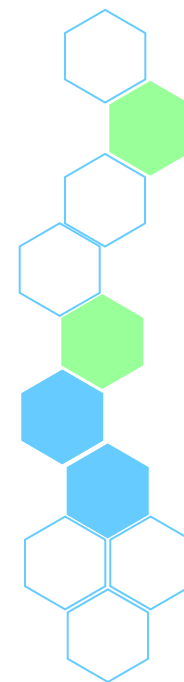
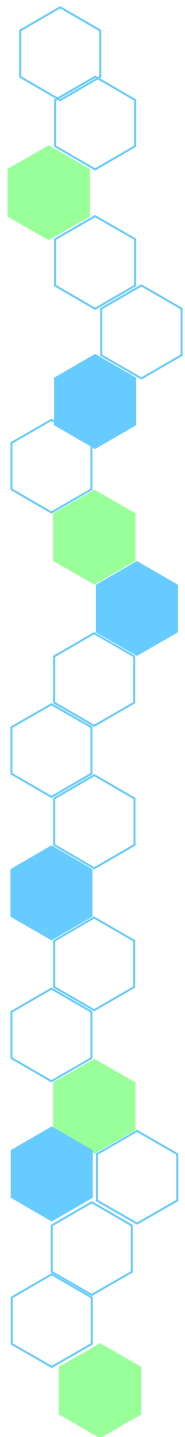




R で グラフ 作成

ggplot2 入門【ver. 3.3.0 版】

舟尾 暢男

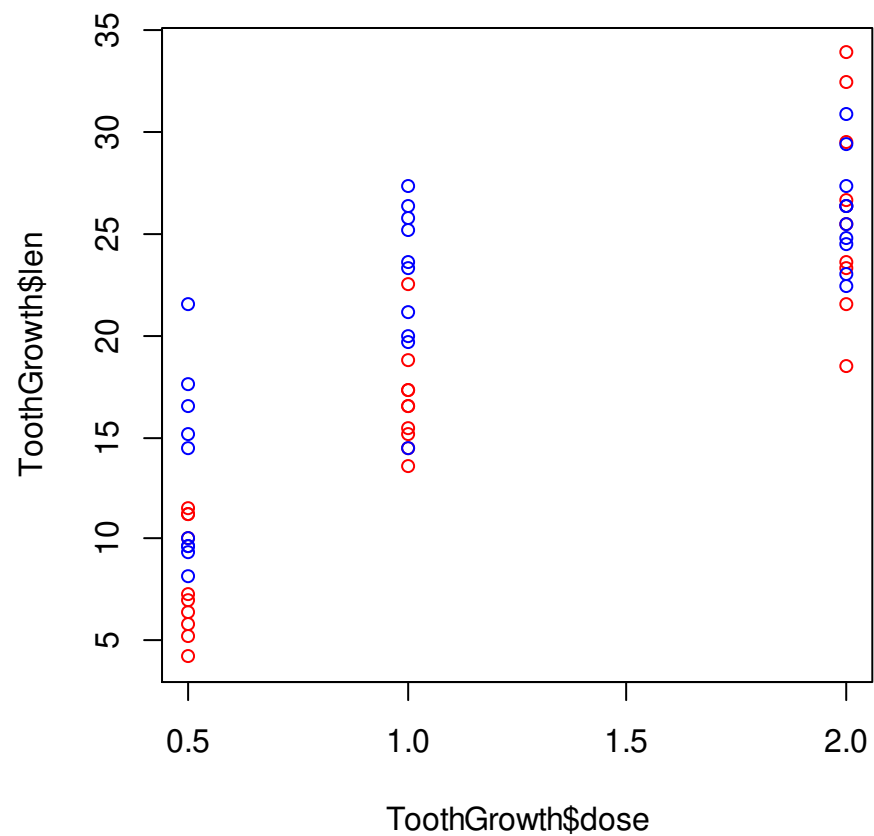


メニュー

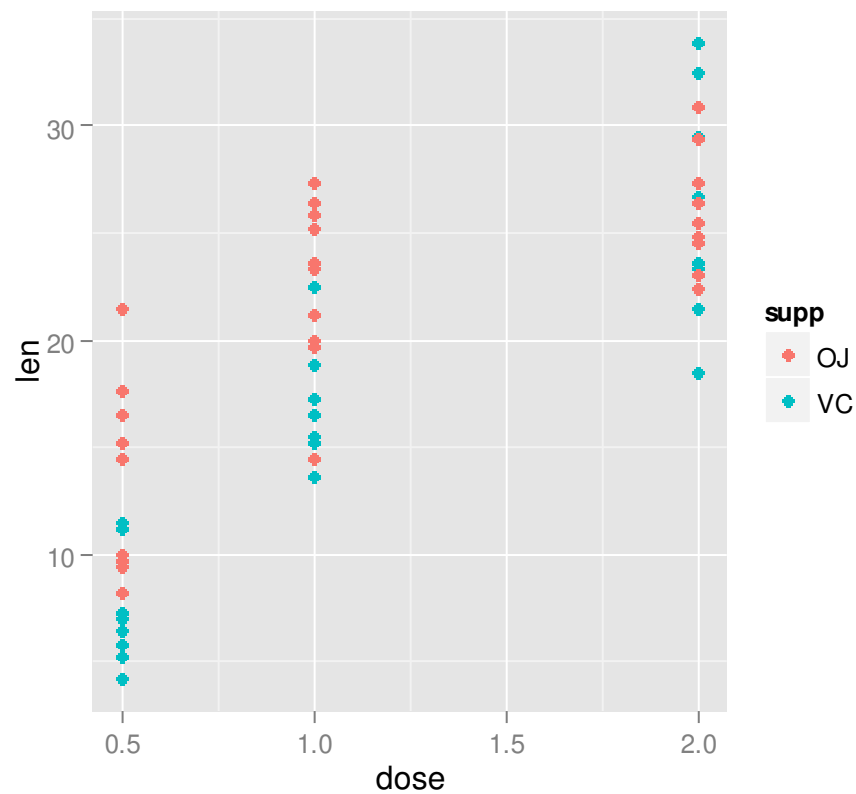
- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

Traditional vs. *ggplot2*

Traditional

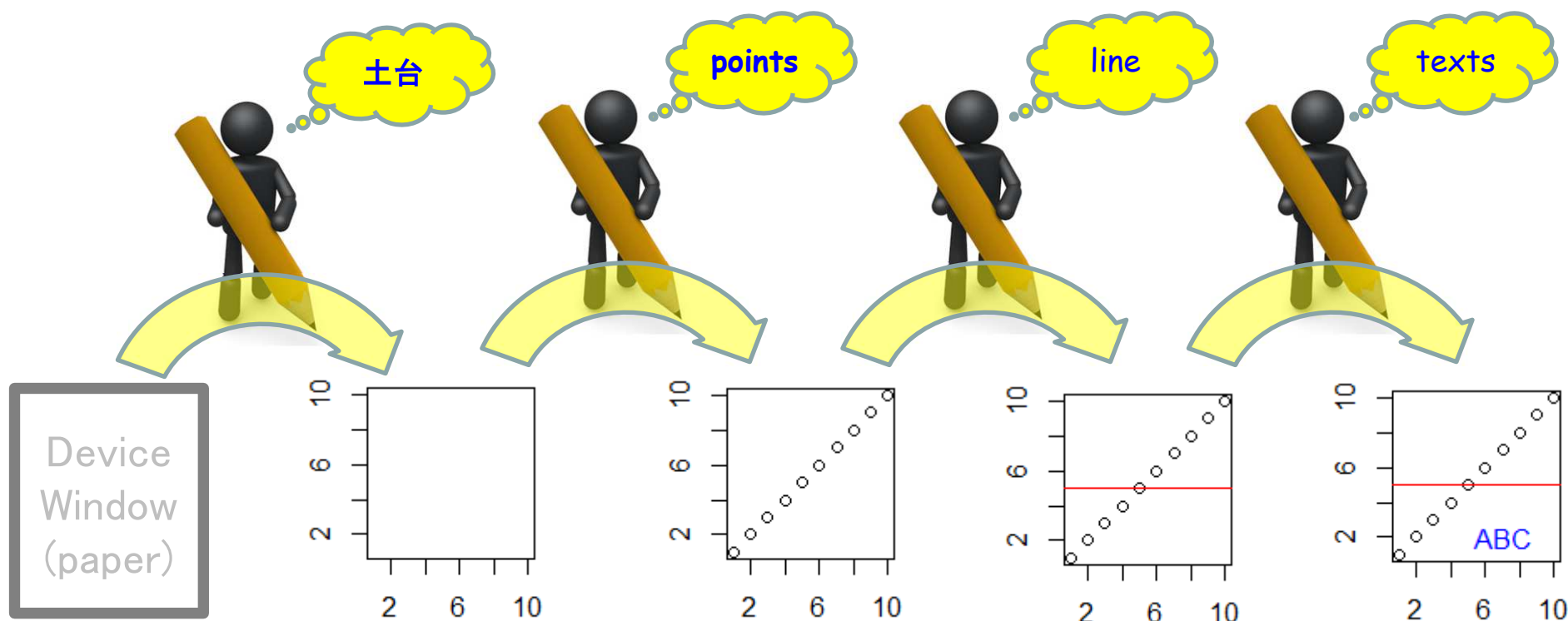


ggplot2



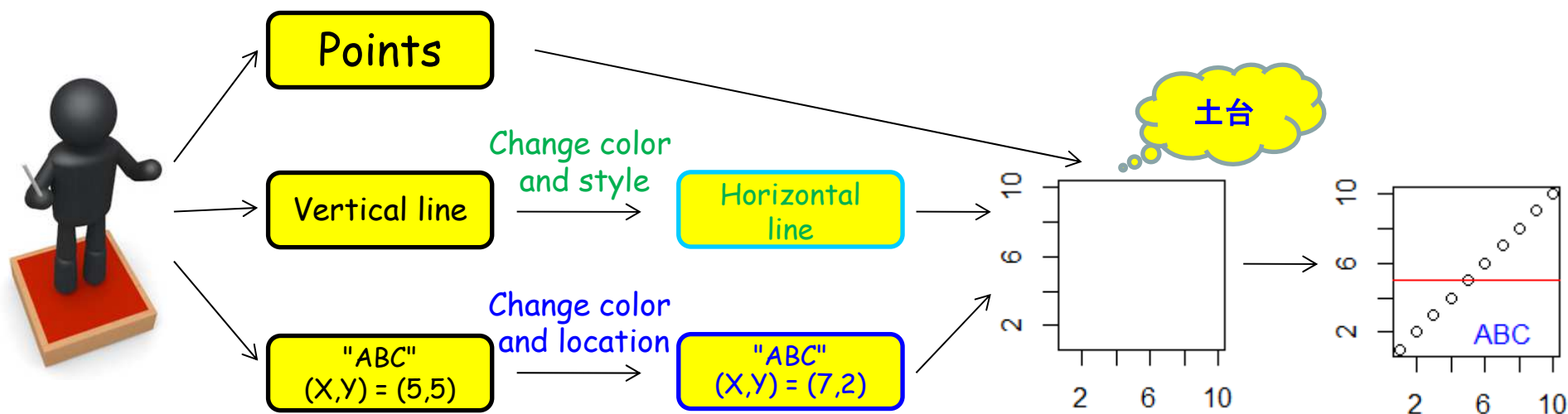
Traditional なグラフ

- 「ペンと紙を使って描く」スタイル
- 土台となるグラフを作った後、点や線や文字等を追記するスタイル
- 一度描いたグラフを、別のグラフを描くために再利用することは不可



ggplot2 で作成するグラフ

- Wilkinson (2005) "The Grammar of Graphics, Statistics and Computing" での統計グラフィックスの文法を具現化したパッケージ
- 「グラフに関するオブジェクト」を使って描くスタイル
- ggplot() で土台となるグラフを作った後, 点や線や文字に関するオブジェクトを geom_XXX() 等で作成し, 必要に応じてカスタマイズした後, 土台に貼り付けるスタイル(オブジェクトは再利用が可能)
- コマンド(文法)が非常に体系的で洗練されている



本日使うパッケージの呼び出し

- ggplot2 パッケージ等、関連パッケージのインストールと呼び出し

```
> # install.packages("ggplot2", dep=T)
> # install.packages("dplyr",    dep=T)
> # install.packages("scales",  dep=T)
> library(ggplot2)
> library(dplyr)
> library(scales)
```

- 「応用例」の章で使用するパッケージのインストールと呼び出し

```
> # install.packages("ggthemes", dep=T)
> # install.packages("survminer", dep=T)
> library(survminer)
> library(ggthemes)
> library(grid)
> library(gridExtra)
```

使用するデータ①: *ToothGrowth*

- モルモットにビタミン C 又はオレンジジュースを与えた時の歯の長さを調べる
 - **len**: 長さ(mm)
 - **supp**: サプリの種類(VC(ビタミンC) 又は OJ(オレンジジュース))
 - **dose**: 用量(0.5mg, 1.0mg, 2.0mg)

```
> head(ToothGrowth, n=3)
```

	len	supp	dose
1	4.2	VC	0.5
2	11.5	VC	0.5
3	7.3	VC	0.5

```
> tail(ToothGrowth, n=3)
```

	len	supp	dose
58	27.3	OJ	2
59	29.4	OJ	2
60	23.0	OJ	2

ggplot2 事始

```
> base <- ggplot(ToothGrowth,  
                 aes(x=dose, y=len, color=supp))
```

- **関数 ggplot()**: プロットオブジェクト(土台)を作成する
 - ggplot(データフレーム名, aes(x 座標の変数, y 座標の変数, エステ属性))
- **関数 aes()**: x 座標の変数, y 座標の変数, エステ属性※を指定する
(全て指定する必要は無い)
- エステ属性: 色、大きさ、線の種類、プロット点の形等
 - ここではサプリの種類(supp)と色を紐付けしており、種類ごとに色を変えたり、サプリの種類を凡例に盛り込む際の手掛かりとなる
- 上記はただの土台(変数 base)を作成しただけなので、これだけではグラフを作成したことにならない

※ aesthetic attribute: 審美的属性、気持ち悪い言葉ですが随所出てきますので慣れましょう

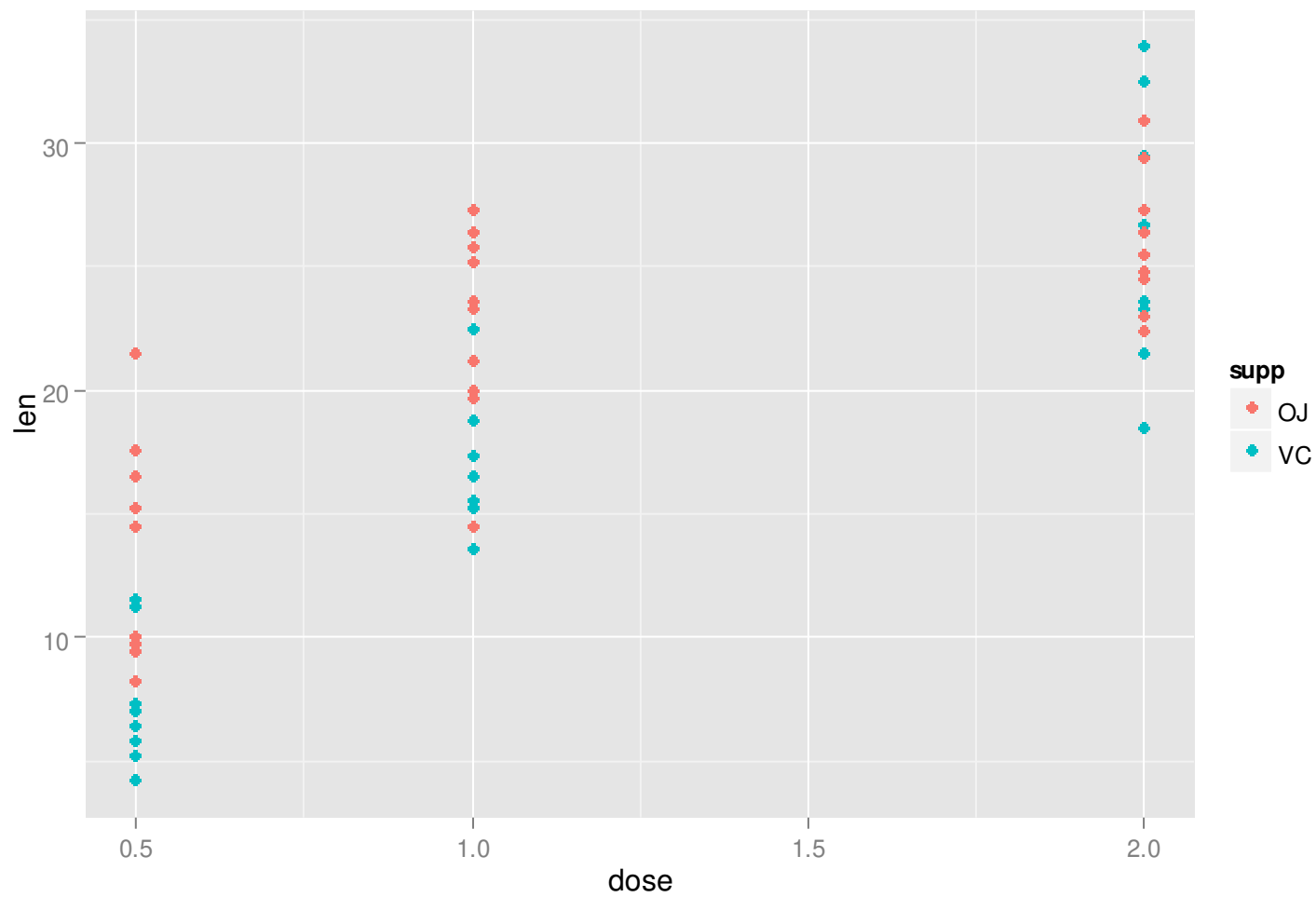
ggplot2 事始

```
> points <- base + geom_point()
> plot(points)
> base + geom_point()      # plot(points) と同じ働き
```

- 先程作成した土台(変数 base)にレイヤー※を追加した変数を作成する
- レイヤーとは「データに関連する要素」のことで、例えば上記の関数 `geom_point()` では「点レイヤー」を追加、すなわち「グラフの種類は散布図ですよ」という属性を変数 base に与えていることになる
 - 関数 `plot()` に変数 `points` を指定することでグラフが表示されるが、この関数はあまり使用されておらず、3 行目の書式「`base + geom_point()`」が一般的
- グラフについて、どういうカスタマイズが可能か(どういう引数があるか)は、11～15 頁の表にて確認(例えば、関数 `geom_point()` は `x`, `y`, `alpha`, `color`, `fill`, `group`, `shape`, `size`, `stroke` 等の引数がある)
- グラフを保存する場合はメニューから、又は関数 `ggsave()` を使用する
 - `ggsave("xxx.pdf", device="pdf", scale=1, dpi=300)` # 直前のグラフを保存
 - `ggsave("yyy.pdf", plot=グラフオブジェクト, width=9, height=9, units=c("in", "cm", "mm"))`

※ あまり聞き慣れない言葉かもしれませんが慣れましょう(レイヤーの種類は後述)

散布図が完成



関数 *geom_XXX* (グラフ) の種類①

関数	種類	エステ属性
<i>geom_abline</i>	直線	slope, intercept, alpha, color, linetype, size
<i>geom_area</i>	曲線下面積	x, ymax, ymin, alpha, color, fill, group, linetype, size
<i>geom_bar</i>	棒グラフ	x, y, alpha, color, fill, group, linetype, size
<i>geom_col</i>	棒グラフ	x, y, alpha, color, fill, group, linetype, size
<i>geom_bin2d</i>	ヒートマップ	bins, binwidth
<i>geom_blank</i>	ブランク(表示なし)	
<i>geom_boxplot</i>	箱ひげ図	lower, middle, upper, x, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
<i>geom_contour</i>	等高線プロット	x, y, alpha, color, group, linetype, size, weight
<i>geom_count</i>	同じ値の数を表示	x, y, alpha, color, fill, group, shape, size, stroke
<i>geom_crossbar</i>	箱ひげ図の箱だけ	x, ymax, ymin
<i>geom_curve</i>	曲線を描く	x, xend, y, yend, alpha, color, group, linetype, size
<i>geom_density</i>	密度曲線	x, y, alpha, color, fill, group, linetype, size, weight
<i>geom_density2d</i>	2次元密度推定	x, y, alpha, color, group, linetype, size

※ 青字: 必ず指定しなければならない引数

関数 *geom_XXX* (グラフ) の種類②

関数	種類	エステ属性
<i>geom_dotplot</i>	ドットプロット	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>stroke</i>
<i>geom_errorbar</i>	誤差に関するエラーバー (縦)	<i>x</i> , <i>y</i> <i>max</i> , <i>y</i> <i>min</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>height</i> , <i>linetype</i> , <i>size</i>
<i>geom_errorbarh</i>	誤差に関するエラーバー (横)	<i>x</i> <i>max</i> , <i>x</i> <i>min</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>height</i> , <i>linetype</i> , <i>size</i>
<i>geom_freqpoly</i>	頻度ポリゴン	<i>binwidth</i> , <i>alpha</i> , <i>color</i> , <i>linetype</i> , <i>size</i>
<i>geom_hex</i>	六角形のヒートマップ	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_histogram</i>	ヒストグラム	<i>bins</i> , <i>binwidth</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_hline</i>	水平線を描く	<i>yintercept</i> , <i>alpha</i> , <i>color</i> , <i>linetype</i> , <i>size</i>
<i>geom_jitter</i>	データをズラす(点等の 重なりを緩和するため)	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>shape</i> , <i>size</i> , <i>stroke</i>
<i>geom_label</i>	枠付きで文字列を描く	<i>x</i> , <i>y</i> , <i>label</i> , <i>alpha</i> , <i>angle</i> , <i>color</i> , <i>family</i> , <i>fontface</i> , <i>group</i> , <i>hjust</i> , <i>lineheight</i> , <i>size</i> , <i>vjust</i>
<i>geom_line</i>	線を描く	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>

※ 青字: 必ず指定しなければならない引数

関数 *geom_XXX* (グラフ) の種類③

関数	種類	エステ属性
<i>geom_linerange</i>	箱ひげ図の箱を線で表した プロット	<i>x</i> , <i>y</i> max , <i>y</i> min , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_map</i>	地図にヒートマップを追記する	<i>map_id</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i> , <i>subgroup</i>
<i>geom_path</i>	データの上から順に線で繋ぐ	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_point</i>	散布図	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>shape</i> , <i>size</i> , <i>stroke</i>
<i>geom_pointrange</i>	平均値±標準偏差のプロット	<i>x</i> , <i>y</i> , <i>y</i> max , <i>y</i> min , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>linetype</i> , <i>shape</i> , <i>size</i>
<i>geom_polygon</i>	ポリゴンプロット	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i> , <i>subgroup</i>
<i>geom_qq</i>	QQ プロット	<i>sample</i> , <i>group</i> , <i>x</i> , <i>y</i>
<i>geom_qq_line</i>	QQ line	<i>sample</i> , <i>group</i> , <i>x</i> , <i>y</i>
<i>geom_quantile</i>	箱ひげ図の連続変数版 (<i>stat_quantile()</i> を参照)	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i> , <i>weight</i>
<i>geom_raster</i>	<i>geom_tile</i> のハイパフォーマンス版	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>height</i> , <i>linetype</i> , <i>size</i> , <i>width</i>

※ 青字: 必ず指定しなければならない引数

関数 *geom_XXX* (グラフ) の種類④

関数	種類	エステ属性
<i>geom_rect</i>	矩形を描く	<i>xmax</i> , <i>xmin</i> , <i>ymax</i> , <i>ymin</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>height</i> , <i>linetype</i> , <i>size</i> , <i>width</i>
<i>geom_ribbon</i>	折れ線グラフにバンド幅を加えたプロット	<i>x</i> , <i>ymax</i> , <i>ymin</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_rug</i>	ラグプロット(軸にデータ線を追記)	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_segment</i>	線分を描く	<i>x</i> , <i>xend</i> , <i>y</i> , <i>yend</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_sf</i>	GISデータ(sf object)を描く	
<i>geom_smooth</i>	平滑線	<i>x</i> , <i>y</i> , <i>ymax</i> , <i>ymin</i> , <i>alpha</i> , <i>color</i> , <i>fill</i> , <i>group</i> , <i>linetype</i> , <i>size</i> , <i>weight</i>
<i>geom_spoke</i>	角度を表す線分を描く	<i>x</i> , <i>y</i> , <i>angle</i> , <i>radius</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_step</i>	階段関数	<i>x</i> , <i>y</i> , <i>alpha</i> , <i>color</i> , <i>group</i> , <i>linetype</i> , <i>size</i>
<i>geom_text</i>	文字列を描く	<i>x</i> , <i>y</i> , <i>label</i> , <i>alpha</i> , <i>angle</i> , <i>color</i> , <i>family</i> , <i>fontface</i> , <i>group</i> , <i>hjust</i> , <i>lineheight</i> , <i>size</i> , <i>vjust</i>

※ 青字: 必ず指定しなければならない引数

関数 *geom_XXX* (グラフ) の種類⑤

関数	種類	エステ属性
<i>geom_tile</i>	タイル・プロット	<i>x</i> , <i>y</i> , alpha, color, fill, group, height, linetype, size, width
<i>geom_violin</i>	バイオリン・プロット	<i>x</i> , <i>y</i> , alpha, color, fill, group, linetype, size, weight
<i>geom_vline</i>	縦線を描く	xintercept, alpha, color, linetype, size

※ 青字: 必ず指定しなければならない引数

※ 詳細は ggplot2 パッケージのマニュアルを参照

<https://ggplot2.tidyverse.org/reference/>

関数 *stat_XXX* (統計量) の種類①

関数	種類	エステ属性
<i>stat_bin</i>	データの bin の幅 (ヒストグラムの棒の横幅)	
<i>stat_bin2d</i>	矩形 (rectangle) の中のデータ数	<i>x</i> , <i>y</i> , fill, group
<i>stat_binhex</i>	六角形のヒートマップを描くためのデータ	
<i>stat_boxplot</i>	箱ひげ図で出てくる要約統計量	
<i>stat_contour</i>	等高線	<i>x</i> , <i>y</i> , <i>z</i> , group, order
<i>stat_count</i>	頻度集計	<i>x</i> , group, weight, <i>y</i>
<i>stat_density</i>	1 次元の密度推定	
<i>stat_density2d</i>	2 次元の密度推定	<i>x</i> , <i>y</i> , color, size
<i>stat_ecdf</i>	経験累積分布関数	<i>x</i> , <i>y</i> ,
<i>stat_ellipse</i>	確率楕円	<i>x</i> , <i>y</i> ,
<i>stat_function</i>	ユーザーが指定した関数 (で計算する)	<i>y</i> , group
<i>stat_identity</i>	データの変換をしない (データのまま)	なし
<i>stat_qq</i>	QQ プロット	<i>sample</i> , group, <i>x</i> , <i>y</i>
<i>stat_qq_line</i>	QQ line	<i>sample</i> , group, <i>x</i> , <i>y</i>

※ 青字: 必ず指定しなければならない引数

関数 *stat_XXX* (統計量) の種類②

関数	種類	エステ属性
<i>stat_quantile</i>	分位点	
<i>stat_sf</i>	GISデータ (sf object)	
<i>stat_smooth</i>	平滑化曲線	
<i>stat_spoke</i>	極座標変換 (x と y の範囲を使用)	
<i>stat_summary</i>	データの要約統計量	x, y, group
<i>stat_summary_bin</i>	データの頻度集計	x, y, group
<i>stat_summary_2d</i>	ヒートマップの各矩形のデータ数	x, y, z
<i>stat_summary_hex</i>	ヒートマップの各六角形のデータ数	x, y, z
<i>stat_unique</i>	データの重複を除去	group
<i>stat_ydensity</i>	密度推定値 (バイオリンプロット用)	x, y

※ 青字: 必ず指定しなければならない引数

※ 詳細は ggplot2 パッケージのマニュアルを参照

<https://ggplot2.tidyverse.org/reference/>

メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

例1: 平均値±標準偏差の図を描く①

1. パッケージ dplyr の関数を用いて、以下の命令を実行することによりデータフレーム ToothGrowth の変数 dose ごとの変数 len の平均値と標準偏差を求め、結果をデータフレーム MS に格納する
 - 関数 `group_by()`: データフレームを第 2 引数の変数でグループ化
 - 関数 `summarise()`: グループ化されたデータフレームに対して処理

```
> MS <- summarise(group_by(ToothGrowth, dose),  
+                  m=mean(len), s=sd(len))  
> MS  
# A tibble: 3 × 3  
  dose      m      s  
  <dbl> <dbl> <dbl>  
1  0.5 10.605 4.499763  
2  1.0 19.735 4.415436  
3  2.0 26.100 3.774150
```

