main procedures needed to generate standard Young tableau using the Greene-Nijenhuis-Wilf algorithm and to calculate statistics using bootstrapping methods.

`RandSYT(L)` inputs a shape `L` of a Standard Young Tableau, encoded as a list of weakly decreasing positive integers corresponding to the row lengths, and outputs a standard Young tableau of shape `L` produced uniformly at random. This is accomplished using the Greene-Nijenhuis-Wilf algorithm discussed in Section 1.

To simulate the bootstrapping process, we also have the procedure `GenerateManySYT(L,K)`. For an input tableau shape `L` and a positive integer `K`, this procedure outputs a set of `K` standard Young tableau of shape `L` produced with the Greene-Nijenhuis-Wilf algorithm, i.e., using `RandSYT(L)`. Then `GenerateManySYT(L)` effectively gives us a bootstrap sample of size `K` since each tableau in the sample is produced uniformly at random, thereby simulating sampling with replacement from the set of all tableau of shape `L`.

Once we have a sample set of standard Young tableau to pull from, we must develop statistical tools to generate bootstrapped samples, calculate numerical properties across those samples, and analyze the results. Procedures used to perform statistical analysis include the following:

- `BSsamp(L):` inputs a list `L` and outputs a bootstrap sample of size `n := nops(L)` from the list using the bootstrapping method described in Section 2.

- `BSse(L,B)`: inputs a list of numbers `L` and a positive integer `B` indicating the number of bootstrap samples to construct. Then outputs the standard error across all bootstrapped samples.
- `BSstat(L,B,c,property)`: Inputs a list of standard Young tableau `L`, a positive integer `B`, a cell location `c` in the tableau, and a property from the list of property calculating procedures (see below). The procedure then outputs the average of the desired property across `B` total bootstrap samples.

  This parent procedure has also been separated into two different child procedures with the same general outline, but which take a slightly different approach to generating bootstrap samples:

  - `BSstat1(L,B,c,property)`: First generates a sample space of standard Young tableau by using `GenerateManySYT(L,B)`. Then generates each bootstrap sample using `BSsamp` on the list obtained in this way.
  - `BSstat2(L,B,c,property)`: Uses `RandSYT(L)` a total of B^2 times to generate a bootstrap sample from the entire set of standard Young tableau of shape `L`.

The goal is to approximate statistics over the space of all standard Young tableaux of a fixed shape using sampling via the Greene-Nijenhuis-Wilf algorithm. Since exact enumeration is not feasible for large shapes, we instead compute statistics on a finite random sample of Young tableaux and use bootstrap resampling to estimate the standard errors of these estimates. Some of the procedures used to calculate these properties are:

- `rowSum(T,c)`: Inputs a standard Young tableau `T` and a cell `c := [i,j]` in the shape `T`. Outputs the sum of all entries in the row containing the cell `c`, i.e., the row indexed by `i := c[1]`.
- `colSum(T,c)`: Analogous to `rowSum(T,c)`, but instead of calculating the sum of all entries in row `i := c[1]`, this procedure calculates the sum of all entries in column `j := c[2]`.
- `backHook(T,c)`: Inputs a standard Young tableau `T` and a cell `c := [i,j]` in the shape `T`. Outputs the number of cells in column `j-1` and in row `i+1` or lower which contain entries smaller than the entry in position `c`. Note that this seemingly random statistic is of interest to the current research question of one of the authors.
- `SumBackHook(T)`: Inputs a standard Young tableau `T` and outputs the sum of `backHook(T,c)` over all cells `c` in the tableau `T`.

4 Data

Recall from Section 3 that we implemented two different procedures to calculate statistics across bootstrap samples: `BSstat1` and `BSstat2`. In general, both methods produced similar results, which is not unexpected since both procedures implement a version of bootstrapping, just with slight discrepancies in how many times a list is sampled and which list is being sampled. However, the big difference in the two procedures was that `BSstat1` has a much faster runtime, especially for larger standard Young tableaux. This also is not surprising, since `BSstat1` generates a single list of size `B` and resamples from that list, while `BSstat2` generates `B` lists of length `B`, meaning that the Greene-Nijenhuis-Wilf algorithm is executed a total of B^2 times for `BSstat2` and only `B` times

for `BSstat1`. For this reason, we deferred to `BSstat1` when tabulating our results in the following sections.

The data summarized in Tables 1-4 was generated using `BSstat1` with a bootstrap sample size of `B = 100`.

