回帰分析による統計的分析

glimps() ：定義した表を出す関数

Rでは、**lm()** で回帰式を推定することができる

基本的な結果は、**summary()** または**tidy()** で見ることができる

得票率（結果変数）を議員経験（説明変数）で説明するモデル

mutate関数：表に追加する

1. expが選挙費用なのでこれを10^6（100万円）で割る

```
hr <- mutate(hr,
       exp_m = exp / 1e6)
```

（mutateで選挙費用を１００万円で割って、exp_mを表に追加し、hrを再定義）

2. 新人が議員経験なし。現職、元職は議員経験ありなので、そこを利用して**新たな議員経験あるかないかを示す欄を作る**

```
hr <- mutate(hr,
             experience = as.numeric(status == "現職"| status == "元職"))
#（mutate関数でexperienceの欄を作る。Experienceのなかでstatusが現職と元職は議員経験ありなので、1と定義、ほかは0と定義し、表hrを再定義する）
```

3. 2009年のデータをhrから抽出し、そのデータをhr09と定義する

```
hr09 <- filter(hr, year == 2009)
```

線形回帰分析の実装

単回帰の場合

```
fit1 <- lm(voteshare ~ experience, data = hr09)
```

summary(fit1) または tidy(fit1) で回帰直線を表示

傾きを求める

```
傾き＝xとy共分散/xの分散
```

であるので、この公式を用いる。

ここでは＄演算子を用いることで、表の一部の場所を抽出可能

```
cov()：共分散  var():分散

cov(hr09$voteshare, hr09$experience) / var(hr09$experience)
```

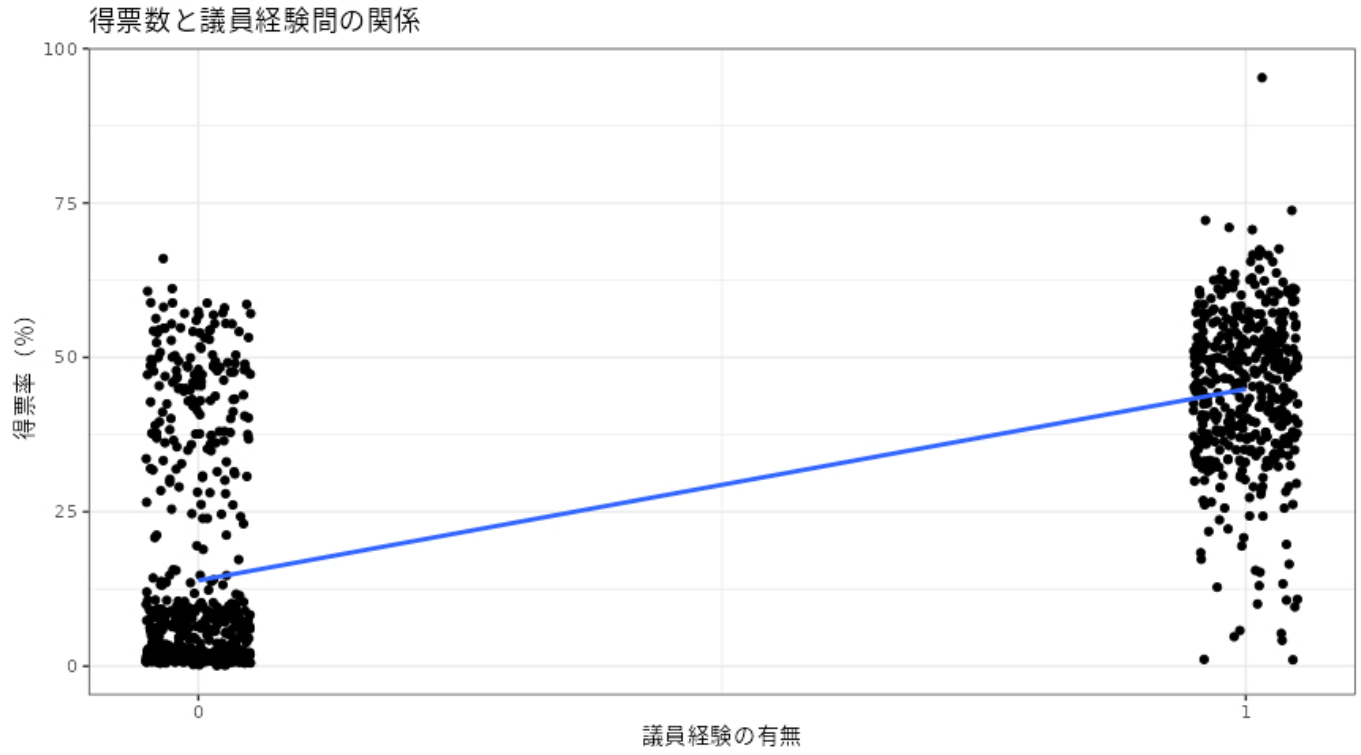
散布図を表示する

ggplotで表示、aesで縦軸横軸を定義 geomが図を表す jitter：散布図ダミー変数の時、かたまるから見やすくするために間をあける（散布図を描くpoint関数で書くと、重なって見えないので…） geom_smooth(method = "lm", se = FALSE)でlmを指定。あたらしく関数の式を書く必要がない scale_x_continuous(breaks = c(0, 1))：ダミー変数なので、０と１の間に値があったらおかしいので除く

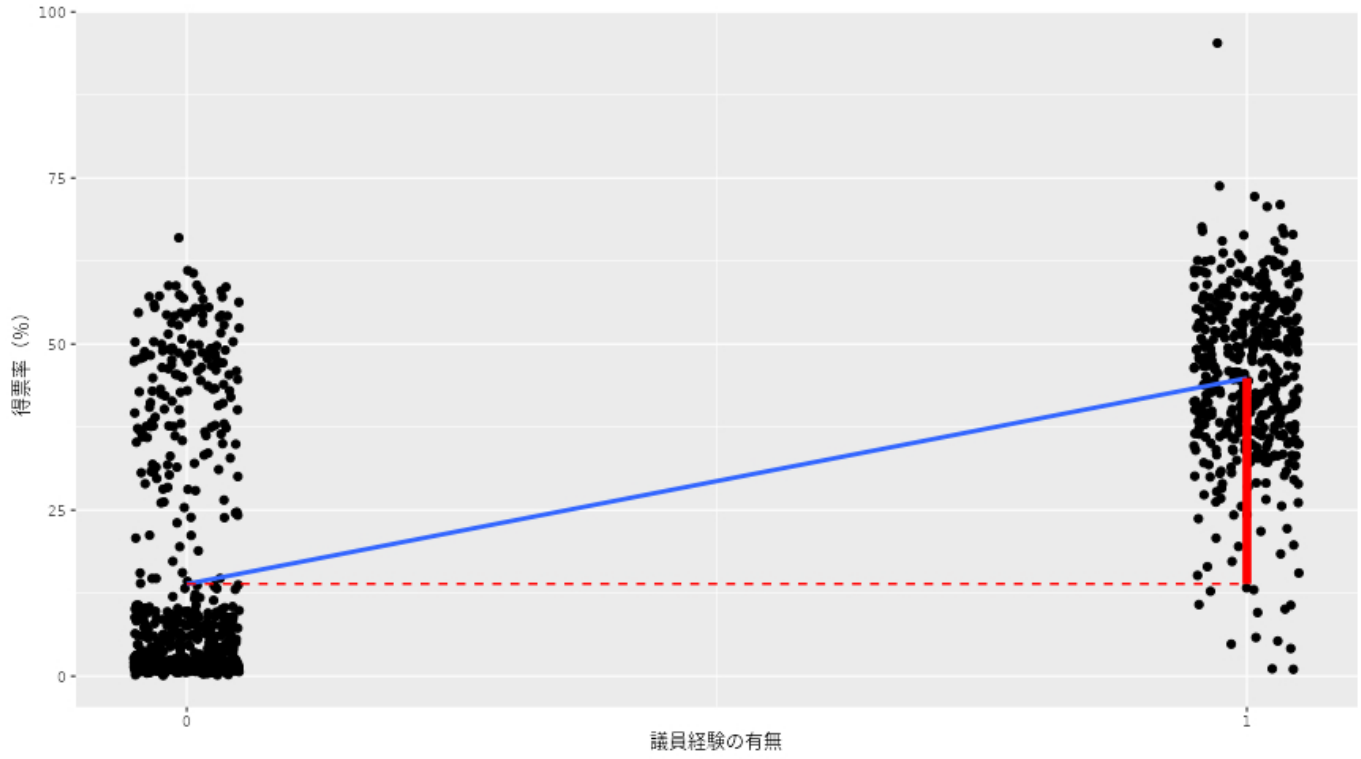
```
p1 <-   ggplot(hr09, aes(x = experience, y = voteshare))+
 #ggplotで表示、aesで縦軸横軸を定義
  geom_jitter(position = position_jitter(width = 0.05)) +
  #geomが図を表す
  #jitter：散布図ダミー変数の時、かたまるから見やすくするために間をあける（散布図を描くpoint関数で書くと、重なって見えないので…）
  geom_smooth(method = "lm", se = FALSE)+  #lmを指定。あたらしく関数の式を書く必要がない
  scale_x_continuous(breaks = c(0, 1)) + #ダミー変数なので、０と１の間に値があったらおかしいので除く
  labs(x = "議員経験の有無", y = "得票率（％）")  #labsでラベル表示
```

装飾

```
p1 +
  ggtitle("得票数と議員経験間の関係")  + #タイトル表示
  theme_bw()  #背景を白黒に変え見やすくする
```




```
p1 +
  annotate("segment", x = 0, xend = 1, y = 13.9, yend = 13.9,  #横の長さを塗る（赤の点線）
           color = "red", linetype = "dashed") +
  annotate("segment", x = 1, xend = 1, y = 13.9, yend = 13.9 + 31,  #縦の長さを塗る（赤の太線）
           color = "red", linewidth = 2 )
```



平均

mean関数：平均の関数

hr09の中のvoteshareの値を平均する。ただし、experienceの1と示されているところを抽出

```
mean(hr09$voteshare[hr09$experience == 1])
```

hr09の中のvoteshareの値を平均する。ただし、experienceの0と示されているところを抽出

```
mean(hr09$voteshare[hr09$experience == 0])
```

説明変数が量的変数のとき

hr09の中で、被説明変数voteshareを説明変数xp_mで単回帰したものをfit2に定義

```
fit2 <- lm(voteshare ~ xp_m, data = hr09)
```

tidy(fit2) で表示

散布図を表示する

```
p2 <- ggplot(hr09, aes(x = exp_m, y = voteshare))+
  geom_point( size = 1) +
  geom_smooth ( method = "lm", se = FALSE) +
  labs(x="選挙費用（百万円）", y = "得票率（％）")
```

