2022年9月22日@京都大学

KUDH Basics
統計ソフトウェア「R」ワークショップ

2日目

（後半）ggplot2による詳細な作図 + Rで仮説検定

講師：山崎 大暉

（立命館大学・日本学術振興会）

前半：dplyrでデータフレームを操作

紹介したdplyrの関数

- 行の一部を取り出す　filter()関数
- 列の一部を取り出す　select()関数
- 新しい列を追加する　mutate()関数
- データをグループ化する　group_by()関数
- データを要約する　summarise()関数

関数の合わせ技

- group_by() + summarise()
- group_by() + mutate()
- mutate() + if_else()
- etc.

その他Tips

- 行を並び替える　arrange()関数
- 列名を変える　rename()関数
- 欠損値を除外　filter(df, !is.na(列名))
- パイプ演算子　%>%

ggplot2パッケージを使って きれいなグラフを作ろう

グラフを描こう

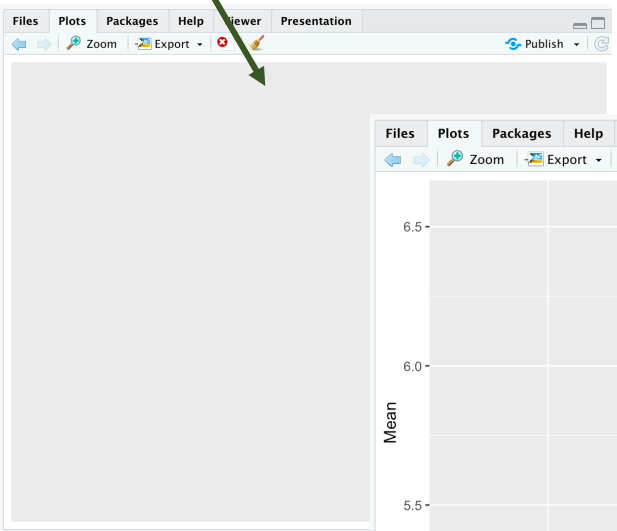
```
ggplot()
```

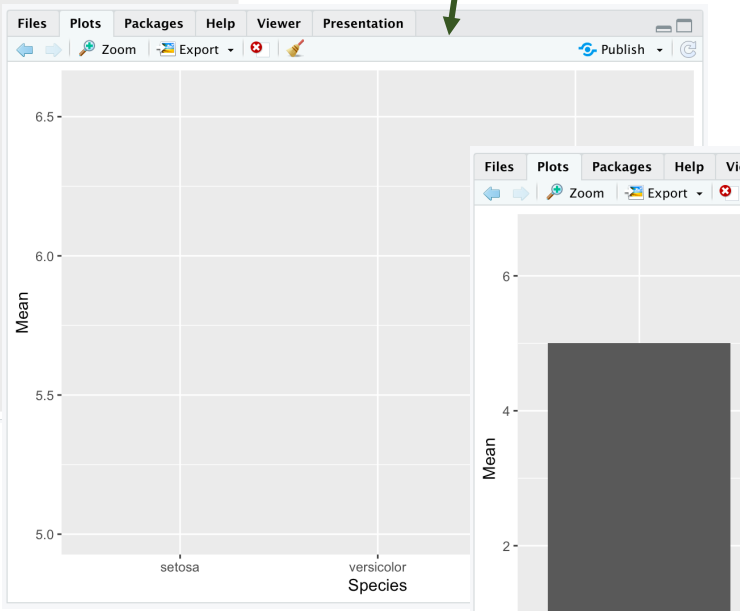
```
ggplot(aes(x = Species, y = Mean))
```

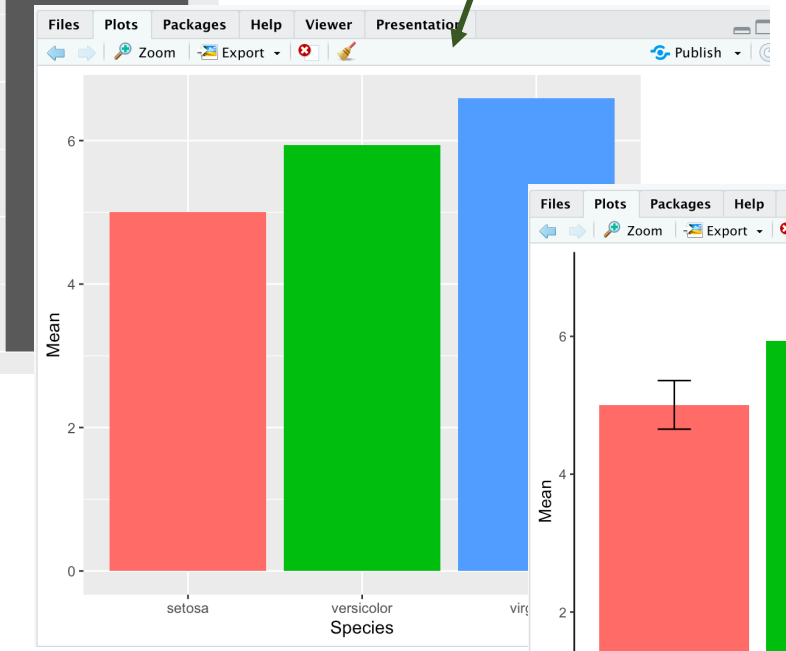
```
ggplot(aes(x = Species, y = Mean))+
geom_bar(stat="identity")
```

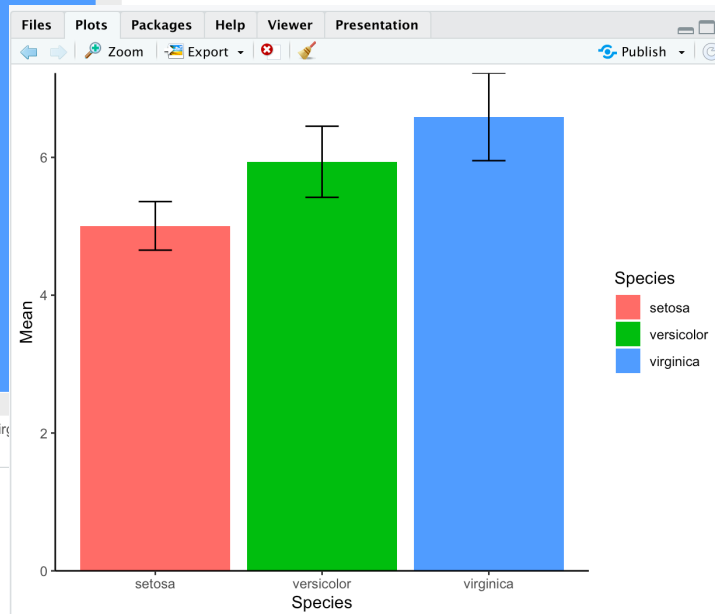
```
ggplot(aes(x = Species, y = Mean, fill=Species))+
geom_bar(stat="identity")+
```

```
ggplot(aes(x = Species, y = Mean, fill=Species))+
geom_bar(stat="identity")+
geom_errorbar(aes(ymax = Mean+SD, ymin = Mean-SD, width = 0.2))+
theme_classic()+
scale_y_continuous(expand = c(0,0))
```









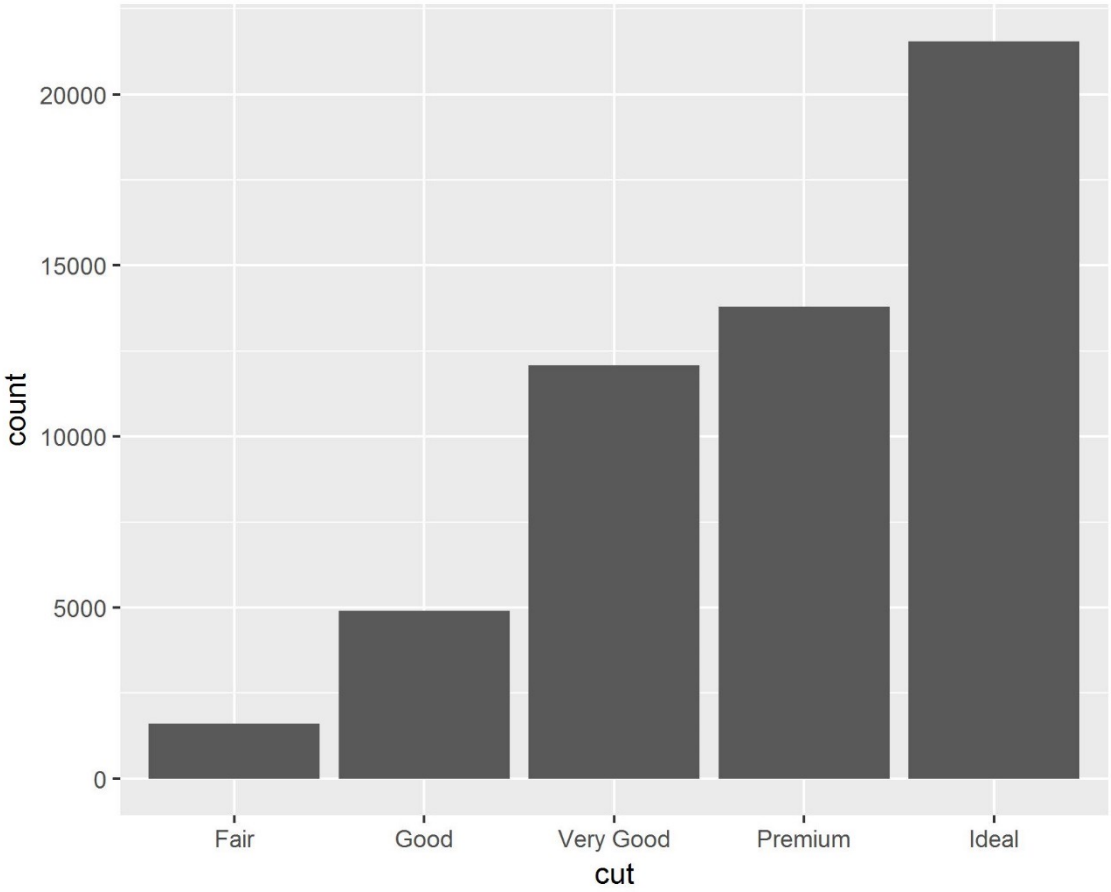
棒グラフ(度数分布)

- ggplot()関数+geom_bar()関数

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

審美的属性

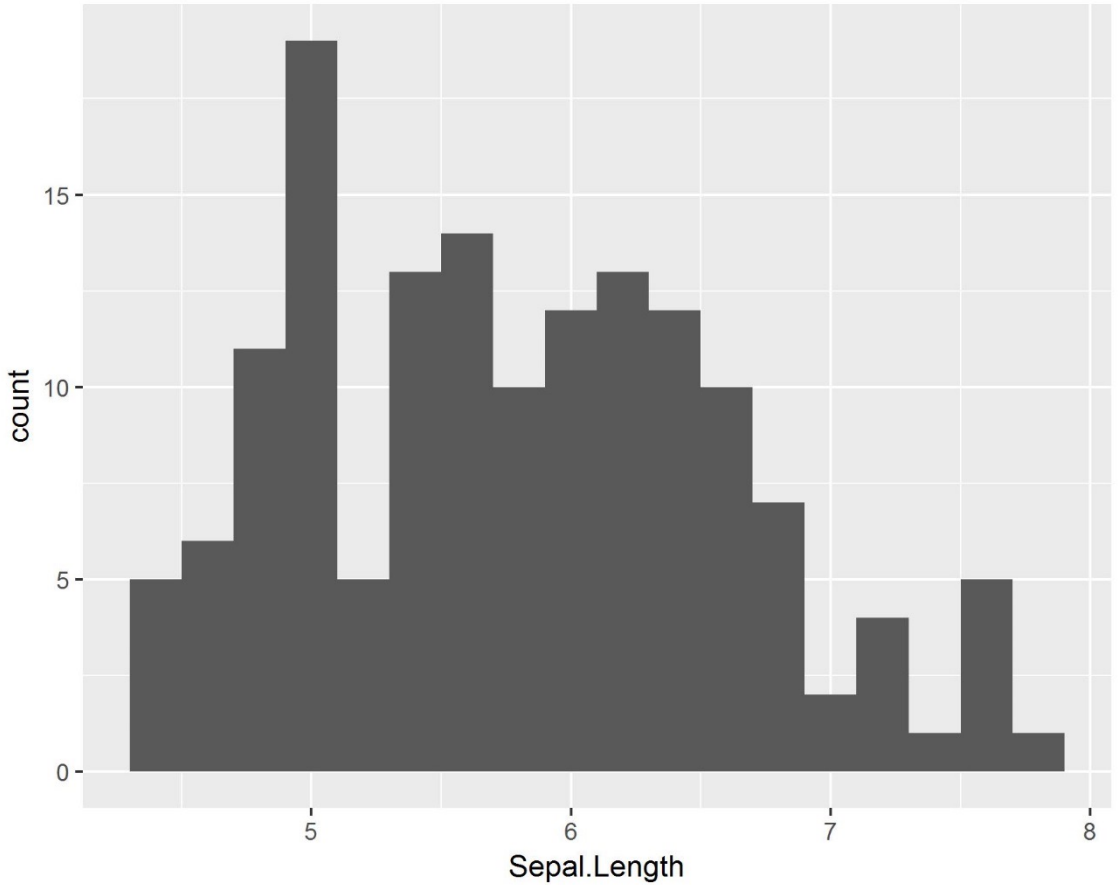
- alpha (透明度；0–1)
- color (棒の境界線の色)
- fill　(面の色)



ヒストグラム

- ggplot()関数+geom_histogram()関数

```
iris %>%
  ggplot(aes(x = Sepal.Length))+
  geom_histogram(binwidth = 0.2)
```



