

**応用プログラミング3**  
**第5回 ggplot2パッケージに  
よるグラフィックス作成（1）**

専修大学ネットワーク情報学部  
田中健太

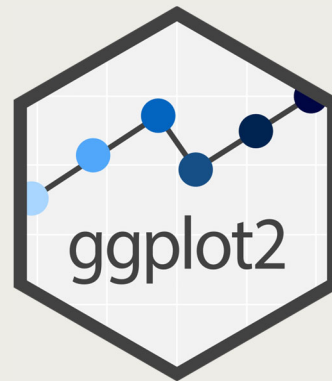
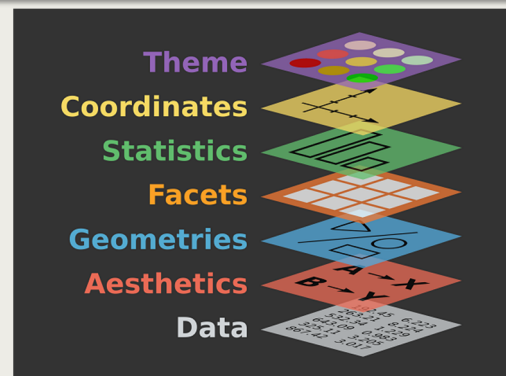
# 1. ggplot2パッケージの概要

1

今回と次回の講義では、tidyverseパッケージ群の中でも中心的な役割を担う、ggplot2パッケージについて紹介します。機能が豊富で、また実際にビジネス・研究の中で利用する機会も多いため、2回に分けて詳しく紹介します。

## 1.1 ggplot2とは

- **ggplot2はtidyverseを構成するパッケージの一つ**
- "Create Elegant Data Visualisations Using the **Grammar of Graphics**"
- "グラフィックスの文法" という考え方に則って開発されている
- 統一された書式で、さまざまな見栄えの良いグラフィックスを作成できる



2

ggplot2は、Rにおいて見栄えの良い高度な(複雑な)グラフィックスを、簡潔な記述で作成できるパッケージです。他のtidyverseパッケージ群と同様、Posit社のHadley Wickhamが中心となって開発されています。

ggplot2のコンセプトは **"Grammar of graphics"** (グラフィックスの文法) というもので、元々は2005年に発売された書籍 (Leland Wilkinson, 2005, "The Grammar of Graphics", Springer) で提唱された、データをグラフィックスで表現するための理論に由来します。

グラフィックスを階層的に構造化し、それぞれの階層を表現するプログラム (=文法) を統一することで、どのようなデータをどのようなグラフィックスとして表現する際にも、共通した書式で記述できるように設計されています。

もっとざっくりと表現すれば、「Rの標準のグラフィックス関数は、グラフの種類によって書式がバラバラで使いにくいから、統一した」というパッケージです。

## 1.2 ggplot2ベースの拡張パッケージ

- **ggplot2はRのグラフィックス作成におけるスタンダード**となり、また拡張性の高さからggplot2をベースにしたさまざまなパッケージが提供されている
- ベースとなるggplot2を知っていれば、拡張パッケージにおけるカスタマイズが行いやすくなる



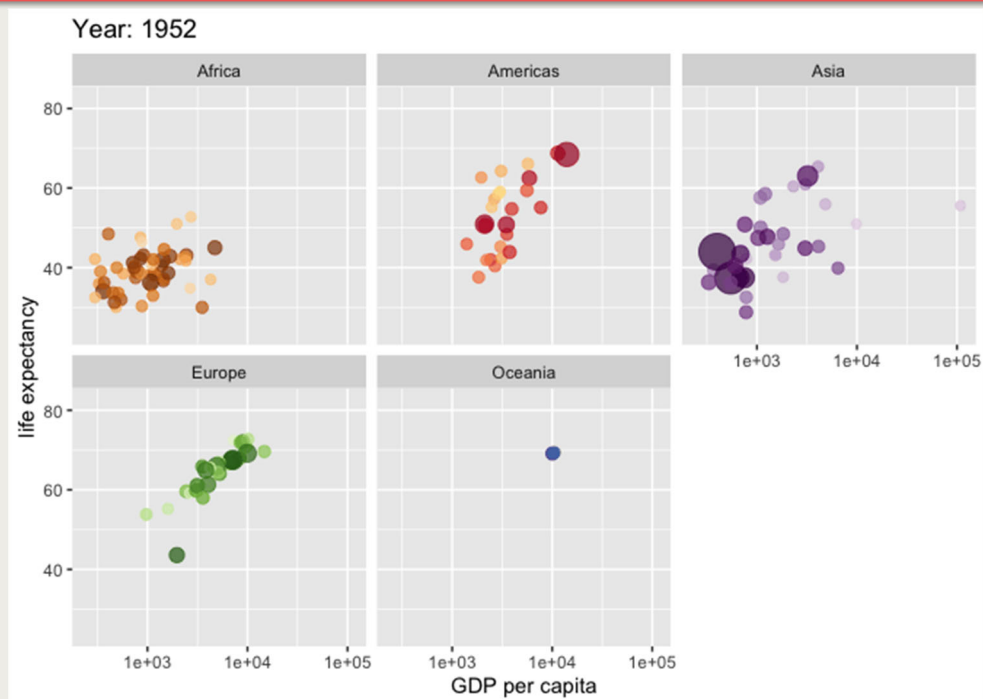
ggplot2に依存するパッケージ  
(見えないのはわかっていますが...)