タイトルをつける

```
p2 + ggtitle("選挙費用と得票率の関係")
```

以上の結果を並べて比較する

```
fits <- list("Model 1" = fit1,
             "Model 2" = fit2)
```

表にする

```
screenreg(fits)
```

（応答結果）

```
=====================================
             Model 1      Model 2
-------------------------------------
(Intercept)    13.88 ***     7.74 ***
               (0.62)       (0.76)
experience     30.99 ***
               (0.98)
exp_m                        3.07 ***
                            (0.10)
-------------------------------------
R^2             0.47         0.48
Adj. R^2        0.47         0.48
Num. obs.    1139         1124
=====================================
*** p < 0.001; ** p < 0.01; * p < 0.05
```

これを表にする modelsummary(fits, coef_rename = c("(Intercept)" = "切片", "experience" = "議員経験", "exp_m" = "選挙費用（100万円）"))

Modelsummary：回帰分析の結果を表を出す　　coef_rename：変数名を変える関数

ファイルの読み込み

CSV形式の読み込み

```
my_csv1 <- read_csv("data/fake_data_02.csv")
my_csv2 <- read_csv("data/fake_data_03.csv")
```

Excel データの読み込み

```
my_xls <- read_excel("data/fake_data_03.xlsx")
```

Stata形式のデータの読み込み

```
my_dta <- read_dta("data/fake_data_stata.dta")
```

SAV形式のデータの読み込み

```
my_sav <- read_sav("data/fake_data_s
pss.sav")
```

複数のデータセットの扱い

表形式データの変数名（＝列の名前をすべて）出力

```
names(my_csv1)
names(my_csv2)
```

使うデータの指定

```
mean(my_csv1$age)
mean(my_csv2$age)
```

データの前処理

欠損値の処理

```
myd <- read_csv("data/hr-data.csv")
```

csvファイルをmydと定義

データの抽出

Ex）1996年の自民党議員のみ抽出したい

表の中から年と政党名を抽出

table関数:表にする

```
table(myd$year, myd$party_jpn)
```

または

```
with(myd, table(year, party_jpn))
```

行の抽出

filter関数を使う

```
filter(myd,
       year == 1996 & party_jpn == "自民党")
```

または、パイプ演算子|>を使う（＝第一引数を省略する）

```
ldp1996 <- myd |>
  filter(myd, year == 1996 & party_jpn == "自民党")
```

列を抽出

select関数を使う

year, name, vsの列を一気に抽出

```
sub1 <- myd |>
  select(year, name, vs)
```

kunからpartyまでの列を抽出する

```
myd |> select(kun:party)
```

列を除外する

- か、 !を使う

```
myd |>
  select(-c(kun:party))
```

列の名前にpartyがあるものだけを抽出する

```
myd |>
  select(starts_with("party"))
```

最後にeが入っているデータだけを抜き出す

```
myd |>
  select(ends_with("e"))
```

列名にteが入っているデータだけを抜き出す

```
myd |>
  select(contains("te"))
```

以上の技を組み合わせて、列と行を一気に処理する

```
ldp1996_sub1 <- myd |>
  filter(year == 1996 & party_jpn == "自民党") |>
  select(year, name, vs)

# 表示する
ldp1996_sub1
```

mutate関数による列の操作・追加

単位などをそろえる

```
myd <- myd |>
  mutate(turnout2 = turnout / 100)
#（turnout2という列を作り、その値はturnoutを100で割ったもの（＝割合）である）

# 表示する
glimpse(myd)
summary(myd$turnout)
summary(myd$turnout2)
```

ダミー変数を作る

```
myd <-
  myd |>   #（mutateでダミー変数の列を追加する）
  mutate(ldp = if_else(party_jpn == "自民党", 1, 0 ),
         dpj = if_else(party_jpn == "民主党", 1, 0 ))
#（もしparty_jpnに自民党と書いてあったら1、違ったら0をldpの欄に記入。もしparty_jpnに民主党と書いてあったら1、違ったら0をdpjの欄に記入。）

#表示する
glimpse(myd)
```

ldpダミー変数の部分を抽出

```
with(myd, table(party_jpn, ldp))

# 表示する
ldp
```

そうすると答えは

```
party_jpn        0    1
  さきがけ      13    0
  その他        87    0
  みんな        79    0
  保守党        16    0
  保守新党      11    0
  公明党        70    0
  共産党      2123    0
  国民党        11    0
  国民新党      21    0
  大地           8    0
  希望         198    0
  幸福実現党   312    0
  新党日本       9    0
  新社会党      38    0
  新進党       235    0
  未来         111    0
  次世代        39    0
  民主党      1654    0
  無所属       562    0
  無所属の会     9    0
  生活          13    0
  社民党       307    0
  立憲民主党    63    0
  維新の会     198    0
  維新の党      77    0
  自民党         0 2266
  自由党        61    0
  自由連合     212    0
```

とでる。

うまく自民党だけ抽出できている。

同じように民主党もやってみよう

```
with(myd, table(party_jpn, dpj))
```

答えは省略する。

装飾

見やすいようにダミー変数の部分を大文字に直す

```
myd <- myd |>
  rename(LDP = lpd, DPJ = dpj)
```

表をWide型からLong型へ変換

データの準備

```
gdp <- read_csv("data/wide-table.csv")
```

この時点での表はこんな感じ

```
# A tibble: 3 × 4
  country gdp2000 gdp2005 gdp2010
  <chr>     <dbl>   <dbl>   <dbl>
1 A         40000   42000   44000
2 B         38000   40000   43000
3 C         52000   52100   52500
```

pivot_longer関数を用いる

```
gdp |>
  pivot_longer(cols      = !country, # 縦長にする範囲：country 以外
               names_to  = "year", # 元の列がどれかを区別する変数：年
               values_to = "gdp") |> # 元の値を保持する変数名
  mutate(year = str_remove(year, "gdp"))  #mutate関数のgdpの欄を作る
```

または、このように一つにまとめられる

```
gdp_long <- gdp |>
  pivot_longer(cols         = !country,
               names_to     = "year",
               names_prefix = "gdp", # 元の変数名のうち、区別に不要な部分
               values_to    = "gdp")
```

これをすると、

```
# A tibble: 9 × 3
  country year    gdp
  <chr>   <chr> <dbl>
1 A       2000  40000
2 A       2005  42000
3 A       2010  44000
4 B       2000  38000
5 B       2005  40000
6 B       2010  43000
7 C       2000  52000
8 C       2005  52100
9 C       2010  52500
```

year を数値型に変換する

```
gdp_long <- gdp_long |>
  mutate(year = as.numeric(year))
```

as.numeric：文字列や因子を数値型に変換する関数

Long型からWide型へ逆変換

pivot_wider関数を用いる

```
gdp_wide <- gdp_long |>
  pivot_wider(cols = !country, #country以外すべてを抽出
              names_from = "year",
              values_from = "gdp")
```

自分で表を作成する

Tibble関数

idで行数指定　nameで数値指定

```
tibble(id = 1:3,
       name = c("Kochi", "Ehime", "Kagawa"))
```

この答えが

```
# A tibble: 3 × 2
     id name
  <int> <chr>
1     1 Kochi
2     2 Ehime
3     3 Kagawa
```

ここから

```
newd <- tibble(id = 1:100,
               x  = rnorm(100, mean = 10, sd = 2)) |>
  mutate(y = 1.2 + 0.8 * x + rnorm(100))
```

回帰直線を描く

Lm関数を用いる

先ほどのnewdの中のデータから回帰式を出す

```
lm(y = x, data = newd) |>
  summary() #表示する
```

回帰分析のシミュレーション

応答変数Xと説明変数Yの真の関係（つまり、母集団における関係）が以下の式で表されるとする。

$Y_i=β_0+β_1X_i+ε_i$

ここで、$β_0$が切片、$β_1$が回帰直線の傾き

$ε_i$は誤差項

$σ$は正規分布 (normal distribution) の標準偏差

つまり、誤差$ε_i$は平均0、標準偏差$σ$の正規分布に従う。

サンプルサイズを決める

```
N <- 5
```

乱数生成　runif関数

次に、Xの値を決める。とりあえず、$[-5, 5]$の一様分布からの実現値（観測値）xをランダムに作ってみよう。

```
x <- runif(N, min = -5,max = 5)
```

これで５つの乱数が形成された

[1] -4.093501 2.369863 -2.426209 1.131144 -1.871808

パラメータ設定する

今回は例として　$β_0=2$,　$β_1=0.8$　として考えよう。

標準偏差$σ=2$として設定する。

パラメーターを設定する

```
b0 <- 2
b1 <- 0.8
sigma <- 2
```

ランダム抽出rnorm関数

rnorm関数でランダムに抽出　epsilonを定義

```
epsilon <- rnorm(N, 0, sigma)
# 0=mean=平均、sigma=標準偏差
```

回帰式の公式に代入

回帰式の公式

$Y_i=β_0+β_1X_i+ε_i$

に代入

```
y <- b0 + b1 * x +epsilon
```

yが回帰式になっている

これをdfに定義して格納

```
df <- tibble(x, y)
```

このdfをlm関数で回帰分析

```
fit <- lm(y ~ x, data = df)
```

表示する

```
tidy(fit)
```

複数回のシミュレーション

単回帰のシミュレーションを実行するための関数を作成

複雑怪奇なので関数として定義してしまう

```
simple_reg <- function(n, beta0 = 0, beta1 = 1, sigma = 1) {
  ## 引数：n = 標本サイズ
  ##       beta0 = 真のy切片（既定値は0）
  ##       beta1 = 真の傾き（既定値は1）
  ##       sigma = 誤差項の標準偏差（既定値は1）

  # x を一様分布 Uniform(-5, 5) から作る
  x <- runif(n, min = -5, max = 5)

  # epsilon を正規分布 N(0, sigma^2) から作る
  epsilon <- rnorm(n, mean = 0, sd = sigma)

  # 真のモデルからyを作る
  y <- beta0 + beta1 * x + epsilon

  # 回帰分析を実行する
  fit <- lm(y ~ x)

  # beta の推定値を関数の出力として返す
  return(coef(fit))
}
```

上記の関数通りに引数を代入

```
simple_reg(n = 5, beta = 2, beta1 = 0.8, sigma = 2)
```

この作業を1000回繰り返したものを定義して行列化

```
n_sims <- 1000
result <- matrix(NA, nrow = n_sims, ncol = 2)
# matrix が行列を作成する関数
# NAが欠損値 空の行列を生成
# nrow：行数, ncol：列数
```

