

figbox マニュアル

@monaqa

1. figbox とは

figbox は SATYSI 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。以後のすべての例で figbox パッケージが適切にインポートされていることを前提とします。

```
@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.
```

figbox パッケージが提供するのは `+fig-center` コマンドと `textBox-with-width` 関数です。`+fig-center` は引数として与えられた `figbox` 型の図を紙面に中央揃えで配置します。それに対して `textBox-with-width` は `length -> inline-text -> figbox` 型の関数であり、指定した長さを横幅に、指定したテキストを内容に持つよう行分割された段落を「図」として生成します。

このように、`figbox` パッケージが提供する機能は主に以下の 2 種類です。

- `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。
- `figbox` 型の値を生成・変換する函数。

上の説明から分かる通り、`figbox` パッケージではパッケージ名そのままの `figbox` という型がきわめて重要な役割を担っています。

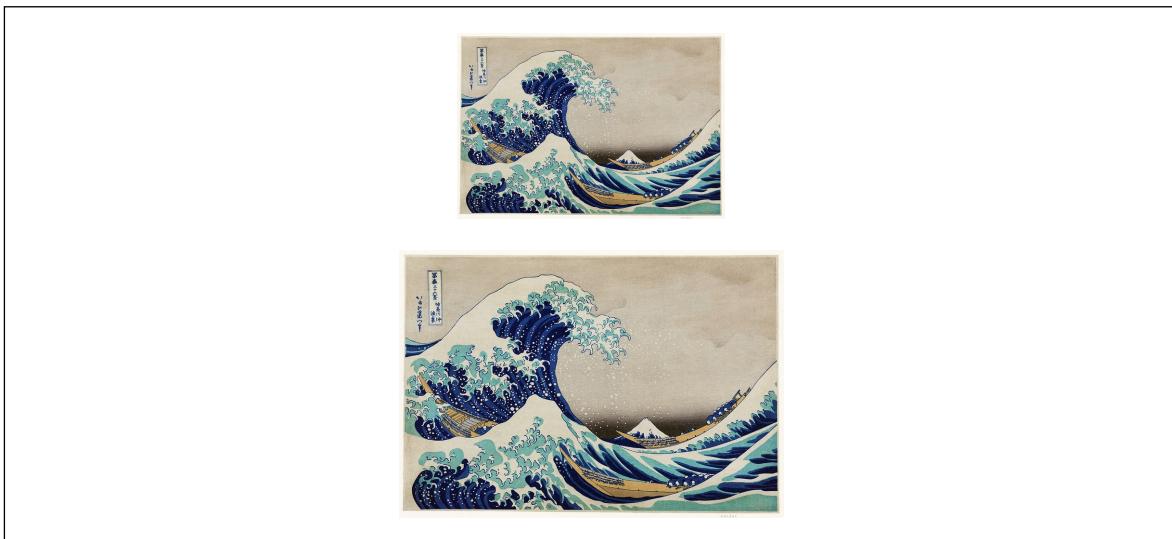
`figbox` を受け取って埋め込むコマンドは `+fig-center` だけではありません。後に述べるように、図を文書内の適切な位置に配置するための様々なコマンドが提供されています。また、`figbox` を生成するための関数も同様に様々なものがあります。単に `figbox` を生成するだけでなく、複数の `figbox` を組み合わせて新たな `figbox` を生成する、といったことも自由にできます。`figbox` パッケージでは、これらのインターフェースによって「複雑な図を文書内の自由な位置に配置する」ことを実現しています。次の章で具体例を見てみましょう。

