

AcroTeX.Net

The rangen Package

Random Generation of Integer, Rational, and Real Numbers with Applications to the **exercise**, **quiz**, and **shortquiz** Environments of **Exerquiz**

D. P. Story

Copyright © 2016 dpstory@acrotex.net
Prepared: February 20, 2016

www.acrotex.net
Version 1.4

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 2 | Requirements | 4 |
| 2.1 | TeX Package Requirements | 4 |
| 2.2 | PDF Creator Requirements | 4 |
| 3 | Installation | 4 |
| 4 | Package Options | 5 |
| 5 | Basic Commands | 5 |
| 5.1 | \RandomZ | 6 |
| 5.2 | \RandomQ | 8 |
| 5.3 | \RandomR | 11 |
| 5.4 | \RandomL | 14 |
| 5.5 | \RandomI | 15 |
| 5.6 | \RandomP | 15 |
| 5.7 | \RandomS | 17 |
| 5.8 | Commands that Operate on Numbers | 18 |
| | • \nof and \dof | 18 |
| | • Special Formatting Commands: The \ds and \fmt families | 19 |
| | • Getting the Data type with \typeOf | 20 |
| 5.9 | \defineZ, \defineQ, and \defineR | 21 |
| 6 | rangen and fp | 22 |
| 7 | rangen and exerquiz | 23 |
| 7.1 | Creating Quizzes using rangen | 23 |
| 7.2 | Creating Solutions to Random Quizzes | 27 |
| | Solutions to Quizzes | 29 |

1. Introduction

This is a package that I began back in the year 2000 AD; at that time, I managed to obtain a working version up and running with many bugs, then forgot about it. Now, in my retirement, I stumbled across the work and decided to give it another go.

The `rangen` package, as the title implies, can (pseudo-)randomly generate integers, rationals, and real numbers. Generate said numbers using the `\RandomZ`, `\RandomQ`, and `\RandomR` commands, respectively; in addition to these, there is `\RandomL` for creating a list of numbers, from which one number is selected at random, and `\RandomI` for generating a random index value that can be used in conjunction with `\RandomL`.

The AcroTeX eEducation Bundle (AeB) contains a package called `exerquiz` that is used to create exercises and quizzes. My goal in writing the `rangen` package was to integrate it with the quiz system of `exerquiz` so that quiz questions could be composed using the “natural” syntax of `rangen`, each time the source file is \LaTeX ed, new random numbers populate the question. To get your interest, here is an example,

Quiz Arithmetic. A simple arithmetic problem, I’ve created one problem, then copied it to make two problems.

$$1. \frac{3}{16} - \frac{5}{8} = \qquad 2. \frac{7}{8} - \frac{3}{8} =$$

Indefinite Integration. There are two integration problems, again, the second question is a copy and paste of the first. The parameters of the problem were, of course, populated by different random numbers.

$$3. \int \frac{2}{3}x^2 + \frac{5}{6}x + 3 \, dx =$$

$$4. \int \frac{3}{2}x^2 + x + 1 \, dx =$$

Definite integration can also be posed, but is not illustrated here.

Analytic Geometry. Find the equation of the line that passes through P and Q .

$$5. P(3, 0), Q(5, 6):$$

6. $P(7,4)$, $Q(8,10)$:

Each time the manual is compiled, a new set of problems of the same type is generated. This package is pretty impressive, I'm sure you'll agree. ~~DS~~
These examples were taken from the demo file `rangen_tst.tex`.

2. Requirements

The requirements for your \LaTeX system, and well as any other software, is highlighted in this section.

2.1. \LaTeX Package Requirements

The following packages, in addition to the standard \LaTeX distribution, are required:

1. The `lcg` package (2008/09/10 v1.2) by Erich Janka.
2. The `hyperref` package, a recent version.
3. If you want to use `rangen` to create quizzes, then `exerquiz` of `AeB` is required.¹

2.2. PDF Creator Requirements

The package works for all PDF creators: Acrobat Distiller, `pdftex`, and `dvipdfm`.

3. Installation

Unzip `rangen.zip` into your \LaTeX tree, the folder `rangen` is constructed with to contain the installation.

¹AeB: <http://www.math.uakron.edu/~dpstory/webeq.html>

4. Package Options

Currently, there is only one option, `testmode`. Then this option is used, each time the file is run, the random number generator of `lcd` is re-seeded. Normally, the seed is based on the time, the date and other factors; the clock of the $\text{T}_{\text{E}}\text{X}$ compiler gives the time to the nearest minute, so one must wait at least a minute before getting a new seed, this is not acceptable when testing a package. When `testmode` is used, the initial seed is `seed=1`, and increments by one thereafter; after the increment, this value is saved to the file `\jobname.seed` and input back in on the next compile.

Any other options that are passed to `rangen`, are passed on to the `lcg` package. Useful options for `lcg` are `quiet` and `seed=<number>`.

5. Basic Commands

This package defines the commands `\RandomZ`, `\RandomQ`, and `\RandomR`, `\RandomL`, and `\RandomI`. We describe these commands in this section.

For convenience of terminology, a number created by one of the above commands will be referred to as a RV (random variable).

`\RandomZ` and `\RandomQ` use the count registers, so there is a restriction on the size of any RV generated by these two commands, we must have

$$-2^{31} + 1 \leq RV \leq 2^{31} - 1 \implies -2147483647 \leq RV \leq 2147483647$$

For simple applications envisioned for `rangen`, this range should be plenty enough.

The `\RandomR` command uses the dimension registers, so a RV generated by `\RandomR` is restricted to

$$-2^{14} < RV < 2^{14} \implies -16384 < RV < 16384$$

Again, this is not a package for making floating point calculations, it is a package for generating integers, rationals, and decimal numbers with an eye towards application to academic problem generation. Floating point arithmetic can be accomplished using the `fp` package; `rangen` and `fp` seem to be compatible.

