#1 Read in the adult_sal.csv file and set it to a data frame called adult, and Check the head of adult.

```
adult <- read.csv('C:/Users/shrey/Downloads/adult_sal.csv')
adult
head(adult)
```

#2You should notice the index has been repeated. Drop this column.

library(dplyr)
adult <- select(adult,-X)

```
library(dplyr)
adult <- select(adult,-X)
adult
```

#3Check the head,str, and summary of the data now.

```
head(adult)
str(adult)
summary(adult)
```

#4Use table() to check out the frequency of the type_employer column.

```
table(adult$type_employer)
```

#5 How many Null values are there for type_employer? What are the two smallest groups?

```
sum((adult$type_employer == "?"))
```

#6 Combine these two smallest groups into a single group called "Unemployed". There are lots of ways to do this, so feel free to get creative. Hint: It may be helpful to convert these objects into character data types (as.character() and then use sapply with a custom function)

```
unemp <- function(job){
  job <- as.character(job)
```

```
 if (job=='Never-worked' | job=='Without-pay'){
  return('Unemployed')
 }else{
  return(job)
 }
}

adult$type_employer <- sapply(adult$type_employer,unemp)

table(adult$type_employer)
```

#7 What other columns are suitable for combining? Combine State and Local gov jobs into a category called SL-gov and combine self-employed jobs into a category called self-emp.

```
group_emp <- function(job){
 if (job=='Local-gov' | job=='State-gov'){
  return('SL-gov')
 }else if (job=='Self-emp-inc' | job=='Self-emp-not-inc'){
  return('self-emp')
 }else{
  return(job)
 }
}
adult$type_employer <- sapply(adult$type_employer,group_emp)
table(adult$type_employer)
```

#8 Use table() to look at the marital column.

```
table(adult$marital)
```

#9 Reduce this to three groups:

- Married
- Not-Married
- Never-Married

```
group_marital <- function(mar){
 mar <- as.character(mar)
```

```
  # Not-Married
  if (mar=='Separated' | mar=='Divorced' | mar=='Widowed'){
    return('Not-Married')

    # Never-Married
  }else if(mar=='Never-married'){
    return(mar)

    #Married
  }else{
    return('Married')
  }
}

adult$marital <- sapply(adult$marital,group_marital)
table(adult$marital)
```

#10 Check the country column using table().

```
table(adult$country)
```

#11 Group these countries together however you see fit. You have flexibility here because there is no right/wrong way to do this, possibly group by continents. You should be able to reduce the number of groups here significantly though.

```
levels(adult$country)

Asia <- c('China','Hong','India','Iran','Cambodia','Japan', 'Laos', 'Philippines' ,'Vietnam'
,'Taiwan', 'Thailand')
North.America <- c('Canada','United-States','Puerto-Rico' )
Europe <- c('England' ,'France', 'Germany' ,'Greece','Holand-Netherlands','Hungary',
'Ireland','Italy','Poland','Portugal','Scotland','Yugoslavia')
Latin.and.South.America <- c('Columbia','Cuba','Dominican-Republic','Ecuador', 'El-
Salvador','Guatemala','Haiti','Honduras', 'Mexico','Nicaragua','Outlying-US(Guam-
USVI-etc)','Peru',          'Jamaica','Trinadad&Tobago')

Other <- c('South')

group_country <- function(ctry){
  if (ctry %in% Asia){
```

```
  return('Asia')    }
 else if (ctry %in% North.America){
  return('North.America')    }
 else if (ctry %in% Europe){
  return('Europe')    }
 else if (ctry %in% Latin.and.South.America){
  return('Latin.and.South.America')    }
 else{
  return('Other')         }
}

adult$country <- sapply(adult$country,group_country)
adult$country
```

#12 Use table() to confirm the groupings.

```
table(adult$country)
```

#13Check the str() of adult again. Make sure any of the columns we changed have factor levels with factor().

```
str(adult)
adult$type_employer <- sapply(adult$type_employer,factor)
adult$country <- sapply(adult$country,factor)
adult$marital <- sapply(adult$marital,factor)
adult$type_employer
adult$country
adult$marital
```

#14 Install and load the Amelia package.

