Extending the Theme System

zztable1

2026-02-18

Introduction

The theme system in zztable1 allows you to customize the appearance of your Table 1 output for different formats (console, LaTeX, HTML). This vignette demonstrates how to create custom themes and extend the theme system for your specific needs.

Built-in Themes

zztable1 comes with several pre-configured themes for major medical journals:

```r
available_themes <- list_available_themes()
cat("Available themes:", paste(available_themes, collapse = ", "), "\n")
```

Available themes: console, nejm, lancet, jama, bmj, simple

```r
# Get details on NEJM theme
nejm_theme <- get_theme("nejm")
cat("\nNEJM Theme Details:\n")
```

NEJM Theme Details:

```r
cat("- Name:", nejm_theme$name, "\n")
```

- Name: New England Journal of Medicine

```r
cat("- Decimal places:", nejm_theme$decimal_places, "\n")
```

- Decimal places: 1

```r
cat("- CSS class:", nejm_theme$css_class, "\n")
```

- CSS class: table1-ewnglandournalofedicine

Creating Custom Themes

Basic Custom Theme

The simplest way to create a custom theme is using `create_custom_theme()`, which creates a theme based on an existing theme:

```r
# Create a custom theme starting from NEJM
my_theme <- create_custom_theme(
  name = "MyCustom",
  base_theme = "nejm",
  decimal_places = 2
)
```

```r
cat("Custom theme created:\n")
```

Custom theme created:

```r
cat("- Name:", my_theme$name, "\n")
```

- Name: MyCustom

```r
cat("- Decimal places:", my_theme$decimal_places, "\n")
```

- Decimal places: 2

```r
cat("- CSS class:", my_theme$css_class, "\n")
```

- CSS class: table1-yustom

Advanced Custom Theme

For more control, create a theme from scratch with individual property parameters:

```r
# Create a highly customized theme using individual parameters
corporate_theme <- create_custom_theme(
  name = "Corporate",
  base_theme = "console",
  decimal_places = 1,
  font_family = "Arial, sans-serif",
  font_size = "11pt",
  background_color = "#e8f4f8",
  border_color = "#003366"
)

cat("Corporate theme created with custom properties:\n")
```

Corporate theme created with custom properties:

```r
str(corporate_theme[c("name", "decimal_places")])
```

List of 2 $ name : chr “Corporate” $ decimal_places: num 1

Modifying Existing Themes

You can also modify an existing theme after creation using `customize_theme()`:

```r
# Modify an existing theme
modified_lancet <- customize_theme(
  "lancet",
  decimal_places = 3,
  css_properties = list(
    font_size = "12pt"
  )
)

cat("Modified Lancet theme:\n")
```

Modified Lancet theme:

```r
cat("- Decimal places:", modified_lancet$decimal_places, "\n")
```

- Decimal places: 3

```r
cat("- Font size:", modified_lancet$css_properties$font_size, "\n")
```

- Font size: 12pt

Using Custom Themes

With table1() Function

Pass your custom theme to the `table1()` function:

```r
# Sample data
data(mtcars)
mtcars$transmission <- factor(ifelse(mtcars$am == 1, "Manual", "Automatic"))
mtcars$engine <- factor(ifelse(mtcars$vs == 1, "V-Engine", "Straight"))

# Create table with custom theme
bp <- table1(
  transmission ~ mpg + hp + wt,
  data = mtcars,
  theme = "nejm",
  pvalue = TRUE
)

# Display the table (format depends on output format)
cat("Table 1: Vehicle Characteristics by Transmission Type\n")
```

Table 1: Vehicle Characteristics by Transmission Type

```r
print(bp)
```

Table1 Blueprint (3 × 4) Formula: transmission ~ mpg + hp + wt Theme: New England Journal of Medicine Populated: 0/12 (0%)

Applying Themes to Existing Blueprints

If you already have a blueprint created, you can apply a theme using `apply_theme()`:

```r
# Create blueprint without theme
bp_unthemed <- table1(
  transmission ~ mpg + hp,
  data = mtcars
)

# Apply theme afterward
bp_themed <- apply_theme(bp_unthemed, "lancet")

cat("Theme applied to existing blueprint\n")
```

Theme applied to existing blueprint

Theme Properties Reference

Structure of a Theme Object

A theme is a list with the following key properties:

```r
# Examine theme structure
nejm_theme <- get_theme("nejm")

cat("NEJM Theme Structure:\n")
```

NEJM Theme Structure:

```r
cat("- name:", nejm_theme$name, "\n")
```

- name: New England Journal of Medicine

```r
cat("- theme_name:", nejm_theme$theme_name, "\n")
```

- theme_name: nejm

```r
cat("- decimal_places:", nejm_theme$decimal_places, "\n")
```

- decimal_places: 1

```r
cat("- css_class:", nejm_theme$css_class, "\n")
```

- css_class: table1-ewnglandournalofedicine

```r
cat("- header_separator:", nejm_theme$header_separator, "\n")
```