5.1. `\RandomZ`

The command `\RandomZ` defines a random integer, the syntax is

```
\RandomZ[⟨key-values⟩]{⟨name⟩}{⟨zLEP⟩}{⟨zUEP⟩}
```

Parameter Description:

⟨key-values⟩ The key-value pairs that modify the choice of the variable. The key-value pairs recognize are

`ne=⟨value⟩` Restrict the choice of the random integer by requiring it *not be equal* to another number, for example, `ne=0` or `ne=\b`. In the latter case, `\b` is a number defined already by either an earlier `\RandomZ` call, or by `\defineZ`, discussed later. Multiple restrictions can be placed as well, for example, if `ne={0, -1}`, `rangen` selects an integer different from 0 or -1.

⟨name⟩ The name of the random integer. For example, `\a`, `\b`, etc.

⟨zLEP⟩ An integer that is the lower endpoint of the interval from which the number is randomly selected. The lower endpoint may be an integer previously calculated by an earlier `\RandomZ` call, for example, `\RandomZ{\b}{\a}{5}`, this will generate an integer `\b` such that $\a \leq \b \leq 5$. To get strict inequality, append an asterisk (*), for example, if the syntax `\RandomZ{\b}{\a*}{5}` is used, then `rangen` attempts to satisfy $\a < \b \leq 5$. The range of `\a` should be such that the upper limit for `\a` is less than the upper limit of `\b`. When the endpoint is a number, the * is ignored.

When the lower endpoint is a command created by the commands `\Random⟨Z|Q|R|L⟩` or by `\define⟨Z|Q|R⟩`, the endpoint is converted to a real number.

⟨zUEP⟩ An integer that is the upper endpoint of the interval from which the number is randomly selected. The upper endpoint may be an integer previously calculated by an earlier `\RandomZ` call, for example, `\RandomZ{\b}{-5}{\a}`, this will generate an integer `\b` such that $-5 \leq \b \leq \a$. To get strict inequality, use the syntax `-5 \le \b < \a*`, then `rangen` attempts to satisfy $-5 \leq \b < \a$. The range of `\a`

should be such that the lower limit for $\backslash a$ is greater than the lower limit of $\backslash b$. When the endpoint is a number, the $*$ is ignored.

When the upper endpoint is a command created by the commands $\backslash\text{Random}\langle Z|Q|R|L\rangle$ or by $\backslash\text{define}\langle Z|Q|R\rangle$, the endpoint is converted to a real number.

Examples:

1. $\backslash\text{RandomZ}\{\backslash a\}\{-5\}\{5\}$: $\backslash a=-5$. To get another random integer, we repeatedly execute $\backslash\text{RandomZ}\{\backslash a\}\{-5\}\{5\}$ followed by $\backslash a$, for example, we copy and paste $\backslash\text{RandomZ}\{\backslash a\}\{-5\}\{5\}\backslash a$ three times to get 2, -1, -4.
2. Illustrate ne : Consider $\backslash\text{RandomZ}[\text{ne}=\{0, -1\}]\{\backslash a\}\{-5\}\{5\}\backslash a$, copy and paste repeatedly: -3, 4, 3, -4, and 1. If rangen worked as it should, the list of five number should not contain a 0 or a -1, does it?
3. Illustrate $\backslash a \leq \backslash b$: We use the code

```
 $\backslash\text{RandomZ}\{\backslash a\}\{-5\}\{5\}\backslash\text{RandomZ}\{\backslash b\}\{\backslash a\}\{10\}\backslash a \leq \backslash b$ 
```

We now copy and paste this code: $0 \leq 2$, $2 \leq 9$. To get strict inequality we execute

```
 $\backslash\text{RandomZ}\{\backslash a\}\{-5\}\{5\}\backslash\text{RandomZ}\{\backslash b\}\{\backslash a^*\}\{10\}\backslash a < \backslash b$ 
```

We now copy and paste this code: $2 < 3$, $-4 < 10$

Data Type Properties. When a random number is created, there are several auxiliary commands that are defined.

| | |
|--|--|
| $\backslash\text{nOf}\{\langle name \rangle\}$ | $\backslash\text{dOf}\{\langle name \rangle\}$ |
| $\backslash\text{fmt}\{\langle name \rangle\}$ | $\backslash\text{ds}\{\langle name \rangle\}$ |

Command Description:

$\backslash\text{nOf}\{\langle name \rangle\}$ is the numerator for the number $\langle name \rangle$. For an integer this is just $\langle name \rangle$. This function becomes important for rational numbers. If $\backslash a$ is the rational number $2/3$, then $\backslash\text{nOf}\{\backslash a\}=2$.

`\dof{\langle name \rangle}` is the denominator for the number `\langle name \rangle`. For an integer this is just 1. This function becomes important for rational numbers. If `\a` is the rational number $2/3$, then `\dof{\a}=3`.

`\fmt{\langle name \rangle}` allows for special formatting for in-line numbers. Without one of the special formatting options, `\fmt\a` is the same as `\a`.

`\ds{\langle name \rangle}` allows for special formatting for display style number. `\ds` is relevant for rational numbers. If `\a` represents the rational $1/2$, the `\a` expanded is $1/2$, while `\ds\a` expanded is $\frac{1}{2}$. The `\ds` command also obeys the formatting options.

5.2. `\RandomQ`

The command `\RandomQ` defines a random rational, the syntax is

```
\RandomQ[\langle key-values \rangle]{\langle name \rangle}[\langle max_denom \rangle]{\langle qLEP \rangle}{\langle qUEP \rangle}
```

Parameter Description:

`\langle key-values \rangle` The key-value pairs that modify the choice of the variable. The key-value pairs recognize are

`ne=\langle value \rangle` Restrict the choice of the random rational by requiring it *not be equal* to another number, for example, `ne=0` or `ne=\b`. In the latter case, `\b` is a number defined already by either an earlier `\RandomQ` call, or by `\defineQ`, discussed later. Multiple restrictions can be placed as well, for example, if `ne={0, -1}`, `rangen` selects an integer different from 0 or -1.

`\langle name \rangle` is the name of the random rational. For example, `\a`, `\b`, etc.

`\langle max_denom \rangle` is the largest denominator you want your random rational to have. For example, `\RandomQ{\a}[9]{1/2}{7/2}`: The value of `\a` is a rational number between $1/2$ and $7/2$ having a maximum denominator of 9. If this parameter is not specified, the least common denominator is used; for the example, that would be 2. To contrast the two, consider the following examples:

1. `\RandomQ{\a}[9]{1/2}{7/2}\a`: $8/3$, $17/9$, $22/9$, and $14/9$.
2. `\RandomQ{\a}{1/2}{7/2}\a`: 2, 1, $7/2$, 1.