4.1 Sum of the first row of Young tableaux of shape $m \times n$

We provide computational evidence for the lower bound for the sum in the first row for a Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$. We record the data of the row sums of a Young tableau of shape $m \times n$ in Table 1.

The sequence A202253 in the OEIS starts with the following terms,

$$3, 9, 17, 27, 41, 57, 75, 97, 121, 147, 177, \ldots$$

and it has the following conjectured formula added in 2012,

$$1 + 2\Big\lfloor \frac{2n^2 + 2n}{3} \Big\rfloor, \qquad n \geq 1. \tag{1}$$

The description for the entry A202253 is "number of zero-sum $-n..n$ arrays of 3 elements with adjacent element differences also in $-n..n$." For example, if $n = 10$, then the entries in the triple (a, b, c) must lie in $\{-10, -9, \ldots, 0, 1, \ldots, 9, 10\}$ such that $a + b + c = 0$. Some solutions for $n = 10$ include $(7, 0, -7), (9, 0, -9), (-3, 4, -1)$. According to the 2012 entry, there are 147 solutions for $n = 10$ (conjecturally).

We observe that Equation 1 is a lower bound for the elements in the rows of Table 1 starting with row 4.

Definition 4.1. Let $m, n \in \mathbb{N}$, $S(m, n, r)$ be the sum of row r for the Young tableau of shape $m \times n$.

In our computations, we observe the following about the sum of the first row based on Table 1.

Observation 4.1. If $m \geq 4$, the sum of the first row of the Young tableau of shape $\underbrace{(n, n, \ldots, n)}_{m}$ is bounded below by

$$1 + 2\Big\lfloor \frac{2n^2 - 2n}{3} \Big\rfloor.$$

In other words,

$$1 + 2\Big\lfloor \frac{2n^2 - 2n}{3} \Big\rfloor \leq S(m, n, 1), \quad m \geq 4,\ n \geq 2.$$

m \ n	1	2	3	4	5	6
1	1	3	6	10	15	21
2	1	3.4913	7.3851	13.276	21.329	30.4223
3	1	3.6723	8.9683	15.7326	25.3393	36.507
4	1	3.9782	9.3715	17.6393	27.9691	41.8674
5	1	4.1726	9.6673	18.4498	29.8752	45.2447
6	1	4.2185	10.4007	19.0646	32.1948	47.8858
7	1	4.4881	10.7745	20.7786	32.7309	51.0065
8	1	4.3404	10.8227	21.2033	35.1034	51.6954
9	1	4.5876	10.9885	22.0231	36.2742	52.7278
10	1	4.3833	11.0828	22.1365	35.3323	55.5496

Table 1: Sum across row #1 for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

In what follows, we compare how close the lower bound is to the data given by Table 1. We analyze row 4 below. In Maple, we set `Digits:=15` so that we can compare the values up to 15 digits. We compute the ratio of the lower bound and the numbers in Table 1. In Maple syntax, we run the following line:

```
[seq(BSstat1([n$4],100,[1,1],rowSum)/(1+2*floor((2*(n-1)^2+2*(n-1))/3)),n=1..5)];
```

which outputs:

```
[1.00000000000000, 1.34750000000000, 1.03442222222222, 0.997247058823529,
                                1.00794444444444].
```

Since $1 \leq n \leq 5$ is a low range for n, we run the following in Maple:

```
[seq(BSstat1([n$4],100,[1,1],rowSum)/(1+2*floor((2*(n-1)^2+2*(n-1))/3)),n=90..100)];
```

which outputs:

```
[1.257123893, 1.248257888, 1.248134605, 1.249933666, 1.250907197, 1.250308844,
     1.261738327, 1.254281155, 1.260203700, 1.261013427, 1.261313105] .
```

Observe that as n grows, there is a slight increase in the ratio. However, this ratio stays just above 1, providing some evidence that Observation 4.1 is worth exploring.