列の名前をb0,b1に変える

```
colnames(result) <- c("b0", "b1")
```

for繰り返し関数

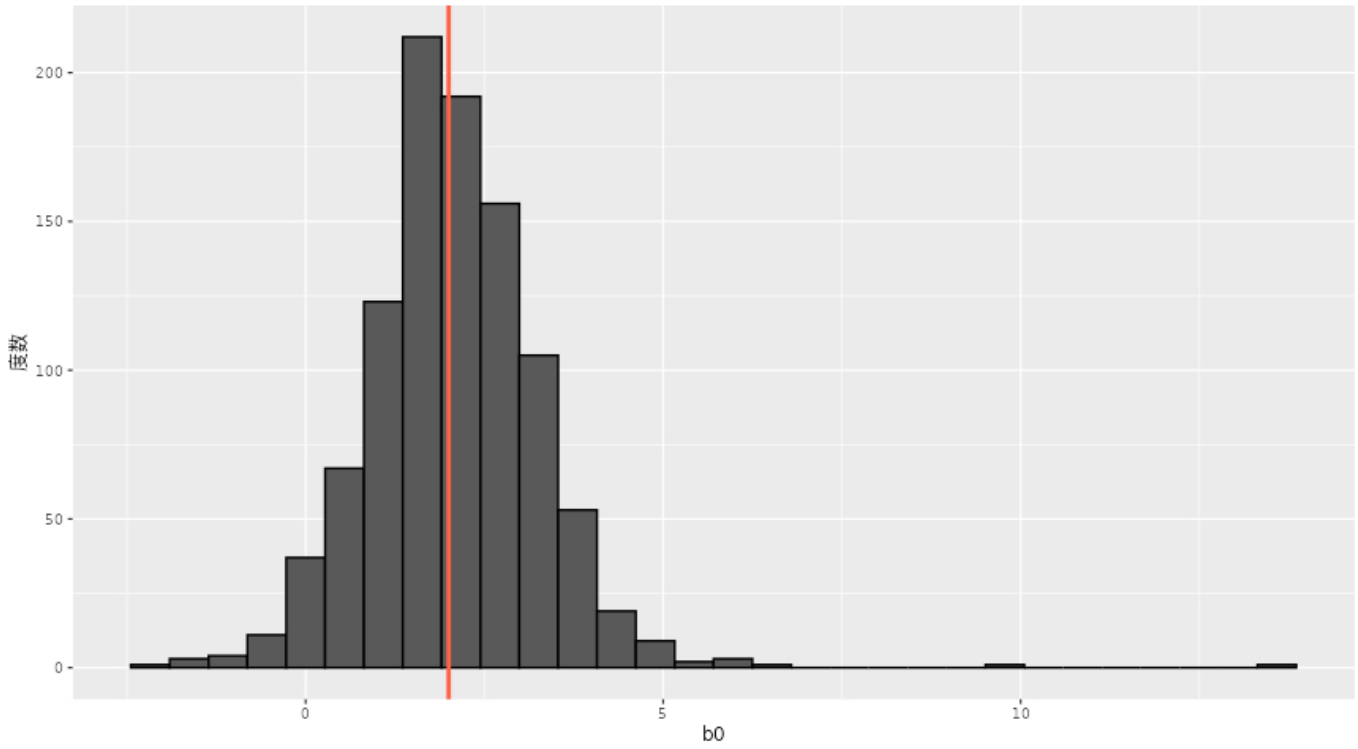
resultのなかにsimple_reg関数一回目を入れたら、次２回目...

n_simsまで繰り返す

```
for (i in 1:n_sims){
  result[i, ] <- simple_reg(n = 5, beta = 2, beta1 = 0.8, sigma = 2)
}

# 行列をデータフレーム型に変換してresult_dfと再定義
result_df <- as_tibble(result)

#結果を出す
result_df
```

この表を使ってヒストグラムを作る

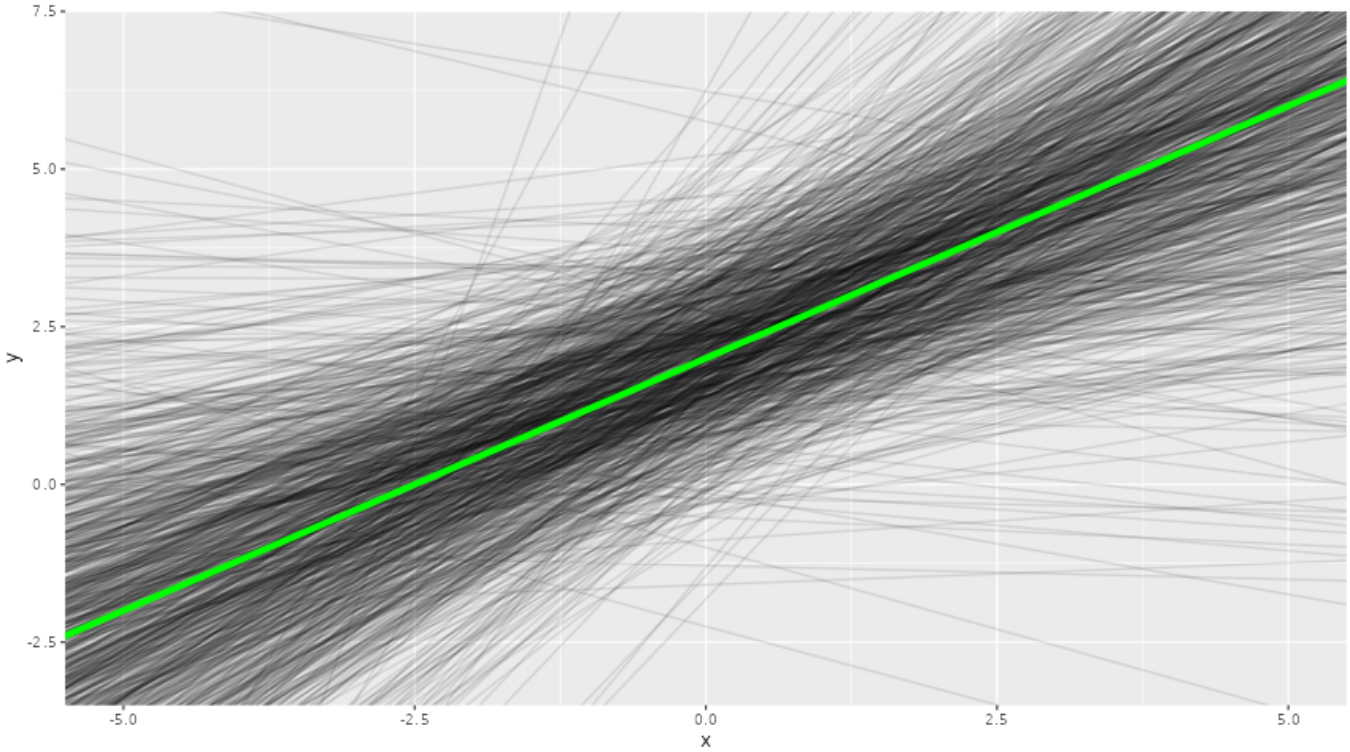
```
result_df |>
  ggplot(aes(x = b0))+
  geom_histogram(color = "black")+
  labs(X = expression(b[0]), y = "度数")+
  geom_vline(xintercept = 2, color = "tomato", linewidth = 1)
```




```
# result_dfのb1の平均値をとる

result_df$b1 |> mean()
```

この千本の推定回帰直線を追加する

```
ggplot(NULL)+ #空白の図を描く
  geom_abline(intercept = result_df$b0,  #interceptで線を引く場所を指定　切片を指定
              slope = result_df$b1,  #slopeで傾き指定　今回はbi傾きを指定
              alpha = 0.1) +
  geom_abline(intercept = 2, #緑が母集団の回帰直線　これが正解の回帰直線
              slope = 0.8,
              color = "green",
              linewidth = 1.5) +
  xlim(-5, 5) +  #グラフの横軸の表示範囲
  ylim(-3, 7) +  #グラフの縦軸の表示範囲
  labs(x = "x", y = "y")  #縦軸横軸のラベル変え
```




```
theme_bw()  #グラフの背景を変える
```

回帰分析における仮説検定

データの準備

```
hr <- read_csv("data/hr-data.csv")
```

元職現職と新人を分けるダミー変数を作る

```
hr <- hr |>
  mutate(experience = if_else(status == "新人", 0, 1))
```

hrから1996年のデータのみ抽出

```
hr1996 <- hr |>
  filter(year == 1996)
```

単回帰の例 1

1996年において議員経験 X が得票率 Y へ与える影響を調べる

lm関数を使って回帰

```
fit1 <- lm(voteshare ~ experience, data = hr1996)
summary(fit1) #結果を出す
```

結果このようにでてくる（値は回帰のため毎回バラバラ）

```
Call:
lm(formula = voteshare ~ experience, data = HR1996)

Residuals:
    Min      1Q  Median      3Q     Max
-38.334 -10.007  -2.207   8.593  67.393

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.0070     0.4608   34.74   <2e-16
experience   22.8274     0.7891   28.93   <2e-16

Residual standard error: 13.28 on 1259 degrees of freedom
Multiple R-squared:  0.3993,    Adjusted R-squared:  0.3988
F-statistic: 836.8 on 1 and 1259 DF,  p-value: < 2.2e-16
```

表示された結果のうち、Coefficients（係数） と書かれたブロックに注目する。

Estimate（推定値）の列の、(Intercept) の行にある数値が $α$ の推定値a、experince の行にある数値が $β$ の推定値 b である。

つまり、

> t valueが、**t値**（$=Estimated予測値/Std.Error標準誤差$）である

または、summary関数を使った後、係数のみ抜き出す方法もある。

coef関数で係数を出す

```
coef(fit1)[2]
```

T分布を用いて仮説検定する

仮説検定は、帰無仮説が正しいと仮定して検定する

t分布で有意水準５％を用いて検定するには、t分布の臨界値の絶対値とt統計量の絶対値とを比較する必要がある。t値はsummary関数で出てくる。

臨界値の絶対値　＜　t統計量の絶対値　の場合、帰無仮説を棄却する

サンプルサイズを調べよう

t分布を決定するのは**自由度**

自由度を求めるためには、

> $自由度 = サンプルサイズN - 説明変数の数K - 1$

の式に代入。

```
length(fit1$fitted.values)  #fitted.valuesは欠損値を除くコード
```

N = 1261と出た場合、t分布の自由度は

> 1261 - 1 - 1 = 1259

となる。

5%水準で知りたい（臨界値を出す）

qt関数を使う

p：有意水準（今回は正負両側を検定しているので÷2）

df：自由度

lower.tail：正の値だけでよく、負の値はいらないためFALSE

```
qt(p = 0.05 / 2, df = 1259, lower.tail = FALSE)
```