The fractions are reduced to lowest terms, and represented as an integer if needed.

Here is more detail on the algorithm used to generate a rational: We illustrate using the example, `\RandomQ{\a}[9]{1/2}{7/2}`, the details are simplified slightly.

1. Convert the range so that the endpoints have a denominator of 9.

$$\text{LEP: } \frac{1}{2} = \frac{9/2}{9} = \frac{4.5}{9} < \frac{5}{9} \quad \text{round up}$$

$$\text{UEP: } \frac{7}{2} = \frac{63/2}{9} = \frac{31.5}{9} > \frac{31}{9} \quad \text{round down}$$

2. We randomly choose an integer between 5 and 31, call it `\z`; our random rational is then `\z/9`, unless there is an `*` affixed to one of both endpoints.
3. If one or both endpoints is itself a random rational (or integer) and the `*` character is used, then the lower end of the range is incremented (from 5 to 6) and/or the upper end is decremented (from 31 to 30).
4. Reduce the fraction obtained in the previous step.

You can see from this example, there are a lot of choices for the random integer, there are 27 possibilities between 5 and 32.

`\langle qLEP \rangle` is a rational (of the form `a/b`) that is the lower endpoint of the interval from which the number is randomly selected. The lower endpoint may be a rational (or integer) previously calculated by an earlier `\RandomQ` call, for example, `\RandomQ{\b}{\a}{4/3}`, this will generate an integer `\b` such that `\a ≤ \b ≤ 4/3`. To get strict inequality, use the syntax `\RandomZ{\b}{\a*}{4/3}`, then `\rangen` attempts to satisfy `\a < \b ≤ 4/3`. The range of `\a` should be such that the upper limit for `\a` is less than the upper limit of `\b`. When the endpoint is a number, the `*` is ignored.

When the lower endpoint is a command created by any of the commands `\Random<Z|Q|R|L>` or `\define<Z|Q|R>`, the endpoint is converted to a real number.

$\langle qUEP \rangle$ is a rational that is the upper endpoint of the interval from which the number is randomly selected. The upper endpoint may be a rational (or integer) previously calculated by an earlier `\RandomQ` call, for example, `\RandomQ{\b}{-4/3}{\a}`, this will generate an integer `\b` such that $-4/3 \leq \b \leq \a$. To get strict inequality, use the syntax $-4/3 \leq \b \leq \a*$, then `rangen` attempts to satisfy $-4/3 \leq \b < \a$. The range of `\a` should be such that the lower limit for `\a` is greater than the lower limit of `\b`. When the endpoint is a number, the `*` is ignored.

When the upper endpoint is a command created by any of the commands `\Random<Z|Q|R|L>` or `\define<Z|Q|R>`, the endpoint is converted to a real number.

Examples

1. `\RandomZ{\a}{-5}{5}`: `\a=-5`. To get another random integer, we repeatedly execute `\RandomZ{\a}{-5}{5}` followed by `\a`, for example, we copy and paste `\RandomZ{\a}{-5}{5}\a` three times to get 2, -1, -4.

2. Illustrate `ne`: Copy and paste

```
\RandomZ[ne={0,-1}]{\a}{-5}{5}\a
```

repeatedly to obtain: -3, 4, 3, -4, and 1. If `rangen` worked as it should, the list of five number should not contain a 0 or a -1, does it?

3. Illustrate $\a \leq \b$: We use the code

```
\RandomZ{\a}{-5}{5}\RandomZ{\b}{\a}{10}$\a \le \b$
```

We now copy and paste this code: $0 \leq 2$, $2 \leq 9$. To get strict inequality we execute

```
\RandomZ{\a}{-5}{5}\RandomZ{\b}{\a*}{10}$\a < \b$
```

We now copy and paste this code: $2 < 3$, $-4 < 10$

4. `\RandomQ{\a}{-5/2}{5/2}`: `\a=1/2`. We repeatedly copy and paste `\RandomQ{\a}{-5/2}{5/2}\a` three times to get -1, 2, -2.

5. Illustrate `\RandomQ[ne={0,-1}]{a}{-5/2}{5/2}`a, copy and paste this code repeatedly: 1, 2, 5/2, -2, and -2. If `\rangen` worked as it should, the list of five number should not contain a 0 or a -1, does it?
6. Illustrate $a \leq b$: We use the code

```
\RandomQ{a}{-5/2}{5/2}\RandomQ{b}[4]{a}{10}$a \le b$
```

and copy and paste: $-1 \leq 11/2$, $3/2 \leq 5/2$. To get strict inequality we execute

```
\RandomQ{a}{-5/2}{5/2}\RandomQ{b}[4]{a*}{10}$a < b$
```

to get $-2 < 1$, $0 < 39/4$

For a rational number, the commands `\nOf`, `\dOf`, `\fmt`, and `\ds` are also defined, see ‘[Data Type Properties](#)’ on page 7.

5.3. `\RandomR`

The command `\RandomR` defines a random real number, the syntax is

```
\RandomR[⟨key-values⟩]{⟨name⟩}{⟨rLEP⟩}{⟨rUEP⟩}
```

Parameter Description:

`⟨key-values⟩` The key-value pairs that modify the choice of the variable. The key-value pairs recognize are

`round=⟨value⟩` Round the generated real number so that number of decimal places equals the `⟨value⟩`, the value of the `round` key; for example, `round=2` rounds the result to 2 decimal places.

`showzeros=⟨true|false⟩` Show trailing zeros, only valid when the `round` key is used. For example, `round=4`, `showzeros` might yield a result of 3.2300, whereas without the `showzeros` key (or with `showzeros=false`), the same result would be 3.23.

`ne=⟨value⟩` Restrict the choice of the random real by requiring it *not be equal* to another number real, for example, `ne=-1` or `ne=b`. In the latter case, `b` is a number defined already by either an earlier `\RandomR` call, or by `\defineR`, discussed later. Multiple

restrictions can be placed as well, for example, if $\text{ne}=\{0, -1\}$, `random` selects an integer different from 0 or -1.

Note, comparisons are made *after* rounding.

