skpr
R Package for Generation and Evaluation of Experimental Designs

Tyler Morgan-Wall
Outline

1. Demo
2. Introduction
3. Features of skpr
4. skprGUI
Live Demo
Outline

1. Demo
2. Introduction
3. Features of skpr
4. skprGUI
Motivation for skpr

• Create an open-source design of experiments package to allow for the creation and analysis of designs all within R.
• Implement optimal design algorithms using modern linear algebra libraries to make the underlying code simple, reliable, and auditable.
• Integrate Monte Carlo power evaluation as a standard feature and make it flexible, extensible, and easy to use.
Plan your experiment using the same models and tools that you plan to use when analyzing them.
Shareable and repeatable

With skpr, there is never a question about *how* you ran an analysis—it's all there in the script.

Reproducing your work is as simple as re-running the code.
Outline

1. Demo
2. Introduction
3. Features of skpr
4. skprGUI
Five main functions + plotting

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
- **eval_design_survival_mc**: Censored Monte Carlo power evaluation
- **eval_design_custom_mc**: Custom library Monte Carlo power evaluation
- **plot_correlations & plot_fds**: Plot correlation map and fraction of design space plots
Five main functions + plotting

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
- **eval_design_survival_mc**: Censored Monte Carlo power evaluation
- **eval_design_custom_mc**: Custom library Monte Carlo power evaluation
- **plot_correlations & plot_fds**: Plot correlation map and fraction of design space plots
Workflow in skpr

Step 1 → Step 2 → Step 3

Generate test points

gen_design

eval_design

eval_design_mc

eval_design_survival_mc

eval_design_custom_mc
Example: How much caffeine extracted from coffee?

Factors:

**Bean size**: Large, Small

**Bean type**: Kona, Java

**Water temperature**: 80°C, 90°C, 100°C

**Roast Darkness (hard to change)**: Light, Medium, Dark
Main functions in skpr

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
gen_design – Basic Functionality

**Candidate Set**

<table>
<thead>
<tr>
<th>size</th>
<th>type</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>100</td>
</tr>
</tbody>
</table>

**Model**

\[ Y \sim \text{size} + \text{type} + \text{temp} \]

**Trials**

10

**Inputs**

\[
\text{gen}\_\text{design}(\text{candidate set} = \text{coffeepoints}, \text{model} = \sim\text{size} + \text{type} + \text{temp}, \text{trials} = 10)
\]

**Output**

Default is D-Optimal

**Run** | **Large** | **Kona** | **Temp**  
--- | --- | --- | --- | 
1 | Large | Java | 80  
2 | Small | Java | 100 | 
3 | Small | Kona | 80  
4 | Large | Kona | 100 | 
5 | Small | Kona | 100 | 
6 | Small | Java | 80  
7 | Large | Kona | 80  
8 | Small | Java | 80  
9 | Large | Kona | 80  
10 | Large | Java | 100
Creation of the candidate set is simple

`expand.grid` function in R automatically generates all combinations of input parameters

```r
> expand.grid(size=c("Large","Small"),type=c("Java","Kona"),temp=c(80,90,100))
size type temp
1  Large Java  80
2   Small Java  80
3  Large Kona  80
4   Small Kona  80
5  Large Java  90
6   Small Java  90
7  Large Kona  90
8   Small Kona  90
9  Large Java 100
10  Small Java 100
11 Large Kona 100
12  Small Kona 100
```
Easily constrain your design space

Simple filtering of disallowed combinations to constrain the design space.

Example:
Testing within a flight envelope
Easily constrain your design space

Simple filtering of disallowed combinations to constrain the design space.

Filter out unwanted test points

filteredcandidateset = candidateset %%
filter(70000 - speed * 200 < altitude) >>
filter(57500 - speed * 100 > altitude) >>
filter(-100000 + speed * 500 > altitude)
Easily constrain your design space

Simple filtering of disallowed combinations to constrain the design space.

Use new candidate set to generate design

gen_design(candidateset = filteredcandidateset, ~speed + altitude + I(speed^2) + I(altitude^2), trials = 18)
Easily constrain your design space

Simple filtering of disallowed combinations to constrain the design space.

Use new candidate set to generate design
Easily constrain your design space

Simple filtering of disallowed combinations to constrain the design space.

