

応用プログラミング3
**第6回 ggplot2パッケージに
よるグラフィックス作成 (2)**

専修大学ネットワーク情報学部
田中健太

1. 課題の振り返り

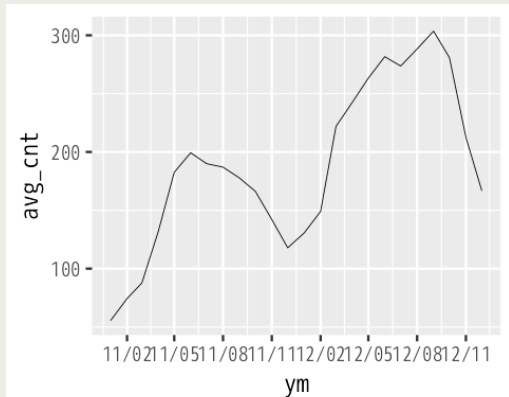
2. さまざまな グラフィックス (2)

2

今回も、引き続きggplot2の機能を紹介していきます。

2.1 折れ線グラフ

```
df_monthly <- df %>%  
  group_by(ym = lubridate::floor_date(dteday, "month")) %>%  
  summarise(avg_cnt = mean(cnt))  
  
ggplot(df_monthly, aes(x = ym, y = avg_cnt)) +  
  geom_line(size = 0.3) +  
  scale_x_date(date_breaks = "3 month",  
    date_labels = "%y/%m")
```



3

折れ線グラフは、一般的にx軸に時系列を配置し、y軸の値の推移を表現します。ggplot2では、geom_line() 関数で作成します。ここでは、レンタル自転車データの利用者数について月別に平均を算出し、その推移をプロットしています。

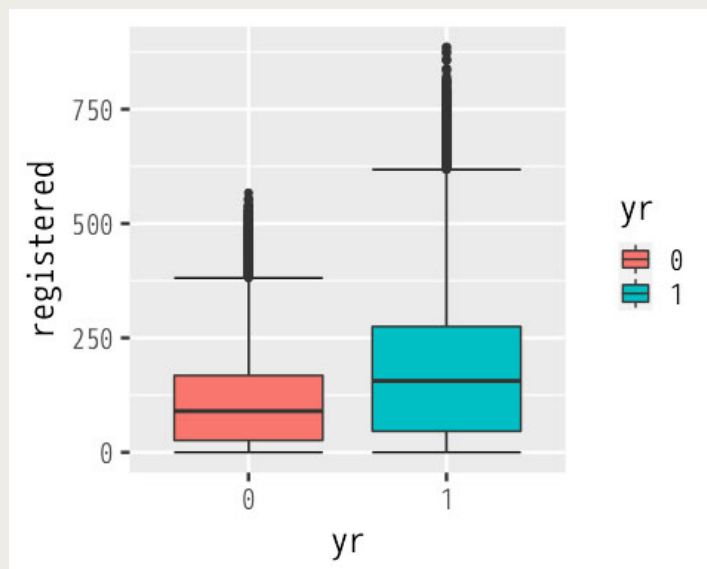
上記のプログラム例では、lubridateパッケージの floor_date() 関数を使っています。これは、日付 (Date) 型のデータを指定の単位で切り下げます。つまり、floor_date(as.Date("2023-10-31"), "month") という日付は、"2023-10-01" に変換されます。"year" と指定すれば、"2023-01-01" になります。他に、切り上げを行う ceiling_date() 関数や丸めを行う round_date() 関数もあります。実際には、日付の丸めはトラブルが起こりやすいので、シンプルに切り下げか切り上げを選択したほうがよいでしょう。

また、x軸の表示を scale_x_date() 関数で調整しています。Date型のデータに対応して、軸ラベルを "年/月" として3か月ごとに表示するなど、柔軟に調整できる関数です。なお、Rでは年月日までのDate型と、時刻まで含むdatetime型は異なっており、datetime型のデータにここで紹介した関数は適用できません。さらに言えば、組み込みのPOSIXct型とPOSIXlt型があるなど、Rにおける日付、時刻の扱いは複雑です。ggplot2に限りませんが、日時データを扱う際には、型をしっかりと確認する必要があります。lubridateパッケージのドキュメントは比較的わかりやすいので、原文を参照すると理解が深まるでしょう。

- 参考1: Make Dealing with Dates a Little Easier • lubridate
<https://lubridate.tidyverse.org/index.html>
- 参考2: 【R前処理講座32】 {lubridate} : 時間処理 【tidyverse】 - データサイエンスの道標
<https://datasciencemore.com/lubridate/>

2.2 箱ひげ図

```
ggplot(df, aes(x = yr, y = registered)) +  
  stat_boxplot(geom = "errorbar") +  
  geom_boxplot()
```