その答えは、

> 1.96185

つまり、

> 臨界値は1.96

信頼区間を求める

coef関数で係数のみ出す

tidy関数内につっこんでしまう

[2]でタブ experience を指定

experience(今回検定している$β_1$)の信頼区間を求めるには、式変形して$<B_1<$にしたいので、

$β_1の係数 \pm 臨界値\times 標準誤差$

```
coef(fit1)[2] - 1.96 * 0.7891 # 95%信頼区間の下限
coef(fit1)[2] + 1.96 * 0.7891 # 95%信頼区間の上限
```

○　さらに簡単に出すには

```
tidy(fit1, conf.int = TRUE)
```

○　99%信頼区間に広げる

＝精度を下げる

```
tidy(fit1, conf.int = TRUE, conf.level = 0.99)
```

検定結果

> **臨界値の絶対値　＜　t統計量の絶対値　の場合、帰無仮説を棄却する**

臨界値の絶対値：1.96

t統計量の絶対値：summary関数より28.93

よって、帰無仮説を棄却する

つまり、統計的に有意である。

単回帰の例２

> 「選挙費用 [単位：百万円] (expm;) が得票率 (voteshare;) に影響する」という仮説を検証する

帰無仮説は、$β_1≠0$ であること

lm関数で回帰する

```
fit2 <- lm(voteshare ~ expm, data = hr1996)
tidy(fit2)
```

結果

```
# A tibble: 2 × 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)     7.44    0.665       11.2 9.82e- 28
2 expm            1.88    0.0609      30.8 3.80e-154
```

T分布を用いて仮説検定する

仮説検定は、帰無仮説が正しいと仮定して検定する

t分布で有意水準５％を用いて検定するには、t分布の臨界値の絶対値とt統計量の絶対値とを比較する必要がある。t値はsummary関数で出てくる。

> **臨界値の絶対値　＜　t統計量の絶対値　の場合、帰無仮説を棄却する**

サンプルサイズを調べよう

t分布を決定するのは**自由度**

自由度を求めるためには、

> $自由度 = サンプルサイズN - 説明変数の数K - 1$

の式に代入。

```
length(fit2$fitted.values)
```

これで **N = 1124** と出る

5%水準で知りたい

qt関数を使う

p：有意水準（今回は正負両側を検定しているので÷2）

> 7%有意水準に上げる　＝　pの値を7に大きくして÷2する

df：自由度

lower.tail：正の値だけでよく、負の値はいらないためFALSE

```
qt(p = 0.07 / 2, df = 1998 - 2, lower.tail = FALSE)
```

臨界値は1.81

95%信頼区間を出す

coef関数で係数のみ出す

[2]でタブ experience を指定

experience(今回検定している$β_1$)の信頼区間を求めるには、式変形して$<B_1<$にしたいので、

$β_1の係数 \pm 臨界値\times 標準誤差$

```
1.88 - 1.81 * 0.0609 # 95%信頼区間の下限
1.88 + 1.81 * 0.0609 # 95%信頼区間の上限
#または
tidy(fit2, conf.int = TRUE, conf.level = 0.93)
```

結果

```
# A tibble: 2 × 7
  term        estimate std.error statistic   p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
1 (Intercept)     7.44    0.665       11.2 9.82e- 28     6.24      8.65
2 expm            1.88    0.0609      30.8 3.80e-154     1.77      1.99
```

検定結果

> **臨界値の絶対値　＜　t統計量の絶対値　の場合、帰無仮説を棄却する**

臨界値の絶対値：1.81

t統計量の絶対値：summary関数より30.87

よって、帰無仮説を棄却する

つまり、統計的に有意である。

重回帰の例

> 過去の議員経験（experience; ）と選挙費用 [単位：百万円] (expm;) が 得票率 (voteshare;) に影響する」という仮説を検証

lm関数で回帰

```
fit3 <- lm(voteshare ~ experience + expm, data = hr1996)
# 結果を出す
summary(fit3)
```

結果

```
Call:
lm(formula = voteshare ~ experience + expm, data = hr1996)

Residuals:
    Min      1Q  Median      3Q     Max
-31.919  -7.419  -0.936   6.088  53.340

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.77407    0.59742   13.01   <2e-16 ***
experience  13.61318    0.80134   16.99   <2e-16 ***
expm         1.31223    0.06396   20.52   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.34 on 1195 degrees of freedom
  (63 observations deleted due to missingness)
Multiple R-squared:  0.5513,    Adjusted R-squared:  0.5506
F-statistic: 734.2 on 2 and 1195 DF,  p-value: < 2.2e-16
```

重回帰分析は t検定ではなく f検定

◆ 包括的仮説検定（結合仮説の検定）でおこなうと...

重回帰分析は、F統計量を用いる

F分布から1000個くらい抽出してヒストグラムへ

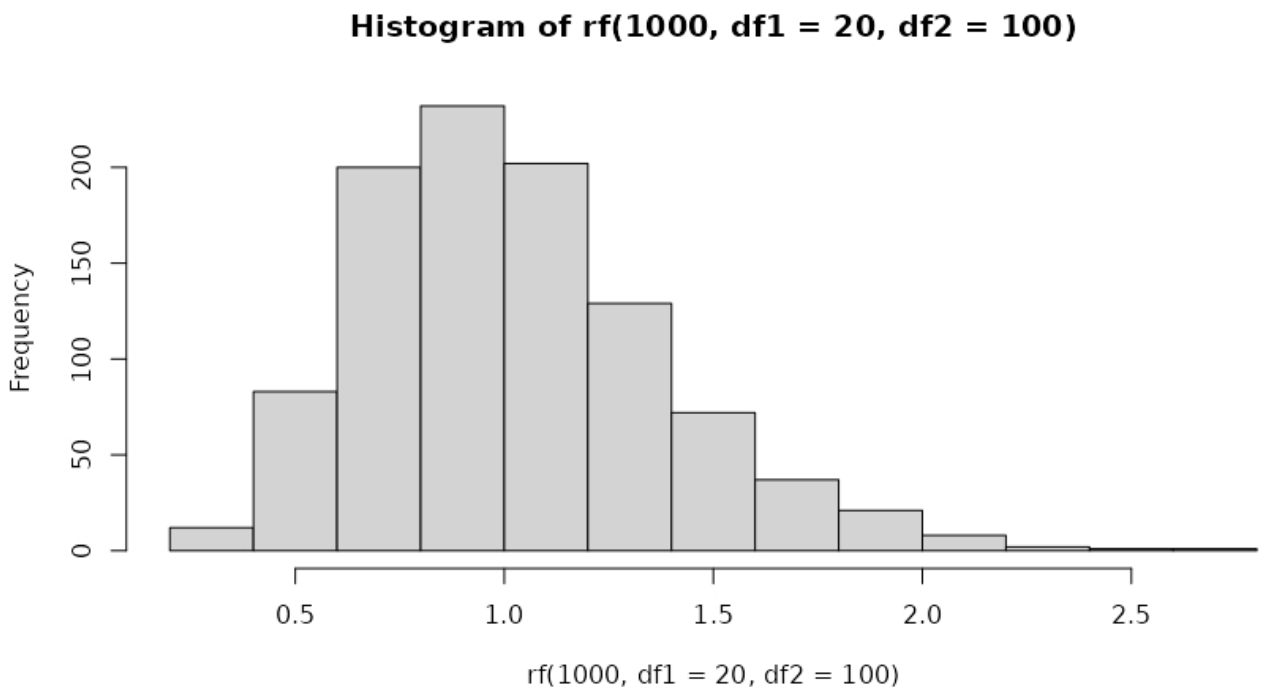
f値はsummary関数のF-satistic で出る

```
F-statistic: 734.2
```

もしくはrf関数で、f値を出すこともできる

今回は、f値をそのままヒストグラムへいれたいので、rfでだし、そのままヒストグラムと接続する

```
rf(1000, df1 = 20, df2 = 100) |> hist()
```



サンプルサイズを出す

自由度を求めるためには、

> $自由度 = サンプルサイズN - 説明変数の数K - 1$

の式に代入。

```
length(fit3$fitted.values)
 df1 = 2, df2 = 1198 - 2 - 1
```

5%水準で知りたい

qf関数を使ってF値の臨界値を出す

p：有意水準　F分布は正の値しかとらない

df：f値の自由度

lower.tail：F分布は正の値しかとらない

```
qf(p = 0.05, df1 = 2, df2 = 1195, lower.tail = FALSE)
```

答えは

```
[1] 3.003255
```

と出る。

検定結果

> 臨界値の絶対値　＜　f統計量の絶対値　の場合、帰無仮説を棄却する

臨界値の絶対値：3.00

f統計量の絶対値：上述のrf関数より734.2

よって、帰無仮説を棄却する

つまり、**統計的に有意**である。

◆ 個別的仮説検定で行うと…

tidy関数を使った省略Ver.で書くと（無省略版は単回帰参照）

```
tidy(fit3, conf.int = TRUE)
```

答えが

```
# A tibble: 3 × 7
  term        estimate std.error statistic  p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)     7.77    0.597       13.0 2.67e-36     6.60      8.95
2 experience     13.6     0.801       17.0 3.83e-58    12.0      15.2
3 expm            1.31    0.0640      20.5 2.22e-80     1.19      1.44
```

と出てくる。

t値が、上の表のstatistic の列に表示されている。この値をt分布の臨界値と比較する。

5%水準で知りたい

qt関数を使う

p：有意水準（今回は正負両側を検定しているので÷2）

df：自由度

lower.tail：正の値だけでよく、負の値はいらないためFALSE

```
qt(p = 0.05 / 2, df = 1195, lower.tail = FALSE)
```

臨界値は、1.961951になる

検定結果

> 臨界値の絶対値　＜　t統計量の絶対値　の場合、帰無仮説を棄却する

臨界値の絶対値：1.96

f統計量の絶対値：tidy関数より17.0

よって、帰無仮説を棄却する

つまり、**統計的に有意**である。

ここからわかること

ここから、議員経験と選挙費用はどちらも得票率に影響すると考える。

$b_1(今回はexperience)≈13.6$ なので、**選挙費用が同じ候補者同士を比べると、議員経験がある者のほうが経験がない者よりも平均して13.6ポイント得票率が高い**と予測できる。

同様に、$b_2(今回はexmp)≈1.3$なので、議員経験の有無が同じ場合には、**選挙費用を100万円増やすごとに平均すると1.3ポイント得票が増える**ことが予測される。

シミュレーションで回帰分析を理解する

データの準備

パラメータ設定

```
beta1 <- 10    # 切片の値を決める
beta2 <- 3     # 傾き（予測変数の係数）を決める
sigma <- 6     # 誤差の標準偏差を決める
N <- 100    ## サンプルサイズを決める
```

x ~ U(0, 10) とする：他の設定でもかまわない

```
x <- runif(N, min = 0, max = 10)    ## 最小値0, 最大値10の一様分布から長さNのxベクトルを抽出する
y <- rnorm(N, beta1 + beta2 * x, sd = sigma)  ## 設定に従って、yを正規分布から抽出する
df <- tibble(x,y)
df
```