Explore different options (e.g. I-optimal design)
Sequential construction of split-plot designs

```r
candidatelist = expand.grid(size = as.factor(c("Large", "Small")),
                          type = as.factor(c("Kona", "Java")),
                          temp = c(80, 100),
                          roastdarkness = c("Light", "Medium", "Dark"))

gen_design(candidateset = candidatelist, ~roastdarkness, trials = 6) -> htcdesign

gen_design(candidateset = candidatelist, ~size + temp + type, trials = 18,
            splitplotdesign = htcdesign, splitplotsizes = 3)
```

<table>
<thead>
<tr>
<th>roastdarkness</th>
<th>size</th>
<th>temp</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Dark</td>
<td>100</td>
<td>Java</td>
</tr>
<tr>
<td>1.2</td>
<td>Dark</td>
<td>80</td>
<td>Kona</td>
</tr>
<tr>
<td>1.3</td>
<td>Dark</td>
<td>100</td>
<td>Kona</td>
</tr>
<tr>
<td>2.1</td>
<td>Light</td>
<td>80</td>
<td>Kona</td>
</tr>
<tr>
<td>2.2</td>
<td>Light</td>
<td>100</td>
<td>Kona</td>
</tr>
<tr>
<td>2.3</td>
<td>Light</td>
<td>100</td>
<td>Java</td>
</tr>
<tr>
<td>3.1</td>
<td>Dark</td>
<td>80</td>
<td>Kona</td>
</tr>
<tr>
<td>3.2</td>
<td>Dark</td>
<td>80</td>
<td>Java</td>
</tr>
<tr>
<td>3.3</td>
<td>Dark</td>
<td>100</td>
<td>Java</td>
</tr>
<tr>
<td>4.1</td>
<td>Medium</td>
<td>80</td>
<td>Java</td>
</tr>
<tr>
<td>4.2</td>
<td>Medium</td>
<td>100</td>
<td>Kona</td>
</tr>
<tr>
<td>4.3</td>
<td>Medium</td>
<td>80</td>
<td>Java</td>
</tr>
<tr>
<td>5.1</td>
<td>Light</td>
<td>80</td>
<td>Kona</td>
</tr>
<tr>
<td>5.2</td>
<td>Light</td>
<td>80</td>
<td>Java</td>
</tr>
<tr>
<td>5.3</td>
<td>Light</td>
<td>100</td>
<td>Java</td>
</tr>
<tr>
<td>6.1</td>
<td>Medium</td>
<td>80</td>
<td>Java</td>
</tr>
<tr>
<td>6.2</td>
<td>Medium</td>
<td>100</td>
<td>Kona</td>
</tr>
<tr>
<td>6.3</td>
<td>Medium</td>
<td>100</td>
<td>Kona</td>
</tr>
</tbody>
</table>
Supports N-depth split-plot designs

gen_design(candidateset = candidatelist, ~roastdarkness, trials=4) -> vhtcdesign

gen_design(candidateset = candidatelist, ~type, trials=8,
    splitplotdesign = vhtcdesign, splitplotsizes = 2) -> htcdesign

gen_design(candidateset = candidatelist, ~size+temp, trials=24,
    splitplotdesign = htcdesign, splitplotsizes = 3)

<table>
<thead>
<tr>
<th>roastdarkness</th>
<th>type</th>
<th>size</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>Dark</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Dark</td>
<td>Kona</td>
<td>Small</td>
</tr>
<tr>
<td>1.1.3</td>
<td>Dark</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Dark</td>
<td>Java</td>
<td>Large</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Dark</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Dark</td>
<td>Java</td>
<td>Large</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Medium</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Medium</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Medium</td>
<td>Java</td>
<td>Large</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Medium</td>
<td>Kona</td>
<td>Small</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Medium</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Medium</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Light</td>
<td>Kona</td>
<td>Small</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Light</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Light</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Light</td>
<td>Java</td>
<td>Large</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Light</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Light</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Light</td>
<td>Kona</td>
<td>Large</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Light</td>
<td>Kona</td>
<td>Small</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Light</td>
<td>Kona</td>
<td>Small</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Light</td>
<td>Java</td>
<td>Small</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Light</td>
<td>Java</td>
<td>Large</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Light</td>
<td>Java</td>
<td>Large</td>
</tr>
</tbody>
</table>
Main functions in skpr

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
**eval_design – Basic Functionality**

