

figbox マニュアル

@monaqa

1. figbox とは

figbox は SATYSFI で図を整列配置するためのパッケージです。

2. figbox パッケージの概要

figbox パッケージを用いるには冒頭に以下のように記述します。

```
@require: figbox/figbox  
  
open FigBox
```

以後、すべての例で figbox パッケージが上の通りインポートされていることを前提とします。

では、早速 figbox パッケージを用いて図を配置してみましょう。以下の例は、有名なパングラムである “The quick brown...” を横幅 100pt で行分割して得られるテキストボックスを、紙面に中央揃えで表示したものです。

```
+fig-center(  
    textbox-with-width 100pt  
    {The quick brown fox jumps over the lazy dog.}  
) ;
```

```
The quick brown fox  
jumps over the lazy  
dog.
```

上のコードのうち、figbox パッケージが提供しているものは 2 つあります。1 つは `+fig-center` コマンドであり、これは引数として与えられた「図」を紙面に中央揃えで配置します。もう 1 つは `textbox-with-width` 関数であり、これは指定した長さを横幅に、指定したテキストを内容を持つよう行分割された段落を「図」として生成します。

上の説明で「図」と表現しましたが、その実体は `figbox` という型です。すなわち、`+fig-center` は `figbox` 型の値を引数に取るコマンド、`textbox-with-width` は `length` → `inline-text` → `figbox` 型の関数ということになります。

`figbox` パッケージでは、主に以下の 2 種類のコマンド又は関数を提供します。

- `figbox` 型の値を、インラインテキストやブロックテキストとして埋め込むためのコマンド（例：`+fig-center`）
- `figbox` 型の値を生成・変換する函数（例：`textbox-with-width`）

`figbox` パッケージでは、これらのインターフェースによって「複雑な図を文書内の自由な位置に配置する」ことを実現しています。次の章で具体例を見てみましょう。

3. Gallery

3.1. `figbox` を生成する関数

「図」として思い浮かぶ代表的なものは画像でしょう。本パッケージにおいて、画像は 2 種類の関数 `include-image` 及び `include-image-with-height` を用いて読み込むことができます。`include-image` では画像の横幅、`include-image-with-height` では縦幅を用いて画像のサイズを決定します。

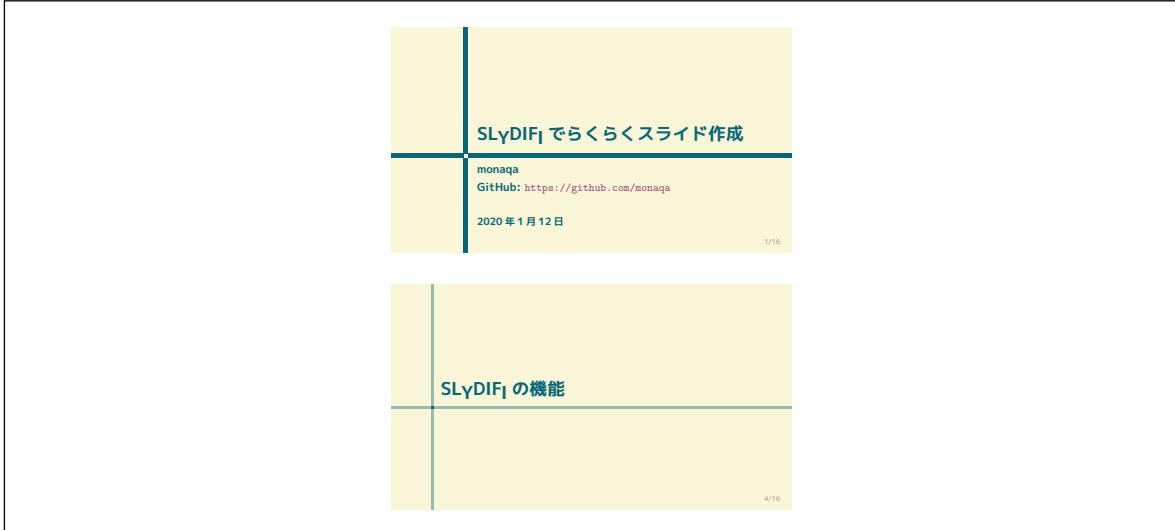
```
+fig-center(include-image 100pt `fig/example1.jpg`);  
+fig-center(include-image-with-height 100pt `fig/example1.jpg`);
```