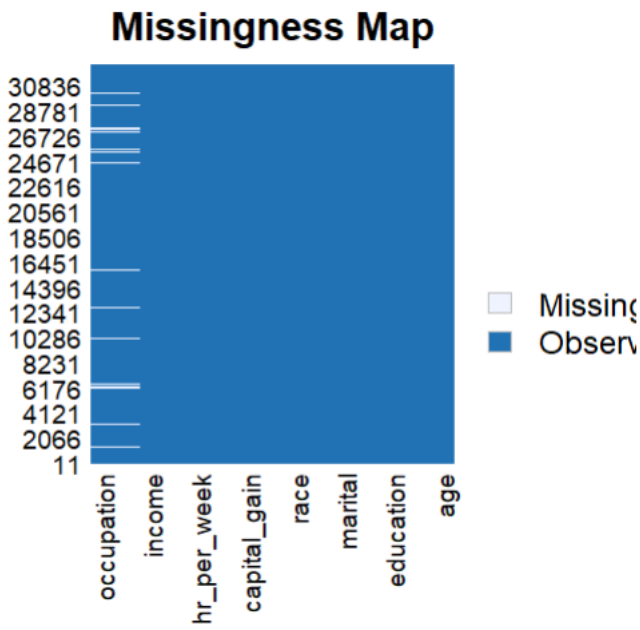
```
install.packages('Amelia')
library(Amelia)
adult[adult == '?'] <- NA
adult
```

#15 Using table() on a column with NA values should now not display those NA values, instead you'll just see 0 for ?. Optional: Refactor these columns (may take awhile).

```
table(adult$type_employer)
adult$type_employer <- sapply(adult$type_employer,factor)
adult$country <- sapply(adult$country,factor)
adult$marital <- sapply(adult$marital,factor)
adult$occupation <- sapply(adult$occupation,factor)
adult$type_employer
adult$country
adult$marital
adult$occupation
```

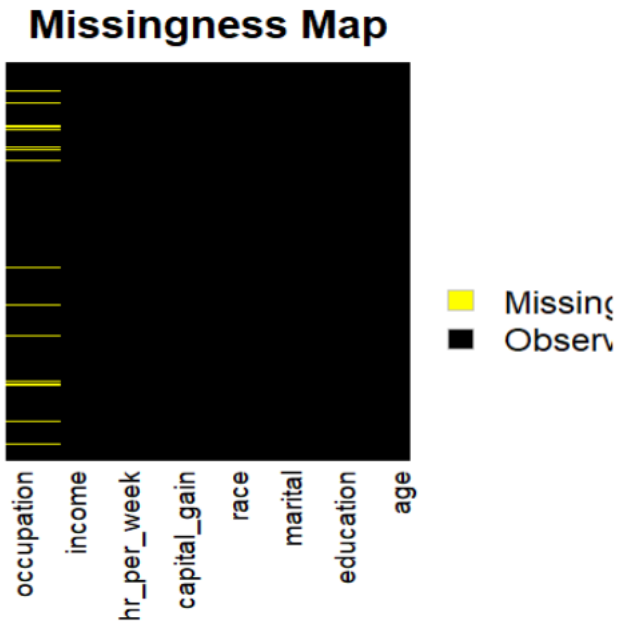
#16 Play around with the missmap function from the Amelia package. Can you figure out what its doing and how to use it?

```
missmap(adult)
```



#17. You should have noticed that using missmap(adult) is bascially a heatmap pointing out missing values (NA). This gives you a quick glance at how much data is missing, in this case, not a whole lot (relatively speaking). You probably also noticed that there is a bunch of y labels, get rid of them by running the command below. What is col=c('yellow','black') doing?
missmap(adult,y.at=c(1),y.labels = c(''),col=c('yellow','black'))

```
missmap(adult,y.at=c(1),y.labels = c(''),col=c('yellow','black'))
```

