Alzheimer's project

Nkosi Sampson

2024-08-19

```
#reload data again so that its not overwritten
bio_data <- NACCdata::biomarker_data %>%
  mutate(TauAssayDate = make_date(CSFTTYR, CSFTTMO, CSFTTDY))

# Assuming mri_data is already loaded and ImagingVisitDate is correctly created
bio_data1 <- bio_data %>%
  arrange(NACCID, TauAssayDate) %>%
  group_by(NACCID) %>%
  mutate(TauAssayNum = row_number())  # Assign visit number




# Find the first visit with a valid recorded measurement of CSFVOL and NACCICV
first_measurement_visit <- bio_data1 %>%
  group_by(NACCID) %>%
  summarize(FirstTauAssayVisit = min(TauAssayNum)) %>%
  ungroup()
```

```
#create an ImagingVisitDate column
bio_data2 <- bio_data1 %>%
  mutate(TauAssayDate = make_date(CSFTTYR, CSFTTMO, CSFTTDY)) %>%
  arrange(NACCID, TauAssayDate) %>%
  group_by(NACCID) %>%
  mutate(TauAssayNum = row_number()) %>%
  ungroup() # Remove the grouping

# Now filter for the first visit number and valid measurements
bio_data3 <- bio_data2 %>%
  filter(TauAssayNum == 1)
```

```
#Convert VISITYR, VISITMO, VISITDAY in UDS to a single Data column
uds <- NACCdata::UDS %>%
  mutate(VisitDate = make_date(VISITYR, VISITMO, VISITDAY))
```

```
#Filter for the first visit per patient in both datasets
uds_first_visit <- uds %>%
  group_by(NACCID) %>%
  filter(VisitDate == min(VisitDate)) %>%
  ungroup()
```

```r
#Join the first UDS visit with the first valid MRI visit and Select required columns.
#Also omit the from the biomarker data
df_incomplete <- na.omit(
  left_join(
    uds_first_visit, bio_data3, by = "NACCID") %>%
    dplyr::select(
      NACCID,  NACCAGE, BPDIAS, BPSYS, HXHYPER, HYPERCHO, HXSTROKE, CVHATT, CVCHF, CVAFIB, NACCBMI,
      DIABETES, B12DEF, DEP2YRS, SMOKYRS, ALCOHOL, NACCNIHR, HISPANIC,
      EDUC, SEX, MARISTAT, NACCALZD, CSFTTAU, CSFABETA))
```

```r
#change unknown and not collected data points to NA
df_incomplete1 <- df_incomplete %>%