4.2 Sum of the second row of Young tableaux of shape $m \times n$

We analyze the sum of the second row in an $m \times n$ standard Young tableau with $m \geq 2$. Whenever $n = 1$, the row sum equals to 2 because there is only one column and the entry in the cell $(2, 1)$ must be equal to 2. We record the date in Table 2.

m \ n	1	2	3	4	5	6
2	2	6.6225	13.368	22.5438	34.4313	48.6547
3	2	6.9453	14.6747	26.0495	39.7052	57.4629
4	2	7.4195	16.5274	28.8628	44.8041	64.4912
5	2	7.7569	17.1036	29.9127	48.4158	69.5084
6	2	7.6991	17.5705	31.5533	51.3487	74.6861
7	2	7.9652	18.237	33.9202	53.9115	79.7636
8	2	8.3673	18.6328	35.505	54.4733	81.6526
9	2	8.2149	19.6675	35.5543	56.4635	86.2543
10	2	8.365	19.8075	36.1206	60.37	88.0244

Table 2: Sum across row #2 for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

We propose a lower bound for the sum of the second row of a standard Young tableaux of shape $L = (n, n)$. In particular, we estimate the quantity $S(2, n, 2)$ based on the first row of Table 2, where $S(2, n, 2)$ is described in Definition 4.1. We begin by defining *Beatty sequences*. In 1926, Samuel Beatty posed the following problem in the American Mathematical Monthly [1]:

If X is a positive irrational number and Y its reciprocal, prove that the sequences

$$\begin{array}{llll} (1+X), & 2(1+X) & 3(1+X), & \ldots \\ (1+Y), & 2(1+Y), & 3(1+Y), & \ldots \end{array}$$

contain one and only one number between each pair of consecutive positive integers.

Let $r \geq 1$ be an irrational number then $\mathcal{B} = \{\lfloor r \rfloor, \lfloor 2r \rfloor, \lfloor 3r \rfloor, \ldots\}$ is called a Beatty sequence. Let $r = \frac{3+\sqrt{3}}{2}$, then the Beatty sequence (OEIS A054406) for such r is the following:

$$2, 4, 7, 9, 11, 14, 16, 18, 21, 23, 26, 28, \ldots$$

If we define the n-th partial sum of the Beatty sequence for $r = \frac{3+\sqrt{3}}{2}$, we obtain the following sequence (OEIS A194143):

$$2, 6, 13, 22, 33, 47, 63, 81, 102, 125, 151, \ldots$$

We propose the following observation regarding the first row of Table 2. Recall that from Definition 4.1, $S(m, n, 2)$ denotes the sum of the second row for the standard Young tableau of shape $m \times n$.

Observation 4.2. Let

$$S(n) = \sum_{j=1}^{n} \left\lfloor j \left(\frac{3+\sqrt{3}}{2} \right) \right\rfloor,$$

which is the n-th partial sum of the Beatty sequence for $r = \frac{3+\sqrt{3}}{2}$. Let $S(2, n, 2)$ be the sum of the second row of the Young tableau of shape $L = (n, n)$ and $C(n)$ be a constant dependent on n. For $n \geq 1$,

$$C(n)S(n) \sim S(2, n, 2),$$

where $\sim$ denotes that

$$\lim_{n \to \infty} \frac{C(n)S(n)}{S(2, n, 2)} = 1.$$

As in Subsection 4.1, we compare how close the approximation in Observation 4.2 is to the data given by Table 2. We compute the ratio without the constant $C(n)$. In Maple syntax, we run the following line,

```
[seq(BSstat1([n$2],100,[2,1],rowSum)/(add(floor(j*(3+sqrt(3))/2),j=1..n)),n=1..5)];
```

which outputs,

```
[1.000000000, 1.079566667, 1.035253846, 1.037431818, 1.039215152] .
```

Again, we run the following in Maple for $90 \leq n \leq 100$:

```
[seq(BSstat1([n$2],100,[2,1],rowSum)/(add(floor(j*(3+sqrt(3))/2),j=1..n)),n=90..100)];
```

which outputs:

```
[0.9168057134, 0.9185373162, 0.9149356193, 0.9154768162, 0.9150371268,
0.9133809905, 0.9115493573, 0.9135424272, 0.9133154432, 0.9137544543,
                          0.9116848979] .
```

Although the previous data is not close to 1, it suggests that the ratio is off by a constant dependent on n, providing some evidence that Observation 4.2 is worth exploring.