4

箱ひげ図は、ヒストグラムと同様、データの分布を表現するグラフです。
`geom_boxplot()` 関数で作成します。

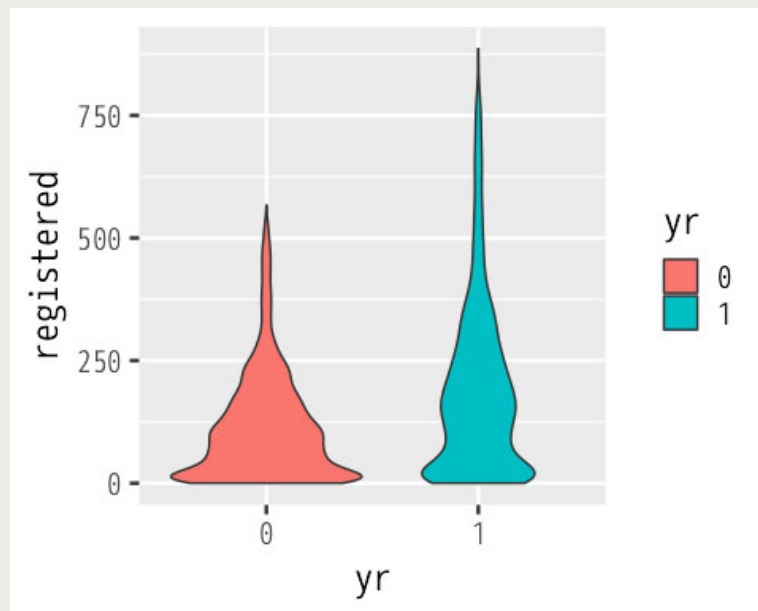
なお、`geom_boxplot()` 関数では、箱とひげの長辺 (?) のみが描かれ、ひげの両端の短辺 (横線) が描かれません。これを描くには、別途 `stat_boxplot(geom = "errorbar")` 関数を組み合わせる必要があります。順番としては、先に `stat_boxplot()` を記述し、続いて `geom_boxplot()` としたほうがよいでしょう。

🔄 順番を入れ替えると何が起こるか、確かめてみましょう。

`geom_boxplot()` 関数で描かれるひげの範囲は、標準で $\text{IQR} \times 1.5$ です。IQRは、「第3四分位数 (75%) - 第1四分位数 (25%)」で求められる値です。

2.3 バイオリンプロット

```
ggplot(df, aes(x = yr, y = registered, fill = yr)) +  
  geom_violin()
```



5

バイオリンプロットは、箱ひげ図に確率密度関数の要素を加えたもので、`geom_violin()` 関数で作成します。箱ひげ図では、「箱」の部分は長方形で、その区間に実際どのような密度でデータが集中しているかはわかりません。そこで、バイオリンプロットでは、箱の代わりにカーネル密度推定（参考1）を適用して得られた曲線を描きます。左右に同じ曲線を描いて結合した形状が、バイオリンのように見えるので、バイオリンプロットと呼ばれます。

`geom_violin()` 関数にはさまざまなオプションがあります。`trim` オプションはデフォルトで `TRUE` になっており、データの両端（最小値と最大値）で曲線を切断します。また、`scale` オプションは、複数のバイオリンプロットを描く際に、何を揃えるかを指定するもので、デフォルトでは `"area"` になっており、それぞれのグラフの面積が等しくなるように調整されます。他に、データの数に比例させる `"count"` やグラフの幅を等しくする `"width"` が指定できます。

箱ひげ図同様、`stat_boxplot()` 関数を記述することで、IQRに基づくひげを書き加えることができます。バイオリンプロットの場合、`geom_violin()` の後に記述したほうが、わかりやすいかもしれません。

- 参考1: カーネル密度推定とは？ - 統計ER
<https://toukeier.hatenablog.com/entry/kernel-density-estimation>
- 参考2: ggplot2でviolinplotを描く - sugioka_R
<https://sugioka.wiki.fc2.com/wiki/ggplot2%E3%81%A7violinplot%E3%82%92%E6%8F%8F%E3%81%8F>

2.4 グラフを重ねる

```
# from https://stackoverflow.com/a/51844068
scaleFactor <- max(df_monthly[["avg_temp"]]) /
max(df_monthly[["avg_cnt"]])

colors <- c("avg_temp" = "red4", "avg_cnt" = "blue4")
ggplot(df_monthly, aes(x = ym)) +
  geom_line(aes(y = avg_temp, colour = "avg_temp"),
    size = 1.5) +
  geom_line(aes(y = avg_cnt * scaleFactor,
    colour = "avg_cnt"), size = 1.5, alpha = 0.6) +
  scale_y_continuous(sec.axis = sec_axis(~./ scaleFactor,
    name = "cnt")) +
  scale_colour_manual(values = colors) +
  theme(legend.position = "bottom")
```