  mutate(across(c(ALCOHOL, HXSTROKE, DEP2YRS, HXHYPER, NACCBMI, BPDIAS, BPSYS, HYPERCHO, CVHATT,
                  CVCHF, CVAFIB, DIABETES, B12DEF, SMOKYRS), ~ na_if(.x, -4))) %>%
  mutate(across(c(ALCOHOL, DEP2YRS, HYPERCHO, CVHATT, CVCHF, CVAFIB, DIABETES, B12DEF, MARISTAT,
                    HISPANIC), ~ na_if(.x, 9))) %>%
  mutate(across(c(BPDIAS, BPSYS), ~ na_if(.x, 888))) %>%
  mutate(across(c(EDUC, SMOKYRS, NACCNIHR), ~ na_if(.x, 99))) %>%
  mutate(NACCBMI = na_if(NACCBMI, 888.8)) %>%
  mutate(BPDIAS = na_if(BPDIAS, 777)) %>%
  mutate(SMOKYRS = replace(SMOKYRS, SMOKYRS %in% c(88), NA))


#remove NA's from data frame
df_incomplete2 <- na.omit(df_incomplete1)
#number of rows in new data frame
nrow(df_incomplete1)
```

```
## [1] 2115
```

```r
library(dplyr)

df_incomplete3 <- df_incomplete2 %>%
  mutate(
    # Recode NACCALZD based on the condition, using integer literals
    NACCALZD = case_when(
      NACCALZD == 8 ~ 0L,  # Use 0L to explicitly denote an integer literal
      TRUE ~ 1L           # Use 1L to explicitly denote an integer literal
    ),
    # Change HXSTROKE variable to binary, handling NA values appropriately
    HXSTROKE = case_when(
      is.na(HXSTROKE) ~ NA_real_,  # Preserve NA values
      HXSTROKE == 2 ~ 1,           # Recode '2' as '1'
      TRUE ~ 0                     # Default to '0'
    ),
    # Change SEX variable to 0/1
    SEX = case_when(
      SEX == 1 ~ 0,
      SEX == 2 ~ 1
    )
  )

df_incomplete4 <- df_incomplete3[, -1]
```

```r
df_incomplete5 <- df_incomplete4 %>%
  mutate(CVHATT = ifelse(CVHATT == "1" | CVHATT == "2", 1, 0),
         CVCHF = ifelse(CVCHF == "1" | CVCHF == "2", 1, 0),
         CVAFIB = ifelse(CVAFIB == "1" | CVAFIB == "2", 1, 0),
         DIABETES = ifelse(DIABETES == "1" | DIABETES == "2", 1, 0),
         B12DEF = ifelse(B12DEF == "1" | B12DEF == "2", 1, 0),
         ALCOHOL = ifelse(ALCOHOL == "1" | ALCOHOL == "2", 1, 0),
