BIOSTAT625 project test

Hengde Ouyang

2022-11-27

Read the Data and Remove the row contains NA

```
raw_data = read.csv("raw1.csv")
# Remove the row contains NA
complete_data = na.omit(raw_data)
```

Transform some of the data

```
transform_data = complete_data
# If the person is female, we code 0
transform_data$Sex = ifelse(transform_data$Sex=="Female",0,1)

# Survival months to numeric, and remove the unknown
# This is our response variable
transform_data$Survival.months = as.numeric(transform_data$Survival.months)
transform_data = na.omit(transform_data)

# Remove "years" in Single Age
# Code, if you want to remove:
# pattern = "years"
# transform_data$Age.recode.with.single.ages.and.85.  #=gsub(transform_data$Age.recode.with.single.ages
# replacement = "")

# Tumor Size to numeric, and remove the Blank
transform_data$CS.tumor.size..2004.2015. = as.numeric(transform_data$CS.tumor.size..2004.2015.)
transform_data = na.omit(transform_data)

# Tumor number to numeric, and remove the Blank
transform_data$Total.number.of.in.situ.malignant.tumors.for.patient = as.numeric(transform_data$Total.nu
transform_data = na.omit(transform_data)

### Silly Code
transform_data$Age.recode.with..1.year.olds = factor(transform_data$Age.recode.with..1.year.olds)
transform_data$Primary.Site...labeled = factor(transform_data$Primary.Site...labeled)
transform_data$Derived.AJCC.Stage.Group..6th.ed..2004.2015. = factor(transform_data$Derived.AJCC.Stage.
transform_data$ER.Status.Recode.Breast.Cancer..1990.. = factor(transform_data$ER.Status.Recode.Breast.Ca
transform_data$PR.Status.Recode.Breast.Cancer..1990.. = factor(transform_data$PR.Status.Recode.Breast.Ca
```

```
transform_data$Survival.months.flag = factor(transform_data$Survival.months.flag)
transform_data$Race.ethnicity = factor(transform_data$Race.ethnicity)
```

Summary Statistics

```
summary(transform_data)
```

```
##  Age.recode.with..1.year.olds      Sex           Year.of.diagnosis
##  60-64 years:28344            Min.   :0.000000   Min.   :2004
##  55-59 years:26558            1st Qu.:0.000000   1st Qu.:2007
##  65-69 years:26145            Median :0.000000   Median :2010
##  50-54 years:25358            Mean   :0.007141   Mean   :2010
##  45-49 years:21527            3rd Qu.:0.000000   3rd Qu.:2013
##  70-74 years:20838            Max.   :1.000000   Max.   :2015
##  (Other)    :66597
##                           Primary.Site...labeled  Primary.Site
##  C50.4-Upper-outer quadrant of breast:70152      Min.   :500.0
##  C50.8-Overlapping lesion of breast  :47570      1st Qu.:504.0
##  C50.9-Breast, NOS                   :31702      Median :504.0
##  C50.2-Upper-inner quadrant of breast:24828      Mean   :505.2
##  C50.5-Lower-outer quadrant of breast:15649      3rd Qu.:508.0
##  C50.3-Lower-inner quadrant of breast:11701      Max.   :509.0
##  (Other)                             :13765
##  Behavior.recode.for.analysis Derived.AJCC.Stage.Group..6th.ed..2004.2015.
##  Length:215367                I        :104360
##  Class :character             IIA      : 48252
##  Mode  :character             IIB      : 21025
##                               IIIA     : 12435
##                               IV       : 10503
##                               UNK Stage:  8831
##                               (Other)  :  9961
##   ER.Status.Recode.Breast.Cancer..1990..
##  Borderline/Unknown  :  7371
##  Negative            : 36293
##  Positive            :171342
##  Recode not available:   361
##
##
##
##   PR.Status.Recode.Breast.Cancer..1990.. CS.tumor.size..2004.2015.
##  Borderline/Unknown  :  8512            Min.   :  0.00
##  Negative            : 59090            1st Qu.: 10.00
##  Positive            :147404            Median : 17.00
##  Recode not available:   361            Mean   : 82.46
##                                         3rd Qu.: 30.00
##                                         Max.   :999.00
##
##  Survival.months
##  Min.   :  0.00
##  1st Qu.: 58.00
```

```
##  Median : 89.00
##  Mean   : 93.44
##  3rd Qu.:130.00
##  Max.   :191.00
##
##                                                                  Survival.months.flag
##  Complete dates are available and there are 0 days of survival           :   126
##  Complete dates are available and there are more than 0 days of survival  :207386
##  Incomplete dates are available and there cannot be zero days of follow-up:  7816
##  Incomplete dates are available and there could be zero days of follow-up :    39
##
##
##
##  Total.number.of.in.situ.malignant.tumors.for.patient
##  Min.   : 1.000
##  1st Qu.: 1.000
##  Median : 1.000
##  Mean   : 1.396
##  3rd Qu.: 2.000
##  Max.   :12.000
##
##  Total.number.of.benign.borderline.tumors.for.patient
##  Min.   :0.000000
##  1st Qu.:0.000000
##  Median :0.000000
##  Mean   :0.008609
##  3rd Qu.:0.000000
##  Max.   :5.000000
##
##  Age.recode.with.single.ages.and.85.  Race.ethnicity
##  Length:215367                        White   :172369
##  Class :character                     Black   : 17927
##  Mode  :character                     Filipino:  4882
##                                       Chinese :  4489
##                                       Japanese:  4392
##                                       Hawaiian:  2567
##                                       (Other) :  8741
```

