# React Ruin Charts

A React component library for visualizing song distributions and difficulty curves.

## Installation

Since this is a private package, you'll need to install it directly from GitHub:

```
# Using yarn
yarn add git+https://github.com/yourusername/react-ruin-charts.git

# Using npm
npm install git+https://github.com/yourusername/react-ruin-charts.git
```

## Development

This project uses Husky to ensure the distribution files are always up-to-date when pushing to the repository.

### Automatic Builds Before Pushing

A pre-push Git hook is configured to automatically run the build process before pushing your changes to the repository. This ensures that the `dist` directory is always up-to-date with the latest source code changes.

When you run `git push`, the following happens:

1. The pre-push hook is triggered
2. The build process runs (`yarn build`)
3. If the build succeeds, the push continues
4. If the build fails, the push is aborted

### Setting Up Husky After Cloning

When you clone the repository, Husky will be automatically set up when you run `yarn install` or `npm install` due to the `prepare` script in package.json.

### Manual Development

During development, you can use the following commands:

```
# Run the build once
yarn build

# Run the build in watch mode (rebuilds on file changes)
yarn dev
```

```
# Run the smoke test
yarn test
```

### Testing

The project uses Jest for testing. Tests are located in the `tests` directory and include:

- Unit tests for the `RandomNumber` utility, demonstrating:
  - Successive calls produce different values
  - The same seed produces the same sequence of values
  - Different seeds produce different sequences
  - Custom parameters work correctly

To run the tests:

```
yarn test
```

This will run all tests in the `tests` directory and provide coverage information.

## Components

### DifficultyChart

A chart component for visualizing and editing difficulty curves.

```
import { DifficultyChart } from 'react-ruin-charts';

function MyComponent() {
  const [values, setValues] = useState([0.2, 0.5, 0.8, 0.3, 0.6]);

  return (
    <DifficultyChart
      totalDuration={300}
      values={values}
      setValues={setValues}
      splineColor="#888"
      controlPointColor="#f00"
      controlPointRadius={5}
      editable={true}
      axisColor="#90caf9"
      labelColor="#90caf9"
      showTicks={true}
      timeUnit="timecode"
    />
  );
}
```

### SongDistributionChart

A chart component for visualizing song distributions.

```
import { SongDistributionChart } from 'react-ruin-charts';

function MyComponent() {
  const songDistribution = [
    {
      title: "Song Title",
      artist: "Artist Name",
      duration: 180,
      price: 10,
      pricePerMin: 3.33,
      color: "#ff0000",
      startTime: 0
    },
    // More songs...
  ];

  return (
    <SongDistributionChart
      songDistribution={songDistribution}
      totalDuration={600}
      axisColor="#90caf9"
      labelColor="#90caf9"
      timeUnit="timecode"
      showTicks={true}
    />
  );
}
```

## Hooks

### useSongDistribution

A hook for distributing songs across a timeline based on difficulty values.

```
import { useSongDistribution, SILENCE_POLICIES } from 'react-ruin-charts';

function MyComponent() {
  const songs = [
    {
      title: "Song Title",
      artist: "Artist Name",
      duration: 180,
      price: 10,
      pricePerMin: 3.33,
      color: "#ff0000"
    },
    // More songs...
  ];

  const difficultyValues = [0.2, 0.5, 0.8, 0.3, 0.6];

  const distributedSongs = useSongDistribution(
    songs,
    600, // totalDuration
    "seed123", // randomSeed
    difficultyValues,
    SILENCE_POLICIES.DISTRIBUTE
  );

  return (
    <div>
      {/* Use distributedSongs which now have startTime properties */}
    </div>
  );
}
```

## Utilities

The package exports various utility functions and constants:

```
import {
  // Constants
  CHART_DIMENSIONS,
  TOOLTIP_STYLES,
  CHART_SCALE,
  TOOLTIP_OFFSET,
  SILENCE_POLICIES,

  // Utility functions
  generateRandomColor,
  RandomNumber,
  calculatePricePerMin,
  calculateRandomRange,
  formatTime
} from 'react-ruin-charts';
```

### RandomNumber

A seedable random number generator that provides deterministic randomness:

```
import { RandomNumber } from 'react-ruin-charts';

// Create a random number generator with a specific seed
const rng = RandomNumber('my-seed');
```

```
// Create a random number generator with a time-based seed
// (automatically generated if no seed is provided)
const autoSeededRng = RandomNumber();

// Generate a random number between 0 and 1
const randomValue = rng.inRange();

// Generate a random number in a custom range
const customRange = rng.inRange({ min: 10, max: 20 });

// Generate a random integer
const randomInt = rng.intInRange({ min: 1, max: 100 });

// Generate a random color
const randomColor = rng.colorHex();

// Generate a random blue color
const blueColor = rng.colorHex({
  minHue: 180,
  maxHue: 240,
  minSaturation: 0.8
});
```

When no seed is provided, a seed string is automatically generated based on the current time (to the millisecond). This provides non-deterministic randomness when desired, while still allowing for deterministic sequences when a specific seed is provided.

## License

MIT