          HYPERCHO = ifelse(HYPERCHO == "1" | HYPERCHO == "2", 1, 0),
          MARISTAT = ifelse(MARISTAT == "2" | MARISTAT == "3" | MARISTAT == "4" |
                              MARISTAT == "5" | MARISTAT == "6", 0, 1))
```

```r
#Make categoricals factors
df_incomplete6 <- df_incomplete5 %>%
  mutate(
         HXHYPER = as.factor(HXHYPER),
         HXSTROKE = as.factor(HXSTROKE),
         CVHATT = as.factor(CVHATT),
         CVCHF = as.factor(CVCHF),
         CVAFIB = as.factor(CVAFIB),
         DIABETES = as.factor(DIABETES),
         B12DEF = as.factor(B12DEF),
         DEP2YRS = as.factor(DEP2YRS),
         ALCOHOL = as.factor(ALCOHOL),
         HYPERCHO = as.factor(HYPERCHO),
         NACCNIHR = as.factor(NACCNIHR),
         HISPANIC = as.factor(HISPANIC),
         SEX = as.factor(SEX),
         MARISTAT = as.factor(MARISTAT),
         NACCALZD = as.factor(NACCALZD))
```

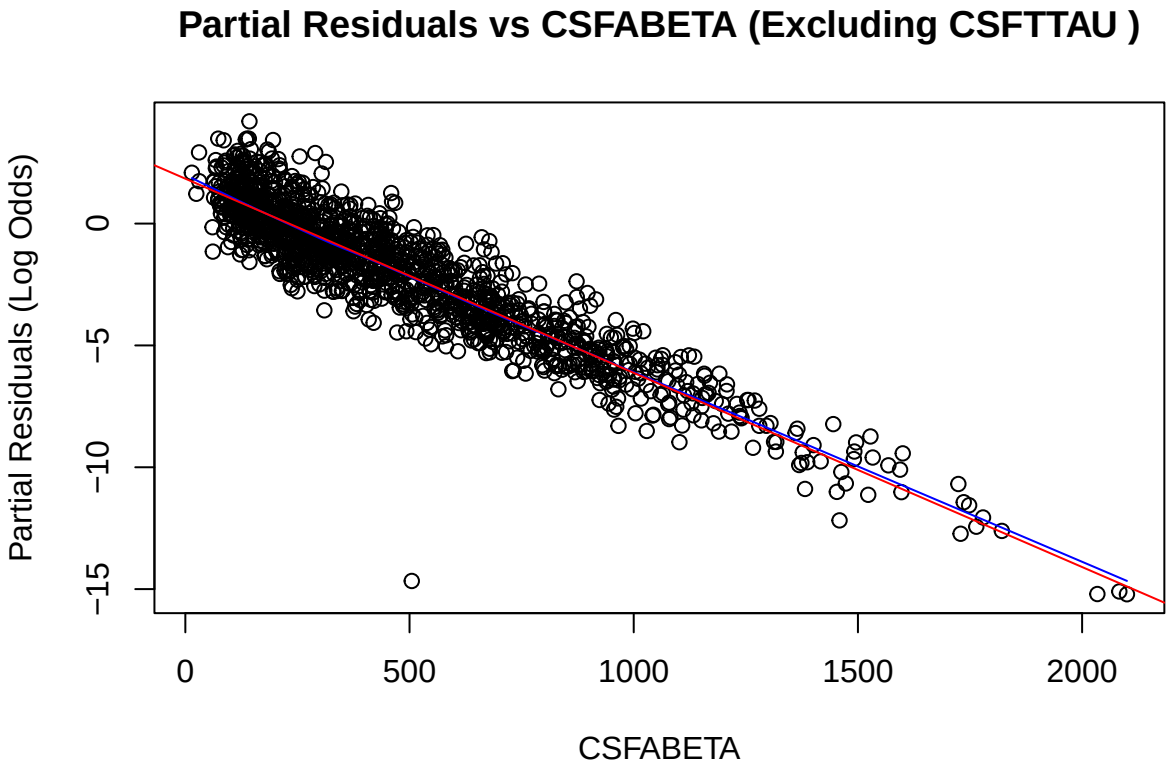
```r
#order variables so that numerical are first (useful for later)
df <- dplyr::select(df_incomplete6, CSFTTAU, CSFABETA, NACCAGE, BPDIAS, BPSYS, NACCBMI, SMOKYRS, EDUC,
                    everything())
```

For the LASSO model, we first want to check the linearity assumption for continuous variables

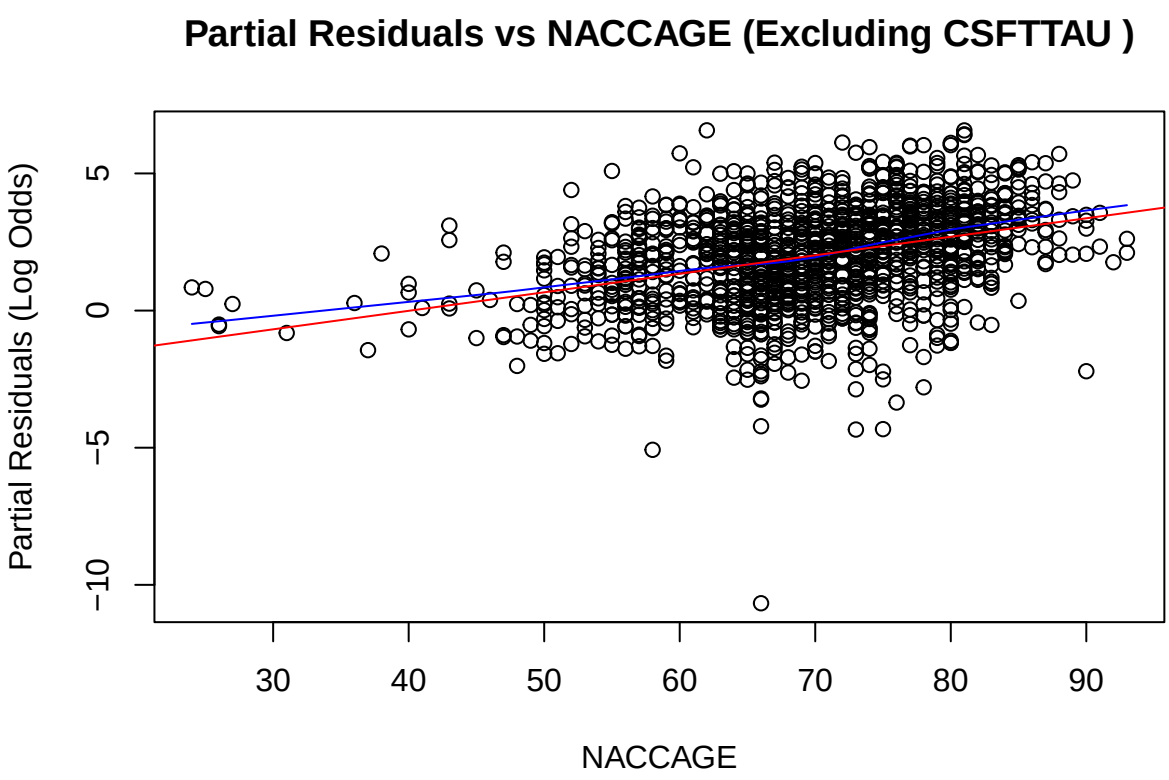
```r
# Full model including all predictors
full_model <- glm(NACCALZD ~ CSFTTAU + CSFABETA + NACCAGE + BPDIAS + BPSYS + HXHYPER + HXSTROKE
      + NACCBMI + DEP2YRS + SMOKYRS + HISPANIC + EDUC + SEX + ALCOHOL + HYPERCHO
      + CVHATT + CVCHF + DIABETES + NACCNIHR + B12DEF + MARISTAT + CVAFIB,
                  data = df, family = binomial)

# Variables to be iteratively excluded
variables_to_exclude <- c("CSFTTAU", "CSFABETA", "NACCAGE", "EDUC", "SMOKYRS", "NACCBMI", "BPDIAS",
                          "BPSYS")

# Loop over each variable to exclude
for (var in variables_to_exclude) {

