---

title: "Lab 6"

author: "Feiyi Ma"

output: pdf_document

date: "11:59PM April 15, 2021"

#Visualization with the package ggplot2

I highly recommend using the [ggplot cheat sheet](https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf) as a reference resource. You will see questions that say "Create the best-looking plot". Among other things you may choose to do, remember to label the axes using real English, provide a title, subtitle. You may want to pick a theme and color scheme that you like and keep that constant throughout this lab. The default is fine if you are running short of time.

Load up the `GSSvocab` dataset in package `carData` as `X` and drop all observations with missing measurements.

```{r}
pacman::p_load(carData)


data(GSSvocab)

GSSvocab = na.omit(GSSvocab)

?GSSvocab
```

Briefly summarize the documentation on this dataset. What is the data type of each variable? What do you think is the response variable the collectors of this data had in mind?

#TO-DO There are 8 different data variables. (1)Year- district integer. (2) gender- binoary variable (3) nativeBorn - string (4) ageGroup - continuous variable

#(5) education level - continuous variable (6) educ - continuous variable (7) vocab - district integer (8) age - continuous variable

#I think the response variable is vocab, the number of vocab the person have based on the other given information.

Create two different plots and identify the best-looking plot you can to examine the `age` variable. Save the best looking plot as an appropriately-named PDF.

```{r}
pacman::p_load(ggplot2)

ggplot(GSSvocab) +

  aes(x=age) +

  geom_histogram(bins = 50)
```

Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

```{r}
pacman::p_load(ggplot2)

ggplot(GSSvocab) +

  aes(x=factor(vocab)) +

  geom_bar() #not continuous
```

Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```{r}
ggplot(GSSvocab) +
```

```
  aes(x=ageGroup,y =gender) +
  geom_jitter(size=0.05)
```

Create the best-looking plot you can to examine the `vocab` variable by `age`. Does there appear to be an association?

```{r}
ggplot(GSSvocab) +
  aes(x=age,y =vocab) +
  geom_jitter()
```

Add an estimate of $f(x)$ using the smoothing geometry to the previous plot. Does there appear to be an association now?

```{r}
ggplot(GSSvocab) +
  aes(x=age,y =vocab) +
  geom_jitter()+
  geom_smooth() #col = " red"
```

Using the plot from the previous question, create the best looking plot overloading with variable `gender`. Does there appear to be an interaction of `gender` and `age`?

```{r}
ggplot(GSSvocab) +
  aes(x=age,y =vocab) +
  geom_jitter(aes(col=gender))+
```

  geom_smooth() #with third variable, aes add another variable

```

Using the plot from the previous question, create the best looking plot overloading with variable `nativeBorn`. Does there appear to be an interaction of `nativeBorn` and `age`?

```{r}
ggplot(GSSvocab) +
  aes(x=age,y =vocab) +
  geom_jitter(aes(col=nativeBorn),size = 0.5,alpha = 0.5)+
  geom_smooth()
```

Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

```{r}
ggplot(GSSvocab) +
  aes(x=educGroup,y =vocab) + #educ is category, vocab is deistic continuous
  geom_boxplot()
ggplot(GSSvocab) +
  aes(x=vocab)+
  geom_density(aes(fill = educGroup), adjust = 2,alpha = 0.5)

```

Using the best-looking plot from the previous question, create the best looking overloading plot with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?
```

```{r}
ggplot(GSSvocab) +
  aes(x=educGroup,y =vocab) + #add third varible gender
  geom_boxplot(aes(col=gender))
```

Using facets, examine the relationship between `vocab` and `ageGroup`. Are we getting dumber?

```{r}
ggplot(GSSvocab) +
  aes(x=vocab)+     #facets is add other variable
  geom_density(adjust =2, fill="black")+
  facet_wrap(ageGroup~. )

```

Probability Estimation and Model Selection

Load up the `adult` in the package `ucidata` dataset and remove missingness and the variable `fnlwgt`:

```{r}
pacman::p_load_gh("coatless/ucidata")
data(adult)
adult = na.omit(adult) #kill any observations with missingness
adult$fnlwgt = NULL
```

Cast income to binary where 1 is the `>50K` level.

