Rにおける効率的な分析

石田基広

2019年09月03日

第二部

分析の効率化

- 探索的データ分析
- 機械学習
  - スライド掲載のコードと出力は一致しない場合があります

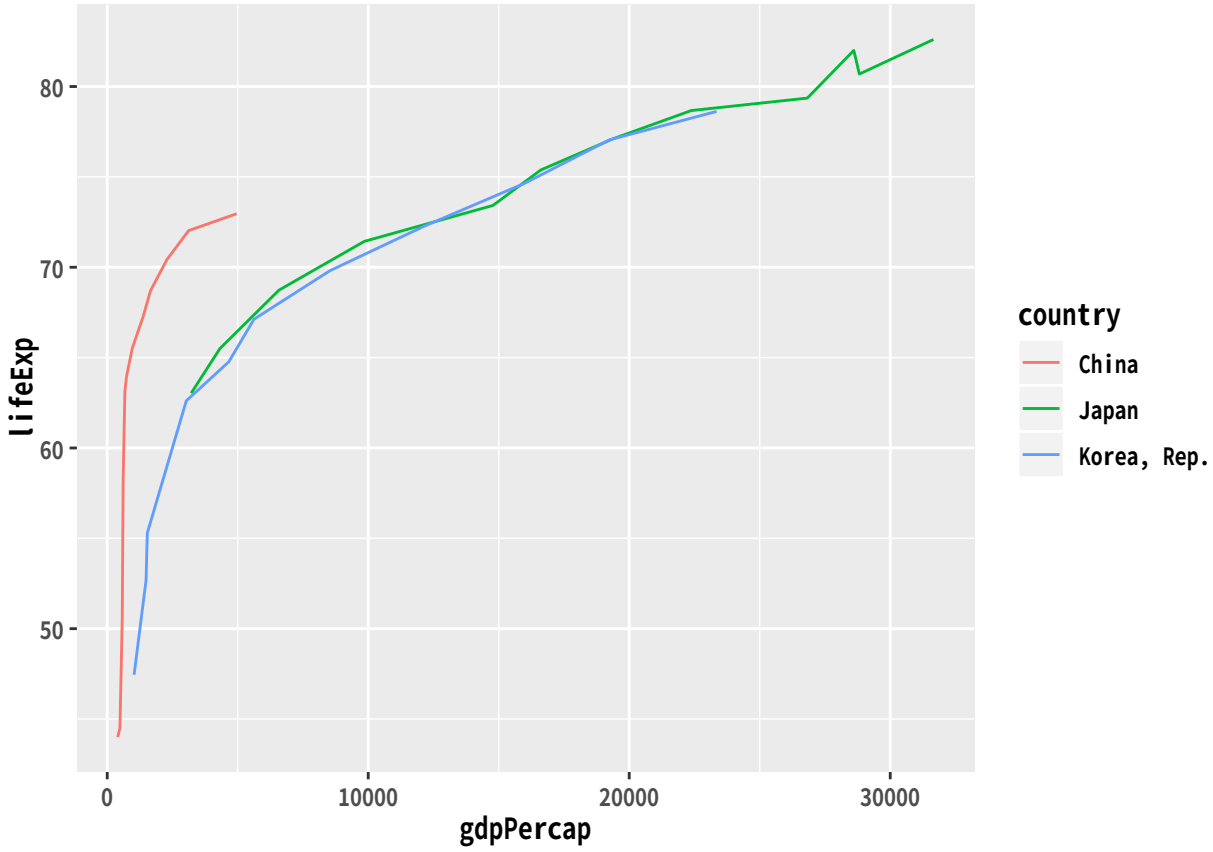
探索的データ分析

- 水準ごとに回帰分析
- 回帰分析の結果をテーブルに

gapminder

```
gap <- gapminder %>%
  filter(country %in%
  c("Japan", "China", "Korea, Rep."))
```

```
p <- gap %>%
  ggplot() +
  aes(gdpPercap,
      lifeExp,
      col = country) +
  geom_line()
```



相関係数

国を分けて相関係数

国名	相関係数
日本	cor日本X, 日本Y)
中国	cor(中国X, 中国Y)
韓国	cor(韓国X, 韓国Y)

相関係数

国を分けて相関係数

```
gap %>% filter(country=="China") %>%
  select(gdpPercap, lifeExp) %>% cor()

gap %>% filter(country=="Korea, Rep.")%>%
  select(gdpPercap, lifeExp) %>% cor()

gap %>% filter(country=="Japan") %>%
  select(gdpPercap, lifeExp) %>% cor()
```

相関の検定

中国の場合

```
gap %>% filter(country == "China") %>%
  cor.test(data=., ~lifeExp +gdpPercap)
```

```
    Pearson's product-moment correlation

data:  lifeExp and gdpPercap
t = 3.0991, df = 10, p-value = 0.01127
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2106440 0.9087826
sample estimates:
      cor
0.6999316
```

map tidy

```
gap %>%  nest(-country) %>%
  mutate(test = map(data,
          ~cor.test(data=.,
              ~ lifeExp + gdpPercap)),
          res = map(test, tidy)) %>%
              unnest(res, .drop = TRUE)
```

```
# A tibble: 3 x 9
  country estimate statistic p.value parameter conf.low conf.high method
  <fct>      <dbl>     <dbl>   <dbl>     <int>    <dbl>     <dbl> <chr>
1 China      0.700      3.10 1.13e-2        10    0.211     0.909 Pears~
2 Japan      0.979     15.3  2.95e-8        10    0.925     0.994 Pears~
3 Korea,~    0.903      6.63 5.87e-5        10    0.682     0.973 Pears~
# ... with 1 more variable: alternative <chr>
```