`\langle name \rangle` is the name of the random rational. For example, `\a`, `\b`, etc.

`\langle rLEP \rangle` is a real number (or integer) that is the lower endpoint of the interval from which the number is randomly selected. The lower endpoint may be a number previously calculated by an earlier `\RandomR` call, for example, `\RandomR{\b}{\a}{1.3}`, this will generate an integer `\b` such that $\a \leq \b \leq 1.3$. To get strict inequality, use the syntax `\RandomZ{\b}{\a*}{1.3}`, then `random` attempts to satisfy $\a < \b \leq 4/3$. The range of `\a` should be such that the upper limit for `\a` is less than the upper limit of `\b`. When the endpoint is a number, the `*` is ignored.

When the lower endpoint is a command created by `\ant` if the command `\Random<Z|Q|R|L>`, or by `\define<Z|Q|R>`, the endpoint is converted to a real number.

`\langle rUEP \rangle` is a real number that is the upper endpoint of the interval from which the number is randomly selected. The upper endpoint may be a number previously calculated by an earlier `\RandomR` call, for example, `\RandomR{\b}{-1.3}{\a}`, this will generate an integer `\b` such that $-1.3 \leq \b \leq \a$. To get strict inequality, use the syntax `-1.3 \leq \b \leq \a*`, then `random` attempts to satisfy $-1.3 \leq \b < \a$. The range of `\a` should be such that the lower limit for `\a` is greater than the lower limit of `\b`. When the endpoint is a number, the `*` is ignored.

When the upper endpoint is a command created by `\ant` if the command `\Random<Z|Q|R|L>`, or by `\define<Z|Q|R>`, the endpoint is converted to a real number.

The `\RandomR` command divides range ($\langle rUEP \rangle - \langle rLEP \rangle$) into equal sub-intervals, and randomly chooses node (an endpoint of one of the sub-intervals). The number of subdivisions is determined by `\RNGpowerOfTen`, and can be set by `\nDivisionsPowerOfTen`. This latter command takes an integer argument, n , $1 \leq n \leq 4$, the number of subdivisions is then 10^n . Strictly speaking `\RNGpowerOfTen` does not have to be a power of 10, you

can make the definition `\def\RNGpowerOfTen{16}`, and that should work as well. The default is `\nDivisionsPowerOfTen{2}`, that is, divide the range into 100 equal subdivisions.

Examples:

1. `\RandomR{\a}{-2.3}{2.3}`: $\a=-1.8401$. To obtain more random real numbers, repeatedly execute `\RandomR{\a}{-2.3}{2.3}` followed by `\a`; for example, `\RandomR{\a}{-2.3}{2.3}\a` repeated three times yields the following:

2.20702, - 1.33421, - 0.9203

2. `round`: We use `\RandomR[round=4]{\a}{-2}{2}\a` to get

1.9193, - 1.1601, - 0.8002

3. `showzeros`: `\RandomR[round=4,showzeros]{\a}{-2}{2}\a`:

1.9193, - 1.1601, - 0.8002, - 1.4401, - 1.1202

4. Illustrate $\a \leq \b$: We use the code

`\RandomR{\a}{-5}{5}\RandomR{\b}{\a}{10}\a \le \b`

We now copy and paste this code twice:

4.7991 \leq 5.89114, - 2.00027 \leq -0.32034

To get strict inequality we execute

`\RandomR{\a}{-5}{5}\RandomR{\b}{\a*}{10}\a < \b`

We now copy and paste this code a couple of times:

4.7991 < 6.20316, - 2.00027 < 2.43954

For real numbers, there is little chance the two values are equal, but just in case, strict inequality may be specified in this way.

5.4. `\RandomL`

The command `\RandomL` defines a list of numbers (integer, rational, decimal), and randomly selects a number from the list.

```
\RandomL[⟨key-values⟩]{⟨name⟩}{⟨n1, n2, n3, . . .⟩}
```

Parameter Description:

⟨*key-values*⟩ The only key-value pairs recognized is `index=⟨posZ⟩`. The `index` is a base-1 index, thus `index=1` references the first number in the list.

The `index` key can be used to retrieve a particular number from this list; for example, declaring `\RandomL[index=2]{\a}{17, 1/2, 1.3}`, the value of `\a` is `1/2`.

The value of `index` can be any positive integer, even one generated using `\RandomI`. If the value of `index` is greater than the number of items in the list, modular arithmetic is performed to put the index back into the proper range.

When the `index` key is not present, a number is randomly selected from the list.

⟨*name*⟩ is the name of the number generated. The number generated will be defined as integer, rational, or real; consequently `\nof`, `\dof`, `\fmt`, and `\ds` are defined.

⟨*n₁, n₂, n₃, . . .*⟩ A (possibly mixed) list of numbers. The numbers can be literal (12, 1.2, 3/4), or control sequences of numbers (commands) defined earlier by `\Random⟨Z|Q|R|L⟩` or by `\define⟨Z|Q|R⟩`.

Examples

1. After declaring `\RandomL{\a}{17, 3.14, 88, 3/4, 1/2}`, the value of `\a` is `\a = 88`, Select a number from this list at random `\a=17`, again `\a=3.14`, and again `\a=3.14`.
2. `\RandomL[index=3]{\a}{17, 3.14, 88, 3/4, 1/2}\a, \a=88.`

5.5. `\RandomI`

The command `\RandomI` defines a list of integers, $\{1, 2, 3, \dots, n\}$ and randomly selects an integer, thought of as an index value, from the list.

```
\RandomI{\langle name \rangle}{\langle n \rangle}
```

Parameter Description:

`\langle name \rangle` is the name of the number generated, the number will be defined as an integer number.

`\langle n \rangle` is a positive number greater than 1. The list $\{1, 2, 3, \dots, n\}$ is implicitly created.

Example: `\RandomI{\indx}{20}\indx` yields 3, 16, 7, 12, when copied and pasted four times.

My thought in creating `\RandomI` is to use it in conjunction with `\RandomL` (using the `index` key). For example,

```
\RandomI{\indx}{4}
\RandomL[index=\indx]{\a}{1/2,1/3,1/4,1/5}
\RandomL[index=\indx]{\b}{5/3,6/5,7/2,5/6}
\begin{equation*}
(\a)+(\b) =
\end{equation*}
```

This code results in the following arithmetic problem:

$$(1/2) + (5/3) =$$

This is probably not a good example of the usage of `\RandomI`. See the next section on `\RandomP`.

