

# Package: gridsampler (via r-universe)

November 3, 2024

**License** GPL-3

**Title** A Simulation Tool to Determine the Required Sample Size for  
Repertory Grid Studies

**Type** Package

**LazyLoad** yes

**Description** Simulation tool to facilitate determination of required  
sample size to achieve category saturation for studies using  
multiple repertory grids in conjunction with content analysis.

**Version** 0.6

**Date** 2016-11-23

**Imports** shiny, ggplot2, reshape2, plyr, shinythemes, BiasedUrn,  
shinyBS

**Suggests** knitr, testthat, rmarkdown

**Encoding** UTF-8

**URL** <https://github.com/markheckmann/gridsampler>

**BugReports** <https://github.com/markheckmann/gridsampler/issues>

**VignetteBuilder** knitr

**RoxygenNote** 5.0.1

**Repository** <https://markheckmann.r-universe.dev>

**RemoteUrl** <https://github.com/markheckmann/gridsampler>

**RemoteRef** HEAD

**RemoteSha** d2ad5fe81e2c34a7f31a8f62a5162e893d5987a5

## Contents

gridsampler-package . . . . .	2
calc_probabilities . . . . .	2
draw_multiple_n_persons_x_times . . . . .	3
draw_n_person_sample . . . . .	4

expected_frequencies . . . . .	4
gridsampler . . . . .	5
prob_categories . . . . .	5
sim_n_persons . . . . .	6
sim_n_persons_x_times . . . . .	7
sim_n_persons_x_times_many_n . . . . .	7
sim_one_person . . . . .	8

## Index 10

---

gridsampler-package    **gridsampler** - *A sample size simulation software for repertory grid studies*

---

### Description

**gridsampler** - A sample size simulation software for repertory grid studies

### References

- Green, B. (2004). Personal construct psychology and content analysis. *Personal Construct Theory & Practice*, 1(3), 82-91.
- Jankowicz, D. (2004). *The easy guide to repertory grids*. Chichester, England: John Wiley & Sons.

---

calc\_probabilities    *Probability for certain degree of saturation*

---

### Description

Calculate probability for getting certain proportion of categories with at least m constructs

### Usage

```
calc_probabilities(r, n, ms, min.props = c(0.9, 0.95, 0.99))
```

### Arguments

r	A dataframe. The result returned from <a href="#">sim_n_persons_x_times_many_n</a> .
n	Vector of n for which to calculate probabilities.
ms	minimal number of constructs in each category
min.props	Proportion of categories to contain at least m constructs.

### See Also

Other Utilities: [expected\\_frequencies](#), [prob\\_categories](#)

### Examples

```
prob <- dexp(1:30, .05)
n <- seq(10, 80, by = 20)
r <- sim_n_persons_x_times_many_n(prob, n, a = 7, times = 100)
dd <- calc_probabilities(r, n, ms=1:5, min.props = c(0.9, .95, 1))
head(dd)
```

---

draw\_multiple\_n\_persons\_x\_times

*Draw and redraw results of simulation*

---

### Description

Draw and redraw results of simulation

### Usage

```
draw_multiple_n_persons_x_times(d)
```

### Arguments

d                    A dataframe as returned by [calc\\_probabilities](#).

### See Also

Other Plotting: [draw\\_n\\_person\\_sample](#)

### Examples

```
## simulate
prob <- dexp(1:30, .05)        # probabilities for categories
N <- seq(10, 80, by = 10)    # sample sizes to simulate
r <- sim_n_persons_x_times_many_n(prob, n = N, a = 7, times = 100, progress = "none")

# calculate and draw
M <- 1:5                    # minimal number of categories to evaluate
p <- c(0.9, .95, 1)        # proportion of categories for which minimal m holds
d <- calc_probabilities(r, n = N, ms = M, min.props = p)
draw_multiple_n_persons_x_times(d)
```

`draw_n_person_sample` *Produce graphic for a single sample of n persons*

---

### Description

Produce graphic for a single sample of n persons

### Usage

```
draw_n_person_sample(prob, n, a = 10, ap = rep(1/length(a), length(a)))
```

### Arguments

<code>prob</code>	Probability to draw a construct from a certain category.
<code>n</code>	Number of persons, i.e. grids to be sampled.
<code>a</code>	Possible number of attributes sampled from.
<code>ap</code>	Attribute probabilities, i.e. for each number of attributes given in a.

### See Also

Other Plotting: [draw\\_multiple\\_n\\_persons\\_x\\_times](#)

### Examples

```
draw_n_person_sample(dexp(1:30, rate = .05), n = 100, a = 10)
draw_n_person_sample(dexp(1:30, rate = .05), n = 100, a = 1:5, ap = 5:1)
```

---

`expected_frequencies` *Produce ggplot of percentiles for simulated frequencies*

---

### Description

Produce ggplot of percentiles for simulated frequencies

### Usage

```
expected_frequencies(r)
```

### Arguments

<code>r</code>	A dataframe. The result returned from <a href="#">sim_n_persons_x_times</a> .
----------------	---

### Value

Draws a ggplot

**See Also**

Other Utilities: [calc\\_probabilities](#), [prob\\_categories](#)

**Examples**

```
r <- sim_n_persons_x_times(dexp(1:30, rate = .05), n = 50, a = 5:7, ap = 1:3, 100)
expected_frequencies(r)
```

---

gridsampler	<i>Run gridsampler app</i>
-------------	----------------------------

---

**Description**

This function starts the gridsampler shiny app.

**Usage**

```
gridsampler(display.mode = "auto",
            launch.browser = getOption("shiny.launch.browser", interactive()))
```

**Arguments**

`display.mode` auto by default, can also be showcase. See [runApp](#).