仕組み

nestで水準ごとに分けたdat列を生成

```
gap %>% nest(-country)
```

```
# A tibble: 3 x 2
  country     data
  <fct>       <list>
1 China       <tibble [12 x 5]>
2 Japan       <tibble [12 x 5]>
3 Korea, Rep. <tibble [12 x 5]>
```

仕組み

dataの中身

```
gap %>% nest(-country) %>%
              select(data) %>% unnest()
```

```
# A tibble: 36 x 5
   continent  year lifeExp        pop gdpPercap
   <fct>     <int>   <dbl>      <int>     <dbl>
```

```
 1 Asia       1952    44    556263527      400.
 2 Asia       1957    50.5  637408000      576.
 3 Asia       1962    44.5  665770000      488.
 4 Asia       1967    58.4  754550000      613.
 5 Asia       1972    63.1  862030000      677.
 6 Asia       1977    64.0  943455000      741.
 7 Asia       1982    65.5 1000281000      962.
 8 Asia       1987    67.3 1084035000     1379.
 9 Asia       1992    68.7 1164970000     1656.
10 Asia       1997    70.4 1230075000     2289.
# ... with 26 more rows
```

map

dataごとに平均を求めてAVG列に追加

```
gap %>%  nest(-country) %>%
 mutate(AVG=map(data,~mean(.$lifeExp)))
```

```
# A tibble: 3 x 3
  country     data              AVG
  <fct>       <list>            <list>
1 China       <tibble [12 x 5]> <dbl [1]>
2 Japan       <tibble [12 x 5]> <dbl [1]>
3 Korea, Rep. <tibble [12 x 5]> <dbl [1]>
```

仕組み

purrr::mapは複数のデータセットそれぞれに関数を適用する

```
list(x = 1:3, y=1:6, z= 1:10) %>%
  map(mean)
```

```
$x
[1] 2

$y
[1] 3.5

$z
[1] 5.5
```

無名関数

```
list(x = 1:3, y=1:6, z= 1:10) %>%
  map(mean)
list(x = 1:3, y=1:6, z= 1:10) %>%
  map(~mean(.))
```

ドットがデータを表す

```
~mean(.) -> function(.x){mean(.x)}
```

map_func

処理例：あるフォルダのCSVファイルを全部読み込んで結合する

```
dat <-list.files(pattern = "*.csv")%>%
          map_df(read_csv)
```

list.files(pattern = "*.csv"でフォルダのcsvファイルの一覧を取り

map_dfでそのリストの要素一つ一つに read_csv を適用して結合

仕組み

AVG列をunnestで展開

```
gap %>%  nest(-country) %>%
       mutate(AVG = map(data,
                ~ mean(.$lifeExp))) %>%
            unnest(AVG)
```

```
# A tibble: 3 x 3
  country     data               AVG
  <fct>       <list>           <dbl>
1 China       <tibble [12 x 5]>  61.8
2 Japan       <tibble [12 x 5]>  74.8
3 Korea, Rep. <tibble [12 x 5]>  65.0
```

回帰

```
gap %>% filter(country == "Japan") %>%
  lm(data=., lifeExp ~ gdpPercap) %>%
     summary()
```

```
Call:
lm(formula = lifeExp ~ gdpPercap, data = .)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6739 -0.9849  0.4094  1.0194  1.5639

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.368e+01  8.316e-01   76.58 3.52e-15 ***
gdpPercap   6.277e-04  4.111e-05   15.27 2.95e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.381 on 10 degrees of freedom
Multiple R-squared:  0.9589,    Adjusted R-squared:  0.9548
F-statistic: 233.1 on 1 and 10 DF,  p-value: 2.949e-08
```

回帰

出力表に変える：broom::tidy

```
gap %>% filter(country == "Japan") %>%
  lm(data=.,lifeExp ~ gdpPercap) %>%
    summary() %>% tidy()
```

```
# A tibble: 2 x 5
  term         estimate std.error statistic  p.value
  <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept) 63.7      0.832          76.6 3.52e-15
2 gdpPercap    0.000628 0.0000411      15.3 2.95e- 8
```