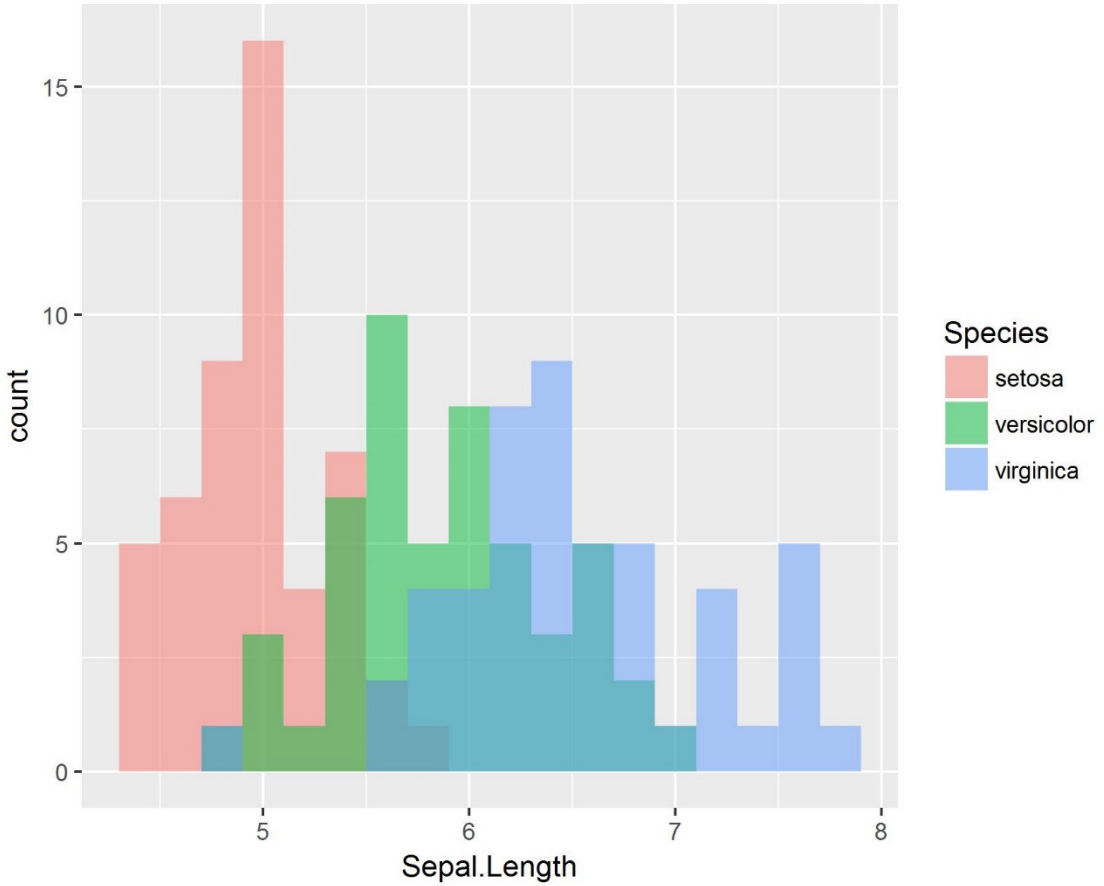
・binwidthで階級幅を指定
　ー階級幅によって印象が変わるので注意

審美的属性
　ーalpha (透明度；0–1)
　ーcolor (棒の境界線の色)
　ーfill　　(面の色)

ヒストグラム(層別)

・ggplot()関数+geom_histogram(position = “identity”)関数

```
iris %>%
  ggplot(aes(x = Sepal.Length, fill = Species))+
  geom_histogram(binwidth = 0.2,
                 position = "identity",
                 alpha = 0.5)
```



審美的属性

－alpha (透明度；0–1)

－color (棒の境界線の色)

－fill　(面の色)

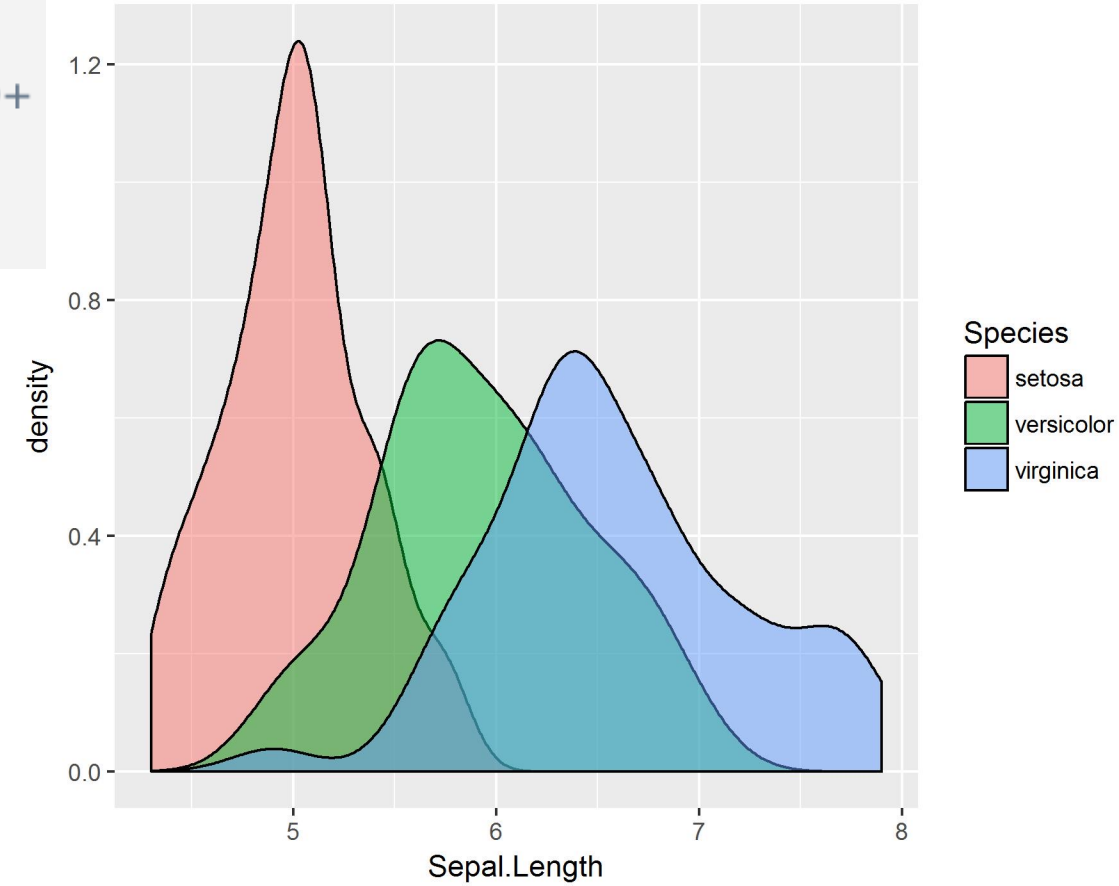
密度プロット

- ggplot()関数+geom_density()関数

```
iris %>%
  ggplot(aes(x = Sepal.Length, fill = Species))+
  geom_density(position = "identity",
               alpha = 0.5)
```

審美的属性

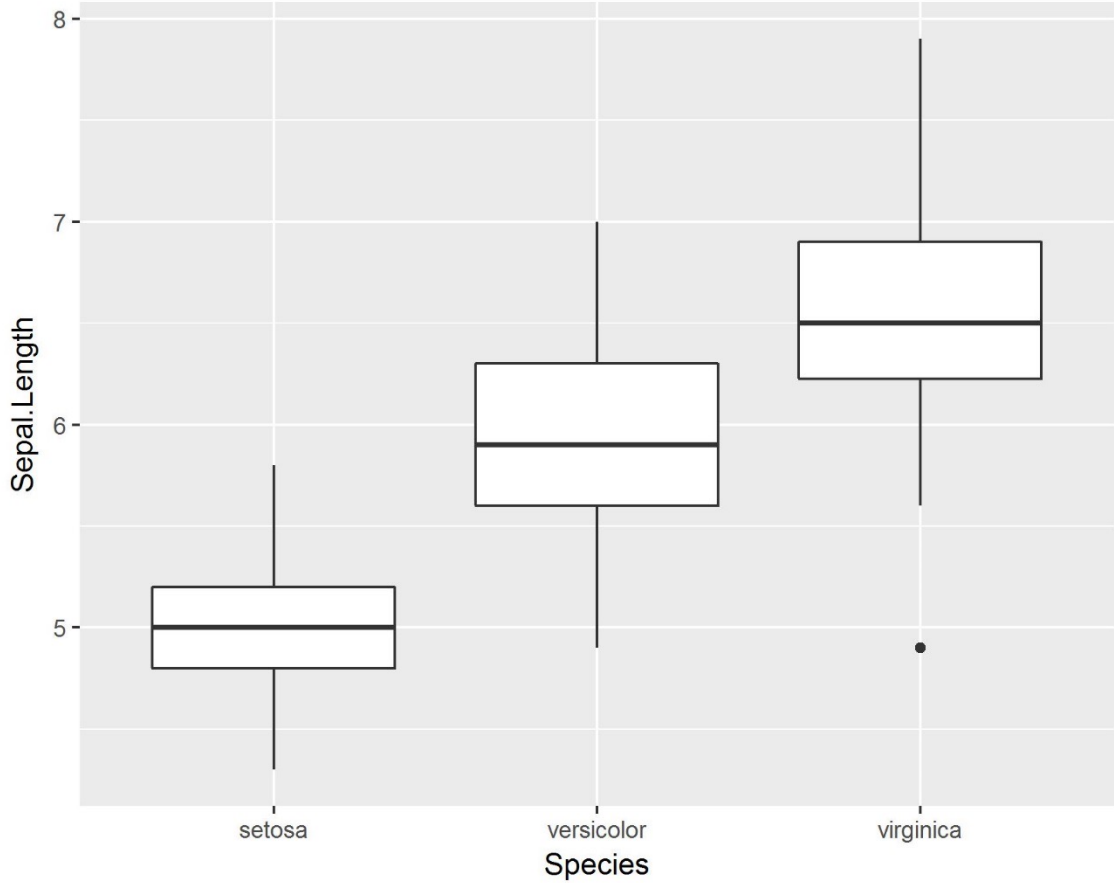
- alpha (内側の透明度；0–1)
- color (境界線の色)
- fill　(面の色)



箱ひげ図

- ggplot()関数+geom_boxplot()関数

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



審美的属性

- alpha (箱の内側の透明度；0–1)
- color (箱の境界線とヒゲの色)
- fill　　(箱面の色)

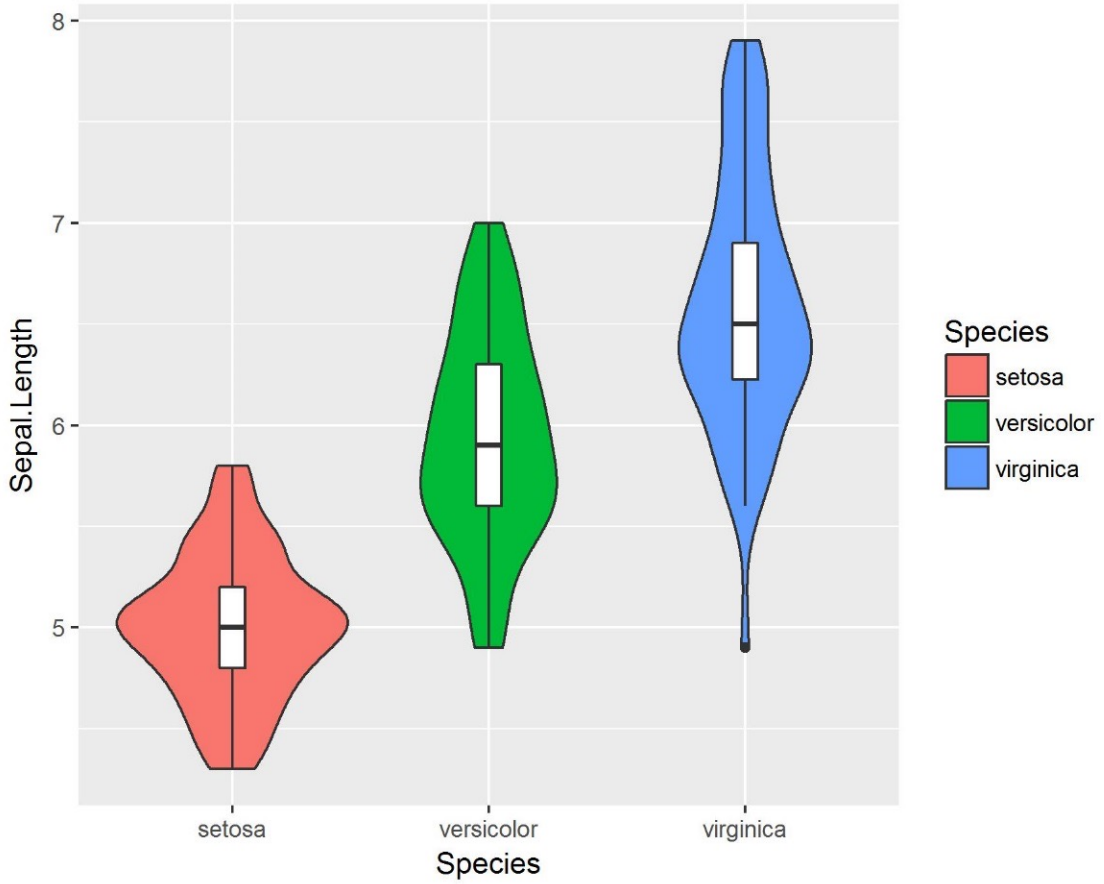
バイオリンプロット

- ggplot()関数+geom_violin()関数+geom_boxplot()関数

```
iris %>%
  ggplot(aes(x = Species, y = Sepal.Length,
             fill = Species))+
  geom_violin()+
  geom_boxplot(width = 0.1, fill = "white")
```

審美的属性 (geom_violin()内)