- header_separator:

```r
cat("\nCSS Properties:\n")
```

CSS Properties:

```r
for (prop in names(nejm_theme$css_properties)) {
  cat("  -", prop, ":", nejm_theme$css_properties[[prop]], "\n")
}
```

- background_color : #ffffff
- stripe_color : #fefcf0
- border_color : transparent
- font_family : 'Arial', Helvetica, sans-serif
- font_size : 10px
- line_height : 1.2

CSS Properties

Themes support the following CSS properties:

Property	Purpose	Example
`font_family`	Font used for the table	"Arial, sans-serif"
`font_size`	Base font size	"11pt"
`header_background`	Header row background color	"#f0f0f0"
`header_text_color`	Header text color	"black"
`row_stripe_color`	Alternating row color	"#fafafa"
`border_color`	Table border color	"#cccccc"
`padding`	Cell padding	"8px"

Dimension Rules

Themes can specify how variables and factors are displayed:

```r
# Get dimension rules from a theme
theme <- get_theme("nejm")

if (!is.null(theme$dimension_rules)) {
  cat("Dimension Rules:\n")
  str(theme$dimension_rules)
} else {
  cat("No custom dimension rules defined\n")
}
```

Dimension Rules: List of 8 $ factor_separator : chr “none” $ stratum_separator : chr “line” $ variable_grouping : chr “compact” $ missing_presentation : chr “inline” $ summary_style : chr “combined” $ footnote_style : chr “numbered” $ strip_empty_rows : logi TRUE $ factor_level_grouping: chr “flat”

Advanced: Output Format Handling

Understanding Format-Specific Behavior

Themes work across three output formats, with format-specific rendering:

Console Format

- Plain text output for terminal/console display
- Uses ASCII formatting
- Suitable for interactive analysis

LaTeX Format

- PDF generation for publication
- Uses LaTeX commands and packages
- Best for journal submissions
- Supports advanced formatting (bold, colors, striping)

HTML Format

- Web-based tables
- CSS styling for appearance
- Interactive browser display

Creating Format-Specific Customizations

For advanced use cases, you can create a theme with format-specific properties:

```r
# Advanced theme with format-specific properties
publication_theme <- create_custom_theme(
  name = "Publication",
  base_theme = "nejm",
  decimal_places = 1,
  font_family = "Georgia, serif",
  background_color = "#fefcf0"
)

# This theme works across all formats automatically
# Format-specific rendering adjusts appearance
cat("Publication theme created\n")
```

Publication theme created

Best Practices

Naming Conventions

- Use descriptive names for your themes
- Include organization name if appropriate
- Avoid special characters
- Examples: "MyOrg_Publications", "Clinical_Standard", "Conference_2024"

Consistency

- Define themes once and reuse them
- Store custom themes in a package or configuration
- Document your theme choices in methods sections

Accessibility

- Ensure sufficient color contrast in themes
- Use patterns/striping in addition to colors
- Test output in multiple formats (console, PDF, HTML)

Testing

Always test your custom themes:

```r
# Test theme with different data types
test_data <- mtcars[1:10, ]
test_data$group <- factor(c(rep("A", 5), rep("B", 5)))

bp_test <- table1(
  group ~ mpg + hp + wt + cyl,
  data = test_data,
  theme = "nejm"
)

# Verify appearance
cat("Testing custom theme...\n")
```

Testing custom theme…

Troubleshooting

Common Issues

Theme Not Found

```r
# This will use console as fallback
my_table <- table1(x ~ y, data = df, theme = "nonexistent")
```

Solution: Use `list_available_themes()` to verify theme names

CSS Not Applied

Ensure you're using the rendered output in the correct format:

```r
# CSS only applies to HTML/LaTeX output
bp <- table1(transmission ~ mpg, data = mtcars, theme = "nejm")

# Console: plain text, no CSS
console_output <- render_console(bp)

# HTML: CSS applied
html_output <- render_html(bp)

# LaTeX: LaTeX formatting applied
latex_output <- render_latex(bp)
```

Rendering Issues

If theme colors don't appear in PDF: - Verify LaTeX packages are included in YAML header - Check PDF viewer supports colors - Try a simpler theme first

Getting Help

For issues with custom themes: 1. Verify theme structure with `str(your_theme)` 2. Check that all required fields are present 3. Test with built-in themes first 4. Review examples in the vignette

Summary

The theme system provides flexible control over Table 1 appearance:

- **Built-in themes** for major journals (NEJM, Lancet, JAMA, BMJ)
- **Custom themes** via `create_custom_theme()`
- **Theme modification** with `customize_theme()`
- **Multi-format support** (console, LaTeX, HTML)
- **CSS properties** for HTML styling
- **Dimension rules** for variable/factor formatting

With these tools, you can create professional tables tailored to your publication requirements while maintaining consistent formatting across formats.

References

See also: - `vignette("theming_system")` - Built-in theme demonstrations - `vignette("zztable1_guide")` - Package overview - `?create_custom_theme` - Function documentation - `?apply_theme` - Theme application