  # Create a formula for the reduced model by excluding the current variable
  reduced_formula <- as.formula(paste("NACCALZD ~", paste(setdiff(variables_to_exclude, var),
    collapse = " + "), "+ HXHYPER + HXSTROKE + DEP2YRS + HISPANIC + SEX + ALCOHOL+ HYPERCHO + CVHATT + C
                                      DIABETES + NACCNIHR + B12DEF + MARISTAT + CVAFIB"))
```

```r
  # Fit the reduced model
  reduced_model <- glm(reduced_formula, data = df, family = binomial)

  # Calculate the linear predictor (log odds) from the reduced model
  linear_predictor_reduced <- predict(reduced_model, type = "link")

  # Define the predictor variables (excluding the outcome variable)
  predictors <- names(coef(reduced_model))

  # Loop over each predictor to calculate and plot partial residuals, one at a time
  for (predictor in predictors) {

    # Skip the intercept
    if (predictor == "(Intercept)") next

    print(paste("Processing:", predictor))  # Debugging line

    # Extract the coefficient for the current predictor from the full model
    beta_X <- coef(full_model)[predictor]

    # Extract the values of the predictor of interest
    X_values <- df[[predictor]]

    # Ensure the predictor exists in the dataframe
    if (is.null(X_values)) {
      print(paste("", predictor, ""))
      next
    }

    # Calculate the partial residuals for the log odds
    partial_residuals <- linear_predictor_reduced + beta_X * X_values

    # Plot the partial residuals against the predictor, one plot per page
    plot(X_values, partial_residuals,
         main = paste("Partial Residuals vs", predictor, "(Excluding", var, ")"),
         xlab = predictor,
         ylab = "Partial Residuals (Log Odds)")

    # Fit a lowess smoother to check linearity
    lines(lowess(X_values, partial_residuals), col = "blue")

    # Add a regression line to visualize the relationship
    abline(lm(partial_residuals ~ X_values), col = "red")