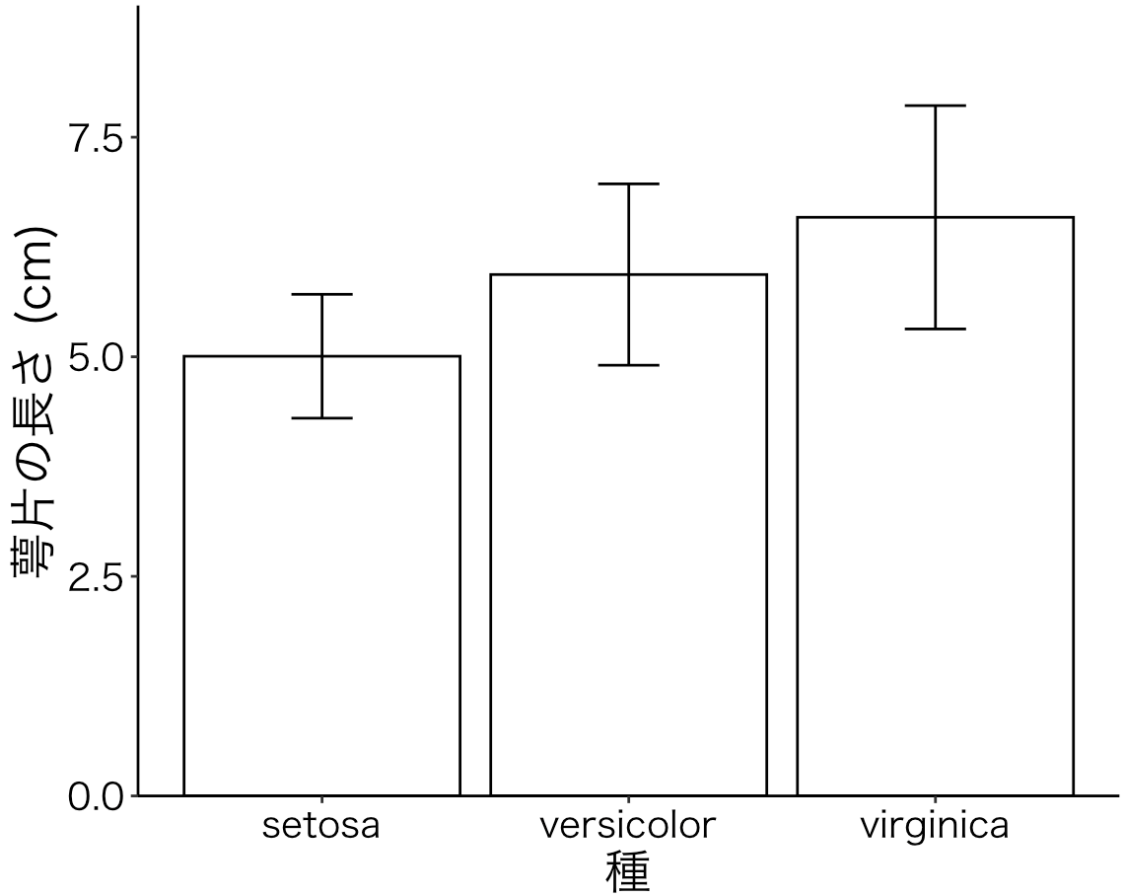
- alpha (内側の透明度；0–1)
- color (境界線の色)
- fill　(面の色)



irisを使ってエラーバー付きの棒グラフを描こう

こんな感じのグラフにしたい

- SpeciesごとにSepal.Lengthの平均値±SDを示す棒グラフ
- シンプルな白黒の見た目
- 軸の名前を自由に付けたい
- 文字のフォントや大きさを調整したい



まず，ggplot()関数に渡すデータフレームを準備する

- Speciesでグループ化し，平均値と標準偏差を出しておけばよい

```
iris %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length),
            SD = sd(Sepal.Length)) %>%
```

パイプ演算子でパイプ処理！

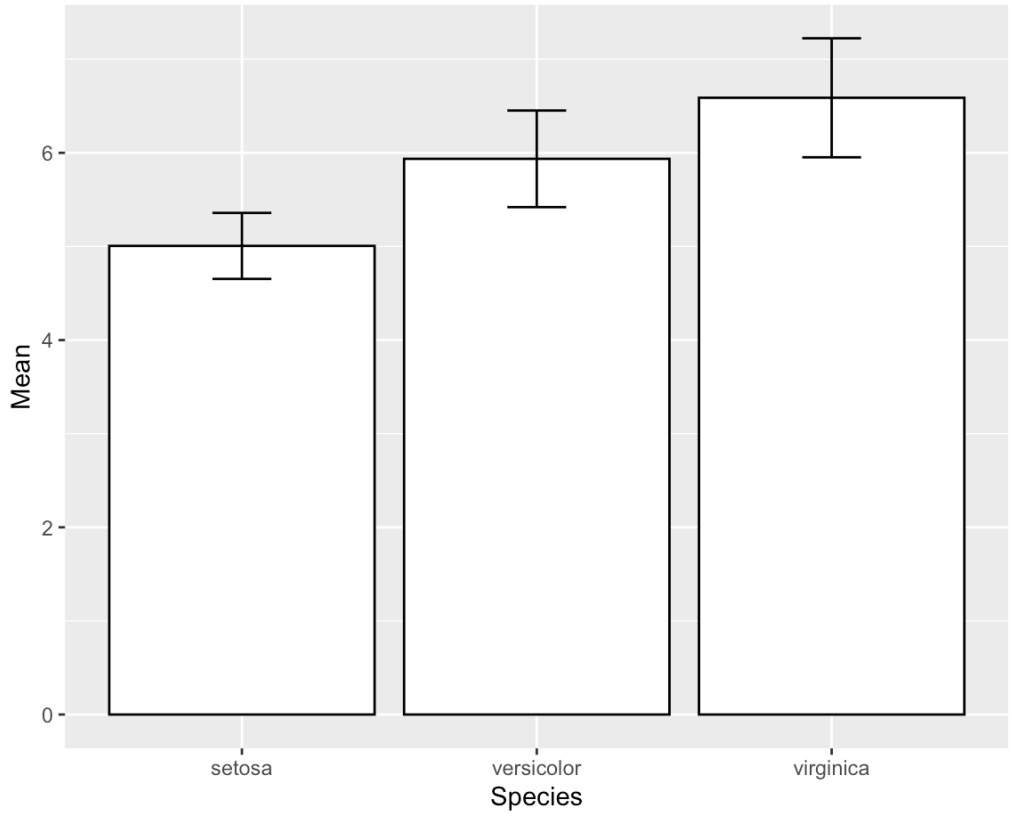
こんな感じのデータフレームになっているはず（この時点で確認してもOK）

```
# A tibble: 3 × 3
  Species      Mean    SD
  <fct>       <dbl> <dbl>
1 setosa       5.01 0.352
2 versicolor   5.94 0.516
3 virginica    6.59 0.636
```

このデータフレームをggplot()関数に渡す

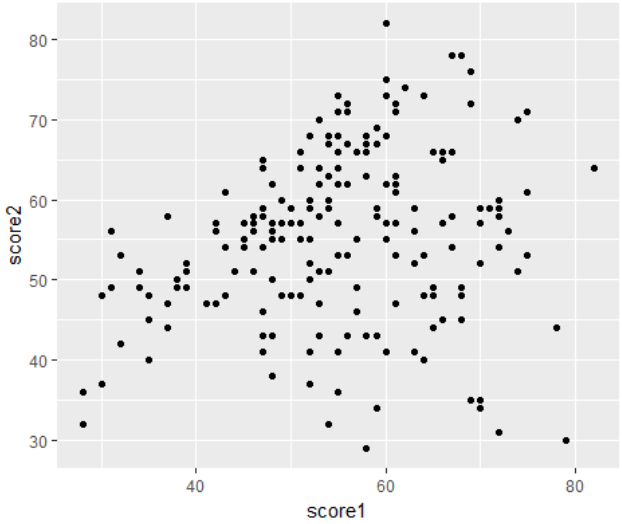
```
iris %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length),
            SD = sd(Sepal.Length)) %>%

  ggplot(aes(x = Species, y = Mean))+
  geom_bar(stat="identity", col = "black",fill="white")+
  geom_errorbar(aes(ymax = Mean+SD, ymin = Mean-SD, width
```

ggplot(aes(縦横軸の指定))+
geom_bar(位置，色，塗りつぶしの指定)+
geom_errorbar(上下端の値，幅の指定)

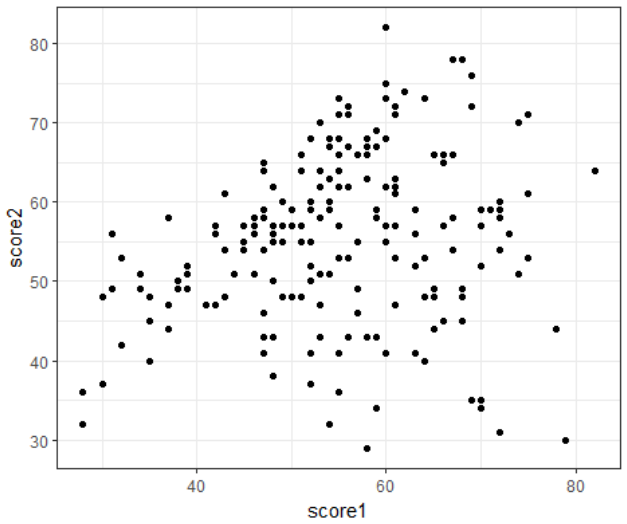


図のテーマを決める：theme_*()関数

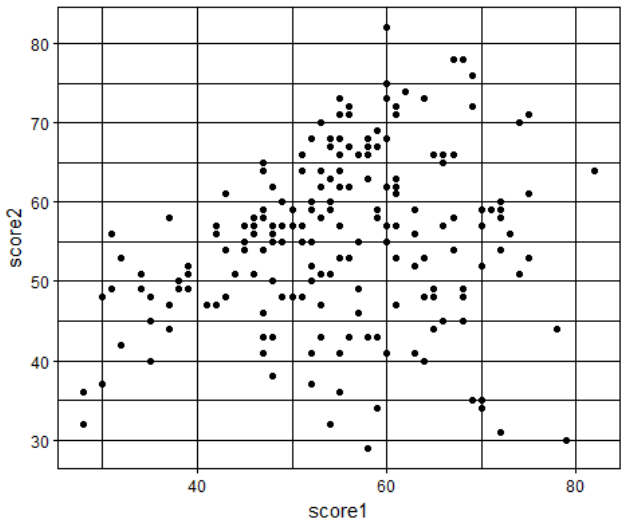
default



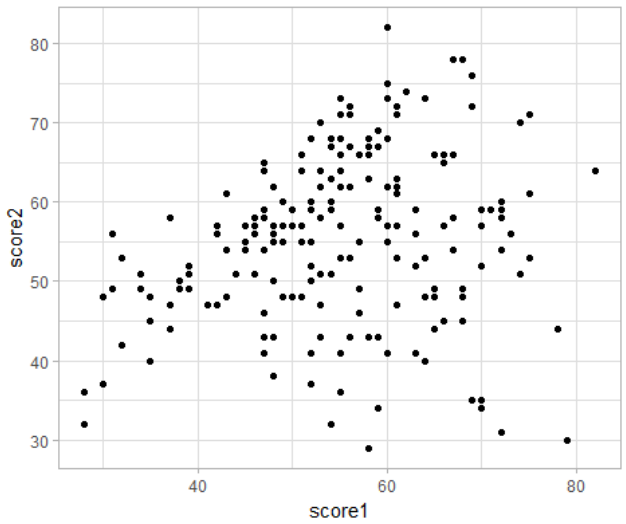
theme_bw()



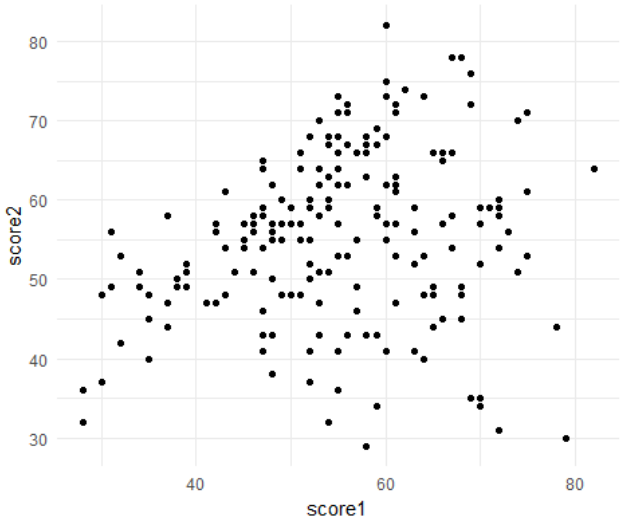
theme_linedraw()



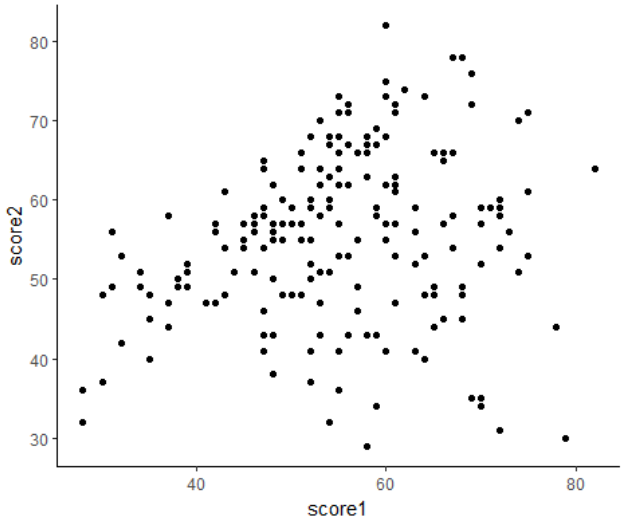
theme_light()



theme_minimal()



