logistic_regression - Mini Project Homework

Tom Thorpe

Juy 8, 2018

Logistic Regression - Mini Project Homework

```
setwd("C:/Users/Tom/git/datasciencefoundation/ForestCoverage/logistic_regression/mini_project")

##   You might also start by listing the files in your working directory

getwd() # where am I?
```

```
## [1] "C:/Users/Tom/git/datasciencefoundation/ForestCoverage/logistic_regression/mini_project"
```

```
list.files("dataSets") # files in the dataSets folder
```

```
## [1] "NatHealth2008MI"   "NatHealth2011.rds"
```

Regression with binary outcomes

Logistic regression

This far we have used the "lm" function to fit our regression models. "lm" is great, but limited–in particular it only fits models for continuous dependent variables. For categorical dependent variables we can use the "glm()" function.

For these models we will use a different dataset, drawn from the National Health Interview Survey. From the [CDC website]:

The National Health Interview Survey (NHIS) has monitored the health of the nation since 1957. NHIS data on a broad range of health topics are collected through personal household interviews. For over 50 years, the U.S. Census Bureau has been the data collection agent for the National Health Interview Survey. Survey results have been instrumental in providing data to track health status, health care access, and progress toward achieving national health objectives.

Load the National Health Interview Survey data:

```
NH11 <- readRDS("dataSets/NatHealth2011.rds")
labs <- attributes(NH11)$labels
labs
```

```
##         name
## 6        fmx
## 7        fpx
## 8    wtia_sa
## 9    wtfa_sa
## 10    region
## 11   strat_p
## 12     psu_p
## 13       sex
## 14  hispan_i
## 16  mracrpi2
## 18     age_p
```

```
## 19  r_maritl
## 27   everwrk
## 41     hypev
## 49    aasmev
## 51    aasmyr
## 117    dibev
## 119   dibage
## 120  difage2
## 121    insln
## 122  dibpill
## 148    arth1
## 149  arthlmt
## 171   wkdayr
## 172  beddayr
## 205 aflhca18
## 271 aldura10
## 306 aldura17
## 311 aldura18
## 406    smkev
## 416  cigsday
## 423   vigmin
## 429   modmin
## 453      bmi
## 454    sleep
## 455 ausualpl
##                                                                  label
## 6                                                          Family Number
## 7                                          Person Number (Within family)
## 8                                                Weight - Interim Annual
## 9                                                  Weight - Final Annual
## 10                                                               Region
## 11                  Pseudo-stratum for public use file variance estimation
## 12                      Pseudo-PSU for public use file variance estimation
## 13                                                                   Sex
## 14                                          Hispanic subgroup detail
## 16                               Race coded to single/multiple race group
## 18                                                                   Age
## 19                                                     Marital Status
## 27                                                        Ever worked
## 41                                  Ever been told you have hypertension
## 49                                          Ever been told you had asthma
## 51                                   Had an asthma episode/attack past 12 m
## 117                                    Ever been told that you have diabetes
## 119                                           Age first diagnosed w/diabetes
## 120                                  Years since first diagnosed w/diabetes
## 121                                                  NOW taking insulin
## 122                                               NOW taking diabetic pills
## 148                                          Ever been told you had arthritis
## 149                                Limited due to arthritis or joint symptoms
## 171                                  Number of work loss days, past 12 months
## 172                                        Number of bed days, past 12 months
## 205                           Weight problem causes difficulty with activity
## 271                                Duration (in years) of diabetes, recode 1
## 306 Duration (in years) of depression/anxiety/emotional problem, rec ode 1
```

```
## 311                          Duration (in years) of weight problem, recode 1
## 406                                              Ever smoked 100 cigarettes
## 416                         Number of cigarettes a day (all current smokers)
## 423                                 Duration vigorous activity (in minutes)
## 429                            Duration light/moderate activity (in minutes)
## 453                                                 Body Mass Index (BMI)
## 454                                                       Hours of sleep
## 455                                              Place USUALLY go when sick
```

```
# [CDC website] http://www.cdc.gov/nchs/nhis.htm
```

Logistic regression example

Let's predict the probability of being diagnosed with hypertension based on age, sex, sleep, and bmi

```
str(NH11$hypev) # check stucture of hypev
```

```
##  Factor w/ 5 levels "1 Yes","2 No",..: 2 2 1 2 2 1 2 2 1 2 ...
```

```
levels(NH11$hypev) # check levels of hypev
```

```
## [1] "1 Yes"             "2 No"              "7 Refused"
## [4] "8 Not ascertained" "9 Don't know"
```

```
# collapse all missing values to NA
NH11$hypev <- factor(NH11$hypev, levels=c("2 No", "1 Yes"))
# run our regression model
hyp.out <- glm(hypev~age_p+sex+sleep+bmi,
               data=NH11, family="binomial")
coef(summary(hyp.out))
```

```
##                 Estimate   Std. Error    z value     Pr(>|z|)
## (Intercept) -4.269466028 0.0564947294 -75.572820 0.000000e+00
## age_p        0.060699303 0.0008227207  73.778743 0.000000e+00
## sex2 Female -0.144025092 0.0267976605  -5.374540 7.677854e-08
## sleep       -0.007035776 0.0016397197  -4.290841 1.779981e-05
## bmi          0.018571704 0.0009510828  19.526906 6.485172e-85
```

Logistic regression coefficients

Generalized linear models use link functions, so raw coefficients are difficult to interpret. For example, the age coefficient of .06 in the previous model tells us that for every one unit increase in age, the log odds of hypertension diagnosis increases by 0.06. Since most of us are not used to thinking in log odds this is not too helpful!

One solution is to transform the coefficients to make them easier to interpret.

```
hyp.out.tab <- coef(summary(hyp.out))
hyp.out.tab[, "Estimate"] <- exp(coef(hyp.out))
hyp.out.tab
```

```
##                Estimate   Std. Error    z value     Pr(>|z|)
## (Intercept) 0.01398925 0.0564947294 -75.572820 0.000000e+00
## age_p       1.06257935 0.0008227207  73.778743 0.000000e+00
## sex2 Female 0.86586602 0.0267976605  -5.374540 7.677854e-08
## sleep       0.99298892 0.0016397197  -4.290841 1.779981e-05
```

```
## bmi         1.01874523 0.0009510828  19.526906 6.485172e-85
```

Generating predicted values

In addition to transforming the log-odds produced by `glm' to odds, we can use the`predict()' function to make direct statements about the predictors in our model. For example, we can ask "How much more likely is a 63 year old female to have hypertension compared to a 33 year old female?".

Create a dataset with predictors set at desired levels.

```
predDat <- with(NH11,
                expand.grid(age_p = c(33, 63),
                            sex = "2 Female",
                            bmi = mean(bmi, na.rm = TRUE),
                            sleep = mean(sleep, na.rm = TRUE)))
```

Predict hypertension at those levels.

```
cbind(predDat, predict(hyp.out, type = "response",
                       se.fit = TRUE, interval="confidence",
                       newdata = predDat))
```

```
##   age_p      sex      bmi   sleep       fit      se.fit residual.scale
## 1    33 2 Female 29.89565 7.86221 0.1289227 0.002849622              1
## 2    63 2 Female 29.89565 7.86221 0.4776303 0.004816059              1
```

This tells us that a 33 year old female has a 13% probability of having been diagnosed with hypertension, while and 63 year old female has a 48% probability of having been diagnosed.

Packages for computing and graphing predicted values

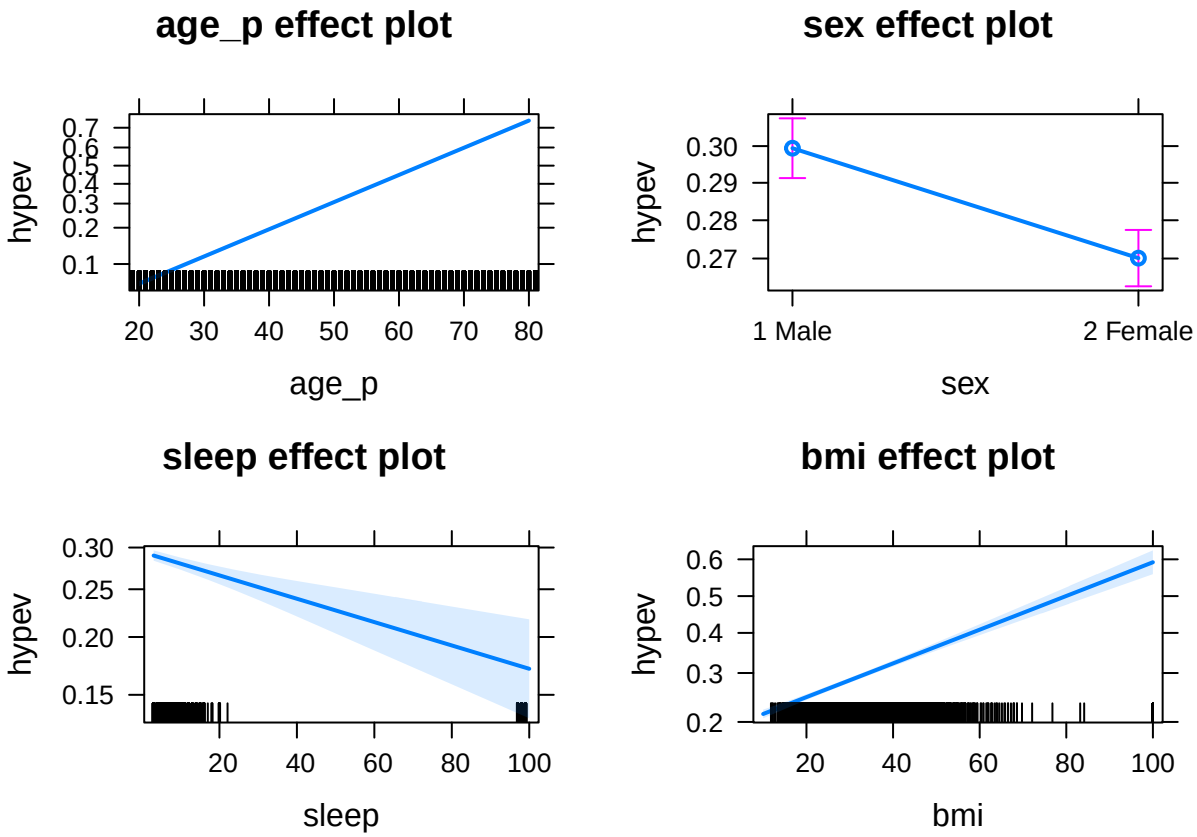
Instead of doing all this ourselves, we can use the effects package to compute quantities of interest for us (cf. the Zelig package).

```
library(effects)
```

```
## Loading required package: carData
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
plot(allEffects(hyp.out))
```



Exercise: logistic regression

Use the NH11 data set that we loaded earlier.

1. Use glm to conduct a logistic regression to predict ever worked (everwrk) using age (age_p) and marital status (r_maritl).