上の例では JPEG 形式の図を読み込みましたが、同じ関数を用いて PDF 形式の図を指定す

ることもできます。PDF の場合はデフォルトで 1 ページ目の図を読み込むものの、オプション引数にページ番号を与えることで 1 ページ目以外の図を読み込むことも可能です。

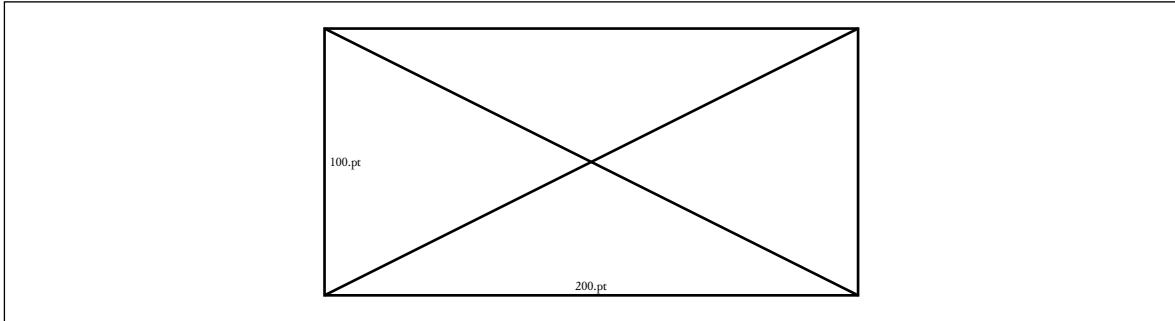
```
+fig-center(include-image 150pt `fig/example.pdf`);  
% 4 ページ目を読み込む  
+fig-center(include-image ?:4 150pt `fig/example.pdf`);
```



これらの関数は、画像の形式が JPEG と PDF のどちらであるかをファイルの拡張子から判断しています^{*1}。JPEG と PDF で異なる関数を使い分ける必要はありません。

縦横の長さを指定して、ダミーの図を配置することもできます。図の大きさが判明してさえいれば、まだ図そのものが無くても図を除いたレイアウトが再現できて便利です。

```
+fig-center(dummy-box 200pt 100pt);
```



最初の例で示したとおり、テキストや数式をはじめとしたインラインテキストやブロック

¹ v0.0.6 時点で SATYSFI が画像の読み込みをサポートしているフォーマットは JPEG と PDF の 2 種類です。

テキストも `figbox` 型に変換することができます。

```
+fig-center(textbox {The quick brown fox jumps over the lazy dog.});  
+fig-center(textbox {$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$});  
+fig-center(textbox-with-width 100pt {The quick brown fox jumps over  
the lazy dog.});  
+fig-center(textblock 150pt '<  
    +p{The quick brown fox jumps over the lazy dog.}  
    +p{The quick brown fox jumps over the lazy dog.}  
>');
```

The quick brown fox jumps over the lazy dog.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The quick brown fox
jumps over the lazy
dog.

The quick brown fox jumps
over the lazy dog.

The quick brown fox jumps
over the lazy dog.

以下は `easytable` パッケージを用いて組んだ表を中心揃えにする例です。

```
+fig-center(textbox {\easytable[l; c; r]{  
    | header1 | header2 | header3  
    | align left | align center | align right  
    | a | b | c  
}});
```

header1	header2	header3
align left	align center	align right
a	b	c

オプション引数にテキスト処理文脈の変換関数 (`context -> context` 型)を入れることで書式を変更することもできます。

```
+fig-center(
    textbox ?: (set-font-size 9pt)
    {The quick brown fox jumps over the lazy dog.}
);

+fig-center(
    textbox-with-width
    ?: (fun ctx -> ctx |> set-font-size 14pt
        |> set-text-color Color.blue)
    200pt
    {The quick brown fox jumps over the lazy dog.});
```

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the
lazy dog.

