R による探索的データ分析入門 - 可視化

発電基盤開発課　高津一誠

2018 年 *10* 月 *12* 日

1　可視化から学ぶ

前回お話したように、データプロセスの全体をサポートできるツールが R です。[*1]

では、どこから勉強を始めるべきでしょうか？

- プログラムミング言語だから文法から？
- 他の言語との違いを確認すべき？
- サポートするデータ形式から学ぶ？

そうではないと思います。

R を使ってデータ分析を論理的に行えるようになることが、このコースの目的です。ですから、データ分析プロセスのそれぞれを具体例を元に勉強するのがよいはずです。その中でも、可視化は効果が高く分かりやすいことから、学び始めるのに最も適していると思います。

2　割当（mapping）と階層 (layer)

R における可視化[*2]は、論理的に行えるようになっています。論理的とは、簡潔で一貫性のある方法でということです。Excel でのグラフ作成を反例として、確認していきましょう。

Excel で散布図を作るときは、データ範囲とグラフ種類を選べば一発です。ただし、これはデータ配置に依存していて、Excel の想定する「X に対応する列が一番左」というルールに合致すれば簡単ですが、そうでないと 1 つづつ系列を編集することになり急に非効率になります。

そもそもグラフ作成とは何をしているのでしょうか？それはデータをグラフ要素の視覚的属性に割り当てることと、グラフ要素を階層的に組み合わせることです。R ではこの割当と階層の組み合わせを、明示的でかつ必要最低限の記述によって指定します。

具体例を元に、割当と階層化を見ていきましょう

[*1] データ分析をすべて R を使って行うべき、ということではありません。Excel の方がやりやすいこと、計測器に付属する分析ツールや解析ツールのポスト処理を使う方が効率的なことも、もちろんあります。R を使った方が有効なときに使ってください。その判断は、これから勉強していくうちにできるようになります。

[*2] より正確には「tidyverse パッケージに含まれる ggplot2 パッケージにおける可視化」というべきですが、それについては後日お話します。

データは R に組み込まれたデータを使います。前回と同じで以下のとおりです。[*3]

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

このデータのガクの幅と長さの散布図を種ごとに色を変えて表示したいなら、割当は以下のようになります。

表1: 割当の一覧

データ列 (変数)	グラフの属性
Sepal.Length	X 値
Sepal.Width	Y 値
Species	色

この割当を R では以下のように書きます。なお、`%>%` は左側のデータを右側の処理に渡す演算子です。（この詳細についても後日お話します。）

```
iris %>% ggplot(aes(x = Sepal.Width, y = Sepal.Length, color = Species))
```

[*3] 生態学のデータで、アヤメの花弁（petal）とガク（sepal）の長さと幅を、3 つの種について 50 個体ずつ計測したもの。

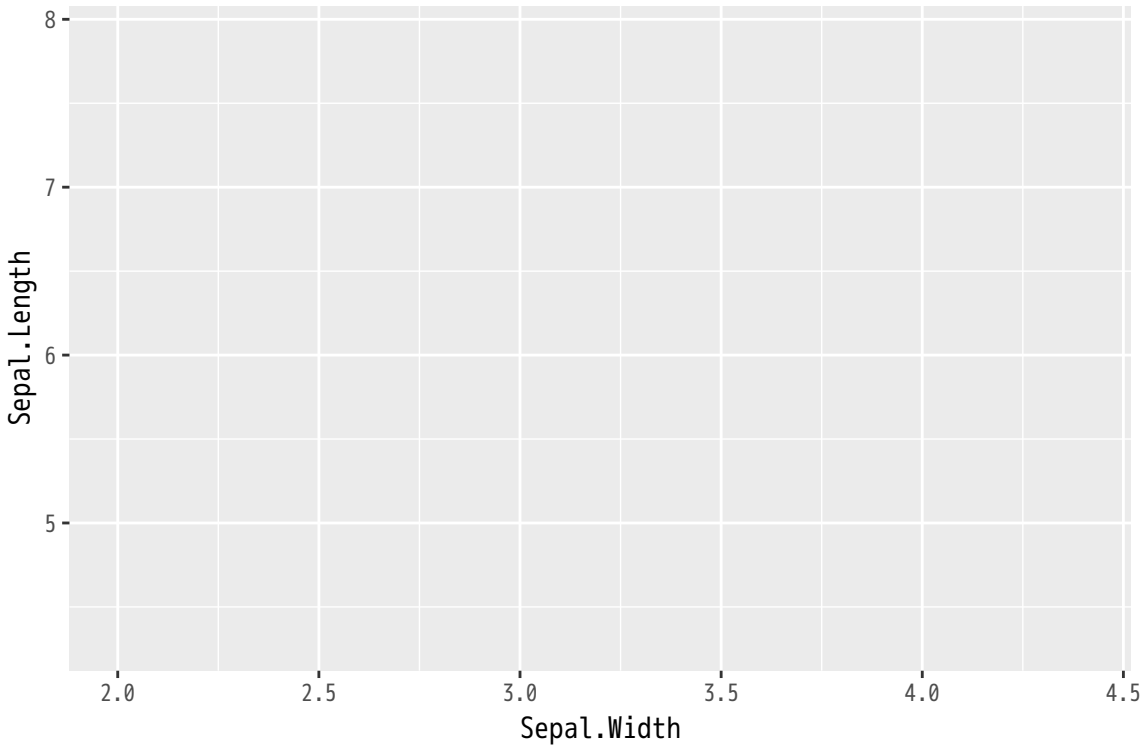


図1　割当を行っただけのグラフ

するとなにもプロットされていないワクが表示されましたが、X と Y のレンジは指定されています。これは与えられたデータのレンジに合っており、データ割当だけ行った状態で描けるものが描かれています。

次に階層を追加します。散布図なので、点を描く階層を以下のように追加します。

```
iris %>% ggplot(aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point()
```