Note: that the data is not perfectly clean and ready to be modeled. You will need to clean up at least some of the variables before fitting the model.

Data Cleaning

To begin, we look at the data to determine what needs to be cleaned.

```
str(NH11)
```

```
## 'data.frame':    33014 obs. of  36 variables:
##  $ fmx     : chr  "01" "01" "01" "01" ...
##  $ fpx     : chr  "03" "03" "01" "01" ...
##  $ wtia_sa : num  7521 5784 2512 3086 12530 ...
##  $ wtfa_sa : num  8814 10427 2791 3888 16609 ...
##  $ region  : num  3 3 1 3 3 1 3 3 3 3 ...
##  $ strat_p : num  223 201 3 166 125 31 190 190 217 173 ...
##  $ psu_p   : num  1 2 1 1 2 1 1 1 1 1 ...
##  $ sex     : Factor w/ 2 levels "1 Male","2 Female": 2 2 2 2 2 2 2 2 1 1 ...
```

```
##  $ hispan_i: Factor w/ 13 levels "00 Multiple Hispanic",..: 13 13 13 13 13 13 7 13 13 13 ...
##  $ mracrpi2: Factor w/ 9 levels "01 White","02 Black/African American",..: 1 2 2 2 1 1 1 1 2 1 ...
##  $ age_p   : num  47 18 79 51 43 41 21 20 33 56 ...
##  $ r_maritl: Factor w/ 10 levels "0 Under 14 years",..: 6 8 5 7 2 2 8 8 8 2 ...
##  $ everwrk : Factor w/ 5 levels "1 Yes","2 No",..: NA NA 1 NA NA NA NA NA 1 1 ...
##  $ hypev   : Factor w/ 2 levels "2 No","1 Yes": 1 1 2 1 1 2 1 1 2 1 ...
##  $ aasmev  : Factor w/ 5 levels "1 Yes","2 No",..: 1 2 2 2 2 2 2 2 2 2 ...
##  $ aasmyr  : Factor w/ 5 levels "1 Yes","2 No",..: 1 NA NA NA NA NA NA NA NA NA ...
##  $ dibev   : Factor w/ 6 levels "1 Yes","2 No",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ dibage  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ difage2 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ insln   : Factor w/ 5 levels "1 Yes","2 No",..: 2 NA NA NA NA NA NA NA NA NA ...
##  $ dibpill : Factor w/ 5 levels "1 Yes","2 No",..: 2 NA NA NA NA NA NA NA NA NA ...
##  $ arth1   : Factor w/ 5 levels "1 Yes","2 No",..: 1 2 1 2 2 1 2 2 1 2 ...
##  $ arthlmt : Factor w/ 5 levels "1 Yes","2 No",..: 2 NA 1 NA NA 2 NA 2 2 NA ...
##  $ wkdayr  : num  3 0 NA 0 1 0 0 1 NA 0 ...
##  $ beddayr : num  3 0 0 0 1 0 0 0 0 0 ...
##  $ aflhca18: Factor w/ 5 levels "1 Mentioned",..: 2 NA 2 NA NA 2 2 NA 2 NA ...
##  $ aldura10: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ aldura17: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ aldura18: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ smkev   : Factor w/ 5 levels "1 Yes","2 No",..: 2 2 2 1 3 2 2 2 2 1 ...
##  $ cigsday : num  NA NA NA 5 NA NA NA NA NA NA ...
##  $ vigmin  : num  NA NA NA NA NA 60 120 30 NA 120 ...
##  $ modmin  : num  15 NA 10 NA NA 30 30 120 NA 45 ...
##  $ bmi     : num  100 21.6 32.3 100 100 ...
##  $ sleep   : num  6 8 6 8 9 8 7 6 10 8 ...
##  $ ausualpl: Factor w/ 6 levels "1 Yes","2 There is NO place",..: 1 2 1 2 1 1 1 2 1 1 ...
##  - attr(*, "labels")='data.frame':   36 obs. of  2 variables:
##   ..$ name : Factor w/ 591 levels "aaseryr1","aasmev",..: 452 453 590 589 538 567 534 541 455 520 ..
##   ..$ label: Factor w/ 590 levels " AAU.050_01.010: Doesn't need doctor/haven't had problems",..: 35
```

The dependent variable, 'everwrk', is a factor with 5 possibilities that need to be converted to a binary outcome. Examine the 'everwrk' variable more closely to see all the values.

```
summary(NH11$everwrk)
```

```
##             1 Yes              2 No         7 Refused 8 Not ascertained
##             12153              1887                17                 0
##      9 Don't know              NA's
##                 8             18949
```

```
table(NH11$everwrk)
```

```
##
##             1 Yes              2 No         7 Refused 8 Not ascertained
##             12153              1887                17                 0
##      9 Don't know
##                 8
```

The 'everwrk' variable can be converted to numeric by subscrpting just the first character and converting to numeric.

```
NH11$everwrk=as.character(NH11$everwrk) # so that we can set new values.
NH11$everwrk=substr(NH11$everwrk,1,1)   # get the first character for the value
NH11$everwrk=as.integer(NH11$everwrk)   # convert to numeric
```

Make the "No" value zero so that the outcome will have "0" meaning "never worked" and "1" for "having worked".

```
NH11$everwrk[NH11$everwrk == 2]<-0
table(NH11$everwrk)
```

```
##
##     0     1     7     9
##  1887 12153    17     8
```

There are only 25 rows that are not "1" or "0". Remove them from the study by setting them to "NA".

```
NH11$everwrk[NH11$everwrk==7 | NH11$everwrk==9]=NA
#NH11_complete = NH11[NH11$everwrk!=7 & NH11$everwrk!=9 , ]
table(NH11$everwrk)
```

```
##
##     0     1
##  1887 12153
```

The "everwrk" outcome variable is now ready.

Look at "age_p" variable again. It is ready as a numeric independent variable without any "NA" entries.

```
summary(NH11$age_p)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   33.00   47.00   48.11   62.00   85.00
```

```
table(NH11$age_p)
```

```
##
##  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
## 329 443 467 443 505 572 532 586 581 640 612 612 622 626 595 598 596 609
##  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
## 607 510 538 568 635 537 566 534 571 567 564 582 600 596 584 606 531 595
##  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
## 534 558 551 519 516 548 549 541 533 491 513 472 425 410 411 395 341 326
##  72  73  74  75  76  77  78  79  80  81  82  83  84  85
## 340 287 296 252 277 252 242 253 234 195 212 181 177 924
```

Check "r_maritl", marital status next. Marital status does not have any "NA" values.

```
summary(NH11$r_maritl)
```

```
##                          0 Under 14 years
##                                         0
##          1 Married - spouse in household
##                                     13943
##      2 Married - spouse not in household
##                                       531
## 3 Married - spouse in household unknown
##                                         0
##                                  4 Widowed
##                                       3069
##                                 5 Divorced
##                                       4511
##                                6 Separated
##                                       1121
##                           7 Never married
```

```
##                                    7763
##                    8 Living with partner
##                                    2002
##                 9 Unknown marital status
##                                      74
```

The r_maritl variable looks good as is for use as a factor variable for the logistic regression. Look at other ways to represent the marital status, converting the value to a number by subscripting the first character of the value and another numeric by ordering the priority from unmarried to married as follows:

- Under 14 years
- Never Married
- Unknown Marital Status
- Widowed
- Divorced
- Separated
- Living with partner
- Married - Spouse in household unknown
- Married - Spouse not in household
- Married - Spouse in household

The values will be placed into new variables, "maritlNum" and maritlPri respectively.

```
# Create binary variables for marriage types for possible later use
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
#NH11 <- mutate(NH11,r_maritlNum = 0) # Marital as numeric values
#NH11 <- mutate(NH11,r_maritlPri = 0) # Marital as numeric values, prioritized from unmarried to marrie

#NH11$r_maritlNum=substr(NH11$r_maritl,1,1)

#NH11$r_maritlPri[NH11$r_maritl=="0 Under 14 years"]<-0
#NH11$r_maritlPri[NH11$r_maritl=="7 Never married"]<-1
#NH11$r_maritlPri[NH11$r_maritl=="9 Unknown marital status"]<-2
#NH11$r_maritlPri[NH11$r_maritl=="4 Widowed"]<-3
#NH11$r_maritlPri[NH11$r_maritl=="5 Divorced"]<-4
#NH11$r_maritlPri[NH11$r_maritl=="6 Separated"]<-5
#NH11$r_maritlPri[NH11$r_maritl=="8 Living with partner"]<-6
#NH11$r_maritlPri[NH11$r_maritl=="3 Married - spouse in household unknown"]<-7
#NH11$r_maritlPri[NH11$r_maritl=="2 Married - spouse not in household"]<-8
#NH11$r_maritlPri[NH11$r_maritl=="1 Married - spouse in household"]<-9

#NH11$r_maritl=as.numeric(NH11$r_maritl)
#print(summary(NH11$r_maritlNum))
#print(table(NH11$r_maritlNum))
#print(summary(NH11$r_maritlPri))
#print(table(NH11$r_maritlPri))
```

The data is cleaned except for the "NA" values for "everwrk".

To handle the "NA" values, two new datasets will be created. One will have the records with the approximately 18000 "everwrk"="NA" records removed so there are only complete cases. This data set will be called, "NH11_complete". It will contain about 14000 records which should be sufficient for training and testing.

The second data set will have the "NA" values in "everwrk" updated using the impute method. This dataset will be called NH11_imputed and will contain the full data set. An intermediate dataset with factors converted to numeric will be created for use by the imputing method. It will be called, "NH11_numeric".

The coefficients for the logistic regression for each model will be compared to see if there is any major difference from the two methods.