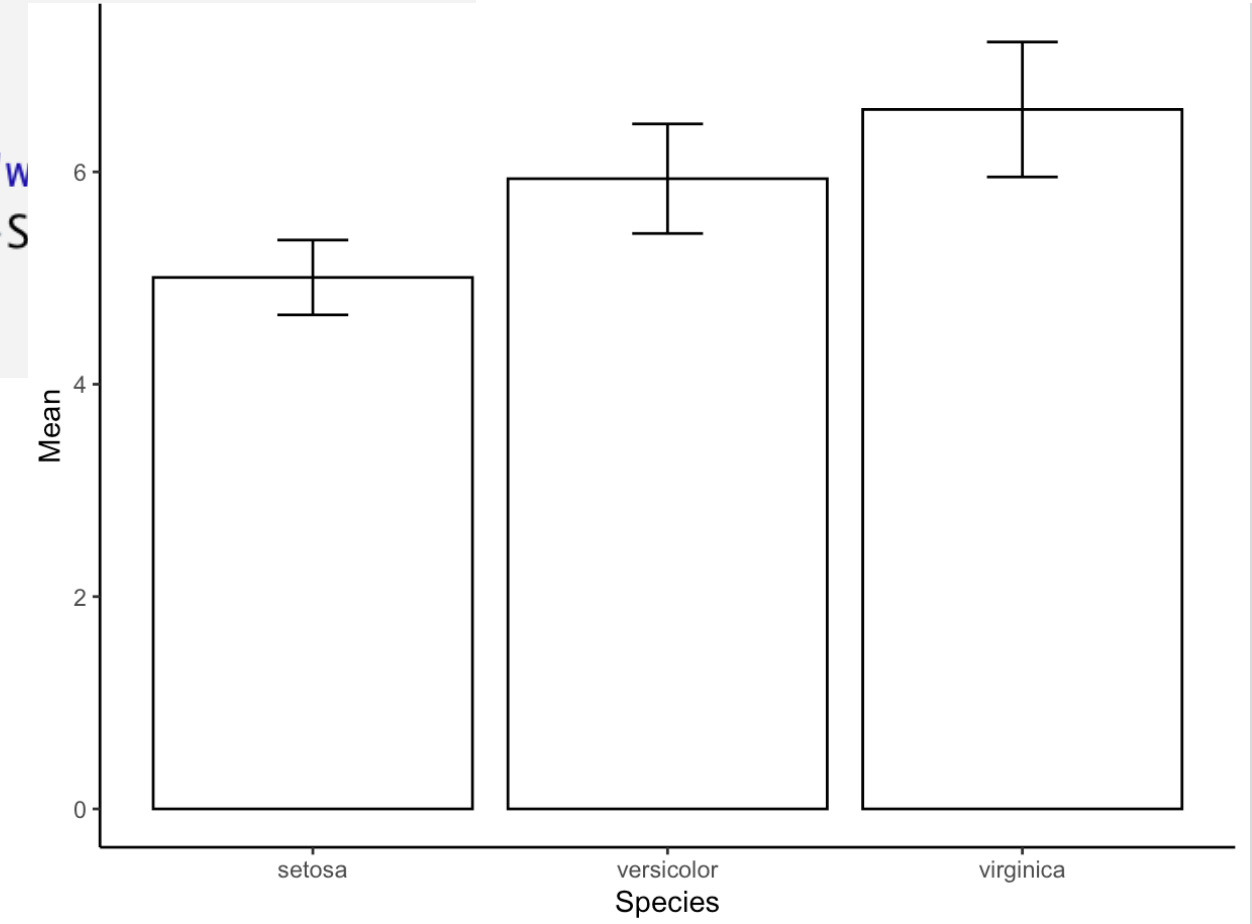
theme_classic()



theme_void()

図のテーマを決める：theme_*()関数

```
iris %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length),
            SD = sd(Sepal.Length)) %>%

  ggplot(aes(x = Species, y = Mean))+
  geom_bar(stat="identity", col = "black",fill="w
  geom_errorbar(aes(ymax = Mean+SD, ymin = Mean-S

  theme_classic()+
```



theme_classic()にしてみました

軸の名前や範囲を変える：scale_x_*(), scale_y_*()関数

```
iris %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length),
            SD = sd(Sepal.Length)) %>%

  ggplot(aes(x = Species, y = Mean))+
  geom_bar(stat="identity", col = "black",fill="white")+
  geom_errorbar(aes(ymax = Mean+SD, ymin = Mean-SD, width = 0.2))+

  theme_classic()+
  scale_x_discrete(name = "種")+
  scale_y_continuous(expand = c(0,0),limits = c(0,9), name = "萼片の長さ (cm)")
```

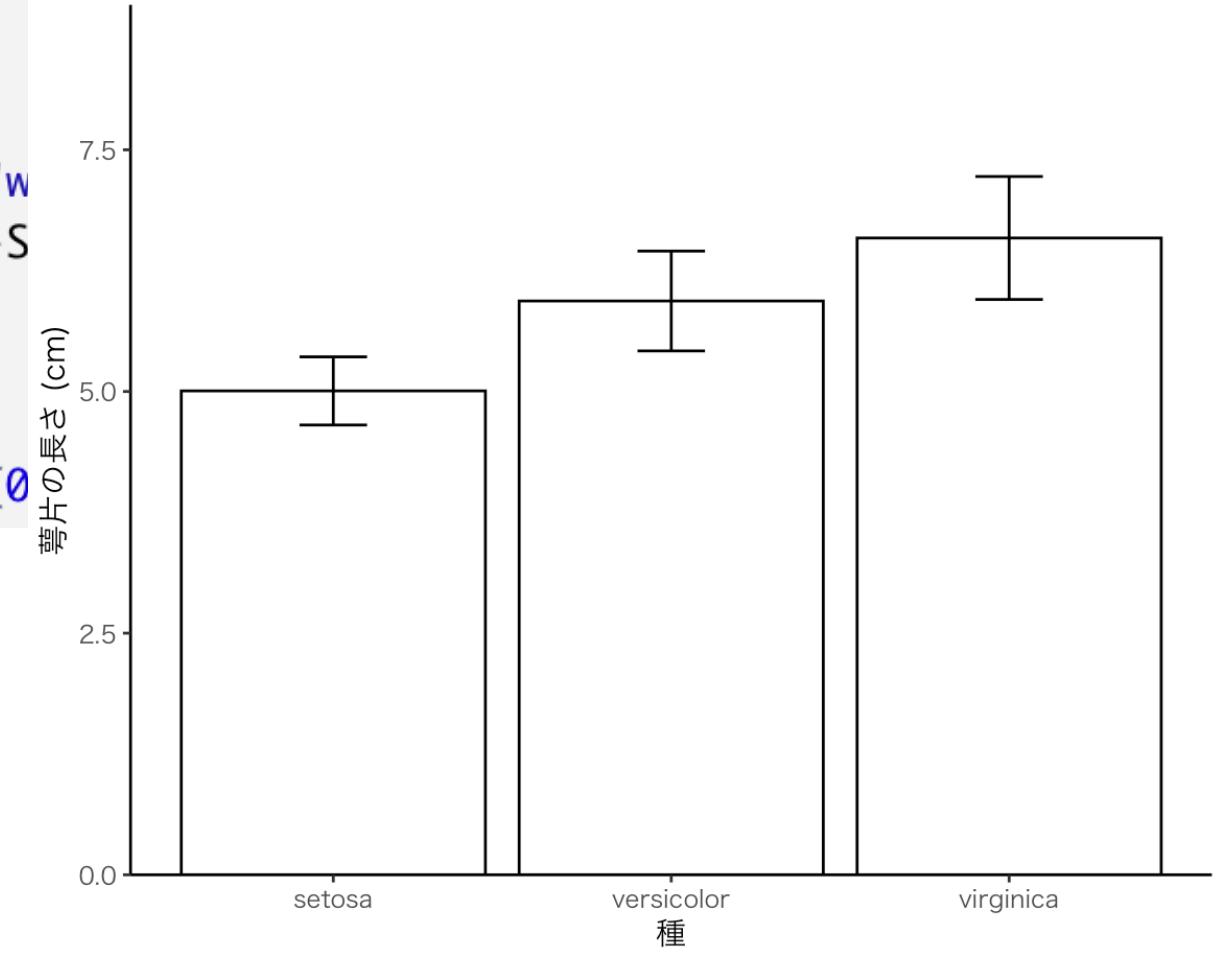
※xlim()関数，ylim()関数でも

scale_x_discrete()：離散値

scale_y_continuous()：連続値

軸の名前や範囲を変える：scale_x_*(), scale_y_*()関数

```
iris %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length),
            SD = sd(Sepal.Length)) %>%

  ggplot(aes(x = Species, y = Mean))+
  geom_bar(stat="identity", col = "black",fill="w
  geom_errorbar(aes(ymax = Mean+SD, ymin = Mean-S

  theme_classic()+
  scale_x_discrete(name = "種")+
  scale_y_continuous(expand = c(0,0),limits = c(0
```



scale_y_continuous()内

- name = “”：軸の名前
- expand = c()：軸からの距離
- limits = c()：軸の範囲

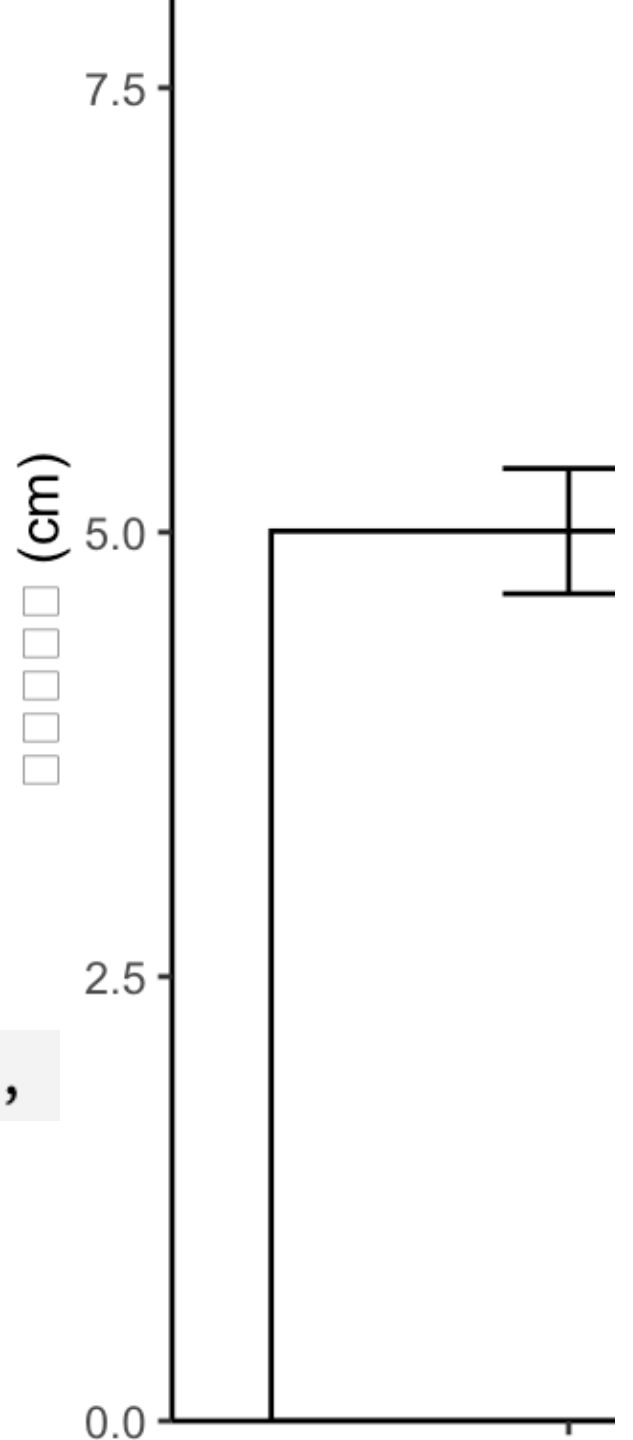
Macは日本語フォントが豆腐（□）になる

- theme_*()関数などで使えるフォントを指定する
  - ヒラギノ角ゴ Proがおすすめ

```
theme_classic(base_family = "HiraKakuPro-W3")+
```

- この後扱うtheme()関数内でも指定可能

```
theme(axis.title = element_text(family = "HiraKakuPro-W3",
```

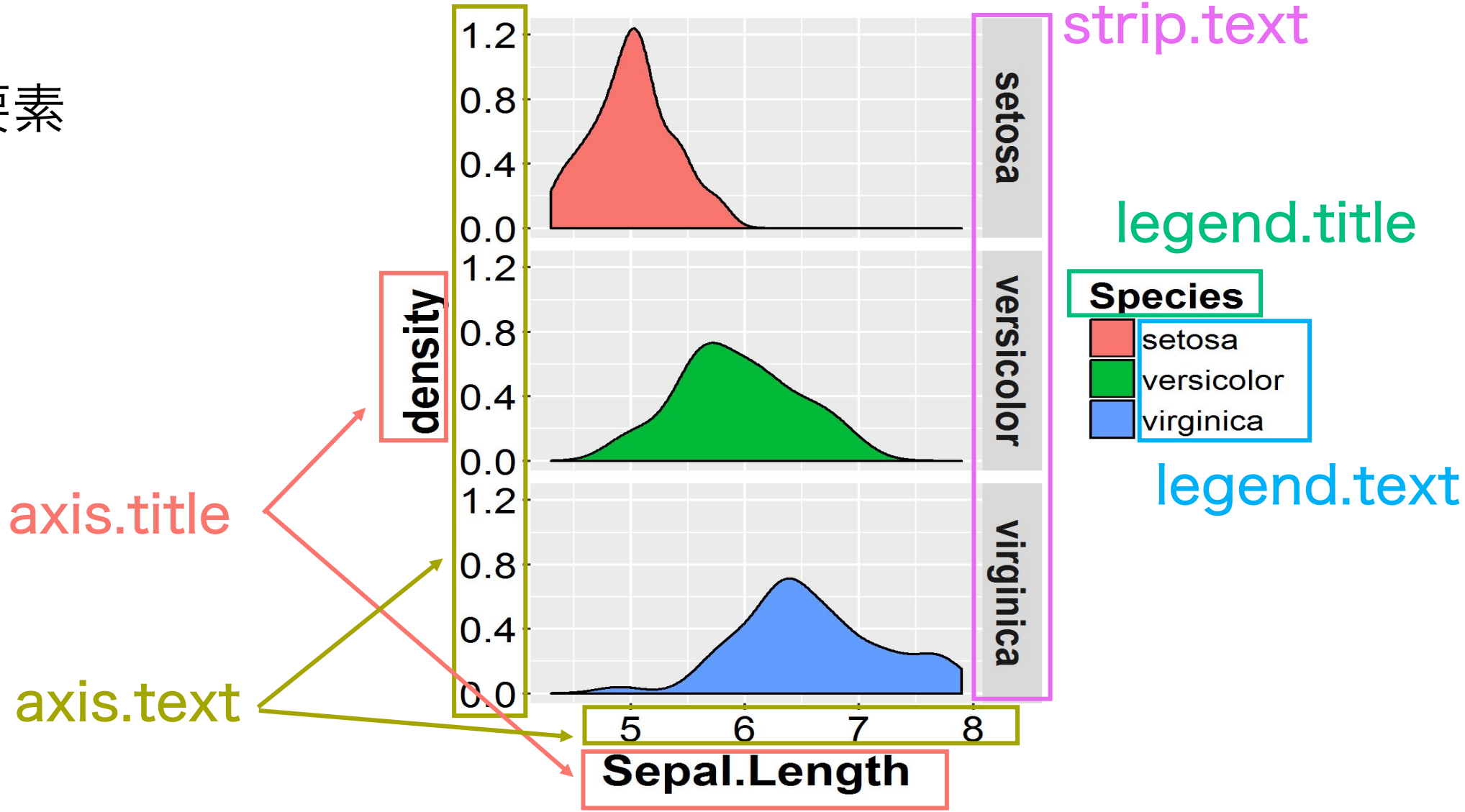


軸や凡例の細かい設定をする：theme()関数

テーマ要素	意味
axis.title	両軸のタイトル
axis.title.x	x軸のタイトル
axis.title.y	y軸のタイトル
axis.text	両軸の目盛
axis.text.x	x軸の目盛
axis.text.y	y軸の目盛
legend.title	凡例タイトル
legend.text	凡例項目
strip.text	サブプロットの項目

