What do you assume?

HYPOTHESIS TESTING IN R

Richie Cotton
Data Evangelist at DataCamp

Randomness

Assumption

> The samples are random subsets of larger populations.

Consequence

- Sample is not representative of population.

How to check this

- Understand how your data was collected.
- Speak to the data collector/domain expert.

[1] Sampling techniques are discussed in "Sampling in R".

Independence of observations

Assumption

> Each observation (row) in the dataset is independent.

Consequence

- Increased chance of false negative/positive error.

How to check this

- Understand how your data was collected.

Large sample size

Assumption

> The sample is big enough to mitigate uncertainty, and so that the Central Limit Theorem applies.

Consequence

- Really wide confidence intervals.
- Increased chance of false negative/positive error.

How to check this

- It depends on the test.

Large sample size: t-test

One sample

- At least 30[1] observations in the sample.

$n \geq 30$

n: sample size

Two samples

- At least 30 observations in each sample.

$n_1 \geq 30, n_2 \geq 30$

n_i: sample size for group i

Paired samples

- At least 30 pairs of observations across the samples.

Number of rows in your data ≥ 30

ANOVA

- At least pairs of 30 observations in each sample.

$n_i \geq 30$ for all values of i

[1] Sometimes you can get away with less than 30; the important thing is that the null distribution appears normal.

Large sample size: proportion tests

One sample

- Number of successes in sample is greater than or equal to 10.

$$n \times \hat{p} \geq 10$$

- Number of failures in sample is greater than or equal to 10.

$$n \times (1 - \hat{p}) \geq 10$$

n: sample size

$\hat{p}$: proportion of successes in sample

Two samples

- Number of successes in each sample is greater than or equal to 10.

$$n_1 \times \hat{p}_1 \geq 10$$

$$n_2 \times \hat{p}_2 \geq 10$$

- Number of failures in each sample is greater than or equal to 10.

$$n_1 \times (1 - \hat{p}_1) \geq 10$$

$$n_2 \times (1 - \hat{p}_2) \geq 10$$

Large sample size: chi-square tests

- The number of successes in each group in greater than or equal to 5.

$n_i \times \hat{p}_i \geq 5$ for all values of i

- The number of failures in each group in greater than or equal to 5.

$n_i \times (1 - \hat{p}_i) \geq 5$ for all values of i

n_i: sample size for group i
$\hat{p}_i$: proportion of successes in sample group i

Sanity check

> If the bootstrap distribution doesn't look normal, assumptions likely aren't valid.

Let's practice!

HYPOTHESIS TESTING IN R

Strength in numbers

HYPOTHESIS TESTING IN R

Richie Cotton
Data Evangelist at DataCamp

Imbalanced data

```
stack_overflow_imbalanced %>%
  count(hobbyist, age_cat, .drop = FALSE)
```

```
  hobbyist     age_cat    n
1       No At least 30    0
2       No    Under 30  191
3      Yes At least 30   15
4      Yes    Under 30 1025
```

A sample is *imbalanced* if some groups are much bigger than others.

Hypotheses

H_0: The proportion of hobbyists under 30 is **the same as** the proportion of hobbyists at least 30.

H_A: The proportion of hobbyists under 30 is **different from** the proportion of hobbyists at least 30.

```
alpha <- 0.1
```

Proceeding with a proportion test regardless

```
stack_overflow_imbalanced %>%
  prop_test(
    hobbyist ~ age_cat,
    order = c("At least 30", "Under 30"),
    success = "Yes",
    alternative = "two.sided",
    correct = FALSE
  )
```

```
# A tibble: 1 x 6
  statistic chisq_df p_value alternative lower_ci upper_ci
      <dbl>    <dbl>   <dbl> <chr>          <dbl>    <dbl>
1      2.79        1  0.0949 two.sided    0.00718   0.0217
```

A grammar of graphics

Plot type	base-R	ggplot2
Scatter plot	`plot(, type = "p")`	`ggplot() + geom_point()`
Line plot	`plot(, type = "l")`	`ggplot() + geom_line()`
Histogram	`hist()`	`ggplot() + geom_histogram()`
Box plot	`boxplot()`	`ggplot() + geom_boxplot()`
Bar plot	`barplot()`	`ggplot() + geom_bar()`
Pie plot	`pie()`	`ggplot() + geom_bar() + coord_polar()`

A grammar of hypothesis tests

- Allen Downey's **There is only one test** framework.
- Implemented in R in the `infer` package.
- `generate()` makes simulated data.
  - Computationally expensive.
  - Robust against small samples or imbalanced data.

```
null_distn <- dataset %>%
  specify() %>%
  hypothesize() %>%
  generate() %>%
  calculate()
```

```
obs_stat <- dataset %>%
  specify() %>%
  calculate()
```

```
get_p_value(null_distn, obs_stat)
```

[1] Allen Downey teaches "Exploratory Data Analysis in Python".