5.6. `\RandomP`

The command `\RandomP` defines a list of strings (literal expressions), and randomly selects one from the list. (The “P” in `\RandomP` stands for “Problem.”)

```
\RandomP[\langle key-values \rangle]{\langle name \rangle}{\langle list of literals \rangle}
```

Parameter Description:

$\langle key-values \rangle$ The only key-value pairs recognized is `index= $\langle posZ \rangle$` . The index is a base-1 index, thus `index=1` references the first number in the list.

The `index` key can be used to retrieve a particular literal from this list; for example, by executing `\RandomP[index=2]{\a}{d,p,s}`, the value of `\a` is `p`.

The value of `index` can be any positive integer, even one generated using `\RandomI`, or by another list. If the value of `index` is greater than the number of items in the list, modular arithmetic is performed to put the index back into the proper range.

When the `index` key is not present, a number is randomly selected from the list.

$\langle name \rangle$ is the name of the literal generated.

$\langle list\ of\ literals \rangle$ is a comma-delimited list of literal strings, selected literal is not interpreted as a number, but is passed into the definition of $\langle name \rangle$.

Examples

1. Executing `\RandomP{\a}{1+16,\cos(\pi),%
 $\frac{d}{dx}\frac{1}{2}x^2,\int \cos(x)\,dx}$
 $\texttt{\string\a} = \a$, we get $\a = \cos(\pi)$, and then again, $\a = 1 + 16$.`
2. Use `\RandomI` with `\RandomP`. You can create a series of questions and answers using these two:

```
\RandomI{\indx}{5}
\RandomP[index=\indx]{\q}{1+16,\cos(\pi),\pi\sin(\pi),%
\frac{d}{dx}\frac{1}{2}x^2,\int \cos(x)\,dx}
\RandomP[index=\indx]{\a}{17,-1,0,x,\sin(x)+C}
\begin{equation*}
\q = \a
\end{equation*}
```


The execution of these lines becomes

$$\frac{d}{dx} \frac{1}{2} x^2 = x$$

You can create a switch to include the answer or not.

- There is an alternate approach to this previous example. Random lists (`\RandomL` and `\RandomP`) define a macro `\iOf`, the value of which is the index of the item selected (at random). We can use `\iOf` in the above problem as follows:

```
\RandomP{\q}{1+16,\cos(\pi),\pi\sin(\pi),%
  \displaystyle\frac{d}{dx}\frac{1}{2}x^2,%
  {\int \cos(x)\,dx}}
\RandomP[index=\iOf{\q}]{\a}{17,-1,0,x,\sin(x)+C}
\begin{equation*}
  \q = \a
\end{equation*}
```

The execution of these lines gives the output...

$$1 + 16 = 17$$

Here, we select the answer to the randomly chosen question.

5.7. `\RandomS`

The command `\RandomS` generates a random sign, either + or -. This may be useful for creating addition/subtraction problems.

```
\RandomS[<dec>]{\<name>}
```

Parameter Description:

`<dec>` is a number between 0 and 1. This command generates a + sign with probability `<dec>`. The default value is 0.5.

`\<name>` is the name that references the generated random sign.

Examples

1. Random addition problem:

```
\RandomZ{\a}{1}{20}\RandomZ{\b}{1}{20}\RandomS{\s}
\begin{equation*}
  \a \s \b
\end{equation*}
```

This code expands to

$$19 + 14$$

Whether we add or subtract the summands is determined by the command `\s`.

2. Random Differentiation problem:

```
\RandomQ{\a}[8]{1}{2}\RandomQ{\b}[8]{2}{3}
\RandomZ{\n}{1}{6}\RandomS{\si}\RandomS{\sii}
```

```
Differentiate
\begin{equation*}
  \frac{d}{dx}(\bigl (\a) \si (\b) x^{\sii\ n}\bigr)
\end{equation*}
```

Differentiate

$$\frac{d}{dx}((13/8) + (9/4)x^{-6})$$

5.8. Commands that Operate on Numbers

Associated with each data type (integer, rational, and real) are several useful commands `\nOf`, `\dOf`, `\iOf`, `\fmt`, and `\ds`.

- **`\nOf` and `\dOf`**

For integer, rational, and real numbers `\nOf` and `\dOf` are the numerator and denominator, respectively.

- Integer: `\nOf` is the integer, and `\dOf` is 1; for example, define an integer by `\RandomZ{\a}{-5}{5}`, `\a=-1`, `\nOf{\a}=-1`, `\dOf{\a}=1`, as advertised.

- Rational: `\nOf` is the numerator (an integer), and `\dOf` is the denominator (an integer) of the reduced fraction. For example, define `\a` by `\RandomQ[ne=0]{\a}[9]{-3/2}{3/2}`, then

$$\a = 1/9, \nOf{\a} = 1, \text{ and } \dOf{\a} = 9.$$

- Real: `\nOf` is the numerator (an integer), and `\dOf` is the denominator (an integer) of the reduced fraction, after the real is converted into a rational number. For example, `\RandomR{\a}{.25}{.75}`, then

$$\a = 0.6342, \nOf{\a} = 3171, \text{ and } \dOf{\a} = 5000$$

If we round using with `\RandomR[round=2]{\a}{.25}{.75}`, we get

$$\a = 0.49, \nOf{\a} = 49, \text{ and } \dOf{\a} = 100$$

• Special Formatting Commands: The `\ds` and `\fmt` families

When a RV, such as `\a`, is a rational number type, say `\a=1/3`, the command `\a` expands to `1/3`. To get a display style formatting of the rational use the `\ds` command. The expansion of `\ds\a` is $\frac{1}{3}$.

We have seen in several examples in which the formatting was not always what we'd like. Expressions like x^1 should be x , $1x$ should be x , $-1x$ should be $-x$. The formatting commands `\cfmt` and `\efmt` (and their display style counterparts `\cds` and `\eds`) attempt to format the special cases of 1 and -1, as they appear in an exponent (the 'e' variations) and as they appear as a coefficient (the 'c' variations).