例1: 平均値±標準偏差の図を描く①

2. グラフの土台となる変数 base を作成する

```
> base1 <- ggplot(MS, aes(x=dose, y=m))  
> base1
```

3. 平均値に関する折れ線グラフを作成する

```
> base1 + geom_line()
```

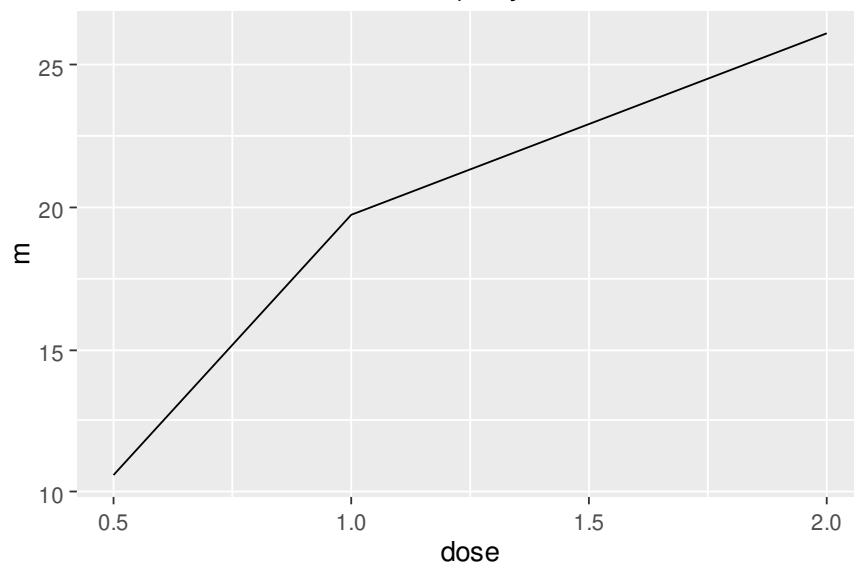
4. 3. の折れ線グラフにエラーバーを追加し、平均値±標準偏差の図を描くことを考える → スライド 11～15 頁より、エラーバーを描く関数を探す → スライド 12 頁の関数 `geom_errorbar()`
5. 4. の関数について、必ず指定しなければならない引数を [ggplot2 パッケージのマニュアル](https://ggplot2.tidyverse.org/reference/)※ にて確認する

例1: 平均値±標準偏差の図を描く①

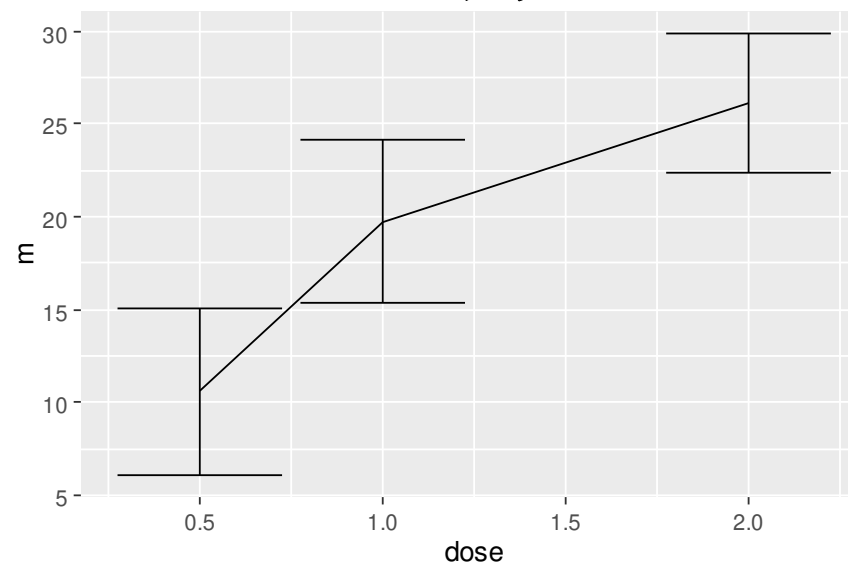
6. 4. ~ 5. を踏まえ、平均値±標準偏差の図を描く
7. 6. のグラフを関数 `ggsave()` にて外部ファイルに保存

```
> out <- base1 + geom_line() +
+   geom_errorbar(aes(ymin=m-s, ymax=m+s))
> out
> ggsave("c:/temp/out.pdf", plot=out, width=9, height=9,
+   units="in")
```

#3 のグラフ



#6 のグラフ



例2: 平均値±標準偏差の図を描く②

1. データフレーム ToothGrowth の変数 dose & supp ごとの変数 len の平均値と標準偏差を求め、データフレーム MS2 に格納する

```
> MS2 <- summarise(group_by(ToothGrowth, dose, supp),  
+                   m=mean(len), s=sd(len))  
> MS2
```

	dose	supp	m	s
	<dbl>	<fctr>	<dbl>	<dbl>
1	0.5	OJ	13.23	4.459709
2	0.5	VC	7.98	2.746634
3	1.0	OJ	22.70	3.910953
4	1.0	VC	16.77	2.515309
5	2.0	OJ	26.06	2.655058
6	2.0	VC	26.14	4.797731

2. 変数 dose & supp ごとの変数 len の平均値の折れ線グラフを描く
(横軸に指定した変数 dose は連続変数として指定)

```
> base21 <- ggplot(MS2, aes(x=dose, y=m, color=supp))  
> base21 + geom_line()
```

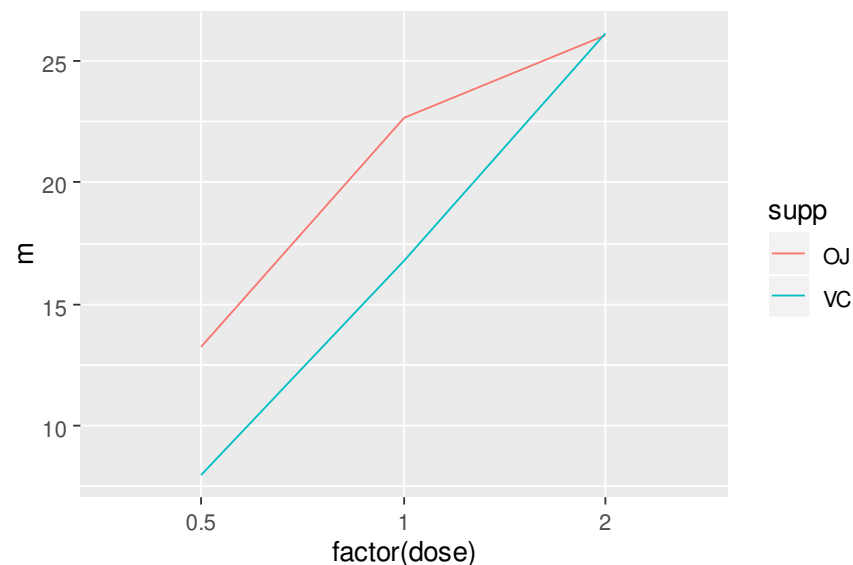
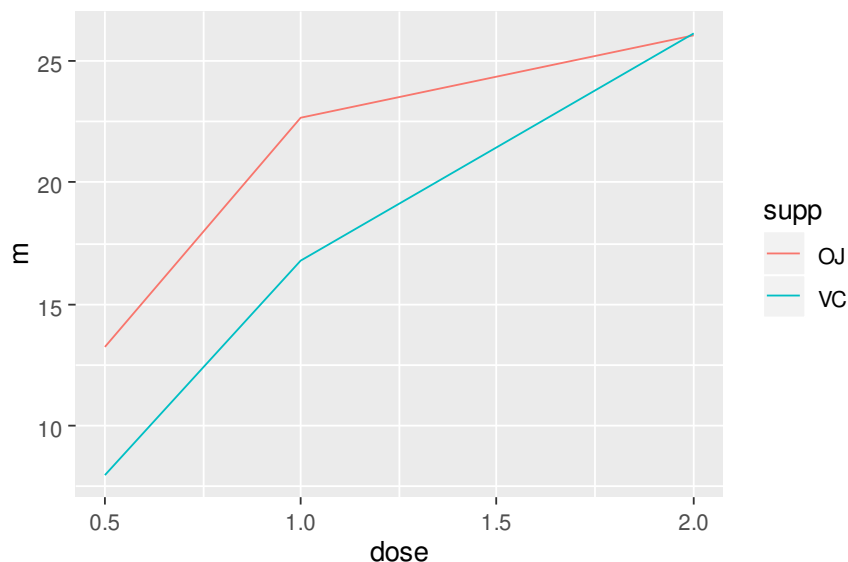
例2: 平均値±標準偏差の図を描く②

3. 横軸に指定した変数 dose をカテゴリ変数とするとエラーが出る
→ ggplot2 側が、dose と supp のどちらをグループ化するかが理解不可

```
> base22 <- ggplot(MS2, aes(x=factor(dose), y=m, color=supp))
> base22 + geom_line()
geom_path: Each group consists of only one observation.
Do you need to adjust the group aesthetic?
```

4. 引数 group にグループ化する変数を指定 → カテゴリが複数ある場合は要注意！

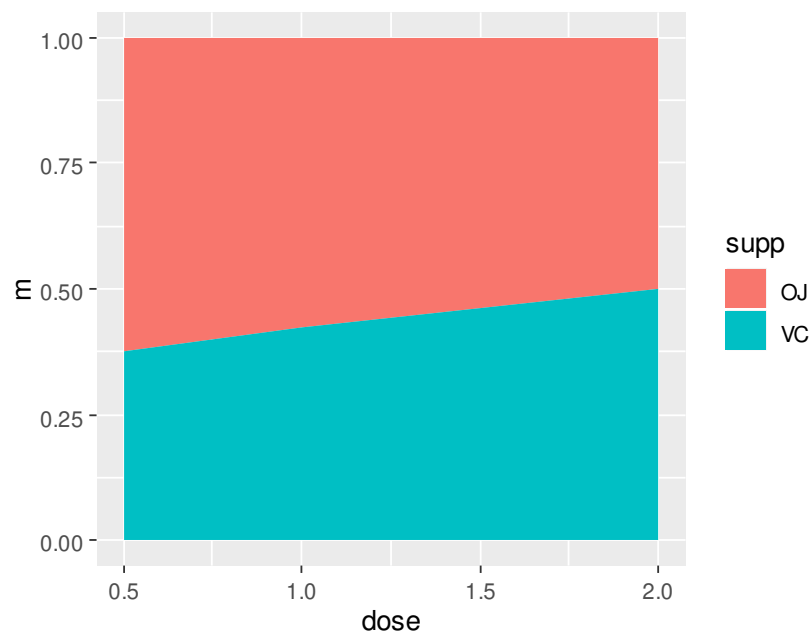
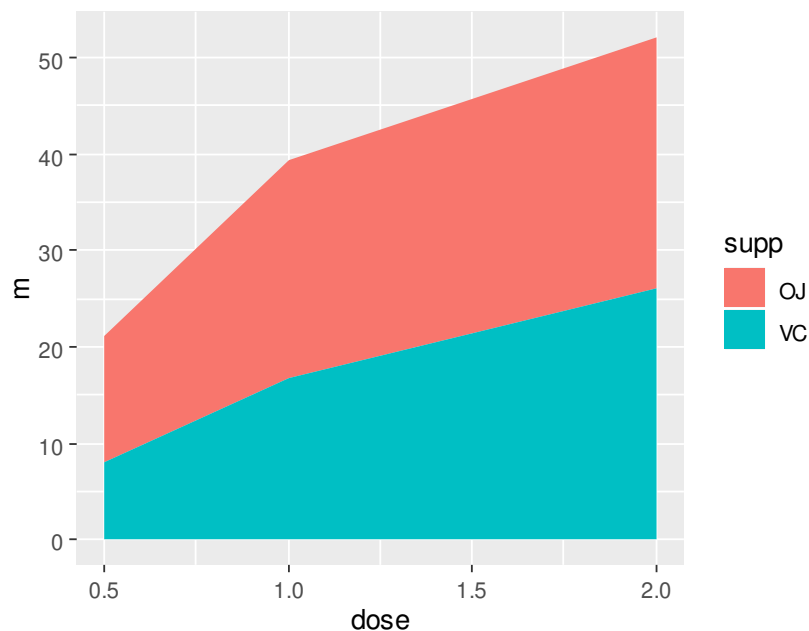
```
> base23 <- ggplot(MS2, aes(x=factor(dose), y=m, color=supp,
+ group=supp))
> base23 + geom_line()
```



例2: 平均値±標準偏差の図を描く③

1. 前頁のグラフを積み上げグラフにしてみる
2. さらに、グラフを「割合に関するグラフ」にしてみる
3. 直前に描いたグラフを関数 `ggsave()` にて外部ファイルに保存

```
> base24 <- ggplot(MS2, aes(x=dose, y=m, fill=supp))
> base24 + geom_area()
> base25 <- ggplot(MS2, aes(x=dose, y=m, fill=supp))
> base25 + geom_area(position="fill")
> ggsave("c:/temp/out.wmf", device="wmf", scale=1, dpi=300)
```



例3: 散布図を描く

1. データフレーム ToothGrowth の変数 dose ごとの変数 len の散布図を描く、その際、変数 supp ごとに色分け(群分け)する

```
> head(ToothGrowth, n=3)
  len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
```

2. スライド 11~15 頁より、散布図を描く関数を探す
→ スライド 13 頁の関数 `geom_point()`
3. 2. の関数について、必ず指定しなければならない引数を
[ggplot2 パッケージのマニュアル](#)※ にて確認する

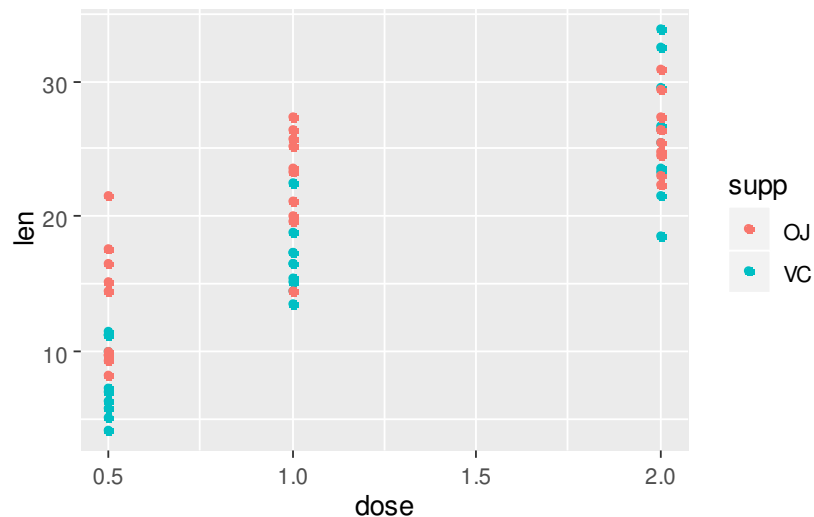
例3: 散布図を描く

4. 下記の様に、土台(例えば変数 base)を作らずにグラフを作成することも可
5. 関数 `ggplot()` の `aes` 内で「`color=supp`」とすると、散布図が `supp` ごとに群分けされる
 - 関数 `geom_point()` で「`color="red"`」を指定すると、点が全て赤になる
 - 群分けしたい場合は、土台の関数 `ggplot()` の `aes` 内で指定する(関数 `geom_point()` では不可)
 - 詳細は「グラフのカスタマイズ」の章にて
6. 引数 `shape`、`color`、`size` 等を適切に使い分ける
7. 慣れてくれば、いろいろ装飾が出来る → 詳細は「グラフのカスタマイズ」の章にて
 - 使用できる数式表現について: <http://cse.naro.affrc.go.jp/takezawa/r-tips/r/56.html>

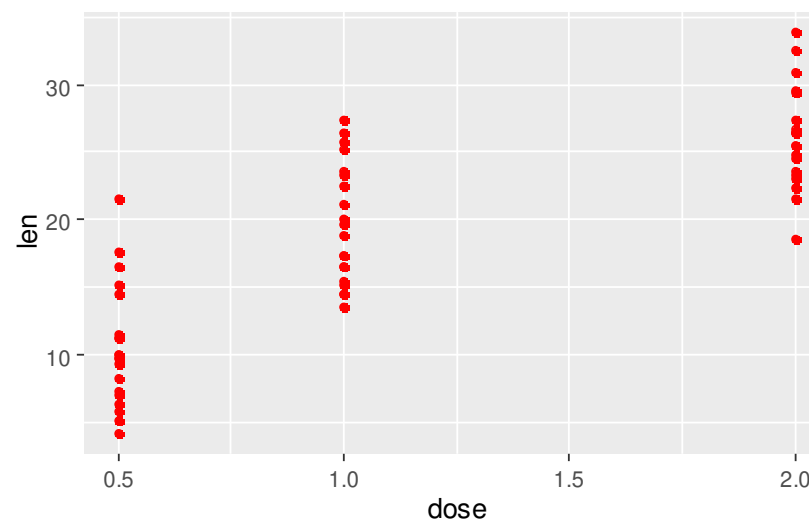
```
> ggplot(ToothGrowth, aes(x=dose, y=len, color=supp)) +  
+   geom_point()  
  
> ggplot(ToothGrowth, aes(x=dose, y=len)) +  
+   geom_point(color="red")  
  
> ggplot(ToothGrowth, aes(x=dose, y=len, shape=supp)) +  
+   geom_point(color="blue", size=2, position=position_dodge(0.1))  
  
> ggplot(ToothGrowth, aes(x=dose, y=len, color=supp)) +  
+   geom_point() + geom_hline(yintercept=15, color="blue") +  
+   annotate("text", x=1.8, y=7, label="r^2 == 0.80", parse=T, size=5)
```

例3: 散布図を描く

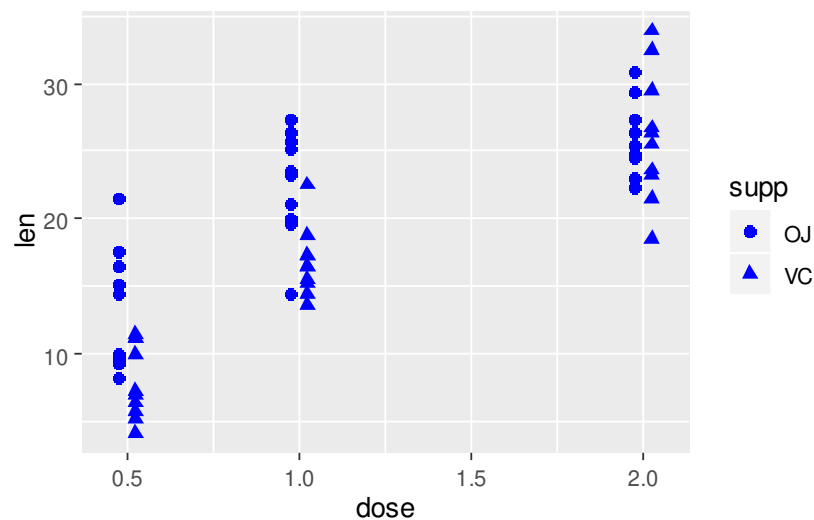
前頁 1つ目のグラフ



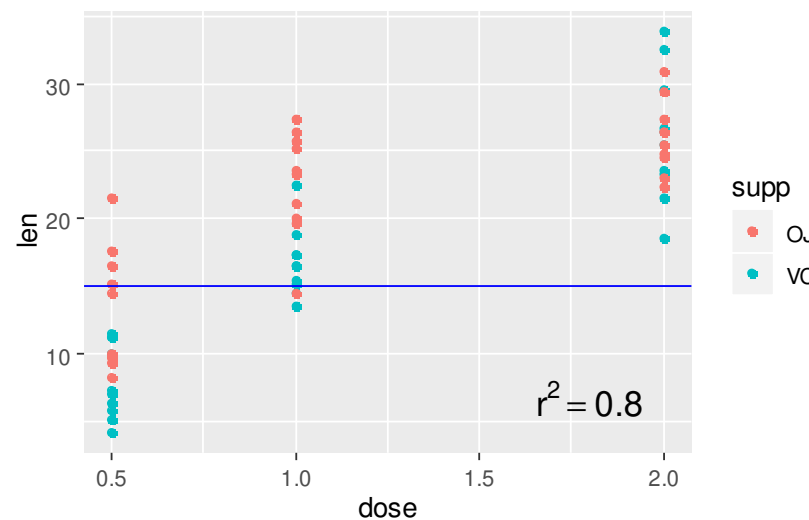
前頁 2つ目のグラフ



前頁 3つ目のグラフ



前頁 4つ目のグラフ



例4: 種々の棒グラフを描く

1. 「変数 len の値が 15 以上であるデータ数」に関する棒グラフを描く → 2 種類のデータを作成する

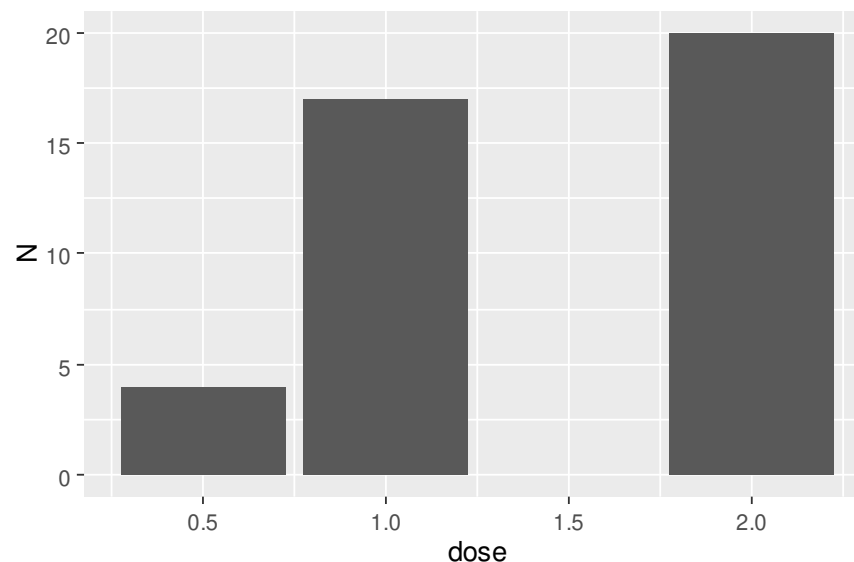
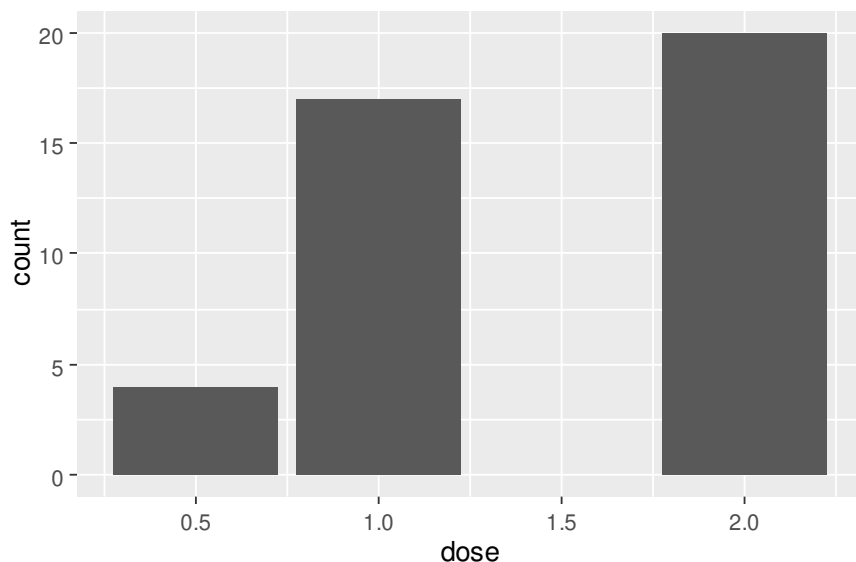
```
> TMP <- filter(ToothGrowth, len>15)
> head(TMP, n=3)      # 元データのレコードを残す
  len supp dose
1 16.5   VC    1
2 16.5   VC    1
3 15.2   VC    1

> TAB <- summarise(group_by(TMP, dose), N=n())
> head(TAB, n=3)      # dose のカテゴリごとに頻度集計を済ませておく
# A tibble: 3 x 2
  dose      N
  <dbl> <int>
1  0.5      4
2  1       17
3  2       20
```

例4: 種々の棒グラフを描く

2. スライド 11～15 頁より、棒グラフを描く関数を探した後、必ず指定しなければいけない引数を [ggplot2 パッケージのマニュアル](#)※ にて確認
→ スライド 11 頁の関数 `geom_bar()` と関数 `geom_col()`

```
> ggplot(TMP, aes(x=dose)) + # 引数 x に指定されたカテゴリのデータ数を数える
+   geom_bar()
> ggplot(TAB, aes(x=dose, y=N)) + # 変数 y に指定された値をそのまま表示
+   geom_col()
```



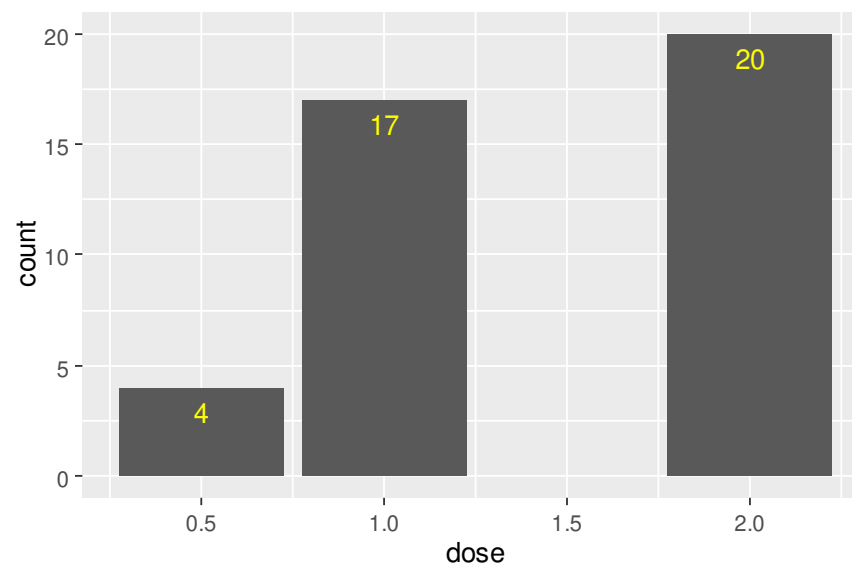
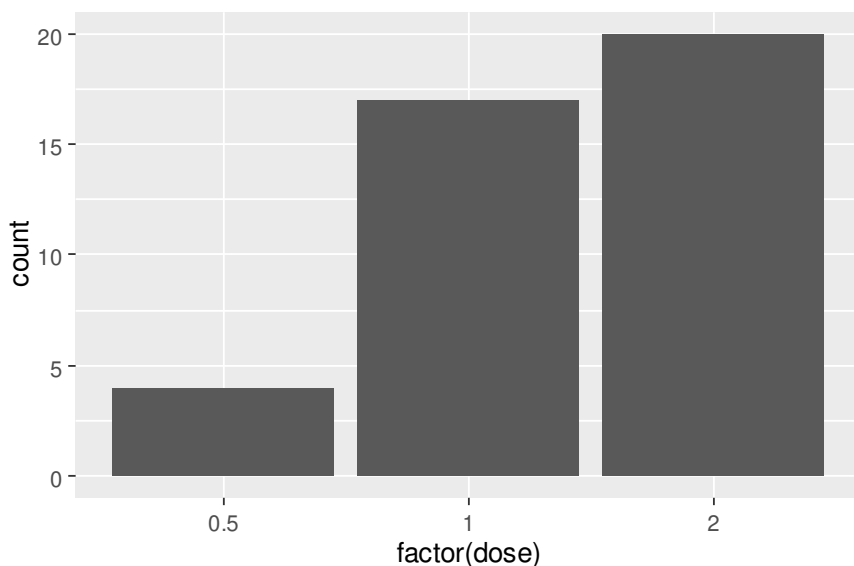
例4: 種々の棒グラフを描く

3. 横軸を因子型にしてみる(出力は前者のみ)

```
> ggplot(TMP, aes(x=factor(dose))) + geom_bar()
> ggplot(TAB, aes(x=factor(dose), y=N)) + geom_col()
```

4. 関数 `geom_text()` で、グラフに文字を追記する(出力は前者のみ)