図を描く

```
df |>
  ggplot(aes(x = x, y = y))+
  geom_point(size = 1.5)+
  geom_smooth(method = "lm") +
  theme_bw()
```

```r
egl <- lm(y ~ x, data = df)
  tidy(egl)

 confint(egl) #95%信頼区間のみ計算
```

```r
sim_ols1 <- function(beta, sigma, n = 100, trials = 10000, x_rng = c(0,10)) {
  ## 単回帰をシミュレートする関数
  ## 引数
  ##   beta: 係数パラメタのベクトル
  ##   sigma: 誤差の標準偏差
  ##   n: 標本サイズ、既定値は100
  ##   trials: シミュレーションの繰り返し回数、既定値は10000
  ##   x_rng: 説明変数xの値の範囲、既定値は (0, 10)
  ## 返り値:
  ##   df: 以下の列を含むデータフレーム
  ##      (1) パラメタの推定値
  ##      (2) 各パラメタの推定値の標準誤差
  ##      (3) 各パラメタの95％信頼区間

  ## 結果を保存するためのデータフレームを作る
  col_names <- c('b1', 'b1_se', 'b1_lower', 'b1_upper',
                 'b2', 'b2_se', 'b2_lower', 'b2_upper', 'sigma_hat')
  df <- as.data.frame(matrix(rep(NA, trials * length(col_names)),
                             ncol = length(col_names)))
  names(df) <- col_names

  for (i in 1:trials) { # ループ
    # データ生成
    x <- runif(n, x_rng[1], x_rng[2])
    y <- rnorm(n, mean = beta[1] + beta[2]*x, sd = sigma)

    # y を x に回帰する
    fit <- lm(y ~ x)

    # 残差平方和
    sigma_hat <- summary(fit)$sigma

    # 係数の推定値
    b1 <- coef(fit)[1]
    b2 <- coef(fit)[2]

    # 推定の標準誤差
    b1_se <- sqrt(summary(fit)$cov.unscaled[1,1]) * sigma_hat
    b2_se <- sqrt(summary(fit)$cov.unscaled[2,2]) * sigma_hat

    # 95% CI
    b1_ci95 <- confint(fit)[1,]
    b2_ci95 <- confint(fit)[2,]

    # 結果をまとめる
```

```
      df[i,] <- c(b1, b1_se, b1_ci95,
                  b2, b2_se, b2_ci95,
                  sigma_hat)
    }

    return(df)
  }

simdf <- sim_ols1(beta = c(10, 3), sigma = 6, trials = 10000)
simdf
```

重回帰分析

lm() で重回帰を行うときは、説明変数を+でつなぐ

hr09の表は抽出したうえで事前に準備している前提で進める。今回は、

回帰して信頼区間を出す

```
fit1 <- lm(voteshare ~ experience + expm , data =hr09)
tidy(fit1, conf.int = TRUE)
```

結果

```
# A tibble: 3 × 7
  term        estimate std.error statistic   p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
1 (Intercept)     8.18    0.261       31.3 6.21e-201     7.67      8.69
2 experience     15.3     0.375       40.8 0            14.5      16.0
3 expm            1.60    0.0334      47.9 0             1.54      1.67
```

実際の予測値を出す

議員経験の係数を出す

experienceの値が変わることによって、voteshareがどれくらい変化するかを調べたい。

つまり

> pred0（experienceが0の時の予測値）とpred1（experienceが1の時の予測値）を計算し、それぞれのmoney（0〜25の範囲における値）に対する予測結果を比較

これにより、experience（経験年数）の値が変化することによって、voteshare（投票シェア）がどのように変化するのかがわかる

まずはpred0（experienceが0の時の予測値）を出す

```
money <- 0:25  #0~25の値をとるmoneyを定義
pred0 <- coef(fit1)[1] +
  coef(fit1)[2] * 0 +
  coef(fit1)[3] * money
```

coef(fit1)[1]は切片

coef(fit1)[2]はexperienceの係数

coef(fit1)[3]はexpmの係数

moneyとpred0の表を作る

```
money_df <- tibble(money, pred0)
```

次にpred1（experienceが1の時の予測値）を出す

```
pred1 <- coef(fit1)[1] +
  coef(fit1)[2] * 1 +
  coef(fit1)[3] * money
```

これを表に格納

```
money_df <- money_df |>
  mutate(pred1 = pred1 )
```

pre1 - pre 0をして変化量を出す

```
money_df$pred1 - money_df$pred0
```

回帰解剖

voteshareとexperienceの関係について単回帰分析

```
reg1 <- lm(voteshare ~ experience, data = hr09)
tidy(reg1)
```

experienceを独立変数としてexpmに対する回帰分析

expmがexperienceの関数であるかどうかを調べるため

```
reg2 <- lm(expm ~ experience, data = hr09)
```

残差を求める

```
res1 <- reg1$restrials
res2 <- reg2$residuals
```

残差の差を求める

```
res3 <- lm(res1 - res2)
tidy(reg3)
```

重回帰分析の図示しよう(もっと簡単な方法は下にある)

議員経験を${0, 1}$のいずれかとして、選挙費用 [100万円] を最小値から最大値まで動かし、それぞれの組み合わせで$hatV_i$を計算したい。そのために、まずは2つの変数の値の組み合わせ考慮するためのデータフレームを作る。2つ以上の変数のすべての組み合わせを作るために、tidyr::expand_grid() を利用する。

```
pred <- expand_grid(
  expm = seq(from = min(hr09$expm, na.rm = TRUE),
             to   = max(hr09$expm, na.rm = TRUE),
             length.out = 100),
  experience = c(0, 1)
)
print(pred, n = Inf)
```

これを利用して予測値を計算する。予測値は、predict() で求めることができる

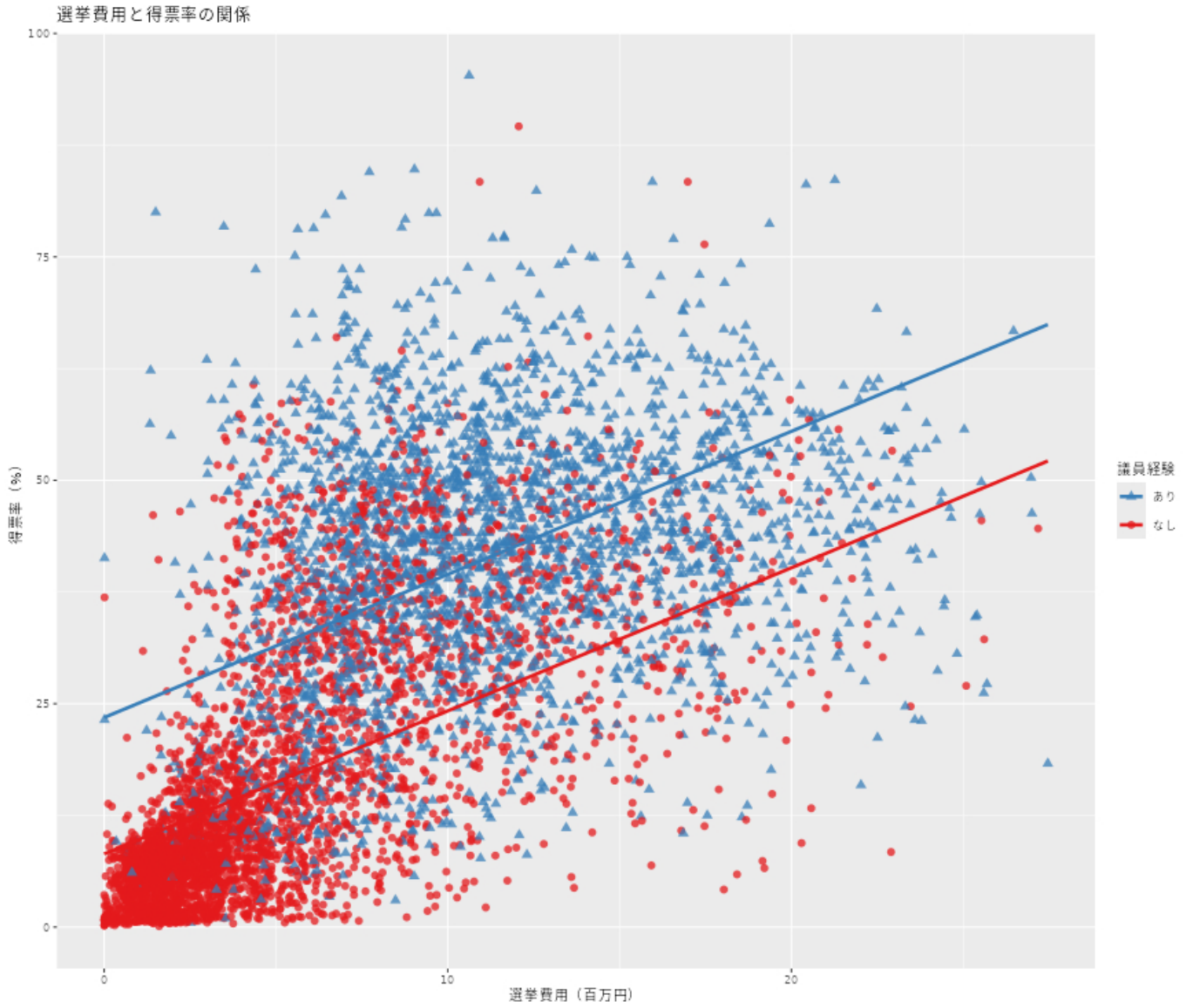
```
pred <- pred |>
  mutate(v_hat = predict(fit1, newdata = pred))
```

結果を図示する

散布図の点 （観測値） は元データ hr09 で描き、回帰直線は予測値 pred で描く

```
hr09 |>
  ggplot(aes(x = expm, y = voteshare, #図を作る
             color = factor(experience),
```

```
            shape = factor(experience))) +
  geom_point(size = 2, alpha = 0.75) +  #散布図を描く
  geom_line(data = pred,  #predから回帰直線を描く
            aes(y = v_hat),
            linewidth = 1) +
  guides(color = guide_legend(reverse = TRUE),
         shape = guide_legend(reverse = TRUE)) +
  scale_color_brewer(palette = "Set1",   #凡例を描く
                     name    = "議員経験",
                     labels  = c("なし", "あり")) +
  scale_shape_discrete(name   = "議員経験",
                       labels = c("なし", "あり")) +
  labs(x = "選挙費用（百万円）", y = "得票率（%）",
       title = "選挙費用と得票率の関係")
```