軸や凡例の細かい設定をする：theme()関数

- テーマ要素



軸や凡例の細かい設定をする：theme()関数

```
theme(axis.title = element_text(size = 18, face = "bold"),
      axis.text = element_text(size =  14, color = "black")
)
```

element_*()関数

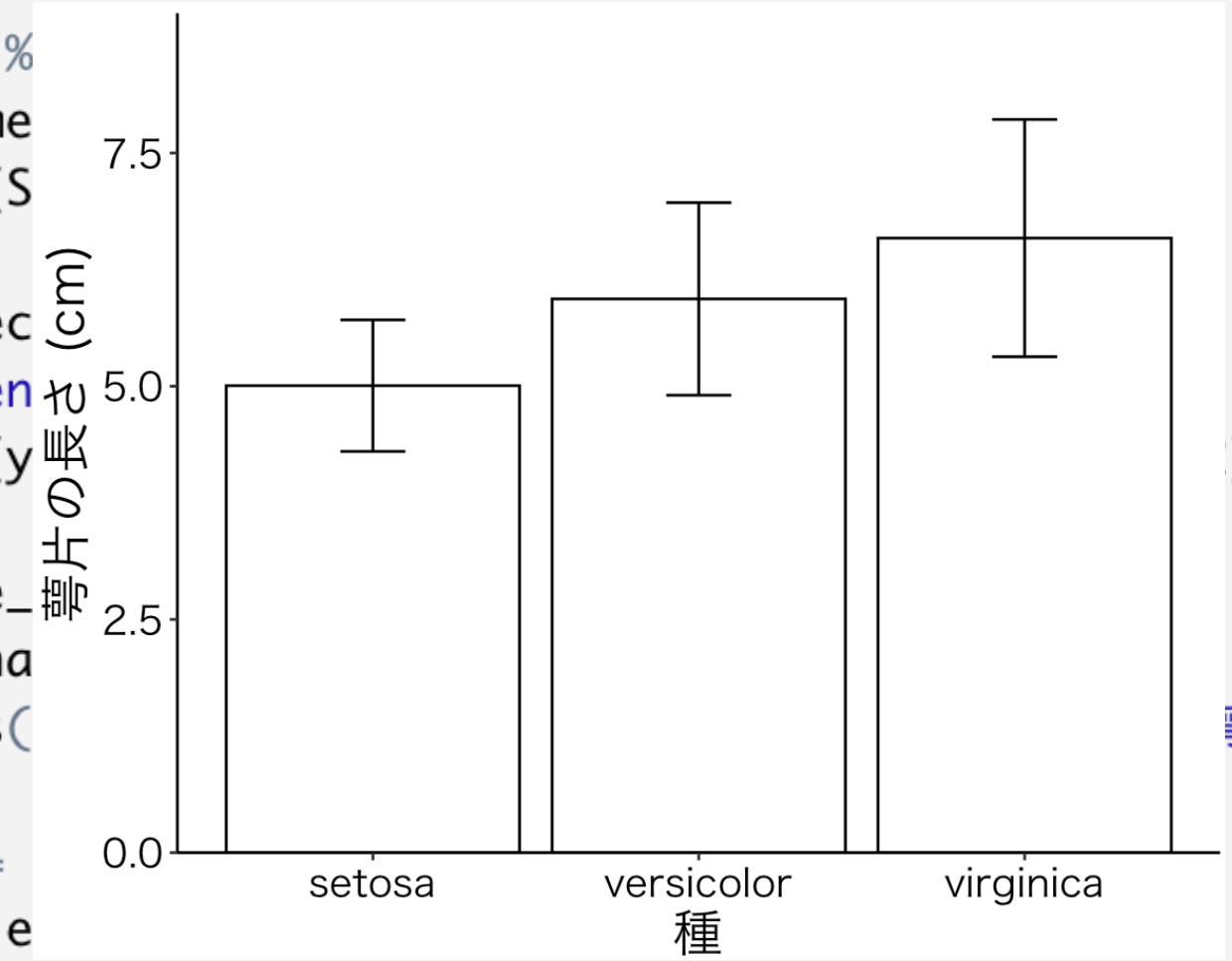
- **element_blank()：要素を消す**
- **element_text()：テキストのコントロール**
- element_rect()：背景や凡例など四角い領域をコントロール
- element_line()：線のコントロール

element_text()関数で設定できる項目

テーマ要素	説明
family	フォント
face	plain, bold, italic, bold.italic
colour/color	色
size	フォントサイズ(ポイント数)
hjust	水平方向の表示位置：0＝左、0.5＝中央、1＝右
vjust	垂直方向の表示位置：0＝左、0.5＝中央、1＝右
angle	角度

軸や凡例の細かい設定をする：theme()関数

```
iris %>%
  group_by(Species) %
  summarise(Mean = me
            SD = sd(S

  ggplot(aes(x = Spec
  geom_bar(stat="iden
  geom_errorbar(aes(y                                    ))+

  theme_classic(base_
  scale_x_discrete(na
  scale_y_continuous(                                    萼片の長さ（cm)")+

  theme(axis.title =
        axis.text = e
  )
```



背景テーマを先にセットする

- theme_set()関数
  - 設定したいtheme_*()やtheme()を入れて，先に実行しておく
  - 以降全てのggplot()に適用

```
theme_set(
  theme_classic(base_family = "HiraKakuPro-W3")+
    theme(axis.title = element_text(size = 18, face = "bold"),
          axis.text = element_text(size =  14, color = "black"),
          strip.text = element_text(size = 18, face = "bold")
          )
  )
```

データフレームから直接平均値のグラフを描きたい！

- stat_summary()関数
  - 連続値を集計して描画する
  - fun = 統計量, geom = グラフの種類，審美的属性などを指定

- 平均値を棒グラフで描画，塗りつぶしは白，枠線は黒

```
stat_summary(fun=mean,geom = "bar", fill = "white", col = "black")+
```

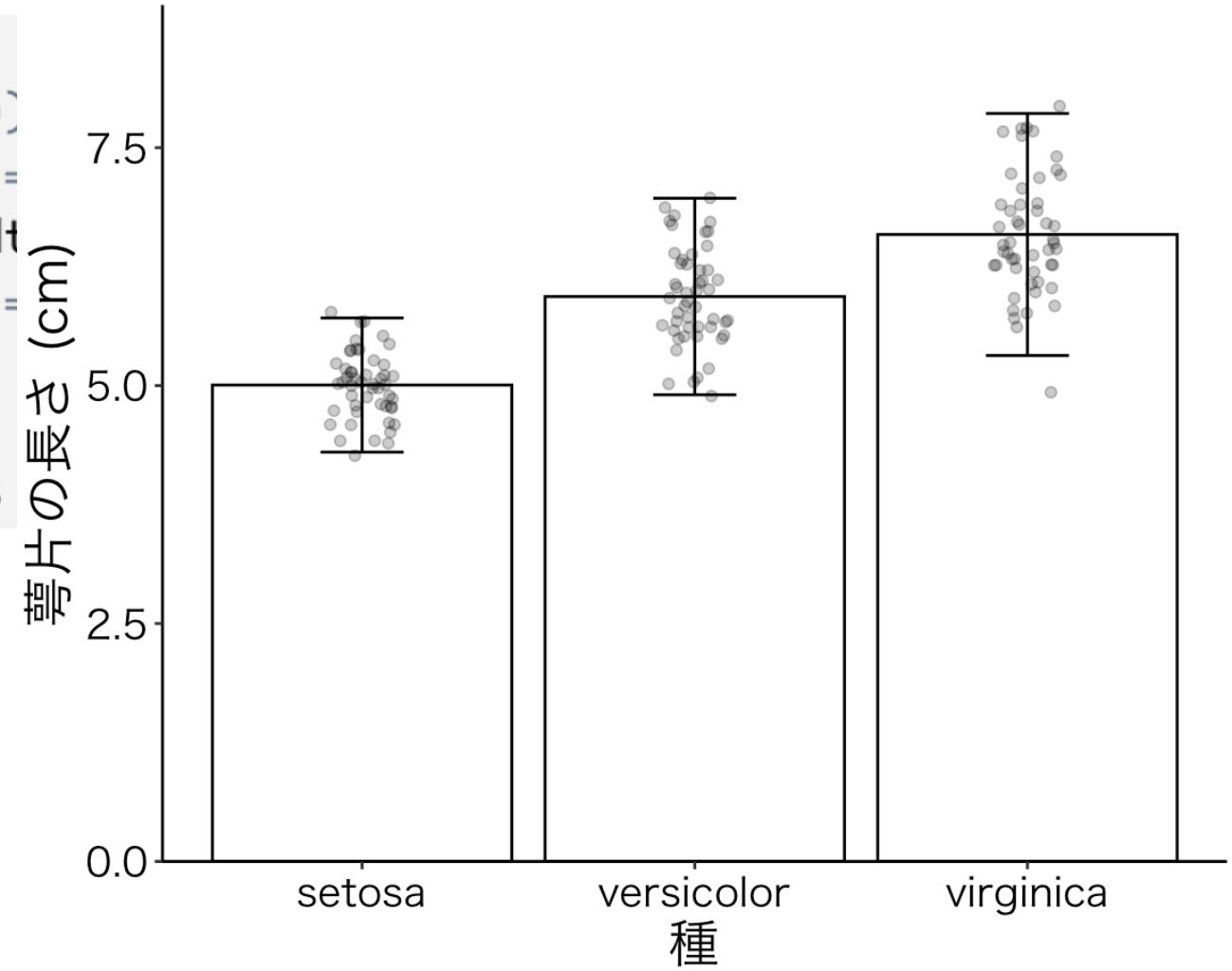
- SDをエラーバーで描画，幅は0.25 ※デフォルトは1

```
stat_summary(fun.data = "mean_sdl", geom = "errorbar", width = 0.25)+
```

stat_summary()で平均値+エラーバーを描き,
さらに個別の観測値をgeom_jitter()でx軸方向に少し散らして描画

```
iris %>%
  ggplot(aes(x = Species, y = Sepal.Length)
  stat_summary(fun=mean,geom = "bar", fill =
  geom_jitter(size = 1.5, alpha = 0.25, widt
  stat_summary(fun.data = "mean_sdl", geom =

  scale_x_discrete(name = "種")+
  scale_y_continuous(expand = c(0,0),limits
```

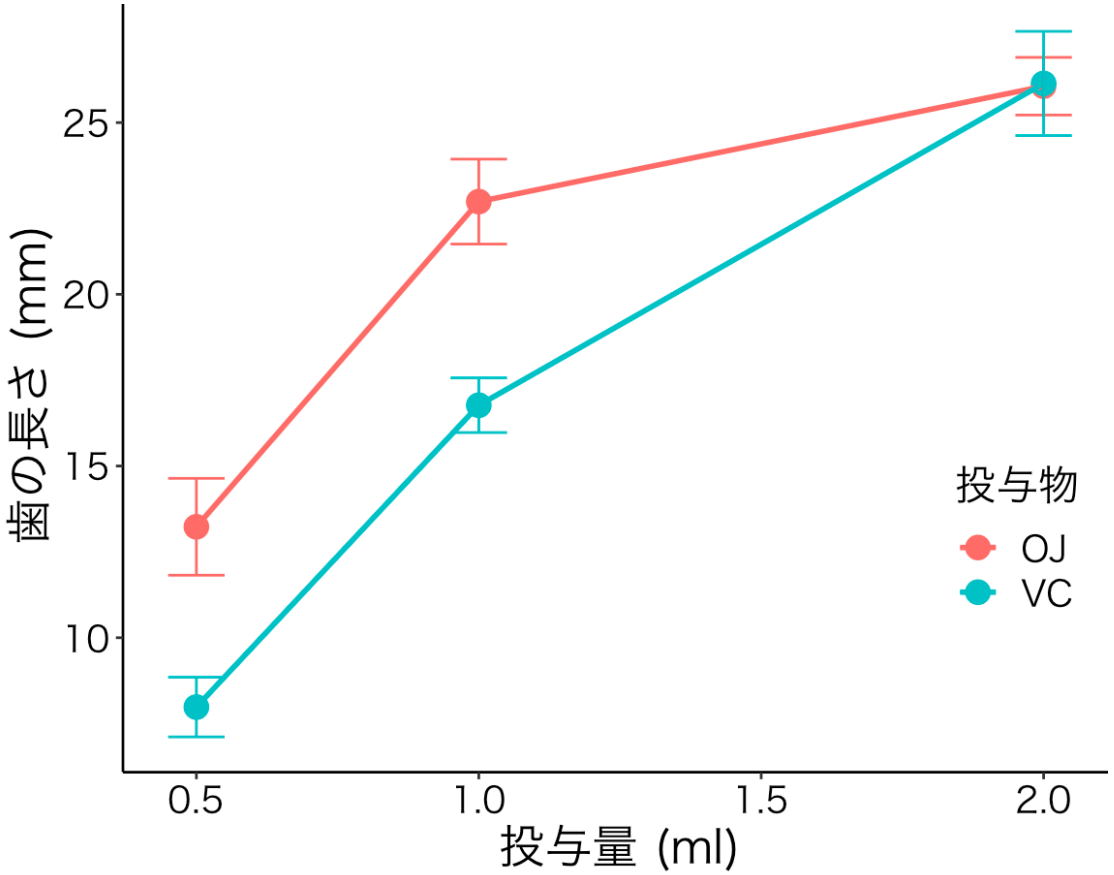
※上から順に描画される



2要因の折れ線グラフを描こう

ToothGrowthを使って，横軸を投与量（dose），縦軸を歯の長さ（len）とする折れ線グラフを投与物（supp）ごとに描こう

```
> head(ToothGrowth)
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
```