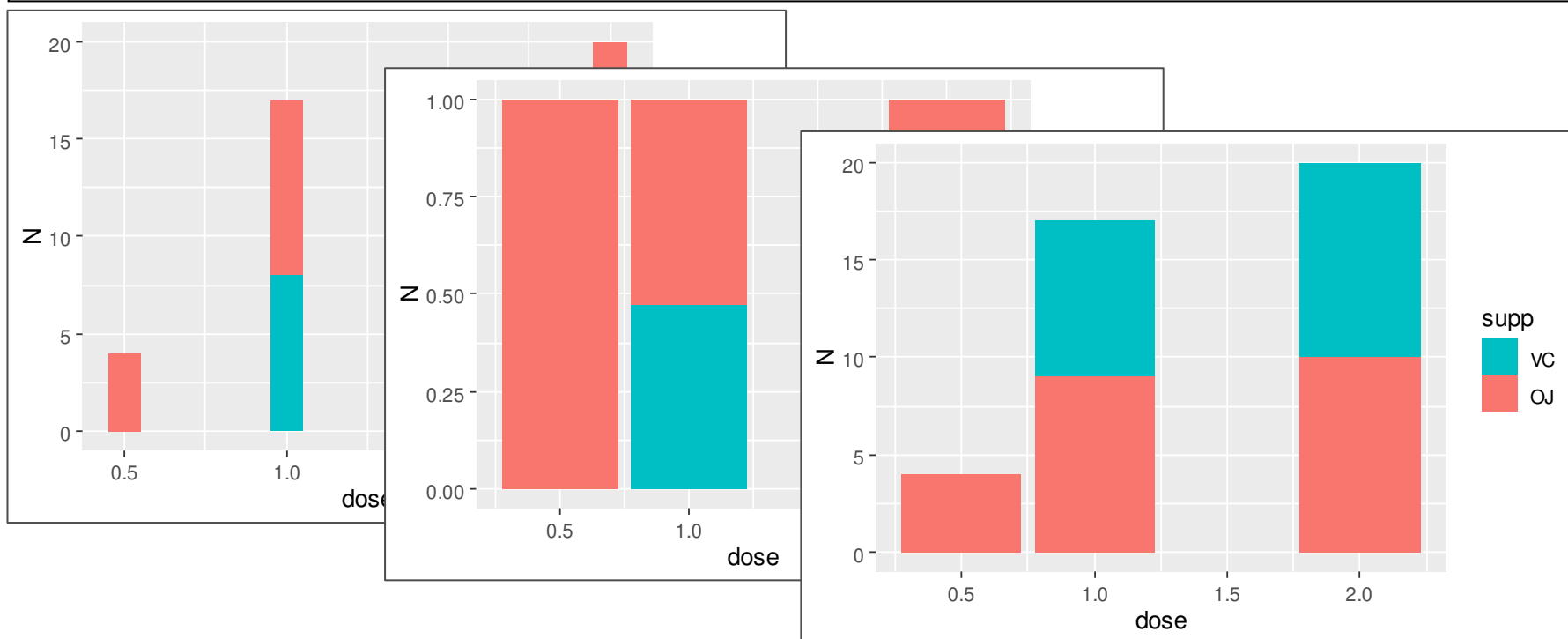
```
> ggplot(TMP, aes(x=dose)) + geom_bar() +
+   geom_text(aes(label=..count..), stat="count", vjust=1.3, color="yellow")
> ggplot(TAB, aes(x=dose, y=N)) + geom_col() +
+   geom_text(aes(label=N), vjust=-0.2)
```



例4: 種々の棒グラフを描く

5. 積み上げ棒グラフにつき「棒の幅を変更」「縦軸を割合に」「カテゴリを逆順」
→ fill=supp で棒の塗りつぶしの色、color=supp で棒の周りの色が変わる

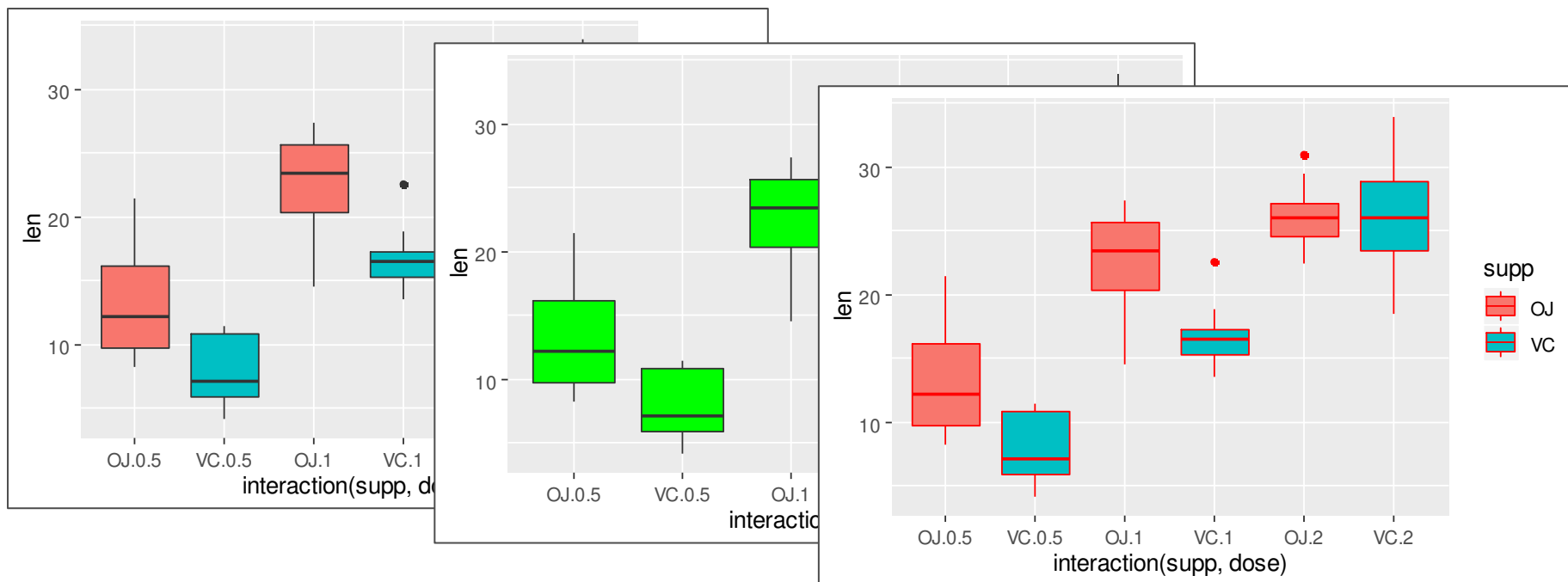
```
> TAB2 <- summarise(group_by(TMP, dose, supp), N=n())
> ggplot(TAB2, aes(x=dose, y=N, fill=supp)) + geom_col(width=0.1)      # 棒の幅
> ggplot(TAB2, aes(x=dose, y=N, fill=supp)) + geom_col(position="fill") # 割合に
> ggplot(TAB2, aes(x=dose, y=N, fill=supp)) +                          # カテゴリ
+   geom_col(position=position_stack(reverse=T)) +                     # を逆順に
+   guides(fill=guide_legend(reverse=T))
```



例5: 箱ひげ図を描く

1. 引数 `fill=supp` を指定するタイミングの違い(1つ目 vs. 2つ目)
2. 関数 `geom_boxplot()` に引数 `color` や `position` を指定してみる(3つ目)

```
> ggplot(ToothGrowth, aes(x=interaction(supp, dose), y=len, fill=supp)) +  
+   geom_boxplot()  
> ggplot(ToothGrowth, aes(x=interaction(supp, dose), y=len)) +  
+   geom_boxplot(fill="green")  
> ggplot(ToothGrowth, aes(x=interaction(supp, dose), y=len, fill=supp)) +  
+   geom_boxplot(color="red", position="dodge")
```



メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

使用するデータ②: iris

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
...

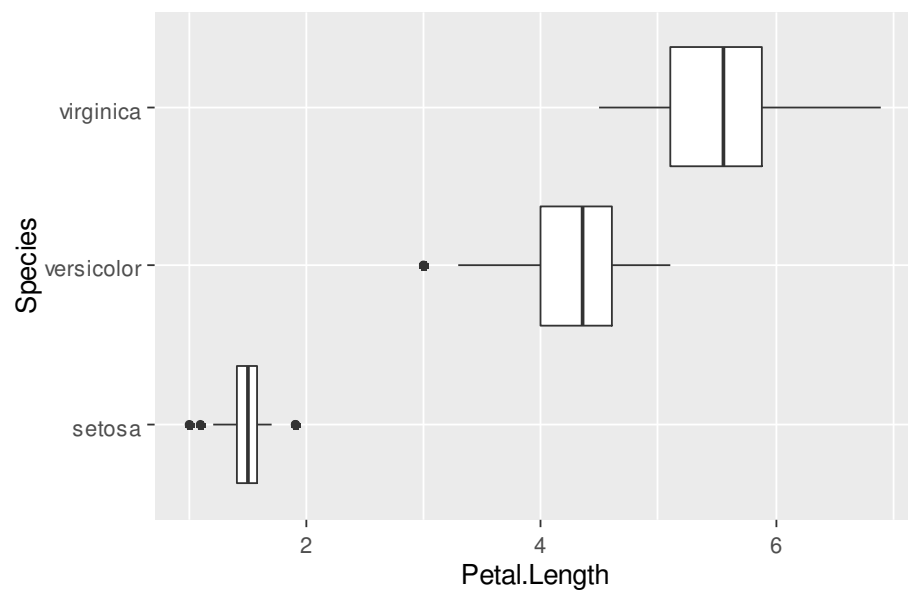
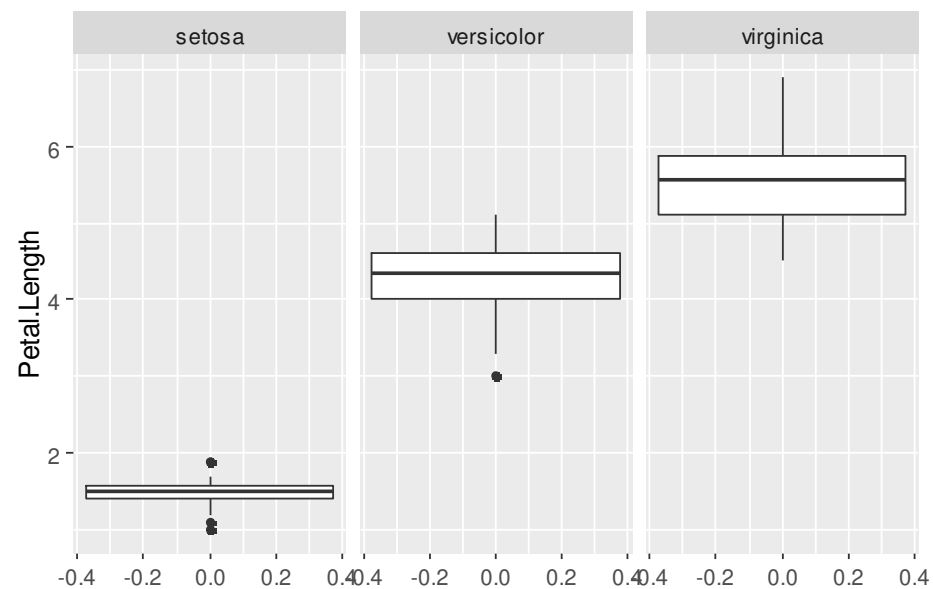
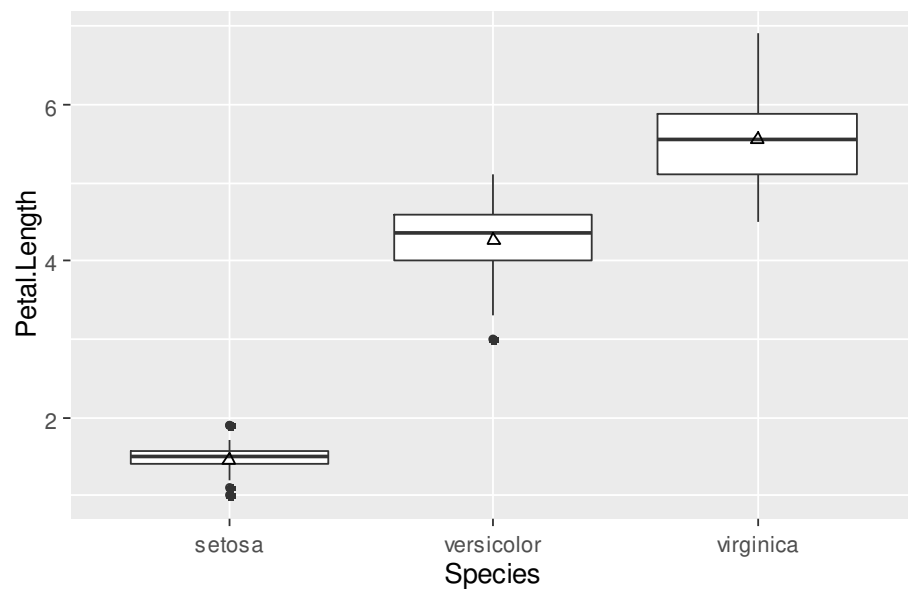
- フィッシャーが判別分析法を紹介するために利用したアヤメの品種分類 (Species: setosa、versicolor、virginica) に関するデータ
⇒ 以下の 4 変数を説明変数としてアヤメの種類を判別しようとした
 - アヤメのがくの長さ (Sepal.Length)
 - アヤメのがくの幅 (Sepal.Width)
 - アヤメの花弁の長さ (Petal.Length)
 - アヤメの花弁の幅 (Petal.Width)

例1: 箱ひげ図

1. 引数 `x` に変数 `Species`、引数 `y` に変数 `Petal.Length` を指定
 - 変数 `Species` の値ごとに変数 `Petal.Length` の箱ひげ図を作成
 - 関数 `stat_summary()` で平均値に関する点 Δ を追記する
2. 関数 `facet_wrap()` で層別グラフの作成
3. 関数 `coord_flip()` でグラフの転置

```
> ggplot(iris, aes(Species, Petal.Length)) +  
+   geom_boxplot() +  
+   stat_summary(fun.y="mean", geom="point", shape=2)  
  
> ggplot(iris, aes(Species, Petal.Length)) +  
+   geom_boxplot() + facet_wrap(~ Species, nrow=1, ncol=3)  
  
> ggplot(iris, aes(Species, Petal.Length)) +  
+   geom_boxplot() + coord_flip()
```

例1: 箱ひげ図

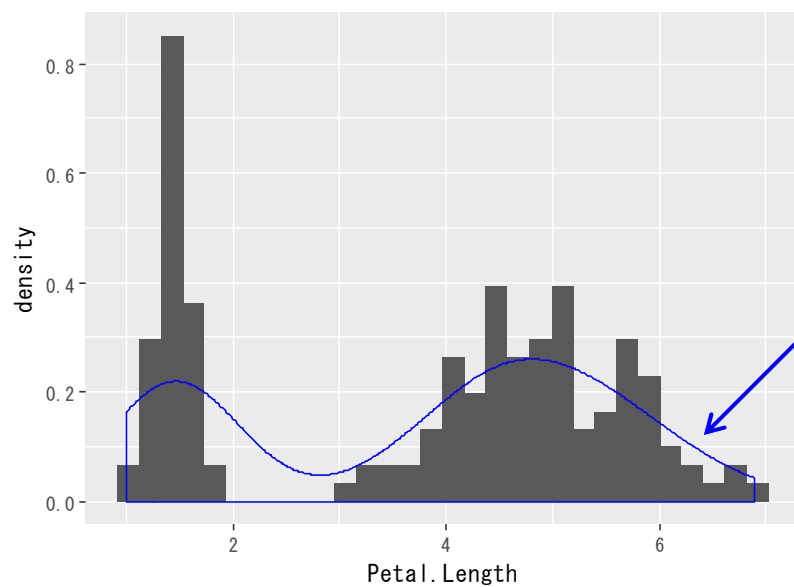
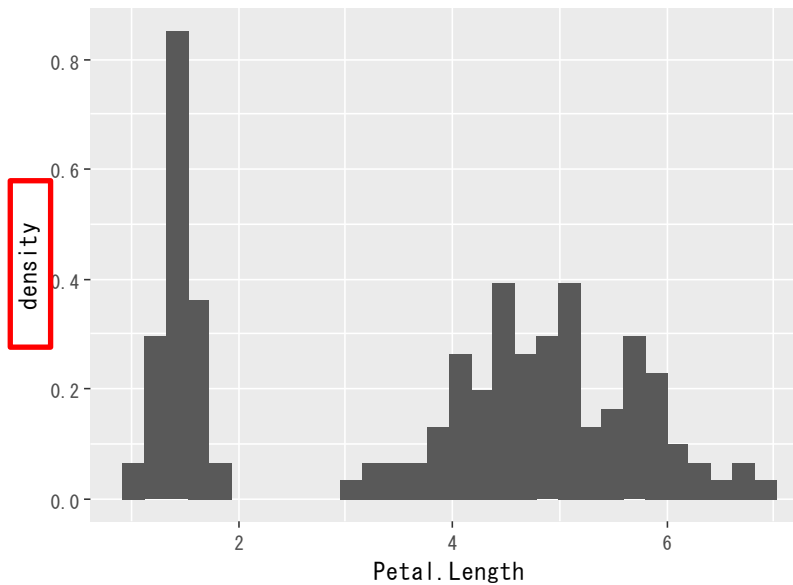
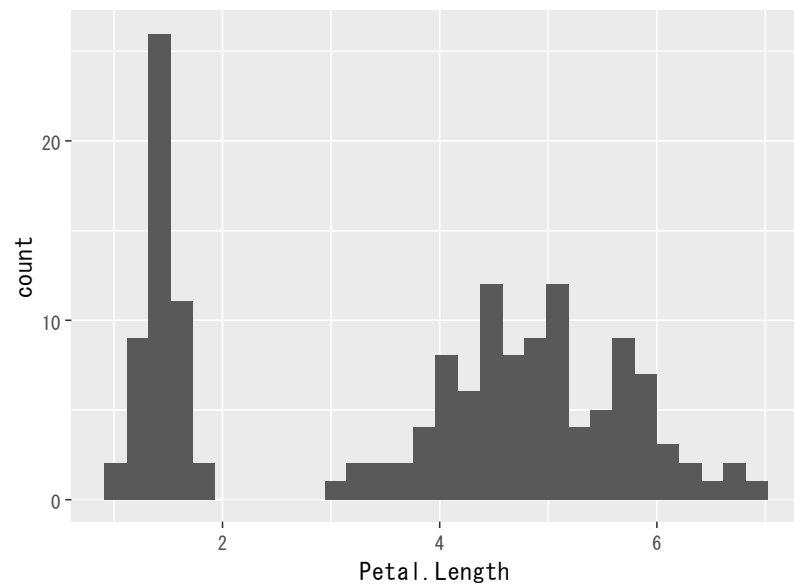


例2: ヒストグラムと密度曲線

1. 引数 `x` に変数 `Petal.Length` を指定してヒストグラムを描く
2. 縦軸を密度にしてみる
3. 変数 `Petal.Length` に関するヒストグラムや密度曲線を描く

```
> ggplot(iris, aes(Petal.Length)) +  
+   geom_histogram()  
  
> ggplot(iris, aes(Petal.Length)) +  
+   geom_histogram(aes(y = ..density..)) # 縦軸=密度  
  
> ggplot(iris, aes(Petal.Length)) +  
+   geom_histogram(aes(y = ..density..)) +  
+   geom_density(color="blue") # 密度曲線を追記
```

例2: ヒストグラムと密度曲線

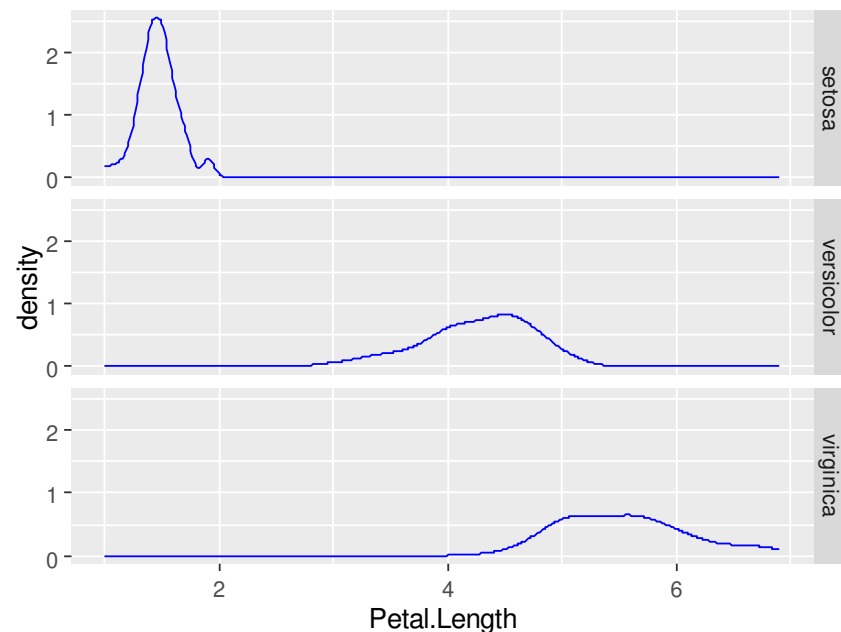
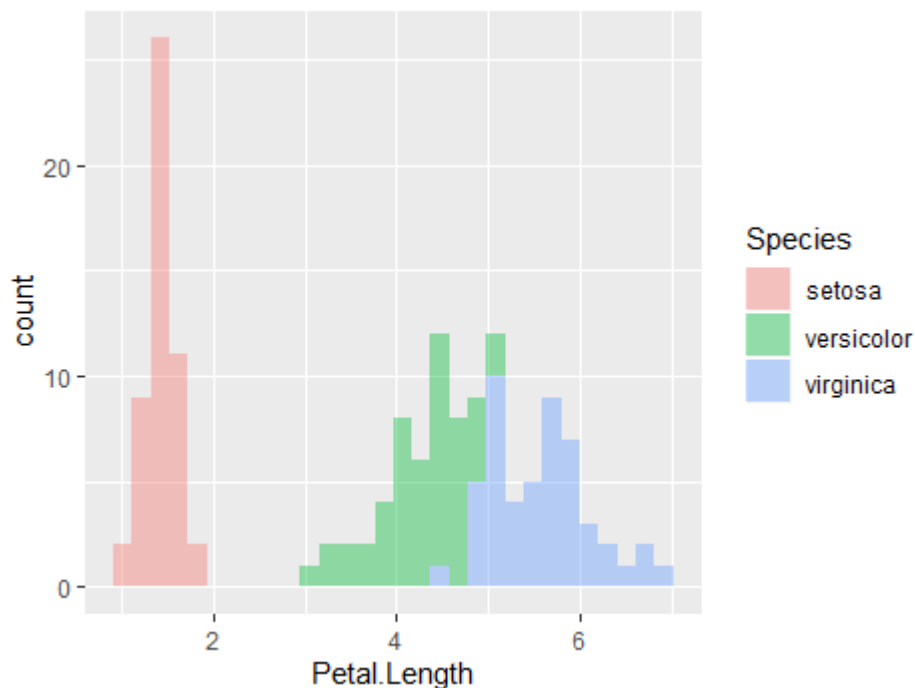


密度曲線

例2: ヒストグラムと密度曲線

1. 変数 Species の値ごとに変数 Petal.Length のヒストグラムを作成
2. 変数 Species の値ごとに変数 Petal.Length の密度曲線を描く

```
> ggplot(iris, aes(Petal.Length, fill=Species)) +  
+   geom_histogram(alpha=0.4)  
> ggplot(iris, aes(Petal.Length)) +  
+   geom_line(stat="density", color="blue") +  
+   facet_grid(Species ~ .)
```

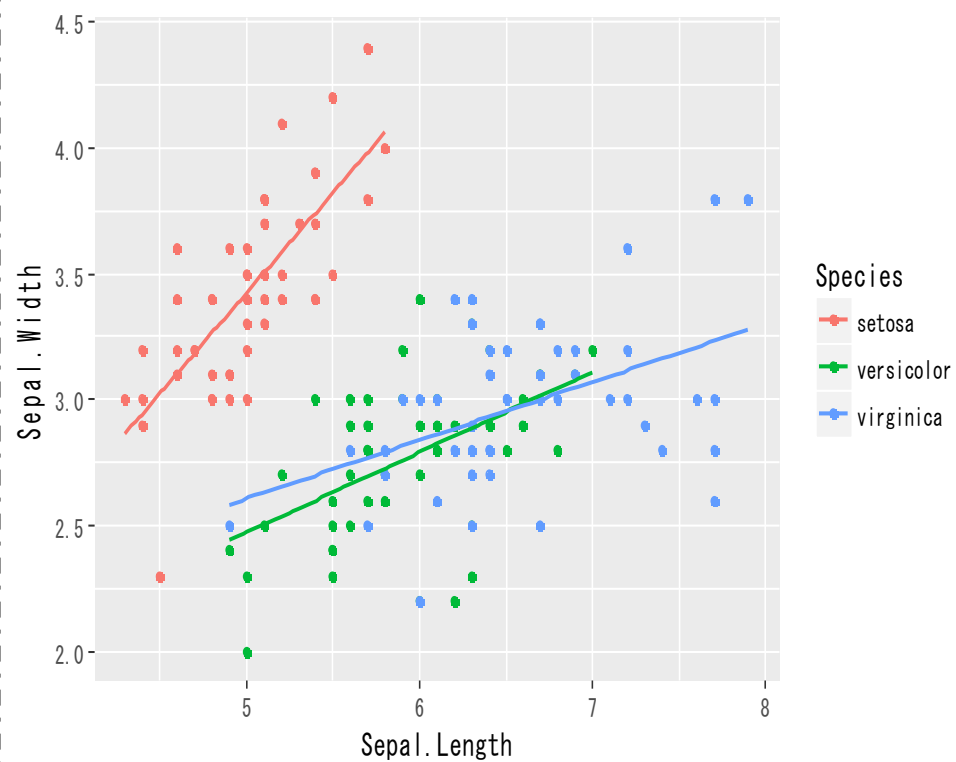
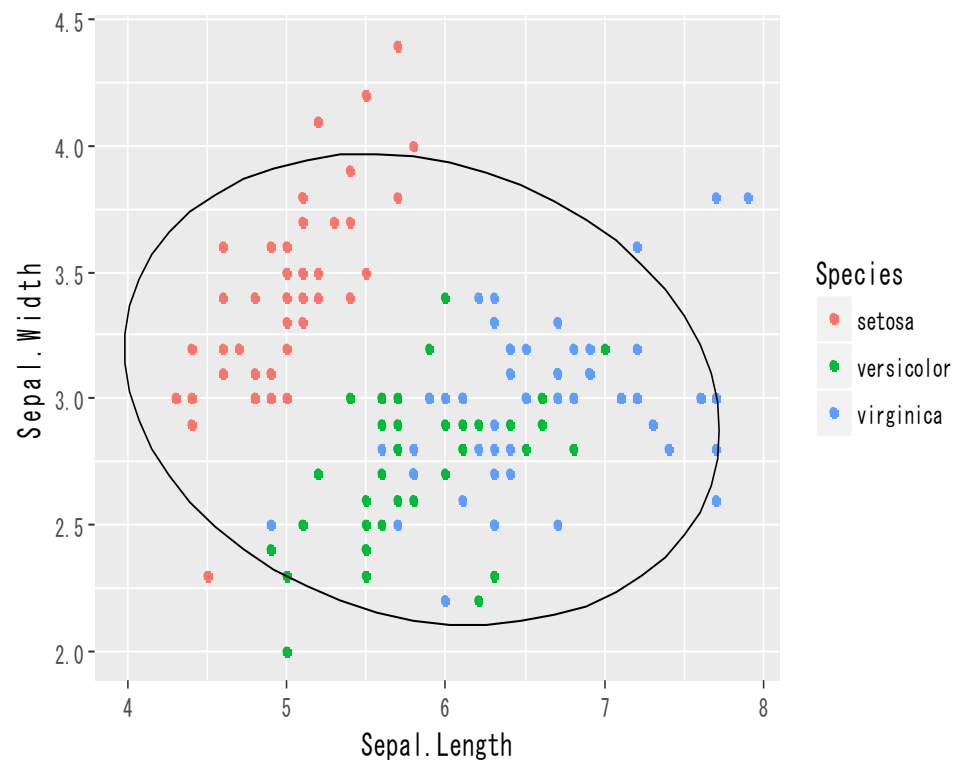


例3: 散布図と確率楕円・回帰直線

1. 引数 x に変数 Sepal.Length、引数 y に Sepal.Width を指定
2. 上記 2 変数に関する散布図を変数 Species ごとに作成、確率楕円や回帰直線を追記する

```
> G1 <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +  
+   geom_point(aes(color=Species)) +  
+   stat_ellipse()  
> G1  
  
> G2 <- ggplot(iris, aes(Sepal.Length, Sepal.Width,  
+   color=Species)) +  
+   geom_point() +  
+   stat_smooth(method=lm, se=FALSE) # デフォルト: formula=y~x  
> G2
```


