HW6

2019150445/Shin Baek Rok

2020 12 17

1.

a)

```
head(Default)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
fit<-glm(default~income+balance,data=Default,family='binomial')
fit
```

```
##
## Call:  glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Coefficients:
## (Intercept)       income      balance
##  -1.154e+01    2.081e-05    5.647e-03
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
## Null Deviance:       2921
## Residual Deviance: 1579  AIC: 1585
```

b)

```
#i)
test_index<-createDataPartition(Default$default,list=F)
Default_test<-Default[test_index,]
Default_train<-Default[-test_index,]

#ii)
fit_train<-glm(default~income+balance,data=Default_train,family='binomial')
fit_train
```

```
##
## Call:  glm(formula = default ~ income + balance, family = "binomial",
##     data = Default_train)
##
## Coefficients:
## (Intercept)       income      balance
##  -1.219e+01    2.751e-05    5.936e-03
##
## Degrees of Freedom: 4998 Total (i.e. Null);  4996 Residual
## Null Deviance:       1457
## Residual Deviance: 779.3     AIC: 785.3
```

```
#iii)
probs <- predict(fit_train, Default_test, type = "response")
y_hat<-ifelse(probs>0.5,'Yes','No')
mean(y_hat!=Default_test$default)
```

```
## [1] 0.02819436
```

c)

```
replicate(3,{
  test_index<-createDataPartition(Default$default,list=F)
  Default_test<-Default[test_index,]
  Default_train<-Default[-test_index,]

  fit_train<-glm(default~income+balance,data=Default_train,family='binomial')
  probs <- predict(fit_train, Default_test, type = "response")
  y_hat<-ifelse(probs>0.5,'Yes','No')
  mean(y_hat!=Default_test$default)
})
```

```
## [1] 0.02499500 0.02659468 0.02759448
```

Since we split test and train set by random, the test error rate can be variable. But it is quite simmilar.

d)

```
test_index<-createDataPartition(Default$default,list=F)
Default_test<-Default[test_index,]
Default_train<-Default[-test_index,]

fit_all<-glm(default~.,data=Default_train,family='binomial')
probs<-predict(fit_all,Default_test,type='response')
y_hat<-ifelse(probs>0.5,'Yes','No')
mean(y_hat!=Default_test$default)
```

```
## [1] 0.02579484
```

It depends on how you split the model, but it doesn't seem to change the test error that much. In other words, the student variable do not leads to a reduction in the test error rate.

2.

a)

```
set.seed(1)
glm(default~income+balance,data=Default,family='binomial') %>% summary()
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

se for income coef=4.985e-06. se for balance coef=2.274e-04.

b)

```
boot.fn<-function(data=Default,index){
  glm(default~income+balance,data=Default, subset=index, family='binomial')$coef[-1]
}
```

c)

```
boot(Default, boot.fn, 100)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 100)
##
```

```
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 2.080898e-05 -3.993598e-07 4.186088e-06
## t2* 5.647103e-03 -4.116657e-06 2.226242e-04
```

d) Obtained standard errors by two methods are quite similar.

3.