Principal Components Regression

Suppose we have the original predictors $X_1, X_2, ..., X_p$,let $Z_1, Z_2, ..., Z_p$ represent $M < p$ linear combinations of our original p predictors, that is:

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

We show an example of Princinpal Components Regression:

```
#Before running the code, you need the "pls" and "ISLR" package
#install.packages("pls")
library(pls)
```

```
##
```

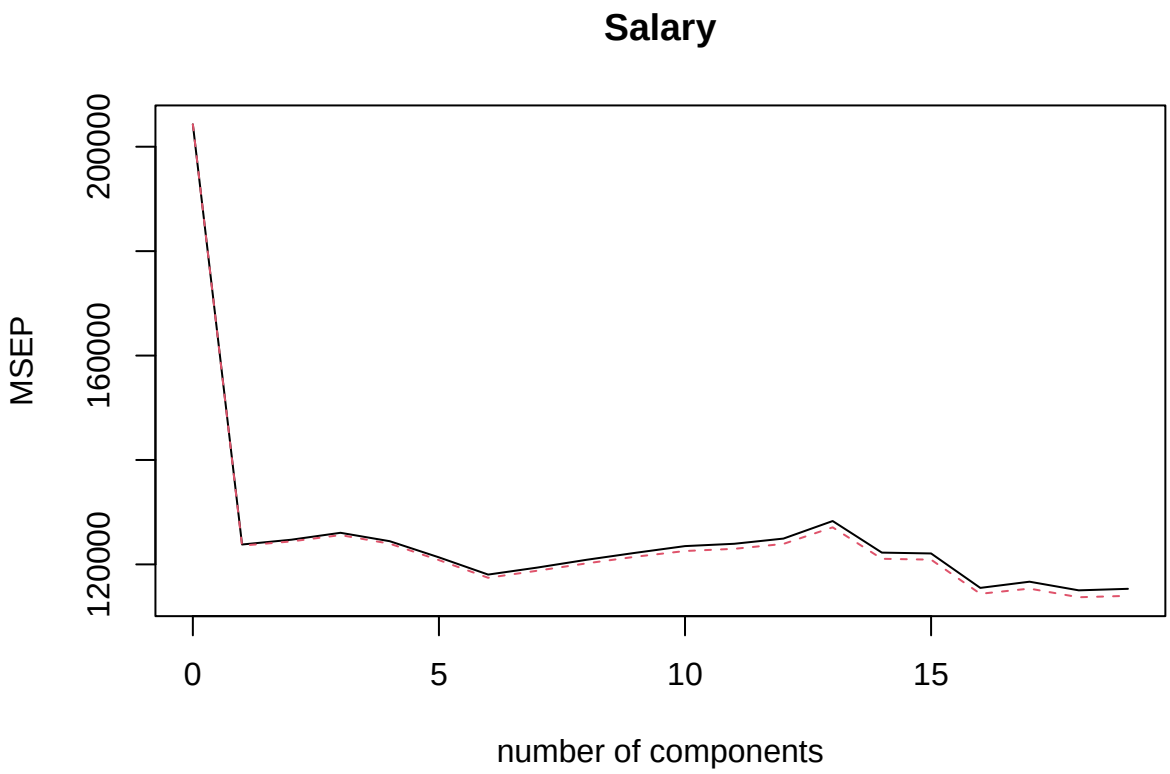
```
##     'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(ISLR)
```

```
set.seed(2)
Hitters =na.omit(Hitters)
pcr.fit=pcr(Salary~.,data=Hitters, scale=TRUE,
validation ="CV")
summary (pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```
validationplot(pcr.fit,val.type="MSEP")
```



training and testing data:

```
set.seed(1)
train=sample(c(TRUE ,FALSE), nrow(Hitters ),rep=TRUE)
test=(!train)


x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary

y.test=y[test]
```

```
set.seed(2)
pcr.fit=pcr(Salary~.,data=Hitters,subset=train,scale=TRUE,
validation ="CV")
pcr.pred=predict (pcr.fit,x[test ,],ncomp =7)
mean((pcr.pred -y.test)^2)
```

```
## [1] 145656
```

```
pcr.pred=predict (pcr.fit,x[test ,],ncomp =2)
mean((pcr.pred -y.test)^2)
```

```
## [1] 133781
```

```
pcr.pred=predict (pcr.fit,x[test ,],ncomp =1)
mean((pcr.pred -y.test)^2)
```

```
## [1] 135538.1
```

Random Forest

```
#Before running the code, you need the "randomForest" package
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(3)
bag.Hitters =  randomForest(Salary~.,data=Hitters, subset=train,
mtry=13,importance =TRUE)

yhat.bag = predict(bag.Hitters,newdata=Hitters[test,])
mean((yhat.bag -y.test)^2)
```

```
## [1] 105171.9
```

Comparision of time

```
system.time({
pcr.fit=pcr(Salary~.,data=Hitters,subset=train,scale=TRUE,
validation ="CV")
})
```

```
##
## 0.03 0.00 0.03
```

```
system.time({
bag.Hitters =  randomForest(Salary~.,data=Hitters, subset=train,
mtry=13,importance =TRUE)
})
```

```
##
## 0.36 0.02 0.38
```

Try to use future package

```
library(future)
plan(multisession)
set.seed(3)
system.time({
bag.Hitters2 =  future(randomForest(Salary~.,data=Hitters, subset=train,
mtry=13,importance =TRUE),seed = TRUE)
})
```

```
##
## 0.02 0.00 0.01
```

```
yhat.bag2 = predict(value(bag.Hitters2),newdata=Hitters[test,])
mean((yhat.bag2 -y.test)^2)
```

```
## [1] 105717
```