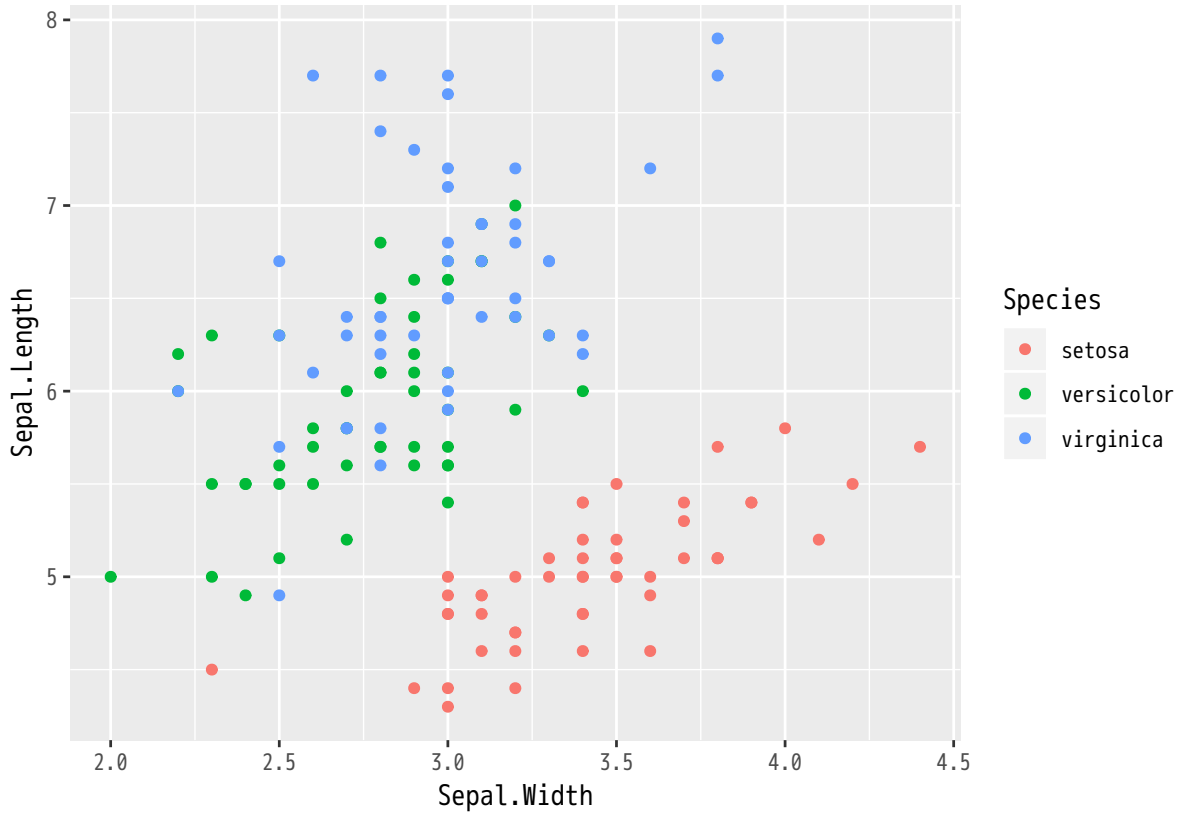


図2　点の階層を追加した

オーバラップしていて分かりにくいので種ごとにサブグラフに分けてみましょう。これも階層として以下のように指定できます。

```
iris %>% ggplot(aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point() +
  facet_wrap(~ Species)
```

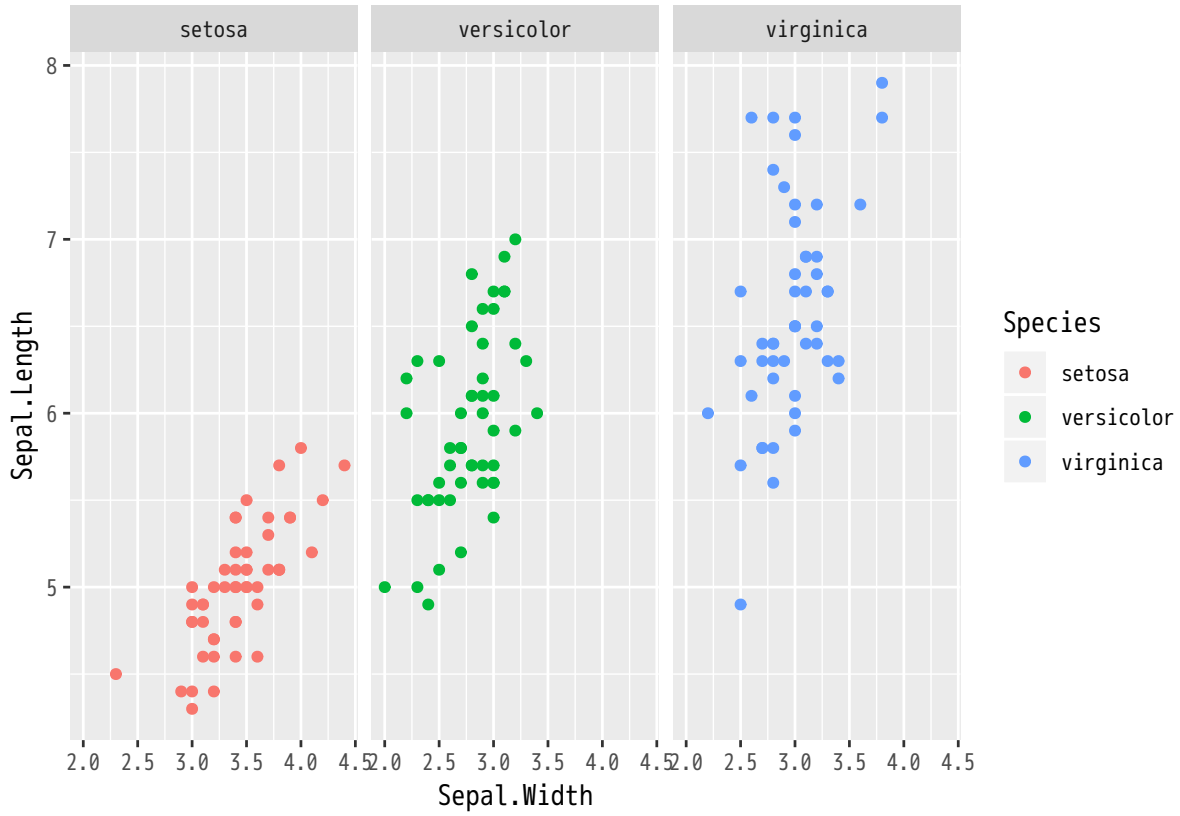


図3　サブグラフの階層を追加した

さらに、近似直線の階層も追加してみましょう。

```
iris %>% ggplot(aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_wrap(~ Species)
```

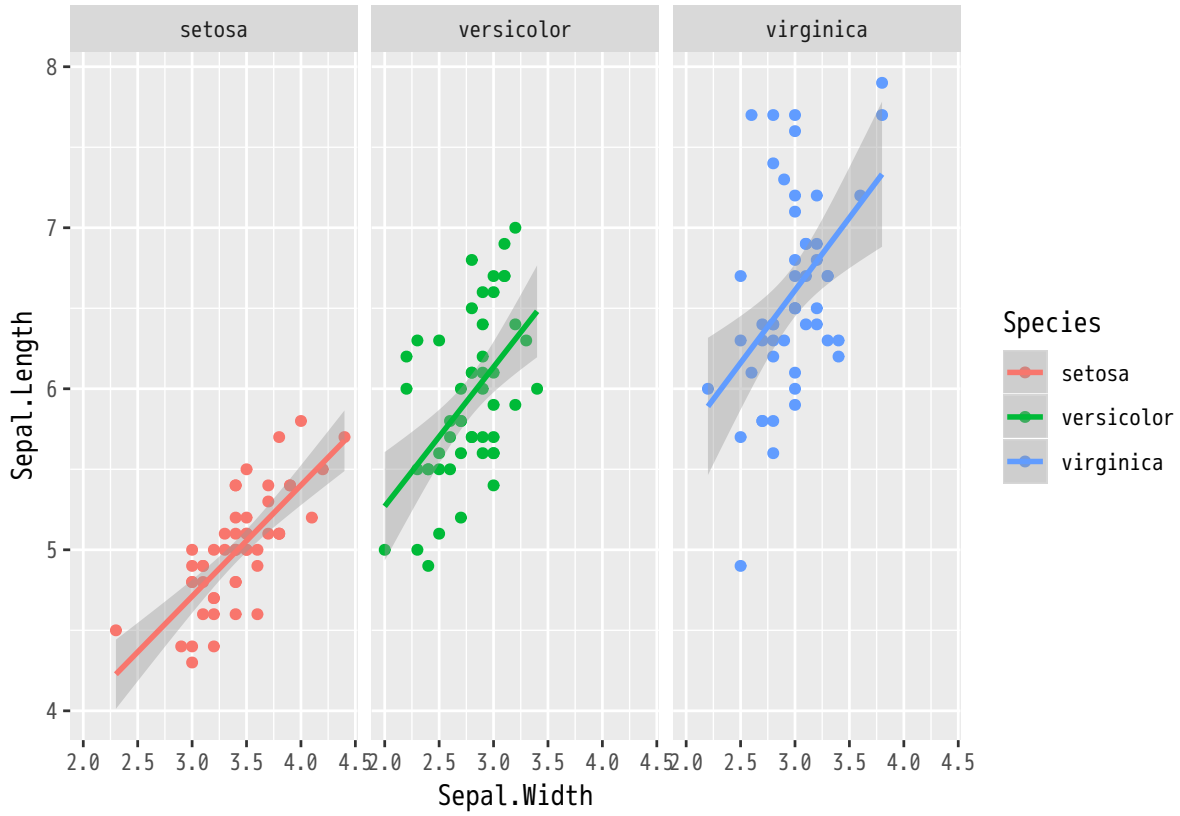


図4　近似直線も追加した

このようにグラフ要素を 1 つづつ重ねてグラフを作成していきます。

花弁のデータもプロットしたいときは、割当を上書きした階層を追加する方法で書けますが、もっと論理的な方法をデータ整形の回に勉強します。[*4]

```
iris %>% ggplot(aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point(aes(shape = "Sepal")) +
  geom_point(aes(x = Petal.Width, y = Petal.Length, shape = "Petal")) +
  facet_wrap(~ Species)
```