Specifying the variables of interest

specify()

> `specify()` selects the variable(s) you want to test.

- For 2 sample tests, use `response ~ explanatory`.
- For 1 sample tests use `response ~ NULL`.

```
stack_overflow_imbalanced %>%
  specify(hobbyist ~ age_cat, success = "Yes")
```

```
Response: hobbyist (factor)
Explanatory: age_cat (factor)
# A tibble: 1,231 x 2
  hobbyist age_cat
  <fct>    <fct>
1 Yes      At least 30
2 Yes      At least 30
3 Yes      At least 30
4 Yes      Under 30
5 Yes      At least 30
6 Yes      At least 30
7 No       Under 30
# ... with 1,224 more rows
```

hypothesize()

> `hypothesize()` declares the type of null hypothesis.

- For 2 sample tests, use `"independence"` or `"point"`.
- For 1 sample tests, use `"point"`.

```
stack_overflow_imbalanced %>%
  specify(hobbyist ~ age_cat, success = "Yes") %>%
  hypothesize(null = "independence")
```

```
Response: hobbyist (factor)
Explanatory: age_cat (factor)
Null Hypothesis: independence
# A tibble: 1,231 x 2
  hobbyist age_cat
  <fct>    <fct>
1 Yes      At least 30
2 Yes      At least 30
3 Yes      At least 30
4 Yes      Under 30
5 Yes      At least 30
6 Yes      At least 30
7 No       Under 30
# ... with 1,224 more rows
```

Let's practice!

HYPOTHESIS TESTING IN R

Strength in numbers

HYPOTHESIS TESTING IN R

Richie Cotton
Data Evangelist at DataCamp

Recap: hypotheses and dataset

H_0: The proportion of hobbyists under 30 is the same as the prop'n of hobbyists at least 30.

H_A: The proportion of hobbyists under 30 is different from the prop'n of hobbyists at least 30.

```
alpha <- 0.1
```

```
stack_overflow_imbalanced %>%
  count(hobbyist, age_cat, .drop = FALSE)
```

```
  hobbyist     age_cat    n
1       No At least 30    0
2       No    Under 30  191
3      Yes At least 30   15
4      Yes    Under 30 1025
```

Recap: workflow

```
null_distn <- dataset %>%
  specify() %>%
  hypothesize() %>%
  generate() %>%
  calculate()
```

```
observed_stat <- dataset %>%
  specify() %>%
  calculate()
```

```
get_p_value(null_distn, observed_stat)
```

```
stack_overflow_imbalanced %>%
  specify(hobbyist ~ age_cat, success = "Yes") %>%
  hypothesize(null = "independence")
```

```
Response: hobbyist (factor)
Explanatory: age_cat (factor)
Null Hypothesis: independence
# A tibble: 1,231 x 2
  hobbyist age_cat
  <fct>    <fct>
1 Yes      At least 30
2 Yes      At least 30
3 Yes      At least 30
4 Yes      Under 30
5 Yes      At least 30
6 Yes      At least 30
7 No       Under 30
# ... with 1,224 more rows
```

Motivating generate()

H_0: The proportion of hobbyists under 30 is the same as the prop'n of hobbyists at least 30.

If H_0 is true, then

- In each row, the hobbyist value could have appeared with either age category with equal probability.
- To simulate this, we can permute (shuffle) the hobbyist values while keeping the age categories fixed.