```
library(tidyverse)
library(purrr)
library(pdftools)
```

```
## Warning: package 'pdftools' was built under R version 4.0.3
```

```
## Using poppler version 0.73.0
```

```
library(dslabs)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
fn <- system.file("extdata", "RD-Mortality-Report_2015-18-180531.pdf",
                  package="dslabs")
dat <- map_df(str_split(pdf_text(fn), "\n"), function(s){
  s <- str_trim(s)
  header_index <- str_which(s, "2015")[1]
  tmp <- str_split(s[header_index], "\\s+", simplify = TRUE)
  month <- tmp[1]
  header <- tmp[-1]
  tail_index  <- str_which(s, "Total")
  n <- str_count(s, "\\d+")
  out <- c(1:header_index, which(n == 1),
           which(n >= 28), tail_index:length(s))
  s[-out] %>%  str_remove_all("[^\\d\\s]") %>% str_trim() %>%
    str_split_fixed("\\s+", n = 6) %>% .[,1:5] %>% as_tibble() %>%
    setNames(c("day", header)) %>%
    mutate(month = month, day = as.numeric(day)) %>%
    gather(year, deaths, -c(day, month)) %>%
    mutate(deaths = as.numeric(deaths))
```

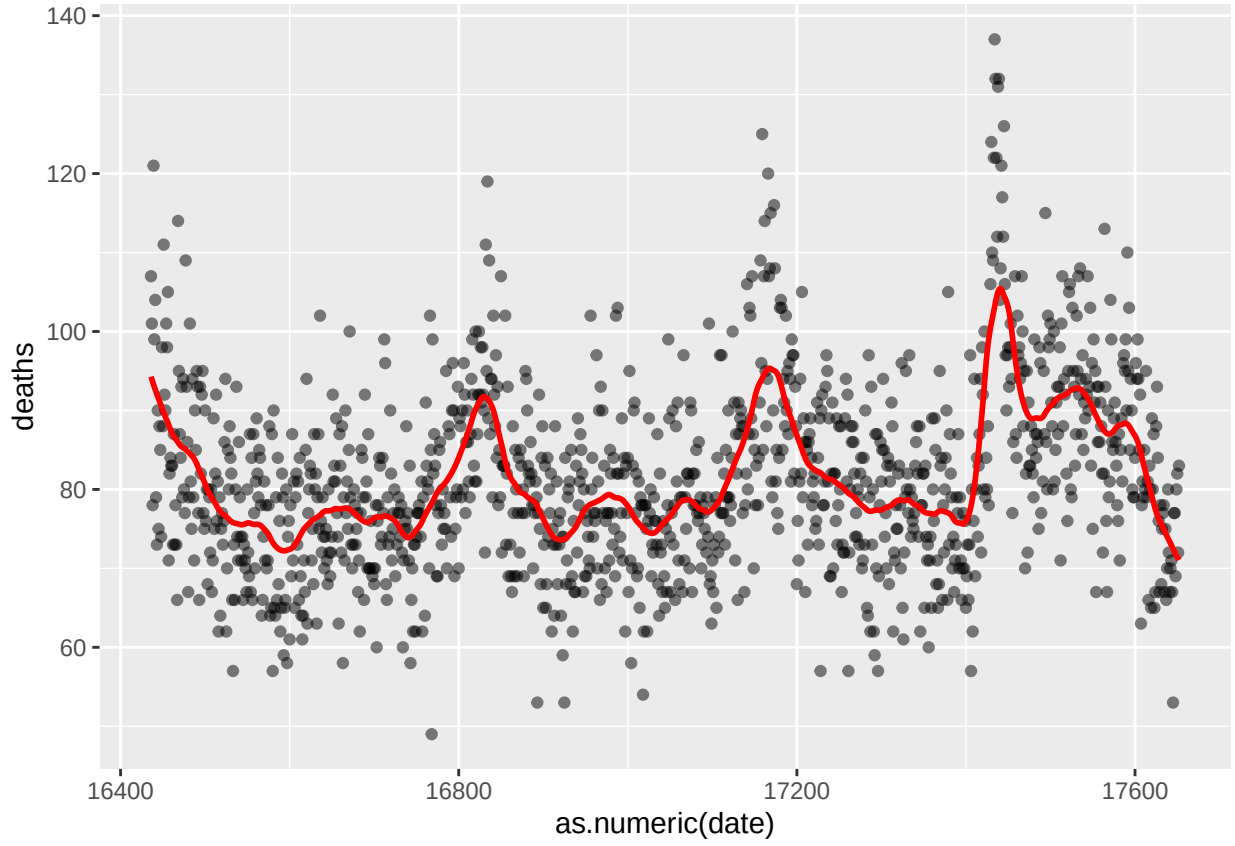
```
}) %>%
  mutate(month = recode(month,
                        "JAN" = 1, "FEB" = 2, "MAR" = 3,
                        "APR" = 4, "MAY" = 5, "JUN" = 6,
                        "JUL" = 7, "AGO" = 8, "SEP" = 9,
                        "OCT" = 10, "NOV" = 11, "DEC" = 12)) %>%
  mutate(date = make_date(year, month, day)) %>%
  filter(date <= "2018-05-01")
```

```
## Warning: The ‘x‘ argument of ‘as_tibble.matrix()‘ must have unique column names if ‘.name_repair‘ is
## Using compatibility ‘.name_repair‘.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