```{r}
adult$income = ifelse(adult$income == ">50K",1,0)
```

We are going to do some dataset cleanup now. But in every cleanup job, there's always more to clean! So don't expect this cleanup to be perfect.

Firstly, a couple of small things. In variable `marital_status` collapse the levels `Married-AF-spouse` (armed force marriage) and `Married-civ-spouse` (civilian marriage) together into one level called `Married`. Then in variable `education` collapse the levels `1st-4th` and `Preschool` together into a level called `<=4th`.

```{r}
adult$marital_status = as.character(adult$marital_status)

adult$marital_status = ifelse(adult$marital_status == "Married-AF-spouse"| adult$marital_status == "Married-civ-spouse", "married",adult$marital_status)

adult$marital_status = as.factor(adult$marital_status)


adult$education = as.character(adult$education)

adult$education = ifelse(adult$education == "1st-4th"| adult$marital_status == "Preschool", "<=4th",adult$education)

adult$education = as.factor(adult$education)
```

Create a model matrix `Xmm` (for this prediction task on just raw features) and show that it is *not* full rank (i.e. the result of `ncol` is greater than the result of `Matrix::rankMatrix`).

```{r}
Xmm = model.matrix(income~., adult)
```

ncol(Xmm)

Matrix::rankMatrix(Xmm)

```

Now tabulate and sort the variable `native_country`.

```{r}

tab = sort(table(adult$native_country)) # one person from Holand-Nether lands that can be dropped.

tab

```

Do you see rare levels in this variable? Explain why this may be a problem.

#TO-DO yes. There is a country that has only one sample size, whcih can be ignored by formular, but will cause error in carculation.

Collapse all levels that have less than 50 observations into a new level called `other`. This is a very common data science trick that will make your life much easier. If you can't hope to model rare levels, just give up and do something practical! I would recommend first casting the variable to type "character" and then do the level reduction and then recasting back to type `factor`. Tabulate and sort the variable `native_country` to make sure you did it right.

```{r}

adult$native_country = as.character(adult$native_country)

adult$native_country = ifelse(adult$native_country %in% names(tab[tab<50]), "other",adult$native_country)

adult$native_country = as.factor(adult$native_country)

```

We're still not done getting this data down to full rank. Take a look at the model matrix just for `workclass` and `occupation`. Is it full rank?
```

```{r}

Xmm = model.matrix(income~ workclass + occupation, adult)

ncol(Xmm)

Matrix::rankMatrix(Xmm)

```

These variables are similar and they probably should be interacted anyway eventually. Let's combine them into one factor. Create a character variable named `worktype` that is the result of concatenating `occupation` and `workclass` together with a ":" in between. Use the `paste` function with the `sep` argument (this casts automatically to type `character`). Then tabulate its levels and sort.

```{r}

worktype = paste(adult$occupation, adult$workclass, sep=": ")

table2=sort(table(worktype))

table2

```

Like the `native_country` exercise, there are a lot of rare levels. Collapse levels with less than 100 observations to type `other` and then cast this variable `worktype` as type `factor`. Recheck the tabulation to ensure you did this correct.

```{r}

worktype = as.character(worktype)

worktype = ifelse(worktype %in% names(tab[tab<100]),  "other",worktype)

worktype = as.factor(worktype)

```

To do at home: merge the two variables `relationship` and `marital_status` together in a similar way to what we did here.

```{r}

status = paste(adult$relationship, adult$marital_status, sep=": ")

table3=sort(table(status))

table3

```

We are finally ready to fit some probability estimation models for `income`! In lecture 16 we spoke about model selection using a cross-validation procedure. Let's build this up step by step. First, split the dataset into `Xtrain`, `ytrain`, `Xtest`, `ytest` using K=5.

```{r}

set.seed(1984)

K = 5

test_prop = 1 / K

train_indices = sample(1 : nrow(adult), round((1 - test_prop) * nrow(adult)))

adult_train = adult[train_indices, ]

y_train = adult_train$income

X_train = adult_train

X_train$income = NULL

test_indices = setdiff(1 : nrow(adult), train_indices)

adult_test = adult[test_indices, ]

y_test = adult_test$income

X_test = adult_test

X_test$income = NULL

```

Create the following four models on the training data in a `list` object named `prob_est_mods`: logit, probit, cloglog and cauchit (which we didn't do in class but might as well). For the linear component within the link function, just use the vanilla raw features using the `formula` object `vanilla`. Each model's key in the list is its link function name + "-vanilla". One for loop should do the trick here.