[*4] これはよくない例だということを覚えておいてください。また、ここでは更に点のマーカーの形への割当も追加していますが、"Sepal" や"Petal" という値を持った名前のない変数を割り当てた、と考えることができます。少し特殊な例なので無視して構いません。

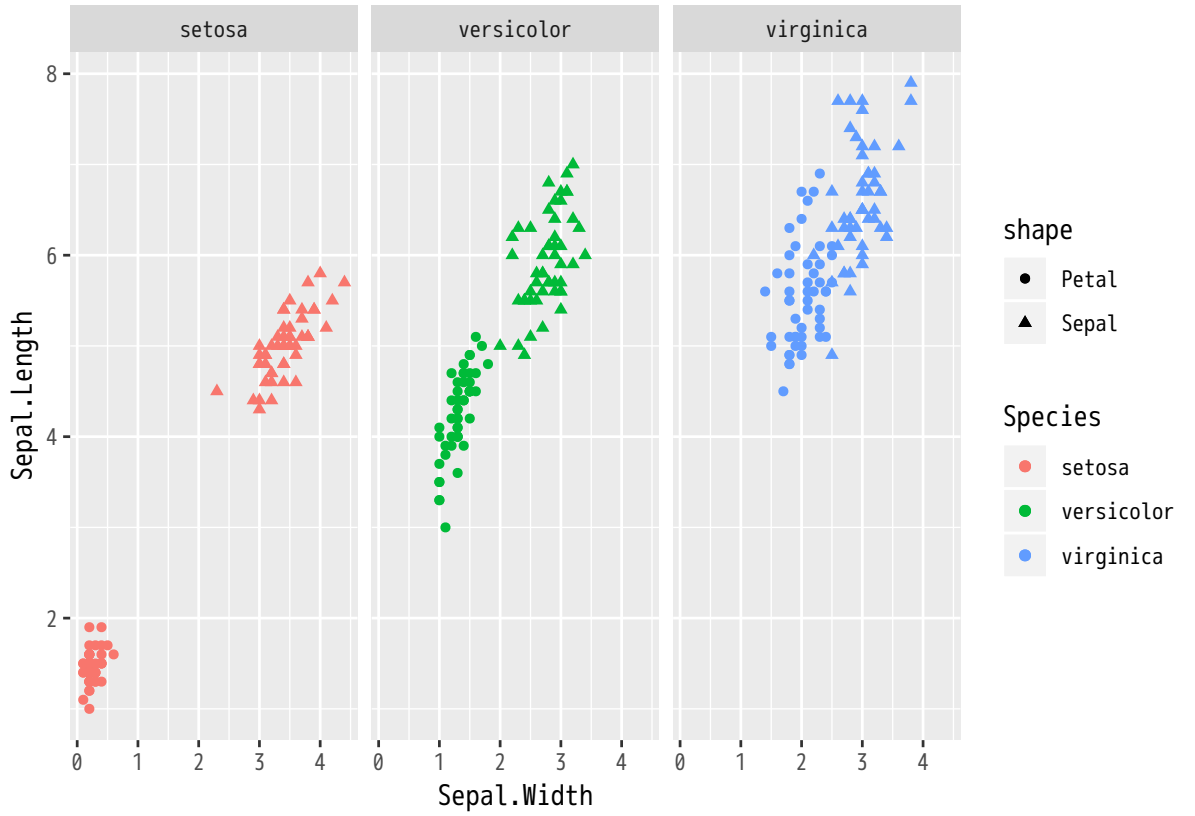


図5　よくない例

3　様々なグラフ

R では様々なグラフを作ることができ、メニュー [Help -> Cheetsheets -> Data Visualization with ggplot2] から、作成できるグラフの一覧を見ることができます。パッケージを追加すれば 3D グラフなども作れるようになります。

ここでは散布図以外のグラフを、いくつか実際に描いてみましょう。
グラフの種類を決めるグラフ要素は、geom_ から始まる名前の関数を使って作ります。

3.1　折れ線グラフ

ここではテスト用に、a と b という 2 つの独立したランダムデータをもつテーブルデータを用意します。[*5]

```
data <- tibble(a = rnorm(1000), b = rnorm(1000))
```