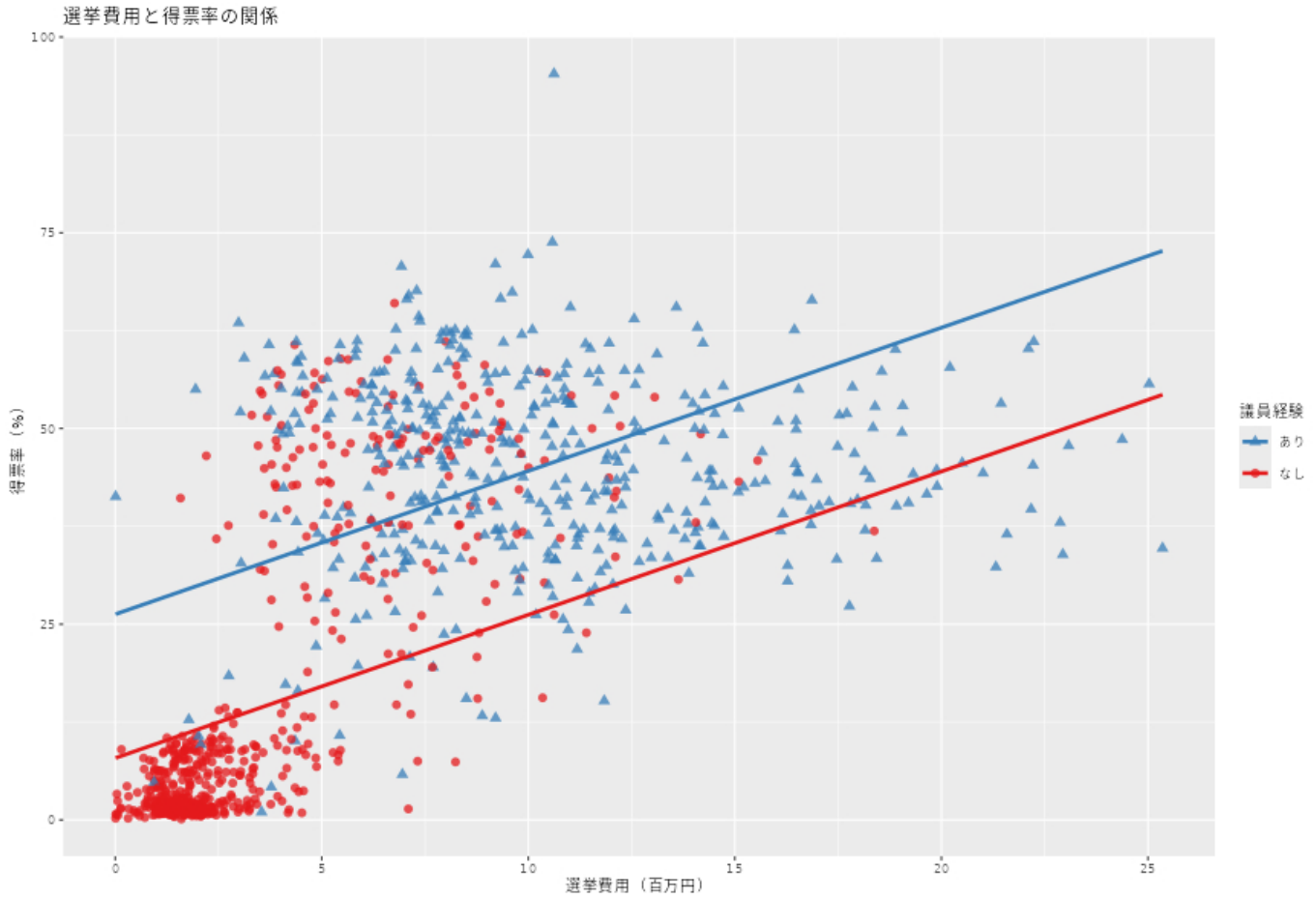
この図に95%推定区間を加える

```
pred_se <- predict(fit1, newdata = pred, se.fit = TRUE)
pred <- pred |>
  mutate(se = pred_se$se.fit)
```

```
pred

pred <- pred |>
  mutate(lower = v_hat + qt(0.025, df = 200) * se, #
         upper = v_hat + qt(0.975, df = 200) * se)
```

図を描く

```
hr09 |>
  ggplot(aes(x = expm, y = voteshare,
             color = factor(experience),
             shape = factor(experience))) +
  geom_point(size = 2, alpha = 0.75) +
  geom_smooth(data = pred,
              aes(y = v_hat,
                  ymin = lower, ymax = upper),
              stat = "identity") +
  guides(color = guide_legend(reverse = TRUE),
         shape = guide_legend(reverse = TRUE)) +
  scale_color_brewer(palette = "Set1",
                     name    = "議員経験",
                     labels  = c("なし", "あり")) +
  scale_shape_discrete(name   = "議員経験",
                       labels = c("なし", "あり")) +
  labs(x = "選挙費用（百万円）", y = "得票率（%）",
       title = "選挙費用と得票率の関係") +
  theme_bw()
```



予測値と信頼区間の簡単な計算方法（おまけ）

```
pacman::p_load(marginaleffects)

fit1 |>
  predictions(newdata = datagrid(experience = c(0, 1),
                                 expm       = 0:25)) |>
  ggplot(aes(x = expm,
             fill = factor(experience),
             color = factor(experience))) +
  geom_point(data = hr09,
             aes(y = voteshare)) +
  geom_ribbon(aes(x = expm, ymin = conf.low,
                  ymax = conf.high,
                  fill = factor(experience)),
              alpha = 0.35) +
  geom_line(aes(x = expm, y = estimate,
                color = factor(experience)),
            linewidth = 1.5)
```

欠落変数バイアスと処置後変数バイアス

欠落変数バイアスのシミュレーション

```
# シミュレーションを再度実行したときに同じ結果が得られるように、乱数の種を指定する
# 異なる結果を得たいときは、set.seed() の中身を変える
set.seed(777)
N <- 100
z1 <- runif(N, -5, 5)
z2 <- runif(N, -5, 5)
z3 <- runif(N, -5, 5)
x <- 0.2 * z1 + rnorm(N)
mu <- 1.5 + 0.4 * x + 0.5 * z1 - 0.2 * z2
y <- rnorm(N, mean = mu, sd = 1)
df <- tibble(y, x, z1, z2, z3)

true_model <- lm(y ~ x + z1 + z2, data = df)
tidy(true_model)


omit_model1 <- lm(y ~ x + z1 , data = df)
omit_model2 <- lm(y ~ x + z2 , data = df)
extra_model <- lm(y ~ x + z1 + z2 + z3 , data = df)

tidy(omit_model1) #DGPにはあるが交絡でないZ2を省略
tidy(omit_model2) #交絡であるz1を省略
tidy(extra_model) #DGPにはないz3を追加
```

処置後変数バイアス

```
true_effect <- .25

ptb_dgp <- function(n = 100) {
  # 処置後変数バイアス問題をシミュレートする関数
  x <- rbinom(n, size = 1, prob = .5)
  z <- 0.3 + 0.2 * x + rnorm(n, mean = 0, sd = 0.1)
  y <- 0.2 + (true_effect / 0.2) * z  + rnorm(n, mean = 0, sd = 0.1)
  return(tibble(y, x, z))
}


set.seed(1986-10-08)
df <- ptb_dgp()
glimpse(df)

fit_with <- lm(y ~ x + z, data = df) #媒介変数あり
fit_wo <- lm(y ~ x, data = df) #媒介変数なし

tidy(fit_with)
tidy(fit_wo)
```

変数変換

線形変換

通常通りに回帰すると

コード

```
fit1 <- lm(voteshare ~ exp, data = hr09)
tidy(fit1, conf.int = TRUE)
```

結果

```
# A tibble: 2 × 7
  term          estimate    std.error statistic   p.value   conf.low  conf.high
  <chr>            <dbl>        <dbl>     <dbl>     <dbl>      <dbl>      <dbl>
1 (Intercept) 8.65       0.291             29.7 5.76e-182 8.07       9.22
2 exp         0.00000234 0.0000000312      75.1 0         0.00000228 0.00000241
```

線形変換すると

コード

```
fit2 <- lm(voteshare ~ expm, data = hr09)
tidy(fit2, conf.int = TRUE)
```

結果

```
  term        estimate std.error statistic   p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
1 (Intercept)     8.65    0.291       29.7 5.76e-182     8.07      9.22
2 expm            2.34    0.0312      75.1 0             2.28      2.41
```

0.00000234　→ 2.34　にできる

t統計量などは線形変換しても変わらない

標準化

標準化は、平均を0, 分散を1にする。

データがきれいにする

標準化の公式

> $$z(x)= \frac{x-xの平均値}{xの不変分散の平方根}$$

標準化するプロセス

この公式に代入する

```
hr09 <- hr09 |>  #公式にいれる  #公式のz(x)はz_expm
  mutate(z_expm = (expm - mean(expm, na.rm = TRUE)) / sd(expm, na.rm = TRUE))
  #na.rm = TRUEは欠損値消去
```

平均はmean関数, 不偏分散の平方根はsd関数

```
mean(hr09$expm) # 6.118181
sd(hr09$expm)   # 4.997211
```

標準化後

数値はこのようになる

標準化は、平均を0, 分散を1にする。

```
mean(hr09$z_expm) # 0
sd(hr09$z_expm)   # 1
 # きれいになっていることがわかる
```

標準化して回帰分析すると

```
fit3 <- lm(voteshare ~ z_expm, data = hr09)
tidy(fit3)
```

結果

```
# A tibble: 2 × 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)     26.3     0.171     154.        0
2 z_expm          12.9     0.171      75.1       0
```

中心化

中心化は、回帰式の切片に意味を持たせるためにやる

> $z(x)= x-xの平均値$

公式に代入

```
hr09 <- hr09 |>
  mutate(c_expm = expm - mean(expm, na.rm = TRUE))
```

この結果

```
mean(hr09$c_expm)  #3.167597e-16
sd(hr09$c_expm) #5.489095
```

中心化したものを回帰する

```
fit4 <- lm(voteshare ~ c_expm, data = hr09)
tidy(fit4)
```

対数変換

lm関数で回帰分析する際に、logをつける

事前に変数を変換してもよいが、分析と同時に変換することもできる

```
fit5 <- lm(log(voteshare) ~ expm, data = hr09)
tidy(fit5)
```

結果

```
# A tibble: 2 × 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)    1.17    0.0521       22.5 3.90e- 93
2 expm           0.217   0.00660      32.9 5.36e-167
```

元の測定単位に戻したい

exp() で計算する

exp()がeの何乗かを示す $$ \exp(n) = e^n $$

```
exp(0.217) - exp(0)    #0.2423441     #exp(0)=log0=1
```