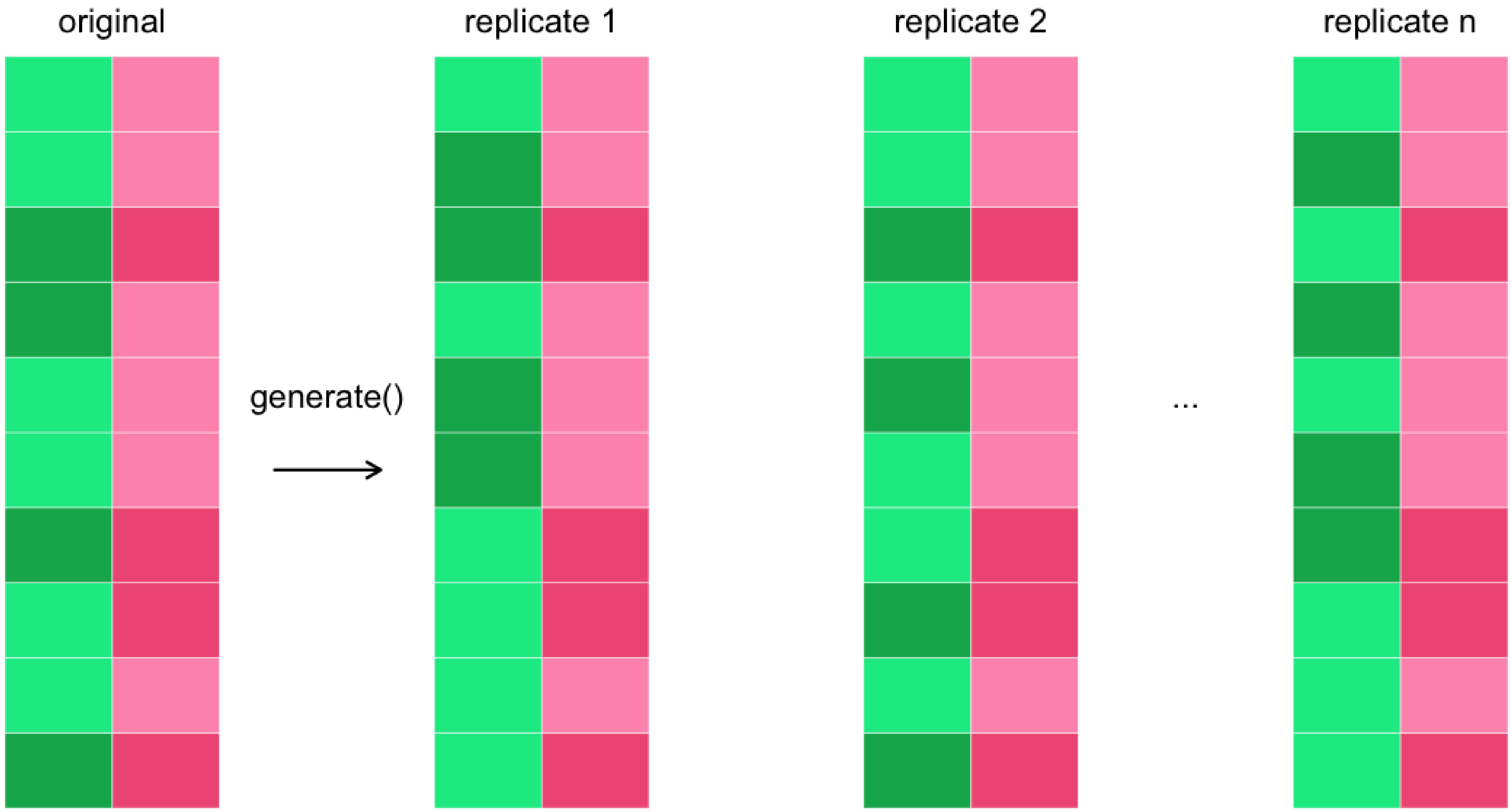
```r
stack_overflow_imbalanced
```

```
# A tibble: 1,231 x 2
  hobbyist age_cat
  <fct>    <fct>
1 Yes      At least 30
2 Yes      At least 30
3 Yes      At least 30
4 Yes      Under 30
5 Yes      At least 30
6 Yes      At least 30
7 No       Under 30
# ... with 1,224 more rows
```

```r
bind_cols(
  stack_overflow_imbalanced %>%
    select(hobbyist) %>%
    slice_sample(prop = 1),
  stack_overflow_imbalanced %>%
    select(age_cat)
)
```

```
# A tibble: 1,231 x 2
  hobbyist age_cat
  <fct>    <fct>
1 Yes      At least 30
2 Yes      At least 30
3 No       At least 30
4 No       Under 30
5 Yes      At least 30
6 Yes      At least 30
7 Yes      Under 30
# ... with 1,224 more rows
```

Generating many replicates



generate()