Create the NH11_complete dataset by removing records that have "everwrk" values of "NA". Then look at the data again.

```
NH11_complete=NH11[complete.cases(NH11[ , "everwrk"]),] # remove everwrk="NA" rows
NH11_complete$everwrk=as.integer(NH11_complete$everwrk)
str(NH11_complete)
```

```
## 'data.frame':    14040 obs. of  36 variables:
##  $ fmx     : chr  "01" "01" "01" "01" ...
##  $ fpx     : chr  "01" "01" "01" "03" ...
##  $ wtia_sa : num  2512 5856 7992 6414 5270 ...
##  $ wtfa_sa : num  2791 10095 10030 8863 5715 ...
##  $ region  : num  1 3 3 3 3 1 3 4 1 3 ...
##  $ strat_p : num  3 217 173 154 213 39 230 268 30 151 ...
##  $ psu_p   : num  1 1 1 1 1 1 2 2 2 1 ...
##  $ sex     : Factor w/ 2 levels "1 Male","2 Female": 2 1 1 2 1 2 1 1 1 2 ...
##  $ hispan_i: Factor w/ 13 levels "00 Multiple Hispanic",..: 13 13 13 13 13 2 13 13 13 2 ...
##  $ mracrpi2: Factor w/ 9 levels "01 White","02 Black/African American",..: 2 2 1 2 2 1 4 1 2 1 ...
##  $ age_p   : num  79 33 56 27 44 33 22 47 54 21 ...
##  $ r_maritl: Factor w/ 10 levels "0 Under 14 years",..: 5 8 2 8 6 2 8 6 6 8 ...
##  $ everwrk : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hypev   : Factor w/ 2 levels "2 No","1 Yes": 2 2 1 2 2 1 1 2 2 1 ...
##  $ aasmev  : Factor w/ 5 levels "1 Yes","2 No",..: 2 2 2 1 2 2 2 2 2 2 ...
##  $ aasmyr  : Factor w/ 5 levels "1 Yes","2 No",..: NA NA NA 1 NA NA NA NA NA NA ...
##  $ dibev   : Factor w/ 6 levels "1 Yes","2 No",..: 2 2 2 2 1 2 2 2 1 2 ...
##  $ dibage  : num  NA NA NA NA 37 NA NA NA 53 NA ...
##  $ difage2 : num  NA NA NA NA 7 NA NA NA 1 NA ...
##  $ insln   : Factor w/ 5 levels "1 Yes","2 No",..: NA NA NA NA 1 NA NA NA 2 NA ...
##  $ dibpill : Factor w/ 5 levels "1 Yes","2 No",..: NA NA NA NA 2 NA NA NA 2 NA ...
##  $ arth1   : Factor w/ 5 levels "1 Yes","2 No",..: 1 1 2 2 1 2 2 2 1 2 ...
##  $ arthlmt : Factor w/ 5 levels "1 Yes","2 No",..: 1 2 NA NA 1 NA NA NA 1 NA ...
##  $ wkdayr  : num  NA NA 0 0 NA NA NA NA NA NA ...
##  $ beddayr : num  0 0 0 0 30 1 0 0 20 2 ...
##  $ aflhca18: Factor w/ 5 levels "1 Mentioned",..: 2 2 NA NA 2 NA NA 2 2 NA ...
##  $ aldura10: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ aldura17: num  NA NA NA NA NA NA NA NA 6 NA ...
##  $ aldura18: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ smkev   : Factor w/ 5 levels "1 Yes","2 No",..: 2 2 1 2 1 1 2 2 1 2 ...
##  $ cigsday : num  NA NA NA NA 20 NA NA NA 10 NA ...
##  $ vigmin  : num  NA NA 120 NA NA NA 60 NA NA 60 ...
##  $ modmin  : num  10 NA 45 NA 10 NA 60 NA NA NA ...
##  $ bmi     : num  32.3 24.8 24.4 29.3 32.4 ...
```

```
##  $ sleep   : num  6 10 8 6 3 7 8 7 5 7 ...
##  $ ausualpl: Factor w/ 6 levels "1 Yes","2 There is NO place",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "labels")='data.frame':   36 obs. of  2 variables:
##   ..$ name : Factor w/ 591 levels "aaseryr1","aasmev",..: 452 453 590 589 538 567 534 541 455 520 ..
##   ..$ label: Factor w/ 590 levels " AAU.050_01.010: Doesn't need doctor/haven't had problems",..: 35
```

```
summary(NH11_complete$everwrk)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  1.0000  1.0000  0.8656  1.0000  1.0000
```

```
table(NH11_complete$everwrk)
```

```
##
##     0     1
##  1887 12153
```

Complete Imputed Values

Create an intermedate data frame of numeric data to use as a source to create the imputed values for the entire data set.

The variables to include in the numeric data frame are selected. All variables except family and person number are included.

```
do_impute=1

if (do_impute) {
  NH11_numeric=NH11[c("wtia_sa",
"wtfa_sa",
"region",
"strat_p",
"psu_p",
"sex",
"hispan_i",
"mracrpi2",
"age_p",
"r_maritl",
"everwrk",
"hypev",
"aasmev",
"aasmyr",
"dibev",
"dibage",
"difage2",
"insln",
"dibpill",
"arth1",
"arthlmt",
"wkdayr",
"beddayr",
"aflhca18",
"aldura10",
"aldura17",
"aldura18",
"smkev",
```

```
"cigsday",
"vigmin",
"modmin",
"bmi",
"sleep",
"ausualpl"
)]

  curTime=Sys.time()
  print(paste("Numeric Conversion started at",curTime))

# Convert the data to numeric as we did before

NH11_numeric$sex=as.character(NH11_numeric$sex)
NH11_numeric$sex=substr(NH11_numeric$sex,1,1)
NH11_numeric$sex=as.numeric(NH11_numeric$sex)
NH11_numeric$sex[NH11_numeric$sex == 2]<-0
table(NH11_numeric$sex)

NH11_numeric$hispan_i=as.character(NH11_numeric$hispan_i)
NH11_numeric$hispan_i=substr(NH11_numeric$hispan_i,1,2)
NH11_numeric$hispan_i=as.numeric(NH11_numeric$hispan_i)
table(NH11_numeric$hispan_i)

NH11_numeric$mracrpi2=as.character(NH11_numeric$mracrpi2)
NH11_numeric$mracrpi2=substr(NH11_numeric$mracrpi2,1,2)
NH11_numeric$mracrpi2=as.numeric(NH11_numeric$mracrpi2)
table(NH11_numeric$mracrpi2)

NH11_numeric$hypev=as.numeric(NH11_numeric$hypev)
NH11_numeric$hypev=substr(NH11_numeric$hypev,1,1)
NH11_numeric$hypev=as.numeric(NH11_numeric$hypev)
table(NH11_numeric$hypev)

NH11_numeric$aasmev=as.numeric(NH11_numeric$aasmev)
NH11_numeric$aasmev=substr(NH11_numeric$aasmev,1,1)
NH11_numeric$aasmev=as.numeric(NH11_numeric$aasmev)
table(NH11_numeric$aasmev)

NH11_numeric$aasmyr=as.numeric(NH11_numeric$aasmyr)
NH11_numeric$aasmyr=substr(NH11_numeric$aasmyr,1,1)
NH11_numeric$aasmyr=as.numeric(NH11_numeric$aasmyr)
table(NH11_numeric$aasmyr)

NH11_numeric$dibev=as.numeric(NH11_numeric$dibev)
NH11_numeric$dibev=substr(NH11_numeric$dibev,1,1)
NH11_numeric$dibev=as.numeric(NH11_numeric$dibev)
table(NH11_numeric$dibev)

NH11_numeric$insln=as.numeric(NH11_numeric$insln)
NH11_numeric$insln=substr(NH11_numeric$insln,1,1)
NH11_numeric$insln=as.numeric(NH11_numeric$insln)
table(NH11_numeric$insln)
```

```
NH11_numeric$dibpill=as.numeric(NH11_numeric$dibpill)
NH11_numeric$dibpill=substr(NH11_numeric$dibpill,1,1)
NH11_numeric$dibpill=as.numeric(NH11_numeric$dibpill)
table(NH11_numeric$dibpill)

NH11_numeric$arth1=as.numeric(NH11_numeric$arth1)
NH11_numeric$arth1=substr(NH11_numeric$arth1,1,1)
NH11_numeric$arth1=as.numeric(NH11_numeric$arth1)
table(NH11_numeric$arth1)

NH11_numeric$arthlmt=as.numeric(NH11_numeric$arthlmt)
NH11_numeric$arthlmt=substr(NH11_numeric$arthlmt,1,1)
NH11_numeric$arthlmt=as.numeric(NH11_numeric$arthlmt)
table(NH11_numeric$arthlmt)

NH11_numeric$aflhca18=as.numeric(NH11_numeric$aflhca18)
NH11_numeric$aflhca18=substr(NH11_numeric$aflhca18,1,1)
NH11_numeric$aflhca18=as.numeric(NH11_numeric$aflhca18)
table(NH11_numeric$aflhca18)

NH11_numeric$smkev=as.numeric(NH11_numeric$smkev)
NH11_numeric$smkev=substr(NH11_numeric$smkev,1,1)
NH11_numeric$smkev=as.numeric(NH11_numeric$smkev)
table(NH11_numeric$smkev)

NH11_numeric$ausualpl=as.numeric(NH11_numeric$ausualpl)
NH11_numeric$ausualpl=substr(NH11_numeric$ausualpl,1,1)
NH11_numeric$ausualpl=as.numeric(NH11_numeric$ausualpl)
table(NH11_numeric$ausualpl)

str(NH11_numeric)
summary(NH11_numeric)

set.seed(144)
library(mice)

curTime=Sys.time()
print(paste("Numeric conversion completed and data completion started at",curTime))

NH11_imputed=complete(mice(NH11_numeric))

curTime=Sys.time()
print(paste("Data Impute completed at",curTime))

summary(NH11_imputed)

} else {
  NH11_imputed = NH11_complete
}
```

```
## [1] "Numeric Conversion started at 2018-07-18 19:52:58"
## 'data.frame':    33014 obs. of  34 variables:
##  $ wtia_sa : num  7521 5784 2512 3086 12530 ...
```

```
##  $ wtfa_sa : num  8814 10427 2791 3888 16609 ...
##  $ region  : num  3 3 1 3 3 1 3 3 3 3 ...
##  $ strat_p : num  223 201 3 166 125 31 190 190 217 173 ...
##  $ psu_p   : num  1 2 1 1 2 1 1 1 1 1 ...
##  $ sex     : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ hispan_i: num  12 12 12 12 12 12 6 12 12 12 ...
##  $ mracrpi2: num  1 2 2 2 1 1 1 1 2 1 ...
##  $ age_p   : num  47 18 79 51 43 41 21 20 33 56 ...
##  $ r_maritl: Factor w/ 10 levels "0 Under 14 years",..: 6 8 5 7 2 2 8 8 8 2 ...
##  $ everwrk : num  NA NA 1 NA NA NA NA NA 1 1 ...
##  $ hypev   : num  1 1 2 1 1 2 1 1 2 1 ...
##  $ aasmev  : num  1 2 2 2 2 2 2 2 2 2 ...
##  $ aasmyr  : num  1 NA NA NA NA NA NA NA NA NA ...
##  $ dibev   : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ dibage  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ difage2 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ insln   : num  2 NA NA NA NA NA NA NA NA NA ...
##  $ dibpill : num  2 NA NA NA NA NA NA NA NA NA ...
##  $ arth1   : num  1 2 1 2 2 1 2 2 1 2 ...
##  $ arthlmt : num  2 NA 1 NA NA 2 NA 2 2 NA ...
##  $ wkdayr  : num  3 0 NA 0 1 0 0 1 NA 0 ...
##  $ beddayr : num  3 0 0 0 1 0 0 0 0 0 ...
##  $ aflhca18: num  2 NA 2 NA NA 2 2 NA 2 NA ...
##  $ aldura10: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ aldura17: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ aldura18: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ smkev   : num  2 2 2 1 3 2 2 2 2 1 ...
##  $ cigsday : num  NA NA NA 5 NA NA NA NA NA NA ...
##  $ vigmin  : num  NA NA NA NA NA 60 120 30 NA 120 ...
##  $ modmin  : num  15 NA 10 NA NA 30 30 120 NA 45 ...
##  $ bmi     : num  100 21.6 32.3 100 100 ...
##  $ sleep   : num  6 8 6 8 9 8 7 6 10 8 ...
##  $ ausualpl: num  1 2 1 2 1 1 1 2 1 1 ...
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
## [1] "Numeric conversion completed and data completion started at 2018-07-18 19:53:00"
##
##  iter imp variable
##   1   1  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   1   2  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   1   3  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   1   4  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   1   5  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   2   1  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   2   2  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   2   3  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   2   4  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   2   5  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
```

```
##   3   1  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   3   2  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   3   3  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   3   4  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   3   5  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   4   1  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   4   2  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   4   3  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   4   4  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   4   5  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   5   1  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   5   2  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   5   3  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   5   4  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1
##   5   5  everwrk  hypev  aasmyr  dibage  difage2  insln  dibpill  arthlmt  wkdayr  aflhca18  aldura1

## Warning: Number of logged events: 400

## [1] "Data Impute completed at 2018-07-18 20:21:43"

##     wtia_sa           wtfa_sa          region         strat_p
##  Min.   :  780.2   Min.   :  846   Min.   :1.000   Min.   :  1
##  1st Qu.: 2933.3   1st Qu.: 3613   1st Qu.:2.000   1st Qu.: 82
##  Median : 4494.4   Median : 5612   Median :3.000   Median :157
##  Mean   : 5607.1   Mean   : 7008   Mean   :2.713   Mean   :155
##  3rd Qu.: 7278.1   3rd Qu.: 9026   3rd Qu.:4.000   3rd Qu.:233
##  Max.   :65211.6   Max.   :71281   Max.   :4.000   Max.   :300
##
##      psu_p           sex            hispan_i        mracrpi2
##  Min.   :1.00   Min.   :0.0000   Min.   : 0.00   Min.   : 1.000
##  1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:12.00   1st Qu.: 1.000
##  Median :1.00   Median :0.0000   Median :12.00   Median : 1.000
##  Mean   :1.49   Mean   :0.4486   Mean   :10.42   Mean   : 1.984
##  3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:12.00   3rd Qu.: 1.000
##  Max.   :2.00   Max.   :1.0000   Max.   :12.00   Max.   :17.000
##
##      age_p                                  r_maritl        everwrk
##  Min.   :18.00   1 Married - spouse in household:13943   Min.   :0.0000
##  1st Qu.:33.00   7 Never married                : 7763   1st Qu.:1.0000
##  Median :47.00   5 Divorced                    : 4511   Median :1.0000
##  Mean   :48.11   4 Widowed                     : 3069   Mean   :0.8537
##  3rd Qu.:62.00   8 Living with partner         : 2002   3rd Qu.:1.0000
##  Max.   :85.00   6 Separated                   : 1121   Max.   :1.0000
##                  (Other)                       :  605
##      hypev           aasmev          aasmyr          dibev
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:2.000
##  Median :1.000   Median :2.000   Median :2.000   Median :2.000
##  Mean   :1.324   Mean   :1.878   Mean   :1.717   Mean   :1.919
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :5.000   Max.   :5.000   Max.   :6.000
##
##      dibage         difage2          insln          dibpill
##  Min.   : 1.00   Min.   : 0.00   Min.   :1.000   Min.   :1.000
##  1st Qu.:25.00   1st Qu.: 2.00   1st Qu.:2.000   1st Qu.:2.000
```

```
##  Median :40.00   Median : 6.00   Median :2.000   Median :2.000
##  Mean   :40.48   Mean   :11.02   Mean   :1.889   Mean   :1.766
##  3rd Qu.:55.00   3rd Qu.:13.00   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :99.00   Max.   :99.00   Max.   :5.000   Max.   :5.000
##
##      arth1          arthlmt          wkdayr          beddayr
##  Min.   :1.000   Min.   :1.000   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:  0.00   1st Qu.:  0.00
##  Median :2.000   Median :2.000   Median :  0.00   Median :  0.00
##  Mean   :1.756   Mean   :1.733   Mean   : 10.78   Mean   : 11.25
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:  3.00   3rd Qu.:  2.00
##  Max.   :5.000   Max.   :5.000   Max.   :999.00   Max.   :999.00
##
##     aflhca18        aldura10        aldura17        aldura18
##  Min.   :1.000   Min.   : 0.000   Min.   : 0.00   Min.   : 0.00
##  1st Qu.:2.000   1st Qu.: 2.000   1st Qu.: 4.00   1st Qu.: 3.00
##  Median :2.000   Median : 6.000   Median :10.00   Median :10.00
##  Mean   :1.988   Mean   : 9.096   Mean   :16.82   Mean   :15.95
##  3rd Qu.:2.000   3rd Qu.:12.000   3rd Qu.:21.00   3rd Qu.:20.00
##  Max.   :5.000   Max.   :99.000   Max.   :99.00   Max.   :99.00
##
##      smkev          cigsday          vigmin          modmin
##  Min.   :1.000   Min.   : 1.00   Min.   : 10.00   Min.   : 10.00
##  1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 30.00   1st Qu.: 20.00
##  Median :2.000   Median :10.00   Median : 45.00   Median : 30.00
##  Mean   :1.597   Mean   :12.65   Mean   : 61.84   Mean   : 57.11
##  3rd Qu.:2.000   3rd Qu.:20.00   3rd Qu.: 60.00   3rd Qu.: 60.00
##  Max.   :5.000   Max.   :99.00   Max.   :999.00   Max.   :999.00
##
##       bmi            sleep          ausualpl
##  Min.   :11.81   Min.   : 3.000   Min.   :1.000
##  1st Qu.:23.57   1st Qu.: 6.000   1st Qu.:1.000
##  Median :26.76   Median : 7.000   Median :1.000
##  Mean   :29.90   Mean   : 7.862   Mean   :1.188
##  3rd Qu.:31.31   3rd Qu.: 8.000   3rd Qu.:1.000
##  Max.   :99.99   Max.   :99.000   Max.   :6.000
##
```