3

ggplot2パッケージの公開以降、その豊富な機能と見通しの良い（整理された）設計のため、Rにおけるグラフィックス作成のスタンダードとして広く利用されています。また、さまざまな機能が高度にモジュール（部品）化されているため、それらを組み合わせたり改変して、独自の拡張を作成することも容易になっています。そのため、ggplot2をベースにしたさまざまなパッケージが多数開発・公開され、利用されています。それらのパッケージの多くは、ggplot2ベースであることを示すように `ggXXX` といった名前がつけられています。

スライドの図は、CRANに登録されている、“ggplot2に依存しているパッケージ”の一覧です。もちろん見えないわけですが、数千ものパッケージがggplot2を利用していることが伝われば、というものです。ggplot2の文法を知っていれば、派生した拡張パッケージも同じように使うことができ、グラフィックス表現の幅がさらに広がります。

## 1.2 ggplot2ベースの拡張パッケージ



A Grammar of Animated Graphics • gganimate <https://gganimate.com/>

4

PDFでは動かないですが、ggplot2を使い、いわゆるアニメーションGIFを作成できます。

<https://gganimate.com/reference/figures/README-unnamed-chunk-4-1.gif>

## 1.2 ggplot2ベースの拡張パッケージ



A Word Cloud Geom for ggplot2 • ggwordcloud <https://lepenec.github.io/ggwordcloud/index.html>

## 1.3 標準グラフィックスとの違い

- 「音楽性の違い」みたいなものでしかない
- データの特徴を適切に伝えるためのグラフィックスの本質は、標準機能であろうがggplot2であろうが変わらない
- ただ、**ggplot2には将来性があり、便利なので、これからRを学ぶなら、ggplot2をベースにしたほうが良い**



出典: <https://flowingdata.com/2016/03/22/comparing-ggplot2-and-r-base-graphics/>

6

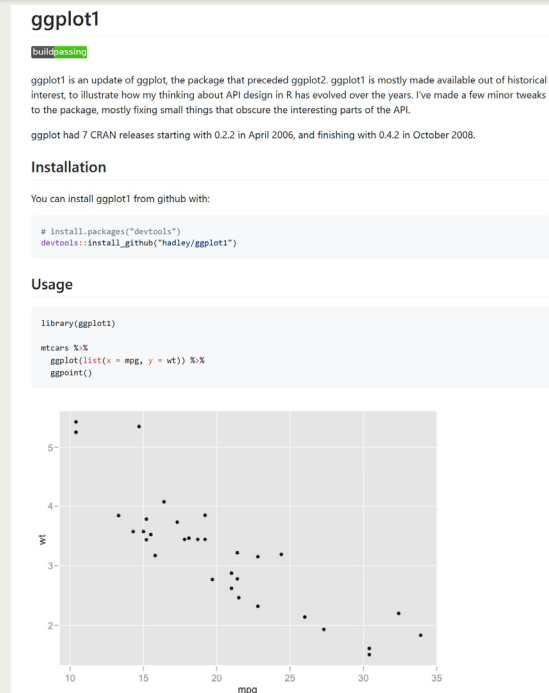
Rには元々、plot() 関数をはじめとした組み込みのグラフィックス関数が多数存在します。それらを使っても、シンプルで科学的に正確なグラフィックスが作成できます。組み込みのグラフィックス関数と、ggplot2の違いは何でしょうか。また、なぜあえてggplot2を使うのでしょうか。

率直に言えば、グラフを効率的に作成できて、意図が相手に伝われば組み込み関数であろうがggplot2であろうが、どちらでも構いません。好みの問題です。実際に、R界限では、市販の書籍やWeb上の記事に対して「組み込み関数で十分に正確なグラフィックスが作成できるのに、なぜわざわざggplot2を使うのか」と反発するユーザーもいます。

ただ、ggplot2は改良が続いており、またggplot2をベースとしたパッケージも増え続けています。何か研究やビジネスの中で作成したいグラフィックスがあって、それを作成できるパッケージを探したらggplot2ベースだった、ということは多く、ggplot2の文法は、2020年代にRを快適に利用するための必須知識と言えるでしょう。

## 1.4 ggplot1は…?

- あるにはあるらしい
- ggplot2以前に、2006年から2008年にかけて開発されていた
- 歴史的資料以上の意味はない



GitHub - hadley/ggplot1: Before there was ggplot2  
<https://github.com/hadley/ggplot1>

7

このページは単なる余談です。

ggplot2があるなら、ggplot1は？ と思いませんか。実際に、ggplot1という名前のパッケージは存在します。ggplot2と同じく、Hadley Wickhamが開発し、公開していました（GitHubリポジトリ自体は現在も存在します）。正確には、公開当時はggplotという名前でした。仮面ライダーと同じく、2号以降が登場してはじめて、最初のライダーが1号と名乗り始めるように、ggplot2ができてから、ggplot1という名前で扱われています。

- 参考: GitHubリポジトリ <https://github.com/hadley/ggplot1>

前述の "Grammar of graphics" という概念が発表されてすぐに、それを実装したプロトタイプと言えるでしょう。その後、どういう事情があったのかはわかりませんが開発が中止され、ggplot2としてリファクタリングされました。リポジトリにも "ggplot1 is mostly made available out of historical interest…" とあるように、現在では開発の歴史をあらわす資料以上の意味はありません。



