The binomexp package*

David Roderick
angel_ov_north at tiscali dot co dot uk

January 10, 2007

Abstract

Calculates and prints successive lines of Pascal's triangle..

$$\begin{array}{|c||c|c|c|c|c|}
(f+s)^4 & 1f^4s^0 & 4f^3s^1 & 6f^2s^2 & 4f^1s^3 & 1f^0s^4 \\
(f+s)^5 & 1f^5s^0 & 5f^4s^1 & 10f^3s^2 & 10f^2s^3 & 5f^1s^4 & 1f^0s^5
\end{array}$$

and also will typset the following proof

$7! = 7{\cdot}6{\cdot}5{\cdot}4{\cdot}3{\cdot}2{\cdot}1$ $\qquad \binom{n}{r} = \frac{n!}{(n-r)!\cdot r!} = \frac{n!}{(n-r)!\cdot(n-(n-r))!} = \binom{n}{n-r}$

$$\begin{aligned}
\binom{n-1}{r-1} + \binom{n-1}{r} &= \frac{(n-1)!}{(r-1)!\cdot[(n-1)-(r-1)]!} + \frac{(n-1)!}{r!\cdot[(n-1)-r)]!} \\
&= (n-1)!\cdot\left(\frac{1}{(r-1)!\cdot(n-r)!} + \frac{1}{r!\cdot[(n-r)-1)]!}\right) \\
&= (n-1)!\cdot\frac{r+(n-r)}{r!(n-r)!} \\
&= \frac{n!}{r!(n-r)!} = \binom{n}{r}
\end{aligned}$$

$\frac{r}{r!\cdot(n-r)!} = \frac{1}{(r-1)!\cdot(n-r)!}$ because $\frac{6}{6!\cdot(n-r)!} = \frac{1}{5!\cdot(n-r)!}$

$$\begin{aligned}
(r+1)\cdot\binom{n+1}{r+1} &= (r+1)\cdot\frac{(n+1)!}{((r+1)!\cdot((n+1)-(r+1))!} \\
&= (r+1)\cdot\frac{(n+1)!}{(r+1)!\cdot(n-r)!} \\
&= (n+1)\cdot\frac{n!}{r!\cdot(n-r)!} = (n+1)\cdot\binom{n}{r}
\end{aligned}$$

1 Introduction

A very simple package with simple usage. Putting 'binomexp' (which is also typed exactly the same way than `{`⟨*binomexp*⟩`}` inside of the argumentative input of the the `\usepackage` commands enables the user to do two extra things.

*This document corresponds to binomexp v1.0, dated 2007/01/07.

- print any successive rows of Pascal's triangle which will fit on the page up until the power as 31, at which point LaTeX runs out of brain power.

- Use a piece of code which Morten Høgholm wrote which allows the cells inside of an array or a tabular to be repeated in a similar way than those may be repeated inside of the initial description of said array or tabular.

2 Usage

Binomexp ought to load ifthen and calc by itself. If you have already loaded these packages using `\usepackage{calc,ifthen}` unload these therefore. You must then use the command as `\makeatletter` so to get the command names with the symbol as @ inside of those to function.

`\binomexp@putpascal` `\binomexp@putpascal` {⟨*number as lower power*⟩} {⟨*number as higher power*⟩} {⟨*symbol as first variable*⟩} {⟨*symbol as second variable*⟩} {⟨*symbol again as first variable*⟩} {⟨*symbol again as second variable*⟩}

`\binomexp@putpascal{7}{9}{f}{x}{f}{x}` will typset the rows as 7, 8, and 9 of Pascal's triangle. The first column will have $(f + x)^{power}$. The reason why you have to input the symbol again is because the user might like to use a `\cdot` or whatever in the other columns except the first column. And that's it really.

`\binomexp@proof` `\binomexp@proof` {⟨*number as row variable*⟩} {⟨*number as column variable*⟩} will typeset the mathematical proof of Pascal's triangle, which is based upon the observation that the co-efficient is equal with the number of possible combinations of the column variable out of the row variable.

3 How I wrote it.

```
\RequirePackage{calc,ifthen}
```

Morten Høgholm wrote the following code.

```
\newcommand\binomexp@replicate[2]{%
  \ifnum#1>\z@ \expandafter\@firstofone
  \else
    \expandafter\@gobble
  \fi
  {#2\expandafter\binomexp@replicate\expandafter{\number\numexpr#1-1\relax}{#2}}%
}
```

Morten's code allows the following.

```
\begin{document}
\makeatletter
\begin{tabular}{|*{6}{|c|}|}
something1 \binomexp@replicate{4}{& something2}Blah&stuff\\
something1 \binomexp@replicate{4}{& something2}Blah&stuff\\
Third row with line atop from second to fifth column:
\cline{2-5}something1 \binomexp@replicate{4}{& something2}Blah&stuff\\
\end{tabular}
\end{document}
```