`launch.browser` Boolean, set TRUE to open the app in the browser. See [runApp](#).

**Examples**

```
## Not run:
gridsampler()

## End(Not run)
```

---

prob_categories	<i>Probability for certain degree of saturation</i>
-----------------	---

---

**Description**

Calculate probability for getting certain proportion of categories with at least m constructs

**Usage**

```
prob_categories(r, m, min.prop = 1)
```

**Arguments**

r	A dataframe. The result returned from <a href="#">sim_n_persons_x_times</a> .
m	minimal number of constructs in each category
min.prop	Proportion of categories to contain at least m constructs.

**See Also**

Other Utilities: [calc\\_probabilities](#), [expected\\_frequencies](#)

**Examples**

```
r <- sim_n_persons_x_times(dexp(1:30, rate = .05), n = 50, a = 5:7, times = 100, progress = "none")
prob_categories(r, 4, min.prop = .9)
```

---

sim_n_persons	<i>Simulate n persons</i>
---------------	---------------------------

---

**Description**

Function is a simple replicate wrapper around `sim_one_person`

**Usage**

```
sim_n_persons(prob, n, a = 10, ap = rep(1/length(a), length(a)))
```

**Arguments**

prob	Probability to draw a construct from a certain category.
n	Number of persons, i.e. grids to be sampled.
a	Possible number of attributes sampled from.
ap	Attribute probabilities, i.e. for each number of attributes given in a.

**See Also**

Other Simulations: [sim\\_n\\_persons\\_x\\_times\\_many\\_n](#), [sim\\_n\\_persons\\_x\\_times](#), [sim\\_one\\_person](#)

**Examples**

```
sim_n_persons(dexp(1:30, .05), n = 2, a = 10)
sim_n_persons(dexp(1:30, .05), n = 2, a = c(1, 30))
sim_n_persons(dexp(1:30, .05), n = 2, a = c(1, 30), ap = c(1,4))
sim_n_persons(dexp(1:30, .05), n = 2, a = 1:5, ap = c(1,1,2,2,3))
```

---

sim\_n\_persons\_x\_times *Complete simulation*

---

### Description

Complete simulation

### Usage

```
sim_n_persons_x_times(prob, n, a, ap = rep(1/length(a), length(a)),
  times = 100, progress = "text")
```

### Arguments

prob	Probability to draw a construct from a certain category. Length of vector determines number of categories.
n	Number of persons, i.e. grids to sample.
a	Number of constructs to be sampled.
ap	Probabilities for each number of attributes to be sampled.
times	Number of times to repeat each simulation.
progress	Type of progress bar shown during simulation.

### See Also

Other Simulations: [sim\\_n\\_persons\\_x\\_times\\_many\\_n](#), [sim\\_n\\_persons](#), [sim\\_one\\_person](#)

### Examples

```
## Not run:
sim_n_persons_x_times(dexp(1:30, .05), n = 2, a = c(1,30), ap = 1:2, times = 100)
sim_n_persons_x_times(dexp(1:30, .05), n = 2, a = c(1,30), times = 200, progress = "tk")

## End(Not run)
```

---

sim\_n\_persons\_x\_times\_many\_n  
*Simulate for different n*

---

### Description

Creates simulation results for different n. Runs [sim\\_n\\_persons\\_x\\_times](#) for different n.

### Usage

```
sim_n_persons_x_times_many_n(prob, n = seq(10, 80, by = 10), a = 7,
  ap = rep(1/length(a), length(a)), times = 100, progress = "text")
```

**Arguments**

prob	Probability to draw a construct from a certain category. Length of vector determines number of categories.
n	Number of persons, i.e. grids to sample.
a	Number of constructs to be sampled.
ap	Probabilities for each number of attributes to be sampled.
times	Number of times to repeat each simulation.
progress	Type of progress bar shown during simulation.

**Value**

A result dataframe.

**See Also**

Other Simulations: [sim\\_n\\_persons\\_x\\_times](#), [sim\\_n\\_persons](#), [sim\\_one\\_person](#)

**Examples**

```
## Not run:
r <- sim_n_persons_x_times_many_n(dexp(1:30, .05), a = 7, times = 100)
r <- sim_n_persons_x_times_many_n(dexp(1:30, .05), a = 5:7, ap = 1:3, times = 100)

## End(Not run)
```

---

sim_one_person	<i>Simulate a single grid</i>
----------------	-------------------------------

---

**Description**

Simulate a single grid

**Usage**

```
sim_one_person(prob, a = 10)
```

**Arguments**

prob	Probability to draw a construct from a certain category.
a	Number of constructs to be sampled.

**See Also**

Other Simulations: [sim\\_n\\_persons\\_x\\_times\\_many\\_n](#), [sim\\_n\\_persons\\_x\\_times](#), [sim\\_n\\_persons](#)



**Examples**

```
# draw from exponential distribution  
p <- dexp(1:20, rate = .1)  
sim_one_person(p, a = 10)
```

# Index

- \* **package**
  - gridsampler-package, 2
- \* **reppgrid**
  - gridsampler-package, 2
- calc\_probabilities, 2, 3, 5, 6
- draw\_multiple\_n\_persons\_x\_times, 3, 4
- draw\_n\_person\_sample, 3, 4
- expected\_frequencies, 2, 4, 6
- gridsampler, 5
- gridsampler-package, 2
- prob\_categories, 2, 5, 5
- runApp, 5
- sim\_n\_persons, 6, 7, 8
- sim\_n\_persons\_x\_times, 4, 6, 7, 7, 8
- sim\_n\_persons\_x\_times\_many\_n, 2, 6, 7, 7,  
8
- sim\_one\_person, 6–8, 8