    # Pause before showing the next plot (remove or comment out to auto advance)
    # readline(prompt="Press [Enter] to see the next plot...")
  }
}
```

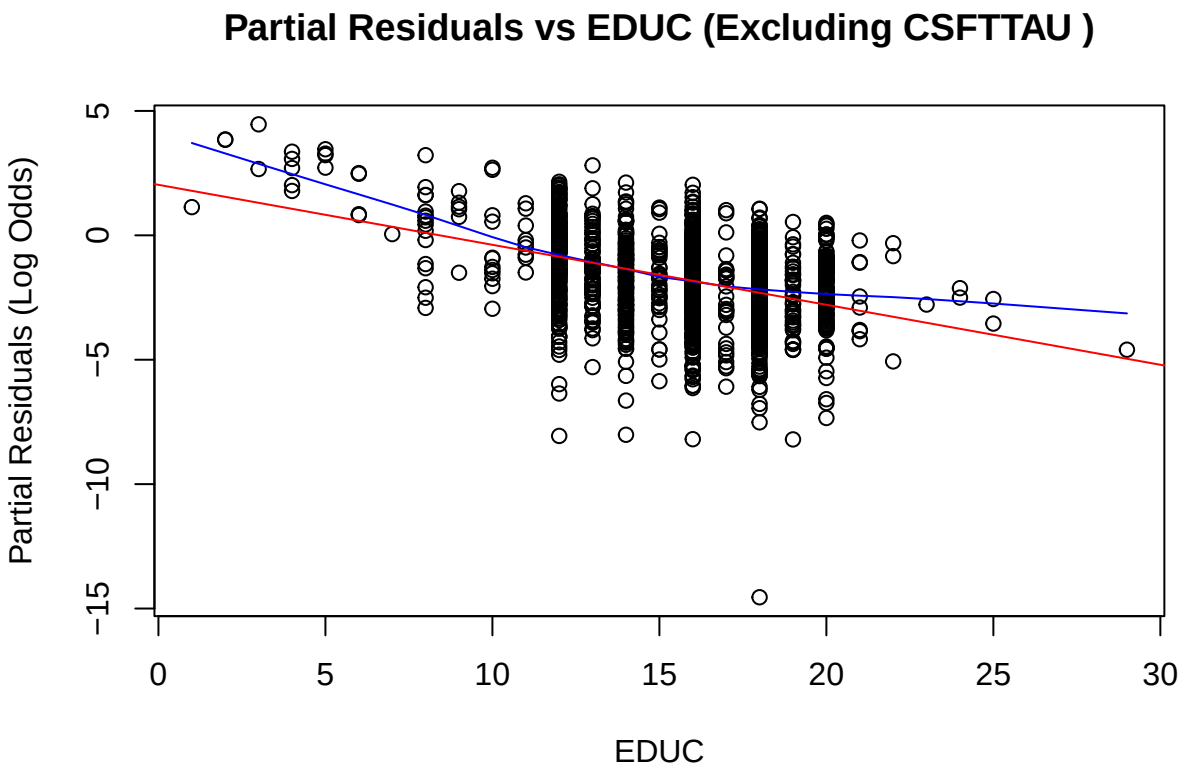
```
## [1] "Processing: CSFABETA"
```



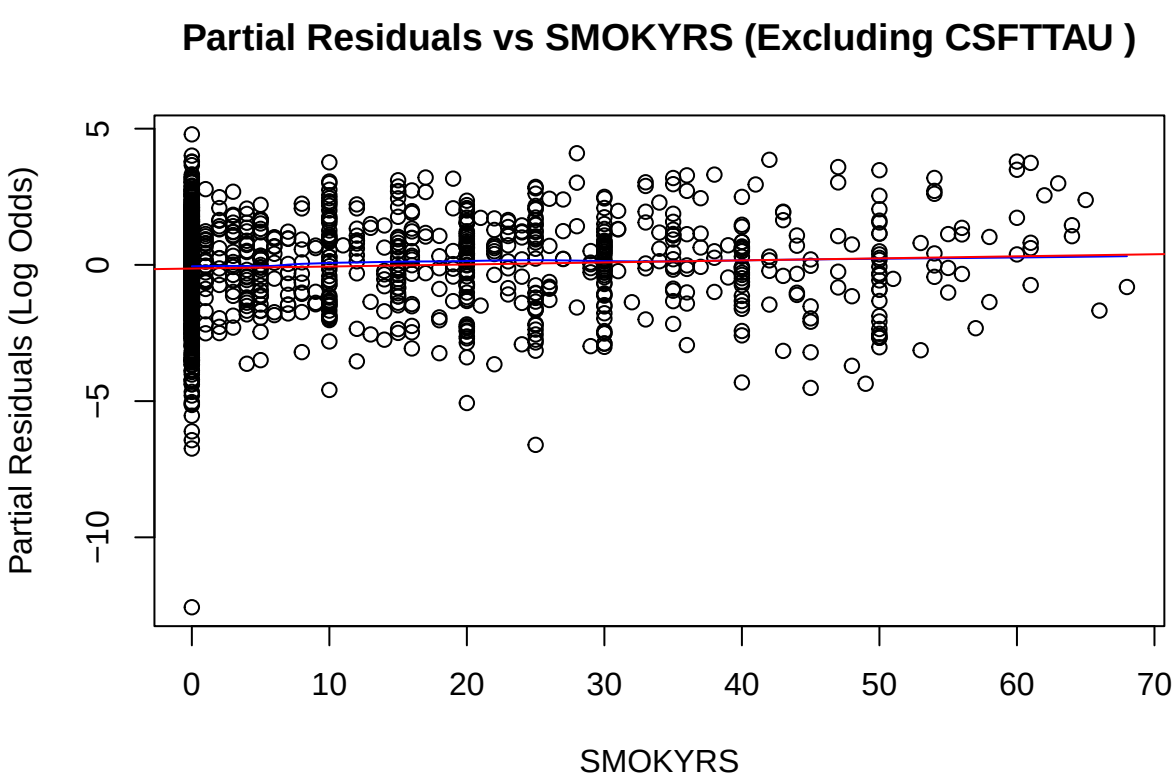

```
## [1] "Processing: NACCAGE"
```