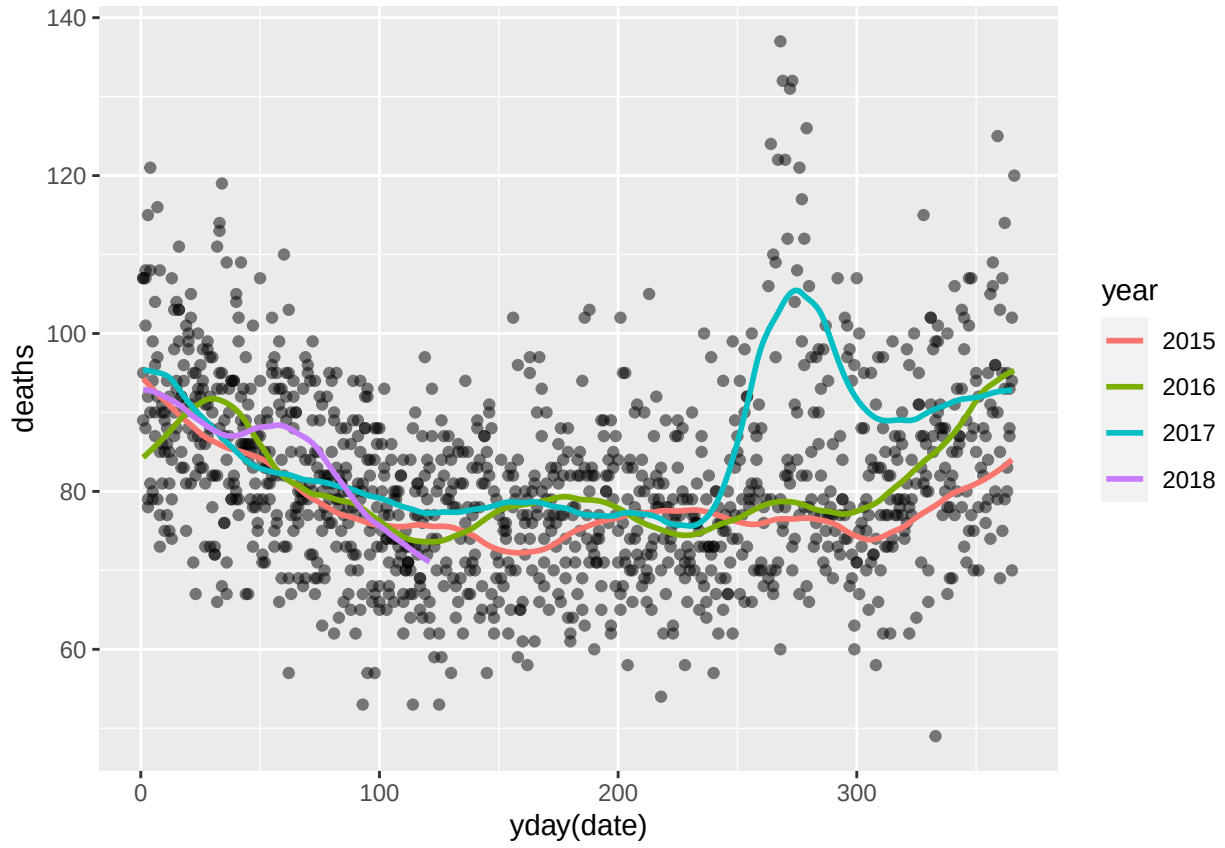
```
#1.
glimpse(dat)
```

```
## Rows: 1,205
## Columns: 5
## $ day    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...
## $ month  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ year   <chr> "2015", "2015", "2015", "2015", "2015", "2015", "2015", "201...
## $ deaths <dbl> 107, 101, 78, 121, 99, 104, 79, 73, 90, 75, 88, 85, 74, 98, ...
## $ date   <date> 2015-01-01, 2015-01-02, 2015-01-03, 2015-01-04, 2015-01-05,...
```

```
total<-as.numeric(dat$date) %>% range() %>% diff()
span<-61/total
fit<-loess(deaths~as.numeric(date),data=dat,span=span,degree=1)
dat<-dat[!is.na(dat$deaths),]
dat %>% mutate(smooth=fit$fitted) %>%
  ggplot(aes(as.numeric(date),deaths))+
  geom_point(alpha=.5)+
  geom_line(aes(as.numeric(date),smooth),col='red',size=1)
```




```
#2.
dat %>% mutate(smooth=predict(fit,as.numeric(dat$date))) %>%
  ggplot(aes(yday(date),deaths))+geom_point(alpha=.5)+
  geom_line(aes(yday(date),smooth,col=year),size=1)
```



4.

```
set.seed(1993)
data("tissue_gene_expression")
tissues <- c("cerebellum", "hippocampus")
ind <- which(tissue_gene_expression$y %in% tissues)
y <- droplevels(tissue_gene_expression$y[ind])
x <- tissue_gene_expression$x[ind, ]
x <- x[, sample(ncol(x), 10)]
```

```
#1.
fit<-train(x,y,method='lda')
fit
```