It shows there were 5 rounds/iterations of imputation to fill in the missing values.

The summary shows there are no missing values.

Create Logistic Model Analysis Function

Before creating logistic regression models, a function that will make it easier to analyze the accuracy of models for a range of thresholds is created. It can produce a "Confusion Matrix", calculate the Sensitivity, specificity and accuracy of a model.

```
calcLogisticModelAccuracy <- function(actualValues, predictedValues,
                          thresholdStart, thresholdEnd, thresholdParts,
                          positiveLabel, negativeLabel, printLevel) {

  # Description
  #   -Calculate accuracy of logistic regression model
```

```
#   -depending on print level option:
#      print accuracy of logistic model and baseline model
#      print confusion matrix
#      print sensitivity and specificty
#
# Input Values
#   -actualValues = actual values of outcome variables, a vector of 0's and 1's
#   -predictedValues = logistic model predicted probalilities between 0 and 1
#   -thresholdStart = threshold initial value for applying to predicted values
#      to determine predicted outcome
#   -thresholdEnd = end value for incrementing the threshold
#   -thresholdParts = number of partitions to apply threshold values between
#      thresholdStart and thresholdEnd
#   -positiveLabel = text to label true outcomes. This will be displayed
#      on the confusion matrix when the print level is greater than 1.
#   -negativeLable = text to label false outcomes. This will be displayed
#      on the confusion matrix when the print level is greater than 1.
#   -printLevel = level of detail printed by calcLogisticModelAccuracy
#      0 - no printed output unless and error is encountered
#      1 - print threshold, logistic model accuracy and baseline accuracy
#      2 - Print level 1 and confusion matrix and sensitivity and specificity values
#      3 - Print level 2 and details of sensitivity and specificity calculations
#      4 - Print level 3 and debug information
#
# Return Values
#   -function status:
#      - "OK":function completed without errors
#      - "ERROR": function did not complete, and error information
#         See other variables for possible additional error information
#   -logistic model accuracy based on last threshold value tested
#   -baseline model accuracy based on last threshold value tested
#   -confusion matrix values in following order: TN, FN, FP, TP
#   -sensitivity
#   -specificity

# set default values in case of errors
accuracy=baseline=retVal="ERROR"

# Calculate increment value to iterate through the threshold values
if ( thresholdParts ==0) { thresholdParts = 1 }
if ( thresholdParts < 0) { thresholdParts = - thresholdParts }
thresholdInc = (thresholdEnd - thresholdStart) / thresholdParts
if (thresholdStart==thresholdEnd | thresholdParts < 2) {
  thresholdEnd=thresholdStart
  thresholdInc=1
}
threshold=thresholdStart

funcStat="OK"

workPerformance = table(actualValues, predictedValues > threshold)

for (row in rownames(workPerformance)) {
```

```
  if(row != "0" & row != "1") {
    funcStat=paste("ERROR:Bad row name:",row,",must be '0' or '1'")}
}
for (col in colnames(workPerformance)) {
  if(col != "TRUE" & col != "FALSE") {
    funcStat=paste("ERROR:Bad column name:",col,", must be 'TRUE' or 'FALSE'")}
}

if (funcStat=="OK") {
  repeat {

    workPerformance = table(actualValues, predictedValues > threshold)

    # create a modelPerformance table and set all the values to zero.
    # This ensures a 2x2 matrix in case the threshold causes all values predicted
    # to be TRUE or FALSE values and produces a 2x1 vector.
    # The table of actual and predicted values with be copied into the
    # modelPerformance table later.
    Actual = c(0, 1)
    Predicted = c(FALSE, TRUE )
    modelPerformance = table(Actual,Predicted)
    modelPerformance["0","TRUE"]=0
    modelPerformance["0","FALSE"]=0
    modelPerformance["1","FALSE"]=0
    modelPerformance["1","TRUE"]=0

    # Descriptions         | Predict Good Care (0) | Predict Poor Care (1)
    # ---------------------|-----------------------|----------------------
    # Actual Good Care (0) |     TN (true neg)     |   FP (false pos)
    # Actual Poor Care (1) |     FN (false neg)    |   TP (true pos)

    # Remember: 0 means negative which means Good care,
    #           1 means positive which means Poor care
    #   (Opposite of intuition)

    # Sensitivity = TP / (TP + FN) = percent of true positives identified

    # Specificity = TN / (TN + FP) = percent of true negatives identified

    # transfer the workPerformance table to the final performance table
    for (row in rownames(workPerformance)) {
      for (col in colnames(workPerformance)) {
        modelPerformance[row,col]=workPerformance[row,col]
        if (printLevel > 3) { print(paste("workPerformance[",row,",",col,"]=",
                                    workPerformance[row,col]))}
      }
    }

    if (printLevel > 3) {print(modelPerformance) }

    #                   Actual,Prediction
    TP = modelPerformance["1","TRUE"]  # Predicted True (1),  and actually TRUE (1) = True Positive
    FN = modelPerformance["1","FALSE"] # Predicted False (0), but actually TRUE (1) = False 0/Negativ
```

```
TN = modelPerformance["0","FALSE"] # Predicted False (0), and actually False (0) = True Negative
FP = modelPerformance["0","TRUE"] # Predicted True (1), but actually False (0) = False 1/Positive

# Prevent and report divide by zero error
if (TP+FN == 0) {
  sensitivity="ERROR:TP+FN=0"
  funcStat=sensitivity
} else { sensitivity = TP / (TP + FN ) }

# Prevent and report divide by zero error
if (TN+FP == 0) {
  specificity="ERROR:TN+FP=0"
  funcStat=specificity
} else { specificity = TN / (TN + FP) }

retVal = c(modelPerformance, sensitivity,specificity) # TN, FN, FP, TP, sens, spec

if (printLevel > 2) {
  modelPerformance["1","TRUE"]  = paste("   ",modelPerformance["1","TRUE"], "(TP)")
  modelPerformance["1","FALSE"] = paste("   ",modelPerformance["1","FALSE"],"(FN)")
  modelPerformance["0","FALSE"] = paste("   ",modelPerformance["0","FALSE"],"(TN)")
  modelPerformance["0","TRUE"]  = paste("   ",modelPerformance["0","TRUE"], "(FP)")
}

c1=paste("FALSE=Predict:",negativeLabel,sep="")
c2=paste("TRUE=Predict:",positiveLabel,sep="")
r1=paste("0=Actual:",negativeLabel,sep="")
r2=paste("1=Actual:",positiveLabel,sep="")
colnames(modelPerformance) <- c(c1,c2)
rownames(modelPerformance) <- c(r1,r2)

if (printLevel > 1) {
  print(paste("Model Performance for threshold=", threshold))
  print("predicted performance=")
  print(modelPerformance)

  sensPrint=paste("Sensitivity=",sensitivity,"(True positive rate of",positiveLabel)

  specPrint=paste("Specificity=",specificity,"(True negative rate of",negativeLabel)

  if (printLevel > 2 ) {
    sensPrint=paste(sensPrint,"= TP/(TP+FN) =",TP,"/(",TP,"+",FN,"))")
    specPrint=paste(specPrint,"= TN/(TN+FP) =",TN,"/(",TN,"+",FP,"))")
  }

  print(sensPrint)
  print(specPrint)
}

# Calculate actual true and actual false totals to calculate baseline accuracy
# and logistic model accuracy
totSamples=TP+FN+TN+FP
actTrue=TP+FN
```

```
      actFalse=TN+FP

      # double check there were actually some non-zero values
      if (totSamples>0) {
        if (actTrue > actFalse) {
          baseline = actTrue / totSamples
          baseModel= positiveLabel
        } else {
          baseline = actFalse / totSamples
          baseModel=negativeLabel
        }

        # the accuracy is the number of TRUE positives and True negatives
        # divided by the number of samples
        accuracy=(TP+TN)/totSamples
      } else {
        baseModel="ERROR:0 samples"
        baseline="ERROR:0 samples"
        accuracy="ERROR:0 samples"
        funcStat=accuracy
      }

      if (printLevel > 0) {

        printAcc=(as.integer(accuracy*1000000))/1000000
        printbaseline=(as.integer(baseline*1000000))/1000000

        print(paste("Threshold=",threshold,", Logistic Accuracy=",printAcc,
                ", Baseline (",baseModel,") Accuracy=",printbaseline,sep=""))
      }

      c(funcStat,accuracy,baseline,retVal)

      #print(paste("threshold=",threshold,",End=",thresholdEnd,",Inc=",thresholdInc))

      threshold=threshold+thresholdInc
      if(thresholdEnd < thresholdStart) {
        if (threshold < thresholdEnd) { break}
      } else { if (threshold > thresholdEnd) { break} }

    }
  } else {
    # Had an error, just return the error information
    print(funcStat)
  }

  c(funcStat,accuracy,baseline,retVal)
}
```