グラフを描こう

変数suppによってcol（color）を変える

```
ToothGrowth %>%
  ggplot(aes(x = dose, y = len, col = supp))+
  stat_summary(fun = mean, geom = "point", size = 4)+
  stat_summary(fun = mean, geom = "line", lwd = 1)+
  stat_summary(fun.data = "mean_se", geom = "errorbar", width = 0.1)+

  scale_x_continuous(name = "投与量 (ml)")+
  scale_y_continuous(name = "歯の長さ (mm)")+
  scale_color_discrete(name = "投与物")+

  theme_classic(base_family = "HiraKakuPro-W3")+
  theme(axis.title = element_text(size = 18, face = "bold"),
        axis.text = element_text(size =  14, color = "black"),
        legend.title = element_text(size=16, face = "bold"),
        legend.text = element_text(size = 14),
        legend.position = c(0.9, 0.3)
        )
```

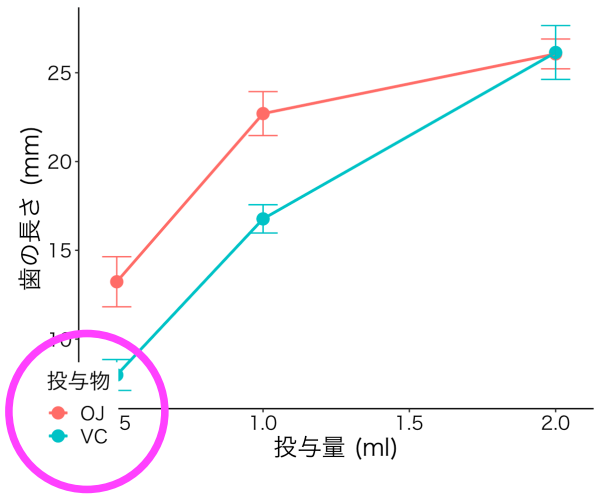
pointでマーカー
lineで線
errorbarでエラーバー（SE）

aes()でcolorを指定した変数の名前を変える：凡例タイトルに

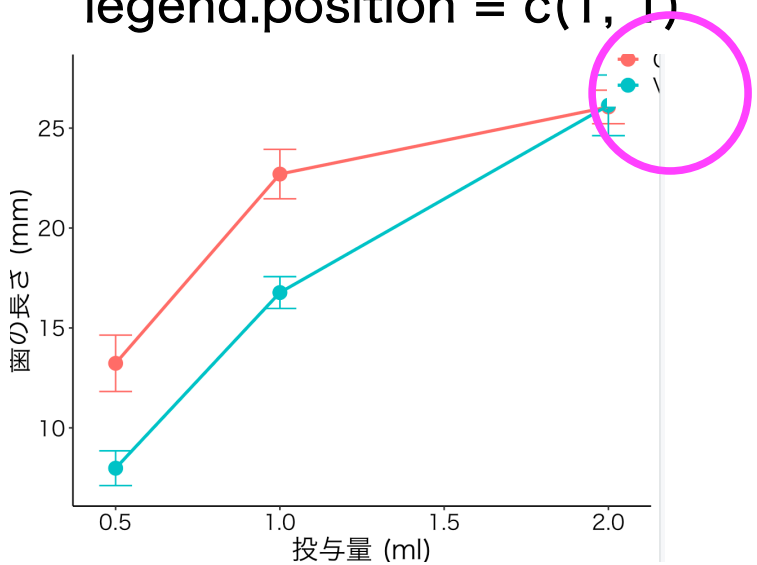
legend.title：凡例タイトル
legend.text：凡例の中身
legend.position：凡例の位置（ベクトルで指定）

凡例の座標：legend.position = c(x, y)

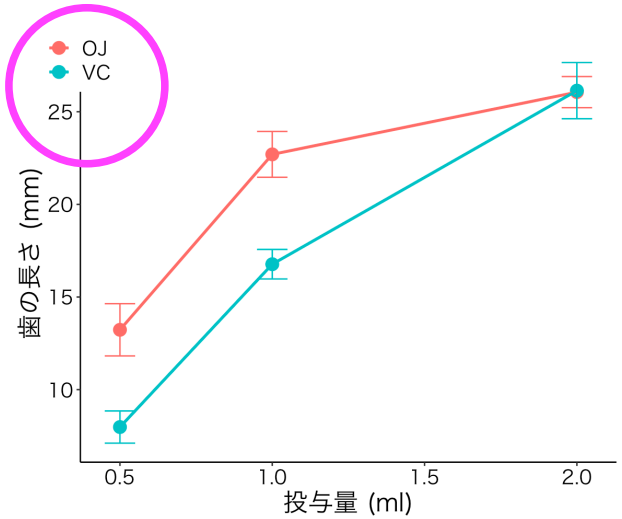
legend.position = c(0, 0)



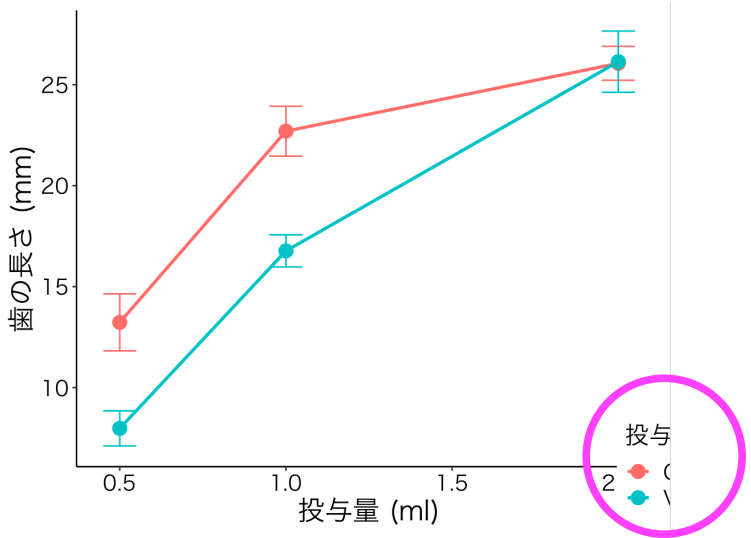
legend.position = c(1, 1)



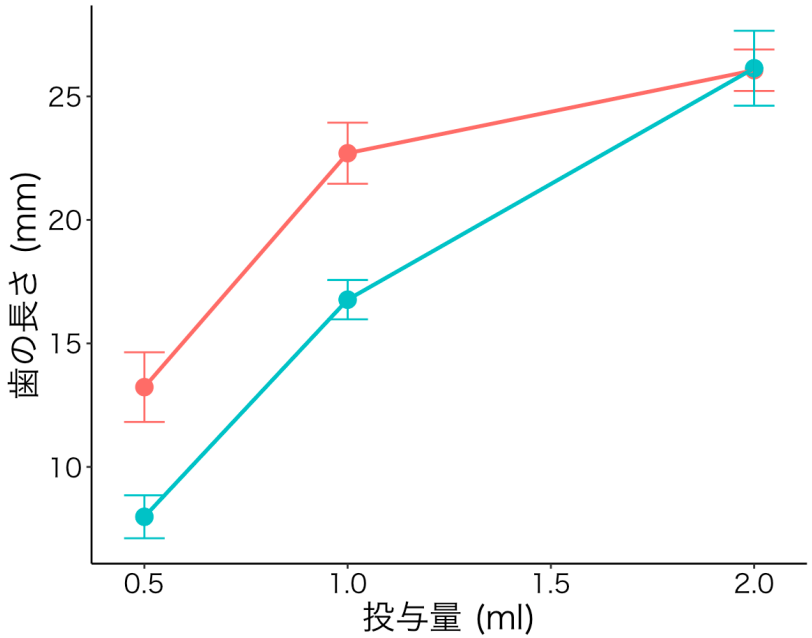
legend.position = c(0, 1)



legend.position = c(1, 0)



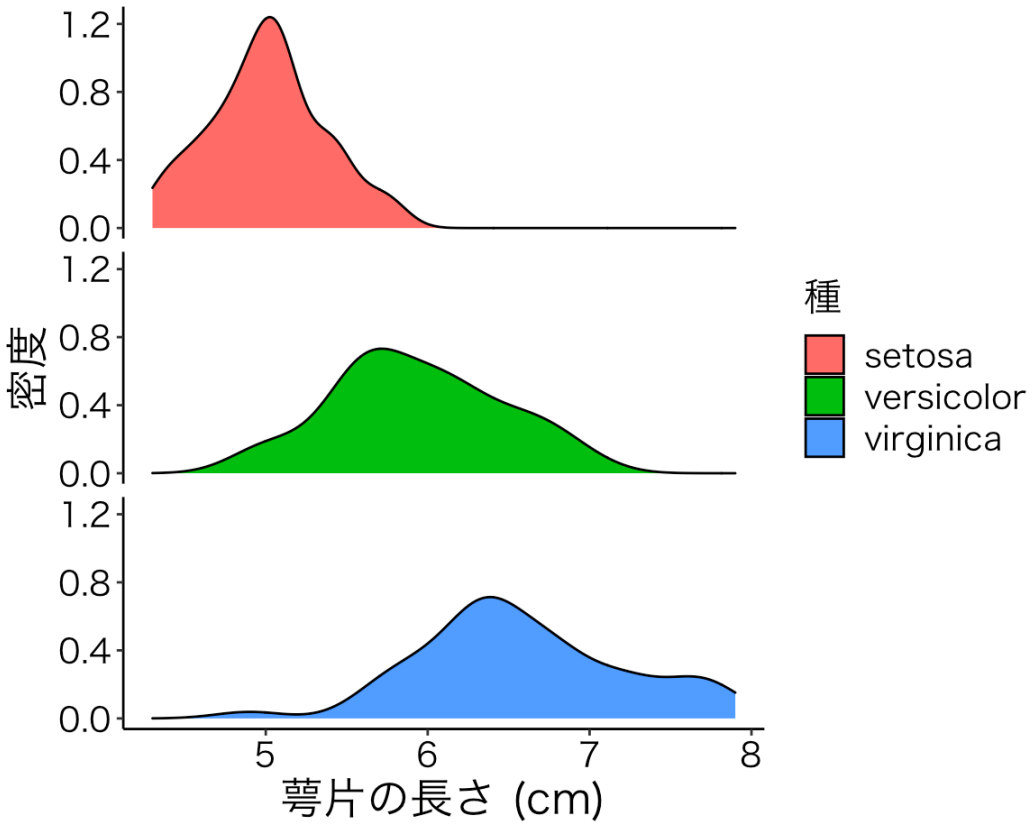
legend.position = NaN
legend.position = “none”



グラフの色を好きに指定しよう

irisから，Speciesで層別化したSepal.Lengthの密度分布を描こう

```
iris %>%
  ggplot(aes(x = Sepal.Length, fill = Species))+
  geom_density()+
  facet_grid(Species~.)+

  theme_classic(base_family = "HiraKakuPro-W3")+
  scale_x_continuous(name = "萼片の長さ (cm)")+
  scale_y_continuous(name = "密度")+
  scale_fill_discrete(name = "種")+

  theme(axis.title = element_text(size = 18, face = "bold"),
        axis.text = element_text(size =  14, color = "black"),
        legend.title = element_text(size=16, face = "bold"),
        legend.text = element_text(size = 14),
        strip.background = element_blank(),
        strip.text = element_blank()
        )
```

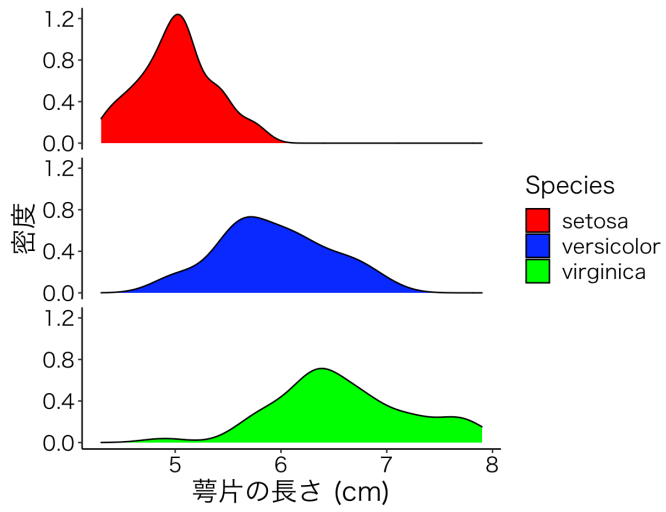


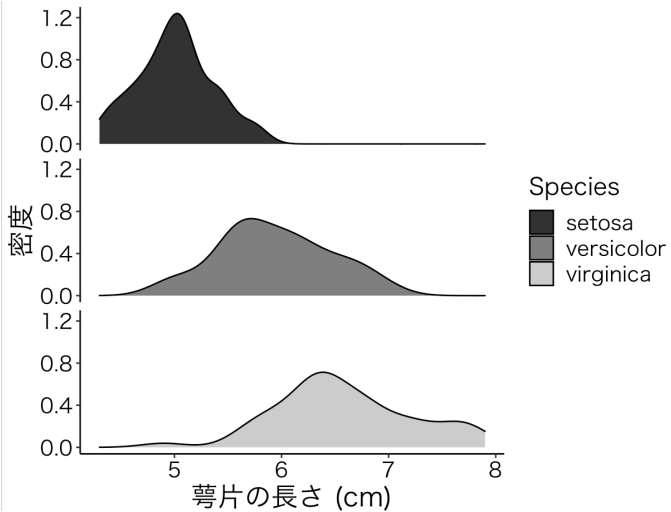
- scale_fill_manual()：塗りつぶし色をマニュアルで指定
- scale_color_manual()：枠線の色をマニュアルで指定