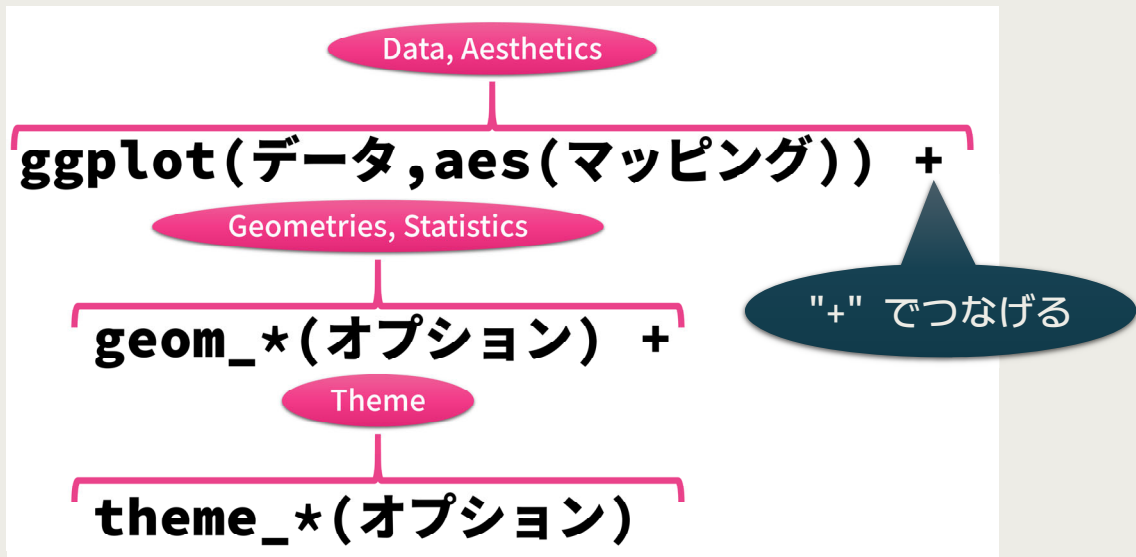
## 2. ggplot2の基本文法

8

ここから、ggplot2の使用法を紹介していきます。何度か述べているように、文法が統一されているので、覚えてしまえば、さまざまなグラフを容易に作成できるようになります。

## 2.1 ggplot2の典型的なプログラム

- ggplot2によるグラフィックスでは、基本的に `ggplot()` 関数、`aes()` 関数、`geom_*()` 関数が使われる
- 見た目を制御するには `theme_*()` 関数などを使う



9

まず、全体として「ggplot2によるグラフィックスはこのようなプログラムになる」という例を示します。ggplot2では、基本的に `ggplot()` 関数、`aes()` 関数、`geom()` 関数の3つを組み合わせて使用します。

`ggplot()` 関数は、ggplot2を使うということを宣言するような関数で、描画領域（グラフを描く白紙のキャンバス）を作成します。引数に、使用するデータフレームを指定します。

引数に指定したデータフレームの各列と、グラフィックスのx軸、y軸を対応付ける（マッピングする）のが `aes()` 関数です。主に `ggplot()` 関数の中で使用します。

次に、データをどのようなグラフィックスとして表現するのか、グラフの種類を指定するのが `geom_*()` 関数です。グラフの種類によって `geom_histogram()` や `geom_line()` などさまざまな関数を指定します。

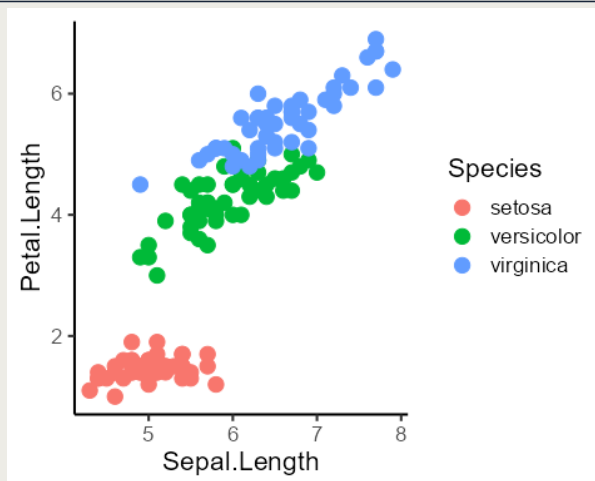
これらの関数に加え、グラフィックス全体の見た目をカスタマイズする場合、`theme_*()` 関数を指定します。標準で `theme_dark()` や `theme_minimal()` などが用意されています。また、追加のテーマが多数パッケージとして公開されているので、それらを使い、特定の雑誌（論文誌）に合わせたデザインのグラフも容易に作成できます。

また、ggplot2においては上記の関数を + でつないでいくという点に注意してください。tidyverseの特徴はパイプ (`%>%`) ですが、ggplot2では + です。

次のページ以降で、それぞれの関数について詳しく紹介していきます。

## 2.1 ggplot2の典型的なプログラム

```
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length,
  colour = Species)) +
  geom_point(size = 4) +
  theme_classic(base_size = 16)
```



## 2.2 データフォーマット

- 基本的にtidyverseに準じて **"tidy data"** を与える
- 特に、色や線種、ファセット（後述）などグラフを塗り分ける要因は1列でラベルとして記録する

| ID | group | type | value |
|----|-------|------|-------|
| 1  | 1     | A    | 71    |
| 2  | 2     | A    | 36    |
| 3  | 2     | B    | 59    |

11

ggplot2に与えるデータについて紹介します。組み込み関数では、ベクトルや行列も引数として扱いますが、ggplot2では、基本的にデータフレームを与えます。特に、"tidy data" と呼ばれるようなデータが求められます。tidy dataは、Hadley Wickhamが提唱した、コンピューターにとって扱いやすいデータ構造です。例えば、Wikipediaでは以下のようなデータがtidy dataである、とされています ( [https://ja.wikipedia.org/wiki/Tidy\\_data](https://ja.wikipedia.org/wiki/Tidy_data) )。

- 個々の変数が1つの列をなす
- 個々の観測が1つの行をなす
- 個々の観測の構成単位の類型が1つの表をなす
- 個々の値が1つのセルをなす

他に、西原史暁氏による解説が広く知られています。

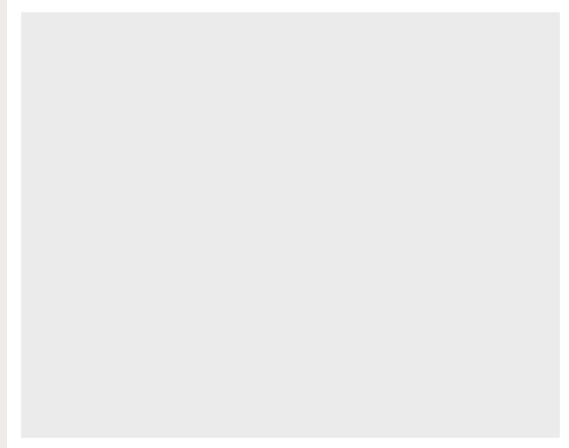
- 整然データとは何か <https://id.fnshr.info/2017/01/09/tidy-data-intro/>

tidy dataは、人間からすると縦に長く、冗長なイメージを持ちますが、一方でコンピューターからすると必要な情報を抽出しやすい構造です。ggplot2だけでなく、tidyverse自体が名前の通り、tidy dataを念頭に設計されているので、データを収集し記録する段階から、tidyになるように意識しておくといよいでしょう。