All the formatting commands `\cfmt`, `\efmt`, `\ds`, `\cds`, and `\eds` take a RV as its argument. `\<c|e>fmt\<c|e>ds\a` expands to `\a` when `\a` is not 1 or -1. These cases are covered below.

- For `\a=1`, `\cfmt\a=\efmt\a=\cds\a=\eds\a={}`, the empty string. Thus, if `\a=1`, and we typeset `\a x^{\a}`, we get $1x^1$, which is not the standard way of writing this expression, but if we typeset `\cfmt\a x^{\efmt\a}` we get x , which is correct. Notice that we used `\cfmt` on the baseline, and `\efmt` in the exponent. It does not make any difference here, but it does if `\a=-1`, see the next bullet point.

- `\a=-1`, then

`\cfmt\a = \cds\a = -` (minus sign)

`\efmt\a = \eds\a = -1` (minus one)

Returning to the same expression in the previous bullet, if `\a=-1`, and we typeset `\a x^{\a}`, we get $-1x^{-1}$, which is not the standard way of writing this expression, but, on the other hand, if we typeset `\cfmt\ax^{\efmt\a}` we get $-x^{-1}$, which is correct. Notice the difference cases if I had typeset `\cfmt\a x^{\cfmt\a}`, I would have gotten $-x^-$, not good.

The ‘c’-variation is used for unitary signs, not binary signs. For example, if `\a=-1`, and we typeset `2 + \cfmt\a x`, we get $2 + -x$, which may be fine in some situations, but most of the time it is not. As a work around, make coefficients positive, and generate a random sign using `\RandomS`; for example, after declaring

```
\RandomS{\s}\RandomZ{\a}{1}{3}
```

and typesetting `2 \s \cfmt\a x`, we get an addition half the time and subtraction the other half: $2 + 3x$, $2 + x$, $2 - 3x$, $2 + x$, $2 - x$, $2 - 3x$.

Similarly, the ‘e’-variation is for unitary sign in the exponent, and should be used when there is a need for these special format rules.

- **Random Sign:** The formatting commands are defined for a random sign created by `\RandomS` and following the same definitions outline above. These are of marginal value in this context.

• Getting the Data type with `\typeOf`

There may be occasions where you want to know the data type of a RV. The `rangen` does change the data type in special cases. For example, if `\a` is created by `\cs{RandomQ}{\a}[2]{1}{3}`, and its value happens to be an integer, `rangen` changes its type of integer. You can determine the type of a RV with the `\typeOf` command, which takes a RV as its argument, the value of `\typeOf` is a nonnegative integer. The following table gives the values of `\typeOf`, and associated data types.

| Data type | <code>\typeOf</code> |
|-----------|----------------------|
| Integer | 0 |
| Rational | 1 |
| Real | 2 |
| Literal | 3 |

A suggested application to `\typeOf`. Suppose, `\a` is a rational RV (for example, `\RandomQ{\a}[2]{1}{3}`), and we want to typeset the expression `\cfmt\a x`. One instance might be $3/2x$, this is not good syntax; so we typeset `\cfmt\a x` to get $(3/2)x$, that's good. But if `\a` is an integer, such as 1, 2, or 3, we get $(2)x$, which contains redundant parentheses. Now we come to the use of `\typeOf`. We now typeset the expression

```
\ifnum\typeOf\a=0\relax\cfmt\a\else(\cfmt\a)\fi x $
```

If `\a` is *not an integer* we get, for `\a=3/2`, we obtain $(3/2)x$, but for `\a=2`, we get $2x$.

5.9. `\defineZ`, `\defineQ`, and `\defineR`

The `rangen` package internally uses `\defineZ`, `\defineQ`, and `\defineR` to define an integer, a rational number, and a real (decimal) number. These command may be used by the document author as well to create non-random variables.

```
\defineZ{\<name>}{<zValue>}
\defineQ{\<name>}{<zNumer>}{<zDenom>}
\defineR{\<name>}{<rValue>}
```

Thus, `\defineZ{\a}{17}` defines `\a=17`, `\defineQ{\a}{-3}{2}` defines `\a=-3/2`, and `\defineR{\a}{17.88}` defines `\a=17.88`.

The various properties data types are created by `\defineZ`, `\defineQ`, and `\defineR`; these are `\nof`, `\dof`, `\typeOf`, `\ds`, `\eds`, `\cds`, `\efmt`, and `\cfmt`.

The following are other important points to remember.

- **Positive Denominators.** Notice that if `\defineQ{\a}{3}{-2}`, then `\a=-3/2`, and `\nof\a=-3`, and `\dof\a=2`. Thus, `rangen` does not allow a negative denominator.

- **Automatic Reduction.** If we declare `\defineQ{\a}{6}{4}`, then $\a=3/2$, a rational number is automatically reduced to lowest terms.
- **Re-classification.** If we make the definition `\defineQ{\a}{6}{2}`, then $\a=3$ is reduced to lowest terms and re-classified as an integer `\typeOf\a=0` (an integer).

6. rangen and fp

After a little bit of testing, it appears that `fp` can work with the `rangen` package. The `rangen` package does not provide any command for combining RVs using such operations as addition, subtraction, multiplication, division, etc.

The `rangen` package does provide several useful commands that `fp` does not, these are `\reduceFrac`, `\gcd`, and `\lcm`.

`\reduceFrac` takes two arguments (numerator and denominator), both integers, and attempts to reduce the implied fraction to lowest terms, and returns the result in two macros `\rfNumer` and `\rfDenom`. For example, to reduce the fraction $4/12$, we execute `\reduceFrac{4}{12}`, which returns `\rfNumer=1`, and `\rfDenom=3`, forming the reduced fraction $1/3$; thus, $4/12 = 1/3$.

The *greatest common divisor* command `\gcd` takes two integers as its arguments and returns its result in the macro `\thegcd`. For example, the `\gcd{4,8}` is 4, while the `\gcd{4}{6}` is 2.

The *least common multiple* command `\lcm` takes two integers as its arguments, and returns its result in the macro `\thelcm`. For example, `\lcm{4}{5}` is `\thelcm=20`, while, `\lcm{4}{6}` is `\thelcm=12`.