4.3 Sum of the first column of Young tableaux of shape $m \times n$

We provide computational evidence for the average sum across row number 2 for an $m \times n$ standard Young tableau. In particular, we give an approximation for column 6 in Table 3.

m \ n	1	2	3	4	5	6
1	1	1	1	1	1	1
2	3	3.5054	3.8144	4.105	4.2049	4.2086
3	6	7.7069	8.8114	9.6567	10.1366	10.7008
4	10	13.2504	15.5947	17.2443	18.7599	19.2245
5	15	20.7322	25.1825	26.3798	29.9172	32.3607
6	21	29.6902	36.6342	41.5711	45.3525	48.0283
7	28	40.6406	49.8694	59.4332	62.1134	69.5031
8	36	54.0671	66.1623	75.3866	83.8613	90.07
9	45	67.8703	85.2771	96.9501	108.1806	117.7492
10	55	84.0506	107.6045	122.2876	139.8708	150.07

Table 3: Sum across column #1 for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

The sequence A034966 in the OEIS starts with the following terms,

$$1, 4, 10, 19, 32, 45, 67, 88, 116, 145, 179, \ldots$$

which does not currently have a formula. Its description is "number of distinct lengths of main diagonals of all $i \times j \times k$ boxes with edge lengths i, j, k in $[0, n]$." We propose the following observation regarding the sixth column of Table 3.

Observation 4.3. Let $A(m)$ be the number of distinct lengths of main diagonals of all $i \times j \times k$ boxes with edge lengths i, j, k in $[0, m]$. Let $B(m, 6, 1)$ be the sum of the first column of Young tableaux of shape $\underbrace{(6, 6, \ldots, 6)}_{m}$ and $C(m)$ be a constant dependent on m. For $m \geq 1$,

$$A(m-1) \sim C(m)B(m, 6, 1),$$

where $\sim$ denotes that

$$\lim_{n\to\infty} \frac{A(m-1)}{C(m)B(m, 6, 1)} = 1.$$

We emphasize that the sequence indexing in the OEIS starts at 0, so we shift the indexing in the statement of Observation 4.3.

Next, we compare how close the approximation in Observation 4.3 is to the data given by column 4 of Table 3. We compute the ratio without the constant $C(m)$. Since that the sequence A034966 does not have a formula, we save the first 150 values of the sequence in a list L. We compute the ratio by running the following line in Maple:

```
[seq(BSstat1([6$m],100,[1,1],colSum)/L[m],m=1..5)];
```

which outputs:

```
[1.000000000, 0.9880750000, 1.027160000, 0.9863894737, 1.014984375].
```

Again, we run the following in Maple for $90 \leq m \leq 100$:

```
[seq(BSstat1([6$m],100,[1,1],colSum)/L[m],m=90..100)];
```

which outputs:

```
[1.172527086, 1.158309047, 1.157598273, 1.167745399, 1.154884269, 1.163689427,
      1.164306386, 1.152138741, 1.164729986, 1.168086669, 1.159826102].
```

Although the previous data is not close to 1, it provides some computational evidence that the ratio is off by a constant dependent on m.

4.4 Backwards Hooks

In Section 3, we defined the "backwards hook" of a cell `c = [i,j]` in a tableau `T` to be the number of cells `[i',j']` such that `i := i-1` and `j < j' <= n` and `T[i,j] > T[i',j']`. To calculate the length of the back hook across a bootstrap sample, we must indicate a specific cell from the tableau shape from which to draw the backwards hook. To simplify our calculations, and to draw conclusions about how the back hook lengths are distributed across the tableau's shape, we instead opt to calculate the sum of all back hook lengths using `SumBackHook(T)`. The results of these computations are tabulated in Table 4.

m \ n	1	2	3	4	5	6
1	0	1	2	3	4	5
2	0	1.5258	2.9593	4.118	5.0659	6.3595
3	0	1.9268	3.1693	4.448	5.9111	7.1446
4	0	2.1643	3.4428	4.9423	6.2242	7.2443
5	0	2.1397	3.9404	4.9676	6.1988	7.4812
6	0	2.3325	4.0944	5.1536	6.6112	7.8091
7	0	2.2707	4.0357	5.4661	6.5983	7.9267
8	0	2.4478	3.9085	5.5521	6.822	8.0639
9	0	2.4913	3.9021	5.7733	6.7289	7.926
10	0	2.4139	4.6701	5.6252	6.8248	8.4535

Table 4: Sum of back hook lengths for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

Table 4, in contrast to the tables we've seen thus far, shows very little variation across the rows. Unsurprisingly, the sum of all backwards hook lengths for a tableau with dimensions $1 \times n$ is $n - 1$, as each of the entries in positions $[1, i]$ for $i \geq 2$ has a backwards hook of length 1 (simply the entry in position $[1, i - 1]$). Furthermore, adding an additional row to an existing tableau may increase the back hook lengths for the above entries by 1, but this only happens if smaller entries are placed in the bottom rows before the upper ones have filled. Empirically, it seems that this does not have a significant impact on the sum of back hook lengths since the sequences down a column are relatively stagnant.