Create Logistic Regression model for NH11_complete Data

Returning back to the exercise now that the data has been cleaned and NA values imputed.

1. Use glm to conduct a logistic regression to predict ever worked (everwrk) using age (age_p) and marital status (r_maritl).

Split data into training and testing data for the NH11_complete data.

```
library(caTools)
set.seed(144)
split = sample.split(NH11_complete$everwrk, 0.65) # we want 65% in the training set
NH11_complete_train = subset(NH11_complete, split == TRUE)
NH11_complete_test  = subset(NH11_complete, split == FALSE)
```

There are 9126 rows in the training set and 4914 rows in the testing set.

Create Logistic Model for complete data

Now create the logistic model for the NH11_complete data.

```
NH11_comp_ModelLog = glm(everwrk ~ age_p + r_maritl , data=NH11_complete_train, family=binomial)
```

Let's check the coefficients.

```
summary(NH11_comp_ModelLog)
```

```
##
## Call:
## glm(formula = everwrk ~ age_p + r_maritl, family = binomial,
##     data = NH11_complete_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7714   0.3264   0.4512   0.5705   1.0833
##
## Coefficients:
##                                           Estimate Std. Error z value
## (Intercept)                                0.384661   0.116105   3.313
## age_p                                      0.030428   0.002045  14.880
## r_maritl2 Married - spouse not in household -0.082884   0.262577  -0.316
## r_maritl4 Widowed                          -0.737471   0.103692  -7.112
## r_maritl5 Divorced                          0.847452   0.142456   5.949
## r_maritl6 Separated                         0.033969   0.179425   0.189
## r_maritl7 Never married                    -0.294772   0.086233  -3.418
## r_maritl8 Living with partner               0.494625   0.168288   2.939
## r_maritl9 Unknown marital status           -0.432546   0.556223  -0.778
##                                           Pr(>|z|)
## (Intercept)                               0.000923 ***
## age_p                                      < 2e-16 ***
## r_maritl2 Married - spouse not in household 0.752263
## r_maritl4 Widowed                          1.14e-12 ***
## r_maritl5 Divorced                         2.70e-09 ***
## r_maritl6 Separated                        0.849841
## r_maritl7 Never married                    0.000630 ***
## r_maritl8 Living with partner              0.003291 **
## r_maritl9 Unknown marital status           0.436777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7205.2  on 9125  degrees of freedom
## Residual deviance: 6699.5  on 9117  degrees of freedom
## AIC: 6717.5
##
## Number of Fisher Scoring iterations: 5
```

The intercept, age, widowed, Divorced, Never Married and Living with Partner are the most significant variables. The marriage factors adding to possibility of EverWrk are: Divorced and Living with Partner. Widowed and Never Married have negative coefficients pushing the probability lower favoring NeverWorked.

Predict the Probability of Working with Complete Data

Predict the probability of working for training data.

```
# Complete Model Data

predict_comp_Train= predict(NH11_comp_ModelLog, type="response")
summary(predict_comp_Train)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5561  0.8280  0.8921  0.8655  0.9258  0.9785
```

Predict the probability of working for training data.

```
predict_comp_Test= predict(NH11_comp_ModelLog, type="response", newdata=NH11_complete_test)
summary(predict_comp_Test)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5933  0.8191  0.8877  0.8612  0.9252  0.9785
```

Next, look at the True Positive and False Positive rates based on threshold value.

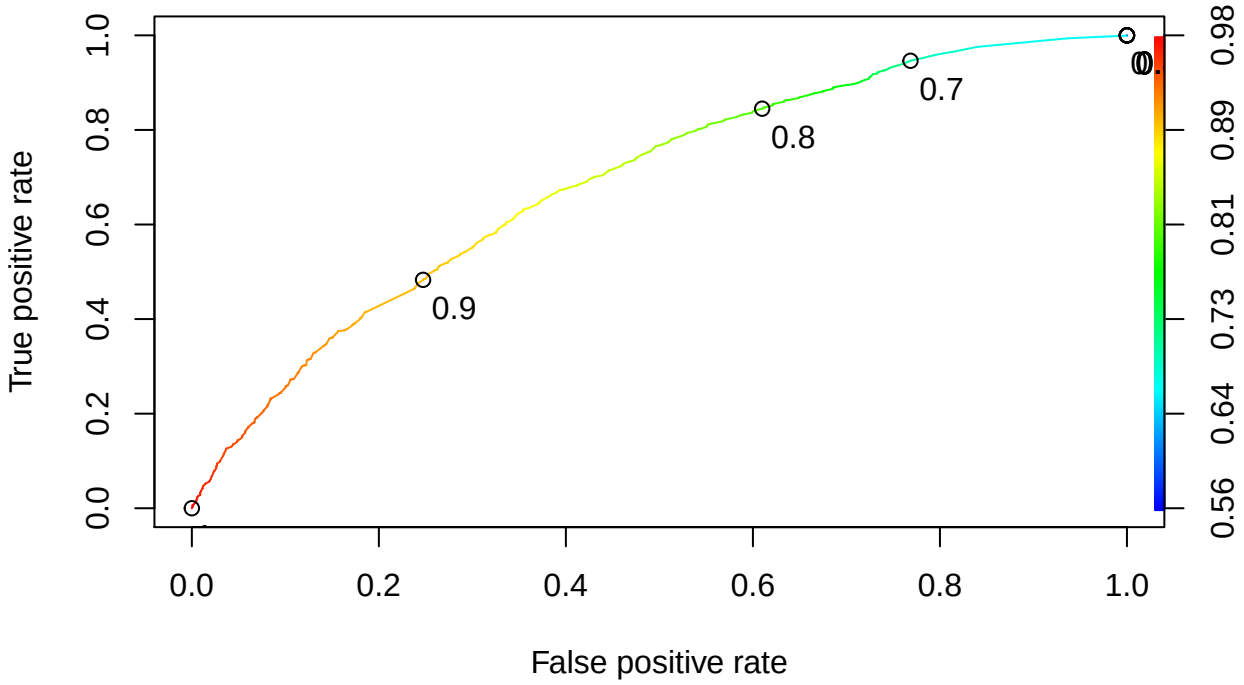
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

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

```
ROCRpred = prediction(predict_comp_Train, NH11_complete_train$everwrk)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
```



Find Threshold Value that gives the best accuracy. Call the calcLogisticModelAccuracy function with the coutcome variable from the training set, 'NH11_complete_train$everwrk', the probabilities from the predict function, 'predict_comp_Train', a range of threshold values from 0.0 to 1.0 in 10 steps, labels for the false and true predictions and a printLevel of 1 to report just the accuracy for each threshold.

```
result = calcLogisticModelAccuracy (NH11_complete_train$everwrk, predict_comp_Train,
                        0.0, 1, 10, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.1, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.2, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.3, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.4, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.5, Logistic Accuracy=0.865548, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.6, Logistic Accuracy=0.865439, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.7, Logistic Accuracy=0.850098, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.8, Logistic Accuracy=0.783804, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.9, Logistic Accuracy=0.518737, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=1, Logistic Accuracy=0.134451, Baseline (EverWorked) Accuracy=0.865548"
```

```
result
```

```
## [1] "OK"                "0.134451019066404" "0.865548980933596"
## [4] "1227"              "7899"              "0"
## [7] "0"                 "0"                 "1"
```