The following example illustrates the use of the `rangen` and `fp` packages to pose a random arithmetic problem, and present a detailed solution.

```
\RandomQ{\a}[6]{2}{4}\RandomQ{\b}[6]{2}{4}
\gcd{\dOf\a}{\dOf\b}
\FPeval\lcd{clip((\dOf\a)*(\dOf\b)/\thegcd)}
\FPeval\si{clip(\lcd/(\dOf\a))}
\FPeval\sii{clip(\lcd/(\dOf\b))}
\FPeval\finalnum{clip((\si)*(\nOf\a)+(\sii)*(\nOf\b))}
\defineQ{\ans}{\finalnum}{\lcd}
$$
\ds\a \thisop \ds\b = \frac{(\si)(\nOf\a)+(\sii)(\nOf\b)}{\lcd}
= \frac{\finalnum}{\lcd}\ifnum\lcd=\dOf\ans\else =\ds\ans\fi
$$
```

An instance of this code might look like this:

$$\frac{8}{3} + \frac{17}{6} = \frac{(2)(8) + (1)(17)}{6} = \frac{33}{6} = \frac{11}{2}$$

If there is any reduction of the fraction (brought on by the `\defineQ` command), this additional expression is included.

Demo file. See the demo file `rangen_fp.tex` for a complete example.

7. rangen and exerquiz

Developing a package for randomly generating numbers that could be used as a basis for creating random quizzes (see the example back in [Section 1](#), page 3) was my original motivation for writing the original package back in the year 2000 AD. In this section, we introduce the techniques that I've developed for creating random quizzes, and, more importantly, how to grade them and to exhibit to the user the correct answer. This system is not a computer algebra system, so, it is difficult, but not impossible to also supply a solution (a opposed to just the answer) to the problem as well.

Demo file. Now, let's see how its done! Examples of this section were taken from the demo file `rangen_tst.tex`.

7.1. Creating Quizzes using rangen

The `rangen` package provides three JavaScript functions that are used with `exerquiz` quizzes, these are

- `rEval(str)`: The function `rEval` evaluates its argument. The `rEval` function searches its argument for `rEval` and `rFrac`, and executes any inner nested functions first.
- `rFrac(str)`: Evaluates a rational number by evaluating the value of the numerator and denominator separately. The function `rFrac` searches its argument for `rEval` and `rFrac`, and executes any inner nested functions first.
- `rngCorrAnsButton`: A function that is used to represent the correct answer to the user.

The best way of illustrating these function is by discussing an example or two.

Example 1. We create two RVs, `\a` and `\b` that are rational numbers. We want to add them, and present the answer as a rational number.

```
\RandomQ{\a}[16]{1/8}{15/16}\RandomQ[ne=\a]{\b}[16]{1/8}{15/16}
```

$$1. \frac{7}{16} - \frac{3}{16} =$$

The question is posed using `\RespBoxMath`.

```
1  \displaystyle\ds\a - \ds\b =
2  \RespBoxMath[\rectW{.5in}]{
3      (\nof\a*\dof\b-\nof\b*\dof\a)/(\dof\a*\dof\b)}
4      {2}{.0001}{[0,2]}
5      [{priorParse: \Array(nodect,NoAddOrSub)}]$\
```

Exerquiz determines whether the user's answer is correct, it by evaluating the author's answer at randomly selected points. Exerquiz uses the floating point arithmetic of JavaScript to evaluate the user's answer. The author's correct answer is given in line (1), and it is just the formula for combining two fractions `\a` and `\b`; note the use of `\nof` and `\dof`. Line (2) is standard parameters for `\RespBoxMath`, the number of random points to use, the precision, and the interval from which to select the points. Line (3) specifies a couple of routines from the `dljslib` package, these prevent the user from using decimals and rational arithmetic to answer the question. (The latter function would, for example, prevent the user from copying the question and pasting it into the answer.)

Now comes the most interesting part, at least to me: The presentation of the correct answer to the user. This is where the JavaScript functions `rEval` and `rFrac` are used. The code for the answer button is shown below.

```
1  \CorrAnsButton{rFrac(
2      rEval(\nof\a*\dof\b-\nof\b*\dof\a)/rEval(\dof\a*\dof\b)
3  )}*{rngCorrAnsButton}
```

Here, this code is broken across several lines to fit on the page. We direct the `\CorrAnsButton` to use the function `rngCorrAnsButton`, as seen in line (3). This is a special function define by `rangen` to help in the presentation of the answer to the user.

Keep in mind, the inner-most `rEval` and `rFrac` functions are evaluated first; consequently, the two `rEval` functions in line (2) are evaluated

first. These two evaluations calculate the numerator and denominator separately, this results in a numerical numerator and denominator. The function `rFrac` is then executed on the resulting rational number, this function reduces the fraction to lower terms. This final calculation is what the user sees when the correct answer button is pressed.

The next example will illustrate a decimal presentation of the answer, and introduces a new command, `\RNGprintf`.

Example 2. We create four RVs, `\a`, `\b`, `\c`, and `\n`, three rational and one integer. The exponent of the power is rational, hence, we represent a decimal answer to the user.

```
\RandomQ{\a}[8]{1/4}{7/6}\RandomZ{\b}{1}{3}
\RandomQ{\n}[8]{1/2}{3/2}\RandomZ[ne=\zZero]{\c}{-3}{3}
```

$$2. \int_{5/8}^3 -2x^{11/8} dx =$$

The question is posed using `\RespBoxMath`.

```
1 \displaystyle\int_{\a}^{\b} \cfmt\c x^{\efmt\n}\,dx =
2 \RespBoxMath{\c((\b)^{\n+1}-\a)^{\n+1})/(\n+1)}
3 {3}{.0001}{[0,2]}$
```

The correct answer is given on line (2), and is based on the known form of the integrand; here, we use standard integration formulas.