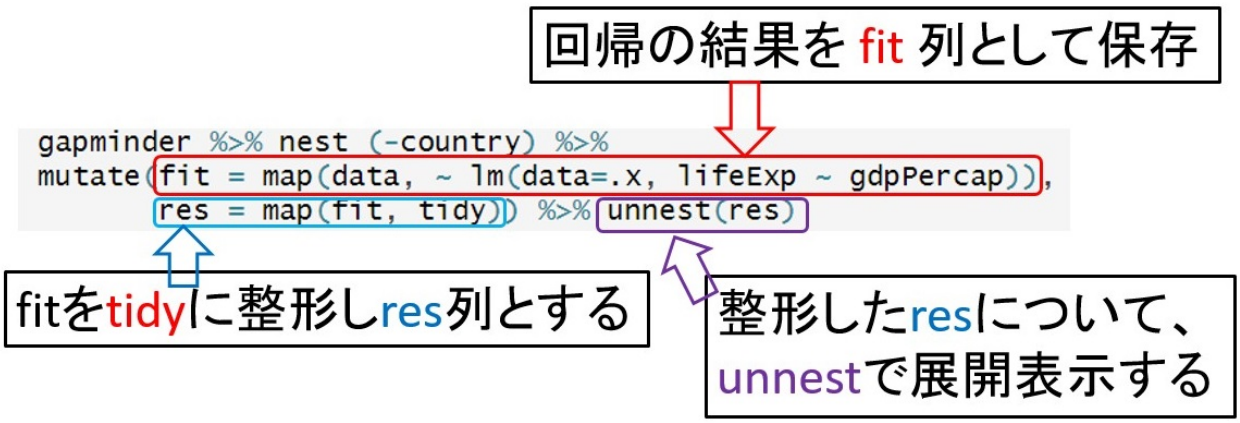
回帰の結果を整頓した形式で保存

回帰

```
gap %>% nest(-country) %>%
 mutate(fit = map(data, ~
  lm(data=., lifeExp ~ gdpPercap)),
  res = map(fit, tidy)) %>% unnest(res)
```

```
# A tibble: 6 x 6
  country     term         estimate std.error statistic  p.value
  <fct>       <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 China       (Intercept) 54.0      3.35          16.1  1.76e- 8
2 China       gdpPercap    0.00523  0.00169        3.10 1.13e- 2
3 Japan       (Intercept) 63.7      0.832         76.6  3.52e-15
4 Japan       gdpPercap    0.000628 0.0000411     15.3  2.95e- 8
5 Korea, Rep. (Intercept) 55.3      1.97          28.1  7.53e-11
6 Korea, Rep. gdpPercap    0.00118  0.000178       6.63 5.87e- 5
```

コードの意味



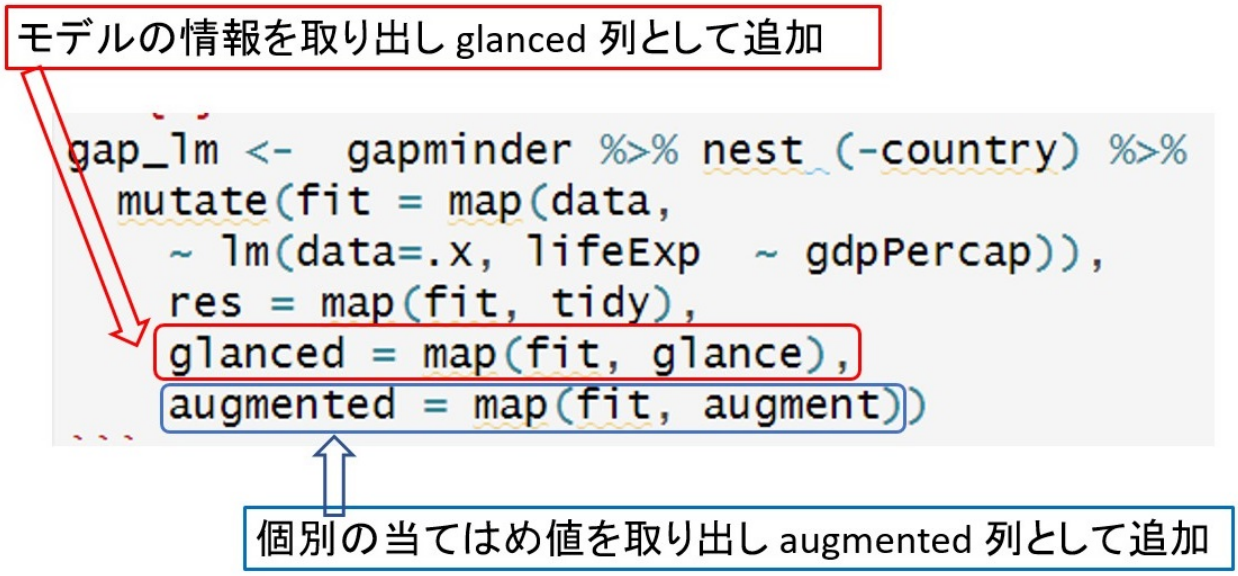
要約

```
gap %>% nest(-country) %>%
  mutate(fit = map(data,
    ~ lm(data=., lifeExp ~ gdpPercap)),
      res = map(fit, tidy),
    glanced = map(fit, glance),
    augmented = map(fit, augment))
```

```
# A tibble: 3 x 6
  country    data          fit    res          glanced       augmented
  <fct>      <list>        <list> <list>       <list>        <list>
1 China      <tibble [12 ~ <lm>   <tibble [2 ~ <tibble [1 x~ <tibble [12 x~
```

```
2 Japan      <tibble [12 ~ <lm>   <tibble [2 ~ <tibble [1 x~ <tibble [12 x~
3 Korea, Re~ <tibble [12 ~ <lm>   <tibble [2 ~ <tibble [1 x~ <tibble [12 x~
```

コードの意味



glance

モデルの詳細情報

```
gap_lm %>% unnest(glanced,.drop = TRUE)
```

```
# A tibble: 3 x 12
  country r.squared adj.r.squared sigma statistic p.value    df logLik
  <fct>       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl>
1 China       0.490         0.439  7.68      9.60 1.13e-2     2  -40.4
2 Japan       0.959         0.955  1.38    233.   2.95e-8     2  -19.8
3 Korea,~     0.815         0.796  4.55     43.9  5.87e-5     2  -34.1
# ... with 4 more variables: AIC <dbl>, BIC <dbl>, deviance <dbl>,
#   df.residual <int>
```

augment

モデルに基づく観測値の予測

```
gap_lm %>% unnest(augmented,.drop = TRUE)
```

```
# A tibble: 36 x 10
   country lifeExp gdpPercap .fitted .se.fit .resid   .hat .sigma .cooksd
   <fct>     <dbl>     <dbl>   <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
 1 China      44        400.    56.1    2.88 -12.1  0.141    6.83 0.236
 2 China      50.5      576.    57.0    2.70  -6.46 0.124    7.76 0.0570
 3 China      44.5      488.    56.5    2.79 -12.0  0.132    6.85 0.215
 4 China      58.4      613.    57.2    2.66   1.18 0.120    8.08 0.00184
 5 China      63.1      677.    57.5    2.61   5.58 0.115    7.85 0.0389
 6 China      64.0      741.    57.9    2.55   6.09 0.110    7.80 0.0439
 7 China      65.5      962.    59.0    2.39   6.49 0.0967   7.77 0.0424
```

```
 8 China      67.3     1379.    61.2    2.22   6.06 0.0839   7.81 0.0312
 9 China      68.7     1656.    62.7    2.23   6.03 0.0847   7.82 0.0312
10 China      70.4     2289.    66.0    2.60   4.45 0.114    7.94 0.0245
# ... with 26 more rows, and 1 more variable: .std.resid <dbl>
```