例3: 散布図と確率楕円・回帰直線



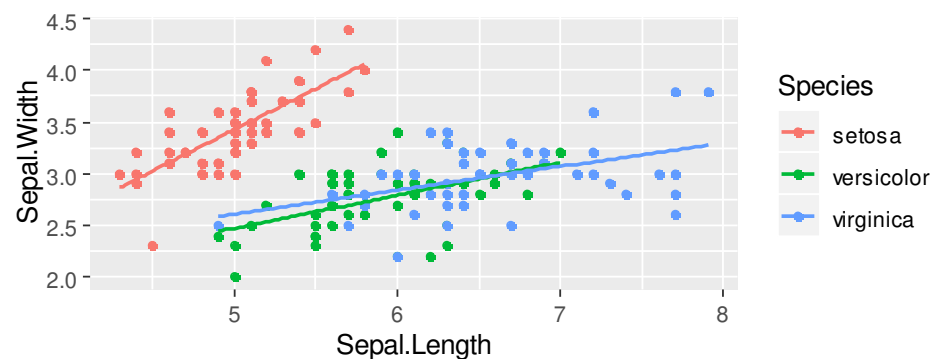
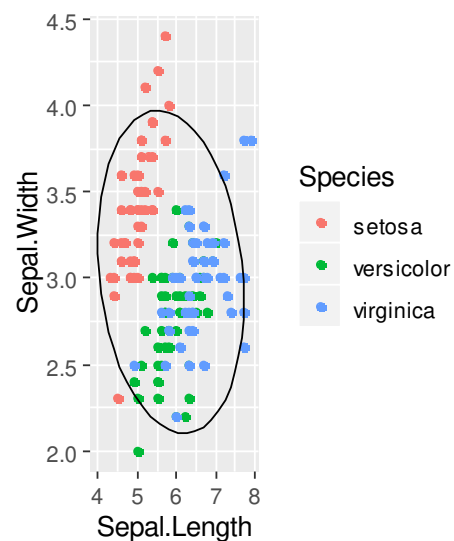
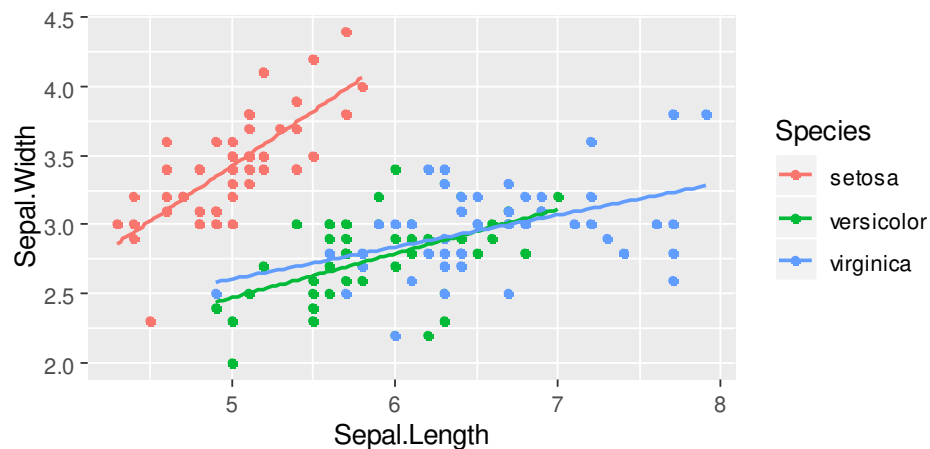
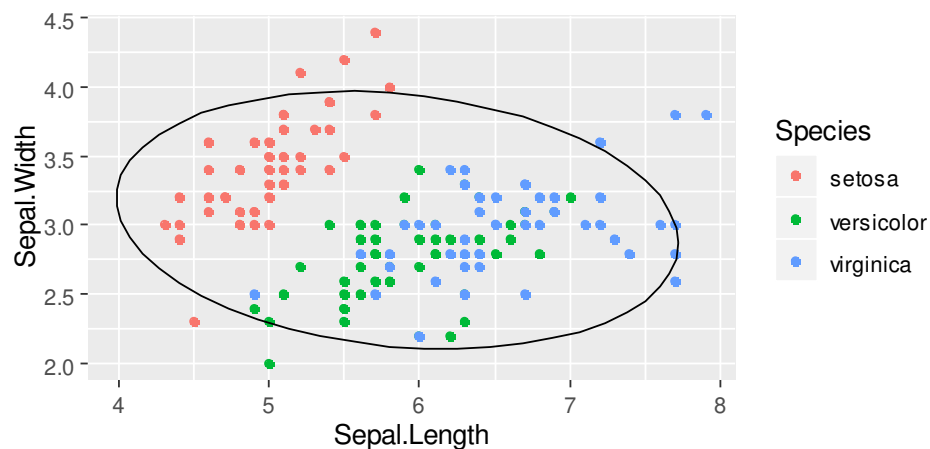
例3: 散布図と確率楕円・回帰直線

- gridExtra パッケージの 関数 `grid.arrange()` を使うことで複数のグラフを 1 枚にまとめることが出来る
 - 関数 `grid.arrange(プロットオブジェクト1, プロットオブジェクト2, ...)` でプロットオブジェクトを並べて表示
 - 引数 `layout_matrix` に行列を指定することで複雑な配置が可能

```
> library(grid)
> library(gridExtra)
> grid.arrange(G1, G2, nrow=2, ncol=1, as.table=TRUE,
+   heights=c(1,1)) # 列が複数ある場合は width=c(1,1) で幅調整が可

> LM <- matrix(c(1,NA,2,2), nrow=2, byrow=T)
> grid.arrange(G1, G2, layout_matrix=LM, heights=c(1.5,1))
```

例3: 散布図と確率楕円・回帰直線

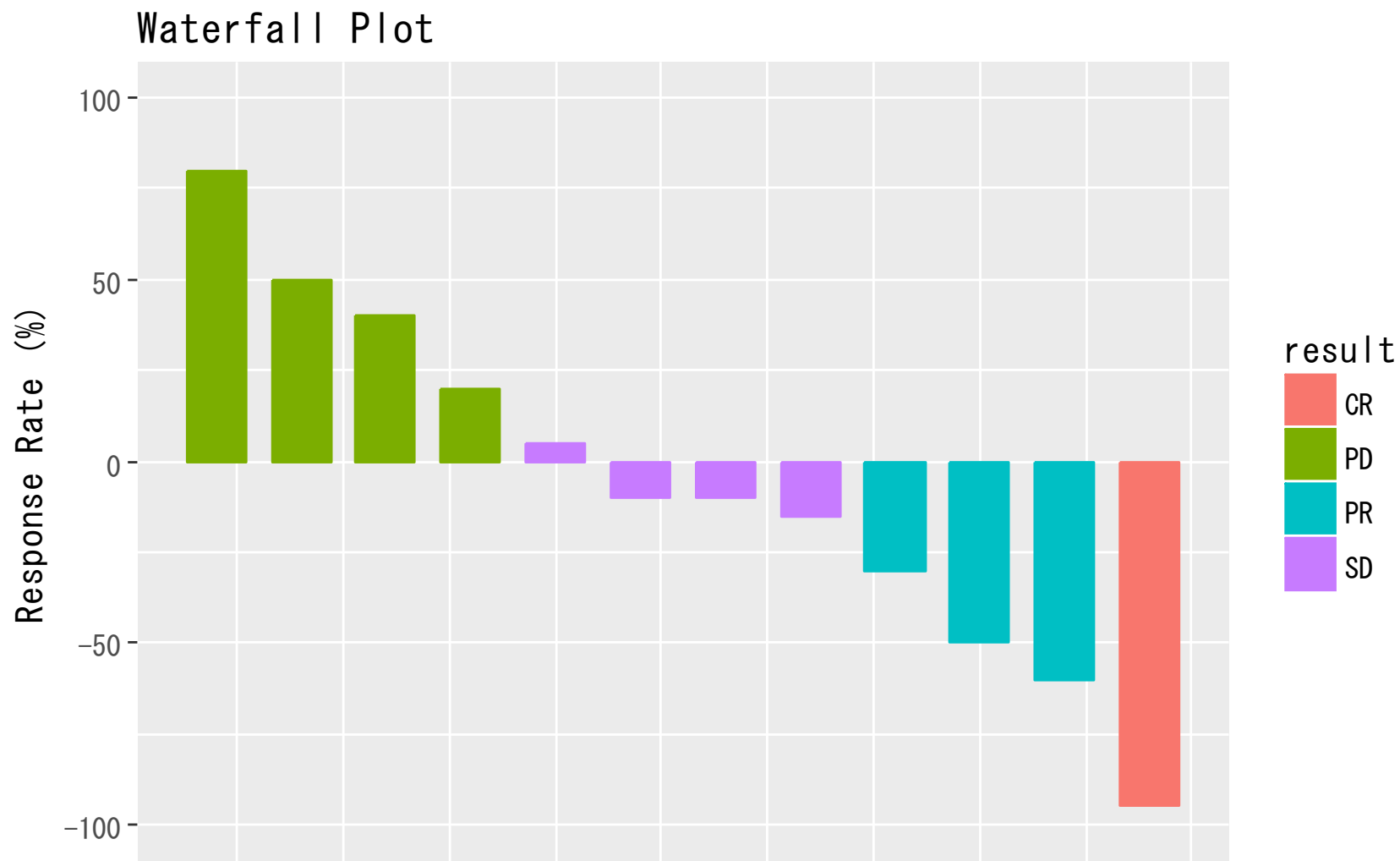


例4: Waterfall Plot

- グラフをいろいろ装飾できるが、詳しくは次節にて

```
> Oncology <- data.frame(  
+   id      =c( 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12),  
+   rrate =c( 80, 50, 40, 20,  5, -10, -10, -15, -30, -50, -60, -95),  
+   result=c("PD","PD","PD","PD","SD","SD","SD","SD","PR","PR","PR","CR"))  
  
> ggplot(Oncology, aes(x=id, y=rrate, fill=result, color=result)) +  
+   geom_bar(stat="identity", width=0.7, position=position_dodge(0.1)) +  
+   labs(list(title="Waterfall Plot", x=NULL, y="Response Rate (%)")) +  
+   theme(axis.line.x=element_blank(), axis.text.x=element_blank(),  
+         axis.ticks.x=element_blank()) +  
+   coord_cartesian(ylim=c(-100,100))
```

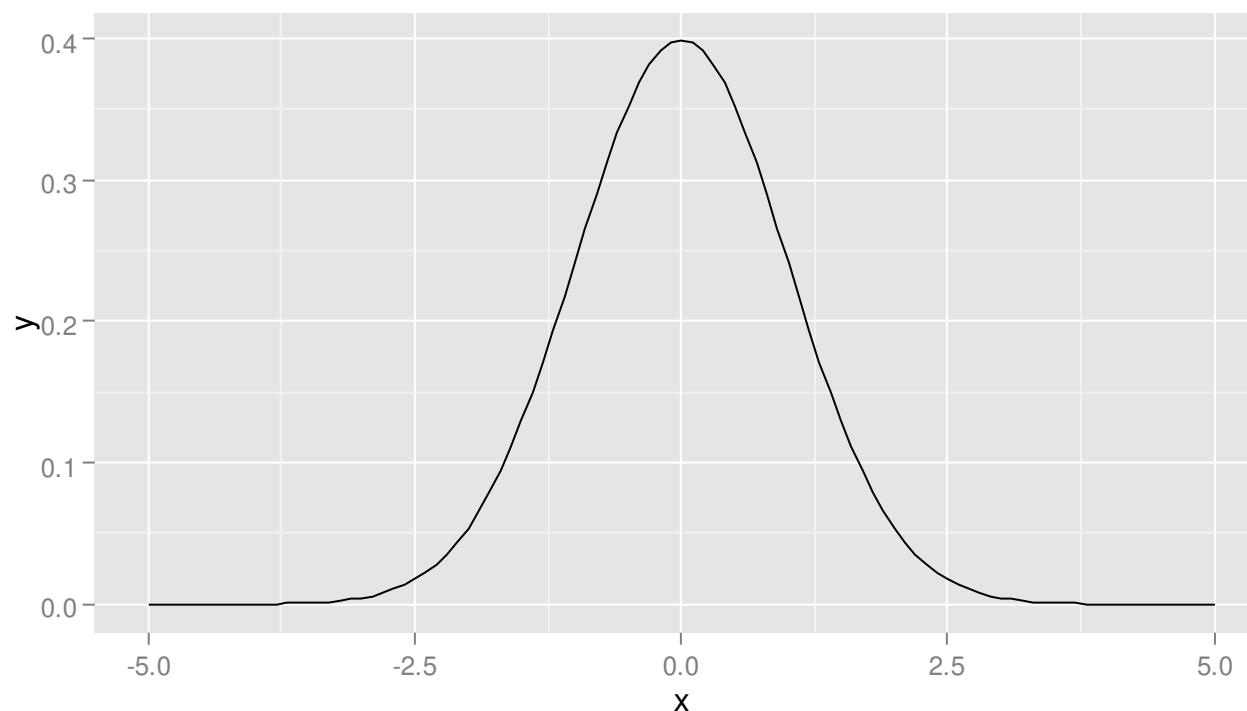
例4: Waterfall Plot



例5: 数学関数のプロット

```
> base <- ggplot(data.frame(x = c(-5, 5)), aes(x))  
> base + stat_function(fun=dnorm)
```

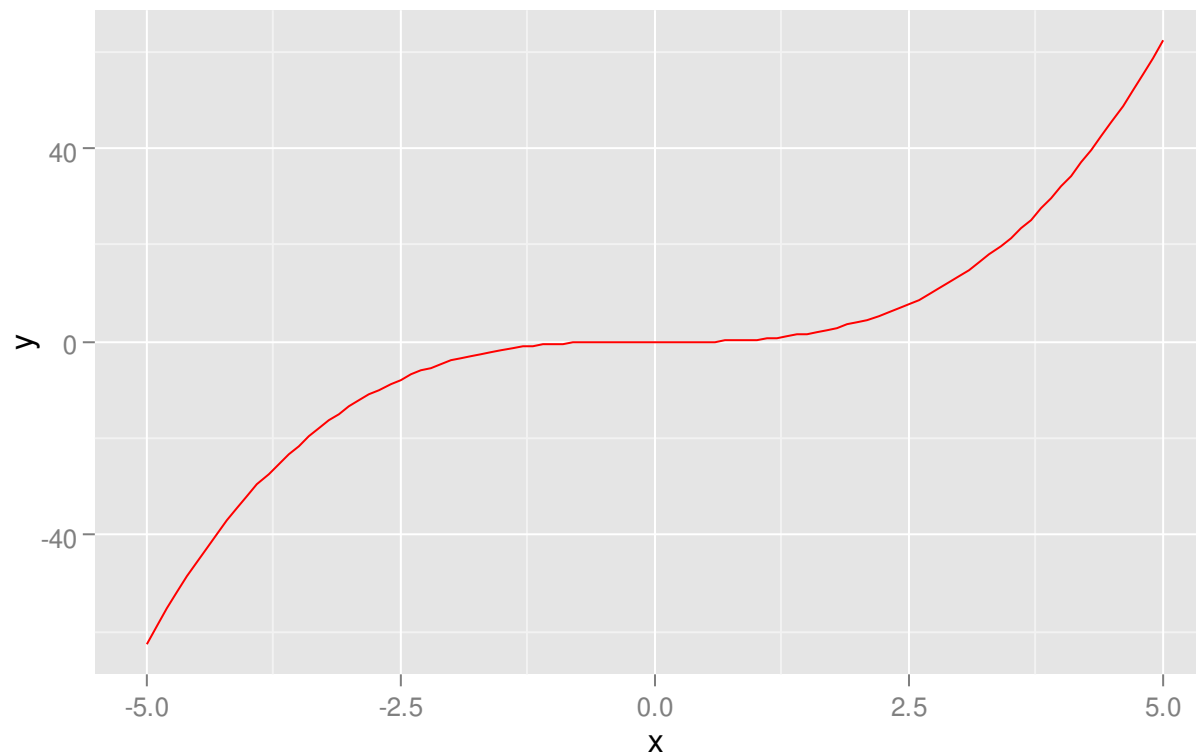
- 数学関数を描く場合は関数 `stat_function()` を用いる



例5: 数学関数のプロット

```
> f <- function(x) x^3/2  
> base <- ggplot(data.frame(x = c(-5, 5)), aes(x))  
> base + stat_function(fun=f, color="red")
```

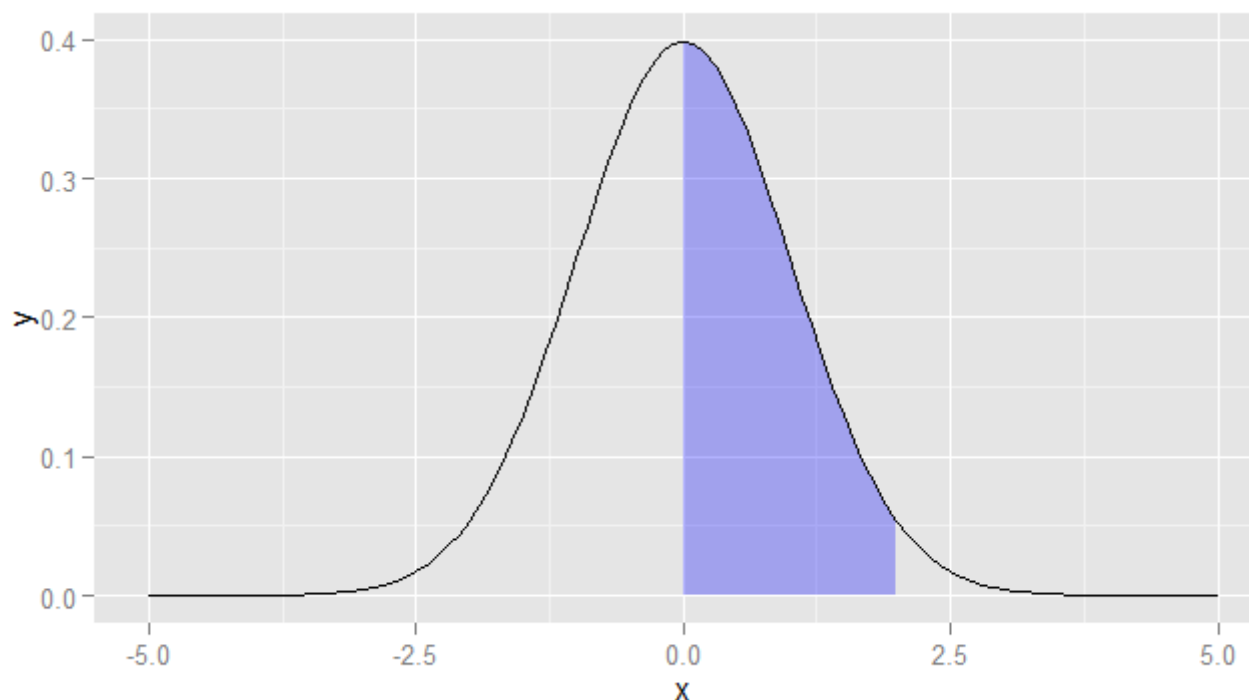
- 数学関数を描く場合は関数 `stat_function()` を用いる



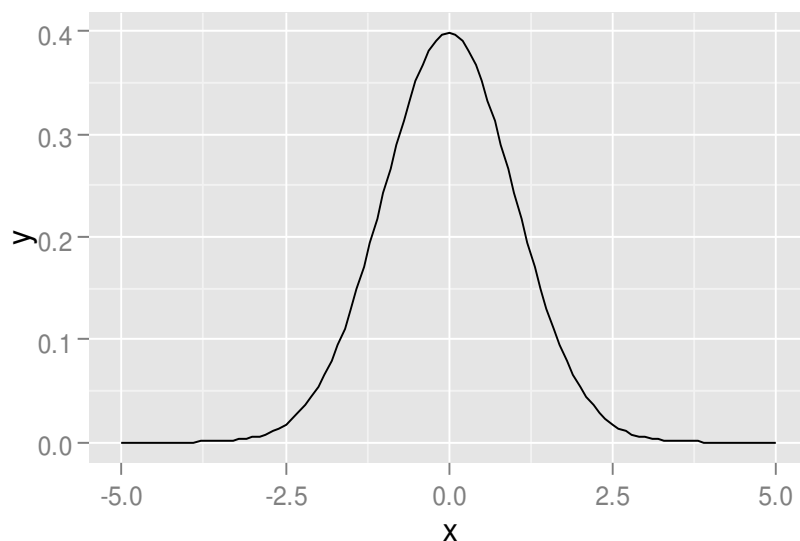
例5: 数学関数のプロット

```
> xlimit <- function(x) { y <- dnorm(x); y[x<0 | x>2] <- NA; y }  
> base <- ggplot(data.frame(x = c(-5, 5)), aes(x))  
> base + stat_function(fun=xlimit, geom="area", fill="blue", alpha=0.3) +  
+ stat_function(fun=dnorm)
```

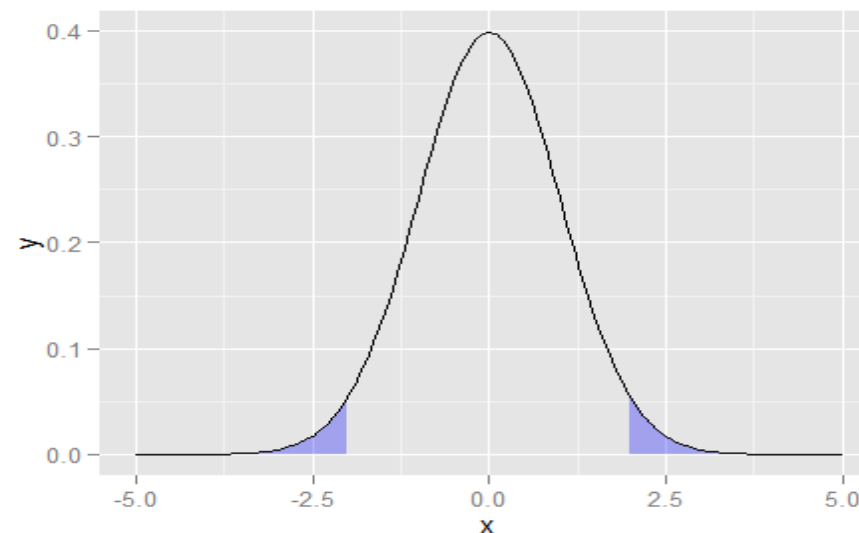
- 数学関数を描く場合は関数 `stat_function()` を用いる
- 一部の領域を色で塗ることも出来る(引数 `alpha` で透過も可)



例5: 数学関数のプロット



```
> ggplot(data.frame(x=c(-5,5)),
+   aes(x)) +
+   stat_function(fun=dnorm,
+   args=list(mean=0, sd=1))
> # 数学関数に x 以外の変数の指定は
> # 引数 args にリストを指定する
```



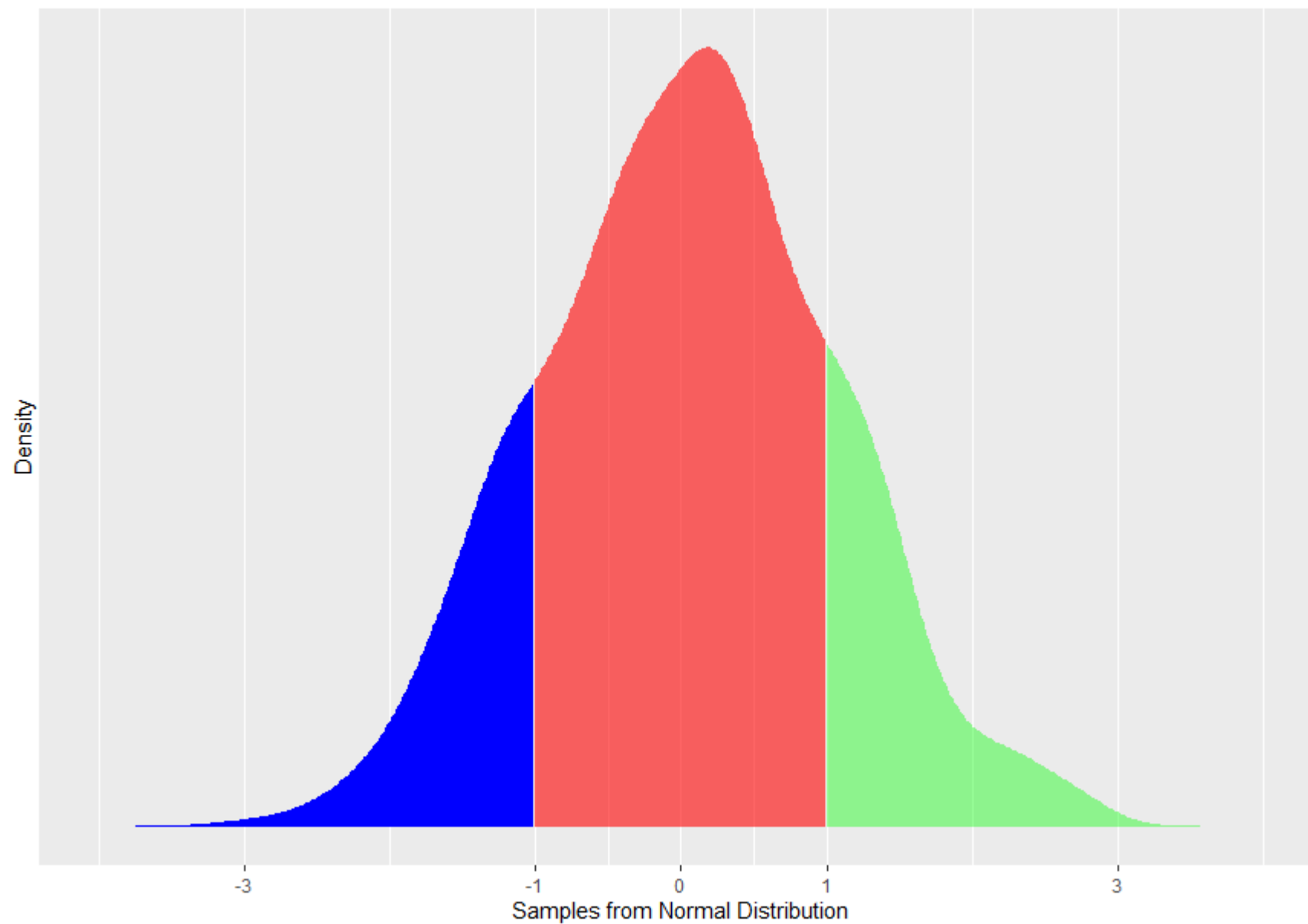
```
> q <- qnorm(0.975)
> mydnorm <- function(x) {
+   y <- ifelse(x < -q | x > q, dnorm(x), NA)
+   return(y)
+ }
> ggplot(data.frame(x=c(-5,5)), aes(x)) +
+   stat_function(fun=mydnorm, geom="area",
+   fill="blue", alpha=0.3) +
+   stat_function(fun=dnorm,
+   args=list(mean=0, sd=1))
```

例6: 密度関数の色分け

- 関数 `geom_ribbon()` と関数 `geom_area()` の適用例

```
> tmp <- rnorm(1000, 0, 1)
> df <- data.frame(density(tmp)$x, density(tmp)$y)
> df$x <- df$density.tmp..x
> df$y <- df$density.tmp..y
> ggplot(data=df, aes(x=x, y=y)) +
+   geom_ribbon(data=subset(df, x < -1), aes(ymax=y), ymin=0, fill="blue") +
+   geom_area(data=subset(df, -1 <= x & x <= 1), fill="red", alpha=.6) +
+   geom_area(data=subset(df, x > 1), fill="green", alpha=0.4) +
+   scale_y_continuous(limits=c(0,max(df$y)), breaks=NULL) +
+   scale_x_continuous(limits=c(-4,4), breaks=c(-3, -1, 0, 1,3)) +
+   labs(x="Samples from Normal Distribution", y="Density") +
+   theme(legend.position="none")
```