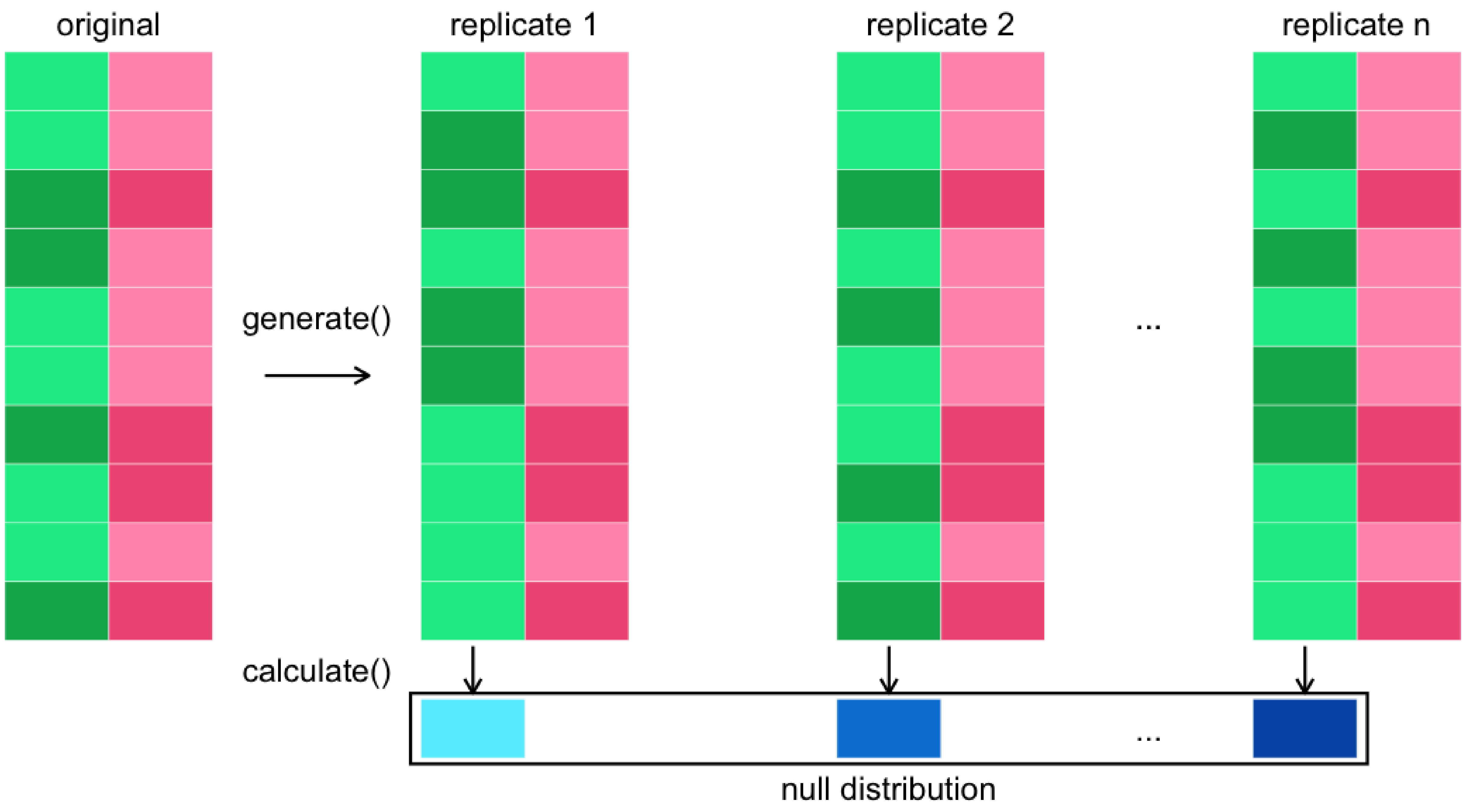
> `generate()` generates simulated data reflecting the null hypothesis.

- For "independence" null hypotheses, set `type` to `"permute"`.
- For "point" null hypotheses, set `type` to `"bootstrap"` or `"simulate"`.

```
stack_overflow_imbalanced %>%
  specify(hobbyist ~ age_cat, success = "Yes") %>%
  hypothesize(null = "independence") %>%
  generate(reps = 5000, type = "permute")
```

```
Response: hobbyist (factor)
Explanatory: age_cat (factor)
Null Hypothesis: independence
# A tibble: 6,155,000 x 3
# Groups:   replicate [5,000]
  hobbyist age_cat     replicate
  <fct>    <fct>           <int>
1 Yes      At least 30         1
2 Yes      At least 30         1
3 Yes      At least 30         1
4 Yes      Under 30            1
5 Yes      At least 30         1
6 Yes      At least 30         1
7 Yes      Under 30            1
# ... with 6,154,993 more rows
```

Calculating the test statistic



calculate()

> `calculate()` calculates a distribution of test statistics known as the *null distribution*.