折れ線グラフには 2 種類あり、横軸の順にプロットするには geom_line() を、データの並び順にプロットす

[*5] tibble() はテーブルデータを作る処理、rnorm() は正規分布に従う乱数を指定された個数作る処理です。詳細は後で勉強します。

るには geom_path() を使います。

- geom_line()

```
data %>% ggplot(aes(x = a, y = b)) + geom_line()
```

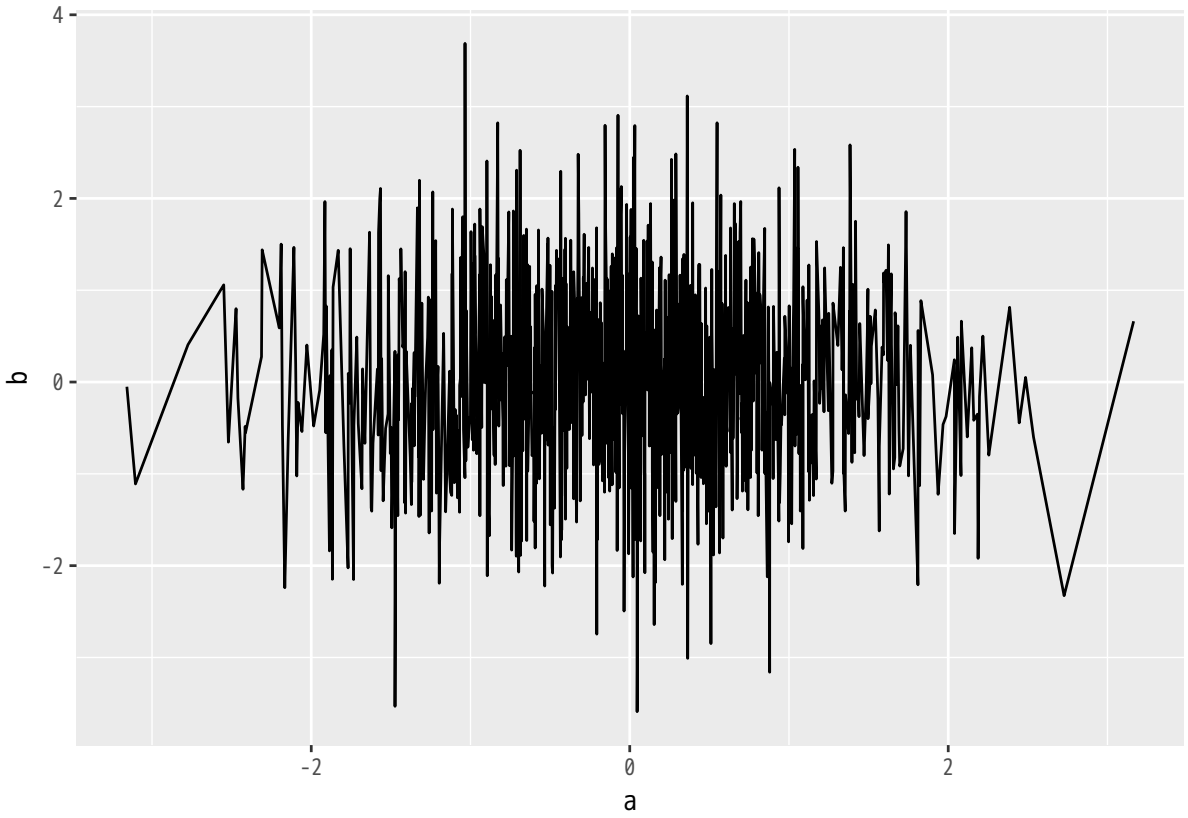


図6　折れ線グラフ (横軸順)

- geom_path()

```
data %>% ggplot(aes(x = a, y = b)) + geom_path()
```

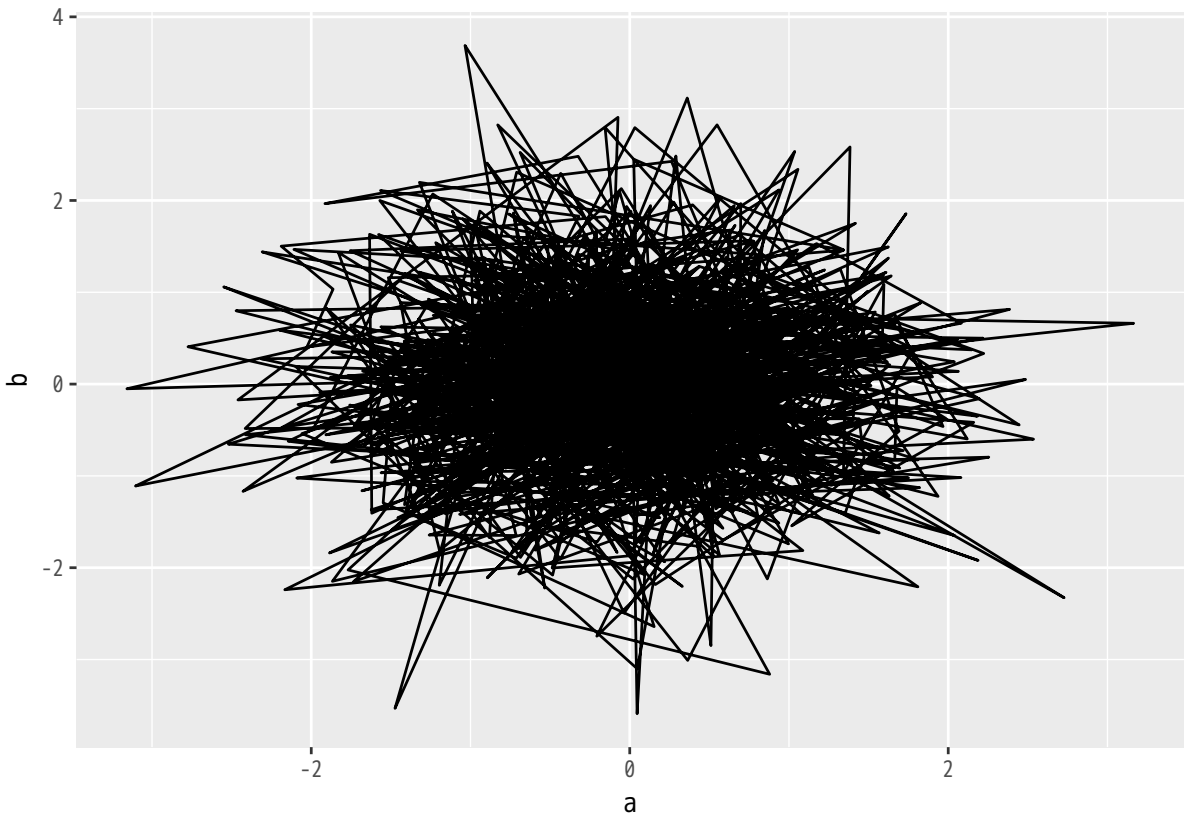


図7　折れ線グラフ (データ並び順)

3.2　棒グラフ

ここでは 50,000 個以上のダイアモンドの品質と価格のデータを集めた diamonds データを使います。棒グラフには 2 種類あり、棒の高さを指定してグラフを描かせる geom_col() と、データ集計結果をグラフに描かせる geom_bar() があります。
以下にそれぞれの例を示します。どちらもカット等級ごとのダイヤモンドの個数をグラフに描きます。

- geom_col()
  x にカット等級を y にその等級のダイアモンドの個数を指定しています。集計処理はデータ変換プロセスの項で勉強するので、ここでは読み飛ばしてください。

```
# カット等級ごとに個数を数える
diamonds_count <- diamonds %>% group_by(cut) %>% count()
diamonds_count
```

```
## # A tibble: 5 x 2
## # Groups:   cut [5]
##   cut           n
```

```
##   <ord>     <int>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