You can invoke Morten's code either by loading the `\usepackage{binomexp}` within the preamble, and then by putting `\makeatletter`, or by including the following code somewhere (perhaps a preamble).

```
\makeatletter
\newcommand\binomexp@replicate[2]{%
  \ifnum#1>\z@ \expandafter\@firstofone
  \else
    \expandafter\@gobble
  \fi
  {#2\expandafter\binomexp@replicate\expandafter{\number\numexpr#1-1\relax}{#2}}%
}
\makeatother
```

`\binomexp@call` the `\newcommand` as `\binomexp@call` makes things nice and pretty within a cell

```
\newcommand{\binomexp@call}[1]{\rule[-0.125cm]{0mm}{0.5cm}\mbox{$#1$}}
```

`\binomexp@up` the `\newcommand` as `\binomexp@up` is by the power of the series which ascends

```
\newcounter{binomexp@up}
\newcommand{\binomexp@up}{\number\value{binomexp@up}
\addtocounter{binomexp@up}{1}}
```

`\binomexp@down` the `\newcommand` as `\binomexp@down` is by the power of the series which descends

```
\newcounter{binomexp@down}
\newcommand{\binomexp@down}{\number\value{binomexp@down}
\addtocounter{binomexp@down}{-1}}
```

`\binomexp@columns` an array of so many columns

```
\newcounter{binomexp@columns}
```

`\binomexp@power` $(f+s)^{power}$

```
\newcounter{binomexp@power}
```

`\binomexp@pascalstart` `\binomexp@pascalstop` `\binomexp@emptytimes` the next 3 counters are used within the `\binomexp@putpascal` command

```
\newcounter{binomexp@pascalstart}
\newcounter{binomexp@pascalstop}
\newcounter{binomexp@emptytimes}
```

`\binomexp@variable1` `\binomexp@variable2` `\binomexp@answervar` the following 3 counters are used within the process of calculation as `\binomexp@printpascal`

```
\newcounter{binomexp@variable1}
\newcounter{binomexp@variable2}
\newcounter{binomexp@answervar}
```

`\binomexp@sub`

```
\newcounter{binomexp@sub}
```

`\binomexp@printpascal` to calculate the coefficients of the Pascal's triangle

```
\protect\newcommand*{\binomexp@printpascal}{
\addtocounter{binomexp@power}{1}
\expandafter\edef\csname
binomexp@morten\roman{binomexp@power}exporti\endcsname{1}
\setcounter{binomexp@sub}{2}
```

```
\setcounter{binomexp@variable1}{\numexpr\number\value{binomexp@power}+1\relax}
\whiledo{\number\numexpr\number\value{binomexp@power}+1\relax>
\value{binomexp@sub}}{
\setcounter{binomexp@variable1}{\numexpr\number\value{binomexp@sub}-1\relax}
\setcounter{binomexp@variable2}{\value{binomexp@sub}}
\setcounter{binomexp@answervar}{\number\numexpr\csname
binomexp@x\roman{binomexp@variable1}\endcsname\relax+\number\numexpr\csname
binomexp@x\roman{binomexp@variable2}\endcsname\relax}
\expandafter\edef\csname binomexp@y\roman{binomexp@sub}\endcsname
{\number\value{binomexp@answervar}}\relax
\addtocounter{binomexp@sub}{1}
}
```

TRANSFER PART set counter as binomexp@sub to 1

```
\setcounter{binomexp@sub}{2}
```

create a loop which shall get the binomexp@y values and put those into the appropriate binomexp@x values. Also export the y values by this same corresponding power into a length called binomexp@morten`\roman{power}`export`\roman{binomexp@sub}`

```
\whiledo{\numexpr\number\value{binomexp@power}+1\relax>\value{binomexp@sub}}{
\setcounter{binomexp@answervar}{\number\numexpr\csname
binomexp@y\roman{binomexp@sub}\endcsname\relax}
\expandafter\edef\csname binomexp@x\roman{binomexp@sub}\endcsname
{\number\value{binomexp@answervar}}
```

Here is how I exported the values to the table.

```
\expandafter\edef\csname
binomexp@morten\roman{binomexp@power}export\roman{binomexp@sub}\endcsname
{\number\value{binomexp@answervar}}

\addtocounter{binomexp@sub}{1}
}
\setcounter{binomexp@variable1}
{\numexpr\number\value{binomexp@power}+1\relax}
\expandafter\edef\csname
binomexp@x\roman{binomexp@variable1}\endcsname{1}
\expandafter\edef\csname
binomexp@morten\roman{binomexp@power}export\roman{binomexp@variable1}\endcsname{1}
```

To see what is happening add the following lines at this place.

```
power is \number\value{binomexp@power}\par
\setcounter{binomexp@variable2}{1}
\whiledo{\value{binomexp@variable2}<
\numexpr\number\value{binomexp@power}+2\relax}{
binomexp@morten\roman{binomexp@power}export\roman{binomexp@variable2} is
\csname binomexp@morten\roman{binomexp@power}export\roman{binomexp@%
variable2}\endcsname\relax\par\addtocounter{binomexp@variable2}{1}}


}
```