インラインボックス列を `figbox` に変換したいときは `inlinebox` 関数または `rawbox` 関数を用います。`inlinebox` 関数は `inline-boxes -> figbox` という型を持ち、`rawbox` 関数は `(context -> inline-boxes) -> figbox` という型を持っています。これらはエンドユーザがマークアップの場で直接用いるというよりも、主にパッケージ開発者が `figbox` の API を呼び出す際に用いることを想定しています。

3.2. `figbox` を変換・結合する関数

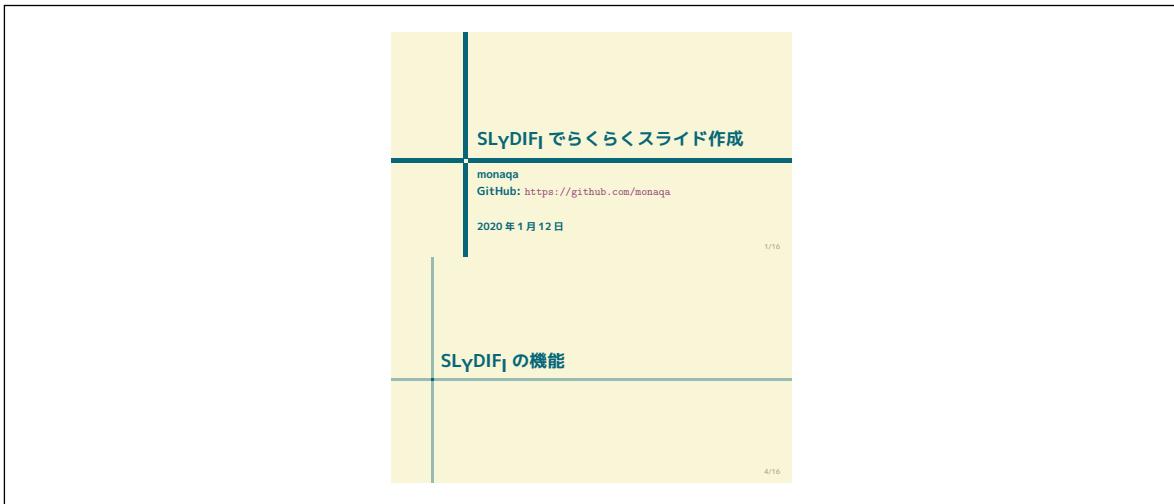
ここまで例では、`figbox` パッケージをわざわざ用いる有難みがさほど感じられなかったかもしれません。このパッケージの強みは、上で述べた関数を使って得られた「図」をいくつも組み合わせ、より複雑な図を簡単に作成できる点にあります。

`figbox` パッケージでは、複数の図を縦や横に結合することができます。

```
+fig-center(hconcat [
    include-image-with-height 100pt `fig/example1.jpg`;
    include-image-with-height 100pt `fig/example2.jpg`;
]);
```



```
+fig-center(vconcat [
  include-image 150pt `fig/example.pdf`;
  include-image ?:4 150pt `fig/example.pdf`;
]);
```



`hconcat` や `vconcat` の返り値もまた `figbox` 型であるということに注意してください。つまり、これらは自由にネストさせることができます。

```
+fig-center(
vconcat[
  hconcat [
    include-image-with-height 100pt `fig/example2.jpg`;
    include-image-with-height 100pt `fig/example1.jpg`;
    include-image-with-height 100pt `fig/example2.jpg`;
  ];
  hconcat [
```

```

    include-image 100pt `fig/example.pdf`;
    include-image ?:4 100pt `fig/example.pdf`;
];
);