In fact, the sequences are not even increasing for all cases! There are some entries in the table, for instance the case of a 3×8 tableau, which are smaller than the entries above, indicating that the back hook sum paradoxically decreases as rows are added to the table. This is one consequence of bootstrapping: particularly when the variation between list elements is very small, a bootstrap sample which includes sampled tableau that *do* have great variation may sway the average computer statistic in an unexpected direction.

In general, the results summarized in Table 4 indicate that the overall backwards hook length is most affected by the number of columns n, and increases very little as m increases for a fixed n.

4.5 Standard Error

Note, of course, that the calculations performed in Section 4 thus far have depended upon the bootstrap sample generated by the Maple procedures. Then a different bootstrapped sample may produce different statistics. The purpose of bootstrapping is to generate a sample which is representative of the whole by sampling uniformly at random and with replacement, but who is to say that our random sample may have randomly been swayed?

For example, consider a random sample of 100 standard Young tableau all with shape `L := [3,3,3]`. What are the odds that each of these tableaux happens to have a first row with entries `[1,2,3]`? Obviously the entry 1 has only one possible location, and then the entry 2 can be placed in either position `[1,2]` or position `[2,1]`. Then the entry 2 is placed in the top row $\frac{1}{2}$ of the time. After 2 is placed in position `[1,2]`, the entry 3 can be placed either in `[1,3]` or `[2,1]`, so once again the entry 3 is placed in the top row $\frac{1}{2}$ of the time. Then the entries `[1,2,3]` make up the top row only $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ of the time. If a random sample of 100 tableaux all have this property, that would be pretty statistically unusual! And depending on what tableau statistic we are interested in

calculating, this characteristic may be important. Note, for example, that the sum of the first row entries of a 3×3 tableau is at least $1 + 2 + 3 = 6$ and at most $1 + 4 + 7 = 12$. So, if all tableaux in our sample have a first row with a sum of 6, the row sum across this bootstrapped sample is lower than we expect the average row sum to be in a theoretical sense.

Thus, we round out our statistical analysis by considering how stable our data is. In other words, if we generated a different bootstrap sample and calculated the same statistics, how close can we expect our new values to be from the old ones? We quantify this stability by calculating the *standard error*, which is calculated for a list L with average μ using the following formula:

$$SE(L) = \frac{1}{|L|} \sum_{\ell \in L} \sqrt{\frac{\ell - \mu}{|L| - 1}}$$

The calculated standard errors for each of the statistics, shapes, and bootstrap sizes calculated in Sections 4.1-4.4 are tabulated in Tables 5, 6, and 7 below.

m \ n	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0.05228129047	0.1035532362	0.1446669459	, 0.1921035863	0.2800231592
3	0	0.06930907210	0.1475655650	0.2503169708	0.4025850811	0.4903119910
4	0	0.09170544634	0.1788586429	0.3178898069	0.4987162814	0.5568919523
5	0	0.09805193429	0.2385439629	0.3042422447	0.5096014981	0.6977927973
6	0	0.1183371318	0.2286516251	0.3714141552	0.5954607417	0.9143467669
7	0	0.1502750004	0.3283271804	0.5041761357	0.7061204579	0.8527275016
8	0	0.1171929256	0.2937183940	0.4947428165	0.8043329880	1.003847467
9	0	0.1296102939	0.2992876897	0.5448832235	0.8196538811	1.067087777
10	0	0.1262696729	0.3007474023	0.5507662247	0.7771487059	1.119383285

Table 5: Standard errors of sum across row #1 for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

m \ n	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0.05136677384	0.08261044031	0.09090182294	0.09714801786	0.1073592572
3	0	0.1023605728	0.1575091468	0.2108712087	0.2322275965	0.2421290039
4	0	0.1473611656	0.2377691776	0.3040923571	0.4127696548	0.3660592213
5	0	0.2127775905	0.3132769130	0.3860552115	0.6169223958	0.5768205698
6	0	0.3543652741	0.3691362975	0.5862391348	0.7678892689	0.7931461200
7	0	0.3776383151	0.5448760400	0.9193854777	1.031287319	1.016005747
8	0	0.5590908566	0.7254721556	0.9609186419	1.133957310	1.342367918
9	0	0.5015320367	0.8214577090	1.204396303	1.478614459	1.689638212
10	0	0.6314416646	0.9900234667	1.324110740	1.802123984	2.093111583