<table>
<thead>
<tr>
<th>Size</th>
<th>Type</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>100</td>
</tr>
</tbody>
</table>

**Model**

\[ Y \sim \text{size} + \text{type} + \text{temp} \]

**Alpha**

0.2

**Inputs**

```r
eval_design(RunMatrix = coffeedesign, model = ~size + type + temp, alpha = 0.2)
```

**Output**

```
parameters          type       power
(Intercept)  effect.power  0.9329381
size          effect.power  0.7439050
type          effect.power  0.9329381
temp          effect.power  0.9329381
```

coffeedesign
## Change effect size

### Design

<table>
<thead>
<tr>
<th>Size</th>
<th>Type</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>100</td>
</tr>
</tbody>
</table>

### Model

\[ Y \sim \text{size} + \text{type} + \text{temp} \]

### Alpha

0.2

### Effect size

1.2

### Inputs

\[
\text{eval\_design}(\text{RunMatrix} = \text{coffeedesign}, \\
\text{model} = ~\text{size} + \text{type} + \text{temp}, \\
\text{alpha} = 0.2 \\
\text{effectsize} = 1.2)
\]

### Output

```
parameters       type       power    
(Intercept)      effect.power 0.6546176  
size             effect.power 0.4523370  
type             effect.power 0.6546176  
temp             effect.power 0.6546176  
(Intercept)      parameter.power 0.6546176  
size1            parameter.power 0.4962631  
size2            parameter.power 0.4570906  
type1            parameter.power 0.6546176  
temp             parameter.power 0.6546176  
```
Import and evaluate external designs

Simply import csv/excel file and skpr can evaluate it.

skpr automatically detects and converts JMP-generated split-plot designs.
Main functions in skpr

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
No need to approximate when you can simulate

Calculating power for anything more complex than a non-blocked design with a normal response requires making approximations.

Those approximations tend to fail at low numbers of runs (where power matters most).

In many cases, no approximations exist.
**eval_design_mc: Simple Monte Carlo power**

**Parametric**

```r
> expand.grid(size=as.factor(c("Large","Medium","Small")),
+     type=as.factor(c("Kona","Java")),
+     temp=c(80,100))
+ %>%
+ gen_design(~size+type+temp, 18)  %>%
+ eval_design(~size+type+temp, 0.2)
```

<table>
<thead>
<tr>
<th>parameters</th>
<th>type</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Intercept)</td>
<td>effect.power</td>
<td>0.9975907</td>
</tr>
<tr>
<td>2</td>
<td>size</td>
<td>0.9521264</td>
</tr>
<tr>
<td>3</td>
<td>type</td>
<td>0.9973936</td>
</tr>
<tr>
<td>4</td>
<td>temp</td>
<td>0.9973936</td>
</tr>
<tr>
<td>5 (Intercept)</td>
<td>parameter.power</td>
<td>0.9975907</td>
</tr>
<tr>
<td>6</td>
<td>size1</td>
<td>0.9473817</td>
</tr>
<tr>
<td>7</td>
<td>size2</td>
<td>0.9473817</td>
</tr>
<tr>
<td>8</td>
<td>type1</td>
<td>0.9973936</td>
</tr>
<tr>
<td>9</td>
<td>temp</td>
<td>0.9973936</td>
</tr>
</tbody>
</table>

**Monte Carlo**

```r
> expand.grid(size=as.factor(c("Large","Medium","Small")),
+     type=as.factor(c("Kona","Java")),
+     temp=c(80,100))
+ %>%
+ gen_design(~size+type+temp, 18)  %>%
+ eval_design_mc(~size+type+temp, 0.2)
```

<table>
<thead>
<tr>
<th>parameters</th>
<th>type</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Intercept)</td>
<td>parameter.power.mc</td>
<td>0.998</td>
</tr>
<tr>
<td>2</td>
<td>size1 parameter.power.mc</td>
<td>0.943</td>
</tr>
<tr>
<td>3</td>
<td>size2 parameter.power.mc</td>
<td>0.949</td>
</tr>
<tr>
<td>4</td>
<td>type1 parameter.power.mc</td>
<td>0.999</td>
</tr>
<tr>
<td>5</td>
<td>temp parameter.power.mc</td>
<td>0.999</td>
</tr>
</tbody>
</table>
eval_design_mc – Basic Functionality

**Inputs**

- **Model**
  
  \[ Y \sim size + type + temp \]