```
diamonds_count %>% ggplot(aes(x = cut, y = n)) + geom_col()
```

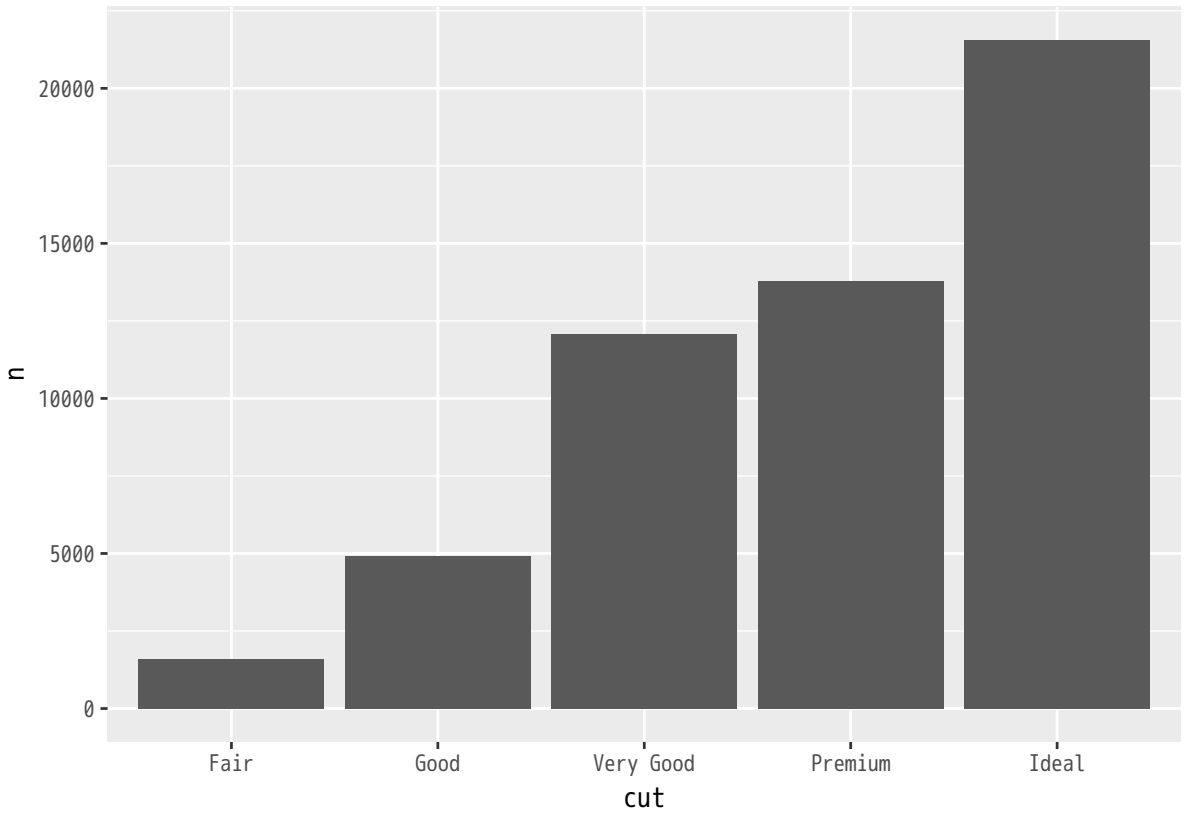
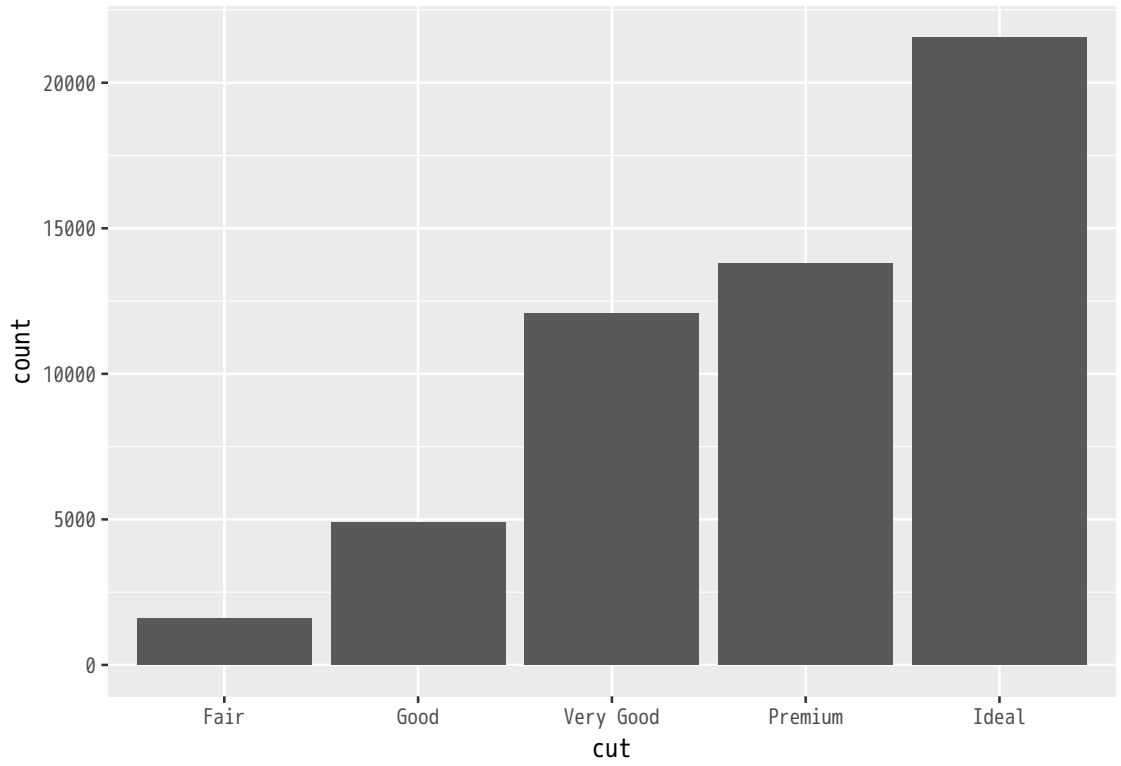


図8　棒グラフ（高さ指定）

- geom_bar()
  データの集計を R が行うので、x に割り当てる変数を指定するだけです。

```
diamonds %>% ggplot(aes(x = cut)) + geom_bar()
```



3.3 複数の変数の分布を比較するグラフ

複数の変数の分布を比較するには、色々な方法があります。場合に応じて分かりやすいものを使ってください。

- ヒストグラム
  指定した区間に含まれるデータの数をプロットしたグラフです。先ほどの棒グラフと同様に集計処理を R が行います。Excel だと、あらかじめ集計して度数分布表を作成するが必要がある[*6]ので、ずっと楽に使えます。区間幅 `binwidth` か区間数 `bins` を指定してください。

```
iris %>% ggplot(aes(x = Petal.Length)) +
  geom_histogram(binwidth = 0.2) +
  facet_wrap(~ Species, ncol = 1)
```

[*6] アドオンの分析ツールを使う方法もありますが、階級の境界値をすべて入力する必要があり、手間は変わりません。

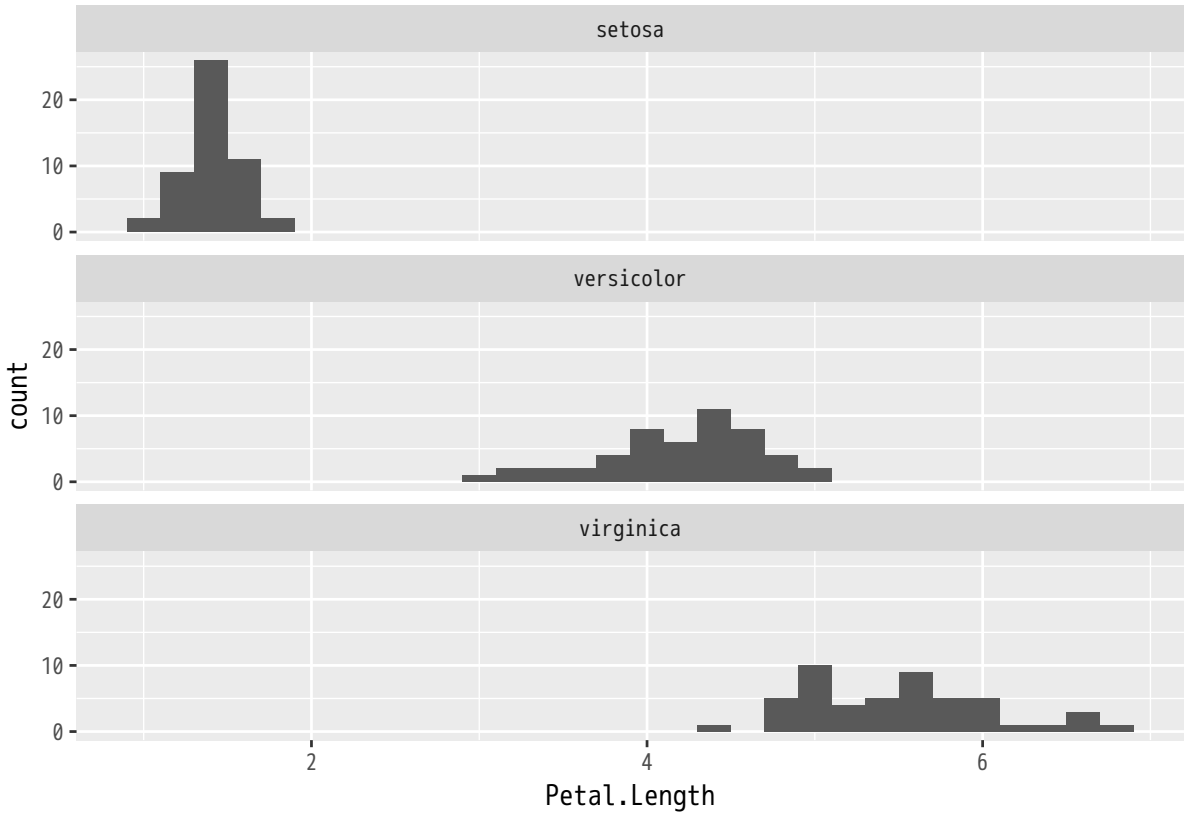


図9　ヒストグラム

- 確率密度分布

  ヒストグラムを平滑化したもので、サブグラフを作らなくても分布を比較しやすいグラフです。