例6: 密度関数の色分け

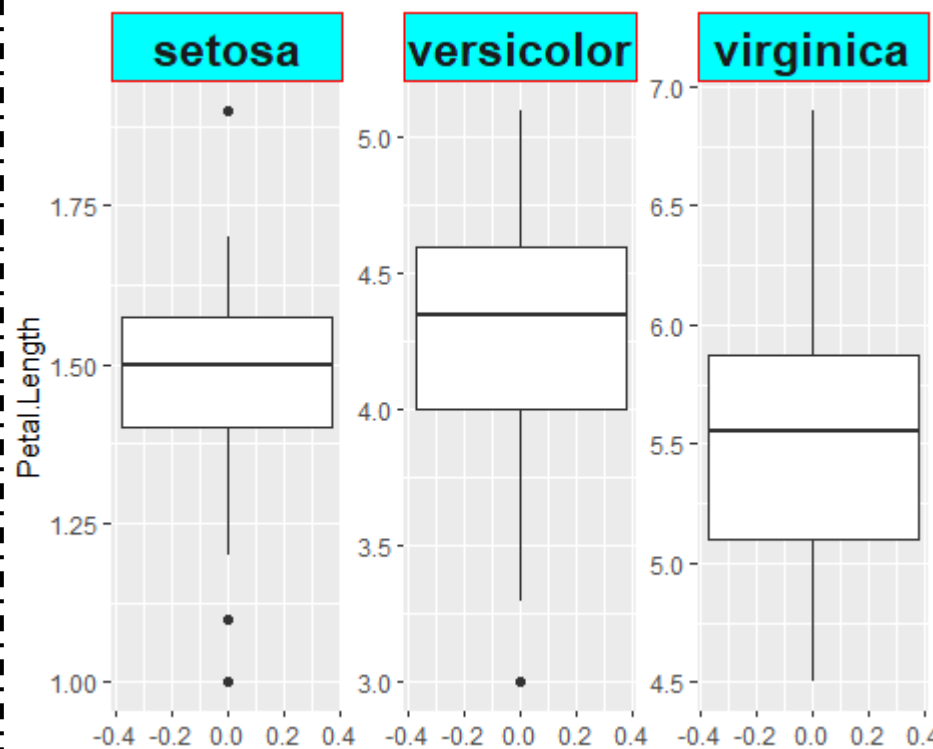
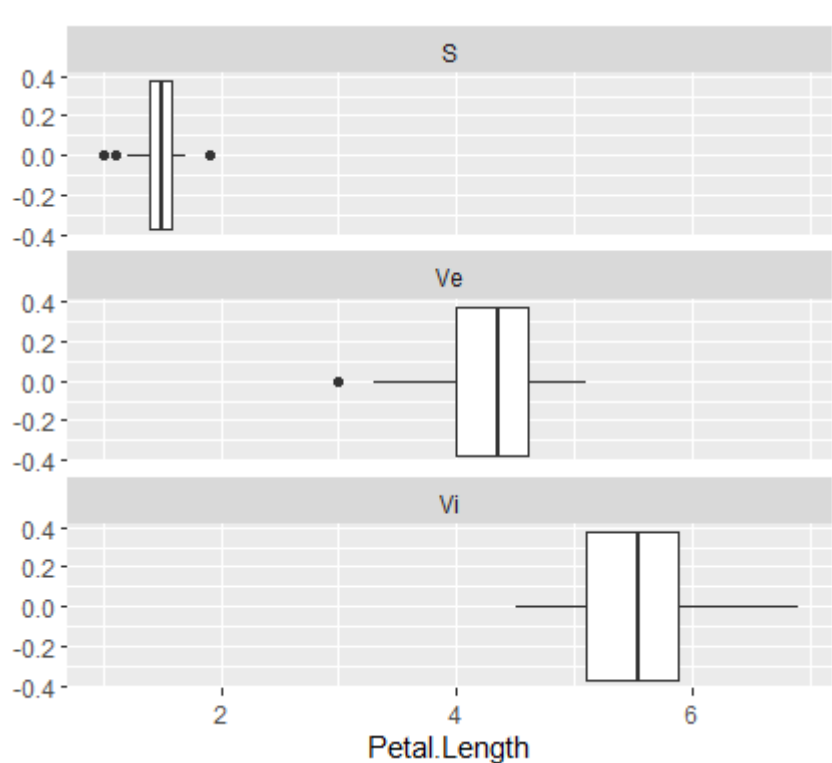


参考: 層別グラフの作成

- 例1の通り、関数 `facet_wrap()` と `facet_grid()` で層別グラフが作成出来る
 - `facet_wrap(~ 変数, nrow=1, ncol=2)`: 1×2 の層別グラフ
 - `facet_wrap(~ 変数1 + 変数2, nrow=1, ...)`: 複数の変数による層別グラフ
 - `facet_grid(行変数 ~)`: 行変数にて層別グラフ
 - `facet_grid(~ 列変数)`: 列変数にて層別グラフ
 - `facet_grid(行変数 ~ 列変数)`: 行変数と列変数にて層別グラフ
- 関数 `facet_wrap()` と `facet_grid()` の
 - 引数 `labeller` と関数 `labeller()` で各グラフのラベルを制御
 - 引数 `scales` に "free_x" (x軸)、"free_y" (y軸)、"free" (両軸) を指定
- 例1の通り、関数 `coord_flip()` でグラフの転置を行う
- 関数 `theme()` で見た目の制御 (各グラフのラベルの書式等)

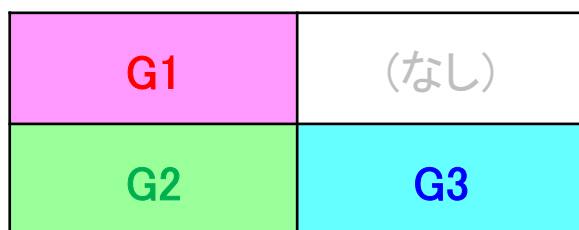
参考: 層別グラフの作成

```
> mylabel=c("setosa"="S", "versicolor"="Ve", "virginica"="Vi")
> ggplot(iris, aes(y=Petal.Length)) + coord_flip() + geom_boxplot() +
+   facet_wrap(~ Species, nrow=3, ncol=1, labeller=labeler(Species=mylabel))
>
> ggplot(iris, aes(y=Petal.Length)) +
+   geom_boxplot() + facet_wrap(~ Species, nrow=1, ncol=3, scales="free_y") +
+   theme(strip.text=element_text(face="bold", size=rel(1.5)),
+         strip.background=element_rect(fill="cyan", color="red"))
```

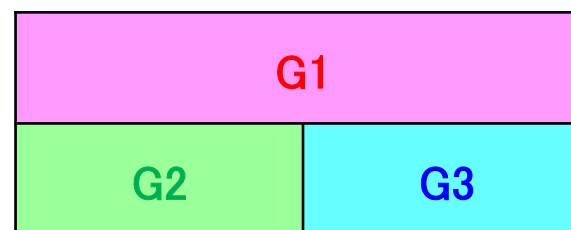


参考: 複数のグラフ作成

- 例3の通り、gridExtra パッケージの関数 `grid.arrange()` で複数のグラフを1枚にまとめることができる
- 関数 `grid.arrange(プロットオブジェクト1, プロットオブジェクト2, ...)`
 - 引数 `nrow` と `ncol` で行数と列数を指定
 - 引数 `heights` と `widths` で高さと幅を指定
 - 引数 `nrow` と `ncol` の代わりに、引数 `layout_matrix` に行列を指定することで複雑な配置が可能
⇒ プロットオブジェクト **G1**、**G2**、**G3** を順番に指定した後...



```
layout_matrix=
  matrix(c(1, NA,
           2, 3), nrow=2, byrow=T)
```



```
layout_matrix=
  matrix(c(1, 1,
           2, 3), nrow=2, byrow=T)
```

- プロットの配置の詳細については“Arranging multiple grobs on a page” by Baptiste Auguie にて <https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html>

参考: 複数のグラフ作成

- プロットオブジェクト G1、G2、G3 を作成

```
> G0 <- ggplot(iris, aes(Petal.Length))  
> G1 <- G0 + geom_histogram()  
> G2 <- G0 + geom_density(color="blue")  
> G3 <- G0 + geom_histogram(aes(y = ..density..)) +  
+       geom_density(color="red")
```

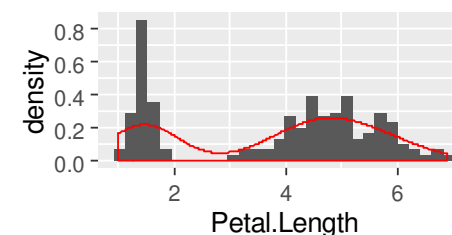
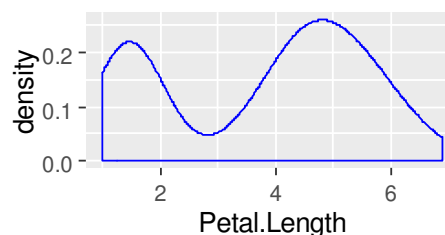
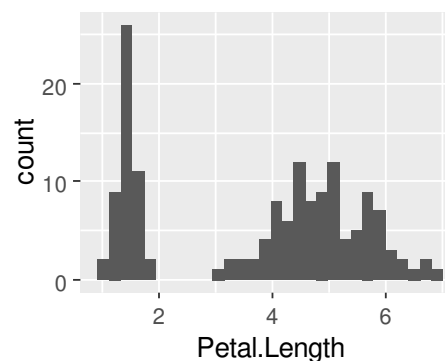
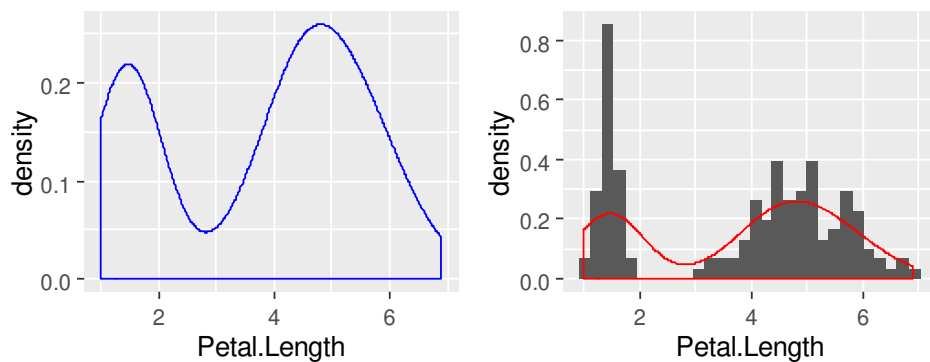
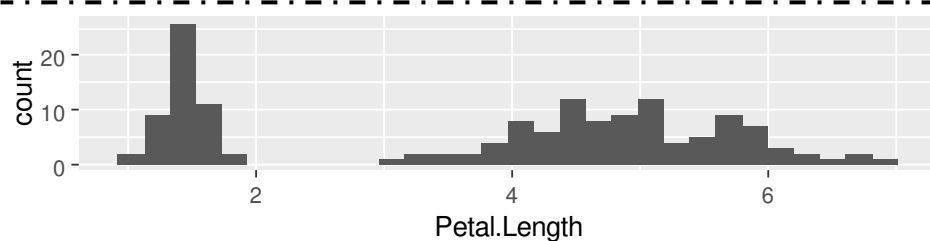
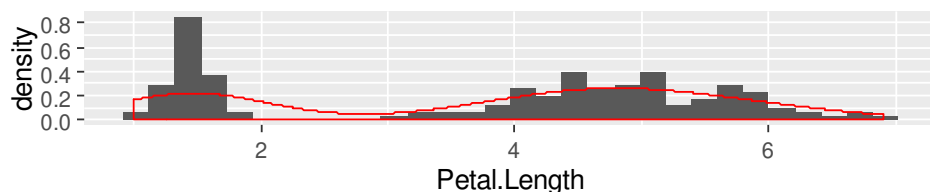
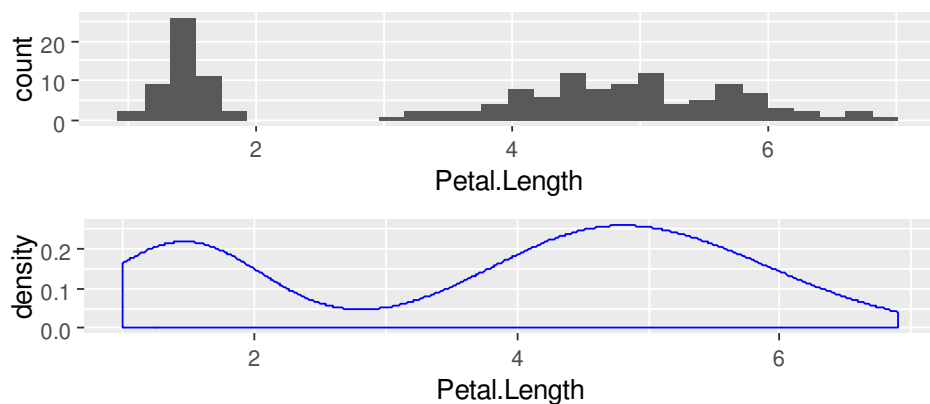
- プロットオブジェクト G1、G2、G3 を縦に並べる

```
> library(gridExtra)  
> grid.arrange(G1, G2, G3, nrow=3, heights=c(1,1,1))
```

- プロットオブジェクト G1、G2、G3 を行列にて配置する

```
> LM1 <- matrix(c(1,NA,2,3), nrow=2, byrow=T)  
> grid.arrange(G1, G2, layout_matrix=LM1, heights=c(1.5,1))  
  
> LM2 <- matrix(c(1,1,2,3), nrow=2, byrow=T)  
> grid.arrange(G1, G2, layout_matrix=LM2, heights=c(1.5,1))
```

参考: 複数のグラフ作成



参考: 表の追加

- パッケージ「grid」「gridExtra」で、グラフに表を追加することが出来る
 - まず、プロットオブジェクトと表オブジェクトを作成する
 - 表オブジェクトは関数 `tableGrob()` で作成
 - `tableGrob(データフレーム, theme=表のテーマ, rows=行ラベルに関するベクトル or NULL, cols=列ラベルに関するベクトル or NULL)`
 - 表のテーマは以下が使用可能
 - `ttheme_default(base_size=12, base_color="black", ...)`
 - `ttheme_minimal(base_size=12, base_color="black", ...)`
- プロットオブジェクトと表オブジェクトの配置方法は 54 頁と同じ
⇒ 関数 `grid.arrange(プロットオブジェクト1, プロットオブジェクト2, ...)` で配置
 - 引数 `nrow` と `ncol` で行数と列数、引数 `heights` と `widths` で高さと幅を指定
 - 引数 `nrow` と `ncol` の代わりに、引数 `layout_matrix` に行列を指定することで複雑な配置が可能

参考: 表の追加

- プロットオブジェクト G と表オブジェクト T1 を作成後、関数 `grid.arrange` で横に並べる

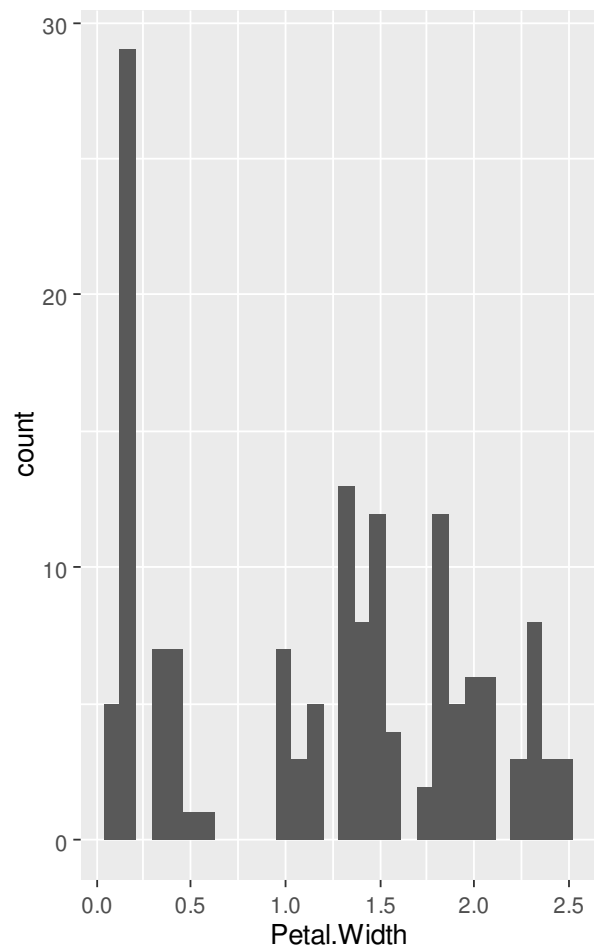
```
> library(gridExtra)
> G <- ggplot(iris, aes(Petal.Width)) + geom_histogram()
> T1 <- tableGrob(iris[1:16,4:5], rows=NULL, cols=c("Data", "Species"),
+               theme=ttheme_minimal())
> grid.arrange(G, T1, nrow=1, widths=c(2,1))
```

- 表オブジェクト T0 を追加作成後、関数 `grid.arrange` で結合

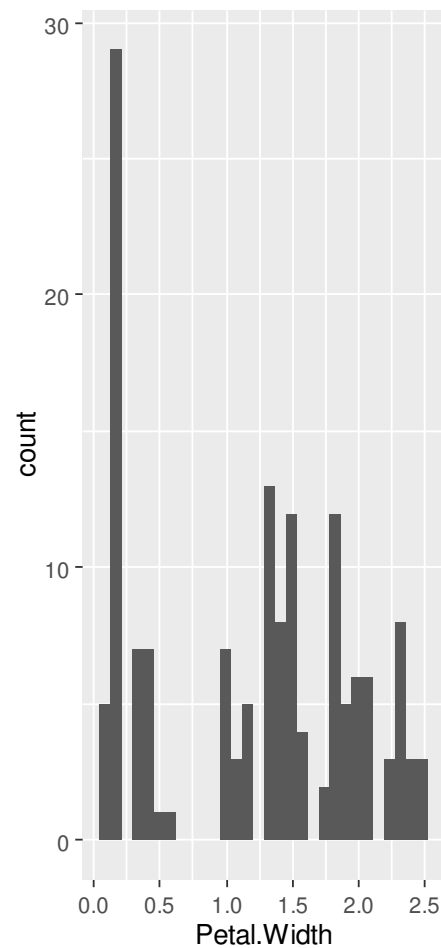
```
> T0 <- tableGrob(data.frame(x="Iris Data"), rows=NULL, cols=NULL,
+               theme=ttheme_minimal(base_size=15, base_color="black"))
> LM <- matrix(c(1,2,1,3), nrow=2, byrow=T)
> grid.arrange(G, T0, T1, layout_matrix=LM, heights=c(1,10))
```

- 表オブジェクトの詳細は “Displaying tables as grid graphics” by Baptiste Auguie にて <https://cran.r-project.org/web/packages/gridExtra/vignettes/tableGrob.html>
- プロットと表の配置については “Arranging multiple grobs on a page” by Baptiste Auguie にて <https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html>

参考: 表の追加



Data	Species
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
0.4	setosa
0.3	setosa
0.2	setosa
0.2	setosa
0.1	setosa
0.2	setosa
0.2	setosa
0.1	setosa
0.1	setosa
0.2	setosa
0.4	setosa
0.4	setosa



Data	Species
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
0.2	setosa
0.4	setosa
0.3	setosa
0.2	setosa
0.2	setosa
0.1	setosa
0.2	setosa
0.2	setosa
0.1	setosa
0.1	setosa
0.2	setosa
0.4	setosa

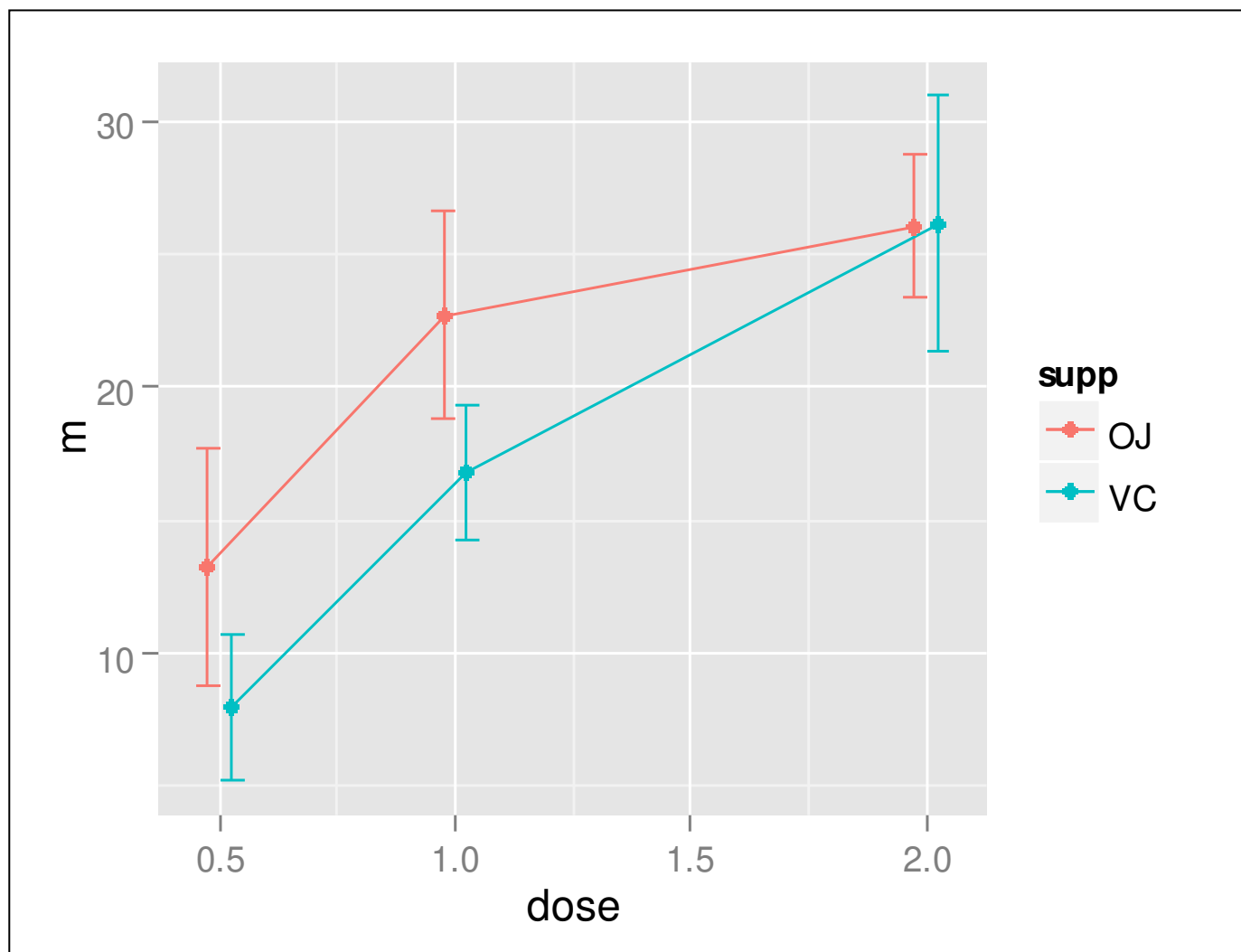
メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

グラフのカスタマイズ

- 今までの説明で、ggplot2 の仕組みは体系だっているように見えるが、実際にいくつかグラフを作成すると結構困ることが…
- 個人的には以下の流れで習得されるのが宜しいかと思われる
 1. 本スライドに出てくる基本的な事項は押さえる
(基本的な事項を知らずに 2 以降に進むと、余計に混乱します)
 2. 実際にグラフを描いてみる、エラーが出た場合はヘルプを参照する
 3. それでも分からないときは以下にて調べる
(自分で書いてみて「困った」という経験をしないと理解が深まりません)
 - R Graphics Cookbook, 2nd edition
<https://r-graphics.org/>
 - ggplot2 のマニュアル
<https://ggplot2.tidyverse.org/reference/>
 - 上記でダメなら Google でいろいろ検索

平均値の推移図とカスタマイズ



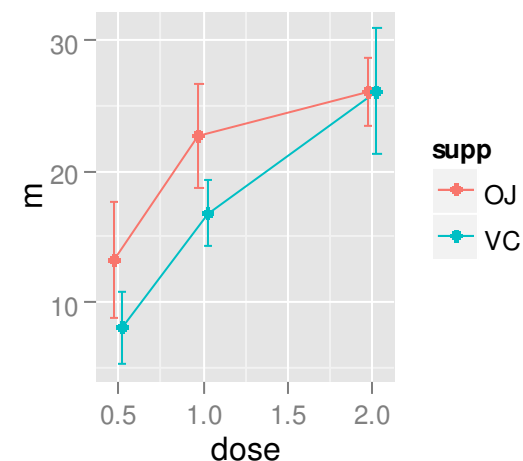
平均値の推移図

```
> ( TG <- summarise(group_by(ToothGrowth, supp, dose),
                           m=mean(len), s=sd(len)) )
```

Source: local data frame [6 x 4]

Groups: supp [?]

	supp (fctr)	dose (dbl)	m (dbl)	s (dbl)
1	OJ	0.5	13.23	4.459709
2	OJ	1.0	22.70	3.910953
3	OJ	2.0	26.06	2.655058
4	VC	0.5	7.98	2.746634
5	VC	1.0	16.77	2.515309
6	VC	2.0	26.14	4.797731



```
> pd <- position_dodge(.1)
> ggplot(TG, aes(x=dose, y=m, color=supp)) +
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +
+   geom_line(position=pd) + geom_point(position=pd)
```

- 各 supp、各 dose の「len の平均値 m」と「len の標準偏差 s」を算出した後、関数 **ggplot()** 等で平均値の推移図を描くことが出来る

平均値の推移図

```
> ( TG <- summarise(group_by(ToothGrowth, supp, dose),
                           m=mean(len), s=sd(len)) )
```

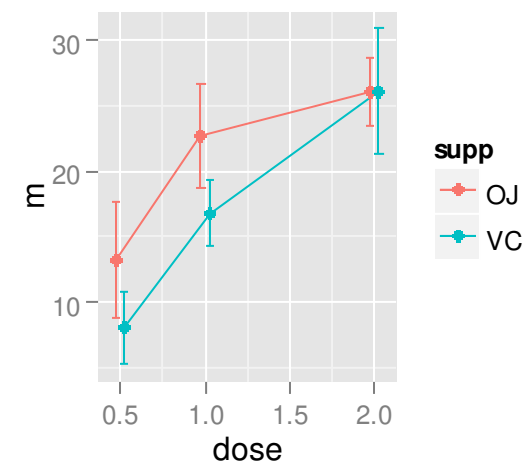
Source: local data frame [6 x 4]

Groups: supp [?]

	supp (fctr)	dose (dbl)	m (dbl)	s (dbl)
1	OJ	0.5	13.23	4.459709
2	OJ	1.0	22.70	3.910953
3	OJ	2.0	26.06	2.655058
4	VC	0.5	7.98	2.746634
5	VC	1.0	16.77	2.515309
6	VC	2.0	26.14	4.797731

```
> pd <- position_dodge(.1)
```

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +
+   geom_line(position=pd) + geom_point(position=pd)
```

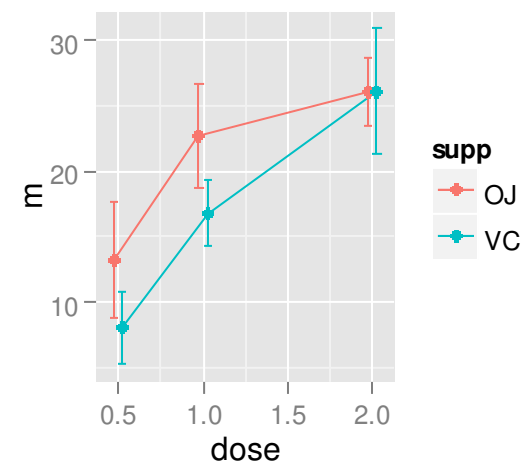


- 各 supp、各 dose の「len の平均値 m」「len の標準偏差 s」を算出
- 変数 pd に「プロットをズラす幅」を代入する
- 関数 **ggplot()** に横軸(dose)、縦軸(m)、エステ属性(supp)を指定

平均値の推移図