## 2.3 ggplot() 関数

- **ggplot() 関数**はグラフィックス領域（枠）のオブジェクトを作成する
- 描画対象のデータフレームと、次に紹介する **aes() 関数**によるマッピングを引数に指定する
- **関数単体では何も描画されない**

```
# library(tidyverse)
library(ggplot2)
p <- ggplot()
p
```



ggplot() 関数だけでは何も描画されない













12

次に、ggplot() 関数を紹介します。ggplot() 関数の動作はシンプルで、引数にデータフレームを与えると、グラフィックスオブジェクトを生成します。また、関数の中で aes() 関数を用いることで、データとx軸、y軸のマッピングを管理します。

ggplot() 関数を実行するだけでは、何も描画されません（真っ白な領域が表示されます）。また、**関数の実行結果をグラフィックスオブジェクトとして適当な変数に代入できます。**一般に、`p <- ggplot(...)` と `p` という名前を使うことが多いようです。

## 2.4 aes() 関数

- Aesthetic（美的）という関数
- 実際には、"x軸の値", "y軸の値", "色を塗り分ける基準"などをデータとマッピングする機能
- 代表的なものに x, y, colour, fill, group などがある

| x, y    | colour<br>color | fill        | linetype  | shape   | size    |
|---------|-----------------|-------------|---|---|---------|
| 軸のマッピング | 点や線の色           | 棒グラフの塗りつぶし色 | 線種  | 点種  | 線や点のサイズ |
| x = 列名  | "name"          | "name"      | solid<br>    | 0<br>  | 単位: mm  |
| y = 列名  | "#RRGGBB"       | "#RRGGBB"   | dashed<br>   | 1<br>  |         |
|         | NA (透明)         | NA (透明)     | dotted<br>   | 2<br>  |         |
|         |                 |             | dotdash<br>  | 3<br>  |         |
|         |                 |             | longdash<br> | 4<br>  |         |
|         |                 |             | twodash<br>  | 15<br> |         |

13

データと、グラフィックスにおける軸や色分けなどの要素の対応関係を指定するのが `aes()` 関数です。Aestheticという単語の略称で、美的という意味です。R関連の書籍では直訳した「審美的」という訳が当てられることも多いです。

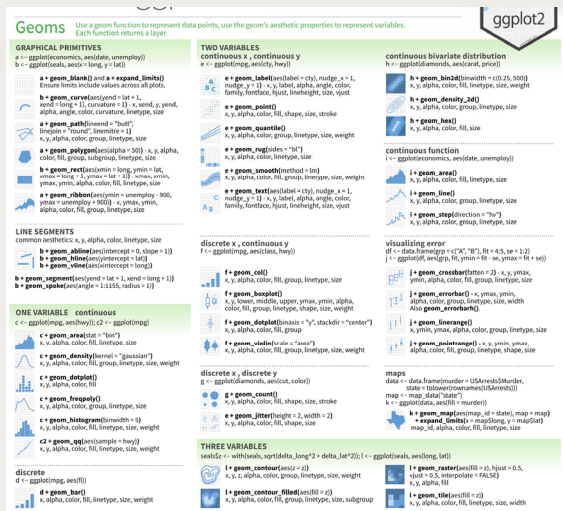
あまりピンと来ない言葉ですし、日本では「エステティック」という言葉が美容・マッサージ関係の意味で広く使われているため、イメージがしづらいですが、上述のようにデータとグラフィックスの基本要素を対応付けるもの、と捉えればよいでしょう。

`aes()` 関数は基本的に `ggplot()` 関数の中で使います。引数には、以下のようなパラメーターを指定します。

- x=x軸に配置する値
- y=y軸に配置する値
- colour=折れ線グラフを塗り分ける基準列
- fill=棒グラフを塗り分ける基準列
- linetype=線種を描き分ける基準列
- shape=散布図の点を塗り分ける基準列
- size=線や点のサイズ: 列の値に応じて可変させるよりは、size=6 といったように固定値を指定することが多いでしょう

すべては列挙できないので、ハンズオンや課題、ビジネスや研究での実践を通じて、慣れていきましょう。

## 2.5 geom\_\*() 関数



出典: "RStudio Data Visualization Cheatsheet"  
<https://www.rstudio.com/resources/cheatsheets/>

- Geometry（幾何学、配置）という関数
- `ggplot()`, `aes()` 関数で指定したデータを、どのような形状（図形）で描画するかを決定する
- 代表的なものに `geom_line()`, `geom_point()`, `geom_bar()`, `geom_histogram()` などがある

14

実際に「何グラフ」を描画するかを指定するのが `geom_*()` 関数群です。Geometryという単語に由来します。種類に応じて `geom_histogram()` や `geom_bar()` など多数存在します。スライドの図は、RStudioのWebサイトで公開されているチートシート（早見表）です。データを、どのグラフィックスで表現すれば、最も意図が伝わるか、を考えて適切なものを選びましょう。標準で提供されているものについては、次のページにまとめています。