6

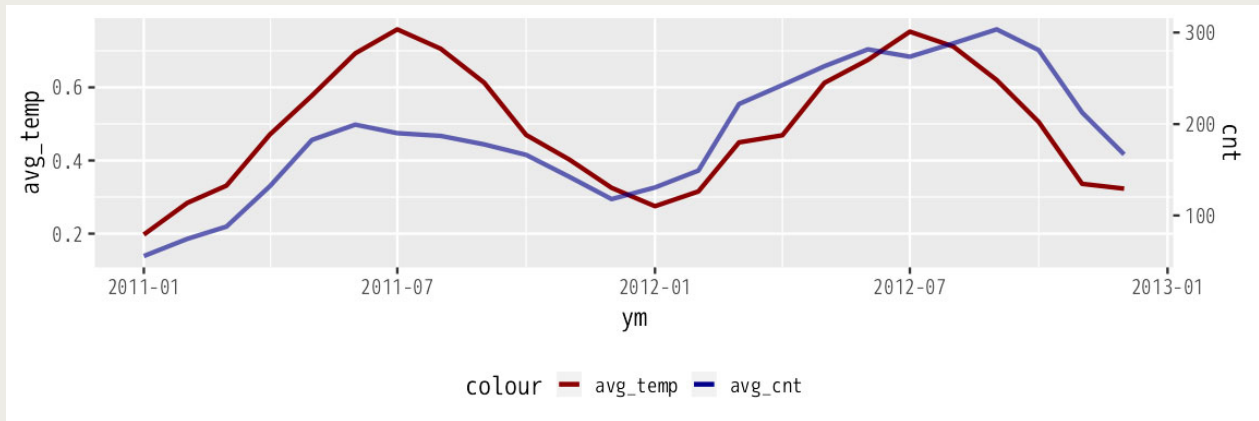
複数の折れ線グラフなどを重ねて描画したい場合、基本的には `geom_*()` 関数を `+` で繋げて記述すれば良いです。このプログラム例は、やや複雑なことをしていますが、単位の異なるデータを、左右の軸を変えて重ね描きしています。はじめに、標準化された気温 (`avg_temp`) と利用者数 (`cnt`) を同じ範囲に描くために、両者の比 (`scaleFactor`) を取っています。

`geom_line()` 関数でグラフを描く際に、利用者数にこの値を掛けることで、範囲を揃えています。一方で、軸の表示では実際の値を使いたいので、`scale_y_continuous()` 関数の `sec.axis` (Second Axis; 第2軸) オプションで再変換した値を指定しています。

このプログラム例については、海外のIT系掲示板であるStack Overflowで情報を得ました。知りたいことを英語で表現できる (単語の羅列で十分です) と、欲しい情報にアクセスできる確率が高まります。

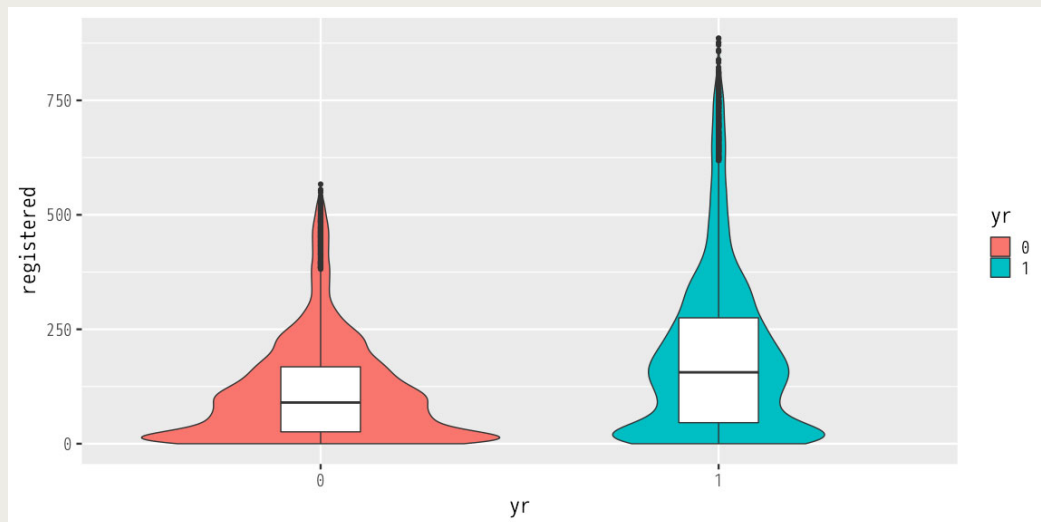
- 参考1: ggplot with 2 y axes on each side and different scales - Stack Overflow
<https://stackoverflow.com/a/51844068>
- 参考2: Specify a secondary axis — `sec_axis` • ggplot2
https://ggplot2.tidyverse.org/reference/sec_axis.html

2.4 グラフを重ねる



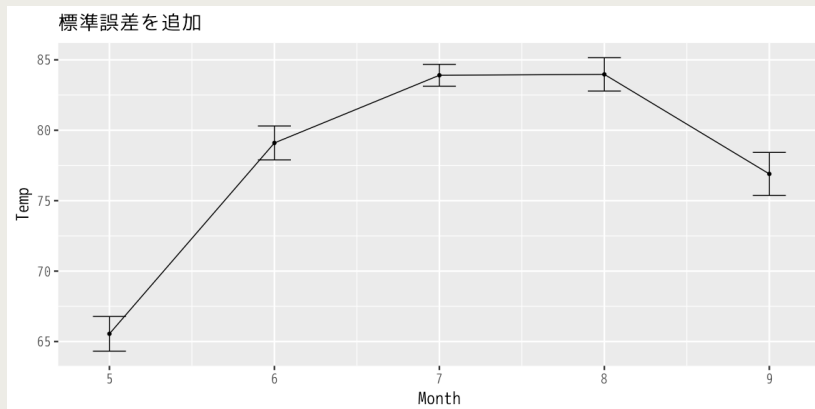
2.4 グラフを重ねる

```
ggplot(df, aes(x = yr, y = registered, fill = yr)) +  
  geom_violin() +  
  geom_boxplot(fill = "white", width = 0.2)
```