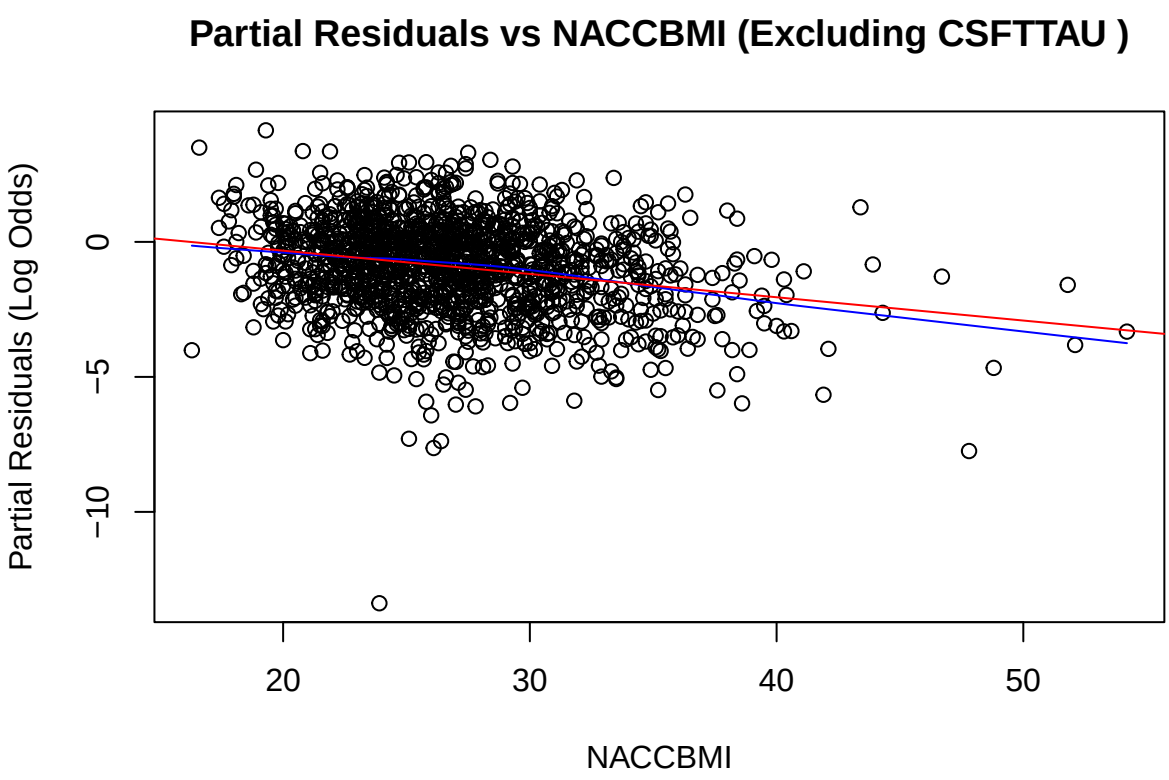

```
## [1] "Processing: EDUC"
```



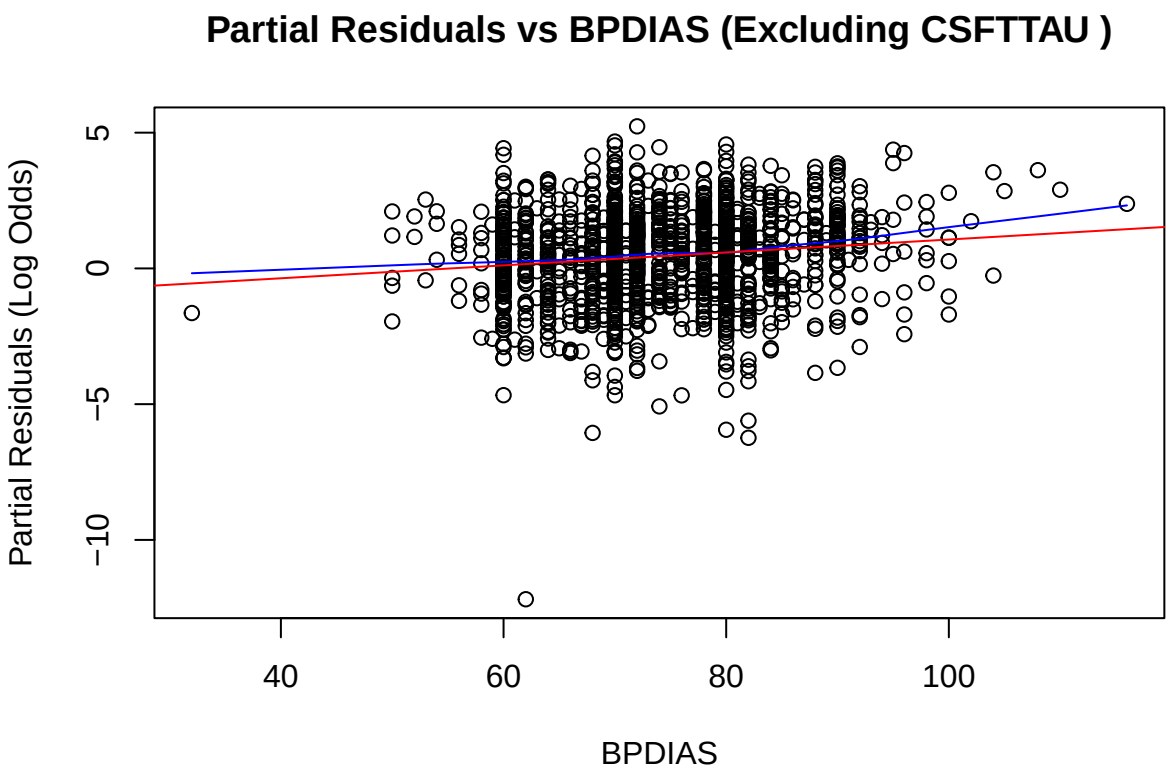

```
## [1] "Processing: SMOKYRS"
```