機械学習

一般的な手順

- データを訓練用（および検証用）とテスト用に分割
  - データを標準化（主成分化）
- 分析手法に固有のパラメータ探索
- 訓練データにモデルを学習
- 学習モデルでテストデータを検証

メソッドの統一

R

```r
# fit
stats::glm(Y~X,
    "binomial")
kernlab::ksvm(Y~X)
# predict
predict(testX)
```

Python

```python
#
clf1 = Lasso()
#
clf2 = svm.SVC()
# fit
clf1.fit(X, y)
#
clf1.predict(testX)
```

Python Pipeline

```python
pipeline = Pipeline(
  [('vec',CountVectorizer()),
    ('clf', SGDClassifier()),])
parameters = {
    'vec__ngram_range': ((1, 1), (1, 2)),
    'clf__alpha': (0.00001, 0.000001),}
grid_search = GridSearchCV(pipeline,
      parameters, n_jobs=-1, verbose=1)
grid_search.fit(X, y)
```

テキスト分析

奈良先端科学技術大学院大学鈴木優氏 http://bigdata.naist.jp/~ysuzuki/data/twitter/

Yu Suzuki

Associate Professor, Nara Institute of Science and Technology

Data Lecture News Publications

Twitter日本語評判分析データセット

Nov 29, 2017
1 min read

ツイートの評判情報をクラウドソーシングにより分析し，分析結果を公開しています．

Tweetsを取得

Tweetsのstatus_idと、その評価のデータ(Tweets本文は取得する必要あり)

```
library(rtweet)
iPhone <- tw_lists %>% select(status_id) %>%
            pull() %>% lookup_tweets()
```

評判分析とは

ポジティブな投稿

```
iPhone_tweets %>% slice(14) %>% select(text) %>% pull()
```

```
[1] "iPhone6     "
```

ネガティブな投稿

```
iPhone_tweets %>% slice(13) %>% select(text) %>% pull()
```

```
[1] "iPhone6  "
```

予測モデル

メッセージに出現する単語から、内容がポジティブかネガティブかを二値判定

- 肯定的か否定的 ~ Word1 + Word2 + ⋯ + Word9999 + Word10000

データの用意

status_idごとに出現単語とその頻度

```
iPhone_df %>% arrange(TERM, status_id)
```

```
# A tibble: 18,247 x 3
   status_id          TERM             n
   <chr>              <chr>        <int>
 1 703580075891884032         1
 2 697419801073831936             1
```

```
 3 551308290882154498               1
 4 631470217957224449               1
 5 704299116335992832               1
 6 569511238411030529              1
 7 579902070888083456              1
 8 691241852272422912              1
 9 694381603263393792              1
10 696548443322318848              1
# ... with 18,237 more rows
```

機械学習用messyデータ

```
iPhone %>% dim()
```

```
[1] 5344 1437
```

機械学習の手法

- 罰則項付きロジスティック回帰 (glmnet)
- サポートベクターマシン (kernlab)
- 決定木 (rpart)
- ランダムフォレスト (randomForest)
- XGBoost (xgboost)

ロジスティック回帰

評判（1か0）を単語頻度で回帰

```
#
table(iPhone$Y)
```

```
FALSE  TRUE
 3503  1841
```

```
# Y ~ .
iPhone_glm <- glm(
 Y ~ .,
 data= iPhone,
 family = "binomial")
```

ロジスティック回帰

単語それぞれが独立した説明変数になる

```
library(broom)
iPhone_glm %>% tidy()
```

```
# A tibble: 1,303 x 5
   term        estimate  std.error statistic  p.value
   <chr>          <dbl>      <dbl>     <dbl>    <dbl>
 1 (Intercept)   -0.677     0.0574 -11.8     4.25e-32
 2 janetter     -19.2   10754.       -0.00179 9.99e- 1
 3 jojo          18.4   10754.        0.00171 9.99e- 1
 4 Kindle        17.4    1778.        0.00981 9.92e- 1
 5 KREVA        -17.1   10754.       -0.00159 9.99e- 1
```

```
 6 line          -19.2   10754.      -0.00179 9.99e- 1
 7 LIVE          -24.9    5367.      -0.00465 9.96e- 1
 8 ll            -19.7    7334.      -0.00268 9.98e- 1
 9 lP            -34.9   12567.      -0.00278 9.98e- 1
10 m              -9.67   1953.      -0.00495 9.96e- 1
# ... with 1,293 more rows
```

P値からキーワード

```
iPhone_glm %>% tidy() %>%
  filter(p.value < .01) %>%
    arrange(p.value)
```

```
# A tibble: 16 x 5
   term        estimate std.error statistic  p.value
   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)   -0.677    0.0574    -11.8  4.25e-32
 2              3.61     0.471       7.66 1.85e-14
 3              -0.799    0.116      -6.87 6.30e-12
 4              1.76     0.303       5.82 5.81e- 9
 5             -1.27     0.248      -5.14 2.74e- 7
 6 s              0.358    0.0742      4.82 1.41e- 6
 7              2.61     0.617       4.23 2.37e- 5
 8              -0.844    0.225      -3.76 1.73e- 4
 9              1.26     0.365       3.44 5.79e- 4
10              3.73     1.14        3.27 1.09e- 3
11             1.69     0.525       3.22 1.30e- 3
12              3.28     1.05        3.13 1.73e- 3
13             -1.13     0.413      -2.75 5.97e- 3
14        -0.542    0.198      -2.74 6.18e- 3
15 S              0.345    0.130       2.66 7.85e- 3
16              1.56     0.592       2.64 8.27e- 3
```