```{r}
link_functions = c("logit", "probit", "cloglog", "cauchit")
vanilla = income ~ .
prob_est_mods = list()

for(link_function in link_functions){
  prob_est_mods[[paste(link_function, "vanilla", sep="-")]]= glm(vanilla, adult_train, family = binomial(link= link_function))
}
```

Now let's get fancier. Let's do some variable transforms. Add `log_capital_loss` derived from `capital_loss` and `log_capital_gain` derived from `capital_gain`. Since there are zeroes here, use log_x = log(1 + x) instead of log_x = log(x). That's always a neat trick. Just add them directly to the data frame so they'll be picked up with the `.` inside of a formula.

```{r}

adult$log_capital_loss = log(1+ strtoi(adult$capital_loss))
adult$log_capital_gain =log(1+ strtoi(adult$capital_gain))
head(adult)

```

Create a density plot that shows the age distribution by `income`.

```{r}
ggplot(adult) +
  aes(x=age) +
  geom_density(aes(fill = age))

```

What do you see? Is this expected using common sense?

no, usually not this way, hard to see.

Now let's fit the same models with all link functions on a formula called `age_interactions` that uses interactions for `age` with all of the variables. Add all these models to the `prob_est_mods` list.

```{r}
age_interactions = age ~.
#TO-DO
```

Create a function called `brier_score` that takes in a probability estimation model, a dataframe `X` and its responses `y` and then calculates the brier score.

```{r}
brier_score = function(prob_est_mod, X, y){
  phat = predict(prob_est_mod,X)
  mean(-(y-phat)^2)


}
```

Now, calculate the in-sample Brier scores for all models. You can use the function `lapply` to iterate over the list and pass in in the function `brier_score`.

```{r}
lapply(prob_est_mods, brier_score,X_train,y_train)
```

Now, calculate the out-of-sample Brier scores for all models. You can use the function `lapply` to iterate over the list and pass in the function `brier_score`.

```{r}
lapply(prob_est_mods, brier_score,X_test,y_test)
```

Which model wins in sample and which wins out of sample? Do you expect these results? Explain.

#brier_score wins both. it means the difference squre negative, ususally small.

What is wrong with this model selection procedure? There are a few things wrong.

(1)use age compare with other data, may overfit, (2) omit some data will result error.

Run all the models again. This time do three splits: subtrain, select and test. After selecting the best model, provide a true oos Brier score for the winning model.

```{r}
set.seed(1984)
K = 5
```

````
test_prop = 1 / K

subtrain_indices = sample(1 : nrow(adult), round((1 - test_prop) * nrow(adult)))

adult_subtrain = adult[subtrain_indices, ]

y_subtrain = adult_subtrain$income

X_subtrain = adult_subtrain

X_subtrain$income = NULL


select_indices = sample(subtrain_indices,size = round((1 - test_prop) * nrow(adult))/K)

adult_select = adult[select_indices, ]

y_select = adult_select$income

X_select = adult_select

X_select$income = NULL


test_indices = setdiff(subtrain_indices, select_indices)

adult_test = adult[test_indices, ]

y_test = adult_test$income

X_test = adult_test

X_test$income = NULL
```




```{r}
prob_est_mods = list()


for(link_function in link_functions){

  prob_est_mods[[paste(link_function, "vanilla", sep="-")]]= glm(vanilla, adult_subtrain, family = binomial(link= link_function))

}
````

````
for(link_function in link_functions){
  prob_est_mods[[paste(link_function, "age_interactions", sep="-")]]= glm(interactions, adult_subtrain, family = binomial(link= link_function))
}
```

```{r}
brier_insample = lapply(prob_est_mods, brier_score,adult_select,y_select)
brier_outsample = lapply(prob_est_mods, brier_score,X_test,y_test)

insample_se = which.max(brier_insample)
oos_se = which.max(brier_outsample)
insample_se
oos_se
````