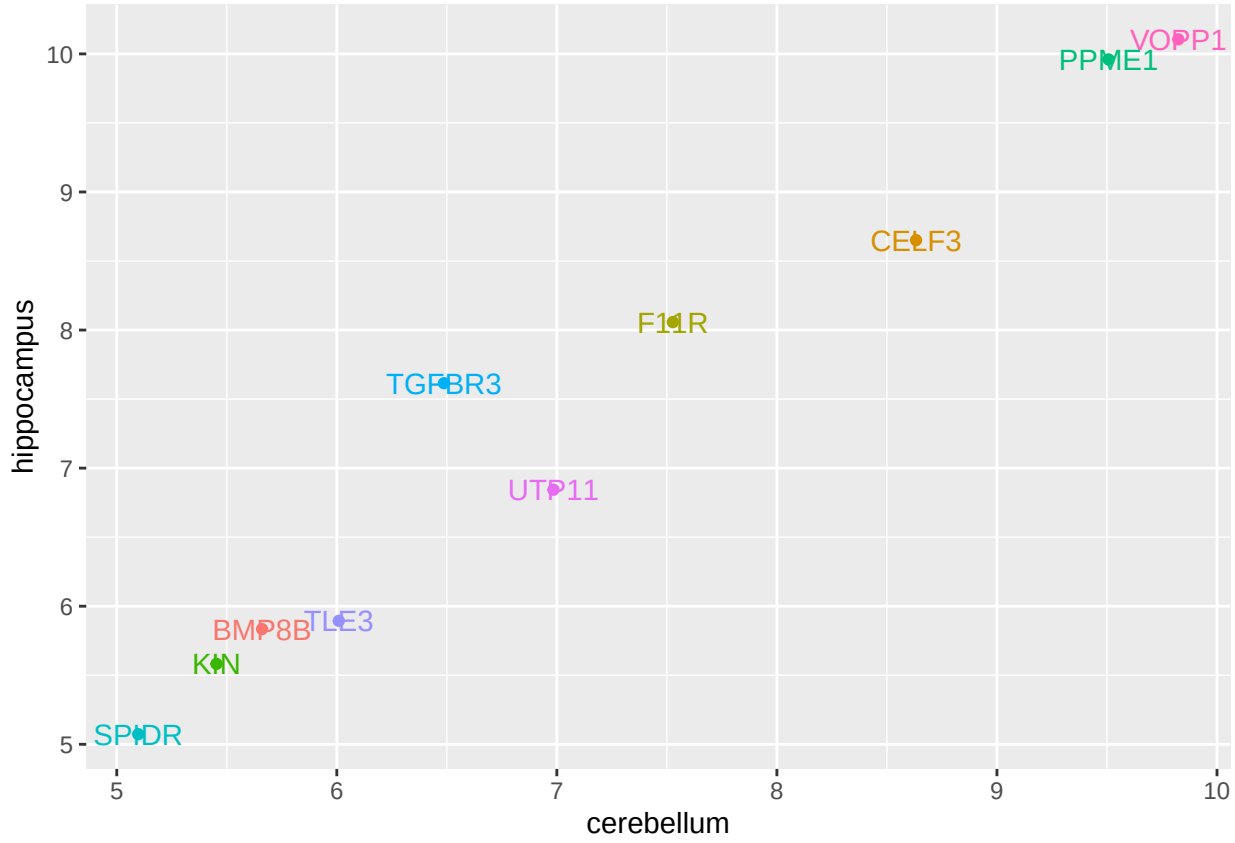
```
## Linear Discriminant Analysis
##
## 69 samples
## 10 predictors
##  2 classes: 'cerebellum', 'hippocampus'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 69, 69, 69, 69, 69, 69, ...
```

```
## Resampling results:
##
##   Accuracy  Kappa
##   0.974834  0.9483923
```

```
#2.
dat<-t(fit$final$means)
dat
```

```
##        cerebellum hippocampus
## BMP8B    5.660265    5.834400
## VOPP1    9.824540   10.105227
## KIN      5.453120    5.582600
## TGFBR3   6.486941    7.614880
## PPME1    9.507488    9.960071
## TLE3     6.008919    5.894071
## SPIDR    5.097781    5.073355
## F11R     7.527174    8.057552
## CELF3    8.632449    8.650689
## UTP11    6.984611    6.843593
```

```
dat %>% as_tibble() %>%
  mutate(label=as.factor(rownames(dat))) %>%
  ggplot(aes(cerebellum,hippocampus,col=label,label=label))+
  geom_point()+geom_text()+theme(legend.position="none")
```



```
#VOPP1 & PPME1 appear to be driving the algorithm


#3.
fit2<-train(x,y,method='qda')
fit2#lower accuracy than LDA
```