```
> pd <- position_dodge(.1)
> ggplot(TG, aes(x=dose, y=m, color=supp)) +
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +
+   geom_line(position=pd) +
+   geom_point(position=pd)
```

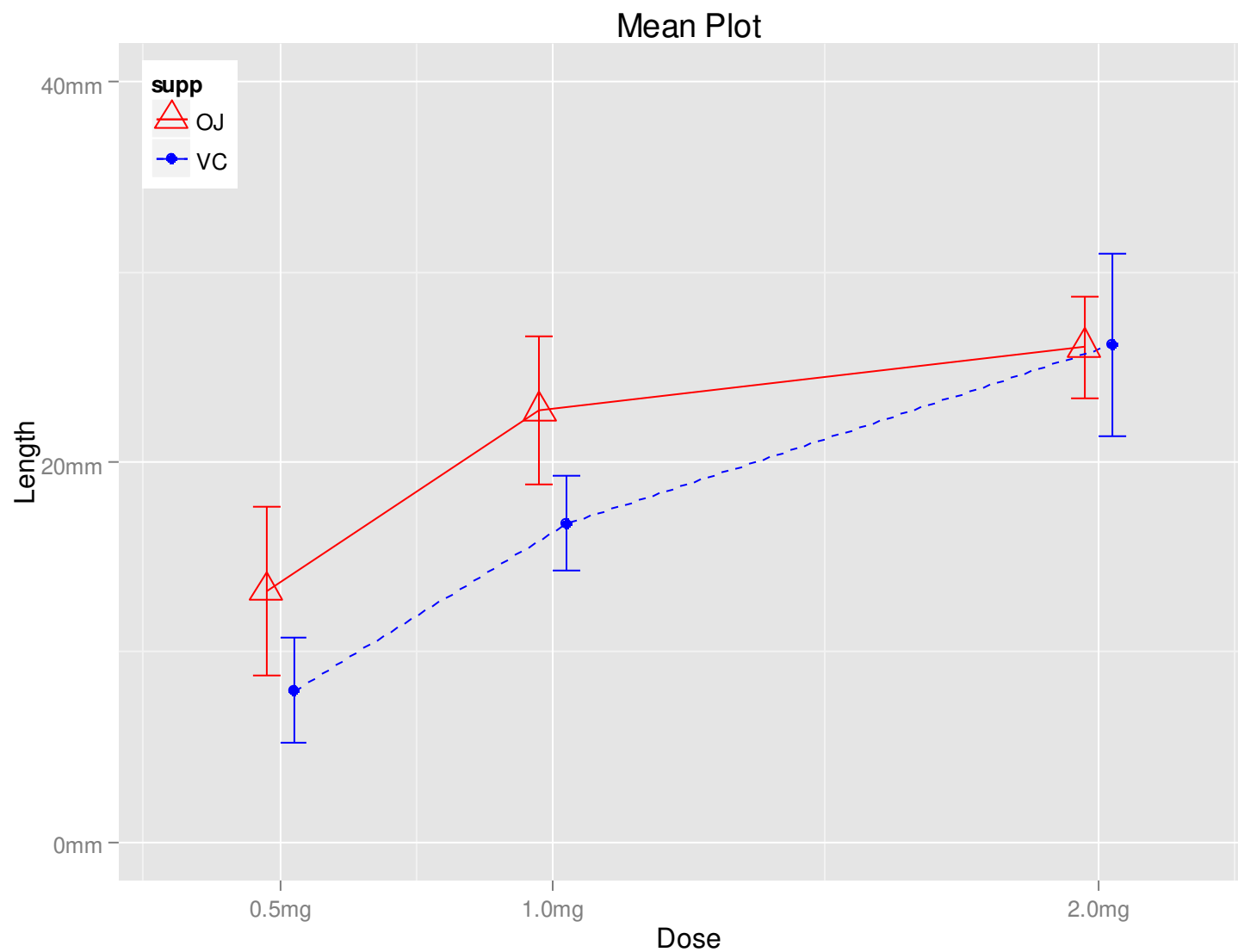
- 関数 `geom_errorbar()` でエラーバーのプロット
 - ✓ 引数 `ymin` と `ymax` でエラーバーの長さを指定
 - ✓ 引数 `width` でエラーバーの線の幅を指定
 - ✓ 引数 `position` で「プロットをズラす幅」を指定
- 関数 `geom_line()` で線のプロット
 - ✓ 引数 `position` で「プロットをズラす幅」を指定
 - ✓ 線の色は既にエステ属性 (`color=supp`) に紐付けられている
- 関数 `geom_point()` で点のプロット
 - ✓ 引数 `position` で「プロットをズラす幅」を指定
 - ✓ 線の色は既にエステ属性 (`color=supp`) に紐付けられている
- ちなみに、この図をもう少しカスタマイズすることも出来る



平均値の推移図

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(aes(linetype=supp), position=pd) +  
+   geom_point(aes(shape=supp, fill=supp, size=supp), position=pd) +  
+   scale_color_manual(values=c("red", "blue")) +  
+   scale_linetype_manual(values=c("solid", "dashed")) +  
+   scale_shape_manual(values=c(2, 19)) +  
+   scale_fill_manual(values=c("green", "yellow")) +  
+   scale_size_manual(values=c(5, 2)) +  
+   theme(legend.position=c(0.0, 0.8), legend.justification=c(0, 0)) +  
+   xlab("Dose") + ylab("Length") + ggtitle("Mean Plot") +  
+   scale_x_continuous(limits=c(0.3, 2.2), breaks=c(0.5, 1, 2),  
+     labels=c("0.5mg", "1.0mg", "2.0mg")) +  
+   scale_y_continuous(limits=c(0, 40), breaks=seq(0, 40, 20),  
+     labels=c("0mm", "20mm", "40mm"))
```

平均値の推移図



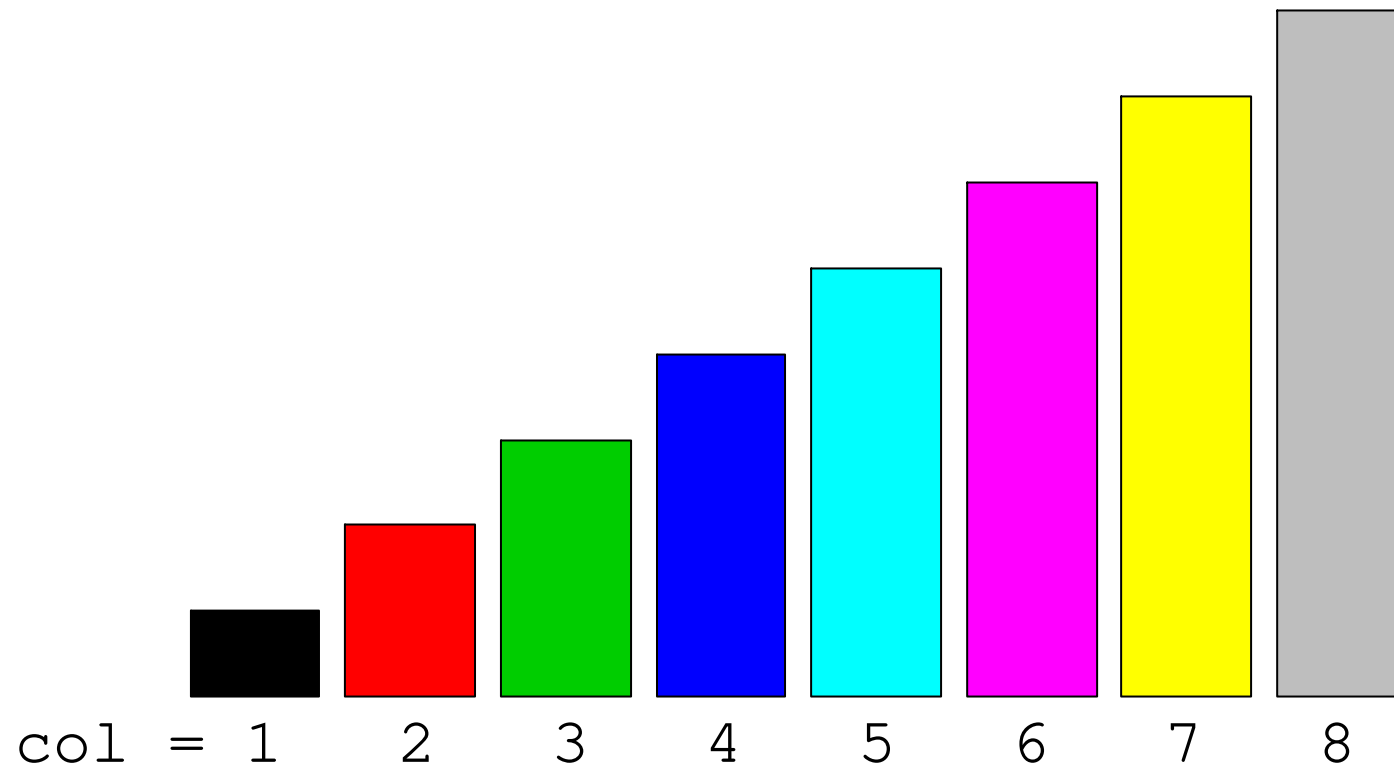
エステ属性と色の変更

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(aes(linetype=supp), position=pd) +  
+   geom_point(aes(shape=supp, fill=supp, size=supp), position=pd) +  
+   scale_color_manual(values=c("red", "blue")) +  
+   scale_linetype_manual(values=c("solid", "dashed")) +  
+   scale_shape_manual(values=c(2, 19)) +  
+   scale_fill_manual(values=c("green", "yellow")) +  
+   scale_size_manual(values=c(5, 2)) +  
+   theme(legend.position=c(0.0, 0.8), legend.justification=c(0, 0)) +  
+   xlab("Dose") + ylab("Length") + ggtitle("Mean Plot") +  
+   scale_x_continuous(limits=c(0.3, 2.2), breaks=c(0.5, 1, 2),  
+     labels=c("0.5mg", "1.0mg", "2.0mg")) +  
+   scale_y_continuous(limits=c(0, 40), breaks=seq(0, 40, 20),  
+     labels=c("0mm", "20mm", "40mm"))
```

- データをエステ属性 (color、linetype、shape...) に紐づけておくと、関数 `scale_XXX_manual()` で各群のプロットの見栄えが変更可能
- まず、関数 `ggplot()` で `color` のエステ属性を指定しているので、グラフ全体として変数 `supp` のカテゴリごとに色分けがなされる
- 色自体の調整は、関数 `scale_color_manual()` にて行う

色の種類

```
> barplot(1:8, col=1:8, axes=F)
```



色の種類

```
> colors()

[1] "white"           "aliceblue"       "antiquewhite"
[4] "antiquewhite1"   "antiquewhite2"   "antiquewhite3"
[7] "antiquewhite4"   "aquamarine"      "aquamarine1"
[10] "aquamarine2"     "aquamarine3"     "aquamarine4"
[13] "azure"           "azure1"          "azure2"
[16] "azure3"          "azure4"          "beige"
[19] "bisque"          "bisque1"         "bisque2"
[22] "bisque3"         "bisque4"         "black"
[25] "blanchedalmond"  "blue"            "blue1"
[28] "blue2"           "blue3"           "blue4"
[31] "blueviolet"      "brown"           "brown1"
[34] "brown2"          "brown3"          "brown4"

... ..

[655] "yellow3"         "yellow4"         "yellowgreen"
```

線種、点種、色塗り、点の大きさ

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(aes(linetype=supp), position=pd) +  
+   geom_point(aes(shape=supp, fill=supp, size=supp), position=pd) +  
+   scale_color_manual(values=c("red", "blue")) +  
+   scale_linetype_manual(values=c("solid", "dashed")) +  
+   scale_shape_manual(values=c(2, 19)) +  
+   scale_fill_manual(values=c("green", "yellow")) +  
+   scale_size_manual(values=c(5, 2)) +  
+   theme(legend.position=c(0.0, 0.8), legend.justification=c(0, 0)) +  
+   xlab("Dose") + ylab("Length") + ggtitle("Mean Plot") +  
+   scale_x_continuous(limits=c(0.3, 2.2), breaks=c(0.5, 1, 2),  
+     labels=c("0.5mg", "1.0mg", "2.0mg")) +  
+   scale_y_continuous(limits=c(0, 40), breaks=seq(0, 40, 20),  
+     labels=c("0mm", "20mm", "40mm"))
```

- 特定のグラフや図形にのみエステ属性を紐づけることも出来る
- 例えば、関数 `geom_line()` で `linetype` のエステ属性を指定しているので、線グラフのみ変数 `supp` のカテゴリごとに線の種類分けがなされる
- 線種の調整は、関数 `scale_line_manual()` にて行う
- 関数 `geom_point()` についても同様の仕組み



























関数 `scale_XXX_Manual()`

- エステ属性を変更するための関数はこちら
 - `scale_color_manual(values, ...)`: 点や線の色
 - `scale_fill_manual(values, ...)`: 塗りつぶしの色
 - `scale_size_manual(values, ...)`: 大きさ
 - `scale_shape_manual(values, ...)`: 点の種類
 - `scale_linetype_manual(values, ...)`: 線の種類
 - `scale_alpha_manual(values, ...)`: 図形の透明度
- 引数は以下のとおり
 - `values=c(2,19)` や `values=c("red","blue")`: 各カテゴリの属性を指定する
 - `labels=c("VitaminC","Orange")`: 各カテゴリの凡例のラベルを指定する
 - `limits=c("VC")`: 出力するカテゴリを絞る
- 当該属性(例えば色: `scale_color_manual()`)の凡例情報を削除したい場合は、然るべき箇所に `NULL/FALSE` を指定する
 - 関数 `guides(color=guide_legend(title=NULL))` で凡例のタイトルを削除
 - 関数 `guides(color=F)` にて一括で凡例を削除することも可

線の種類

引数 lty	種類
lty=0 , lty="blank"	(透明)
lty=1 , lty="solid"	
lty=2 , lty="dashed"	
lty=3 , lty="dotted"	
lty=4 , lty="dotdash"	
lty=5 , lty="longdash"	
lty=6 , lty="twodash"	

点の種類

pch	0	1	2	3	4	5	6	7	8
									
pch	9	10	11	12	13	14	15	16	17
									
pch	18	19	20	21	22	23	24	25	
									

凡例やタイトルの調整

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(aes(linetype=supp), position=pd) +  
+   geom_point(aes(shape=supp, fill=supp, size=supp), position=pd) +  
+   scale_color_manual(values=c("red", "blue")) +  
+   scale_linetype_manual(values=c("solid", "dashed")) +  
+   scale_shape_manual(values=c(2, 19)) +  
+   scale_fill_manual(values=c("green", "yellow")) +  
+   scale_size_manual(values=c(5, 2)) +  
+   theme(legend.position=c(0.0, 0.8), legend.justification=c(0, 0)) +  
+   xlab("Dose") + ylab("Length") + ggtitle("Mean Plot") +  
+   scale_x_continuous(limits=c(0.3, 2.2), breaks=c(0.5, 1, 2),  
+     labels=c("0.5mg", "1.0mg", "2.0mg")) +  
+   scale_y_continuous(limits=c(0, 40), breaks=seq(0, 40, 20),  
+     labels=c("0mm", "20mm", "40mm"))
```

- 関数 **theme()** で凡例の位置やプロット全体の体裁を修正する
- 関数 **xlab()**、**ylab()**、**ggtitle()** で軸やグラフのタイトルを指定する

凡例やタイトルの調整

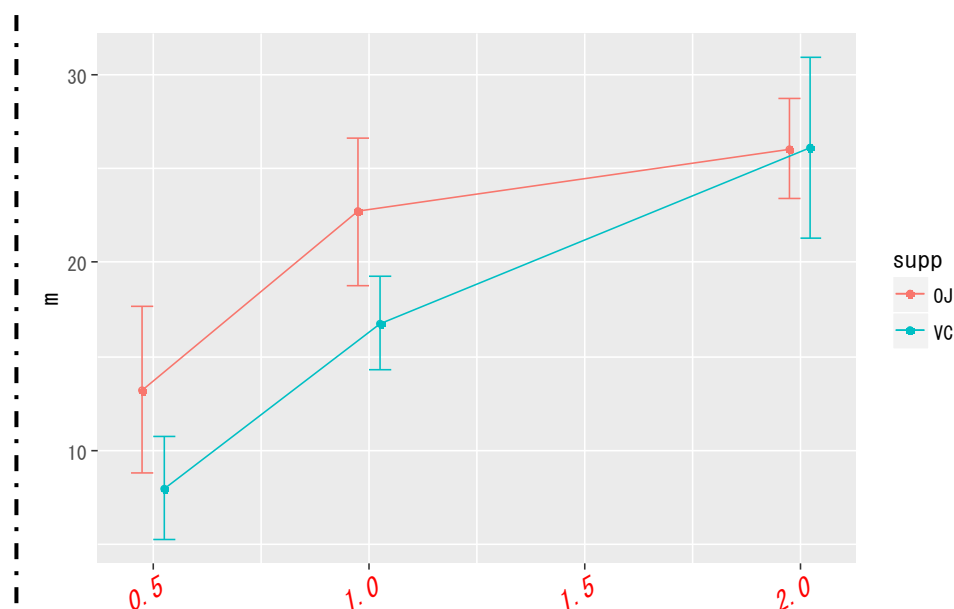
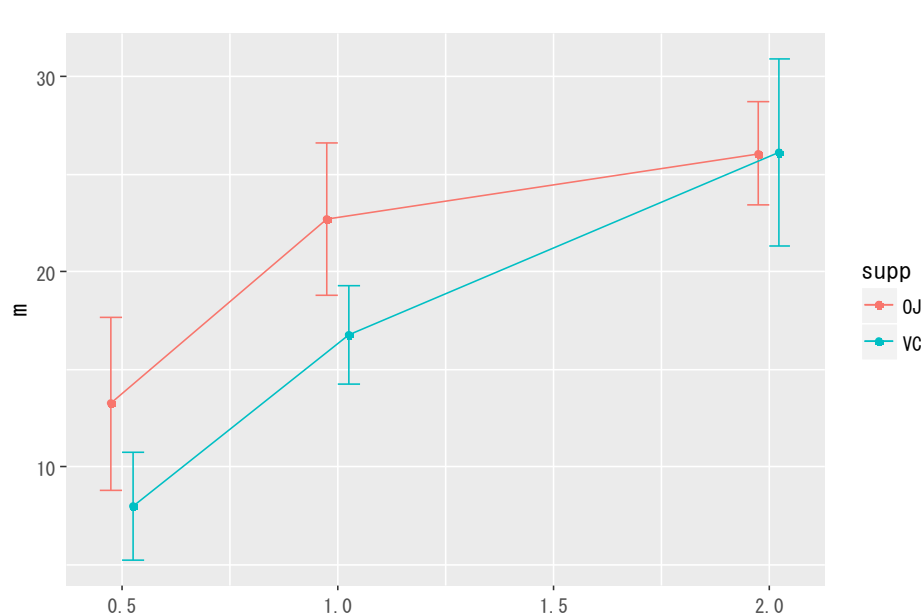
- 凡例の位置
 - `theme(legend.position=c(0.75,0), legend.justification=c(1,0))`: 特定の位置を指定 (0 ~ 1 で指定)
 - `theme(legend.position="top" "right" "bottom" "left" "none")` なる指定も可
- 凡例のラベルの調整については後述
- タイトル関係
 - `xlab("Dose")`: x 軸のタイトルを指定、`xlab(NULL)` や `xlab("")` で非表示に
 - `ylab("Length")`: y 軸のタイトルを指定、`ylab(NULL)` や `ylab("")` で非表示に
 - `ggtitle("Mean Plot")`: グラフのタイトルを指定
 - `labs(color="Supp.")`: 上記の全て + 凡例のタイトルも指定可能



関数 `theme()` によるタイトルや軸ラベルの変更

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(position=pd) + geom_point(position=pd) +  
+   #theme(axis.title.x=element_blank()) # x 軸のラベルを消す  
+   theme(axis.title.x=element_blank(),  
+         axis.text.x=element_text(angle=30, hjust=1, vjust=1,  
+                                   face="italic", colour="red", size=12))
```

- 関数 `theme()` の引数 `axis.text.x` にいろいろ指定することで、
x 軸のメモリのラベルの書式を変更したり、ラベルを消したりすることが出来る



軸のスケール調整

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +  
+   geom_errorbar(aes(ymin=m-s, ymax=m+s), width=.1, position=pd) +  
+   geom_line(aes(linetype=supp), position=pd) +  
+   geom_point(aes(shape=supp, fill=supp, size=supp), position=pd) +  
+   scale_color_manual(values=c("red", "blue")) +  
+   scale_linetype_manual(values=c("solid", "dashed")) +  
+   scale_shape_manual(values=c(2, 19)) +  
+   scale_fill_manual(values=c("green", "yellow")) +  
+   scale_size_manual(values=c(5, 2)) +  
+   theme(legend.position=c(0.0, 0.8), legend.justification=c(0, 0)) +  
+   xlab("Dose") + ylab("Length") + ggtitle("Mean Plot") +  
+   scale_x_continuous(limits=c(0.3, 2.2), breaks=c(0.5, 1, 2),  
+     labels=c("0.5mg", "1.0mg", "2.0mg")) +  
+   scale_y_continuous(limits=c(0, 40), breaks=seq(0, 40, 20),  
+     labels=c("0mm", "20mm", "40mm"))
```

- **scale_x_continuous()** と **scale_y_continuous()** で x 軸や y 軸の範囲等を調整することが出来る
- 引数 **trans** にて "asn" ($\tanh^{-1}(x)$)、"exp" (e^x)、"identity" (無変換)、
"log" ($\log(x)$)、"log10" ($\log_{10}(x)$)、"log2" ($\log_2(x)$)、"logit"
(ロジット関数)、"pow10" (10^x)、"probit" (プロビット関数)、
"recip" ($1/x$)、"reverse" ($-x$)、"sqrt" (平方根) 等の座標変換

座標やスケールの調整

- 座標の範囲
 - `xlim(c(0,2.5))`、`ylim(c(0,40))`: x 軸、y 軸の範囲
 - `scale_x_continuous(breaks=NULL, expand=c(0, 0))`:
x 軸の目盛を非表示にし、x 軸のマージンを 0 に
 - `scale_y_continuous(limits=c(0,40), breaks=seq(0,40,10), labels=c("0mm", "20mm", "40mm"))`: y 軸の範囲 + 刻み幅に関する情報
- ※ 離散データの場合は `scale_x_discrete()`、`scale_y_discrete()` を使用
- スケールの指定
 - `scale_x_date()`、`scale_y_date()`: 日付型
 - `scale_x_datetime()`、`scale_y_datetime()`: 日時型
 - `scale_x_log10()`、`scale_y_log10()`: 対数軸に
 - `scale_x_reverse()`、`scale_y_reverse()`: 逆順に表示
- 座標系の指定
 - `coord_fixed(ratio=1/2)`: 表示の際、y/x の比を 1/2 に
 - `coord_flip()`: x 軸と y 軸を逆に表示
 - `coord_polar()`: 極座標表示
 - `coord_trans(x="関数", y="関数")`: 座標変換(後述)

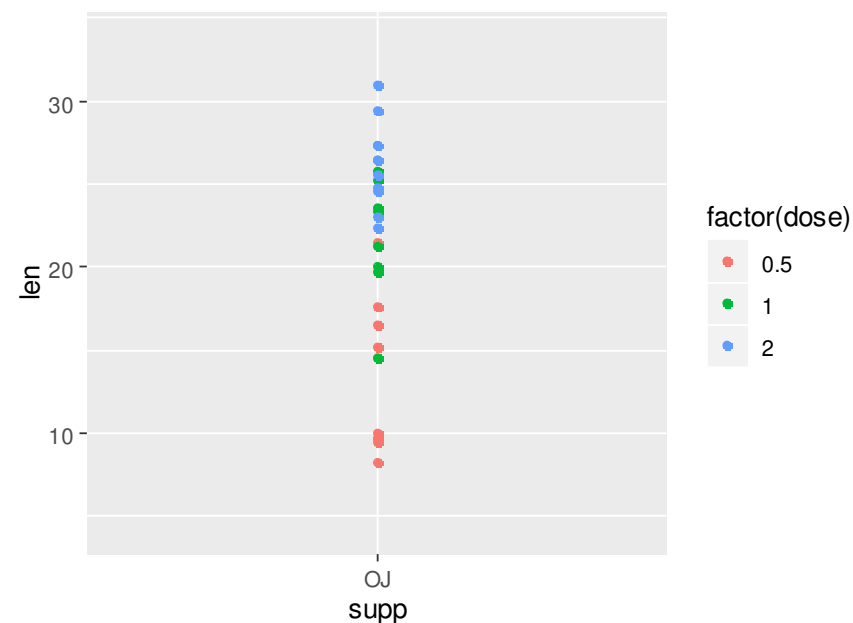
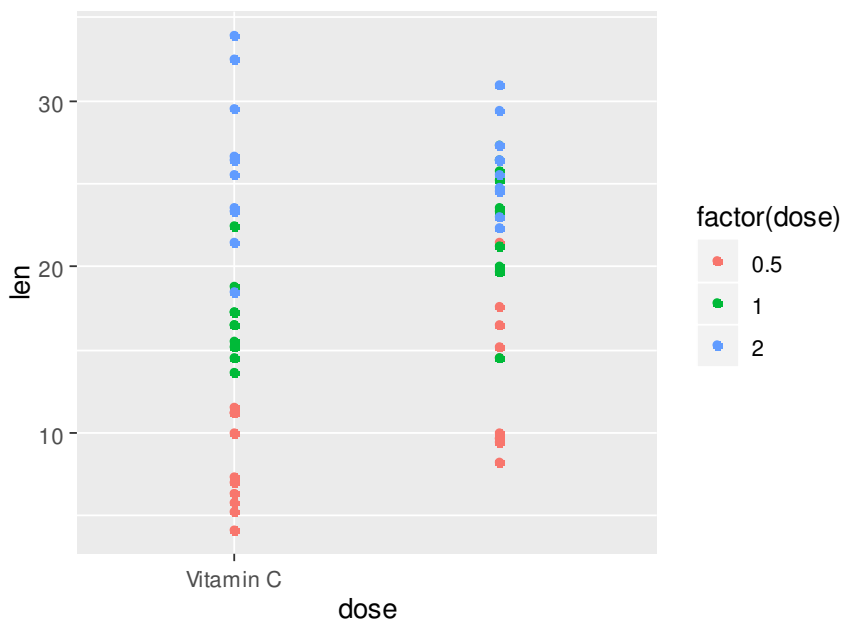
参考: 離散データの座標・スケールの調整

- 離散データの場合は `scale_x_discrete()`、`scale_y_discrete()` を使用
 - 例えば、**離散データのラベル変更**や**表示するカテゴリの選択**が可能

```
> points <- ggplot(ToothGrowth, aes(x=supp, y=len, color=factor(dose))) +
+   geom_point()
> points + scale_x_discrete("dose", limits=rev(levels(ToothGrowth$supp)),
+   labels=c("VC"="Vitamin C"), breaks="VC")
> points + xlim("OJ")
```

警告メッセージ:

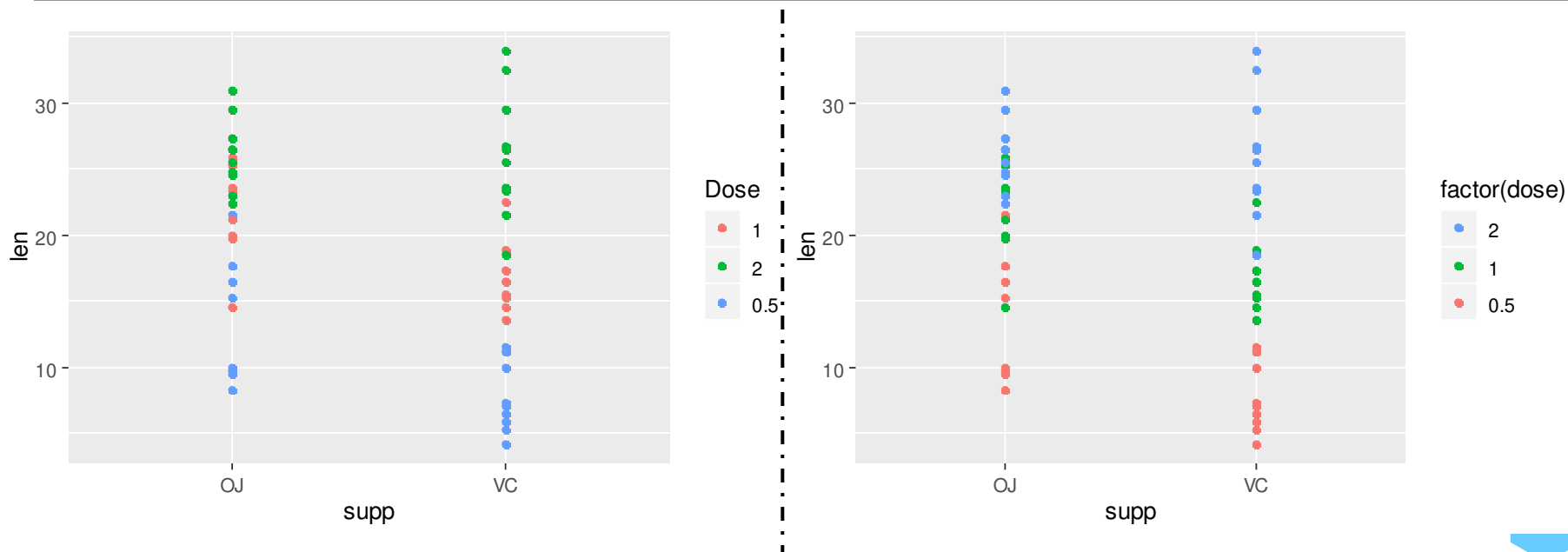
Removed 30 rows containing missing values (geom_point).



参考: 凡例の調整

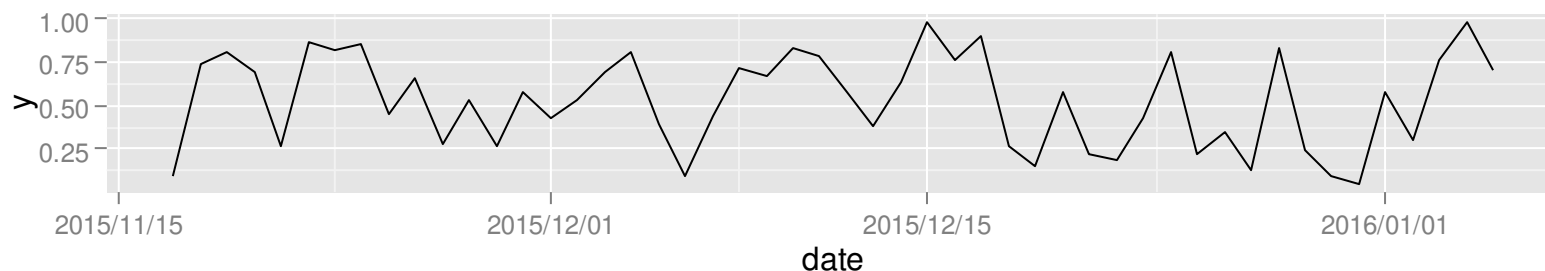
- 前頁の調整を行った場合でも凡例の表示は変わらない
→ 凡例は必要に応じて別途調整する必要がある
- 例えば、土台の関数 `aes` で引数 `color` にて群分けしている場合、関数 `scale_color_discrete()` でラベルの順番、関数 `guides()` で逆順などの指定が可