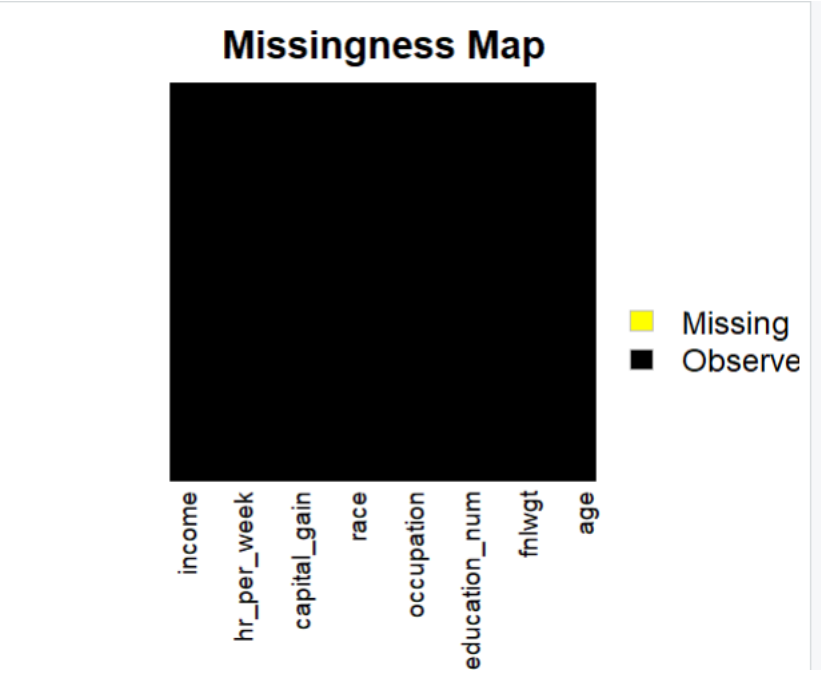


#18 Use na.omit() to omit NA data from the adult data frame. Note, it really depends on the situation and your data to judge whether or not this is a good decision. You shouldn't always just drop NA values.

```
adult <- na.omit(adult)
adult
```

#19 Use missmap() to check that all the NA values were in fact dropped.

```
missmap(adult,y.at=c(1),y.labels = c(''),col=c('yellow','black'))
```

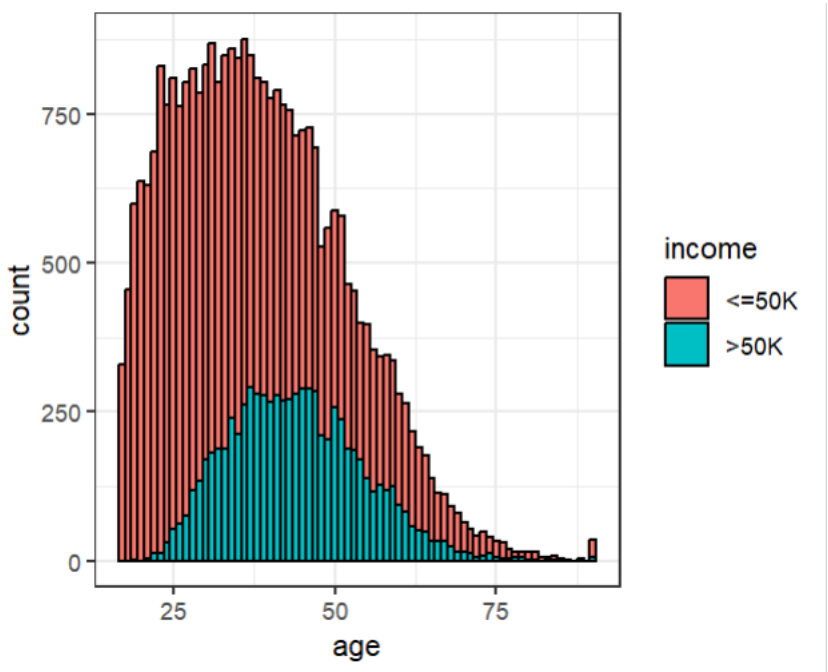


#20 Check the str() of the data.

```
str(adult)
```

#21 Use ggplot2 to create a histogram of ages, colored by income.

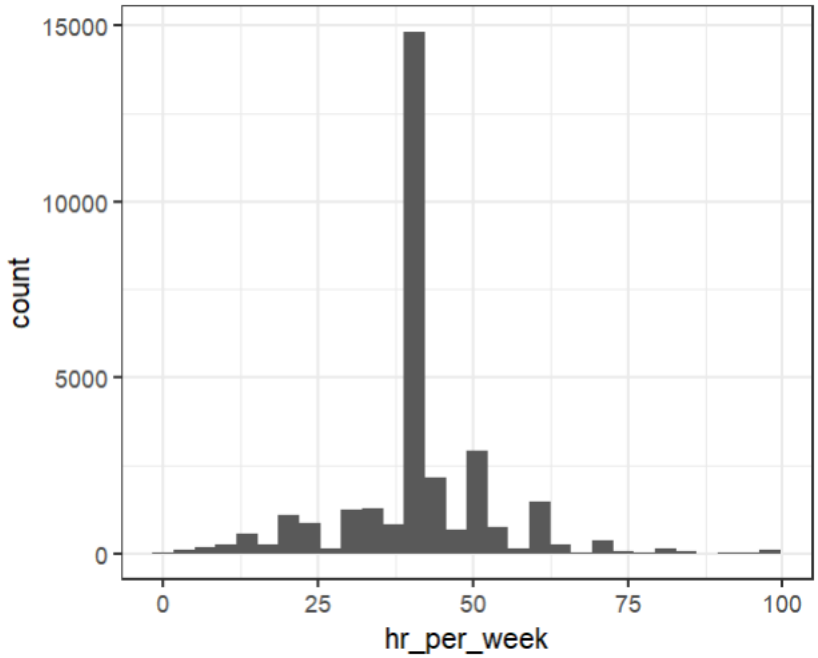
```
library(ggplot2)
library(dplyr)
ggplot(adult,aes(age)) + geom_histogram(aes(fill=income),color='black',binwidth=1)
+ theme_bw()
```



#22 Plot a histogram of hours worked per week.