```



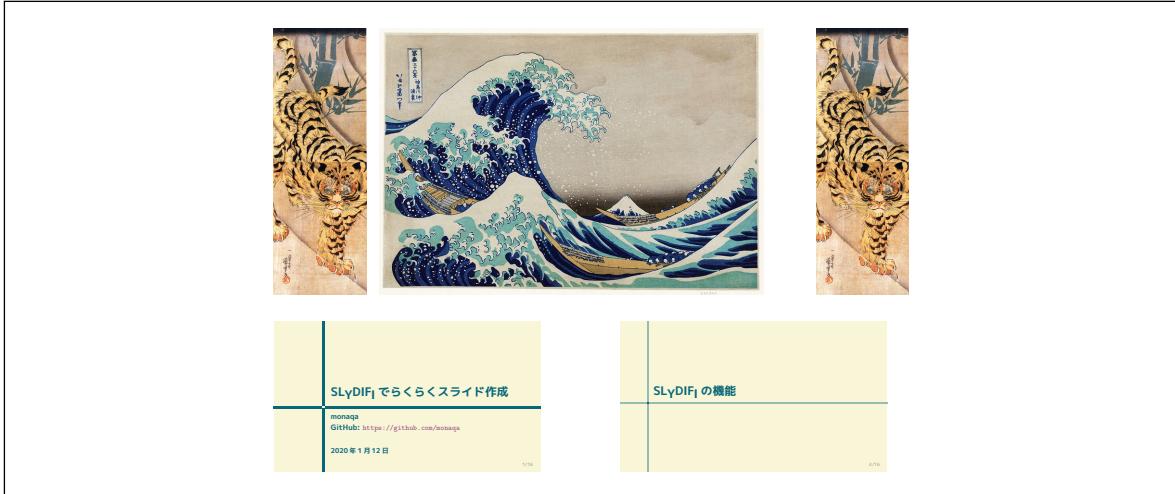
`hconcat` や `vconcat` 関数は単に図をつなげるだけであり、間に余白を入れてくれません。余白を入れるときは間に `gap` という関数により作成される特殊な `figbox` を指定します。これは `hconcat` 関数の中では横に隙間をあけるはたらきを、`vconcat` 関数の中では縦に隙間をあけるはたらきをします。

```

+fig-center(
vconcat[
  hconcat [
    include-image-with-height 100pt `fig/example2.jpg`;
    gap 5pt;
    include-image-with-height 100pt `fig/example1.jpg`;
    gap 20pt;
    include-image-with-height 100pt `fig/example2.jpg`;
  ];
  gap 10pt;
  hconcat [
    include-image 100pt `fig/example.pdf`;
    gap 30pt;
  ];
]
)

```

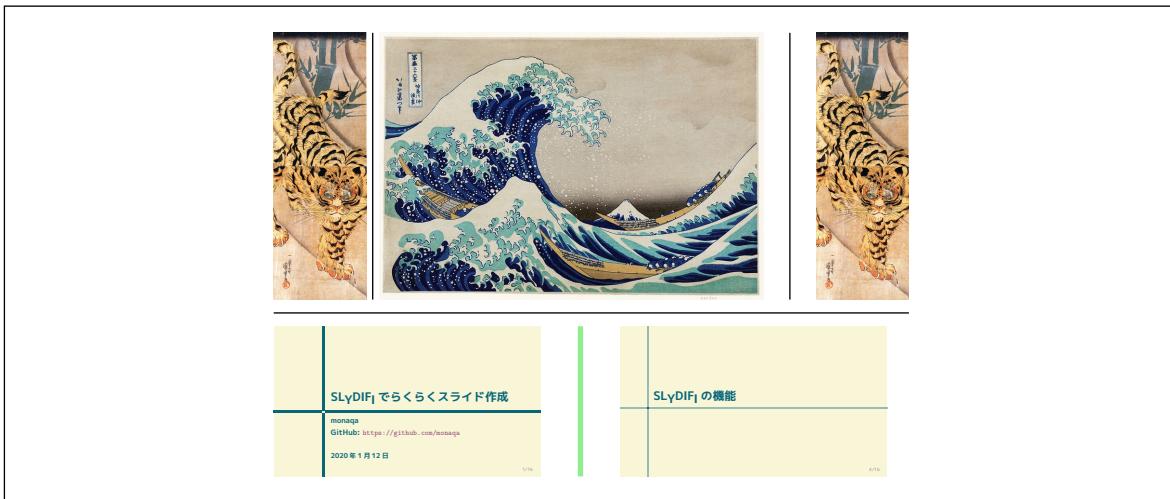
```
    include-image ?:4 100pt `fig/example.pdf`;  
];  
]  
);
```