```
> points <- ggplot(ToothGrowth, aes(x=supp, y=len, color=factor(dose))) +  
+   geom_point()  
> points + scale_color_discrete(limits=c("1", "2", "0.5")) + labs(color="Dose")  
> points + guides(color=guide_legend(reverse=T))
```

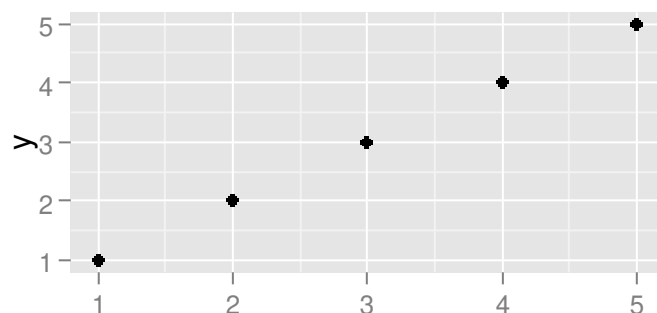


参考: 日付データの表示と座標変換の例

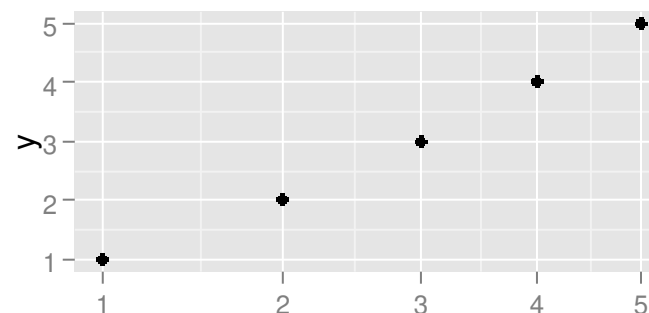
```
> df <- data.frame(date=seq(Sys.Date(), len=50, by="1 day"),
+                   y =runif(50))
> ggplot(df, aes(date,y)) + geom_line() +
+   scale_x_date(labels=date_format("%Y/%m/%d"))
```



```
> df <- data.frame(x=1:5, y=1:5)
> mysqrt_trans <- function() trans_new("mysqrt",
+   function(x) sqrt(x), function(x) x^2) # 変換式と逆変換式を定義
> ggplot(df, aes(x=x,y=y)) + geom_point() + coord_trans(x="mysqrt")
```



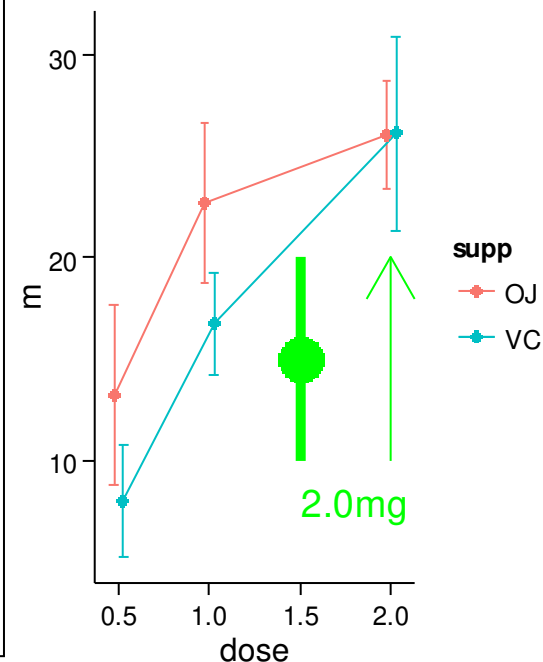
変換前



変換後

図形や文字の追記、見た目の変更

```
> ggplot(TG, aes(x=dose, y=m, color=supp)) +
+   geom_errorbar(aes(ymin=m-s, ymax=m+s),
+     width=.1, position=pd) +
+   geom_line(position=pd) +
+   geom_point(position=pd) +
+   annotate("text", x=1.8, y=8, label="2.0mg",
+     col="green") +
+   annotate("segment", x=2, xend=2, y=10,
+     yend=20, arrow=arrow(), col="green") +
+   annotate("pointrange", x=1.5, y=15, ymin=10,
+     ymax=20, colour="green", size=2) +
+   theme_classic()
```



- 関数 `annotate()` で図形や文字を追記することが出来る
 - 引数: "種類", x, y, xmin, ymin, xmax, ymax, エステ属性(例: color)
 - 描けるものの種類: "point", "pointrange", "rect", "segment", "text"
- 関数 `theme_XXX()` で全体的な見た目を変えることが出来る

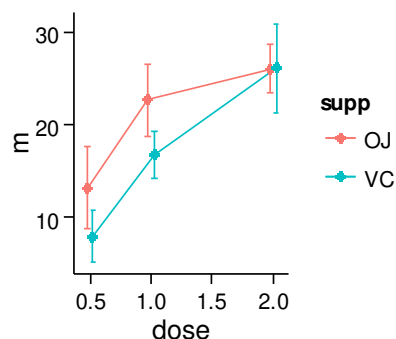


見た目の変更

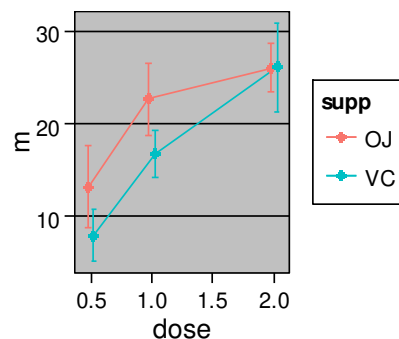
theme_bw()	theme_foundation()	theme_minimal()
theme_calc()	theme_gdocs()	theme_pander()
theme_classic()	theme_grey()	theme_solarized()
theme_economist()	theme_hc()	theme_solarized_2()
theme_economist_white()	theme_igray()	theme_solid()
theme_excel()	theme_light()	theme_stata()
theme_few()	theme_linedraw()	theme_tufte()
theme_fivethirtyeight()	theme_map()	theme_wsj()

※ 青字: 追加パッケージ「ggthemes」の中の関数

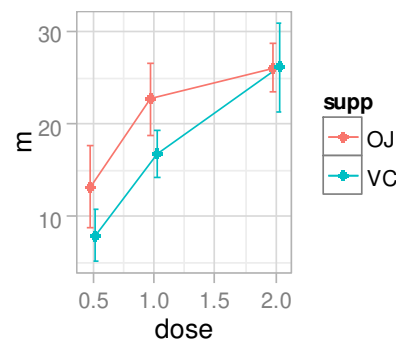
theme_classic()



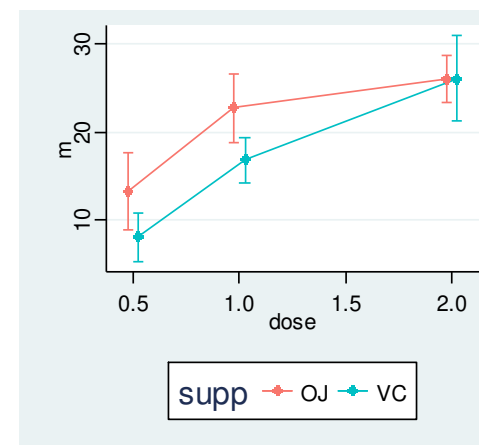
theme_excel()



theme_light()



theme_stata()



メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 theme() の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

関数 *theme()* の引数は多数...

```
line, rect, text, title, aspect.ratio, axis.title, axis.title.x,  
axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left,  
axis.title.y.right, axis.text, axis.text.x, axis.text.x.top,  
axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right,  
axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom,  
axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length,  
axis.ticks.length.x, axis.ticks.length.x.top, axis.ticks.length.x.bottom,  
axis.ticks.length.y, axis.ticks.length.y.left, axis.ticks.length.y.right,  
axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y,  
axis.line.y.left, axis.line.y.right, legend.background, legend.margin,  
legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,  
legend.key.size, legend.key.height, legend.key.width, legend.text,  
legend.text.align, legend.title, legend.title.align, legend.position,  
legend.direction, legend.justification, legend.box, legend.box.just,  
legend.box.margin, legend.box.background, legend.box.spacing,  
panel.background, panel.border, panel.spacing, panel.spacing.x,  
panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor,  
panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x,  
panel.grid.minor.y, panel.ontop, plot.background, plot.title,  
plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin,  
strip.background, strip.background.x, strip.background.y,  
strip.placement, strip.text, strip.text.x, strip.text.y,  
strip.switch.pad.grid, strip.switch.pad.wrap, ...
```

- よく使うものを青字で表記 → 詳細はマニュアル※や Winston(2013)を参照

※ <https://ggplot2.tidyverse.org/reference/>

関数 `theme()` の引数に与える設定関数

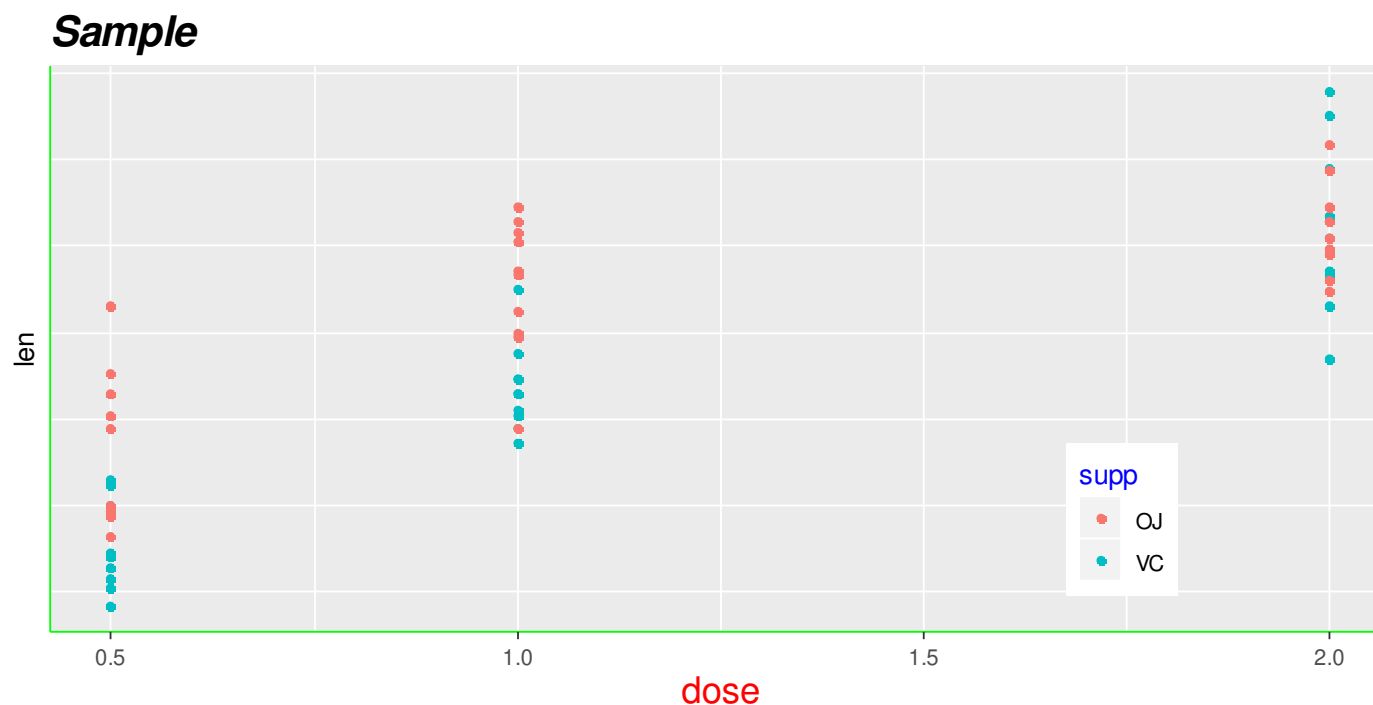
- 関数 `theme()` により様々なカスタマイズが行える
 - 例えば、`axis.title.x` と `axis.title.y` は x 軸と y 軸の文字、`axis.title` は両軸の文字の設定を行う → 慣れると引数を見ただけで何の設定用か分かるようになる
 - 関数 `theme()` は引数が多すぎるので、本資料では抜粋して紹介
- 関数 `theme()` の各引数(前頁)に、以下の関数により設定を与える
 - `element_blank()` : 指定された要素について何も描かず、スペースも空けない
 - `element_rect(fill = NULL, colour = NULL, size = NULL, linetype = NULL, color = NULL, inherit.blank = FALSE)` : 枠に関する設定
 - `element_line(colour = NULL, size = NULL, linetype = NULL, lineend = NULL, color = NULL, arrow = NULL, inherit.blank = FALSE)` : 線に関する設定
 - `element_text(family = NULL, face = NULL, colour = NULL, size = NULL, hjust = NULL, vjust = NULL, angle = NULL, lineheight = NULL, color = NULL, margin = NULL, debug = NULL, inherit.blank = FALSE)` : 文字に関する設定
 - `margin(t = 0, r = 0, b = 0, l = 0, unit = "pt")` : 余白に関する設定
 - `rel(x)` : サイズや大きさ等を相対値で指定する場合に指定、`size=rel(1.5)` 等

関数 *theme()* の引数(抜粋)

引数	機能	主に使用する <code>element_xxx()</code> 又は文字列
<code>line</code>	全ての線	<code>element_line()</code> 、 <code>element_blank()</code>
<code>rect</code>	全ての矩形	<code>element_rect()</code> 、 <code>element_blank()</code>
<code>text</code>	全ての文字	<code>element_text()</code> 、 <code>element_blank()</code>
<code>title</code>	タイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.title</code>	軸タイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.title.x</code>	x 軸タイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.title.y</code>	y 軸タイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.text</code>	軸目盛の文字	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.text.x</code>	x 軸目盛の文字	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.text.y</code>	y 軸目盛の文字	<code>element_text()</code> 、 <code>element_blank()</code>
<code>axis.line</code>	軸の線	<code>element_line()</code> 、 <code>element_blank()</code>
<code>axis.line.x</code>	x 軸の線	<code>element_line()</code> 、 <code>element_blank()</code>
<code>axis.line.y</code>	y 軸の線	<code>element_line()</code> 、 <code>element_blank()</code>
<code>legend.background</code>	凡例の背景色	<code>element_rect()</code> 、 <code>element_blank()</code>
<code>legend.text</code>	凡例の文字	<code>element_text()</code> 、 <code>element_blank()</code>
<code>legend.title</code>	凡例タイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>legend.position</code>	凡例の位置	"bottom"、"left"、"top"、"right"、 <code>c(0~1, 0~1)</code>
<code>plot.title</code>	グラフタイトル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>strip.background</code>	層別グラフのタイトルの背景色	<code>element_rect()</code> 、 <code>element_blank()</code>
<code>strip.text</code>	層別グラフのラベル	<code>element_text()</code> 、 <code>element_blank()</code>
<code>strip.text.x</code>	層別グラフのラベル(x 方向)	<code>element_text()</code> 、 <code>element_blank()</code>
<code>strip.text.y</code>	層別グラフのラベル(y 方向)	<code>element_text()</code> 、 <code>element_blank()</code>

関数 *theme()* の使用例

```
> ggplot(ToothGrowth, aes(x=dose, y=len, color=supp)) +  
+   geom_point() + ggtitle("Sample") +  
+   theme(axis.title.x=element_text(size=15, color="red"),  
+         axis.text.y=element_blank(), axis.ticks.y=element_blank(),  
+         axis.line=element_line(color="green"),  
+         plot.title=element_text(size = rel(1.5), face="bold.italic"),  
+         legend.position=c(0.8, 0.2),  
+         legend.title=element_text(color="blue"))
```



関数 *theme()* の使用例

```
> ggplot(ToothGrowth, aes(x=dose, y=len, color=supp)) +  
+   geom_point() + ggtitle("Sample") +  
+   theme(axis.title.x=element_text(size=15, color="red"),  
+         axis.text.y=element_blank(), axis.ticks.y=element_blank(),  
+         axis.line=element_line(color="green"),  
+         plot.title=element_text(size = rel(1.5), face="bold.italic"),  
+         legend.position=c(0.8, 0.2),  
+         legend.title=element_text(color="blue"))
```

- **axis.title.x**: x 軸ラベルの size を 15、色を赤に
- **axis.text.y**: y 軸目盛の文字を消す → `element_blank()`
- **axis.ticks.y**: y 軸目盛を消す → `element_blank()`
- **axis.line**: 軸の線を緑に
- **plot.title**: グラフタイトルの size を 1.5 倍、フォントを `bold.italic` に
- **legend.position**: グラフの左下の場所を (0, 0) とし、凡例の場所を (0.8, 0.2) に
- **legend.title**: 凡例タイトルの色を青に

メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- [応用例: Kaplan Meier Plot](#)
- 応用例: Forest Plot

使用するデータ③: AML

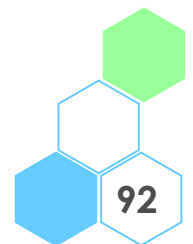
(Acute Myelogenous Leukemia)



- 急性骨髄性白血病(AML)を罹患された患者さんの生存時間データ
 - group: 維持化学療法の有無(1:あり、2:なし)
 - time: 生存時間
 - status: イベント(1)/打ち切り(0)

group	time	status
1	9	1
1	13	1
1	13	0
1	18	1
1	23	1
1	28	0
1	31	1
1	34	1
1	45	0
1	48	1
1	161	0

group	time	status
2	5	1
2	5	1
2	8	1
2	8	1
2	12	1
2	16	0
2	23	1
2	27	1
2	30	1
2	33	1
2	43	1
2	45	1

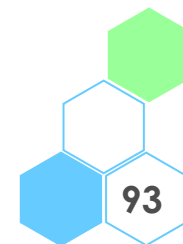


カプランマイヤー・プロット (Kaplan Meier Plot) 用 データ生成



```
> library(survival)
> aml$group <- ifelse(aml$x=="Maintained", 1, 2)
> result    <- survfit(Surv(time, status) ~ group, data = aml)
> # summary(result)
> # plot(result, col=2:1, xlab="Time", ylab="Survival Rate")
> # plot(result, col=2:1, xlab="Time", ylab="Cumulative Incidence", fun="event")
>
> # 1 群の場合
> df_tmp    <- data.frame(time=result$time, n.risk=result$n.risk,
+                          n.event=result$n.event, n.censor=result$n.censor,
+                          surv=result$surv)
> df_zero   <- data.frame(time=0, n.risk=result$n, n.event=0, n.censor=0, surv=1)
> df        <- rbind(df_zero, df_tmp)
> df$fail   <- 1 - df$surv
```

- まず、survival パッケージの関数 Surv() & survfit() を用いてカプランマイヤー推定を行った後、グラフ作成用のデータを作成する（1 群と 2 群以上で生成方法が若干異なる）
- 通常は 4～6 行目にて表やグラフを生成するが、今回は無視



Kaplan-Meier Plot (Kaplan Meier Plot) 用 データ生成



```
> # 2 群以上の場合
> mystrata <- c()
> for(i in 1:length(result$strata)) mystrata <- c(mystrata,
+   rep(names(result$strata)[i], result$strata[i]))
> df_tmp <- data.frame(time=result$time, n.risk=result$n.risk,
+   n.event=result$n.event, n.censor=result$n.censor,
+   surv=result$surv, strata=factor(mystrata))
> df <- NULL
> for(i in 1:length(result$strata)) {
+   mysubset <- subset(df_tmp, strata==names(result$strata)[i])
+   df_zero <- data.frame(time=0, n.risk=result$n[i], n.event=0, n.censor=0,
+   surv=1, strata=names(result$strata[i]))
+   df <- rbind(df, rbind(df_zero, mysubset))
+ }
> df$fail <- 1 - df$surv
> head(df, n=3)
```

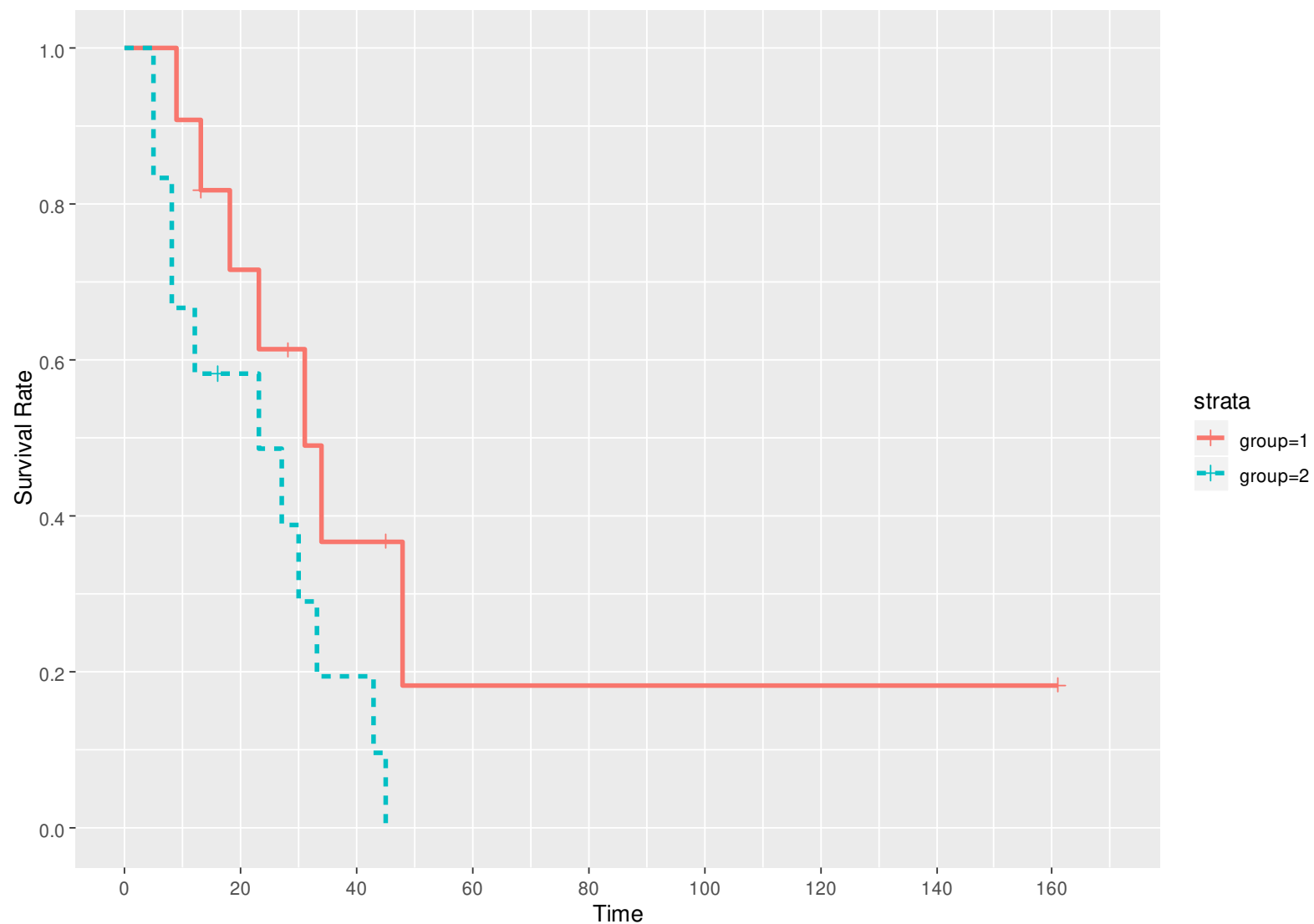
	time	n.risk	n.event	n.censor	surv	strata	fail
1	0	11	0	0	1.0000000	group=1	0.00000000
2	9	11	1	0	0.9090909	group=1	0.09090909
3	13	10	1	1	0.8181818	group=1	0.18181818

例1: *at risk* に関する表なし

```
> ggplot(data=df) +  
+   geom_step(mapping=aes(x=time, y=surv, color=strata,  
+   linetype=strata), size=1.2) +  
+   geom_point(data=subset(df, n.censor>0),  
+   mapping=aes(x=time, y=surv, color=strata), shape=3, size=2) +  
+   xlab("Time") + ylab("Survival Rate") +  
+   scale_x_continuous(breaks=seq(0, 160, by=20), limits=c(0, 170)) +  
+   scale_y_continuous(breaks=seq(0, 1, by=0.2), limits=c(0, 1))
```

- 前頁で生成したデータ df を使って階段関数を、df のうち打ち切りに絞ったデータを使って打ち切り点(+)を描画
- 変数 **surv** を変数 **fail** に変更することで、累積発生率に関するグラフを描くことが出来る
- 異なるデータを使って 1 枚のグラフを描く際は注意点があり、(通常は省略する)「**data=**」「**mapping=**」を明記しないと ggplot2 が混乱し、エラーを返す

例1: *at risk* に関する表なし



例2: *at risk* に関する表を泥臭く作成

- リスク集合に関するデータ *atRisk* を作成した上でグラフと表を併記
- R では統計処理の結果の取り出し&データ加工は手間が掛かる…

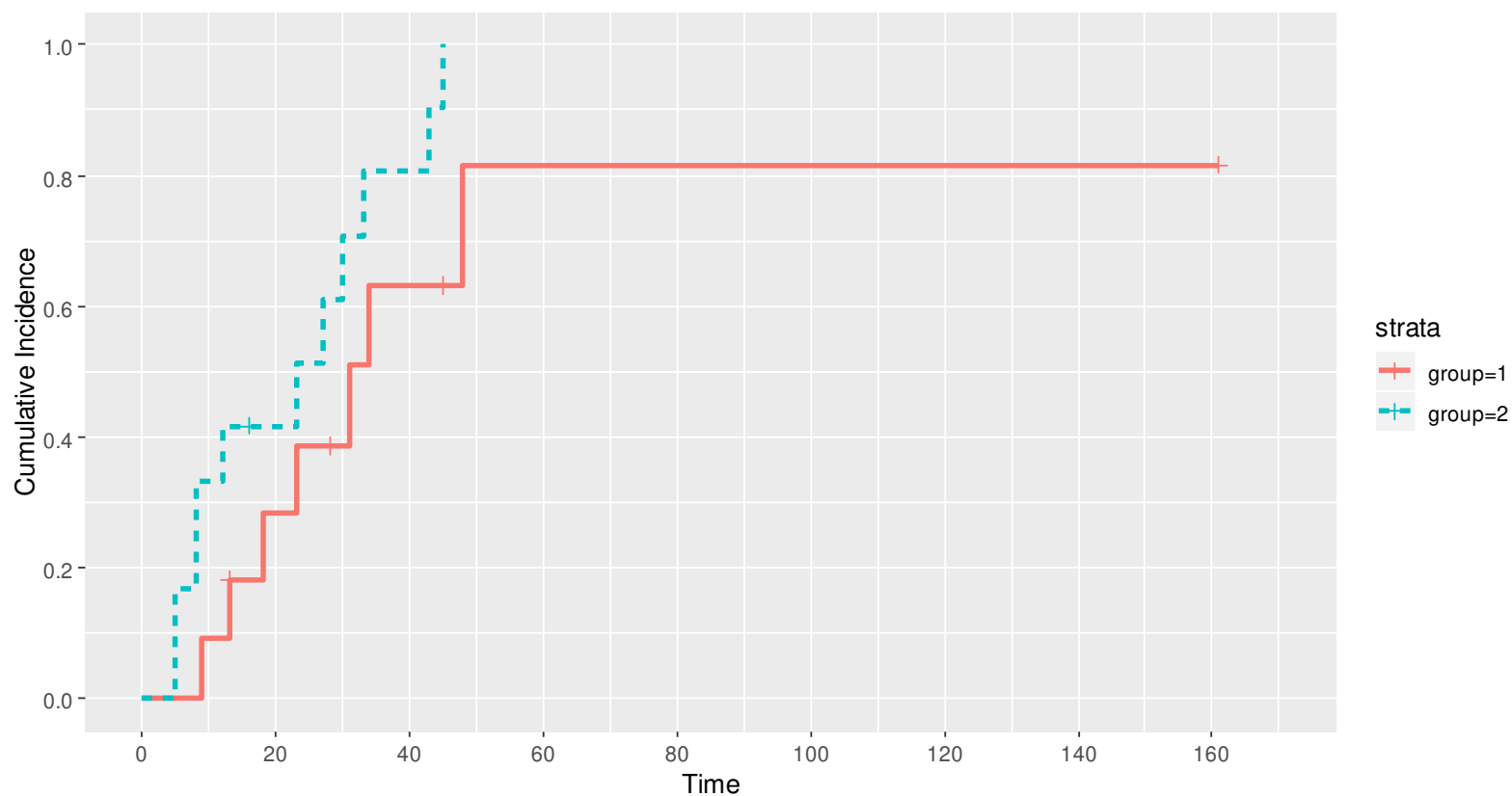
```
> result2      <- summary(result, time=seq(0,180,20))