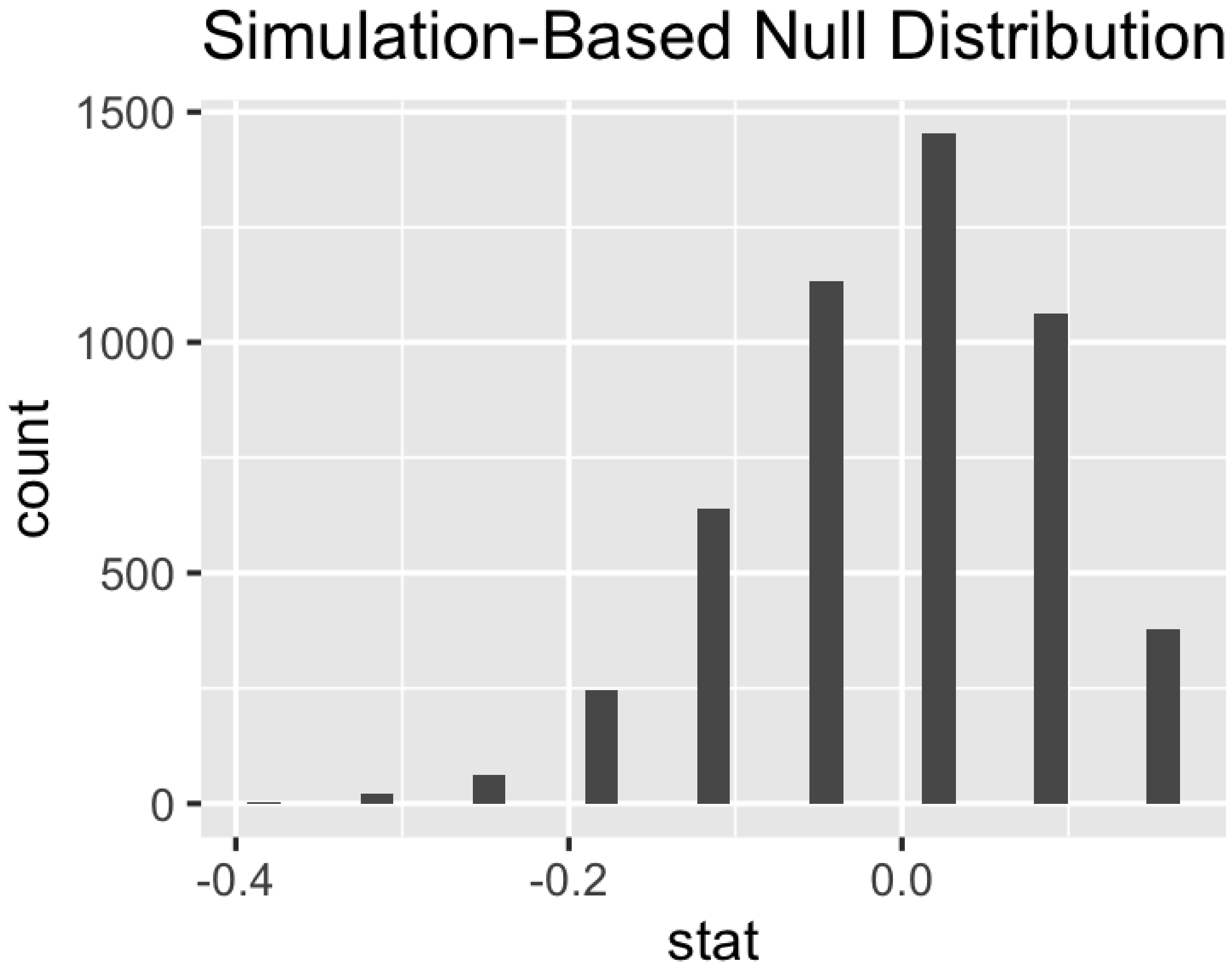
```
null_distn <- stack_overflow_imbalanced %>%
  specify(
    hobbyist ~ age_cat,
    success = "Yes"
  ) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 5000, type = "permute") %>%
  calculate(
    stat = "diff in props",
    order = c("At least 30", "Under 30")
  )
```

```
# A tibble: 5,000 x 2
  replicate    stat
      <int>   <dbl>
1         1  0.0896
2         2  0.0896
3         3 -0.180
4         4  0.157
5         5  0.0896
6         6 -0.113
7         7  0.0221
# ... with 4,993 more rows
```

[1] The ?calculate help page lists all possible test statistics.