`gap` 関数の代わりに `sep` 関数を用いると、間に隙間を空けるだけでなく、さらに分割線を引くことができます。線幅・線の色などを変えたい場合はオプション引数で指定します。以下は `base/color-ext` パッケージを用いて色を指定した例です。

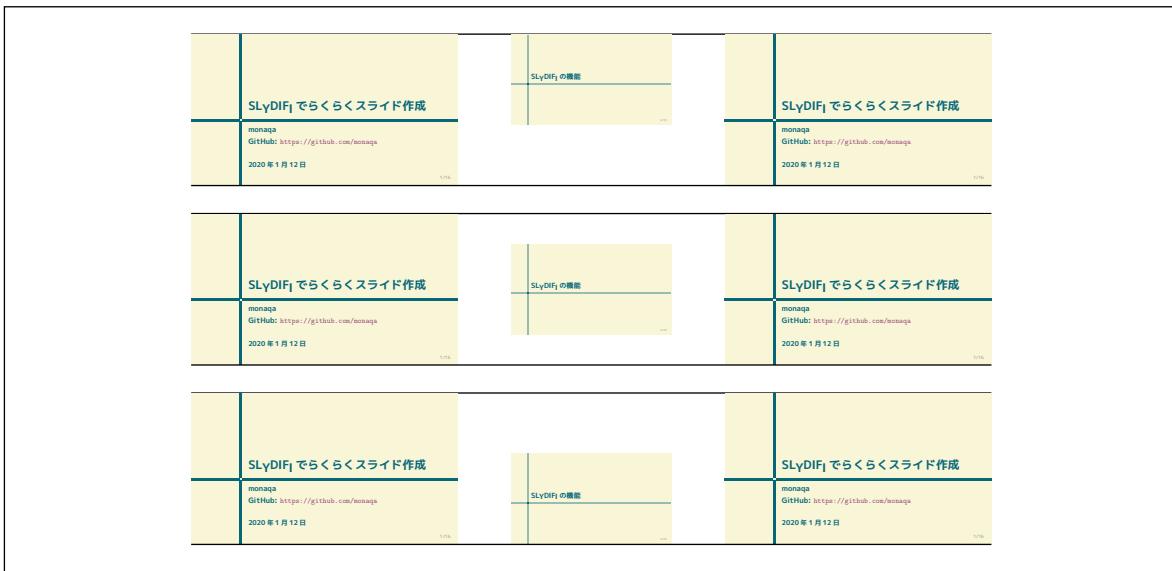
```
+fig-center(  
  vconcat [  
    hconcat [  
      include-image-with-height 100pt `fig/example2.jpg`;  
      sep 5pt;  
      include-image-with-height 100pt `fig/example1.jpg`;  
      sep 20pt;  
      include-image-with-height 100pt `fig/example2.jpg`;  
    ];  
    sep 10pt;  
    hconcat [  
      include-image 100pt `fig/example.pdf`;  
      sep ?: (stroke-line 2pt (Color.of-css `lightgreen`)) 30pt;  
      include-image ?:4 100pt `fig/example.pdf`;  
    ]  
  ]  
)
```

```
];
]
);
```



`hconcat` や `vconcat` を使用する際、オプション引数で `align-left`, `align-center`, `align-right`, `align-top`, `align-bottom` のいずれかを指定すると、図を並べるときの揃え方を変えることができます。

```
+fig-center(
    let img1 = include-image 100pt `fig/example.pdf` in
    let img2 = include-image ?:4 60pt `fig/example.pdf` in
    let separator = sep 0.5pt in
    vconcat [
        separator;
        hconcat ?:align-top [img1; gap 20pt; img2; gap 20pt; img1];
        separator; gap 10pt; separator;
        hconcat ?:align-center [img1; gap 20pt; img2; gap 20pt; img1];
        separator; gap 10pt; separator;
        hconcat ?:align-bottom [img1; gap 20pt; img2; gap 20pt; img1];
        separator;
    ]
);
```