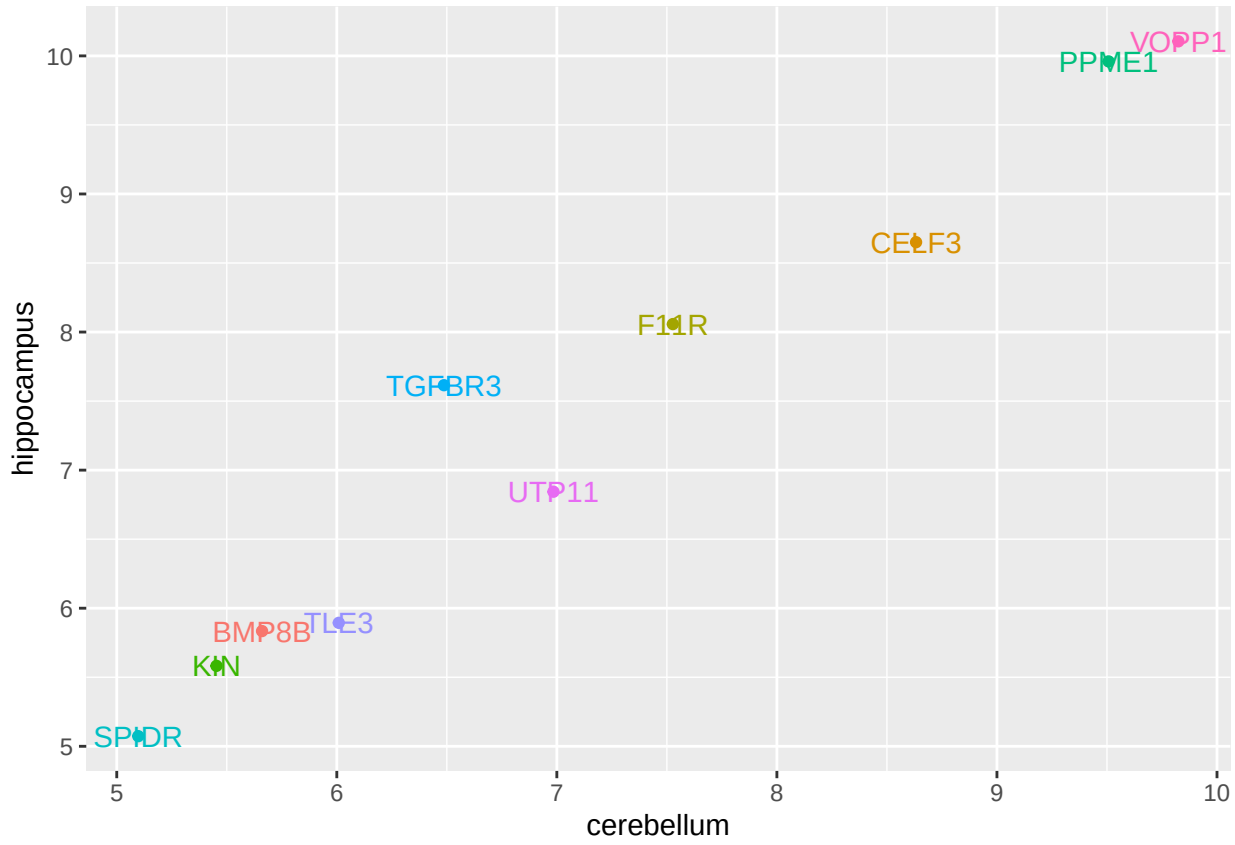
```
## Quadratic Discriminant Analysis
##
## 69 samples
## 10 predictors
##  2 classes: 'cerebellum', 'hippocampus'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 69, 69, 69, 69, 69, 69, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9577786  0.9132712
```

```
#4.

dat2<-t(fit2$final$means)
dat2
```

```
##        cerebellum hippocampus
## BMP8B    5.660265    5.834400
## VOPP1    9.824540   10.105227
## KIN      5.453120    5.582600
## TGFBR3   6.486941    7.614880
## PPME1    9.507488    9.960071
## TLE3     6.008919    5.894071
## SPIDR    5.097781    5.073355
## F11R     7.527174    8.057552
## CELF3    8.632449    8.650689
## UTP11    6.984611    6.843593
```

```
dat2 %>% as_tibble() %>%
  mutate(label=as.factor(rownames(dat))) %>%
  ggplot(aes(cerebellum,hippocampus,col=label,label=label))+
  geom_point()+geom_text()+theme(legend.position="none")
```




```
#VOPP1 & PPME1 appear to be driving the algorithm
```

‘‘‘