Visualizing the null distribution

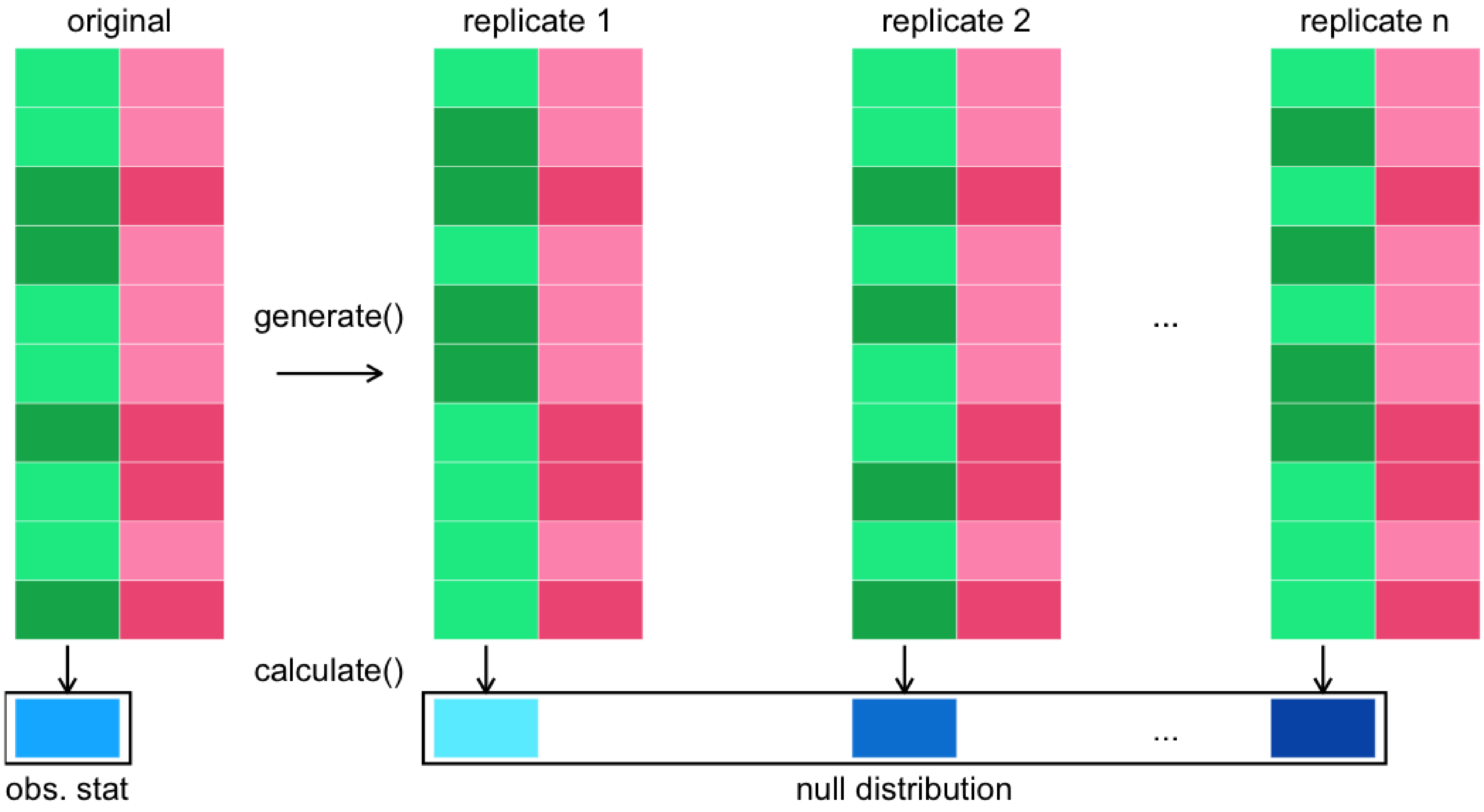
```
visualize(null_distn)
```




```
null_distn %>% count(stat)
```

```
# A tibble: 9 x 2
     stat     n
    <dbl> <int>
1 -0.383      2
2 -0.315     22
3 -0.248     63
4 -0.180    246
5 -0.113    641
6 -0.0454  1132
7  0.0221  1453
8  0.0896  1063
9  0.157    378
```