回帰による予測精度

混合行列で確認

```
preds <- round(
 predict(iPhone_glm,
type="response"))
```

```
table(preds,
      iPhone$Y)
```

```
preds FALSE TRUE
    0  3339  897
    1   164  944
```

汎化モデル

- バイアス（真の値との差）
  - モデルを複雑すると減少
  - ただし汎用性がなくなる
- バリアンス（予測値のばらつき）

  - モデルが簡素化すると減少
  - ただし予測精度は落ちる

汎化モデル

- データ全部を使わない
  - 訓練データで候補モデルを調整
  - テストデータで予測精度
- モデル特有のハイパーパラメータ

caret_vignettes

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

ロジスティック回帰の場合

- 回帰係数 β を調整するハイパーパラメータ
- 複雑さ指標 λ ：小さいほど複雑なモデル（0だとOLS）
- α が1の場合 β の一部は正確に 0 に: L1 正則化
- α が0の場合 β は 0に近づく: L2正則化

データの分割

```
index <- sample(
   N, N * 0.7)
train <-dat[index,]
test <- dat[-index,]
```

```
index <- caret::
createDataPartition
 (y = dat$Y,
   p = 0.7)
train <-dat[ index,]
test  <-dat[-index,]
```

Lasso回帰

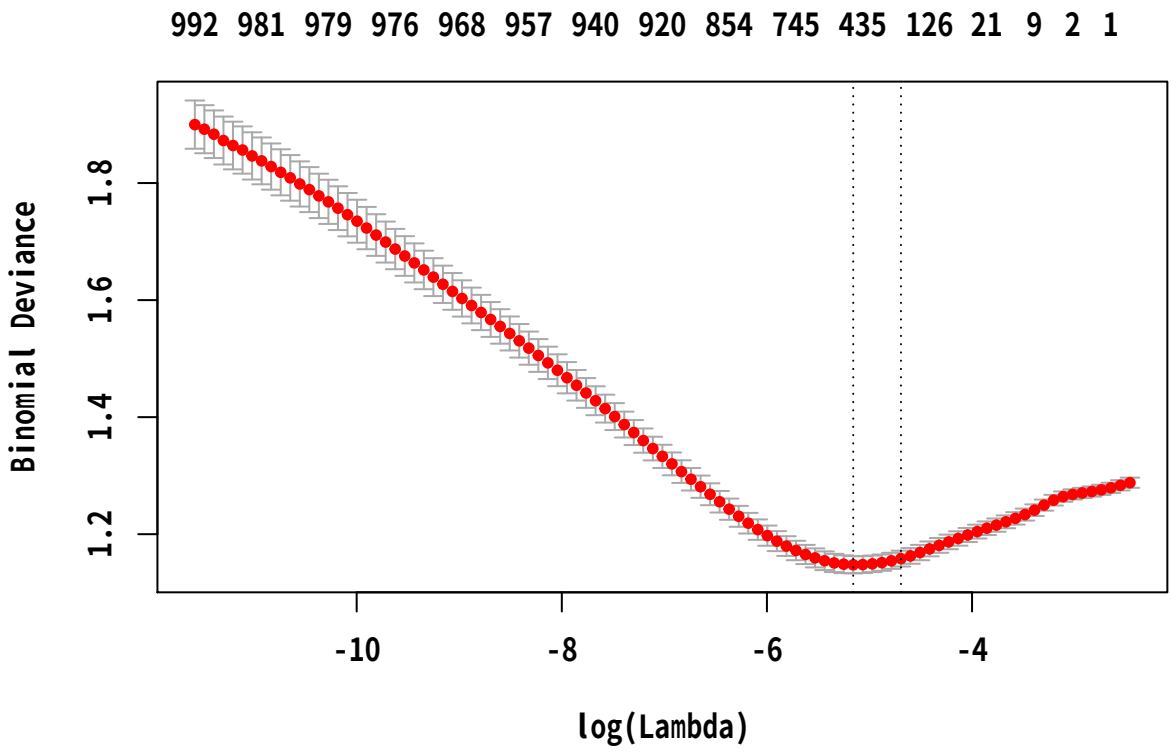
L1正則化：glmnetパッケージ

```
#
train_X <- training %>% select(-Y) %>%
```

```
  as.matrix(dimnames = #
              list(NULL, colnames(.)))
#
train_Y <- training$Y
#
test_X <- testing %>% select(-Y) %>%
  as.matrix(dimnames = list(NULL, colnames(.))) #
test_Y <- testing$Y
```

Lasso回帰

```
lasso <- glmnet(
  x = train_X,
  y = train_Y,
  family="binomial",
   parallel = TRUE)
```



推定結果

最適な λ の値

```
lasso$lambda.min; lasso$lambda.1se
```

```
[1] 0.005750368
```

```
[1] 0.009156212
```

予測精度

```
las_pr <-predict(lasso,newx = test_X,
                        type = "class")
confusionMatrix(table(las_pr, test_Y))
```

```
Confusion Matrix and Statistics

       test_Y
las_pr  FALSE TRUE
  FALSE  1006  434
  TRUE     44  118

               Accuracy : 0.7016
                 95% CI : (0.6786, 0.724)
    No Information Rate : 0.6554
    P-Value [Acc > NIR] : 4.688e-05

                  Kappa : 0.2065

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9581
            Specificity : 0.2138
         Pos Pred Value : 0.6986
         Neg Pred Value : 0.7284
             Prevalence : 0.6554
         Detection Rate : 0.6280
   Detection Prevalence : 0.8989
      Balanced Accuracy : 0.5859

       'Positive' Class : FALSE
```