3. Gallery

3.1. `figbox` を生成する関数

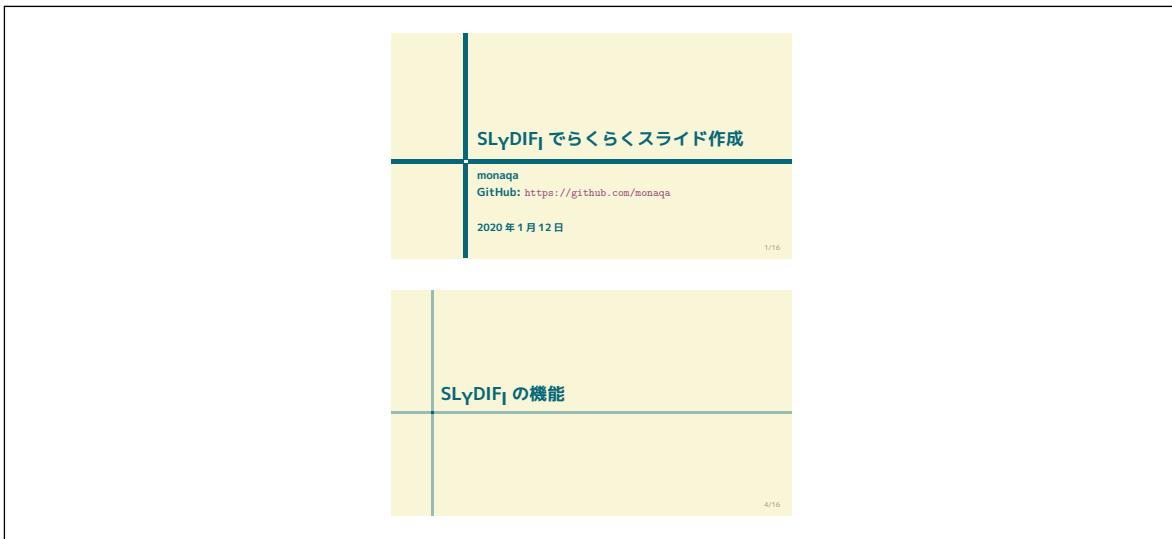
「図」として思い浮かぶ代表的なものは画像でしょう。画像は 2 種類の関数、`include-image` 及び `include-image-with-height` を用いて読み込むことができます。`include-image` は画像の横幅を、`include-image-with-height` は縦幅を指定します。

```
+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、同じ関数にて 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、オプション引数にて読み込みたいページ数を与えることもできます。

```
+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`);
```

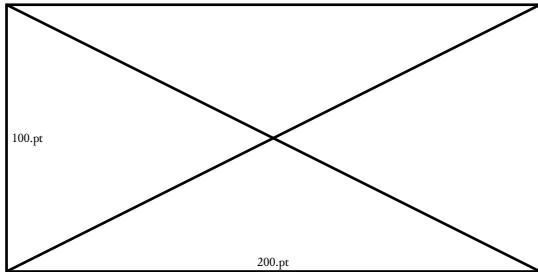


SATYSFI がサポートしている画像の形式は JPEG 及び PDF ですが、上で挙げた関数はファイルの拡張子から画像の形式を自動で判断しています。もちろん、`include-image-with-height` でも同様のことができます。

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);
```



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.});
```

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.

SATySFi では表やグラフィックスもインラインテキストに埋め込むことができるため、実質何でもあり、といえるかもしれません。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.

3.2. figbox を変換・結合する関数

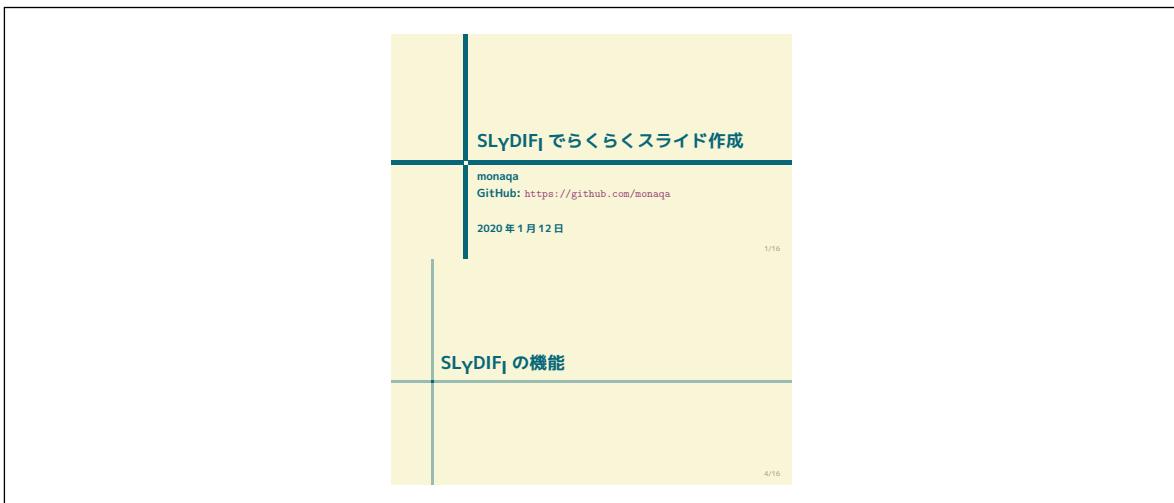
ここまで例では、`figbox` パッケージをわざわざ用いる有難みが感じられなかったかもしれません。このパッケージの強みは、上で述べた関数を使って得られた「図」をいくつも組み合わせ、より複雑な図を簡単に作成できる点にあります。

`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`;
    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`;
];
];
);