Calculating the test statistic on the original dataset



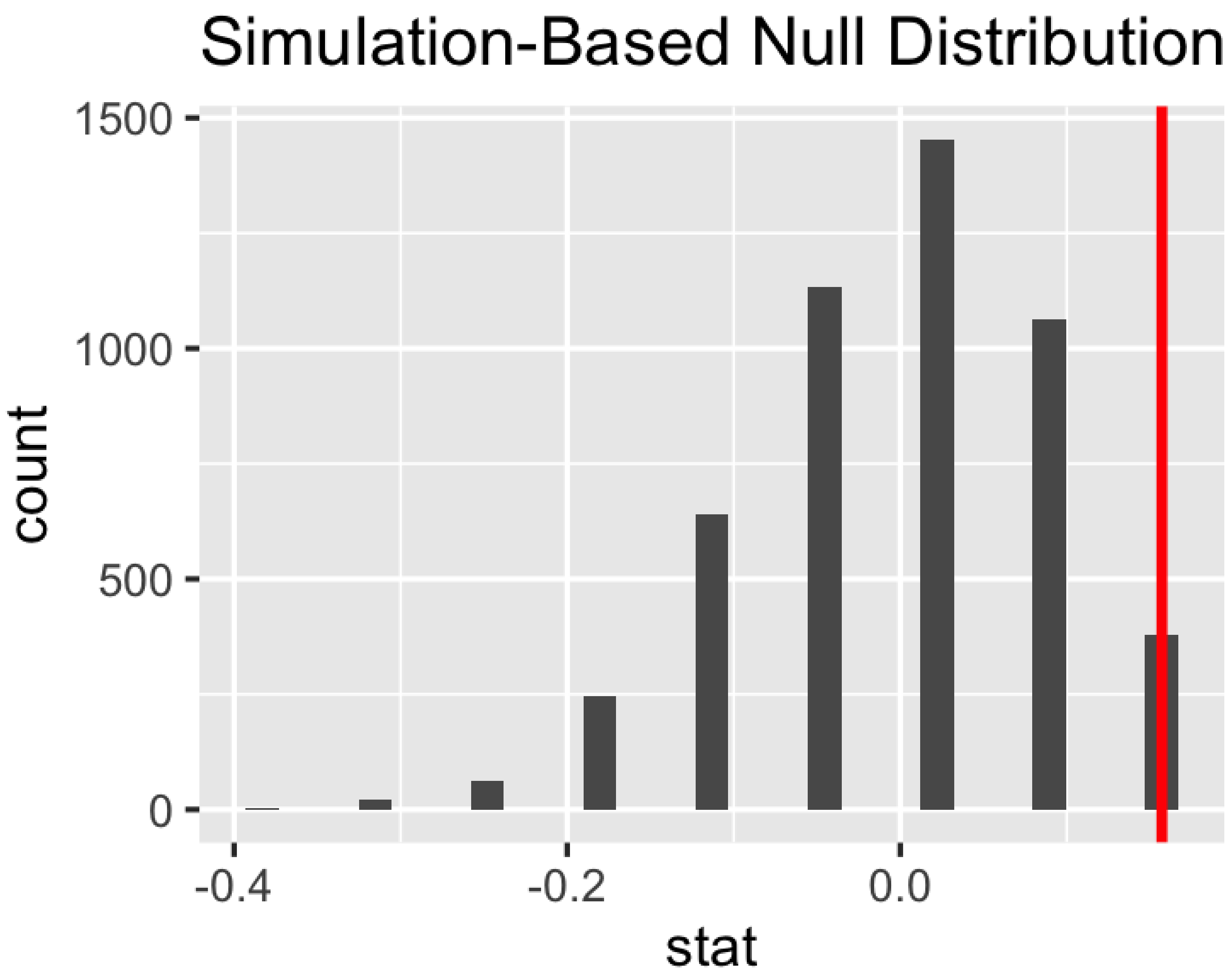
Observed statistic: specify() %>% calculate()

```r
obs_stat <- stack_overflow_imbalanced %>%
  specify(hobbyist ~ age_cat, success = "Yes") %>%
  # hypothesize(null = "independence") %>%
  # generate(reps = 5000, type = "permute") %>%
  calculate(
    stat = "diff in props",
    order = c("At least 30", "Under 30")
  )
```

```
# A tibble: 1 x 1
   stat
  <dbl>
1 0.157
```

Visualizing the null distribution vs the observed stat

```
visualize(null_distn) +
  geom_vline(
    aes(xintercept = stat),
    data = observed_stat,
    color = "red"
  )
```



Get the p-value

```
get_p_value(
  null_distn, obs_stat,
  direction = "two sided"   # Not alternative = "two.sided"
)
```

```
# A tibble: 1 x 1
  p_value
    <dbl>
1   0.151
```

```
# A tibble: 1 x 6
  statistic chisq_df p_value alternative lower_ci upper_ci
      <dbl>    <dbl>   <dbl> <chr>          <dbl>    <dbl>
1      2.79        1  0.0949 two.sided    0.00718   0.0217
```

Let's practice!

HYPOTHESIS TESTING IN R

Look ma! No parameters!