予測精度の指標

- 感度(Sensitivity)：再現率
- 特異度(Specificity)
- 正解度 (Accurarcy)
- 精度(Precision) ：適合率

	予測が真	予測が偽
実際に真	真陽性(TP)	偽陰性(FN)
実際に偽	偽陽性(FP)	真陰性(TN)

ROC

感 度（真陽性率）：実際に T であるケースでTと予測された割合

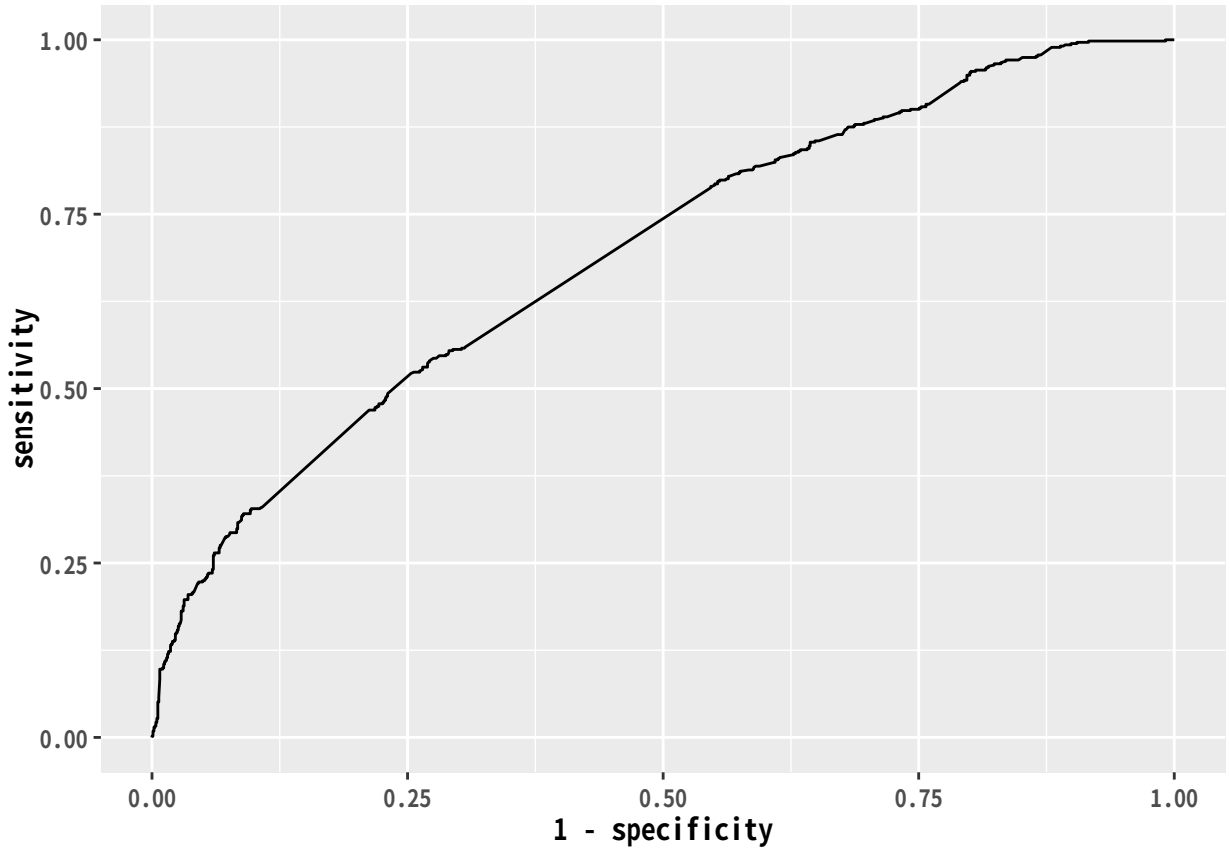
特異度（真陰性率）：実際に F であるケースでFと予測された割合

- 問題は判断の閾値

ROC

```
library(pROC)
lasso_res <-
  predict(
   lasso,
   s="lambda.min",
  newx =test_X,
  type ="response")
lasso_roc <-
  roc(test_Y,
   lasso_res)
```

```
ggroc(lasso_roc,
 legacy.axes=TRUE)
```



Method_auc

AUC基準で最良のモデルを推定する指定

```
lasso_cv_auc <- cv.glmnet(x = X, y = y,
  type.measure = "auc",#auc
  alpha = 1, family = "binomial")
```

サポートベクター

非線形の GaussianRBF (rbfdot) でクラス分類(C-svc)、10交差法：

```
library(kernlab)
svm_kernlab  <- ksvm(Y ~ .,
      data = training, scaled = FALSE,
      kernel="rbfdot", type="C-svc",
      cross = 10)
```

メソッドの統一

R

```
# fit
glmnet::glm(Y ~ X,
    "binomial")

kernlab::ksvm(
  Y ~ X, C=1)
```

