

---

## Field Test

:maple\_leaf: A/B testing for Rails

- Designed for web and email
- Comes with a dashboard to view results and update variants
- Uses your database for storage
- Seamlessly handles the transition from anonymous visitor to logged in user

Uses Bayesian statistics to evaluate results so you don't need to choose a sample size ahead of time.



## Installation

Add this line to your application's Gemfile:

```
1 gem "field_test"
```

Run:

```
1 rails generate field_test:install
2 rails db:migrate
```

And mount the dashboard in your `config/routes.rb`:

```
1 mount FieldTest::Engine, at: "field_test"
```

Be sure to secure the dashboard in production.

## Getting Started

Add an experiment to `config/field_test.yml`.

```
1 experiments:
2   button_color:
3     variants:
4       - red
5       - green
6       - blue
```

Refer to it in controllers, views, and mailers.

```
1 button_color = field_test(:button_color)
```

---

To make testing easier, you can specify a variant with query parameters

```
1 http://localhost:3000/?field_test[button_color]=green
```

When someone converts, record it with:

```
1 field_test_converted(:button_color)
```

When an experiment is over, specify a winner:

```
1 experiments:
2   button_color:
3     winner: green
```

All calls to `field_test` will now return the winner, and metrics will stop being recorded.

You can keep returning the variant for existing participants after a winner is declared:

```
1 experiments:
2   button_color:
3     winner: green
4     keep_variant: true
```

You can also close an experiment to new participants without declaring a winner while still recording metrics for existing participants:

```
1 experiments:
2   button_color:
3     closed: true
```

Calls to `field_test` for new participants will return the control, and they won't be added to the experiment.

You can get the list of experiments and variants for a user with:

```
1 field_test_experiments
```

## JavaScript and Native Apps

For JavaScript and native apps, add calls to your normal endpoints.

```
1 class CheckoutController < ActionController::API
2   def start
3     render json: {button_color: field_test(:button_color)}
4   end
5
6   def finish
7     field_test_converted(:button_color)
```

---

```
8     # ...
9     end
10  end
```

For anonymous visitors in native apps, pass a `Field-Test-Visitor` header with a unique identifier.

## Participants

Any model or string can be a participant in an experiment.

For web requests, it uses `current_user` (if it exists) and an anonymous visitor id to determine the participant. Set your own with:

```
1 class ApplicationController < ActionController::Base
2   def field_test_participant
3     current_company
4   end
5 end
```

For mailers, it tries `@user` then `params[:user]` to determine the participant. Set your own with:

```
1 class ApplicationMailer < ActionMailer::Base
2   def field_test_participant
3     @company
4   end
5 end
```

You can also manually pass a participant with:

```
1 field_test(:button_color, participant: company)
```

## Jobs

To get variants in jobs, models, and other contexts, use:

```
1 experiment = FieldTest::Experiment.find(:button_color)
2 button_color = experiment.variant(user)
```

## Exclusions

By default, bots are returned the first variant and excluded from metrics. Change this with:

---

```
1 exclude:
2   bots: false
```

Exclude certain IP addresses with:

```
1 exclude:
2   ips:
3     - 127.0.0.1
4     - 10.0.0.0/8
```

You can also use custom logic:

```
1 field_test(:button_color, exclude: request.user_agent == "Test")
```

## Config

Keep track of when experiments started and ended. Use any format `Time.parse` accepts. Variants assigned outside this window are not included in metrics.

```
1 experiments:
2   button_color:
3     started_at: Dec 1, 2016 8 am PST
4     ended_at: Dec 8, 2016 2 pm PST
```

Add a friendlier name and description with:

```
1 experiments:
2   button_color:
3     name: Buttons!
4     description: >
5       Different button colors
6       for the landing page.
```

By default, variants are given the same probability of being selected. Change this with:

```
1 experiments:
2   button_color:
3     variants:
4     - red
5     - blue
6     weights:
7     - 85
8     - 15
```

To help with GDPR compliance, you can switch from cookies to anonymity sets for anonymous visitors. Visitors with the same IP mask and user agent are grouped together.

---

```
1 cookies: false
```

## Dashboard Config

If the dashboard gets slow, you can make it faster with:

```
1 cache: true
```

This will use the Rails cache to speed up winning probability calculations.

If you need more precision, set:

```
1 precision: 1
```

## Multiple Goals

You can set multiple goals for an experiment to track conversions at different parts of the funnel. First, run:

```
1 rails generate field_test:events
2 rails db:migrate
```

And add to your config:

```
1 experiments:
2   button_color:
3     goals:
4       - signed_up
5       - ordered
```

Specify a goal during conversion with:

```
1 field_test_converted(:button_color, goal: "ordered")
```

The results for all goals will appear on the dashboard.

## Analytics Platforms

You may also want to send experiment data as properties to other analytics platforms like Segment, Amplitude, and Ahoy. Get the list of experiments and variants with:

```
1 field_test_experiments
```

---

## Ahoy

You can configure Field Test to use Ahoy's visitor token instead of creating its own:

```
1 class ApplicationController < ActionController::Base
2   def field_test_participant
3     [ahoy.user, ahoy.visitor_token]
4   end
5 end
```

## Dashboard Security

### Devise

```
1 authenticate :user, ->(user) { user.admin? } do
2   mount FieldTest::Engine, at: "field_test"
3 end
```

**Basic Authentication** Set the following variables in your environment or an initializer.

```
1 ENV["FIELD_TEST_USERNAME"] = "moonrise"
2 ENV["FIELD_TEST_PASSWORD"] = "kingdom"
```

## Updating Variants

Assign a specific variant to a user with:

```
1 experiment = FieldTest::Experiment.find(:button_color)
2 experiment.variant(participant, variant: "green")
```

You can also change a user's variant from the dashboard.

## Associations

To associate models with field test memberships, use:

```
1 class User < ApplicationRecord
2   has_many :field_test_memberships, class_name: "FieldTest::Membership"
3   , as: :participant
4 end
```

Now you can do:

```
1 user.field_test_memberships
```

---

## Credits

A huge thanks to Evan Miller for deriving the Bayesian formulas.

## History

[View the changelog](#)

## Contributing

Everyone is encouraged to help improve this project. Here are a few ways you can help:

- Report bugs
- Fix bugs and submit pull requests
- Write, clarify, or fix documentation
- Suggest or add new features

To get started with development:

```
1 git clone https://github.com/ankane/field_test.git
2 cd field_test
3 bundle install
4 bundle exec rake compile
5 bundle exec rake test
```