The best threshold is between 0.6 and 0.8. Examine the range more closely.

```
result = calcLogisticModelAccuracy (NH11_complete_train$everwrk, predict_comp_Train,
                        0.6, 0.8, 20, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0.6, Logistic Accuracy=0.865439, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.61, Logistic Accuracy=0.865329, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.62, Logistic Accuracy=0.865329, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.63, Logistic Accuracy=0.865329, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.64, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.65, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.66, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.67, Logistic Accuracy=0.865987, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.68, Logistic Accuracy=0.861056, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.69, Logistic Accuracy=0.853385, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.7, Logistic Accuracy=0.850098, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.71, Logistic Accuracy=0.842537, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.72, Logistic Accuracy=0.836511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.73, Logistic Accuracy=0.83399, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.74, Logistic Accuracy=0.827635, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.75, Logistic Accuracy=0.819417, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.76, Logistic Accuracy=0.814924, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.77, Logistic Accuracy=0.807363, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.78, Logistic Accuracy=0.80287, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.79, Logistic Accuracy=0.793447, Baseline (EverWorked) Accuracy=0.865548"
```

We are getting closer, lets look between 0.65 and 0.67.

```
result = calcLogisticModelAccuracy (NH11_complete_train$everwrk, predict_comp_Train,
                        0.65, 0.67, 20, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0.65, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.651, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.652, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.653, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.654, Logistic Accuracy=0.86511, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.655, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.656, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.657, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.658, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.659, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.66, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.661, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.662, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.663, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.664, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.665, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.666, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.667, Logistic Accuracy=0.867192, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.668, Logistic Accuracy=0.865987, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.669, Logistic Accuracy=0.865987, Baseline (EverWorked) Accuracy=0.865548"
## [1] "Threshold=0.67, Logistic Accuracy=0.865987, Baseline (EverWorked) Accuracy=0.865548"
```

```
result
```

```
## [1] "OK"               "0.865987289064212" "0.865548980933596"
## [4] "197"              "193"               "1030"
## [7] "7706"             "0.975566527408533" "0.160554197229014"
```

There are several threshholds that give the highest accuracy. Let's choose 0.66. The accuracy of the logistic model is only 0.3% better than the baseline model. The logistic model does not give us much improvement. It is what we have for this assignment.

```
result = calcLogisticModelAccuracy (NH11_complete_train$everwrk, predict_comp_Train,
                        0.66, 0.66, 1, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0.66, Logistic Accuracy=0.868617, Baseline (EverWorked) Accuracy=0.865548"
```

```
result
```

```
## [1] "OK"                 "0.868617137847907"  "0.865548980933596"
## [4] "77"                 "49"                 "1150"
## [7] "7850"               "0.993796683124446"  "0.0627546862265689"
```

Now apply the model to the test data.

```
result = calcLogisticModelAccuracy (NH11_complete_test$everwrk, predict_comp_Test,
                        0.66, 0.66, 1, "EverWorked", "NeverWorked", 3)
```

```
## [1] "Model Performance for threshold= 0.66"
## [1] "predicted performance="
##                         Predicted
## Actual                   FALSE=Predict:NeverWorked TRUE=Predict:EverWorked
##   0=Actual:NeverWorked     53 (TN)                   607 (FP)
##   1=Actual:EverWorked      31 (FN)                   4223 (TP)
## [1] "Sensitivity= 0.992712740949694 (True positive rate of EverWorked = TP/(TP+FN) = 4223 /( 4223 + 3
## [1] "Specificity= 0.0803030303030303 (True negative rate of NeverWorked = TN/(TN+FP) = 53 /( 53 + 607
## [1] "Threshold=0.66, Logistic Accuracy=0.870166, Baseline (EverWorked) Accuracy=0.865689"
```

```
result
```

```
## [1] "OK"                 "0.87016687016687"   "0.865689865689866"
## [4] "53"                 "31"                 "607"
## [7] "4223"               "0.992712740949694"  "0.0803030303030303"
```

There is improved accuracy on the test data at 87.01%, but, again, only about 0.45% better than the baseline model for the test data.

Imputed Data Model

The imputed data is examined next to see if it makes any difference on the test data.

Create Logistic Regression model for NH11_imputed Data

Create the logistic model for the NH11_imputed data. Start by creating the test and training data.

```
set.seed(144)
split = sample.split(NH11_imputed$everwrk, 0.65) # we want 65% in the training set
NH11_imputed_train = subset(NH11_imputed, split == TRUE)
NH11_imputed_test  = subset(NH11_imputed, split == FALSE)
```

There are 21459 rows in the training set and 11555 rows in the testing set.

Create the logistic regression model for the NH11_imputed data.

```
NH11_imp_ModelLog = glm(everwrk ~ age_p + r_maritl , data=NH11_imputed_train, family=binomial)
```

Check the coefficients for the imputed data model.

```
summary(NH11_imp_ModelLog)
```

```
##
## Call:
## glm(formula = everwrk ~ age_p + r_maritl, family = binomial,
##     data = NH11_imputed_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6448   0.3631   0.4760   0.5983   0.9126
##
## Coefficients:
##                                             Estimate Std. Error z value
## (Intercept)                                 0.685227   0.073405   9.335
## age_p                                       0.026922   0.001466  18.359
## r_maritl2 Married - spouse not in household -0.221593   0.154279  -1.436
## r_maritl4 Widowed                           -0.670838   0.084453  -7.943
## r_maritl5 Divorced                          0.573791   0.081312   7.057
## r_maritl6 Separated                         0.041301   0.117958   0.350
## r_maritl7 Never married                     -0.486942   0.050024  -9.734
## r_maritl8 Living with partner               0.335449   0.093421   3.591
## r_maritl9 Unknown marital status            0.184000   0.529214   0.348
##                                             Pr(>|z|)
## (Intercept)                                  < 2e-16 ***
## age_p                                        < 2e-16 ***
## r_maritl2 Married - spouse not in household  0.15091
## r_maritl4 Widowed                           1.97e-15 ***
## r_maritl5 Divorced                          1.71e-12 ***
## r_maritl6 Separated                          0.72624
## r_maritl7 Never married                      < 2e-16 ***
## r_maritl8 Living with partner                0.00033 ***
## r_maritl9 Unknown marital status             0.72808
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17866  on 21458  degrees of freedom
## Residual deviance: 16870  on 21450  degrees of freedom
## AIC: 16888
##
## Number of Fisher Scoring iterations: 5
```

Look at the coefficients for the "complete data" logistic model again.

```
summary(NH11_comp_ModelLog)
```

```
##
## Call:
## glm(formula = everwrk ~ age_p + r_maritl, family = binomial,
##     data = NH11_complete_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.7714   0.3264   0.4512   0.5705   1.0833
##
## Coefficients:
##                                            Estimate Std. Error z value
## (Intercept)                                0.384661   0.116105   3.313
## age_p                                      0.030428   0.002045  14.880
## r_maritl2 Married - spouse not in household -0.082884   0.262577  -0.316
## r_maritl4 Widowed                          -0.737471   0.103692  -7.112
## r_maritl5 Divorced                          0.847452   0.142456   5.949
## r_maritl6 Separated                         0.033969   0.179425   0.189
## r_maritl7 Never married                    -0.294772   0.086233  -3.418
## r_maritl8 Living with partner               0.494625   0.168288   2.939
## r_maritl9 Unknown marital status           -0.432546   0.556223  -0.778
##                                            Pr(>|z|)
## (Intercept)                                0.000923 ***
## age_p                                       < 2e-16 ***
## r_maritl2 Married - spouse not in household 0.752263
## r_maritl4 Widowed                          1.14e-12 ***
## r_maritl5 Divorced                         2.70e-09 ***
## r_maritl6 Separated                        0.849841
## r_maritl7 Never married                    0.000630 ***
## r_maritl8 Living with partner              0.003291 **
## r_maritl9 Unknown marital status           0.436777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7205.2  on 9125  degrees of freedom
## Residual deviance: 6699.5  on 9117  degrees of freedom
## AIC: 6717.5
##
## Number of Fisher Scoring iterations: 5
```

The coefficients for the models are within a factor of 2 of each other. The imputed data logistic model variables have the same significance except for 'Living with partner' which is now highly significant.

Find the best threshold for the imputed data. Start by looking at 0 through 1 in 0.1 increments.

```
predict_imp_Train= predict(NH11_imp_ModelLog, type="response")
summary(predict_imp_Train)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6594  0.8165  0.8725  0.8537  0.9111  0.9720
```

```
result = calcLogisticModelAccuracy (NH11_imputed_train$everwrk, predict_imp_Train,
                      0.0, 1.0, 10, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.1, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.2, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.3, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.4, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.5, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.6, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.7, Logistic Accuracy=0.829675, Baseline (EverWorked) Accuracy=0.853674"
```

```
## [1] "Threshold=0.8, Logistic Accuracy=0.75763, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.9, Logistic Accuracy=0.436833, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=1, Logistic Accuracy=0.146325, Baseline (EverWorked) Accuracy=0.853674"
```

Check the range between 0.6 and 0.8 more closely.

```
result = calcLogisticModelAccuracy (NH11_imputed_train$everwrk, predict_imp_Train,
                        0.6, 0.8, 20, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0.6, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.61, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.62, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.63, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.64, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.65, Logistic Accuracy=0.853674, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.66, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.67, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.68, Logistic Accuracy=0.848641, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.69, Logistic Accuracy=0.839414, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.7, Logistic Accuracy=0.829675, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.71, Logistic Accuracy=0.824595, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.72, Logistic Accuracy=0.81672, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.73, Logistic Accuracy=0.808099, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.74, Logistic Accuracy=0.800037, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.75, Logistic Accuracy=0.794771, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.76, Logistic Accuracy=0.789924, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.77, Logistic Accuracy=0.784565, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.78, Logistic Accuracy=0.778834, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.79, Logistic Accuracy=0.770445, Baseline (EverWorked) Accuracy=0.853674"
```

Further refine checing 0.66 through 0.68.

```
result = calcLogisticModelAccuracy (NH11_imputed_train$everwrk, predict_imp_Train,
                        0.66, 0.68, 20, "EverWorked", "NeverWorked", 1)
```

```
## [1] "Threshold=0.66, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.661, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.662, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.663, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.664, Logistic Accuracy=0.853627, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.665, Logistic Accuracy=0.854606, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.666, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.667, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.668, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.669, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.67, Logistic Accuracy=0.854559, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.671, Logistic Accuracy=0.851204, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.672, Logistic Accuracy=0.851111, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.673, Logistic Accuracy=0.851111, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.674, Logistic Accuracy=0.851111, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.675, Logistic Accuracy=0.851111, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.676, Logistic Accuracy=0.851111, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.677, Logistic Accuracy=0.848641, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.678, Logistic Accuracy=0.848641, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.679, Logistic Accuracy=0.848641, Baseline (EverWorked) Accuracy=0.853674"
## [1] "Threshold=0.68, Logistic Accuracy=0.848641, Baseline (EverWorked) Accuracy=0.853674"
```