Python

```
clf1 = Lasso()
clf2 = svm.SVC()
#
clf1.fit(X, y)
clf2.fit(X, y)
```

さまざまなモデルを試行

手法（パッケージ）の違いを吸収： caret, mlr, tidymodels

caret

```
caret::train(
    x =
  , y =
  , method =
  , tuneGrid =
  , trControl =
  , preProcess =
   )
```

caret

```
caret::train(
  x =    ,
  y =    ,
  method =     ,
  tuneGrid =          ,
  trControl =     , #
  # preProcess =       ,
   )
```

caretとglmnet

- 係数を調整する λや α を $(0, 1)$ 範囲で探する

- 交差法の指定： ホールドアウト、K分割交差法、繰り返しK分割交差法
  - パラメータ探索については tuneLength でに任せられる

parameters

パラメータの探索範囲

```
library(caret)
train_cntrl <-trainControl(
     method = "cv", number = 10,
     classProbs = TRUE,
     summaryFunction = twoClassSummary)
train_grid <- expand.grid(
        alpha  = seq(0, 1.0, by = 0.01) ,
        lambda = 10^(0:10 * -1))
```

preProcess

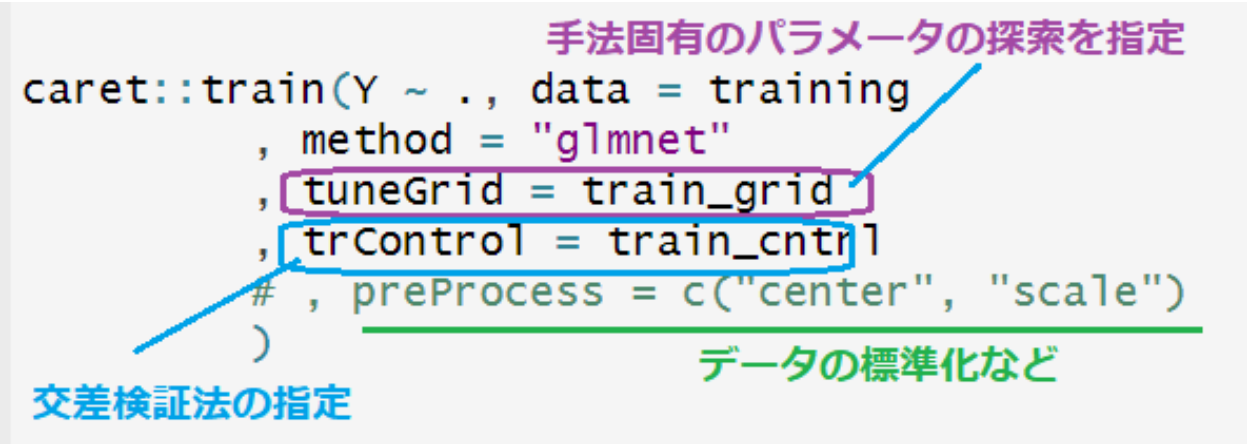
データの中心化、標準化、主成分化を指定

```
preProcess = c("center", "scale")
preProcess = c("pca")#   center scale
```

glmnet

```
caret::train(Y ~ ., data =training
    , method = "glmnet"
    , tuneGrid = train_grid
    , trControl = train_cntrl
    , metric = "ROC"
    #, preProcess = c("center","scale")
         )
```

コードの意味



kernlab

```
caret::train(Y ~ ., data = training,
    , method = "svmRadial",
```

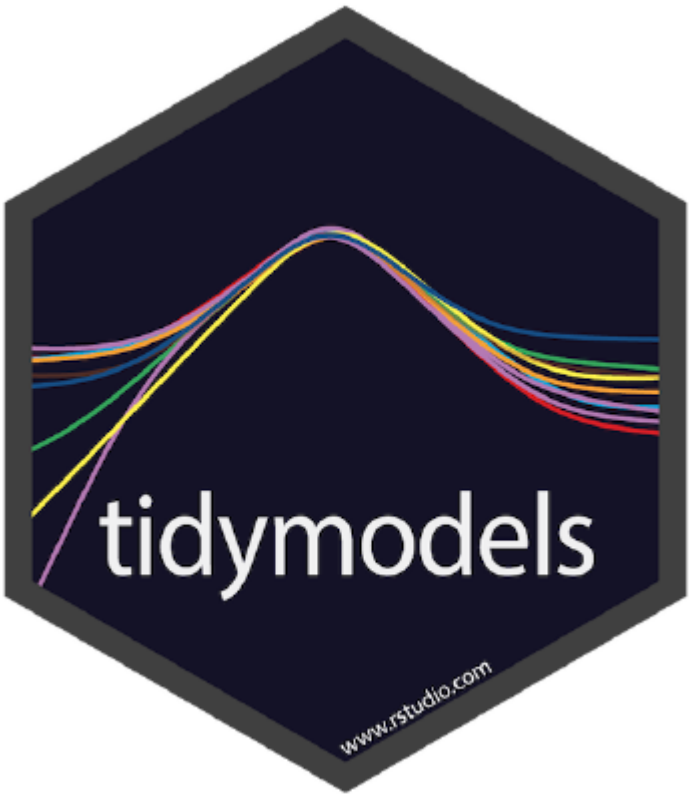
```
    , tuneGrid = expand.grid(C=1:10,
    ,   sigma = seq(0.0, 0.9, by = 0.1),
    , trControl = trainControl(
    ,   method = "cv", number = 10),
   # ,preProcess = c("center","scale")
       )
```

複数のモデル当てはめ

```
my_train <- function(method,formula,data){
   caret::train(formula, data, method
     , trControl = caret::trainControl(
         , method = 'cv')) }
models <- tibble(methods =
        c('lm', 'glmnet', "kernlab"))
models %>% mutate(fits = map(methods,
            train_mds, Y~X, my_train))
```

tidymodels

caret開発者によるtidy インターフェイス



複合的なパッケージ

```
library(tidymodels)
```

```
Registered S3 method overwritten by 'xts':
```

```
  method     from
  as.zoo.xts zoo
```

```
-- Attaching packages -------------- tidymodels 0.0.2 --
```

```
v dials     0.0.2       v recipes   0.1.6
v infer     0.4.0.1     v rsample   0.0.5
v parsnip   0.0.3.1     v yardstick 0.0.4
```

```
-- Conflicts ----------------- tidymodels_conflicts() --
x foreach::accumulate()  masks purrr::accumulate()
x scales::discard()      masks purrr::discard()
x dplyr::filter()        masks stats::filter()
x recipes::fixed()       masks stringr::fixed()
x dplyr::lag()           masks stats::lag()
x caret::lift()          masks purrr::lift()
x yardstick::precision() masks caret::precision()
x yardstick::recall()    masks caret::recall()
x yardstick::spec()      masks readr::spec()
x recipes::step()        masks stats::step()
x foreach::when()        masks purrr::when()
```