- 参考: Posit Cheatsheets <https://posit.co/resources/cheatsheets/>

また、`ggplot2`をベースとした拡張パッケージにおいても、独自のGeometry（例えば、`ggwordcloud`パッケージでワードクラウドを出力する `geom_text_wordcloud()` など）が定義されています。

ggplot2が標準で対応しているグラフィックスは以下の通りです。

- プリミティブ (主に内部で使用する関数)
  - `geom_blank()`: 点や線を描画せず、描画領域だけを表示する
  - `geom_curve()`: 指定した座標間を結ぶ曲線を描画する
  - `geom_path()`: 指定した座標間を結ぶ直線を描画する
  - `geom_polygon()`: 指定した範囲を多角形で囲んで塗りつぶす
  - `geom_rect()`: 指定した範囲を矩形で囲んで塗りつぶす
  - `geom_ribbon()`: 指定した下限、上限からなる領域を塗りつぶす
  - `geom_abline()`: 傾きaと切片bからなる直線を描画する
  - `geom_hline()`: y軸の指定した位置に水平線を描画する
  - `geom_vline()`: x軸の指定した位置に垂直線を描画する
  - `geom_segment()`: a点からb点を結ぶ直線を描画する
  - `geom_spoke()`: 指定した座標から半径 (長さ)、角度を指定して直線を描画する
- 1変量の連続データを描画する
  - `geom_area()`: x, yの対応する座標までの領域を塗りつぶす
  - `geom_density()`: 変数の密度を描画する
  - `geom_dotplot()`: データの頻度を点の積み上げであらわす
  - `geom_freqpoly()`: データの頻度を折れ線であらわす
  - `geom_histogram()`: ヒストグラムを描画する
  - `geom_qq()`: Q-Qプロット (実測値と理論値・確率分布を比較するグラフ) を描画する
- 1変量の離散データを描画する
  - `geom_bar()`: 棒グラフを描画する (デフォルトでは高さは自動集計される頻度)
- 2変量の連続データ (x, yとも) を描画する
  - `geom_label()`: x, yの対応する座標にラベルを描画する
  - `geom_jitter()`: 散布図で点が重ならないように適宜誤差を加えて描画する
  - `geom_point()`: 散布図を描画する
  - `geom_quantile()`: 各四分位点ごとに回帰直線を描画する
  - `geom_rug()`: ラグプロット (1次元データのバーコード風グラフ) をx, yそれぞれについて描画する
  - `geom_smooth()`: x, yの関係性を平滑化関数 (曲線) で描画する
  - `geom_text()`: x, yの対応する座標に文字を描画する

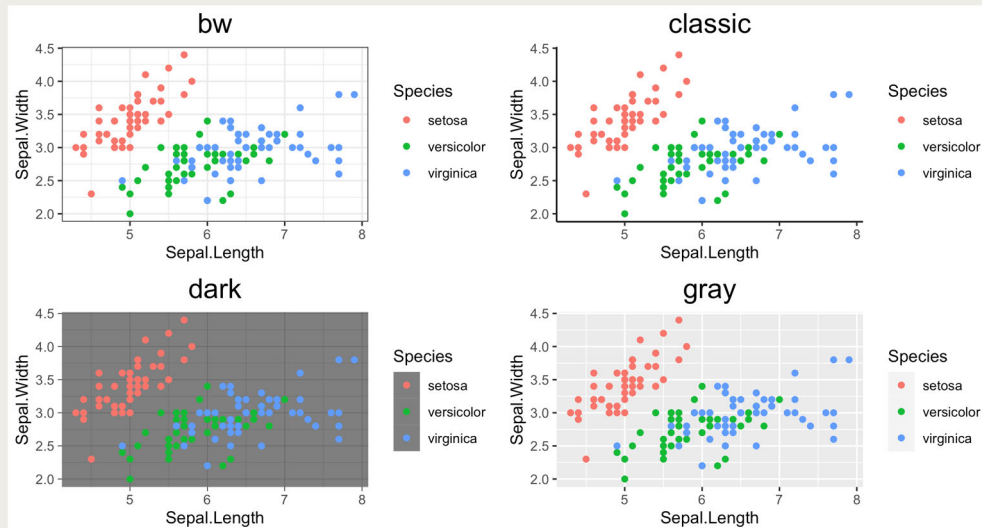
(続きます)



- 2変量の離散・連続データ (x: 離散, y: 連続) を描画する
  - `geom_col()`: `geom_bar()` のy軸にデータ中の値を使用する場合の別名
  - `geom_boxplot()`: 箱ひげ図を描画する
  - `geom_violin()`: バイオリンプロットを描画する
  - `geom_line()`: 折れ線グラフを描画する
- 2変量の離散データを描画する
  - `geom_count()`: x, yの対応する座標における頻度を丸の大きさと描画する
- 連続した2変量の分布を描画する
  - `geom_bin2d()`: 指定した領域サイズごとの密度を四角形で描画する
  - `geom_density2d()`: 変数の密度を描画する
  - `geom_hex()`: 指定した領域サイズごとの密度を六角形で描画する
- 連続した2変量からなる関数を描画する
  - `geom_step()`: ステップ状に変化するデータを描画する
- 誤差を可視化する
  - `geom_crossbar()`: 箱ひげ図の「箱」を描画する
  - `geom_errorbar()`: エラーバーを描画する
  - `geom_linerange()`: データの範囲を直線を描画する
  - `geom_pointrange()`: データと下限、上限を描画する
  - `geom_map()`: x, yを緯度、経度と見なして地図を描画する
- 3変量の離散・連続データを描画する
  - `geom_contour()`: 等高線図を描画する
  - `geom_raster()`: x, y座標におけるzの値をヒートマップ状に描画する
  - `geom_tile()`: x, y座標におけるzの値をタイル状に描画する