二次式における回帰直線の推定

この例では、ミンサー方程式を用いている

> $$log(賃金i) =β_0+β_1就業年数_i+β_2就業経験_i+β_3就業経験^2_i+ε_i$$

この回帰分析における応答変数は所得 income （万円）の自然対数、説明変数は修学年数 education、就業経験年数 experience と就業経験年数の二乗である。

データの読み込み

```
myd <- read_csv("data/fake_income.csv")
glimpse(myd)
```

推定

まず、lm() を使って推定を行う。

推定式の中で自然対数を計算するために $log()$ を、二乗の計算を行うために、二乗した式を $I()$ で囲む。

多項式は、poly方程式で処理する。

まずは公式に代入

```
# fit_mincer <- lm(log(income) ~ education + poly(experience, 2), data = myd)
fit_mincer <- lm(log(income) ~ education + experience + I(experience^2),
                 data = myd)

tidy(fit_mincer, conf.int = TRUE)
```

結果

```
# A tibble: 4 × 7
  term            estimate std.error statistic  p.value conf.low conf.high
  <chr>              <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)      2.07     0.215         9.60 6.09e-21  1.65      2.49
2 education        0.138    0.0143        9.61 5.61e-21  0.109     0.166
3 experience       0.202    0.0162       12.5  2.92e-33  0.170     0.233
4 I(experience^2) -0.00637  0.000683     -9.34 6.32e-20 -0.00771  -0.00503
```

ここから、応答変数が自然対数になっていて、修学年数 (education) の係数の推定値が 0.14なので、教育の収益率は14%ほどであることがわかるが、**就業経験の効果は、修学年数の影響とは異なり、わかりにくい**

なぜなら、**就業経験年数を動かすと、就業経験年数の二乗も一緒に動いてしまう**から

そこで、

修学年数を固定し、就業経験年数 と 就業経験年数の二乗 を動かす

修学年数を固定する

```
#mean_educを平均値にすることで固定する
mean_educ <- mean(myd$education) # 13.832
```

ベクトルにして一番下から上まで等差数列化

exp_vec = 就業経験年数

```
exp_vec <- seq(min(myd$experience),
               max(myd$experience),
               length.out = 100) #長さを100で指定
```

ベクトルの一番下から上までを重回帰モデル（＝ミンサー方程式）へ入れる

```
pred_mean <- coef(fit_mincer)[1] +
  coef(fit_mincer)[2] * mean_educ +
  coef(fit_mincer)[3] * exp_vec +
  coef(fit_mincer)[4] * exp_vec^2
```

ベクトルをデータフレームへ変換

```
pred_df <- tibble(exp = exp_vec,
                  pred_mean = pred_mean)
```

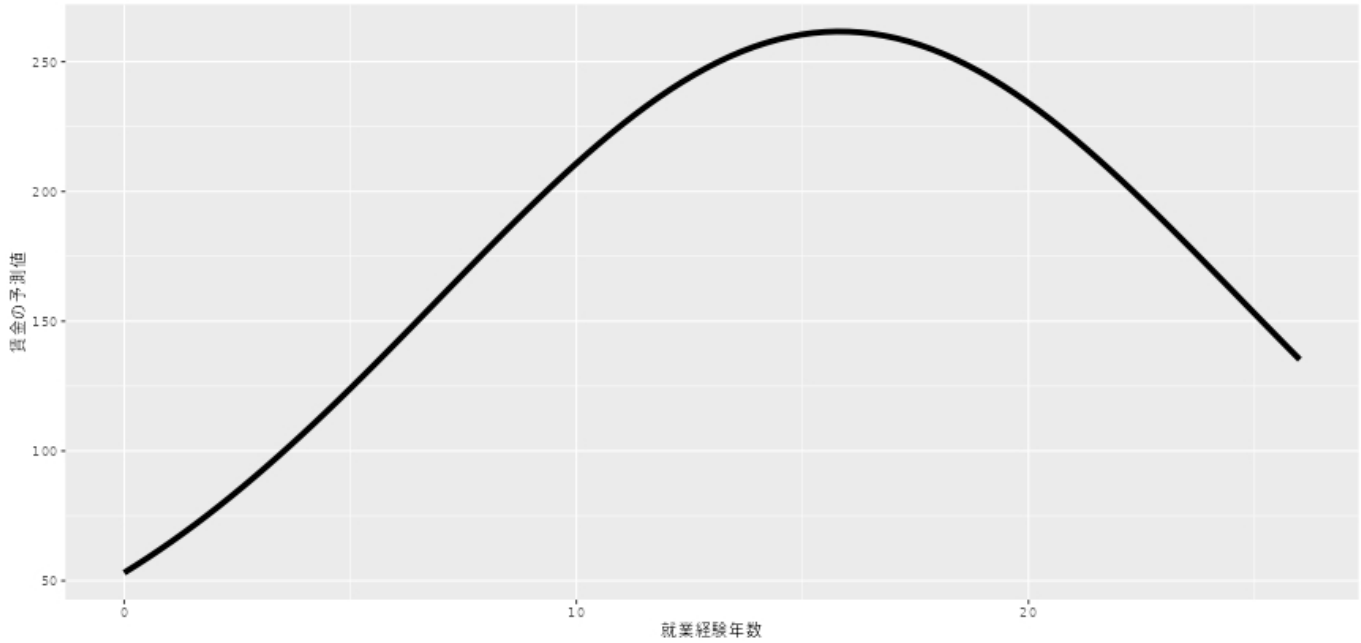
グラフにする

```
pred_df |>
  ggplot(aes(x = exp, y = pred_mean)) +
  geom_line(linewidth = 1.5) +
  labs(x = "就業経験年数", y = "log(賃金)の予測値")
```

logを消す

```
pred_df |>
  ggplot(aes(x = exp, y = exp(pred_mean))) +
  geom_line(linewidth = 1.5) +
  labs(x = "就業経験年数", y = "賃金の予測値")
```

これで賃金の予測値がとりあえずひとまず完成する。



同様に最大値と最小値を固定する方法でも行う

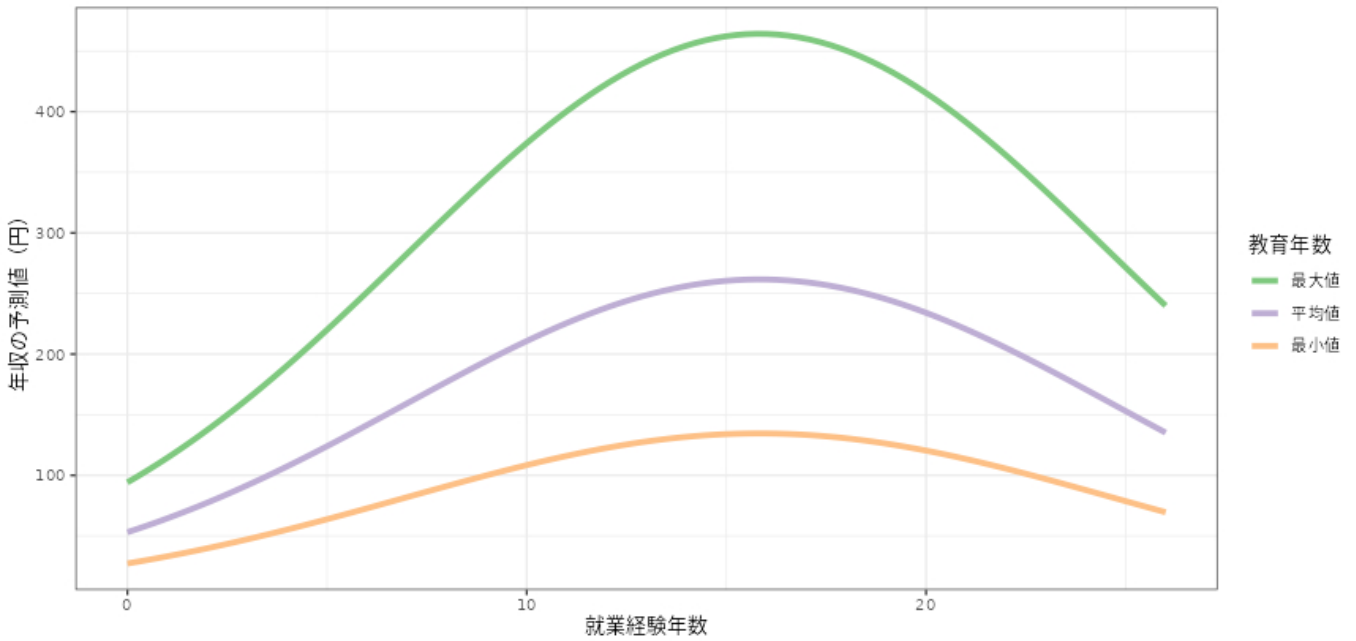
```
pred_min <- coef(fit_mincer)[1] +
  coef(fit_mincer)[2] * min(myd$education) +
  coef(fit_mincer)[3] * exp_vec +
  coef(fit_mincer)[4] * exp_vec^2

pred_max <- coef(fit_mincer)[1] +
  coef(fit_mincer)[2] * max(myd$education) +
  coef(fit_mincer)[3] * exp_vec +
  coef(fit_mincer)[4] * exp_vec^2

pred_df <- pred_df |>  #mutateで固定化した最大値と最小値を表へ追加
  mutate(pred_min = pred_min,
         pred_max = pred_max)
```

long型のデータフレームへ変換

```
pred_long_df <- pred_df |>
  pivot_longer(cols         = -exp,
               names_to     = "educ",
               values_to    = "pred",
               names_prefix = "pred_")
```

これをグラフにする

```
pred_long_df |>
  ggplot(aes(x = exp, y = exp(pred), color = educ)) +
  geom_line(linewidth = 1.5) +
  labs(x = "就業経験年数", y = "年収の予測値（円）") +
  scale_color_brewer(palette = "Accent",
                     name    = "教育年数",
                     labels  = c("最大値", "平均値", "最小値")) +
  theme_bw()
```



分析結果の提示

表をつくる

準備 データセットを作る

```
fit_1 <- lm(voteshare ~ experience, data = hr09)
fit_2 <- lm(voteshare ~ expm, data = hr09)
fit_3 <- lm(voteshare ~ experience + expm, data = hr09)
fit_4 <- lm(voteshare ~ experience * expm, data = hr09)
```

modelsummary関数で表にする

```
modelsummary(fit_1,
             title ="モデル１の推定結果",#タイトルを作る　下に出るのであとから上へ直す
             gof_map = c("nobs", "r.squared"), #いらない情報を削る
             coef_rename = c("(Intercept)" = "切片",  #ラベルを名前つけなおす
                             "experience" = "議員経験")) |>
  theme_html(class = "table caption-top")  #タイトルを上に出す
```