```
iris %>% ggplot(aes(x = Petal.Length, fill = Species)) +
  geom_density(alpha = 0.6)
```

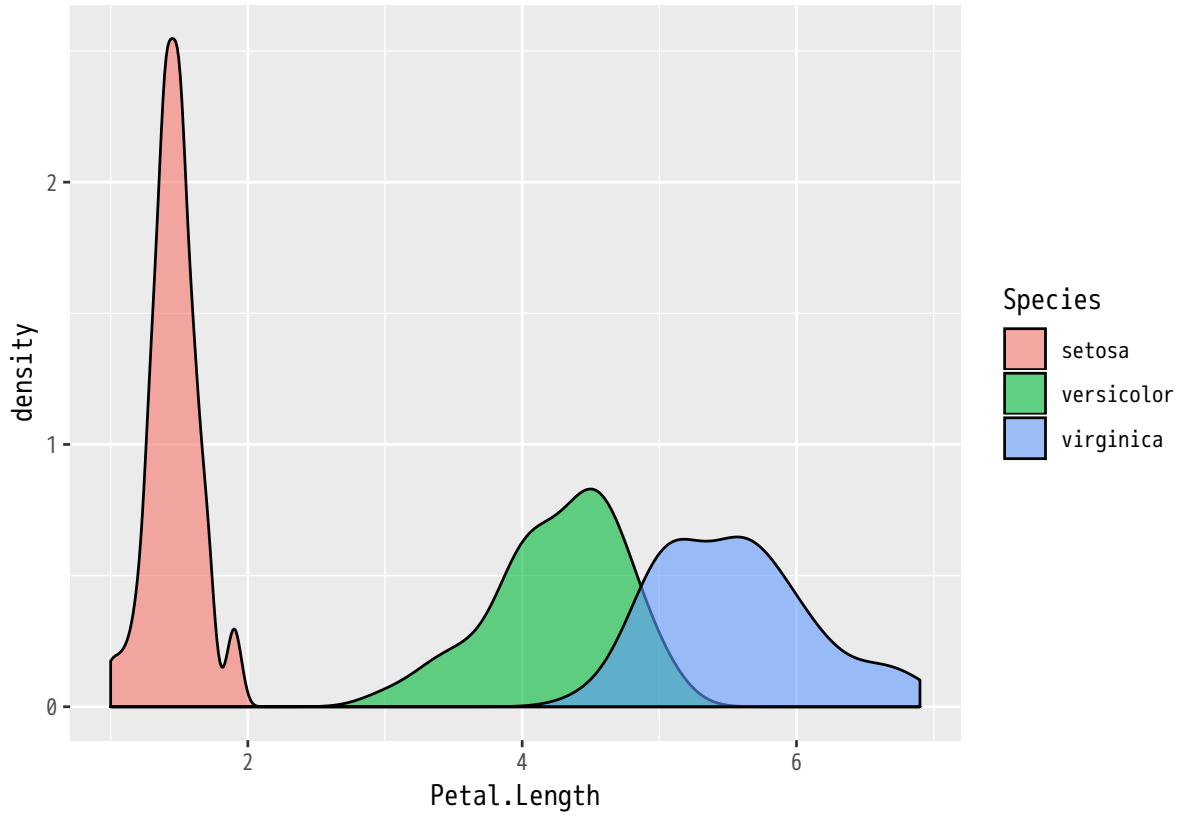


図10　確率密度分布

- バイオリンプロット
  確率密度分布を縦に並べたグラフです。後で説明する箱ひげ図と同様、横軸に連続値[*7]の変数を割り当てることもできるので、幅広く使えます。ここでは、散布図と組み合わせています。

```
iris %>% ggplot(aes(x = Species, y = Petal.Length)) +
  geom_violin(trim = F) +
  geom_point(alpha = 0.2)
```

[*7] ここではカテゴリーデータである Species を x に割当てていますが、数値や日付を割当てた場合でも使えます。

図11　バイオリンプロット

- 箱ひげ図

  四分位点[*8]などを描くグラフで、データ数が小さいときはよく使われます。

```
iris %>% ggplot(aes(x = Species, y = Petal.Length)) + geom_boxplot()
```

[*8] データを値の順に並べたとき全体の個数の 25%、50%、75% の位置にある値のこと。個別には、第 1 四分位点、第 2 四分位点（中央値）、第 3 四分位点と呼ばれる。

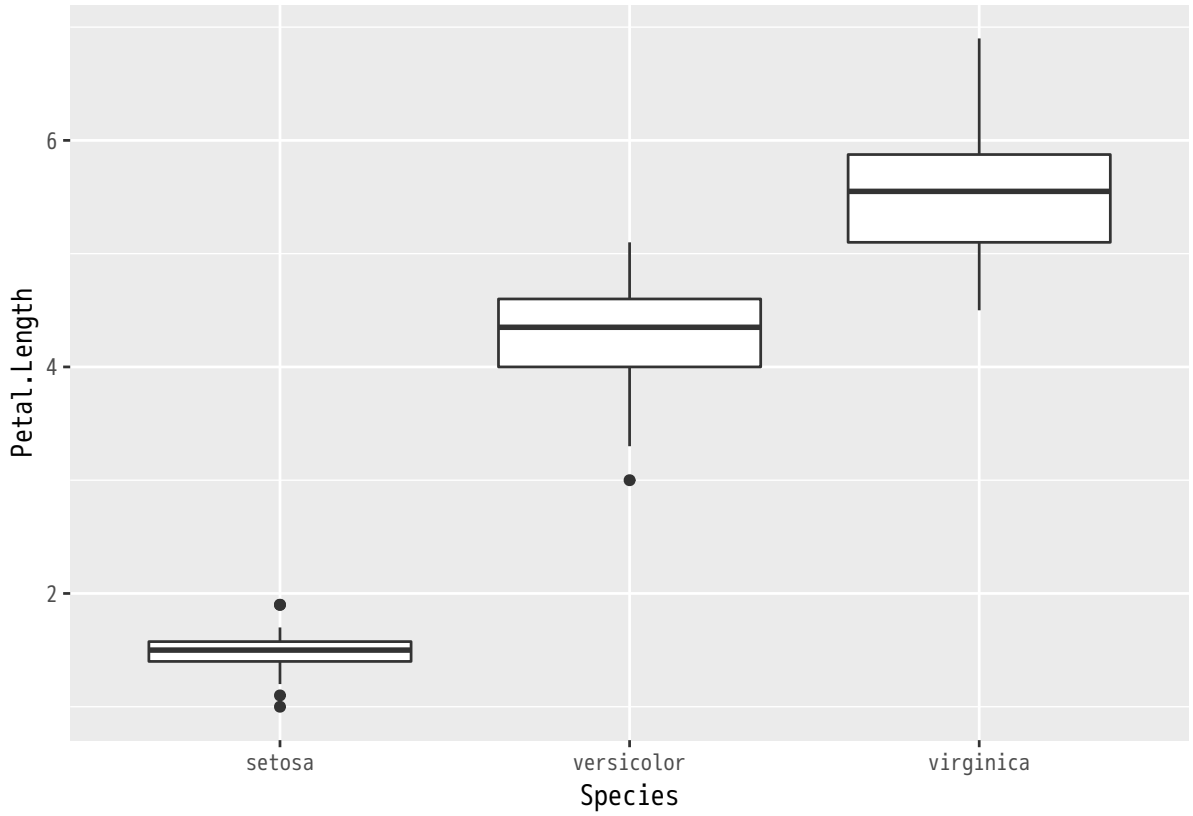


図12　箱ひげ図

表2: 箱ひげ図の意味

グラフの視覚的属性	割り当てられた値
箱の下側の線	第 1 四分位点
箱の中の線	第 2 四分位点（中央値）
箱の上側の線	第 3 四分位点
黒丸	外れ値[*9]
ひげの下端点	外れ値を除いた最小値
ひげの上端点	外れ値を除いた最大値

[*9] 箱の両側から箱の幅の 1.5 倍より離れたデータを外れ値とするのがデフォルト。データ数が大きくなると、外れ値が増えて見づらくなることが多い。

3.4　2 変数の同時分布を確認するグラフ

2 つの変数を組み合わせた分布を可視化する方法を 2 つ見てみましょう。データは折れ線グラフで使ったものと同じです。

- 2 次元ヒストグラム
  2 次元ヒストグラムは、区間内に含まれるデータの個数に応じて色をつけたものです。