## 2.6 theme\_\*() 関数

- グラフィックス全体の見栄えを制御する
- 標準で10個のテーマが利用できる
- **ggthemes**パッケージなどでさらに拡張できる



17

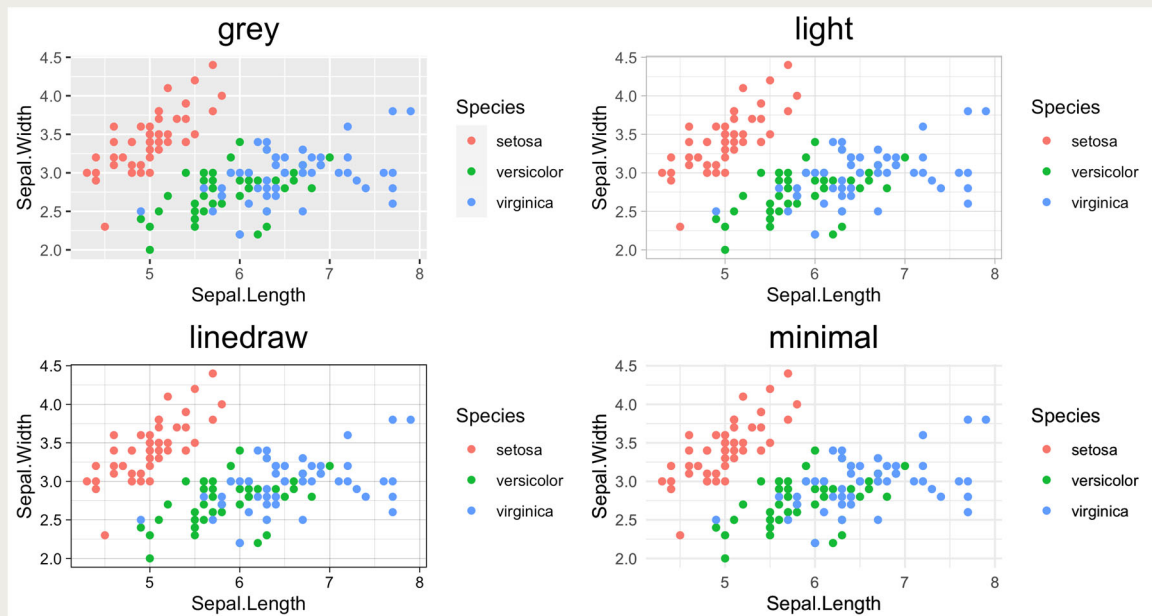
ggplot2では、グラフィックス全体の見栄えを、テーマというかたちで制御できます。作成する文書に合わせて、さまざまなデザインでグラフィックスを出力できます。テーマは、geom\_\*() 関数の後に、theme\_\*() 関数で指定します。標準で10個のテーマが用意されています。なお、theme\_gray() と theme\_grey() はアメリカ英語とイギリス英語のつづりの違いで、効果は同一です。

また、別のパッケージとして、数多くの追加テーマが開発、公開されています。例えば ggthemes パッケージなどが、10を越える追加テーマを提供しています。他に、ggthemr パッケージなどがあります。

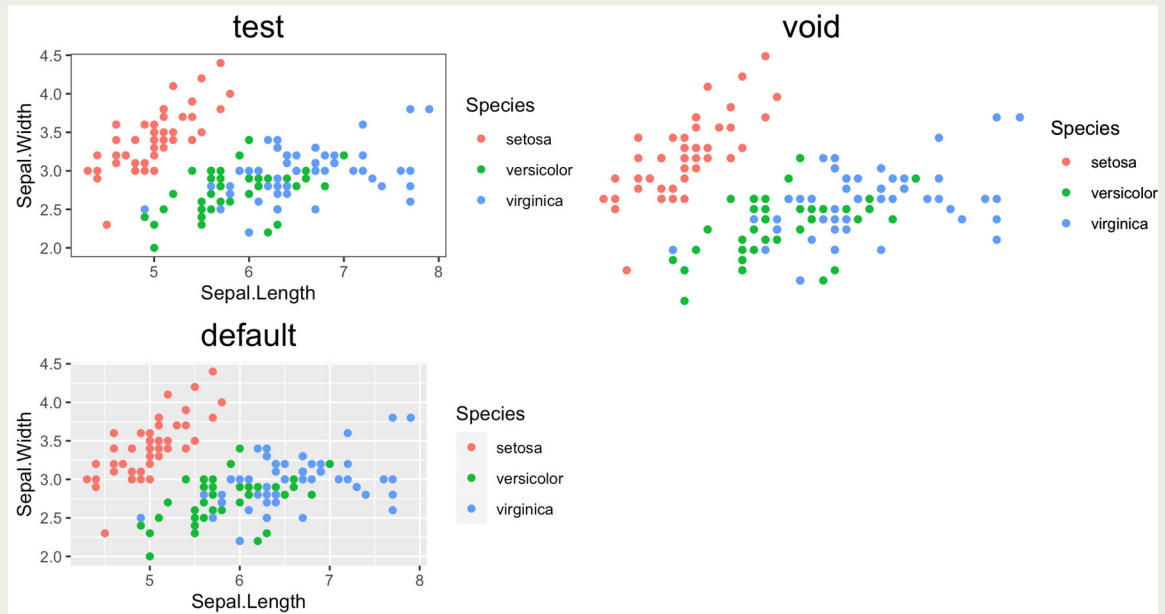
- 参考1: ggthemes パッケージ <https://jrnold.github.io/ggthemes/>
- 参考2: ggthemr パッケージ <https://github.com/Mikata-Project/ggthemr>

それぞれのテーマについて、関数にオプションを指定し、細かなカスタマイズを行うこともできます。また、特定のテーマに依存しない独自の設定は、theme() 関数で行うことができます。

## 2.6 theme\_\*() 関数



## 2.6 theme\_\*() 関数



# 3. グラフィックスの保存

20

ここでは、ggplot2を用いて作成したグラフィックスを保存する方法を紹介します。

## 3.1 RStudioのメニューから保存する

- 作成したグラフィックスは画像として出力したい
- RStudioの "Export" メニューで保存できる
- RMarkdownではコンパイル時に自動的に埋め込まれる



21

はじめに、RStudioのメニューから保存する方法を紹介します。この方法は、組み込み関数によるグラフィックスも保存できます。

RStudioでは、グラフィックスが右下ペインの "Plots" タブに表示されます。"Plots" タブには "Export" メニューがあり、ここからグラフィックスを画像 (PNG) やPDF、クリップボードへのコピーとして保存できます。ただし、画像のサイズやフォント設定などが画面上の表示と異なって保存されることもあるので、確認が必要です。

また、R Markdownで作成したHTMLドキュメントは、グラフィックスがBase64エンコードで埋め込まれているので、ブラウザ上で右クリックから「名前を付けて保存」できます。