2.5 エラーバーなどを追加する

```
ggplot(airquality, aes(x = Month, y = Temp)) +  
  geom_line(stat = "summary", fun = "mean") +  
  stat_summary(fun = "mean", geom = "point") +  
  stat_summary(fun.data = "mean_se",  
              geom = "errorbar", width = 0.2) +  
  ggtitle("標準誤差を追加")
```



9

これも複数のグラフを重ねることの一種ですが、折れ線グラフや棒グラフなどに、誤差をあらわすエラーバーを追加したい場合があります。これには、`geom_summary()` 関数などを使います。この関数は、集計関数を適用した結果を `geom` オプションで指定した形で描画してくれます。このページのプログラム例では、平均を算出して点 (`point`) としてプロットし、重ねて標準誤差を算出し、誤差棒 (`errorbar`) としてプロットしています。`fun` オプションは通常の実行結果を、`fun.data` オプションは関数の実行結果を、`y`, `ymin`, `ymax` となるデータフレームとして返します。`mean_se()` 関数は、`ggplot2` が提供する平均と標準誤差を返す関数です。

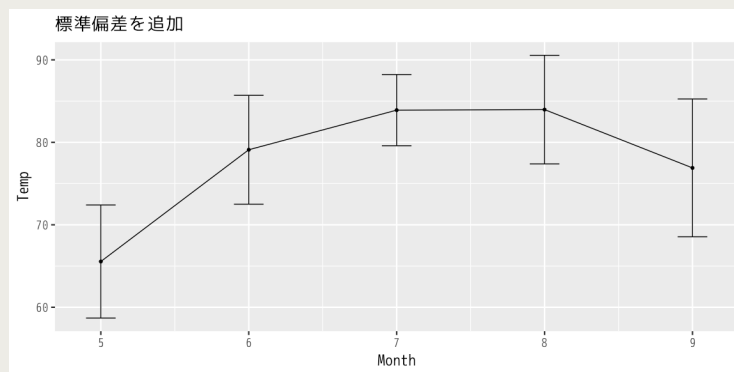
他に、平均、標準偏差、最小値 (下限)、最大値 (上限) などあらかじめ算出している場合は、`geom_errorbar()` 関数や `geom_pointrange()` 関数も利用できます。

```
geom_errorbar(aes(x = データ, y = データ, ymin=平均-標準偏差, ymax=平均+標準偏差))  
geom_pointrange(aes(x = データ, y = データ, ymin = 最小値, ymax = 最大値))
```

- 参考1: Summarise y values at unique/binned x — `stat_summary_bin` • `ggplot2`
https://ggplot2.tidyverse.org/reference/stat_summary.html
- 参考2: `ggplot2` の `stat_*`() 関数についてのまとめ - Qiita
<https://qiita.com/swathci/items/b08496d863bca4b479b3>
- 参考3: Vertical intervals: lines, crossbars & errorbars — `geom_crossbar` • `ggplot2`
https://ggplot2.tidyverse.org/reference/geom_linerange.html

2.5 エラーバーなどを追加する

```
ggplot(airquality, aes(x = Month, y = Temp)) +  
  geom_line(stat = "summary", fun = "mean") +  
  stat_summary(fun = "mean", geom = "point") +  
  stat_summary(fun.min = function(x){mean(x) - sd(x)},  
              fun.max = function(x){mean(x) + sd(x)},  
              geom = "errorbar", width = 0.2) +  
  ggtitle("標準偏差を追加")
```

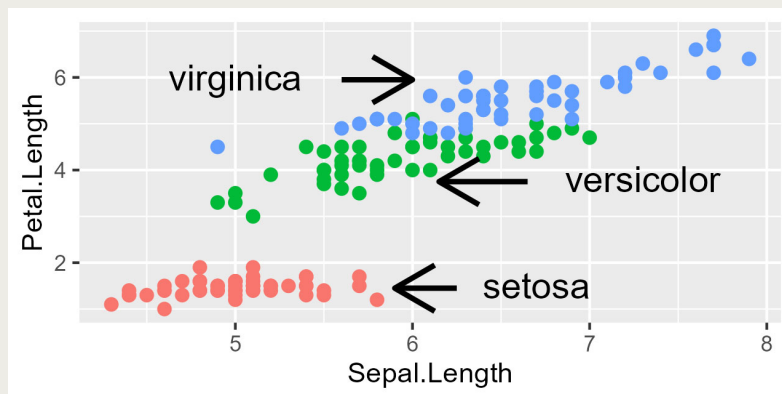


10

`stat_summary()` 関数の `fun*` オプションには、この例のように無名関数を渡すこともできます。ここでは、データ (Temp) が `x` として渡され、平均 - 標準偏差、平均 + 標準偏差を計算して、結果をもとに誤差棒を描いています。