```
data %>% ggplot(aes(x = a, y = b)) + geom_bin2d(binwidth = 0.3)
```

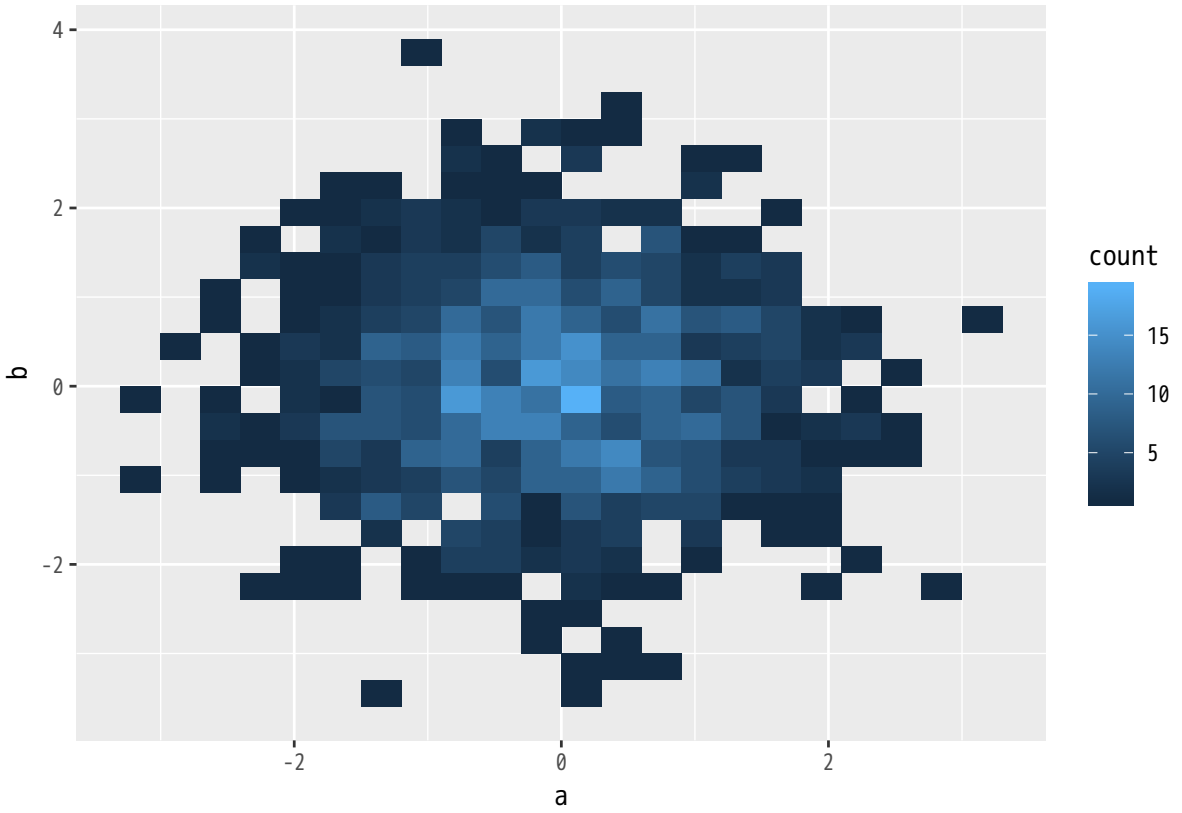


図13　2 次元ヒストグラム

- 2 次元確率密度分布
  2 次元確率密度分布は確率密度分布の 2 次元版（2 次元ヒストグラムを平滑化したもの）で、等高線が描かれます。

```
data %>% ggplot(aes(x = a, y = b)) + geom_density2d()
```

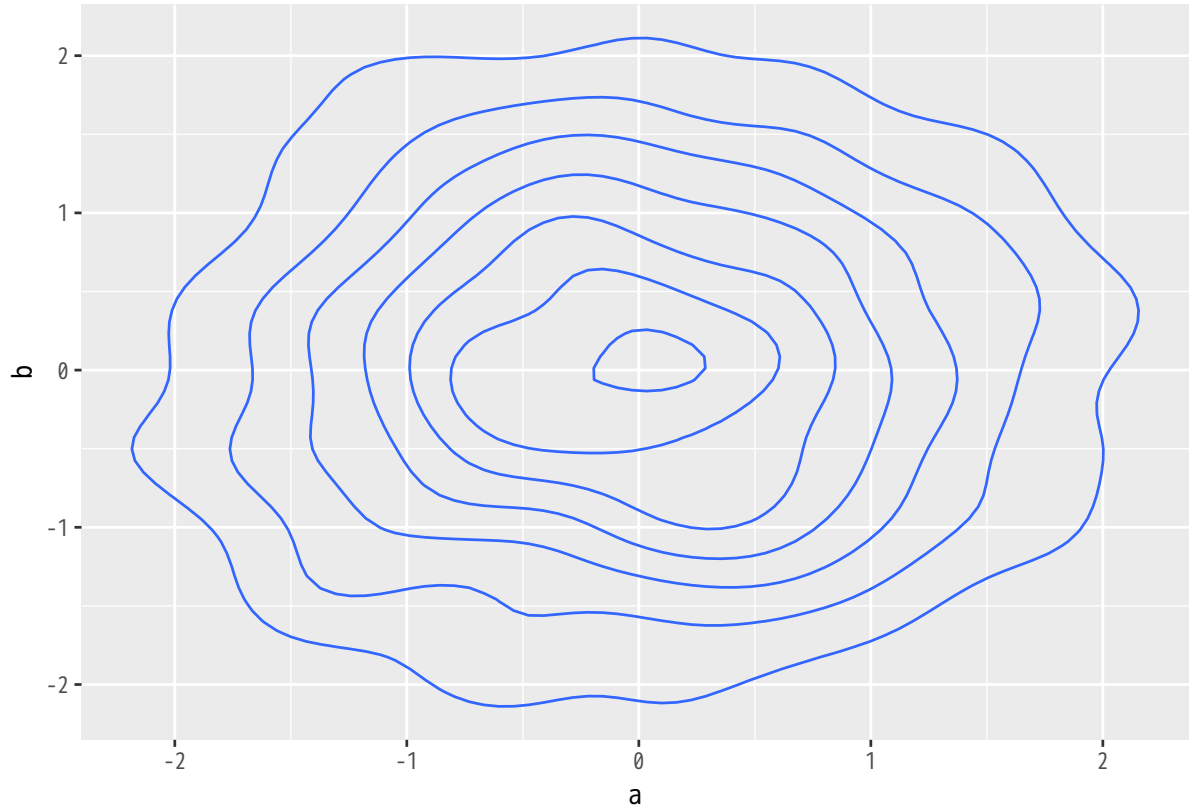


図14　2 次元確率密度分布

4　様々なグラフ要素

グラフ要素には、下表のように色々な種類があります。ジオメトリやファセットはすでに紹介したので、その他も簡単に紹介します。詳細は、メニュー [Help -> Cheetsheets -> Data Visualization with ggplot2] から確認できます。

表3: グラフ要素

グラフ要素	関数名	説明
ジオメトリ	`geom_*()`	グラフの種類を決める要素
スケール	`scales_*()`	スケールを調整する要素
座標系	`coord_*()`	座標系を指定する要素
ファセット	`facet_*()`	サブグラフを作成する要素
ラベル	`labs()`	ラベルを指定する要素
凡例	`guides()`	凡例を調整する要素

グラフ要素	関数名	説明
テーマ	`theme_*()`	全体の見た目を指定する要素

- スケール
  グラフの見た目属性にデータを割り当てるときの尺度（ものさし）を指定します。例えば、対数軸にしたり軸を反転させたり軸目盛りや目盛りラベルの変更が行なえます。また、色への割り当てもカラースケールを選ぶことで変更できます。

- 座標系
  座標系を設定して、X-Y が同比になるように座標系を調整したり、表示する領域を限定したり極座標にしたりすることができます。

- ラベル
  グラフの視覚的属性ごとにラベルを指定することができます。

- 凡例
  グラフの視覚的属性ごとに凡例を調整したり削除したりできます。

- テーマ
  用意されたテーマを指定して、全体的な見た目を変更できます。個別に細かい調整を行うことも可能です。

以下に例を示します。

- 例 1
  以下のようにグラフ要素を追加しています。
  - カラースケール：パッケージ `colorRamps` をロードして MATLAB に似た配色を指定する。
  - 座標系：X-Y を同比にする。
  - ラベル：デフォルトの割当てた変数名から変更する。
  - テーマ：背景が白黒の、プロジェクタで投影するときなどに向いたテーマに変更する。