コンソールで `colors()` と打つと，使える色名（657色）が出る

```
scale_fill_manual(values = c("red", "blue", "green"))+
```

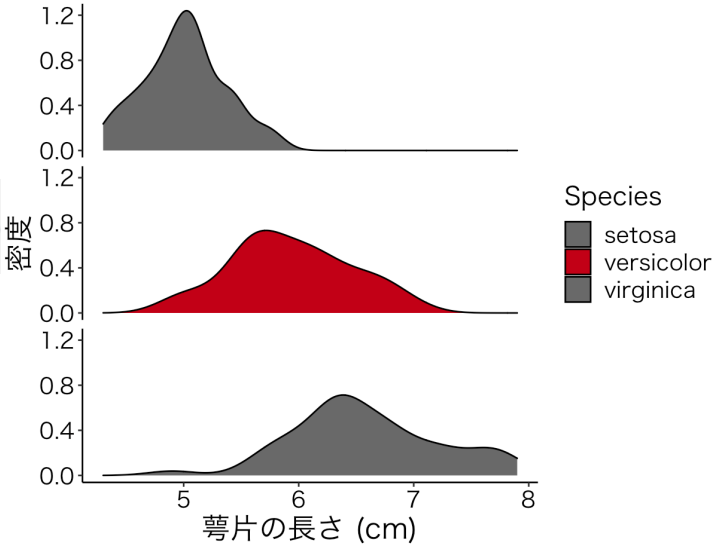
```
scale_fill_manual(values = c("grey20", "grey50", "grey80"))+
```





カラーコードでも指定できる（col = “#FF0000”）

```
scale_fill_manual(values = c("#696969", "#b22222", "#696969"))
```



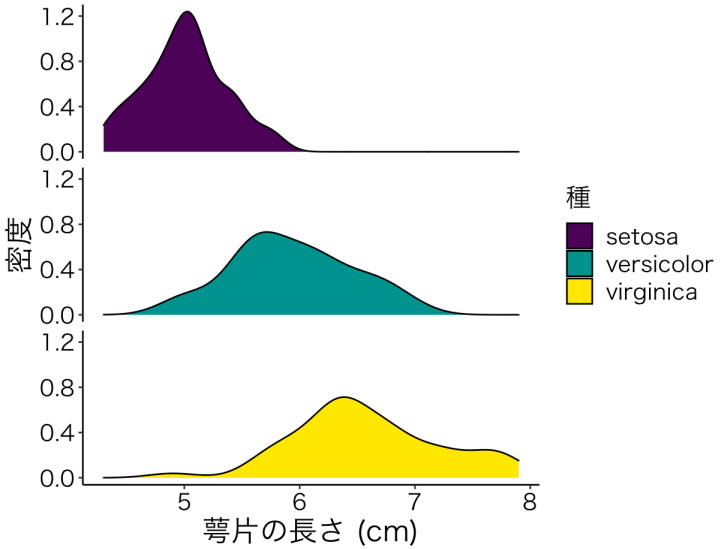
色覚異常に配慮されたカラーマップ「viridis」を使おう

- scale_color_viridis_c(), scale_color_viridis_d()
- scale_fill_viridis_c(), scale_fill_viridis_d()

※*_c()は連続値，*_d()は離散値の変数

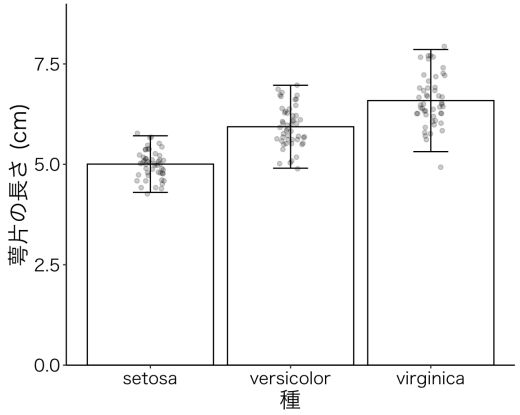
```
scale_fill_viridis_d(name = "種")+
```

質的変数Speciesのfillを変えたいので，*_d()

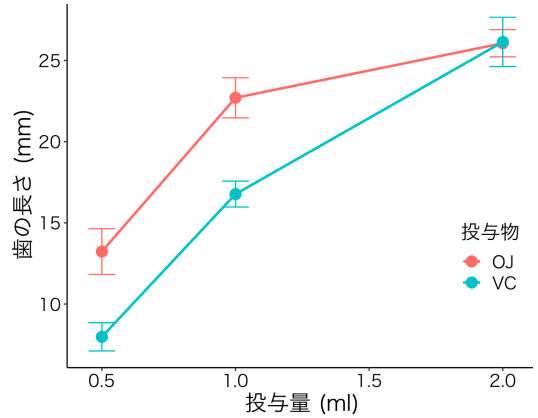


複数のプロットをまとめよう

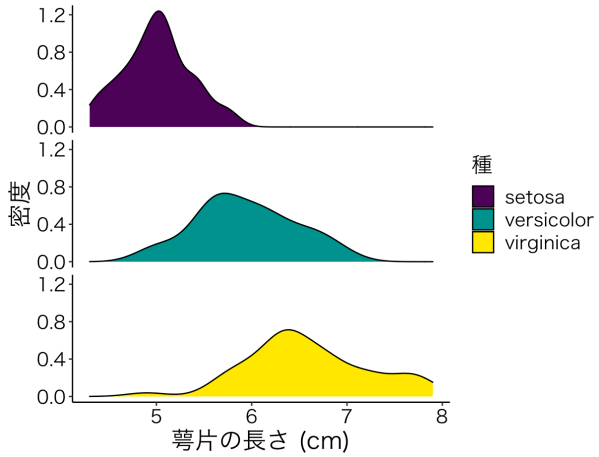
p1 <- ggplot()+…

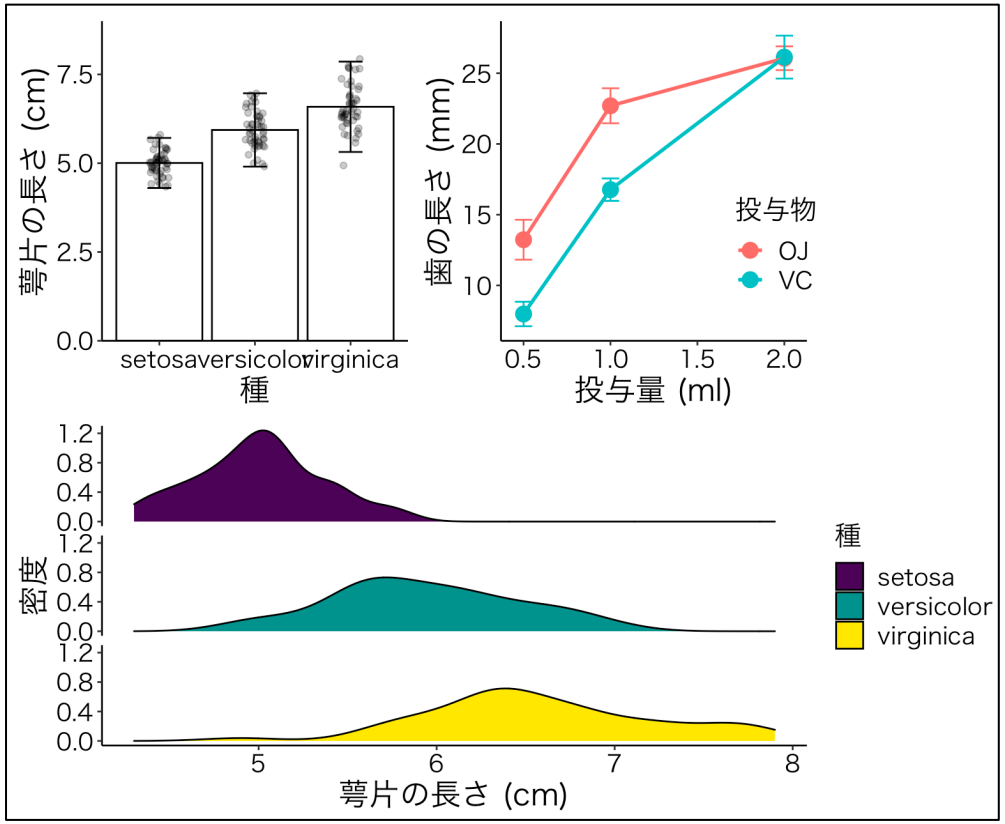


p2 <- ggplot()+…



p3 <- ggplot()+…





複数のグラフを一つのパネルにまとめるパッケージ

- gridExtra
- patchwork
- cowplot
- ggpubr
- egg

今回は”patchwork”パッケージを使ってみましょう

コンソールでインストールして，`> install.packages("patchwork")`

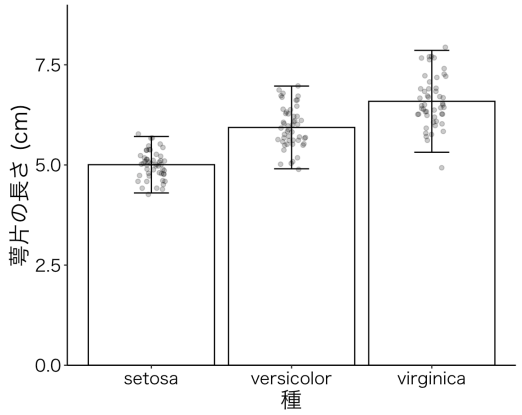
Rmdで読み込み `library(patchwork)`

- まとめたい図に名前を付けておく

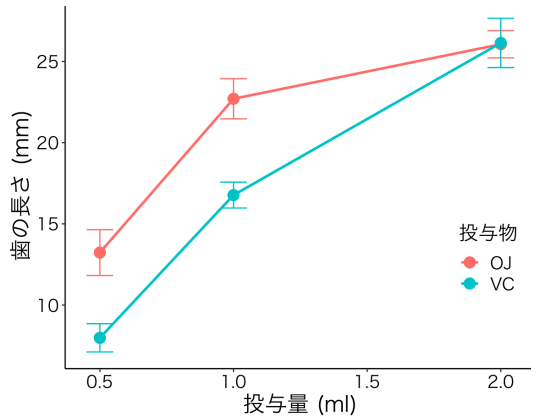
```
legend.text = element_text(size = 14
strip.background = element_blank(),
strip.text = element_blank()
) -> p1
```