- **Alpha**
  
  0.2

- **nsim**
  
  1000

- **glmfamily**
  
  gaussian

**Design**

<table>
<thead>
<tr>
<th>Size</th>
<th>Type</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Medium</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>100</td>
</tr>
</tbody>
</table>

**coffeedesign**

```
eval_design_mc(RunMatrix = coffeedesign, 
model = ~size + type + temp, 
alpha = 0.2, 
nsim = 1000, 
glmfamily = "gaussian")
```

**Output**

- **parameters**
  
  (Intercept) parameter.power.mc 0.998
  size1 parameter.power.mc 0.943
  size2 parameter.power.mc 0.949
  type1 parameter.power.mc 0.999
  temp parameter.power.mc 0.999
eval_design_mc – Four built-in GLM families

<table>
<thead>
<tr>
<th>Design</th>
<th>Model</th>
<th>Alpha</th>
<th>Effect</th>
<th>glmfamily</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y ~ size + type + temp</td>
<td>0.2</td>
<td>2</td>
<td>poisson</td>
<td></td>
</tr>
</tbody>
</table>

Inputs

eval_design_mc(RunMatrix = coffeedesign, model = ~size + type + temp, alpha = 0.2, effectsize = 2, glmfamily = "poisson")

Output

gaussian, binomial, poisson, exponential
**eval_design_mc** – Easy binomial power calculations

**Inputs**
- Design: coffeedesign_50runs
- Model: \( Y \sim size + type + temp \)
- Alpha: 0.2
- glmfamily: binomial

**Output**
```
parameters   type  power
(Intercept) parameter.power.mc 0.673
size1       parameter.power.mc 0.577
size2       parameter.power.mc 0.627
type1       parameter.power.mc 0.805
temp        parameter.power.mc 0.810
```

**Binomial Probabilities**
Range: 50%-80%
eval_design_mc – Split-plot power w/ REML

Design
splitplot_coffee

Model
Y ~ size + type + temp + roastdarkness

Blocking
TRUE

Inputs

```
eval_design_mc(RunMatrix = splitplot_coffee,
               model = ~size + type + temp + roastdarkness,
               alpha = 0.2,
               blocking=TRUE)
```

Output

<table>
<thead>
<tr>
<th></th>
<th>type</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Intercept)</td>
<td>0.781</td>
</tr>
<tr>
<td>2</td>
<td>size1</td>
<td>0.990</td>
</tr>
<tr>
<td>3</td>
<td>type1</td>
<td>0.999</td>
</tr>
<tr>
<td>4</td>
<td>temp</td>
<td>0.995</td>
</tr>
<tr>
<td>5</td>
<td>roastdarkness1</td>
<td>0.612</td>
</tr>
<tr>
<td>6</td>
<td>roastdarkness2</td>
<td>0.616</td>
</tr>
</tbody>
</table>
eval_design_mc - Non-normal split-plot power

Model
Y ~ size + type + temp + roastdarkness

Alpha 0.2

glmfamily binomial

Blocking TRUE

Design
spcoffeedesign_50runs

Binomial Probabilities
Range: 50%-80%

eval_design_mc(RunMatrix = spcoffeedesign_50runs,
model = ~size + type + temp + roastdarkness,
alpha = 0.2,
blocking = TRUE,
glmfamily = "binomial"
effectsize = c(0.5,0.8))

Output

<table>
<thead>
<tr>
<th>parameters</th>
<th>type</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>parameter.power.mc</td>
<td>0.498</td>
</tr>
<tr>
<td>size</td>
<td>parameter.power.mc</td>
<td>0.723</td>
</tr>
<tr>
<td>type1</td>
<td>parameter.power.mc</td>
<td>0.737</td>
</tr>
<tr>
<td>temp</td>
<td>parameter.power.mc</td>
<td>0.763</td>
</tr>
<tr>
<td>roastdarkness1</td>
<td>parameter.power.mc</td>
<td>0.464</td>
</tr>
<tr>
<td>roastdarkness2</td>
<td>parameter.power.mc</td>
<td>0.466</td>
</tr>
</tbody>
</table>

No approximate way to calculate this.
skprGUI
Summary & Questions

Quickly execute your entire design of experiments workflow in one short script. Seamlessly integrate advanced statistical techniques into your planning process. Know exactly how your results were calculated.
Extra Slides
gen_design – Change optimality criterion