図はフレームで囲むことができます。frame 関数は length -> color -> figbox -> figbox 型を持ち、与えられた線幅・線色で、与えられた figbox を囲みます。

```
+fig-center(
  frame 1pt Color.black (include-image 100pt `fig/example1.jpg`)
);
```



少し窮屈ですね。もう少し外側に枠を付けるべきかもしれません。そんなときには図にマージンをつけるのが良いでしょう。vmargin、hmargin、hvmargin はいずれも 1 つの length 型の値を引数にとり、それぞれ上下、左右、上下左右にマージンをつけることができます。

```
+fig-center(
  frame 1pt Color.black
  (hvmargin 5pt (include-image 100pt `fig/example1.jpg`))
);
```



なお、SATySFi 標準で用意されているパイプライン演算子 `|>` を用いると以下のように書くこともできます。

```
+fig-center(
    include-image 100pt `fig/example1.jpg`  

    |> hvmargin 5pt  

    |> frame 1pt Color.black  

);
```

このようにすればカッコのネストを軽減できるだけでなく、

1. `fig/example1.jpg` を横幅 100pt で読み込む
2. 上の図の上下左右に 5pt のマージンを追加する
3. 上の図を幅 1pt の黒線で囲む

と、直感的な順番で処理を記述することができます。

テキストボックスも「図」の一部でしたから、画像にキャプションを付けることだって、それにフレームを付けることだって簡単にできますね。

```
+fig-center(  

    vconcat ?:align-center [  

        include-image 200pt `fig/example.pdf`  

        |> frame 0.5pt Color.black;  

        gap 10pt;  

        textbox {図 1: `slydifi` パッケージで組んだスライドの例};  

    ]  

    |> hvmargin 10pt |> frame 1.5pt (Color.gray 0.8)  

);
```



図 1: `slydifi` パッケージで組んだスライドの例

`figbox` を変換する関数としては、その他にも図の背景色を指定する `bgcolor` などが用意されています。こちらもやはり `margin` 系の関数でマージンを指定しつつ使うのが良いでしょう。また、`figbox` を回転させる `rotate` 関数もあります。テキストや画像を回転させたいときに重宝するでしょう。

3.3. `figbox` を埋め込むコマンド

ここまでの一例では `figbox` を実際にインラインテキストやブロックテキストに埋め込むために全て `+fig-center` を使っていましたが、実際には他にもいくつかのコマンドがあります。

`+fig-block` は `fig-center` と似ていますが、オプション引数に `align-left`, `align-center`, `align-right` を取り、揃える位置を選ぶことができます。

```
+fig-block ?: (align-left) (
    include-image-with-height 100pt `fig/example2.jpg` |> rotate 90.
);
+fig-block ?: (align-center) (
    include-image-with-height 100pt `fig/example2.jpg` |> rotate 90.
);
+fig-block ?: (align-right) (
    include-image-with-height 100pt `fig/example2.jpg` |> rotate 90.
);
```



`+fig-abs-pos` 及び `\fig-abs-pos` は紙面に対する絶対座標を指定して描画するためのコマンドです。それ自身はそれぞれ大きさ 0 のブロックボックス列及びインラインボックス列としてふるまいます。

```
+fig-abs-pos((30pt, 50pt))(
  vconcat [
    include-image-with-height 100pt `fig/example2.jpg`;
    gap 5pt;
    textbox {↑絶対座標を指定して描画する例。}
  ]
);
```

`\fig-inline` を使えば、通常のインラインテキストの中に `figbox` を埋め込むことができます。画像やロゴをテキスト中に入れるときなどに重宝するかもしれません。

```
+p{つまり我々は\fig-inline(
  textbox{$\{E=mc^2\}} |> hvmargin 5pt |> frame 1pt Color.red
); という式を導いたのである。}
```