```



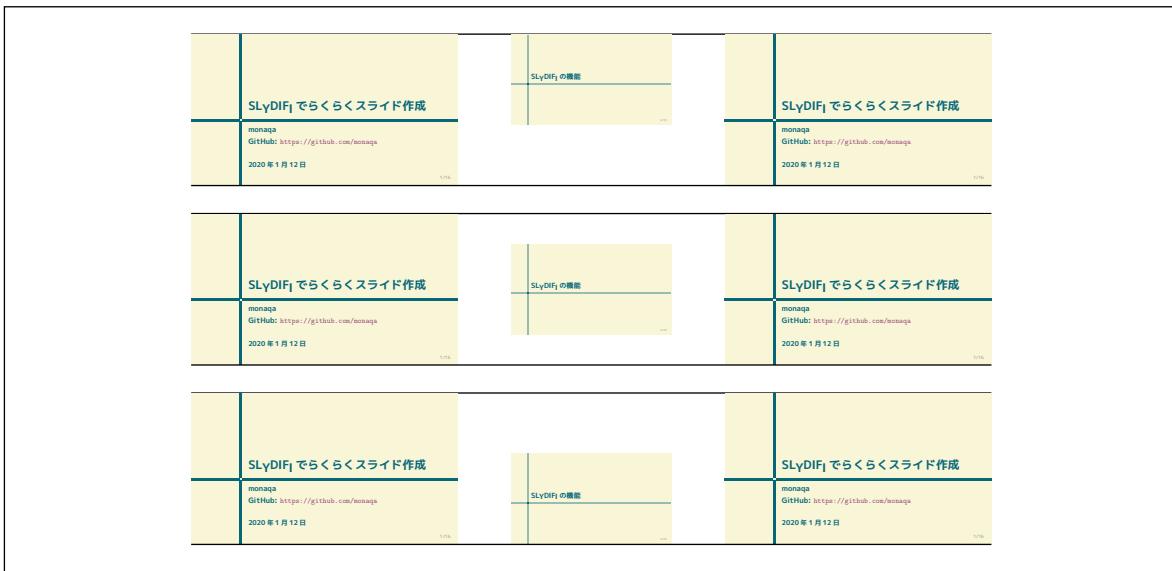
gap 関数の代わりに sep 関数を用いると、間に隙間を空けた後に線を引くことができます。

```
+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`;
    ];
]
```



オプション引数で align-left, align-center, align-right, align-top, align-bottom の 5 種類のいずれかを指定することで、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
    vconcat [
        sep 0pt;
        hconcat ?:align-top [img1; gap 20pt; img2; gap 20pt; img1];
        sep 0pt; gap 10pt; sep 0pt;
        hconcat ?:align-center [img1; gap 20pt; img2; gap 20pt; img1];
        sep 0pt; gap 10pt; sep 0pt;
        hconcat ?:align-bottom [img1; gap 20pt; img2; gap 20pt; img1];
        sep 0pt;
    ]
);
```



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[
        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)
);
```



`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` は紙面への絶対座標を指定して描画することができます。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, 200pt))(
  include-image-with-height 100pt `fig/example2.jpg`
);
```

`\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。

```
+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)(
  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。

テキストや数式の装飾には `inline-frame-inner` や `inline-frame-breakable` などを用いたほうが筋が良いものの、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。

`+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



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。

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。