Candidate Set

<table>
<thead>
<tr>
<th>size</th>
<th>type</th>
<th>temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>Large</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>Small</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>Small</td>
<td>Java</td>
<td>100</td>
</tr>
</tbody>
</table>

Model

\[ Y \sim \text{size} + \text{type} + \text{temp} + \text{temp}^2 \]

Trials

10

Criterion

“I”

Inputs

\[
\text{gen\_design} (\text{candidateset} = \text{coffeepoints}, \\
\text{model} = \sim\text{size} + \text{type} + \text{temp} + \text{I}(\text{temp}^2), \\
\text{trials} = 10, \\
\text{optimality} = \text{“I”})
\]

Outputs

<table>
<thead>
<tr>
<th>Run</th>
<th>Large</th>
<th>Kona</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Large</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>Large</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>Large</td>
<td>Kona</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>Large</td>
<td>Java</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>Small</td>
<td>Java</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>Large</td>
<td>Kona</td>
<td>90</td>
</tr>
<tr>
<td>7</td>
<td>Small</td>
<td>Kona</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>Small</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>Small</td>
<td>Java</td>
<td>90</td>
</tr>
<tr>
<td>10</td>
<td>Small</td>
<td>Kona</td>
<td>90</td>
</tr>
</tbody>
</table>
Five main functions

- **gen_design**: Optimal design generation
- **eval_design**: Parametric power evaluation
- **eval_design_mc**: Monte Carlo power evaluation
- **eval_design_survival_mc**: Censored Monte Carlo power evaluation
- **eval_design_custom_mc**: Custom library Monte Carlo power evaluation
eval_design_survival_mc: Censored power calculations

**Design**
coffeedesign

**Model**
Y ~ size + type + temp

**Alpha**
0.2

**distribution**
exponential

**Inputs**
- Censor Point: 5
- Censor Type: right

**Output**
```
eval_design_survival_mc(RunMatrix = coffeedesign,
model = ~size + type + temp,
alpha = 0.2,
censorpoint=5,
censortype="right",
distribution= "exponential")
```

```
parameters type     power
1 (Intercept) parameter.power.mc 0.930
2 size1 parameter.power.mc 0.702
3 size2 parameter.power.mc 0.267
4 type1 parameter.power.mc 0.780
5 temp parameter.power.mc 0.799
```
eval_design_custom_mc: Extensible framework

Recreate eval_design_survival_mc

```r
rsurvival = function(X,b) {
    Y = rexp(n=nrow(X),rate=exp(-(X %*% b)))
    censored = Y > 1
    Y[censored] = 1
    return(Surv(time=Y,event=!censored,type="right"))
}

fitsurv = function(formula, X, contrastlist=NULL) {
    return(survreg(formula, data=X, dist="exponential")
}

pvalsurv = function(fit) {
    return(summary(fit)$Table[,4])
}

eval_design_custom_mc(RunMatrix=design, model=~a, alpha=0.05, nsim=100,
    fitfunction=fitsurv, pvalfunction=pvalsurv, rfunction=rsurvival, delta=1)
```

Same output

```r
> eval_design_custom_mc(RunMatrix=design, model=~a, alpha=0.05, nsim=10000,
+    fitfunction=fitsurv, pvalfunction=pvalsurv, rfunction=rsurvival, delta=1)
parameters type power
1 (Intercept) parameter.power.mc 0.9388
2 a parameter.power.mc 0.9222
> eval_design_survival_mc(RunMatrix=design, model=~a, alpha=0.05, nsim=10000,
+    censortype="right", distribution="exponential", censorpoint=1, delta=1)
parameters type power
1 (Intercept) parameter.power.mc 0.9351
2 a parameter.power.mc 0.9146
```
Type-I error increase in non-normal split-plot designs

Model
\[ Y \sim size + type + temp + roastdarkness \]

Alpha
0.2

glmfamily
binomial

Design
spcoffeedesign_50runs
10 blocks of 5 runs

Inputs

eval_design_mc(RunMatrix = spcoffeedesign_50runs, model = ~size + type + temp + roastdarkness, alpha = 0.2, glmfamily = "binomial" binomialprobs = c(0.5,0.8))

Binomial Probabilities
Range: 50%-80%

Output

Type-I error no blocking

Type-I error blocking