```
library(colorRamps)

data %>% ggplot(aes(x = a, y = b)) +
  geom_bin2d(binwidth = 0.3) +
  scale_fill_gradientn(colors = matlab.like(10)) +
  coord_fixed() +
  labs(x = "ひずみ ( )", y = "応力 (MPa)", fill = "データ数",
       title = "無負荷時の計測結果") +
  theme_bw()
```

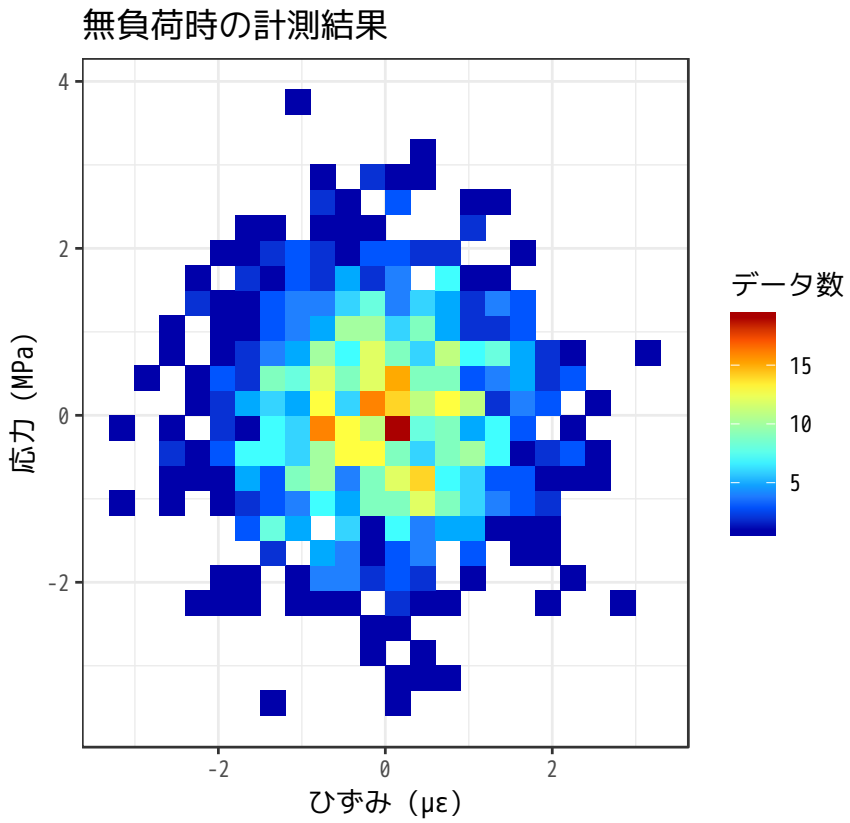


図15　様々なグラフ要素：例 1

- 例 2

  以下のようにグラフ要素を追加しています。

  – 座標系：メルカトル図法を使う。

  – テーマ：背景をなくす。

```
library(maps)

p_map <- map_data("world") %>%
  ggplot(aes(x = long, y = lat, group = group)) +
  geom_polygon(fill = "white", color = "black") +
  theme_void()
p_map + coord_quickmap()
```

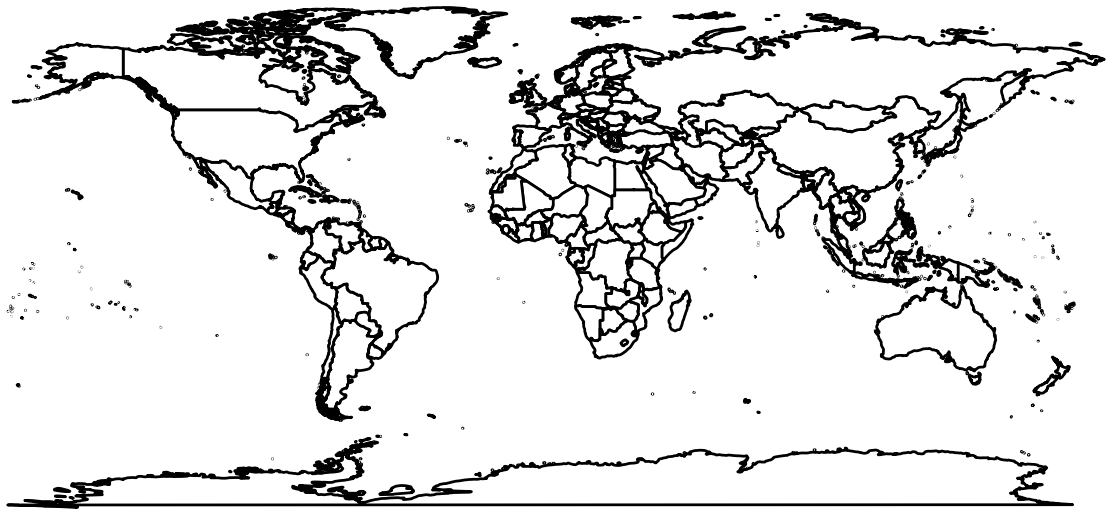


図16　様々なグラフ要素：例 2

- 例 3

  以下のようにグラフ要素を追加しています。

  – 座標系：極座標を使う。

  – テーマ：背景をなくす。

```
p_map + coord_polar()
```

図17　様々なグラフ要素：例 3