2.6 テキストの描画

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length,  
  colour = Species)) +  
  geom_point(size = 4) +  
  annotate(geom="text", label="setosa", x=6.7, y=1.5, size=10) +  
  ...  
  annotate(geom = "segment", x = 6.25, xend = 5.9,  
    y = 1.45, yend = 1.45, size = 2, arrow=arrow()) +  
  ...
```



11

グラフの中に、説明を書き込みたいこともあります。もちろん、WordやPowerPointに貼り付けた後に、テキストボックスを重ねても良いのですが、ggplot2を使ってグラフ中に文字列を書き込むには、`annotate()` 関数を使います。

実際には、`annotate()` 関数は文字だけでなく、`geom_*()` 関数として提供されるほとんどの要素をグラフに重ねることができます。`geom` オプションに、描画したい要素を指定します。ここでは、`geom="text"` としてアヤメの品種を、`geom = "segment"` として、矢印を描いています。

なお、表示位置は座標を指定しますが、画面表示と画像として出力した際に、単位系が異なりずれることも多いので、目視での確認が必要です。

- 参考: Create an annotation layer — `annotate` • `ggplot2`
<https://ggplot2.tidyverse.org/reference/annotate.html>

3. 日本語の取り扱い

12

すでにここまでも、グラフ中で日本語を使用している部分がありますが、あらためて、ggplot2における日本語とフォントの扱いについて整理します。

Windows版でもデフォルトの文字コードがUTF-8化したことで、日本語を含む多言語の扱いが容易になりました。また、ggplot2も比較的容易に多言語フォントを扱えるため、近年では日本語を表示することにさほど苦労はなくなりました。

なお、ggplot2ではない、組み込みのグラフィックス関数においては日本語の表示は依然複雑です。以下に、参考情報を示します。

- 参考1: 須通り_統計_Rにおける作図時のフォント設定を極める
https://sudori.info/stat/stat_fig_font.html
- 参考2: Rで解析：MACとWindowsでRstudioのグラフの日本語文字化けを防ぐ簡単な方法
<https://www.karada-good.net/analyticsr/r-58/>
- 参考3: おまえはもうRのグラフの日本語表示に悩まない (各OS対応) - ill-identified diary
<https://ill-identified.hatenablog.com/entry/2020/10/03/200618> (あえて読み辛い文体になっています)

3.1 フォントの指定

- OSごとにフォントの扱いが異なる
- 個別に `theme_*(base_family = "フォント名")` と設定するか、`theme_set(theme_gray(base_family = "フォント名"))` とデフォルトを設定する
- raggデバイスにより、設定の手間が軽減された
- Posit CloudではIPAexフォントが標準でインストールされている

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length,
  colour = Species)) +
  geom_point() +
  labs(title = "irisデータのプロット", x = "がく片長",
  y = "花弁長") +
  theme_gray(base_family="IPAexMincho", base_size = 18)
```

13

前提として、コンピューターのOSごとに、フォントの扱いは異なります。フォントの形式としてもTrueType (ttf) やOpenType (otf)、さらにTrueType Collection (ttc) や拡張子はttcだがOpenType Collection (otcの場合も) など、さまざまです。さらに、OSがフォントを認識する仕組みも異なるため、Rを動かすOSで、使いたいフォントがどのように「見えて」いるか、どのような名前で指定できるかを確認する必要があります。

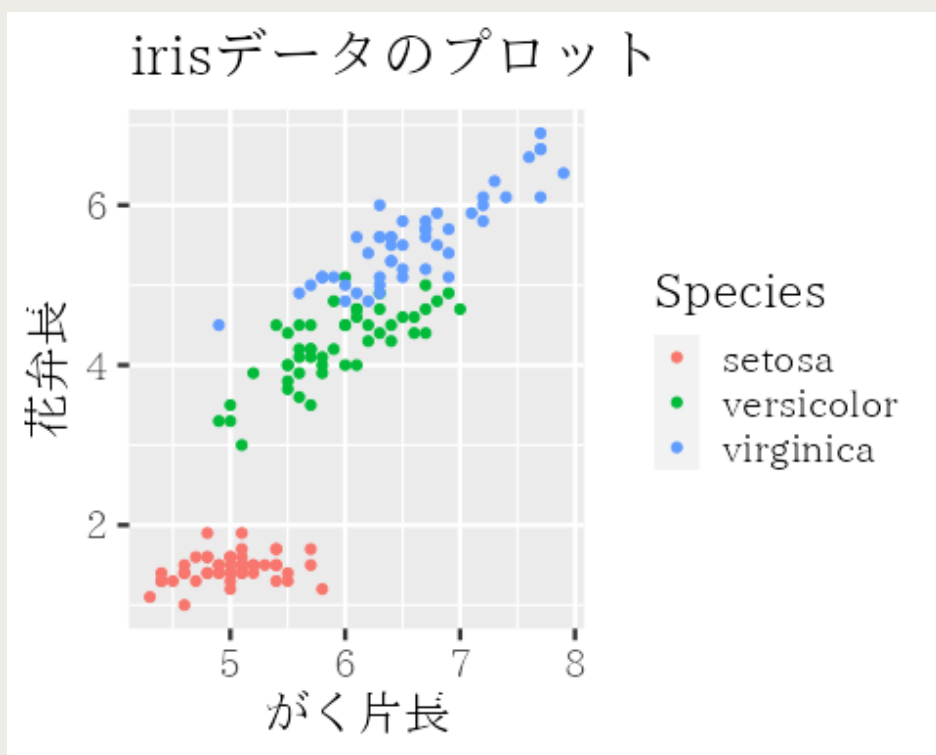
一方で、その部分がクリアできれば、ggplot2でのフォント指定は比較的容易です。`theme_*`() 関数の中で、`base_family="フォント名"` と指定するだけです。フォント名には、OSが認識する名前を指定します。例えば、macOSには標準でヒラギノという高品質フォントがインストールされており、以下のように指定できます。

```
theme_*(base_family="HiraginoSans-W3") # W = 0-9
```