比較する表を作る

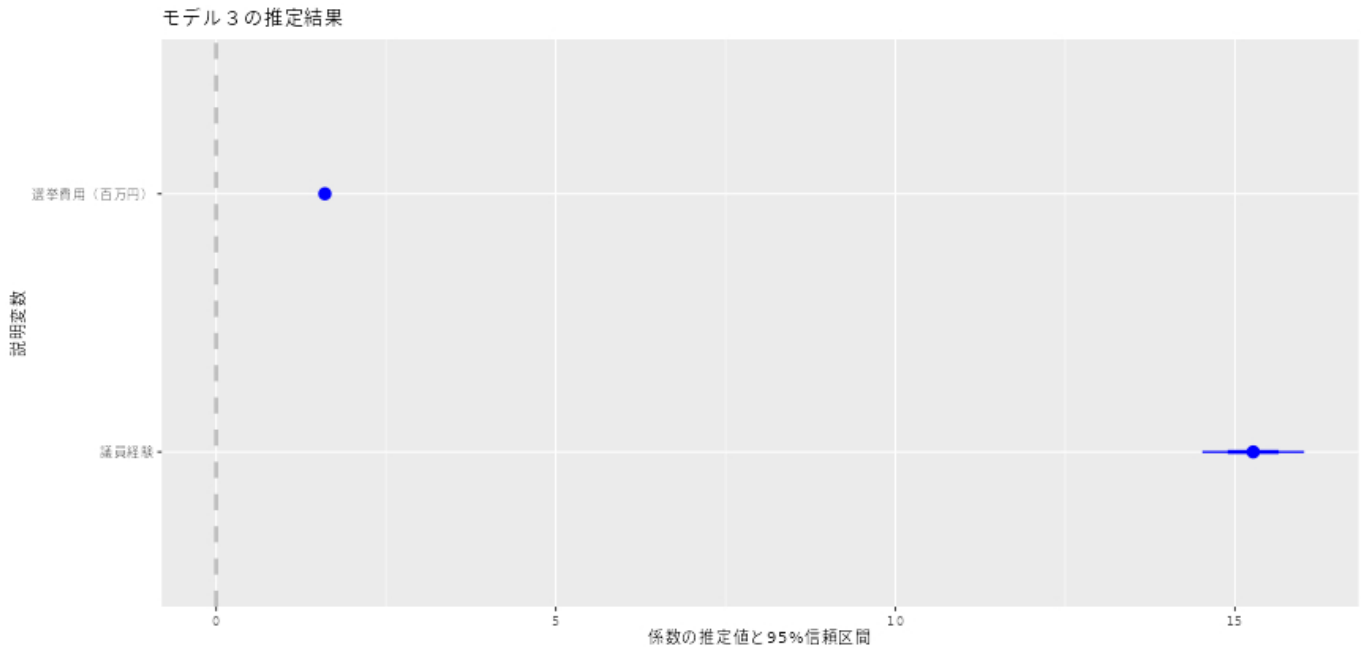
このようにして、listにしてから、modelsummaryに $|>$ で突っ込むことで、表を並べ比較する表を作ることもできる。

```
list("モデル１" = fit_1,
     "モデル2" = fit_2,
     "モデル3" = fit_3,
     "モデル4" = fit_4) |>
  msummary(title = "モデル１～４の推定結果",
           gof_map = c("nobs", "r.squard"))
```

図を作る

回帰モデルが一つのみの場合　coefplot関数

```
coefplot(fit_3,
         intercept = FALSE,  #切片はいらないので削除
         newNames = c("experience" = "議員経験",
                      "expm" = "選挙費用（百万円）"),
         title = "モデル３の推定結果",
         xlab = "係数の推定値と95%信頼区間",
         ylab = "説明変数")
```

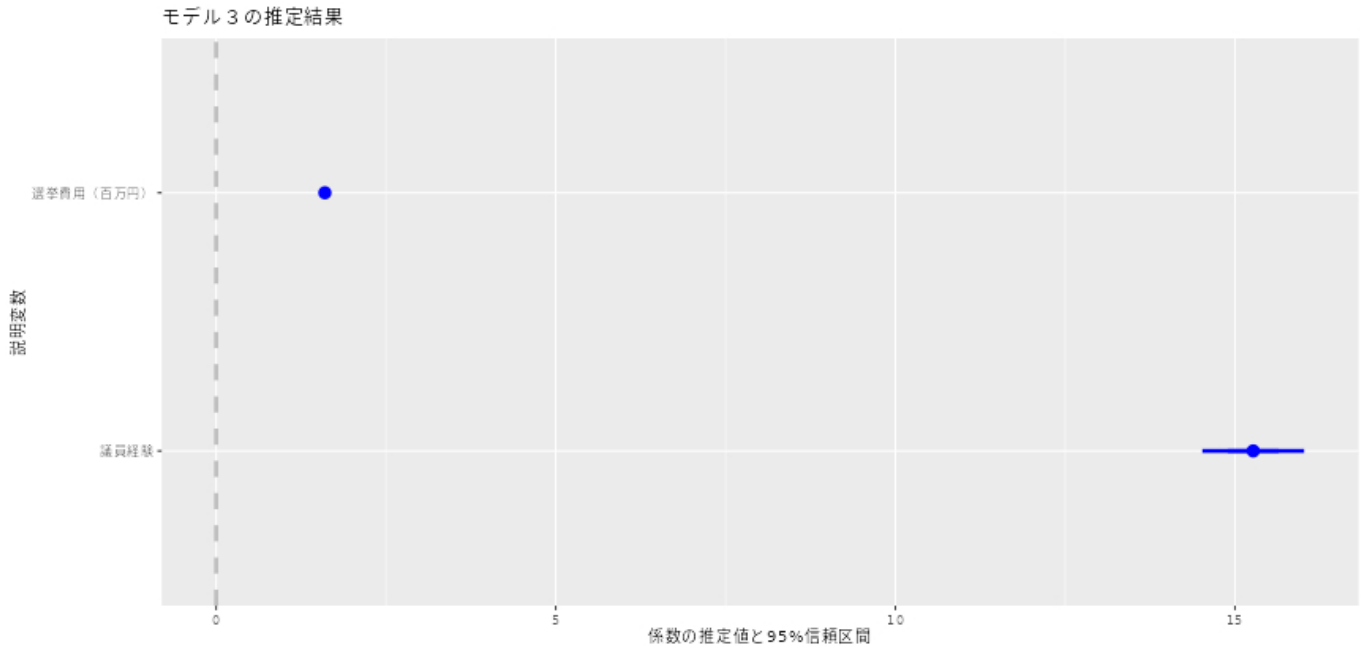


iwdOuter

上記モデルの場合、回帰係数の大きさがばらばらなので、1つの図に収めようとすると信頼区間がよくわからなくなってしまうという問題がある。

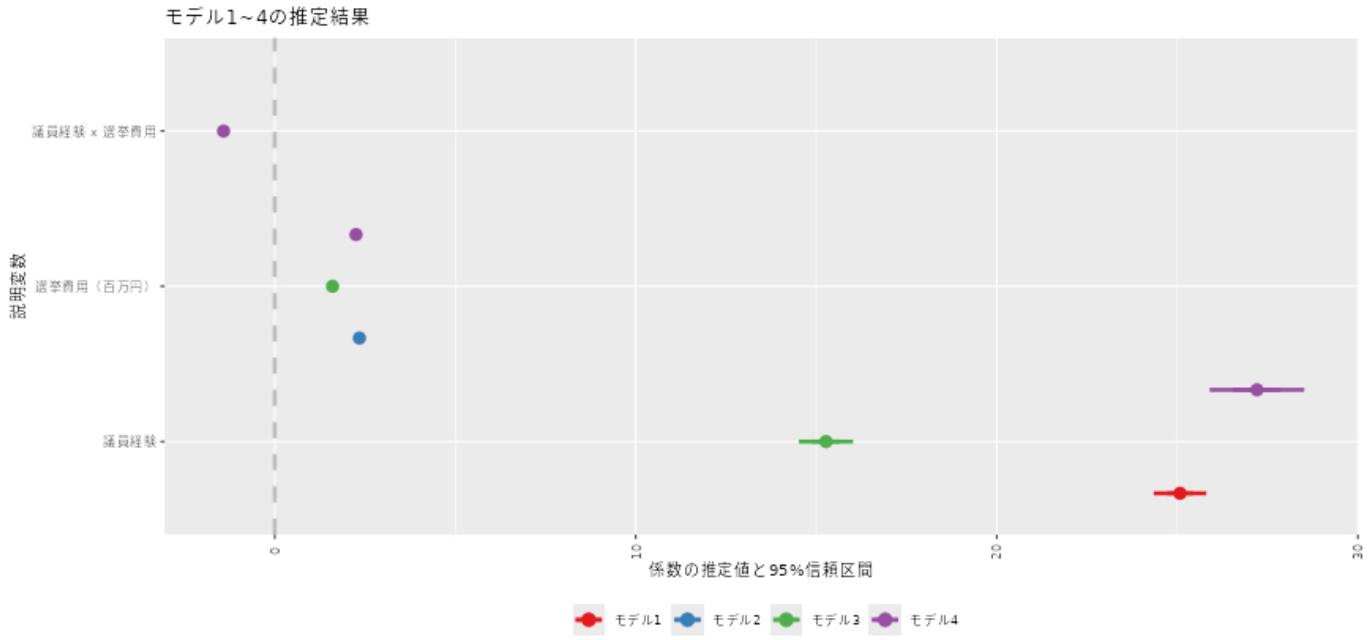
なので、95%信頼区間を表示する線分の太さをわかりやすく

```
coefplot(fit_3,
         intercept = FALSE,  #切片はいらないので削除
         lwdOuter = 1 # 95%信頼区間を表示する線分の太さ
         newNames = c("experience" = "議員経験",
                      "expm" = "選挙費用（百万円）"),
         title = "モデル３の推定結果",
         xlab = "係数の推定値と95%信頼区間",
         ylab = "説明変数")
```



回帰モデルが2つ以上の場合　multiplot関数

```
multiplot(fit_1, fit_2, fit_3, fit_4,
          intercept = FALSE,
          lwdOuter = 1,
          newNames = c("experience" = "議員経験",
                       "expm"       = "選挙費用（百万円）",
                       "experience:expm" = "議員経験 x 選挙費用"),
          title = "モデル1~4の推定結果",
          xlab = "係数の推定値と95%信頼区間",
          ylab = "説明変数") +
  scale_color_brewer(name = "",  #str_cを使うので空欄　scale_color_brewerが凡例表示
                     palette = "Set1",
                     labels = str_c("モデル", 1:4))  #モデル１～４を自動的につけてくれる関
数を使い省略
```


モデル1~4の推定結果
議員経験 × 選挙費用
選挙費用（百万円）
議員経験
説明変数
0
10
20
30
係数の推定値と95%信頼区間
モデル1
モデル2
モデル3
モデル4