The code for the correct answer button has a new element in it

```
1 \CorrAnsButton{rEval(
2 \c((\b)^{\n+1}-\a)^{\n+1})/(\n+1)
3 )}{*\rngCorrAnsButton\RNGprintf{\%.4f}}\kern1bp\sqrTallyBox
```

The `rEval` function evaluates the expression on line (2), the result is a decimal number. As before, we use the `rngCorrAnsButton`, but we've added the `\RNGprintf` command to the end of the function name. This is a kludge that I've developed to be able to format a numerical answer. The `\RNGprintf` command uses the Acrobat JavaScript function `util.printf`. The argument of `\RNGprintf` is passed to `util.printf` as its formatting string. Here, we use `\%.4f`, so the number is presented as a floating point number with four decimal places. See the documentation of `util.printf` in the *JavaScript for Acrobat API Reference*.²

²http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp

The final example is the one seen in ‘Introduction’ on page 3, it uses another new command, `\defineDepQJS`. This command is used to define a new RV as a rational function of other RVs, and to define special JavaScript formatting, `\js`. The results of this command are used exclusively for JavaScript, and are not meant to be typeset.

```
\defineDepQJS{\langle name \rangle}{\langle numer \rangle}{\langle denom \rangle}{\langle script \rangle}
```

This function defines `\langle name \rangle` to be $(\langle numer \rangle)/(\langle denom \rangle)$. The expression `\langle numer \rangle` and `\langle denom \rangle` can be functions of RV defined earlier. The 4th argument is JavaScript for evaluating `\langle name \rangle`; `\langle script \rangle` is accessed by `\js\langle name \rangle` and is used from within a `\CorrAnsButton`. The 4th argument can include such commands as `\langle name \rangle`, `\nOf\langle name \rangle`, and `\dOf\langle name \rangle`. An example will perhaps illustrate.

Example 3. Find the equation of the line that passes through P and Q .

We begin by defining our variables:

```
1 \RandomZ{\a}{-10}{9}\RandomZ{\b}{-10}{9}
2 \RandomZ{\c}{\a*}{10}\RandomZ{\d}{\b*}{10}
3 \defineDepQJS{\m}{\d - \b}{\c - \a}
4   {rFrac(rEval(\nOf\m)/rEval(\dOf\m))}
5 \defineDepQJS{\yIntercept}{\b - \a*\m}{1}
6   {rFrac((rEval(\b*\dOf\m-\a*\nOf\m))/(rEval(\dOf\m)))}
```

Our big problem is to compute the slope of the line, `\m`. I define `\m` as using `\defineDepQJS`. The numerator and denominator are those in the slope calculation, given two points. The expansion of `\m` is $(\langle d \rangle - \langle b \rangle)/(\langle c \rangle - \langle a \rangle)$, and the JavaScript will perform the arithmetic operations. The expression that is accessed with the `\js` is the fourth argument, line (4); here, we calculate slope as a rational number. We make a similar definition for the `\yIntercept` of the line.

Below are the two points $P(\langle a \rangle, \langle b \rangle)$ and $Q(\langle c \rangle, \langle d \rangle)$.

3. $P(8, 3), Q(9, 5)$:

The question is posed using `\RespBoxMath`.

```
1 $P(\, \langle a \rangle, \langle b \rangle, )$, $Q(\, \langle c \rangle, \langle d \rangle, )$:
2   \RespBoxMath{y=\m*x + \yIntercept}(xy)
3   {3}{.0001}{{0,2}x[0,2]}*{ProcRespEq}
```

The answer is given in line (2), and will be evaluated numerically, and compared numerically with the user's response.

The code for the correct answer button has a new element in it

```
1 \CorrAnsButton{y = \js\m\space x + \js\yIntercept}
2   *{rngCorrAnsButton}
```

The display of the answer is done using `\js\m` and `\js\yIntercept` to represent the slope and intercept as a rational number.

Think of `\defineDepQJS` a convenient way of defining (JavaScript) expressions that will appear in `\RespBoxMath` and for `\CorrAnsButton`.

7.2. Creating Solutions to Random Quizzes

Writing a solution to a question that is based on a formula or template can be difficult. \LaTeX is not a computer algebra system, so the possibilities are limited. Still, `rangen` supplies the `writeRVsTo` environment to support a solution.

Example 4. We create two RVs, `\a` and `\b` that are rational numbers. We want to add them, and present the answer as a rational number.

```
\begin{writeRVsTo}{quizzes}
\RandomQ{\a}[16]{1/8}{15/16}\RandomQ[ne=\a]{\b}[16]{1/8}{15/16}
\end{writeRVsTo}
```

We make that same definitions as in **Example 1**, but we enclose these definitions within the `writeRVsTo`. This environment writes its contents to the quiz solutions file, and also executes its contents. This way, the definitions are made both here, and just before the solution to this problem in the solutions file.

$$4. \frac{7}{16} + \frac{3}{16} =$$

The verbatim listing of this quiz is

```
\item $\displaystyle\ds\ a + \ds\ b =
\RespBoxMath[\rectW{.5in}]{
  (\noF\a*\dOf\b+\noF\b*\dOf\a)/(\dOf\a*\dOf\b)}*{2}
  {.0001}{[0,2]}[{\priorParse: \Array(nodect,NoAddOrSub)}]$\hfill
\CorrAnsButton{rFrac(rEval(
  \noF\a * \dOf\b + \noF\b * \dOf\a)/rEval(\dOf\a * \dOf\b))}
```

```

}*{rngCorrAnsButton}\kern1bp\sqrClearButton
\begin{solution}\relax\RNAdd\a\b\defineQ{\ans}{\rfNumer}{\rfDenom}%
The solution to this problem is
\begin{equation*}
\boxed{\ds a - \ds b = \ds ans}
\end{equation*}
Did I forget to tell you that a simple command \cs{RNAdd}
for adding two rational numbers is defined by
\pkg{rangen}. Sorry about that! \dps
\end{solution}

```

The `writeRVsTo` has the following syntax

```

\begin{writeRVsTo}{quizzes|exercises}
  \langle rangen commands creating RVs \rangle
\end{writeRVsTo}

```

The argument can be either the string `quizzes` or `exercises`. In the first case, the content of the environment is written to the solutions file for quizzes, and in the latter case, to the solutions file for the exercises.

That's all for now, I simply must get back to my retirement. ~~DS~~

Solutions to Quizzes

Solution to Quiz: The solution to this problem is

$$\frac{7}{16} + \frac{3}{16} = \frac{5}{8}$$

Did I forget to tell you that a simple command `\RNGadd` for adding two rational numbers is defined by `rangen`. Sorry about that! ~~DS~~ ■