HYPOTHESIS TESTING IN R

Richie Cotton
Data Evangelist at DataCamp

Non-parametric tests

A *non-parametric test* is a hypothesis test that doesn't assume a probability distribution for the test statistic.

There are two types of non-parametric hypothesis test:

1. Simulation-based.
2. Rank-based.

t_test()

H_0: $\mu_{child} - \mu_{adult} = 0$ $\quad$ H_A: $\mu_{child} - \mu_{adult} > 0$

```
library(infer)
stack_overflow %>%
  t_test(
    converted_comp ~ age_first_code_cut,
    order = c("child", "adult"),
    alternative = "greater"
  )
```

```
# A tibble: 1 x 6
  statistic  t_df p_value alternative lower_ci upper_ci
      <dbl> <dbl>   <dbl> <chr>          <dbl>    <dbl>
1      2.40 2083. 0.00814 greater        8438.      Inf
```

Calculating the null distribution

Simulation-based pipeline

```
null_distn <- stack_overflow %>%
  specify(converted_comp ~ age_first_code_cut) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 5000, type = "permute") %>%
  calculate(
    stat = "diff in means",
    order = c("child", "adult")
  )
```

t-test, for comparison

```
library(infer)
stack_overflow %>%
  t_test(
    converted_comp ~ age_first_code_cut,
    order = c("child", "adult"),
    alternative = "greater"
  )
```

Calculating the observed statistic

Simulation-based pipeline

```
obs_stat <- stack_overflow %>%
  specify(converted_comp ~ age_first_code_cut) %>%
  calculate(
    stat = "diff in means",
    order = c("child", "adult")
  )
```

t-test, for comparison

```
library(infer)
stack_overflow %>%
  t_test(
    converted_comp ~ age_first_code_cut,
    order = c("child", "adult"),
    alternative = "greater"
  )
```

Get the p-value

Simulation-based pipeline

```
get_p_value(
  null_distn, obs_stat,
  direction = "greater"
)
```

```
# A tibble: 1 x 1
  p_value
    <dbl>
1  0.0066
```

t-test, for comparison

```
library(infer)
stack_overflow %>%
  t_test(
    converted_comp ~ age_first_code_cut,
    order = c("child", "adult"),
    alternative = "greater"
  )
```

```
# A tibble: 1 x 6
  statistic  t_df p_value alternative lower_ci upper_ci
      <dbl> <dbl>   <dbl> <chr>          <dbl>    <dbl>
1      2.40 2083. 0.00814 greater        8438.      Inf
```

Ranks of vectors

```
x <- c(1, 15, 3, 10, 6)
```

```
rank(x)
```

```
1 5 2 4 3
```

A *Wilcoxon-Mann-Whitney test* (a.k.a. *Wilcoxon rank sum test*) is (very roughly) a t-test on the ranks of the numeric input.

Wilcoxon-Mann-Whitney test

```r
wilcox.test(
  converted_comp ~ age_first_code_cut,
  data = stack_overflow,
  alternative = "greater",
  correct = FALSE
)
```

```
    Wilcoxon rank sum test

data:  converted_comp by age_first_code_cut
W = 967298, p-value <2e-16
alternative hypothesis: true location shift is greater than 0
```

[1] Also known as the "Wilcoxon rank-sum test" and the "Mann-Whitney U test".

Kruskal-Wallis test

Kruskal-Wallis test is to *Wilcoxon-Mann-Whitney test* as *ANOVA* is to *t-test*.

```
kruskal.test(
  converted_comp ~ job_sat,
  data = stack_overflow
)
```

```
    Kruskal-Wallis rank sum test


data:  converted_comp by job_sat
Kruskal-Wallis chi-square = 81, df = 4, p-value <2e-16
```

Let's practice!

HYPOTHESIS TESTING IN R

Congratulations

HYPOTHESIS TESTING IN R

Richie Cotton
Data Evangelist at DataCamp

You learned things

Chapter 1

- Workflow for testing proportions vs. a hypothesized value.
- False negative/false positive errors.

Chapter 2

- Testing differences in sample means between two groups using t-tests.
- Extending this to more than two groups using ANOVA and pairwise t-tests.

Chapter 3

- Testing differences in sample proportions between two groups using proportion tests.
- Using chi-square independence/goodness of fit tests.

Chapter 4

- Reviewing assumptions of parametric hypothesis tests.
- Examined nonparametric alternatives when assumptions aren't valid

More courses

Inference

Statistical Inference with R skill track

Bayesian statistics

Fundamentals of Bayesian Data Analysis in R

Applications

A/B Testing in R

Let's practice!

HYPOTHESIS TESTING IN R