## 3.2 ggsave() 関数

- `ggsave()` 関数で任意の形式で出力できる
- PNG形式については、最近新しいグラフィックスデバイスとして`ragg`が使用されるようになっている
- `ragg`では、日本語フォントの設定をしなくても文字化けしない
- R Markdownでもチャンクオプションに `dev='ragg_png'` とすれば`ragg`を使用できる

```
library(ggplot2)
p <- ggplot(iris,
  aes(x = Sepal.Length,
    y = Petal.Length,
    colour = Species)) +
  geom_point(size = 3)

ggsave("ggplot_example.png",
  p, device = ragg::agg_png)

ggsave("ggplot_example.pdf",
  p)
```

22

ggplot2専用の、`ggsave()` 関数として `ggsave()` があります。使い方はシンプルで、引数に (1) 出力先のファイル名、(2) 出力したいオブジェクト名の順で指定します。先にファイル名を記述するのは、少し直感と外れるところがありますが、順序が決まっています。あるいは、`plot=オブジェクト名`、`filename="ファイル名"` と引数名を付ければ、順序は問われません。

ファイル名に指定した拡張子によって、自動的にファイル形式が決まります。拡張子 (ファイル形式) は、Rがサポートしているpng, jpg, bmp, svgなどを指定できます。

また、`device=デバイス名` オプションを指定すると、各種のパッケージが提供する高度な出力先 (グラフィックスデバイス) を選択できます。近年では、`ragg`パッケージが従来よりもきれいで、日本語フォントも手軽に扱えるデバイスを提供しているので、それらをスライドの例のように指定することが増えています。RStudioでも、標準のグラフィックスデバイスを`ragg`に指定することができます。

R Markdownのチャンクから出力するグラフィックスについても、チャンクオプションに `dev=ragg_png` というように指定すると、`ragg`パッケージが提供するデバイスを使用できます。

## 4. さまざまな グラフィックス（1）

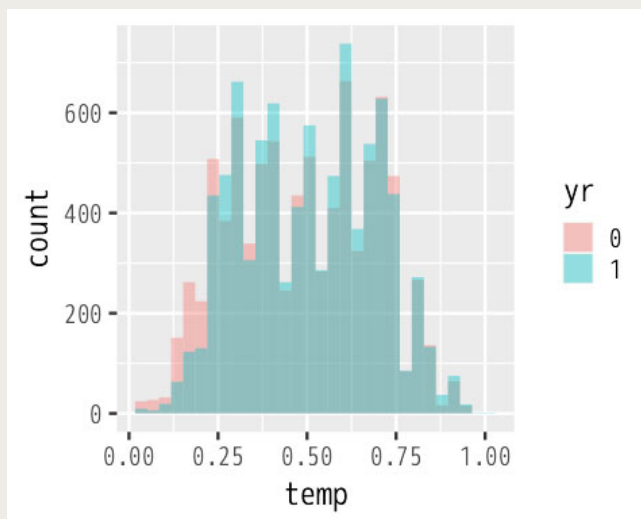
23

ここから、ggplot2で作成できるさまざまなグラフィックスのプログラム例を紹介していきます。この教材では、1ページに収まる範囲での説明にとどめていますので、より詳しい例は、ハンズオン資料 (05\_ggplot2\_01\_handson.R) を参照してください。



## 4.1 ヒストグラム

```
library(tidyverse)
df <- read_csv("day.csv")
ggplot(df, aes(x = temp, fill = yr)) +
  geom_histogram(position = "identity", alpha = 0.4)
```



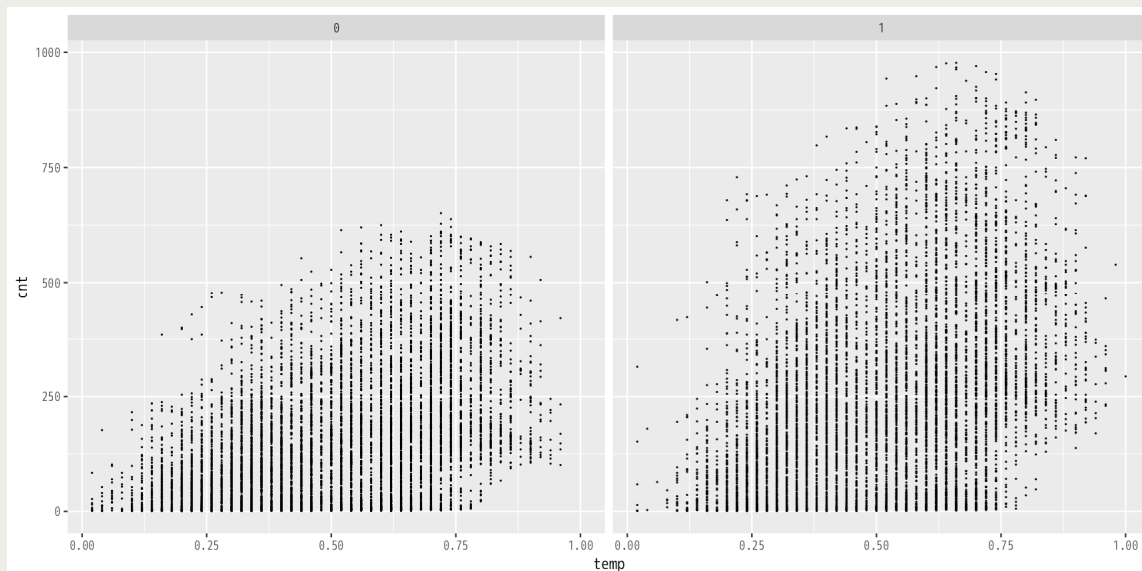
24

はじめにヒストグラムです。ヒストグラムは、ジオメトリに `geom_histogram()` を指定して作成します。`aes()` 関数で `x` 引数にデータの列名を指定します。また、`fill` 引数 (グループ化変数) を指定している場合、`position="identity"` とすることで、複数のヒストグラムを重ねて描くことができます。その際、透明度 (`alpha`) を適切に指定するとよいでしょう。

- 参考: `geom_histogram` | `ggplot2` でヒストグラムを描く方法  
[https://stats.biopapyrus.jp/r/ggplot/geom\\_histogram.html](https://stats.biopapyrus.jp/r/ggplot/geom_histogram.html)