A threshold value of 0.665 gives the best accuracy.

```
result = calcLogisticModelAccuracy (NH11_imputed_train$everwrk, predict_imp_Train,
                        0.665, 0.665, 1, "EverWorked", "NeverWorked", 3)
```

```
## [1] "Model Performance for threshold= 0.665"
## [1] "predicted performance="
##                         Predicted
## Actual                   FALSE=Predict:NeverWorked TRUE=Predict:EverWorked
##   0=Actual:NeverWorked     110 (TN)                  3030 (FP)
##   1=Actual:EverWorked      90 (FN)                   18229 (TP)
## [1] "Sensitivity= 0.995087068071401 (True positive rate of EverWorked = TP/(TP+FN) = 18229 /( 18229 +
## [1] "Specificity= 0.035031847133758 (True negative rate of NeverWorked = TN/(TN+FP) = 110 /( 110 + 3
## [1] "Threshold=0.665, Logistic Accuracy=0.854606, Baseline (EverWorked) Accuracy=0.853674"
```

Determine the accuracy of the imputed model on the imputed test data.

```
predict_imp_Test = predict(NH11_imp_ModelLog, type="response", newdata=NH11_imputed_test)
summary(predict_imp_Test)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6285  0.8165  0.8725  0.8541  0.9118  0.9720
```

```
result = calcLogisticModelAccuracy (NH11_imputed_test$everwrk, predict_imp_Test,
                        0.665, 0.665, 1, "EverWorked", "NeverWorked", 3)
```

```
## [1] "Model Performance for threshold= 0.665"
## [1] "predicted performance="
##                         Predicted
## Actual                   FALSE=Predict:NeverWorked TRUE=Predict:EverWorked
##   0=Actual:NeverWorked     58 (TN)                   1633 (FP)
##   1=Actual:EverWorked      55 (FN)                   9809 (TP)
## [1] "Sensitivity= 0.994424168694242 (True positive rate of EverWorked = TP/(TP+FN) = 9809 /( 9809 + 5
## [1] "Specificity= 0.0342992312241277 (True negative rate of NeverWorked = TN/(TN+FP) = 58 /( 58 + 163
## [1] "Threshold=0.665, Logistic Accuracy=0.853916, Baseline (EverWorked) Accuracy=0.853656"
```

Now determine the accuracy of the imputed model on the complete test data. This is the real test is to determine which model is better.

```
predict_imp_comp_Test = predict(NH11_imp_ModelLog, type="response",
      newdata=NH11_complete_test)
summary(predict_imp_Test)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6285  0.8165  0.8725  0.8541  0.9118  0.9720
```

```
result = calcLogisticModelAccuracy (NH11_complete_test$everwrk, predict_imp_comp_Test,
                        0.665, 0.665, 1, "EverWorked", "NeverWorked", 3)
```

```
## [1] "Model Performance for threshold= 0.665"
## [1] "predicted performance="
##                         Predicted
## Actual                   FALSE=Predict:NeverWorked TRUE=Predict:EverWorked
##   0=Actual:NeverWorked     52 (TN)                   608 (FP)
##   1=Actual:EverWorked      30 (FN)                   4224 (TP)
## [1] "Sensitivity= 0.992947813822285 (True positive rate of EverWorked = TP/(TP+FN) = 4224 /( 4224 + 3
## [1] "Specificity= 0.0787878787878788 (True negative rate of NeverWorked = TN/(TN+FP) = 52 /( 52 + 608
## [1] "Threshold=0.665, Logistic Accuracy=0.870166, Baseline (EverWorked) Accuracy=0.865689"
```

The accuracy of the imputed model on the complete test data is 87.01% which is the same for the complete data model. Both models produce the same results. This could be expected since the imputed data will most likely be proportional to the complete data. The imputed data model will be used for the rest of the assignment since the variables are more significant than in the complete data model.

Predict 'EverWrk' based on Marital Status

Exercise Part 2. Predict the probability of working for each level of marital status.

As in the example, convert the coefficients so they can be more easily understood.

```
ImpModel.out <- coef(summary(NH11_imp_ModelLog))
ImpModel.out[, "Estimate"] <- exp(coef(NH11_imp_ModelLog))
ImpModel.out
```

```
##                                            Estimate  Std. Error
## (Intercept)                               1.9842222 0.073404888
## age_p                                     1.0272876 0.001466444
## r_maritl2 Married - spouse not in household 0.8012416 0.154279292
## r_maritl4 Widowed                         0.5112800 0.084453476
## r_maritl5 Divorced                        1.7749828 0.081311691
## r_maritl6 Separated                       1.0421654 0.117958128
## r_maritl7 Never married                   0.6145026 0.050024033
## r_maritl8 Living with partner             1.3985677 0.093421080
## r_maritl9 Unknown marital status          1.2020161 0.529214401
##                                              z value     Pr(>|z|)
## (Intercept)                               9.3348960 1.010904e-20
## age_p                                    18.3586436 2.815519e-75
## r_maritl2 Married - spouse not in household -1.4363093 1.509143e-01
## r_maritl4 Widowed                        -7.9432824 1.969000e-15
## r_maritl5 Divorced                        7.0566815 1.705258e-12
## r_maritl6 Separated                       0.3501297 7.262414e-01
## r_maritl7 Never married                  -9.7341633 2.155844e-22
## r_maritl8 Living with partner             3.5907167 3.297699e-04
## r_maritl9 Unknown marital status          0.3476856 7.280763e-01
```

Predict Everwork for each Marital Status

Create a test data frame to use against the imputed model to compute everwrk based on marital status. The average age will be used.

```
predDat <- with(NH11,
                expand.grid(age_p= mean(age_p, na.rm = TRUE),
                            #  r_maritl=c(dimnames(table(NH11$r_maritl)))
                            r_maritl=c("1 Married - spouse in household",
                                       "2 Married - spouse not in household",
                                       "4 Widowed",
                                       "5 Divorced",
                                       "6 Separated",
                                       "7 Never married",
                                       "8 Living with partner",
                                       "9 Unknown marital status"
                                       )
```

```
                            )
                )
```

Now bind the results to the data frame.

```
cbind(predDat, predict(NH11_imp_ModelLog, type = "response",
                       se.fit = TRUE, interval="confidence",
                       newdata = predDat))
```

```
##      age_p                              r_maritl       fit      se.fit
## 1 48.10983     1 Married - spouse in household 0.8787276 0.003454428
## 2 48.10983 2 Married - spouse not in household 0.8530647 0.018929420
## 3 48.10983                              4 Widowed 0.7874459 0.012729194
## 4 48.10983                             5 Divorced 0.9278569 0.004981634
## 5 48.10983                            6 Separated 0.8830605 0.011741210
## 6 48.10983                          7 Never married 0.8166021 0.006125056
## 7 48.10983                    8 Living with partner 0.9101841 0.007250475
## 8 48.10983              9 Unknown marital status 0.8970101 0.048796693
##   residual.scale
## 1              1
## 2              1
## 3              1
## 4              1
## 5              1
## 6              1
## 7              1
## 8              1
```

With a threshold of 0.665, the fit predicts Ever Worked is true for each marital status. This is not a very good predictor since we know from data clean up there are 1887 of the 14,040 rows where everwork is false.

Create another data set to examine marital status for each age and see where the model predicts, never having worked.

```
predDat <- with(NH11,
                expand.grid(#age_p= mean(age_p, na.rm = TRUE),
                            age_p= c(18:85),
                           #  r_maritl=c(dimnames(table(NH11$r_maritl)))
                           r_maritl=c("1 Married - spouse in household",
                                      "2 Married - spouse not in household",
                                      "4 Widowed",
                                      "5 Divorced",
                                      "6 Separated",
                                      "7 Never married",
                                      "8 Living with partner",
                                      "9 Unknown marital status"
                                      )
                            )
                )
```

Bind the response probabilities to the data.

```
graphData=cbind(predDat, predict(NH11_imp_ModelLog, type = "response",
                       se.fit = TRUE, interval="confidence",
                       newdata = predDat))
```

Now plot the results of age and marital status. The threshold line of 0.665 is included for reference.