https://static1.squarespace.com/static/51156277e4b0b8b2ffe11c00/t/5b75871e21c67c985a06f481/1534428959053

tidymodelsでglmnet

データ分割： rsample::initial_split(), training(), testing()

```
splitted_data <- initial_split(
  iPhone, p = 0.5, strata = c('Y'))
train_data <- training(splitted_data)
test_data <- testing(splitted_data)
```

recipe

目的変数と説明変数を対応付ける

```
rec <- recipe(Y ~ ., data = train_data)
```

step_func

標準化など

```
#
rec <- rec %>% step_scale(all_numeric()) %>%
  step_corr(all_numeric(),
              - all_outcomes(),
            threshold = 0.5) %>%
  step_dummy(all_nominal()) #baseR
```

https://cran.r-project.org/web/packages/recipes/vignettes/Dummies.html

prep

訓練用データに prep() で下ごしらえ

```
rec_dat <- rec %>% prep(training =
                          train_data)
```

juice

下ごしらえを訓練用データとテスト用データのそれぞれに適用：juice(), bake()

```
train_juiced <- rec_dat %>% juice()
test_baked <- rec_dat %>%
  bake(test_data)
```

set_engine

mixture: α : penalty: λ

```
glmnet_model_tidy <-
  logistic_reg(mixture = 1,
               penalty = 10^(0:5 * -1)) %>%
                  set_engine("glmnet")
translate(glmnet_model_tidy)#
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = c(1, 0.1, 0.01, 0.001, 1e-04, 1e-05)
  mixture = 1

Computational engine: glmnet

Model fit template:
glmnet::glmnet(x = missing_arg(), y = missing_arg(), weights = missing_arg(),
    alpha = 1, family = "binomial")
```

fit

準備が整ったらモデルの学習

```
lasso_tidy <- glmnet_model_tidy %>%
      fit(Y ~ ., data = train_juiced)
```

predict

ペナルティごとに予測: multi_predict()

```
preds<-test_baked%>%
 select(Y) %>%
 bind_cols(fitted=
     multi_predict(
    lasso_tidy,
    test_baked))
```

```
# A tibble: 2,671 x 2
   Y     .pred
   <fct> <list>
 1 FALSE <tibble [6 x 2]>
 2 TRUE  <tibble [6 x 2]>
 3 FALSE <tibble [6 x 2]>
 4 TRUE  <tibble [6 x 2]>
 5 TRUE  <tibble [6 x 2]>
 6 TRUE  <tibble [6 x 2]>
```

```
 7 TRUE  <tibble [6 x 2]>
 8 TRUE  <tibble [6 x 2]>
 9 TRUE  <tibble [6 x 2]>
10 TRUE  <tibble [6 x 2]>
# ... with 2,661 more rows
```

予測値の抽出

最初のレコードに対する予測値を確認

```
preds$.pred[[1]]
```

```
# A tibble: 6 x 2
  penalty .pred
    <dbl> <fct>
1 0.00001 FALSE
2 0.0001  FALSE
3 0.001   FALSE
4 0.01    FALSE
5 0.1     FALSE
6 1       FALSE
```

予測値の展開

unnest() を使って .$pred 列を展開

```
preds_f <- preds %>%
 mutate(.pred =map(
  .pred,bind_rows))%>%
  unnest()
```

```
# A tibble: 16,026 x 3
   Y     penalty .pred
   <fct>   <dbl> <fct>
 1 FALSE 0.00001 FALSE
 2 FALSE 0.0001  FALSE
 3 FALSE 0.001   FALSE
 4 FALSE 0.01    FALSE
 5 FALSE 0.1     FALSE
 6 FALSE 1       FALSE
 7 TRUE  0.00001 FALSE
 8 TRUE  0.0001  FALSE
 9 TRUE  0.001   FALSE
10 TRUE  0.01    FALSE
# ... with 16,016 more rows
```

予測精度の評価

yardstick の各種評価関数

```
preds_f %>% group_by(penalty) %>%
                     metrics(Y, .pred)
```

```
# A tibble: 12 x 4
   penalty .metric  .estimator .estimate
     <dbl> <chr>    <chr>          <dbl>
 1 0.00001 accuracy binary         0.685
```

```
 2 0.0001  accuracy binary         0.686
 3 0.001   accuracy binary         0.691
 4 0.01    accuracy binary         0.701
 5 0.1     accuracy binary         0.656
 6 1       accuracy binary         0.656
 7 0.00001 kap      binary         0.215
 8 0.0001  kap      binary         0.219
 9 0.001   kap      binary         0.221
10 0.01    kap      binary         0.202
11 0.1     kap      binary         0
12 1       kap      binary         0
```

ランダムフォレスト

rand_forest():

```
rf_ <- rand_forest(mode = "classification",
    trees = 20, min_n = 100,
    mtry = 1000) %>%
  set_engine("randomForest", seed = 123)
rf_fit <- tf_ %>%
      fit(Y ~ ., data = train_baked)
```

複数モデルの当てはめ

```
rf <- rand_forest(trees = 10,
          mode = "classification") %>%
             set_engine("ranger")
lasso  <- logistic_reg(penalty=1) %>%
 set_engine("glmnet",family="multinomial")
models <- list(rf, lasso)
fitted <- models %>% map(
   ~ fit(.,Y ~ .,data = train))
```

まとめ

- 探索的分析は
  - map broom caret tidymodels で効率化を
- 退屈な作業はRにやらよう