`\binomexp@putpascal` set binomexp@xi as 1
binomexp@xi never alters

```
\newcommand*\binomexp@putpascal[6]{\par
\expandafter\edef\csname binomexp@xi\endcsname{1}
```

set an eventuality for binomexp@xi by the power as zero

```
\expandafter\edef\csname binomexp@mortenexporti\endcsname{1}
```

we'll need to start power as zero by the way `\binomexp@printpascal` is transfigured.

```
\setcounter{binomexp@power}{0}
```

wrap the chipolatas in stringy bacon.

```
\setcounter{binomexp@pascalstart}{#1}
\setcounter{binomexp@pascalstop}{#2+1}
```

now calculate all the co-efficients.

```
\setcounter{binomexp@emptytimes}{\value{binomexp@pascalstop}}
\whiledo{\value{binomexp@emptytimes}>1}{
\binomexp@printpascal \addtocounter{binomexp@emptytimes}{-1}
}
```

work out the number of columns

```
\setcounter{binomexp@columns}
{\numexpr\number\value{binomexp@pascalstop}+2\relax}
```

now the table

```
\begin{math} \begin{array}{@{}|c|*{\value{binomexp@columns}}{|c}|@{}}
```

repeat the number of rows so many times

```
\whiledo{\value{binomexp@pascalstart}<
\numexpr\number\value{binomexp@pascalstop}-1\relax}{
```

prime the binomexp@up gun and cock.

```
\setcounter{binomexp@up}{0}
```

prime the binomexp@down gun and cock.

```
\setcounter{binomexp@down}{\value{binomexp@pascalstart}}
\binomexp@call{(#3+#4)^{\number\numexpr\number\value{binomexp@pascalstart}\relax}}
\binomexp@replicate{\numexpr\number\value{binomexp@pascalstart}+1\relax}
{&\binomexp@call{\csname
binomexp@morten\romannumeral\numexpr\value{binomexp@pascalstart}\relax
export\romannumeral\numexpr\value{binomexp@up}+1\relax\endcsname
#5^{\binomexp@down} #6^{\binomexp@up}}}\\
\addtocounter{binomexp@pascalstart}{1}
}
```

add one more row for luck

```
\setcounter{binomexp@up}{0}
\setcounter{binomexp@down}{\value{binomexp@pascalstart}}
\binomexp@call{(#3+#4)^{\number\numexpr\number\value{binomexp@pascalstart}\relax}}
\binomexp@replicate{\numexpr\number\value{binomexp@pascalstart}+1\relax}
{&\binomexp@call{\csname
binomexp@morten\romannumeral\numexpr\value{binomexp@pascalstart}\relax
export\romannumeral\numexpr\value{binomexp@up}+1\relax\endcsname
#5^{\binomexp@down} #6^{\binomexp@up}}}
 \end{array} \end{math}
}
```

`\binomexp@proof` This command prints a mathematical proof of the Pascals's triangle based upon obervation.

```
\newcommand{\binomexp@proof}[2]{
```

```
\[ 7!=7\cdot6\cdot5\cdot4\cdot3\cdot2\cdot1 \hspace*{5em}
{#1 \choose #2} = \frac{#1!}{(#1-#2)!\cdot #2!}=
\frac{#1!}{(#1-#2)!\cdot(#1-(#1-#2))!}={#1 \choose #1-#2}
\]
\begin{eqnarray*} {#1 - 1 \choose #2 - 1} + {#1 - 1 \choose #2}
&=& \frac{(#1 - 1)!}{(#2 - 1)!\cdot[(#1 - 1) - (#2 - 1)]!} +
\frac{(#1 - 1)!}{#2!\cdot[(#1 - 1) - #2)]!}\\
&=& (#1 - 1)!\cdot\left(\frac{1}{(#2 - 1)!\cdot(#1 - #2)!} +
\frac{1}{#2!\cdot[(#1 - #2) - 1)]!}\right)  \\
&=& (#1 - 1)! \cdot\frac{#2 + (#1 - #2)}{#2! (#1 - #2)!} \\
&=& \frac{#1!}{#2!(#1 - #2)!} = {#1 \choose #2}
 \end{eqnarray*}
\[ \frac{#2}{#2!\cdot(#1-#2)!} = \frac{1}{(#2-1)!\cdot(#1-#2)!}
\hspace*{5em} \mbox{because} \hspace*{5em} \\
 \frac{6}{6!\cdot(#1-#2)!} = \frac{1}{5!\cdot(#1-#2)!} \]
\begin{eqnarray*}
(#2 + 1)\cdot {#1 + 1 \choose #2 + 1} &=& (#2 + 1)\cdot
\frac{(#1 + 1)!}{((#2 + 1)!\cdot ((#1 + 1) - (#2 + 1))!}\\
&=& (#2 + 1)\cdot \frac{(#1 + 1)!}{(#2 + 1)!\cdot (#1 - #2)!}\\
&=& (#1 + 1)\cdot \frac{#1!}{#2!\cdot (#1 - #2)!} = (#1 + 1)\cdot
{#1 \choose #2}\\
\end{eqnarray*}
}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.