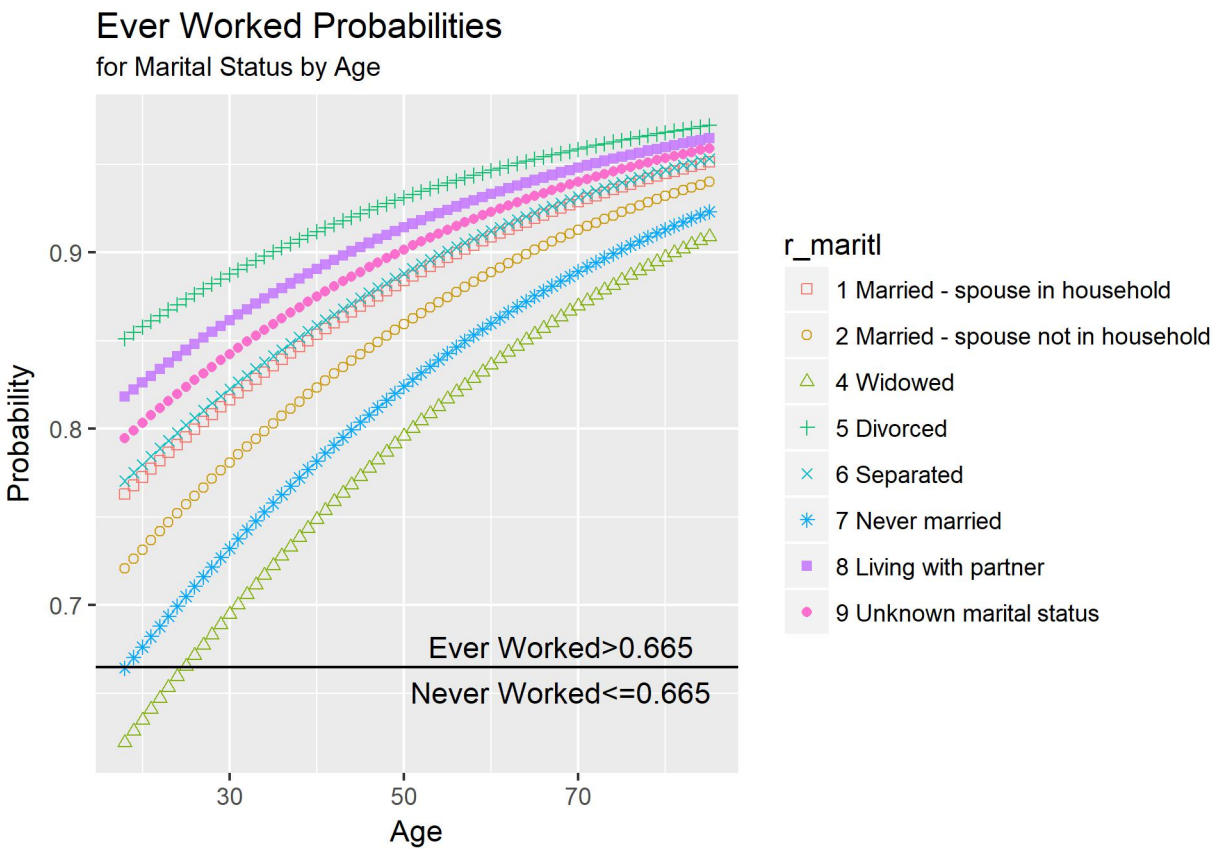


Figure 1: Ever Worked Probabilities for Marital Status by Age

```
library(ggplot2)

#cutoff <- data.frame( x = c(-Inf, Inf), y = 0.66, cutoff = factor(0.66) )

#     geom_line(aes( x, y, linetype = cutoff ), cutoff) +

g <- ggplot(graphData,aes(age_p, fit, col=r_maritl)) +
       geom_point(alpha=0.9,aes(shape=r_maritl)) +
       scale_shape_manual(values=c(0,1,2,3,4,8,15,16))+
       geom_hline(yintercept = 0.665) +
       annotate("text", min(graphData$age_p)+50, 0.658, vjust = -1, label = "Ever Worked>0.665") +
       annotate("text", min(graphData$age_p)+50, 0.632, vjust = -1, label = "Never Worked<=0.665") +
       labs(title = 'Ever Worked Probabilities',
              subtitle = 'for Marital Status by Age',
              x="Age", y="Probability")+
              #colour = "Marital Status")
      # scale_fill_discrete(name = "Marital Status")
         # theme(axis.text.x = element_text(angle=-90))
  ggsave("Fig-EverWrk_Prob_age_maritl.jpg")
```

```
## Saving 6.5 x 4.5 in image
```

The “Everworked Probabilities”, Figure 1, shows that regardless of age, Everwork is predicted to be true for

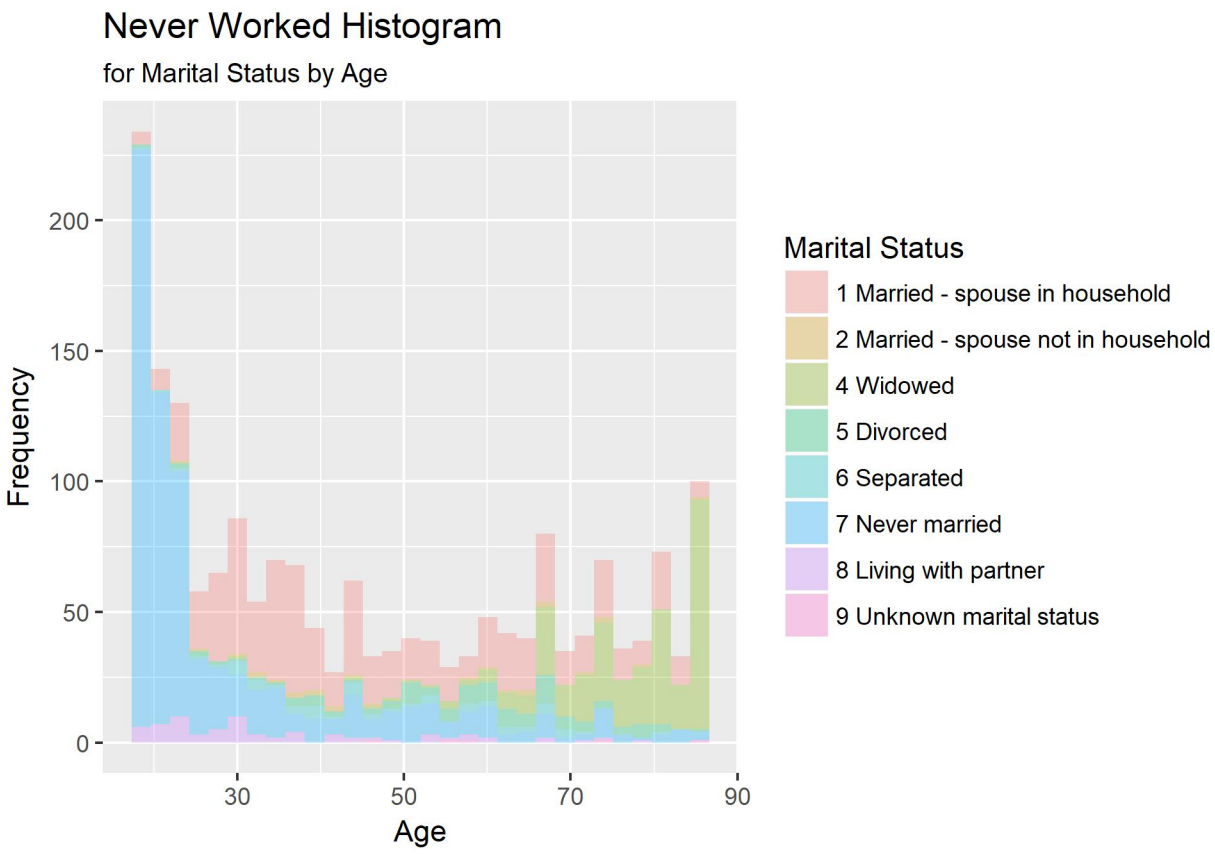


Figure 2: Never Worked Histogram for Marital Status by Age

each marital status except, "Never Married" for age 18 and "Widowed" for ages 18 through 24.

Let's look at the records where Everwork==0 (FALSE) and see the histogram for each age group.

```
NH11Never<-NH11_complete[NH11_complete$everwrk==0,]
print(paste("There are",nrow(NH11Never),"cases that have never worked."))
```

```
## [1] "There are 1887 cases that have never worked."
```

```
g <- ggplot(NH11Never,aes(age_p, fill=r_maritl)) +
       geom_histogram(alpha=0.3) + #,aes(shape=r_maritl)) +
       scale_shape_manual(values=c(0,1,2,3,4,8,15,16))+
       labs(title = 'Never Worked Histogram',
              subtitle = 'for Marital Status by Age',
              x="Age", y="Frequency") +
       scale_fill_discrete(name = "Marital Status")
         # theme(axis.text.x = element_text(angle=-90))
  ggsave("Fig-NeverWrk_hist_age_maritl.jpg")
```

```
## Saving 6.5 x 4.5 in image
```

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

The "Never Worked Histogram", Figure 2, shows there is a larger grouping of "never worked" at the younger ages, especially for "never married" which would be expected. At the higher age ranges there is a slightly higher number of "never worked" for "widowed" which also makes sense. There is a fairly consistent number

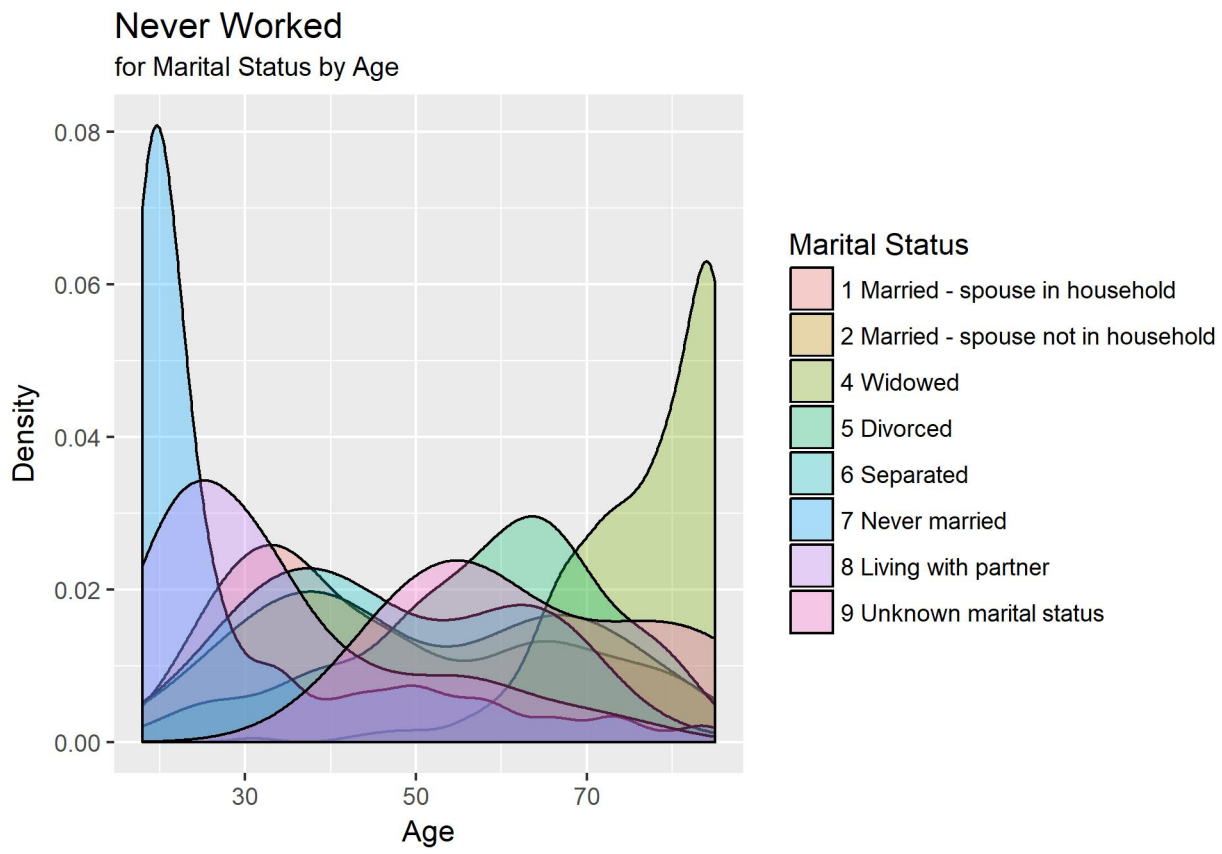


Figure 3: Never Worked Density for Marital Status by Age

of "never worked" for "Married, spouse in household" across all age groups which also makes sense.

It's hard to see all of the marital status in this graph. Let's look at a density plot.

```
g <- ggplot(NH11Never,aes(age_p, fill=r_maritl)) +
       geom_density(alpha=0.3) + #,aes(shape=r_maritl)) +
       scale_shape_manual(values=c(0,1,2,3,4,8,15,16))+
       labs(title = 'Never Worked',
              subtitle = 'for Marital Status by Age',
              x="Age", y="Density") +
       scale_fill_discrete(name = "Marital Status")
         # theme(axis.text.x = element_text(angle=-90))
  ggsave("Fig-NeverWrk_Dens_age_maritl.jpg")
```

```
## Saving 6.5 x 4.5 in image
```

The individual marital status' are a little bit easier to see in the "Never Worked Density", Figure 3, and the conclusions are about the same.

Conclusion

This was a very interesting assignment in terms of cleaning data and learning the tools for logistic regression. Despite the logistic model being pretty poor, this was a good way experience logistic regression.

There may be better parameters to predict everwork but exploring what variables would enhance the model was not the goal of this assighment.