また、Posit Cloudにはオープンソースの日本語フォントIPAexフォント (<https://moji.or.jp/ipafont/>) がインストールされており、スタンダードな明朝体とゴシック体が使用できます。それぞれ、IPAexMincho, IPAexGothic と指定します。

プログラム (R Markdownドキュメント) の中で複数のグラフを作成する場合は、事前に `theme_set()` 関数でフォントを指定すると、以降に作成するグラフ全てでフォントなどの設定が引き継がれます。R Markdownの場合、`setup` チャンクの中で設定するとよいでしょう。

3.1 フォントの指定



4. 軸やラベル、凡例のカスタマイズ

15

すでに一部使用していますが、ggplot2では、さまざまな関数でグラフィックスの軸表示やラベルをカスタマイズできます。デフォルトのままのグラフィックスでは情報が読み取り辛い場合、適切な軸やラベルを設定することで意味が伝わりやすくなります。

ここでは、見た目をカスタマイズするためによく使われる関数を紹介します。なお、ひとつひとつ設定と効果をスライドで紹介するとページ数が多くなるので、詳しくはハンズオン用のプログラムを実行して確認してください。

4.1 軸のカスタマイズ

- 軸のタイプやラベル、凡例についても、+ でつなげる形で設定できる
- `labs(title, x, y)`: タイトル、軸ラベルを指定する
- `scale_*_*`(): x軸、y軸についてさまざまな指定を行う
 - `scale_{x, y}_continuous()`: 軸が連続値の場合
 - `scale_{x, y}_discrete()`: 軸が離散値の場合
 - `scale_x_date()`, `scale_x_datetime()`: 軸が日付・時刻の場合
 - `scale_{x, y}_log10()`: 対数軸として表示する場合
 - `scale_{x, y}_reverse()`: 逆順に表示したい場合
 - `scale_colour_*()`, `scale_fill_*()`: 塗り分けた色の設定

16

グラフィックスのx軸、y軸に表示するラベルについては、`labs()` 関数で指定します。どちらかの軸のみ指定する場合は、`xlab()` または `ylab()` 関数も使えます。

軸の表示をカスタマイズするには、`scale_*_*`() 関数を使います。実際のデータに合わせて多くの種類が用意されています。また、それぞれの関数のオプションもさまざまに指定できるため、詳しくはハンズオンプログラムおよび各関数のドキュメントを参照してください。

- 参考1: Position scales for continuous data (x & y) — `scale_continuous` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_continuous.html
- 参考2: Position scales for discrete data — `scale_x_discrete` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_discrete.html
- 参考3: Position scales for date/time data — `scale_date` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_date.html
- 参考4: Sequential, diverging and qualitative colour scales from ColorBrewer — `scale_colour_brewer` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_brewer.html
- 参考5: Continuous and binned colour scales — `scale_colour_continuous` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_colour_continuous.html
- 参考6: Discrete colour scales — `scale_colour_discrete` • ggplot2
https://ggplot2.tidyverse.org/reference/scale_colour_discrete.html

4.2 凡例のカスタマイズ

- 凡例は `aes()` 関数で `colour` や `fill` に指定した列が自動的に表示される
 - 配置は `theme(legend.position = "配置")` で指定する
 - 凡例を表示しない場合、
`scale_*_(guide = FALSE)` や
`theme(legend.position = "none")` とする
 - ラベルを指定したい場合、
`scale_*_(labels = ラベル)` とする
 - タイトルを指定したい場合、
`scale_*_(title = タイトル)` とする

17

グラフィックスに複数の要素 (条件など) が描画されている場合、それぞれを区別、説明するための凡例が必要です。ggplot2では、`aes()` 関数で `colour` や `fill` に描き分けのためのキー (列) を指定すると、自動的に凡例を出力します。色や形状などそれぞれの凡例の位置を調整したい場合は、`theme()` 関数で `legend.position = 配置` として指定できます。配置は、`"top"` や `"bottomleft"` といったように文字で指定したり、`c(0.8, 0.2)` といったように座標で指定できます。また、`colour` や `fill` に指定した列の値ではない別のラベルを示したい場合、軸を設定する際に `labels` オプションで文字列ベクトルなどを指定します。同様に、凡例のタイトルも `title` オプションで指定できます。