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```
ggplot(adult,aes(hr_per_week)) + geom_histogram() + theme_bw()
```

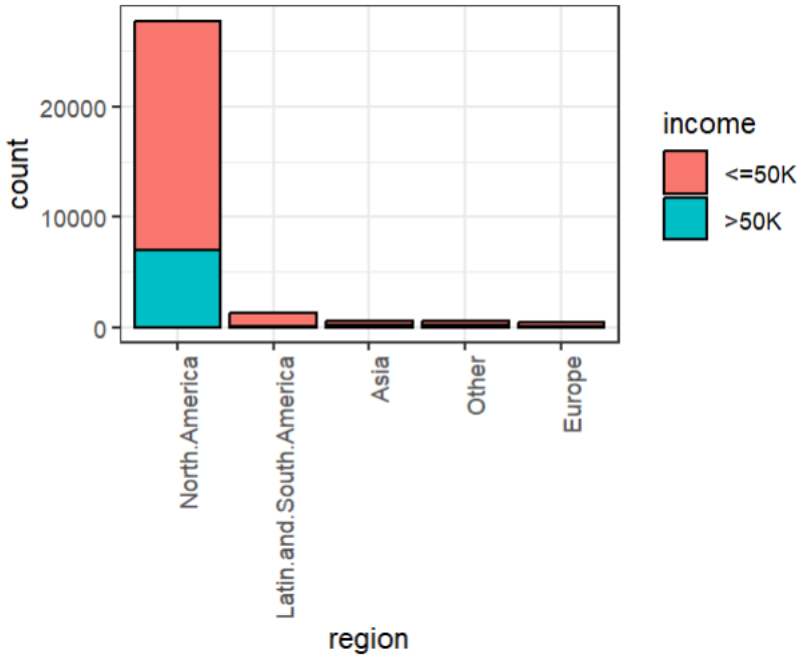


#23 Rename the country column to region column to better reflect the factor levels.

```
names(adult)[names(adult)=="country"] <- "region"
str
```

#24 Create a barplot of region with the fill color defined by income class. Optional: Figure out how rotate the x axis text for readability.

```
ggplot(adult,aes(region)) + geom_bar(aes(fill=income),color='black')+theme_bw()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



#25 Take a quick look at the head() of adult to make sure we have a good overview before going into building the model!

```
head(adult)
```

#26 Train Test Split:
Split the data into a train and test set using the caTools library as done in previous lectures. Reference previous solutions notebooks if you need a refresher.

```
# Import Library
install.packages('caTools')
library(caTools)
# Set a random see so your "random" results are the same as this notebook
set.seed(101)
# Split up the sample, basically randomly assigns a booleans to a new column "sample"
sample <- sample.split(adult$income, SplitRatio = 0.70) # SplitRatio = percent of sample==TRUE
# Training Data
train = subset(adult, sample == TRUE)
# Testing Data
test = subset(adult, sample == FALSE)
```

#27 Training the Model:
Explore the glm() function with help(glm). Read through the documentation.
help(glm)

```
help(glm)
library(caTools)
```

#28 Use all the features to train a glm() model on the training data set, pass the argument family=binomial(logit) into the glm function.

```
model <- glm(factor(income) ~ ., family = binomial(logit), data = train)
```

#29 If you get a warning, this just means that the model may have guessed the probability of a class with a 0% or 100% chance of occuring.
Check the model summary

```
summary(model)
```

#30 Use new.model <- step(your.model.name) to use the step() function to create a new model.

```
new.step.model <- step(model)
```

#31 You should get a bunch of messages informing you of the process. Check the new.model by using summary()

```
summary(new.step.model)
```

#32 Create a confusion matrix using the predict function with type='response' as an argument inside of that function.

```
test$predicted.income = predict(model, newdata=test, type="response")
table(test$income, test$predicted.income > 0.5)
```

#33 What was the accuracy of our model?

```
(6372+1423)/(6372+1423+548+872)
```

#34 Calculate other measures of performance like, recall or precision.

```
6732/(6372+548)
6732/(6372+872)
```