Table 6: Standard errors of sum across column #1 for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

m \ n	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0.04653704943	0.08563276065	0.08518215776	0.09479478100	0.1255499418
3	0	0.07480202827	0.1101567295	0.1141584934	0.1416318674	0.1273339283
4	0	0.08903840960	0.1195363095	0.1373072570	0.1556407090	0.1572522583
5	0	0.1005759173	0.1525763594	0.2014491940	0.1665332799	0.1879897913
6	0	0.1102912123	0.1359433557	0.1480743398	0.1880241764	0.1814020592
7	0	0.1110848252	0.1495219318	0.1700111405	0.1718517978	0.1978648401
8	0	0.1373752510	0.1553840144	0.1621791004	0.1981291029	0.2061948671
9	0	0.1663322706	0.1525895994	0.2032883954	0.1877117928	0.1968036503
10	0	0.1196767954	0.1711872977	0.1831077400	0.2529104125	0.2516451727

Table 7: Standard errors for the sum of back hook lengths for an $m \times n$ Young tableau of shape $L = \underbrace{(n, n, \ldots, n)}_{m}$.

In Table 5, the standard errors are all small relative to the mean, with the largest errors barely exceeding a value of 1. Compared to the original values in Table 1, we see that the row sums do not change by much across different bootstrap samples even if the error is "maximal," so we expect our calculated statistics to be somewhat stable. The same is true for Table 6, although the largest error in the table approaches 2. Once again, when compared to the values in Table 3, we do not see a significant change in the column sums even if the error is maximal, so our bootstrap samples produce relatively consistent results.

Comparing Tables 5 and 6, we see that the standard error across a row of each table increases at around the same rate for each statistic. This suggests that adding columns to the table impacts each of the row sum and column sum at about the same rate and with about the same deviation. By contrast, the standard errors increase down the columns of Table 6 much faster than they do for Table 5. This indicates that the column sums are more volatile to the addition of tableau rows than the row sums are. Notice, of course, that if we add row m to an existing $(m-1) \times n$ tableau, then the entry in position `[m,1]` can be as small as m and as big as $n*m-n$. Meanwhile, adding column n to an existing $m \times (n-1)$ tableau produces a tableau whose entry in position `[1,n]` is as small as n and as big as $n*m-m$. If n and m are close together, than these ranges of eligible values are close together, so the biggest indicator of how the average row or column sum changes is in the distribution of these values across all tableaux with shape $m \times n$. This distribution, in turn, affects the standard error, so it appears as though the distribution is wider spread for the addition of rows than for the addition of columns. However, the column sums themselves in Table 3 increase much faster than the row sums in Table 1, so this likely also contributes to the discrepancy in standard errors between Tables 5 and 6. It is also important to note that the error bars for both the row and column sums both range from about 1%-2% of the averages, so they both seem to scale consistently even though the sizes of the errors increase with n and m.

In Table 7, unlike the previous two tables, the standard errors are all small relative to the mean, and while they do generally increase with the size of the tableau, even the largest tableau has a standard error ≈ 0.25. This is quite small, which tells us that these calculations are quite stable. This leads us to the conclusion that the sum of back hook lengths is relatively constant across all tableaux of shape $m \times n$, so the shape itself suggests to be the biggest indicator of what numerical value this summation obtains. Comparing these errors to the values in 4, we note that the errors are consistently about $5\% - 7\%$ of the size of the average value computed, so the errors seem to scale

consistently with the averages, further supporting our prior conclusions. Lastly, we note that the size of the error bars compared to the actual averages is higher for the backwards hook sum than for the row and column sum, although this may be because the averages themselves are much smaller than the row/column sums.

References

[1] S. Beatty. *Problem 3173*. The American Mathematical Monthly **33** (1926), no. 3, 159.

[2] B. Efron. *Bootstrap Methods: Another Look at the Jackknife*. The Annals of Statistics **7** (1979), no. 1, 1-26.

[3] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2023. Published electronically at `http://oeis.org`.

[4] C. Greene, A. Nijenhuis, and H. S. Wilf. *A Probabilistic Proof of a Formula for the Number of Young Tableaux of a Given Shape*. Advances in Mathematics **31** (1979), 104-109.

[5] R. Richardson. *An Introduction to the Bootstrap Method*. Published electronically at `https://richardson.byu.edu/624/myl10/bootstrapping.pdf`.