ちょっと横着ですが，今日作った図のコードの最後に右向きの代入記号を書き，適当な名前のオブジェクトに格納する

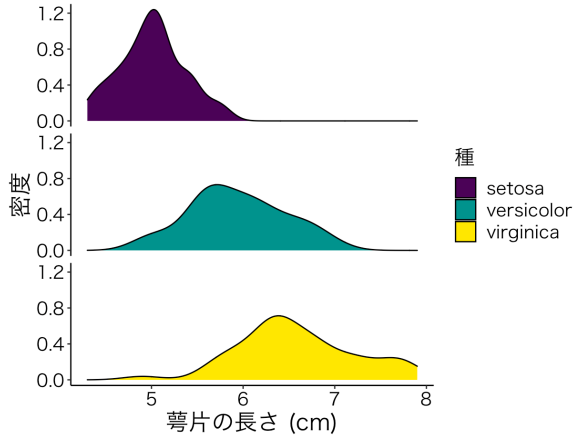
p1



p2



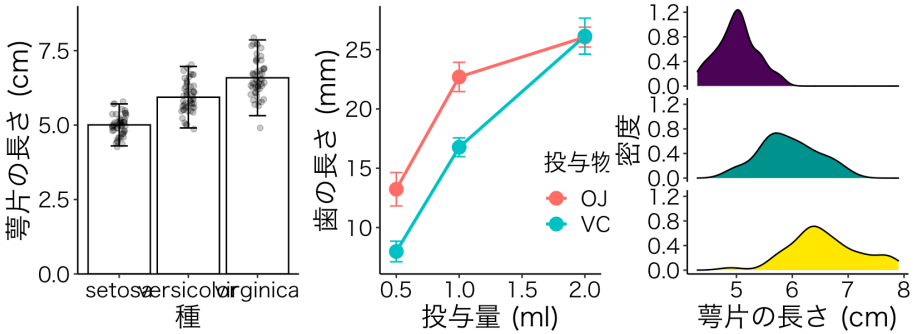
p3



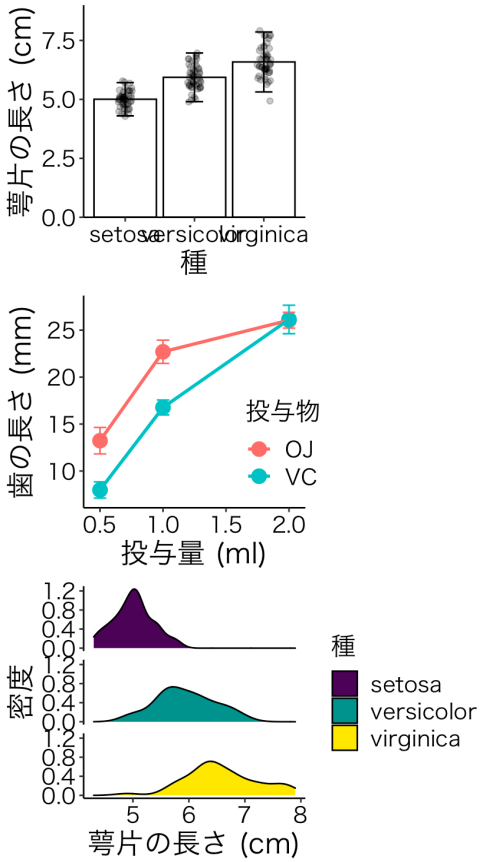
- 図を演算子で連結してまとめられる

()：図をまとめる
+：単純に連結
|：隣に連結
/：下に連結

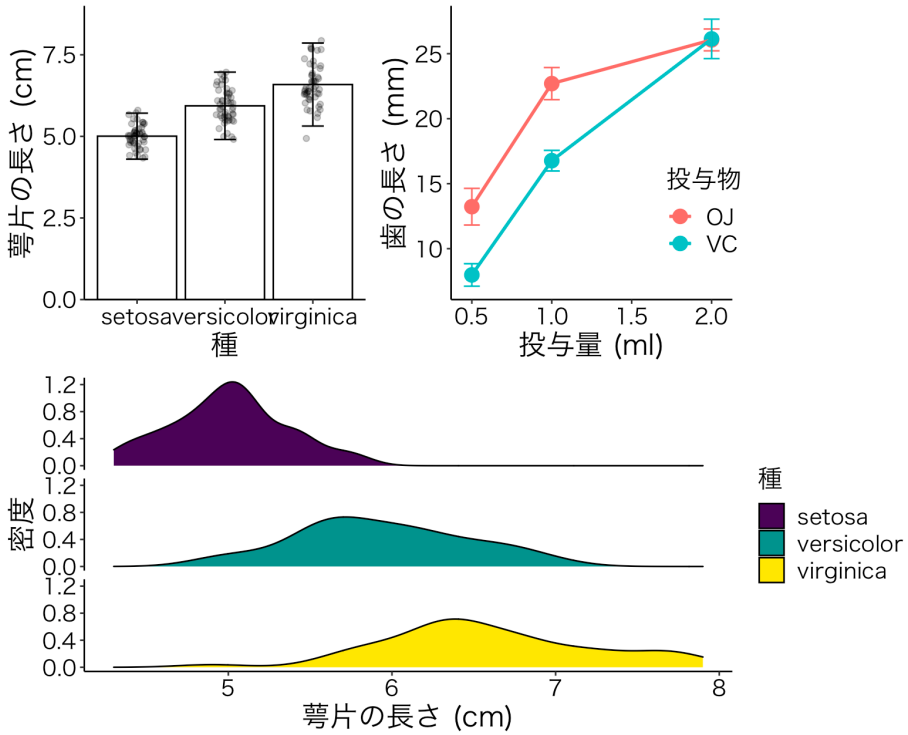
```
p1 + p2 + p3
```




```
p1 / p2 / p3
```




```
(p1|p2)/p3
```

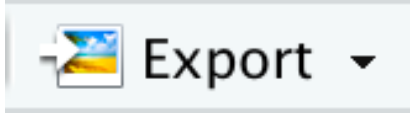


図を保存しよう

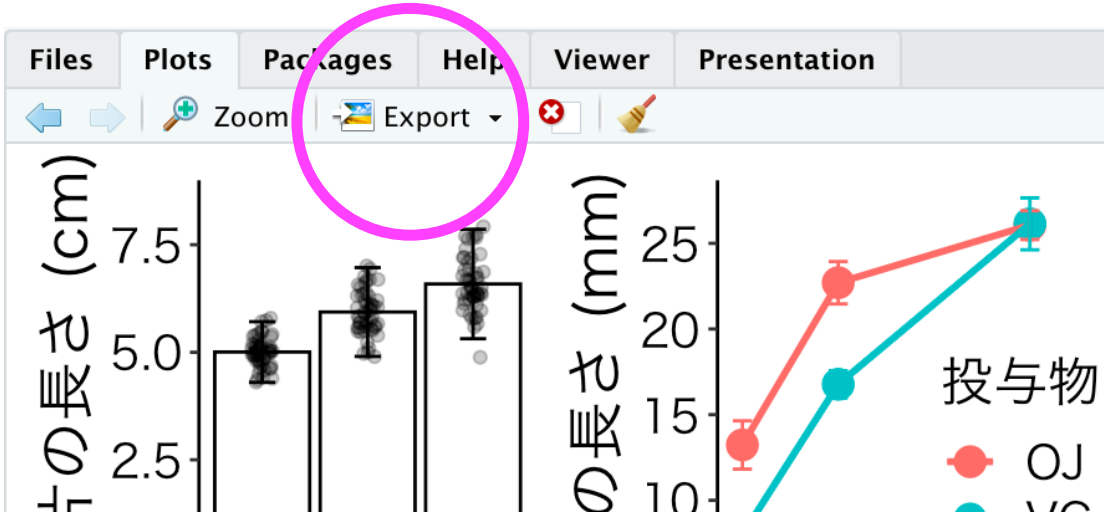
- 保存したい図に名前を付けておく `p_all <- (p1|p2)/p3`

- **ggsave()関数：図を作業ディレクトリに保存する**
  - PNG，PDF，JPEGなどのフォーマットで保存できる
  - ggsave(“ファイル名.拡張子”，保存したい図)

```
ggsave("Figure1.png", p_all)
```

- RStudioの 

からも保存できる



データフレームをCSVファイルに書き出そう

- CSV（コンマ区切り）ファイル
  - 各項目がコンマで区切られた形式のテキストファイル
  - Rで扱いやすく，Excelでも読み書きできる

readr：tidyverse内のデータの入出力用パッケージ

- **write_csv()関数**：CSVファイルを出力
  - write_csv(書き出したいデータ，”ファイル名.csv”)

```
write_csv(iris, "iris.csv")
```

→　現在の作業ディレクトリにiris.csvというCSVファイルができているはず！
　　作業ディレクトリの確認はgetwd()

CSVファイルを読み込もう

- 自分の実験データ（CSVファイル）をRに読み込んで扱いたい！
  - Excelファイルの読み込みは手間がかかるのでCSVにしておく
  - ロング型のデータ構造になっていることが望ましい

	A	B	C	D	E
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa

今作った”iris.csv”を読み込んでみよう

- readrの**read_csv()**関数

  dというオブジェクトに”iris.csv”を格納する

```
d <- read_csv("iris.csv")
```

※現在の作業ディレクトリ外のファイルは絶対パスを指定する必要がある

```
d <- read_csv("/Users/yamasaki/Dropbox/iris.csv")
```

Rで一元配置分散分析をしよう

- RでANOVAができる関数はいろいろある
  - aov()やanova()がデフォルトで使える

  - 大正大学の井関龍太先生が開発「ANOVA君」
    http://riseki.php.xdomain.jp/index.php?ANOVA%E5%90%9B
    ✓とても便利でおすすめなのですが，井関先生の上記HPからインストールが必要なのと，ANOVA君用にデータを整形する必要があるので今回は断念

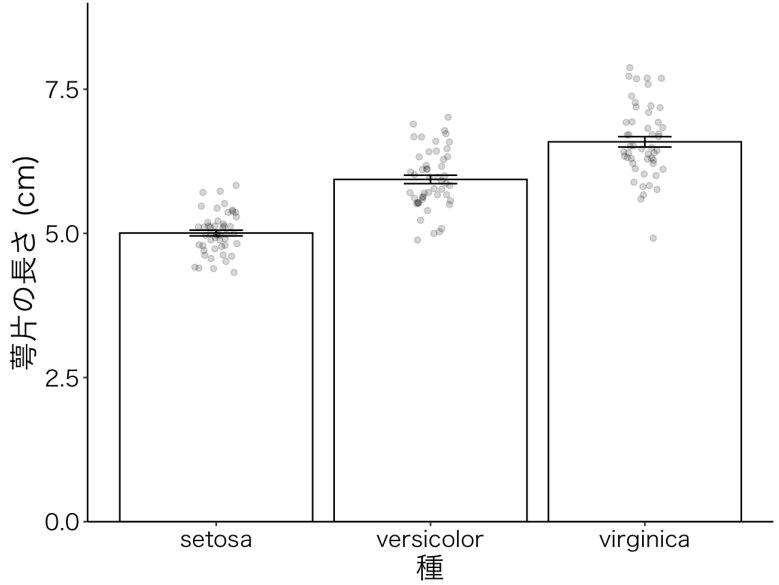
- carパッケージのAnova()関数では平方和タイプが指定でき，アンバランスなデータにも対応できる

```
> install.packages("car")
```

```
library(car)
```

Rで一元配置分散分析をしよう

SpeciesによってSepal.Lengthに違いはあるか？



- 線形モデルを記述するlm()関数
  - lm(従属変数 ~ 独立変数, data = データフレーム名)
  - mと名前を付けておく

```
m <- lm(Sepal.Length ~ Species, data = d)
```

- Anova()関数で分散分析表を出力

```
Anova(m, type = 2)
```

萼片の長さは種によって有意に異なる
$F(2, 147) = 119.26$, $p < .001$

```
Anova Table (Type II tests)

Response: Sepal.Length
             Sum Sq  Df F value    Pr(>F)
Species      63.212   2  119.26 < 2.2e-16 ***
Residuals 38.956 147
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Rで一元配置分散分析をしよう

どう異なっているのかを多重比較したい

- Holmの方法で調整された*p*値を出力

```
pairwise.t.test(d$Sepal.Length, d$Species)
```

```
        Pairwise comparisons using t tests with pooled SD

data:  d$Sepal.Length and d$Species

           setosa  versicolor
versicolor 1.8e-15 -
virginica  < 2e-16 2.8e-09

P value adjustment method: holm
```

- Tukeyの方法

```
TukeyHSD(aov(d$Sepal.Length~d$Species))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = d$Sepal.Length ~ d$Species)

$`d$Species`
                          diff       lwr       upr p adj
versicolor-setosa    0.930 0.6862273 1.1737727     0
virginica-setosa     1.582 1.3382273 1.8257727     0
virginica-versicolor 0.652 0.4082273 0.8957727     0
```

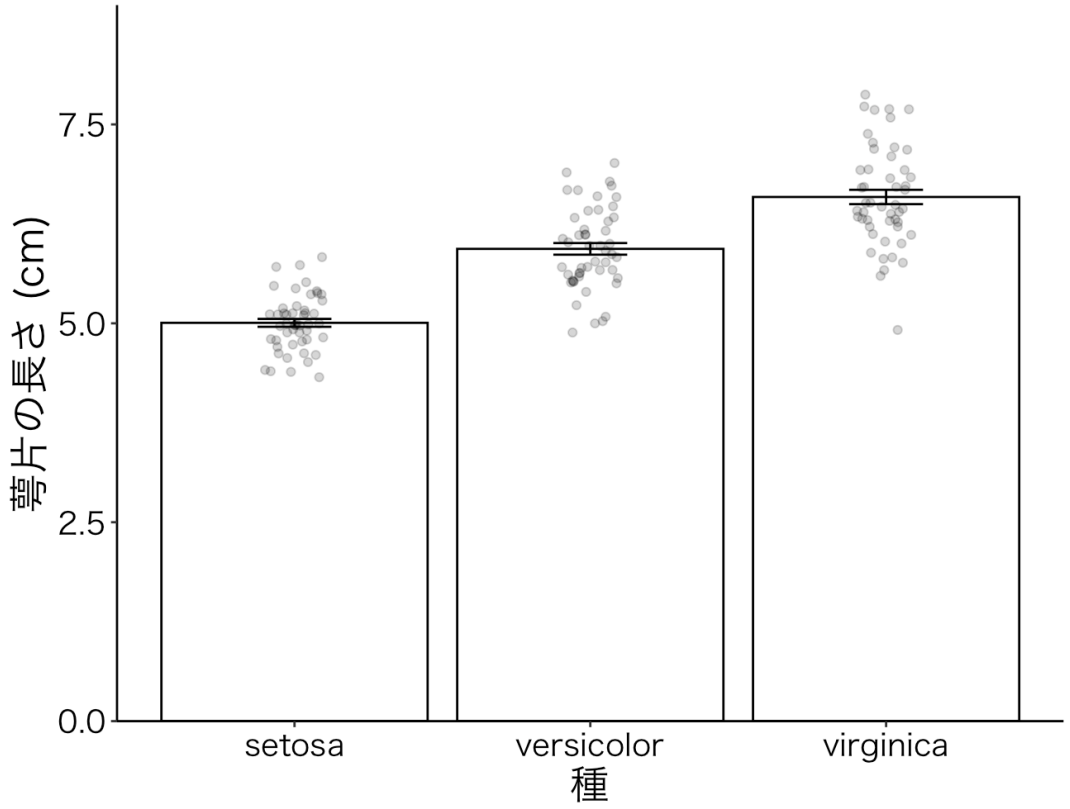
どの水準間にも有意差がある

Rで一元配置分散分析をしよう

- 各種の萼片の長さの平均値を確認

```
d %>%
  group_by(Species) %>%
  summarise(Mean = mean(Sepal.Length))
```

```
# A tibble: 3 × 2
  Species    Mean
  <chr>     <dbl>
1 setosa     5.01
2 versicolor 5.94
3 virginica  6.59
```

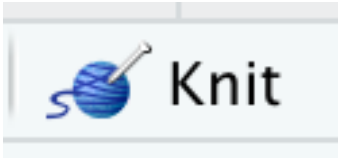


萼片の長さは種によって異なり,
virginica, versicolor, setosaの順に長いことがわかった

さまざまな分析を実行する関数

- t検定：t.test()
- 分散分析：aov()，anova()，Anova()，anovakun()
- 無相関検定：cor.test
- カイ二乗検定：chisq.test()
- ウィルコクソン検定：wilcox.test()
- 線形モデル：lm()
- 一般化線形モデル：glm()
- 一般化線形混合モデル：glmer()，glmmML()，lmer()

最後に



- RMarkdownをknitしておきましょう
  - 分析結果をまとめて確認できたり，htmlファイルを指導教員・学生・共同研究者とシェアできてとても便利です

- RとRStudioでできることは日々増えています
  - パッケージの開発が盛ん
  - やりたいことはググれば大体出てくる

- おすすめの本など
  - 改定2版　RユーザのためのRStudio[実践]入門 –tidyverseによるモダンな分析フローの世界-（松村優哉他著，技術評論社）
  - ゼロから始める統計モデリング（堀裕亮著，ナカニシヤ出版）
  - 私たちのR：ベストプラクティスの探求（宋財泫・矢内勇生, https://www.jaysong.net/RBook/）
  - ggplot2逆引き（http://yutannihilation.github.io/ggplot2-gyakubiki/）