凡例のサイズやその他の全般的なカスタマイズについては、次の `theme()` 関数で行います。

4.3 theme() 関数によるカスタマイズ

- theme() 関数でグラフィックス全体のさまざまな要素をカスタマイズできる
 - plot.title = element_text(size = ポイント数):
タイトル要素のカスタマイズ
 - axis.title.{x, y} = element_text(...):
軸ラベルのカスタマイズ
 - axis.text, axis.text.{x, y} = element_text(...):
軸目盛のカスタマイズ
 - legend.text, legend.title = element_text(...):
凡例タイトル、ラベルのカスタマイズ
- element_blank() と指定すると要素を表示しない

18

グラフィックスのタイトルや軸ラベル、凡例のサイズなど、構成要素をカスタマイズしたい場合、theme() 関数を使い、テーマの指定を上書きする形で変更できます。基本的には、要素名 = element_*(設定) といったように指定します。要素名および設定については、非常に範囲が広いので、以下のドキュメントを参考にしてください。

文字列の設定については、element_text(size = ポイント数) といったように指定します。また、例えば罫線や軸ラベル、目盛を表示したくない場合、その要素について element_blank() と指定すると表示されなくなります。他に、例えばタイトルのみフォントを変えたいといった場合、plot.title = element_text(family = "フォント名") というように指定できます。

- 参考: Modify components of a theme — theme • ggplot2
<https://ggplot2.tidyverse.org/reference/theme.html>

5. ggplot2のGUI

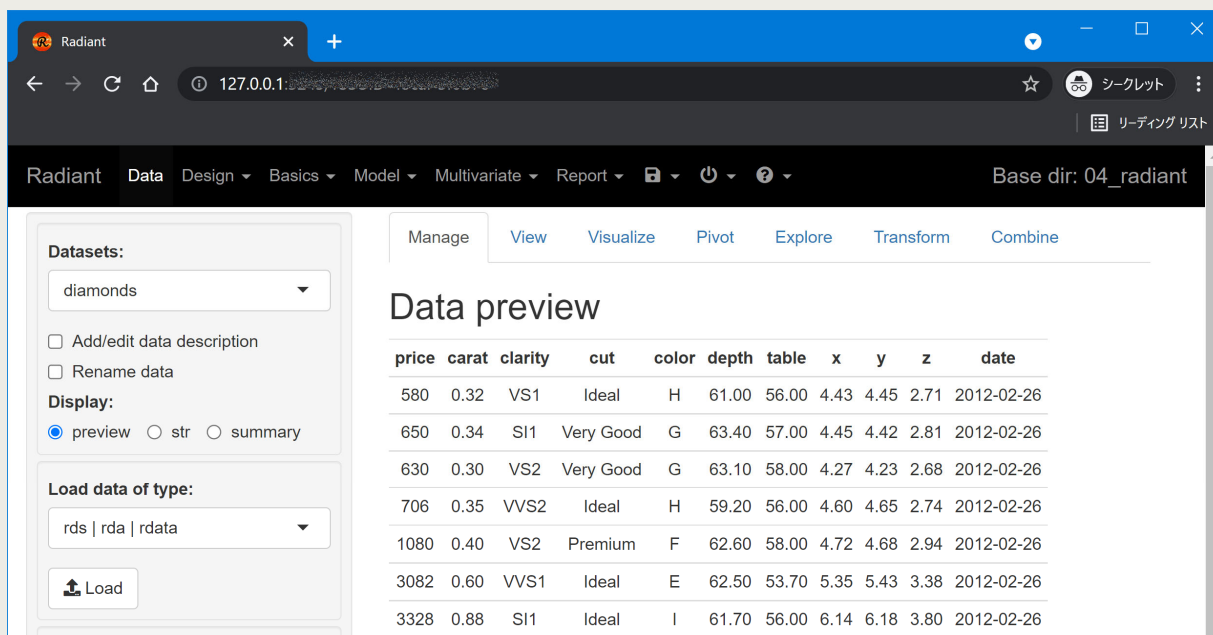
19

ここまで、2回にわたりggplot2の基本的な文法を紹介してきました。統一されているので、理解すれば応用が利きやすいですが、覚えることが多すぎるのも確かです。

そこで、最後にGUIでggplot2を用いたグラフィックスを作成できるパッケージを紹介します。ある程度までGUIでカスタマイズし、最後の細かい部分をプログラムを改変して設定する、といったようにすると効率的だと思います。

5.1 radiant

- 第4回で紹介した通り



The screenshot shows the Radiant web application interface. The top navigation bar includes 'Radiant', 'Data', 'Design', 'Basics', 'Model', 'Multivariate', 'Report', and a 'Base dir: 04_radiant' indicator. The left sidebar contains 'Datasets:' with a dropdown menu showing 'diamonds', and options to 'Add/edit data description' and 'Rename data'. Below this is a 'Display:' section with radio buttons for 'preview' (selected), 'str', and 'summary'. Further down is a 'Load data of type:' section with a dropdown menu showing 'rds | rda | rdata' and a 'Load' button. The main content area is titled 'Data preview' and displays a table of diamond data.