つまり我々は $E = mc^2$ という式を導いたのである。

`\fig-inline` はオプション引数を 1 つ取り、`align-top`などを指定することで埋め込む際に画像を行のどの位置を基準にして置くか指定することができます。

```
+p{つまり我々は\fig-inline?: (align-top)(
```



↑絶対座標を指定して描画する例。

```

    textbox{$E=mc^2} |> hvmargin 5pt |> frame 1pt Color.red
); という式を導いたのである。}
+p{つまり我々は \fig-inline?: (align-center)(
    textbox{$E=mc^2} |> hvmargin 5pt |> frame 1pt Color.red
); という式を導いたのである。}

```

つまり我々は $E = mc^2$ という式を導いたのである。

つまり我々は $E = mc^2$ という式を導いたのである。

箇条書きの中などインラインテキストから呼び出したいが別行立ての図にしたい場合は、
`+fig-block` や `+fig-center` のインライン版である `\fig-block` や `\fig-center` を用
 いることができます。

```

+p{つまり我々は \fig-block?: (align-right)(
    textbox{$E=mc^2} |> hvmargin 5pt |> frame 1pt Color.red
); という式を導いたのである。}
+p{つまり我々は \fig-center(
    textbox{$E=mc^2} |> hvmargin 5pt |> frame 1pt Color.red
); という式を導いたのである。}

```

つまり我々は

$$E = mc^2$$

という式を導いたのである。

つまり我々は

$$E = mc^2$$

という式を導いたのである。

`+fig-on-right` は段落の右側に figbox を配置する機能です。段落の幅は、figbox の幅に
 合わせて自動的に狭まります。

```

+p{ \jugem; }
+fig-on-right(include-image-with-height 100pt `fig/example2.jpg`)<

```

```
+p{ \jugem; }
+p{ \jugem; }
>
+p{ \jugem; }
```

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助。



+fig-on-left は逆に図を左側に配置します。

```
+p{ \jugem; }
+fig-on-left(include-image-with-height 100pt `fig/example2.jpg`)<
+p{ \jugem; }
+p{ \jugem; }
>
+p{ \jugem; }
```

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，



食う寝るところに住むところ、やぶら小路のぶら小路、パイポパイポ、パイポのシユーリンガン、シユーリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無、寿限無、五劫のすりきれ、海砂利水魚の水行末・雲来末・風来末、食う寝るところに住むところ、やぶら小路のぶら小路、パイポパイポ、パイポのシユーリンガン、シユーリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無、寿限無、五劫のすりきれ、海砂利水魚の水行末・雲来末・風来末、食う寝るところに住むところ、やぶら小路のぶら小路、パイポパイポ、パイポのシユーリンガン、シユーリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナーの長久命の長助。

+fig-on-right, +fig-on-left ともに length 型のオプション引数を取り、図と本文の間の余白を調整することができます。デフォルトは 20pt です。

```
+p{ \jugem; }
+fig-on-right?: (0pt) (include-image 120pt `fig/example1.jpg`)<
+p{ \jugem; }
>
+fig-on-right (include-image 120pt `fig/example1.jpg`)<
+p{ \jugem; }
>
+fig-on-right?: (40pt) (include-image 120pt `fig/example1.jpg`)<
+p{ \jugem; }
>
```

寿限無、寿限無、五劫のすりきれ、海砂利水魚の水行末・雲来末・風来末、食う寝るところに住むところ、やぶら小路のぶら小路、パイポパイポ、パイポのシユーリンガン、シユーリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナーの長久命の長助。

寿限無、寿限無、五劫のすりきれ、海砂利水魚の水行末・雲来末・風来末、食う寝るところに住むところ、やぶら小路のぶら小路、パイポパイポ、パイポのシユーリンガン、シユーリンガンのグーリンダイ、グーリンダイのポンポコピーのポンポコナーの



長久命の長助.

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助.

寿限無，寿限無，五劫のすりきれ，海砂利水魚の水行末・雲来末・風来末，食う寝るところに住むところ，やぶら小路のぶら小路，パイポパイポ，パイポのシューリンガン，シューリンガンのグーリンダイ，グーリンダイのポンポコピーのポンポコナーの長久命の長助.