## 4.2 散布図

```
ggplot(df, aes(x = temp, y = cnt)) +  
  geom_point(size = 0.5) + facet_wrap(~yr)
```



25

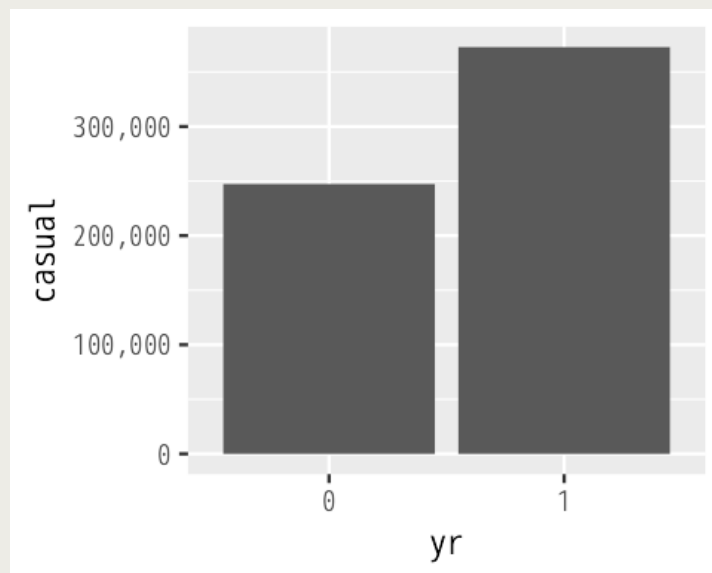
散布図は、`geom_point()` 関数で作成します。`aes()` 関数の `x` と `y` 引数に、それぞれデータを指定します。

また、ここでは変数 `yr` の値 (グループ) ごとに散布図を描き分けるよう、`facet_wrap()` 関数をつなげています。書式は、チルダ (`~`) を挟んで前が行の指定、後ろが列の指定です。つまり、ここでは `~yr` としているので、列方向 (横) に散布図が並びます。`facet` によるグラフィックスの分割については、次回あらためて取り上げます。

- 参考1: `geom_point` | `ggplot2` で散布図と回帰直線を作成する方法  
<https://stats.biopapyrus.jp/r/ggplot/geom-point.html>
- 参考2: `ggplot` の `facet_grid()`, `facet_wrap()` の使い方 | Memo on the Web  
[http://motw.mods.jp/R/ggplot\\_facet.html](http://motw.mods.jp/R/ggplot_facet.html)

## 4.3 棒グラフ (1)

```
ggplot(df, aes(x = yr, y = casual)) +  
  geom_bar(stat = "summary", fun = "sum") +  
  scale_y_continuous(labels = scales::comma)
```



26

棒グラフは、`geom_bar()` 関数で作成します。組み込み関数の `barplot()` と違い、集計していないデータを与えることもできます。デフォルトでは、`aes()` 関数で "男性, 女性, 男性..." といった定性データを `x` に指定すると、その度数を集計して棒の高さにします。

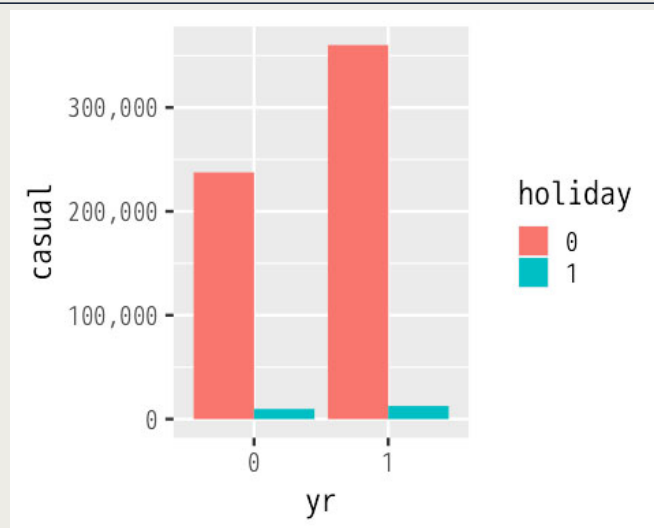
一方、定量データ (数値) の合計や平均を棒の高さとしたい場合は、`geom_bar(stat="summary", fun="集計関数")` といったように指定します。集計関数には `sum` や `mean` を指定します。カッコは不要です。

また、集計済みのデータをプロットしたい場合は、`stat="identity"` とします。

スライドのプログラム例の `scale_y_continuous()` はこの場合必須ではありませんが、縦軸のラベルがデフォルトでは指数表記 (`1e-6` など) になるため、`scales` パッケージの `comma()` 関数と組み合わせ、「縦軸は連続値で、ラベルは適宜コンマで区切る」という指定をしています。`scale_*` については、次回詳しく紹介します。

## 4.4 棒グラフ (2)

```
ggplot(df, aes(x = yr, y = casual, fill = holiday)) +  
  geom_bar(stat = "summary", fun = "sum",  
    position = "dodge") +  
  scale_y_continuous(labels = scales::comma)
```



27

グラフをグループごとに塗分けたい場合は、`fill` オプションにグループを指定します。その際、デフォルトでは積み上げ棒グラフとして表示されます。これを、横に並べて表示したい場合は、`position = "dodge"` オプションを指定します。

また、全体の割合を100%とした帯グラフを描きたい場合は、`position = "fill"` とします。

- 参考: `geom_bar` | `ggplot` で棒グラフを描く方法  
[https://stats.biopapyrus.jp/r/ggplot/geom\\_bar.html](https://stats.biopapyrus.jp/r/ggplot/geom_bar.html)

## 5. まとめ

## 5.1 今日の内容

---

- ggplot2の概要
- ggplot2の基本文法
- グラフィックスの保存
- さまざまなグラフィックス
  - ヒストグラム
  - 散布図
  - 棒グラフ

## 5.2 次回までの課題

---

- RStudio Cloudプロジェクト内の  
05\_ggplot2\_01\_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: 2023-10-30 23:59