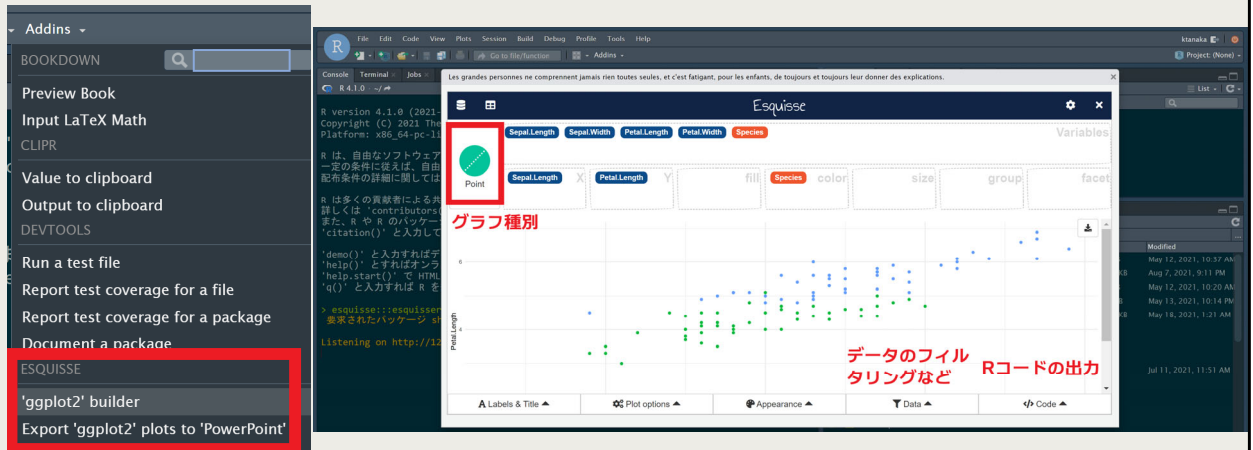
price	carat	clarity	cut	color	depth	table	x	y	z	date
580	0.32	VS1	Ideal	H	61.00	56.00	4.43	4.45	2.71	2012-02-26
650	0.34	SI1	Very Good	G	63.40	57.00	4.45	4.42	2.81	2012-02-26
630	0.30	VS2	Very Good	G	63.10	58.00	4.27	4.23	2.68	2012-02-26
706	0.35	VVS2	Ideal	H	59.20	56.00	4.60	4.65	2.74	2012-02-26
1080	0.40	VS2	Premium	F	62.60	58.00	4.72	4.68	2.94	2012-02-26
3082	0.60	VVS1	Ideal	E	62.50	53.70	5.35	5.43	3.38	2012-02-26
3328	0.88	SI1	Ideal	I	61.70	56.00	6.14	6.18	3.80	2012-02-26

20

Radiantについては、以前紹介した通りです。グラフィックスはggplot2を使っていますが、出力されるプログラムはパッケージの独自関数を多用しているため、ややカスタマイズしづらいかもかもしれません。

5.2 esquisse

- <https://dreamrs.github.io/esquisse/>
- "スケッチ" のことだそう
- **esquisser(データフレーム)** として起動する
- RStudioのアドインも追加される



21

esquisse (スケッチ) は、GUIでグラフィックスが作成できる高機能なパッケージです。esquisser() 関数の引数にデータフレームを与えて実行すると、ウィンドウが表示されます。なお、ノートPCなどディスプレイサイズが小さいとウィンドウ全体が表示されないため、viewer = "browser" とオプションを指定し、ブラウザで表示したほうがよいかもしれません。

操作は直感的で、ドラッグアンドドロップでx軸、y軸や "color" にデータを指定すると、データの性質に合わせて自動的にグラフィックスが作成されます。グラフィックスの種類をクリックすることで、変更できます。ラベルやタイトルについては "Labels & Title"、その他の見た目については、"Appearance" や "Plot options" から変更できます。

作成したグラフィックスは、右側のダウンロードアイコンから画像としてダウンロードしたり、"Code" メニューでプログラムとしてコピーできます。

グラフィックスを作成するだけの目的であれば、非常に便利なパッケージです。

5.3 ggraptR

- <https://github.com/cargomoose/ggraptR>
- 複数のgeometryを重ねることができる



22

ggraptRも、GUIでグラフィックスを作成できるパッケージです。ただ、ここ2年ほどは更新されておらず、またパッケージ内の `aggregate()` 関数が組み込み関数とバッティングしているため、さまざまな集計を行っているとならぶるかもしれません。

あえて積極的に使う必要はないでしょう（じゃあ紹介しなければよいのでは）。

6. まとめ

6.1 今日の内容

- さまざまなグラフィックス
 - 折れ線グラフ
 - 箱ひげ図
 - バイオリンプロット
- グラフィックスのカスタマイズ
- ggplot2のGUI

6.2 次回までの課題

- Posit Cloudプロジェクト内の
`06_ggplot2_02_exercise.R` について、
指示に従ってプログラムを作成してください
- 編集したファイルは、ファイル一覧でチェックを
入れ、[more] メニューから [Export] を選択し、
[Download] ボタンを押してダウンロードして
ください
- ダウンロードしたファイルを、Classroomの
課題ページから提出してください
- 提出期限: 2023-11-06 23:59