> # 1 群の場合
> atrisk_tmp <- data.frame(time=result2$time, n.risk=result2$n.risk)

> # 2 群以上の場合
> atrisk_tmp      <- data.frame(time=result2$time, n.risk=result2$n.risk,
+                               strata=result2$strata)
> atrisk1          <- subset(atrisk_tmp, strata=="group=1")
> atrisk2          <- subset(atrisk_tmp, strata=="group=2")
> atRisk           <- subset(merge(atrisk1, atrisk2, by="time", all=T),
+                               select=c(time, n.risk.x, n.risk.y))
> atRisk$n.risk.x <- replace(atRisk$n.risk.x, which(is.na(atRisk$n.risk.x)), 0)
> atRisk$n.risk.y <- replace(atRisk$n.risk.y, which(is.na(atRisk$n.risk.y)), 0)
> names(atRisk)    <- c("Time", "Group=1", "Group=2")

> library(grid)
> library(gridExtra)
> TB <- tableGrob(t(atRisk), cols=NULL)
> TB$widths <- unit(rep(0.8/ncol(TB), ncol(TB)), "npc")
> grid.arrange(KM, TB, nrow=2, as.table=TRUE, heights=c(3,1))
```

例2: *at risk* に関する表を泥臭く作成

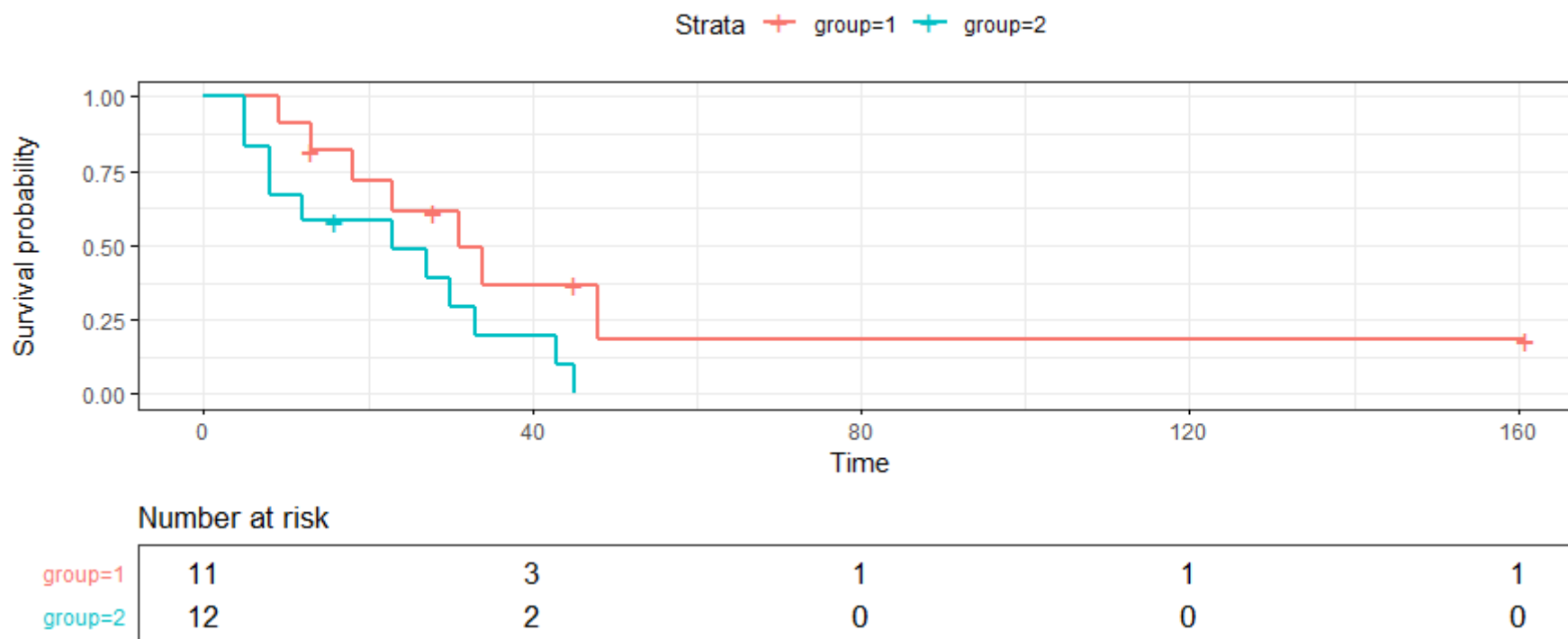


Time	0	20	40	60	80	100	120	140	160
Group=1	11	7	3	1	1	1	1	1	1
Group=2	12	6	2	0	0	0	0	0	0

例3: パッケージ「survminer」

- パッケージ「survminer」の関数 `ggsurvplot()` を使用すると、サクッと作成可
<https://rpkgs.datanovia.com/survminer/index.html>

```
> library(survminer)
> library(ggthemes)
> aml$group <- ifelse(aml$x=="Maintained", 1, 2)
> result <- survfit(Surv(time, status) ~ group, data = aml)
> ggsurvplot(result, data=aml, risk.table=TRUE,
+   ggtheme=theme_bw(), tables.theme=theme_cleantable() +
+   theme(panel.grid=element_blank()) )
```



メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

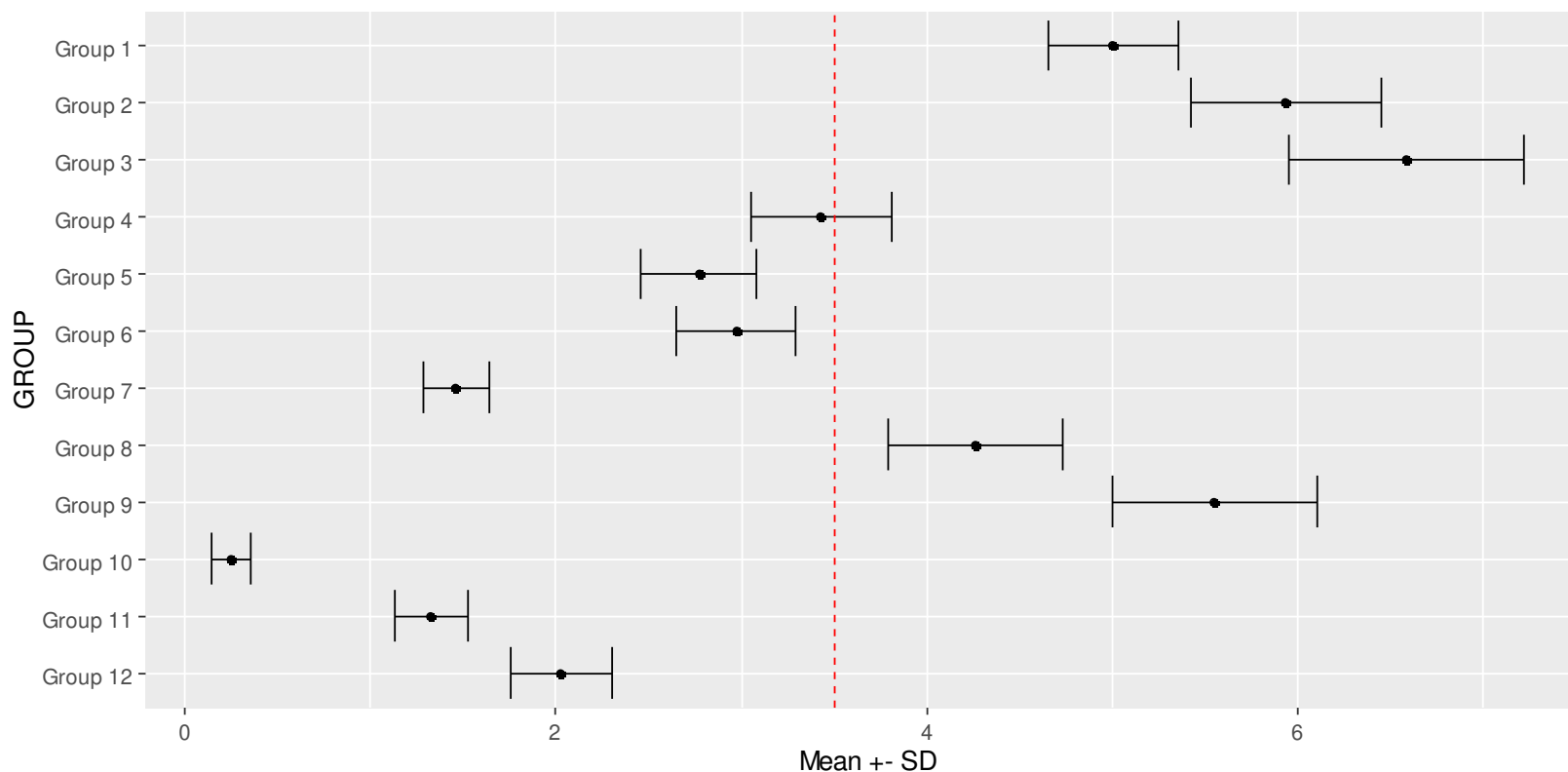
フォレスト・プロット (Forest Plot) 用データ生成

```
> MCI <- summarise(group_by(iris, Species),
+ M=mean(Sepal.Length), S=sd(Sepal.Length))
> MCI <- rbind(MCI, summarise(group_by(iris, Species),
+ M=mean(Sepal.Width ), S=sd(Sepal.Width )))
> MCI <- rbind(MCI, summarise(group_by(iris, Species),
+ M=mean(Petal.Length), S=sd(Petal.Length)))
> MCI <- rbind(MCI, summarise(group_by(iris, Species),
+ M=mean(Petal.Width ), S=sd(Petal.Width )))
> MCI$M <- round(MCI$M, 1)
> MCI$L <- round(MCI$M - MCI$S, 2)
> MCI$U <- round(MCI$M + MCI$S, 2)
> MCI$ID <- as.numeric(row.names(MCI))
> MCI$GROUP <- factor(MCI$ID, labels=paste("Group",1:12))
```

ID	GROUP	Species	M	S	L	U
1	Group 1	setosa	5.0	0.35	4.65	5.35
2	Group 2	versicolor	5.9	0.52	5.38	6.42
3	Group 3	virginica	6.6	0.64	5.96	7.24
4	Group 4	setosa	3.4	0.38	3.02	3.78
5	Group 5	versicolor	2.8	0.31	2.49	3.11
6	Group 6	virginica	3.0	0.32	2.68	3.32
7	Group 7	setosa	1.5	0.17	1.33	1.67
8	Group 8	versicolor	4.3	0.47	3.83	4.77
9	Group 9	virginica	5.6	0.55	5.05	6.15
10	Group 10	setosa	0.2	0.11	0.09	0.31
11	Group 11	versicolor	1.3	0.20	1.10	1.50
12	Group 12	virginica	2.0	0.28	1.73	2.27

例1: 単純なForest Plot

```
> ggplot(data=MCI, aes(x=M, y=GROUP, xmin=L, xmax=U)) +  
+   geom_errorbarh() + geom_point() + xlab("Mean +- SD") + ylab("Group") +  
+   geom_vline(xintercept=3.5, col="red", lty=2) +  
+   scale_y_discrete("GROUP", limits=rev(levels(MCI$GROUP)))
```

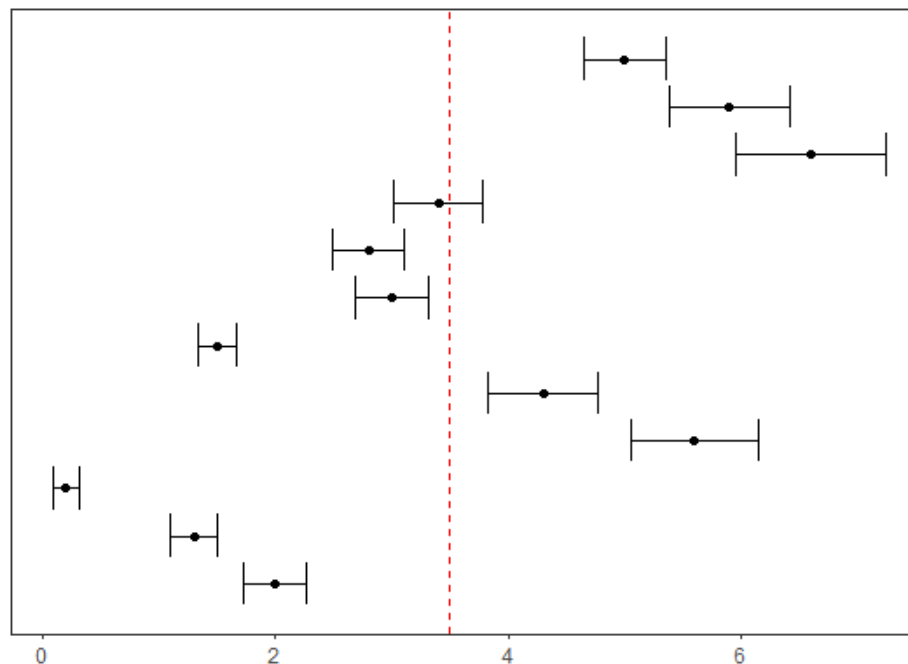


例2: Forest Plotの横につける表を泥臭く作成①

```
> library(grid); library(gridExtra)
> FP <- ggplot(data=MCI, aes(x=M, y=ID, xmin=L, xmax=U)) +
+   geom_errorbarh() + geom_point() + scale_y_reverse() + ggtitle("Sample Plot") +
+   geom_vline(xintercept=3.5, col="red", lty=2) + theme_bw() +
+   theme(axis.title.x=element_blank(), panel.grid=element_blank(),
+         axis.title.y=element_blank(), axis.text.y=element_blank(),
+         axis.line.y=element_blank(), axis.ticks.y=element_blank(),
+         plot.margin=margin(0.07, 0.05, 0.01, 0, "npc"))
> TB <- tableGrob(MCI[,c(7,2,4,5)], rows=NULL, cols=c("Group", "Mean", "Lower", "Upper"),
+   theme=ttheme_minimal())
> grid.arrange(TB, FP, nrow=1, as.table=TRUE, widths=c(1,2))
```

Group	Mean	Lower	Upper
Group 1	5.0	4.65	5.35
Group 2	5.9	5.38	6.42
Group 3	6.6	5.96	7.24
Group 4	3.4	3.02	3.78
Group 5	2.8	2.49	3.11
Group 6	3.0	2.68	3.32
Group 7	1.5	1.33	1.67
Group 8	4.3	3.83	4.77
Group 9	5.6	5.05	6.15
Group 10	0.2	0.09	0.31
Group 11	1.3	1.10	1.50
Group 12	2.0	1.73	2.27

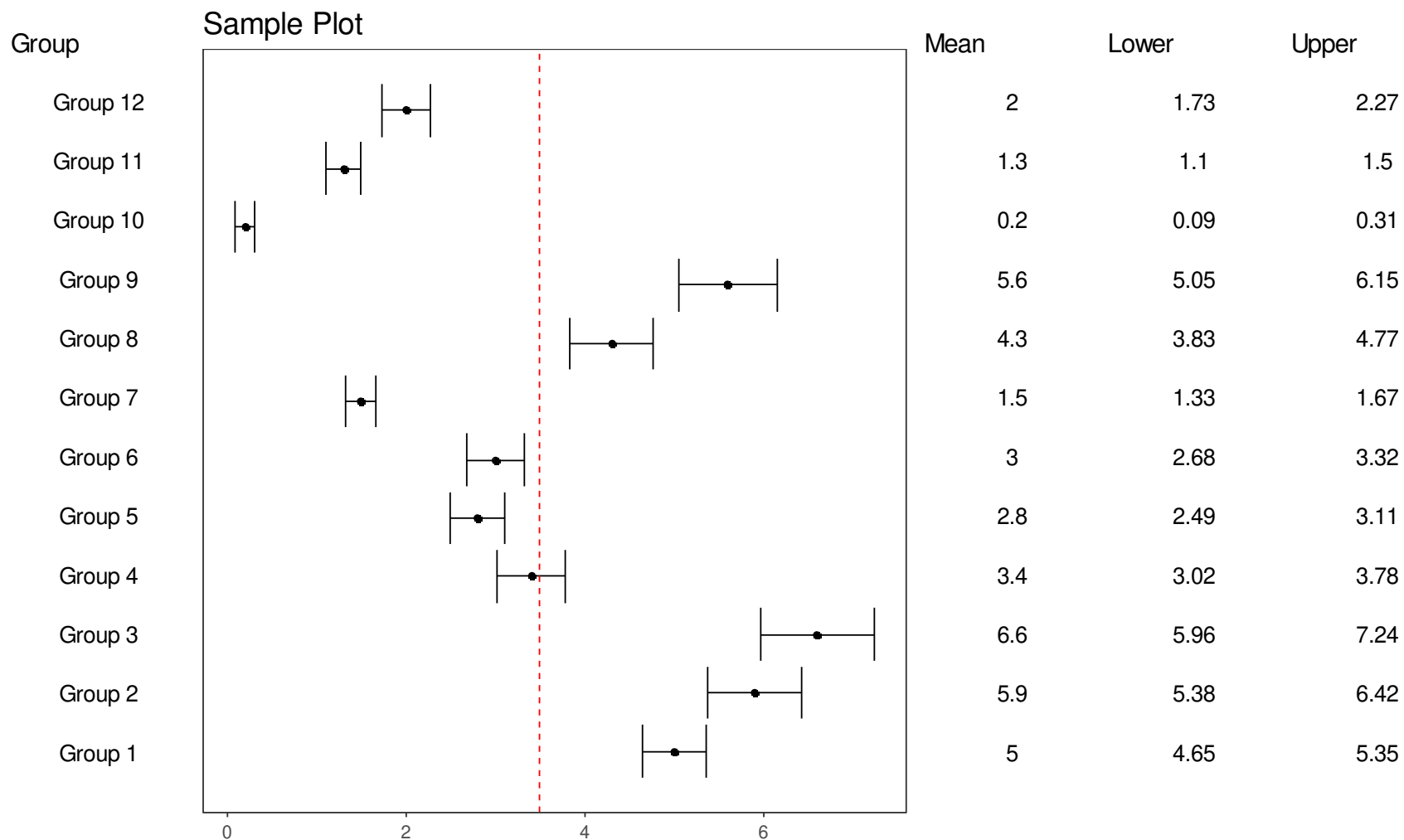
Sample Plot



例3: Forest Plotの横につける表を泥臭く作成②

```
> FP <- ggplot(data=MCI, aes(x=M, y=ID, xmin=L, xmax=U)) +  
+   geom_errorbarh() + geom_point() + ggtitle("Sample Plot") +  
+   geom_vline(xintercept=3.5, col="red", lty=2) + theme_bw() +  
+   theme(axis.title.x=element_blank(), panel.grid=element_blank(),  
+         axis.title.y=element_blank(), axis.text.y=element_blank(),  
+         axis.line.y=element_blank(), axis.ticks.y=element_blank(),  
+         plot.title=element_text(size=15))  
  
> TB <- ggplot(data=MCI, aes(y=ID)) +  
+   xlab("") +  
+   theme(axis.title.y=element_blank(), axis.text=element_blank(),  
+         axis.line=element_blank(), axis.ticks=element_blank(),  
+         panel.background=element_blank(), panel.border=element_blank(),  
+         panel.grid.major=element_blank(), panel.grid.minor=element_blank(),  
+         legend.position="none", plot.background=element_blank(),  
+         plot.margin=margin(0.04, 0.01, 0.04, 0.01, "npc"))  
  
> TB1 <- TB + geom_text(aes(x=1, label=GROUP)) + ggtitle("Group")  
> TB2 <- TB + geom_text(aes(x=1, label=M)) + ggtitle("Mean")  
> TB3 <- TB + geom_text(aes(x=1, label=L)) + ggtitle("Lower")  
> TB4 <- TB + geom_text(aes(x=1, label=U)) + ggtitle("Upper")  
  
> LM <- matrix(c(1,2,2,2,2,3,4,5), nrow=1)  
> grid.arrange(TB1, FP, TB2, TB3, TB4, layout_matrix=LM)
```


例3: Forest Plotの横につける表を泥臭く作成②



例4: Forest Plotの横につける表を泥臭く作成③

```
> FP <- ggplot(data=MCI, aes(x=M, y=ID, xmin=L, xmax=U)) +
+   geom_errorbarh() + geom_point() + ggtitle("") + scale_y_reverse() +
+   geom_vline(xintercept=3.5, col="red", lty=2) + theme_bw() +
+   theme(axis.title.x=element_blank(), panel.grid=element_blank(),
+         axis.title.y=element_blank(), axis.text.y=element_blank(),
+         axis.line.y=element_blank(), axis.ticks.y=element_blank(),
+         plot.title=element_text(size=8))

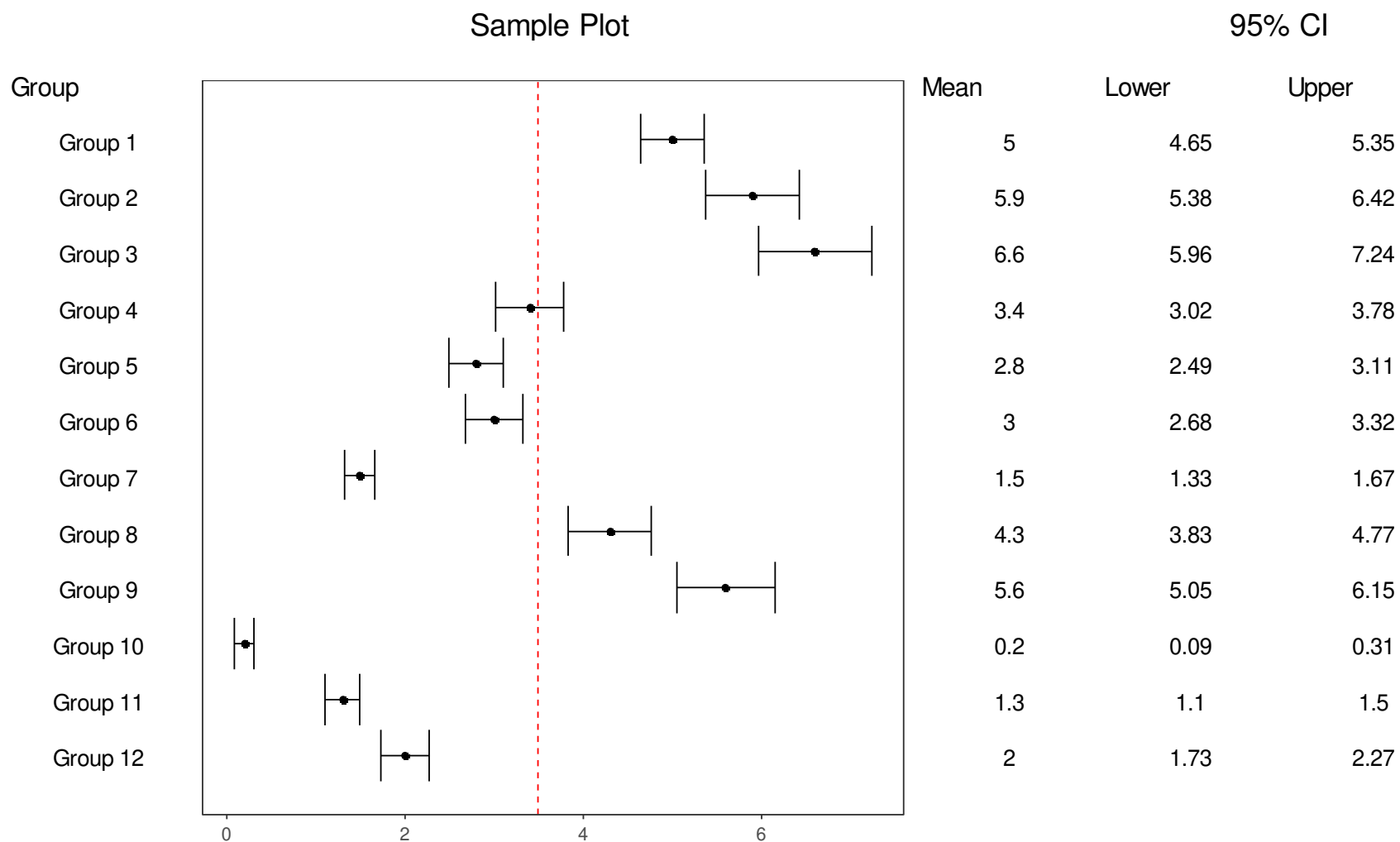
> TB <- ggplot(data=MCI, aes(y=ID)) +
+   xlab("") +
+   theme(axis.title.y=element_blank(), axis.text=element_blank(),
+         axis.line=element_blank(), axis.ticks=element_blank(),
+         panel.background=element_blank(), panel.border=element_blank(),
+         panel.grid.major=element_blank(), panel.grid.minor=element_blank(),
+         legend.position="none", plot.background=element_blank(),
+         plot.margin=margin(0.04, 0.01, 0.04, 0.01, "npc"))

> HD1 <- tableGrob(data.frame(x="Sample Plot"), rows=NULL, cols=NULL,
+   theme=ttheme_minimal(base_size=15, base_color="black"))
> HD2 <- tableGrob(data.frame(x="95% CI"), rows=NULL, cols=NULL,
+   theme=ttheme_minimal(base_size=15, base_color="black"))

> TB1 <- TB + geom_text(aes(x=1, label=GROUP)) + ggtitle("Group") + scale_y_reverse()
> TB2 <- TB + geom_text(aes(x=1, label=M)) + ggtitle("Mean") + scale_y_reverse()
> TB3 <- TB + geom_text(aes(x=1, label=L)) + ggtitle("Lower") + scale_y_reverse()
> TB4 <- TB + geom_text(aes(x=1, label=U)) + ggtitle("Upper") + scale_y_reverse()

> LM <- matrix(c(1,1,1,1,1,1,2,2,
+               3,4,4,4,4,5,6,7), nrow=2, byrow=T)
> grid.arrange(HD1,HD2,TB1, FP, TB2, TB3, TB4, layout_matrix=LM, heights=c(1,20))
```

例4: Forest Plotの横につける表を泥臭く作成③

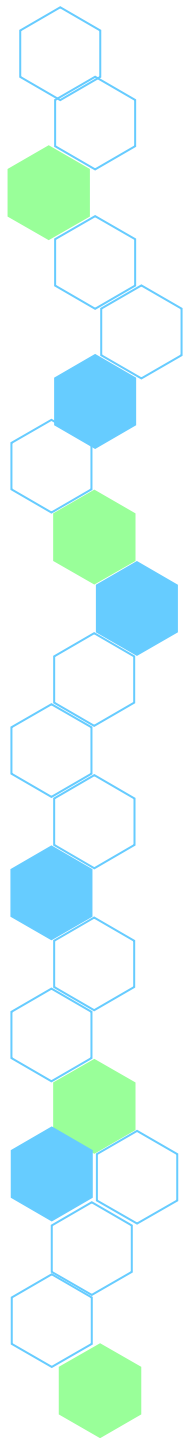


メニュー

- 基本的事項
- グラフの作成の流れ
- 種々のグラフ作成例
- グラフのカスタマイズ
- 関数 `theme()` の補足
- 応用例 : Kaplan Meier Plot
- 応用例 : Forest Plot

参考文献

- Hadley Wickham 著、石田 基広 他翻訳(2012)
「グラフィックスのための R プログラミング(丸善出版)」
- Winston Chang 著、石井 弓美子 他翻訳(2013) → 改訂版もあり
「R グラフィックスクックブック(オライリージャパン)」
- ggplot2 のマニュアル(CRAN、tidyverse 内のリファレンス)
<https://cran.r-project.org/package=ggplot2>
<https://ggplot2.tidyverse.org/reference/>
- “R Graphics Cookbook, 2nd edition” created by Winston Chang
<https://r-graphics.org/>
- “Displaying tables as grid graphics” by Baptiste Auguie
<https://cran.r-project.org/web/packages/gridExtra/vignettes/tableGrob.html>
- “Arranging multiple grobs on a page” by Baptiste Auguie
<https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html>
- survminer のマニュアル
<https://rpkgs.datanovia.com/survminer/index.html>



– End of File –