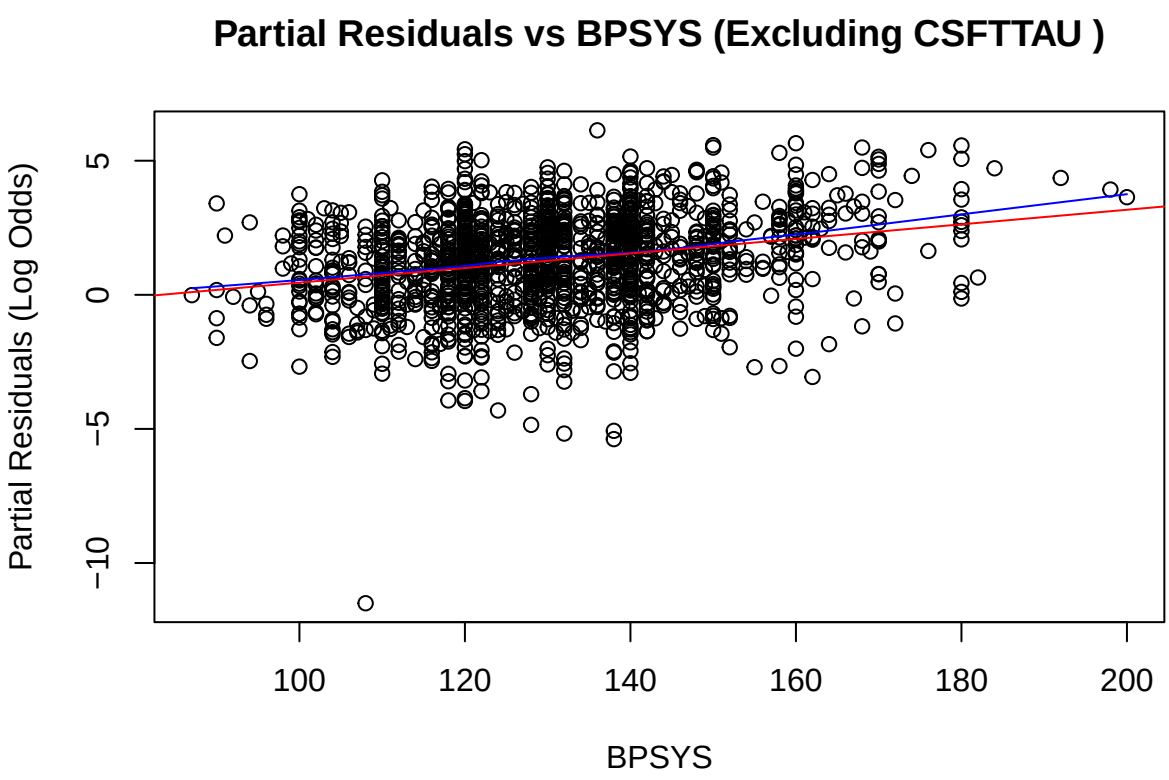

```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```

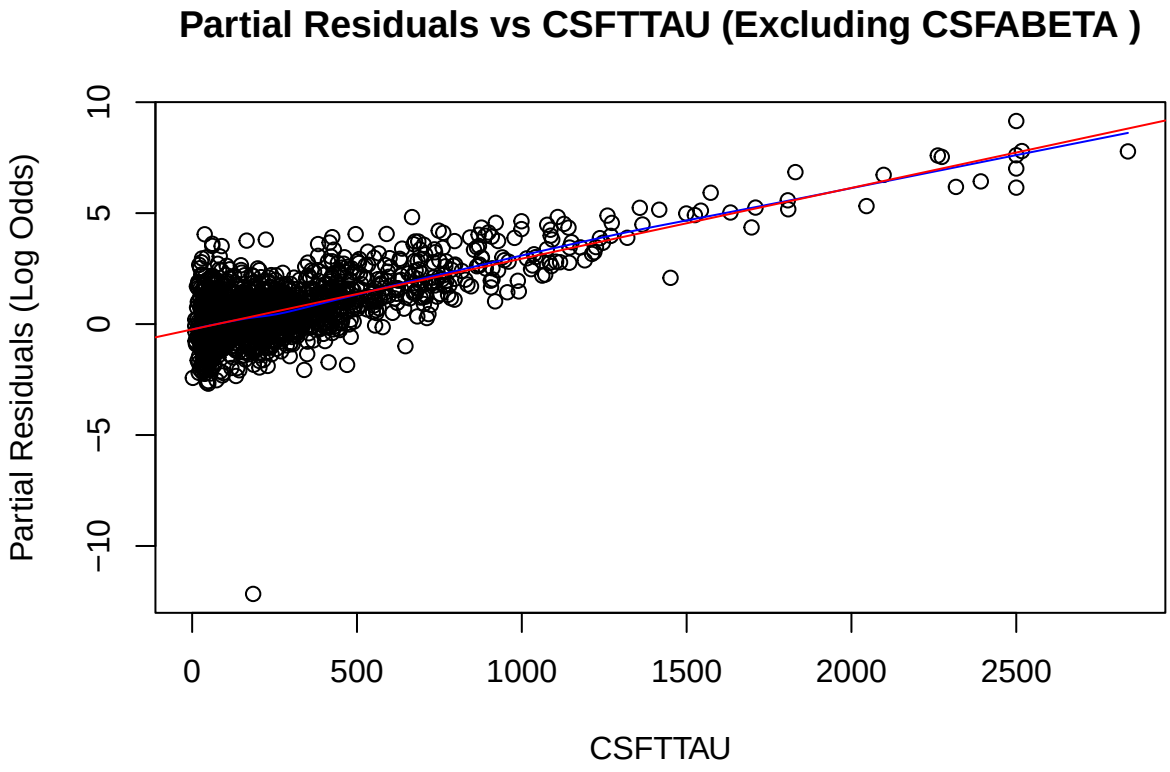



```
## [1] "Processing: BPSYS"
```

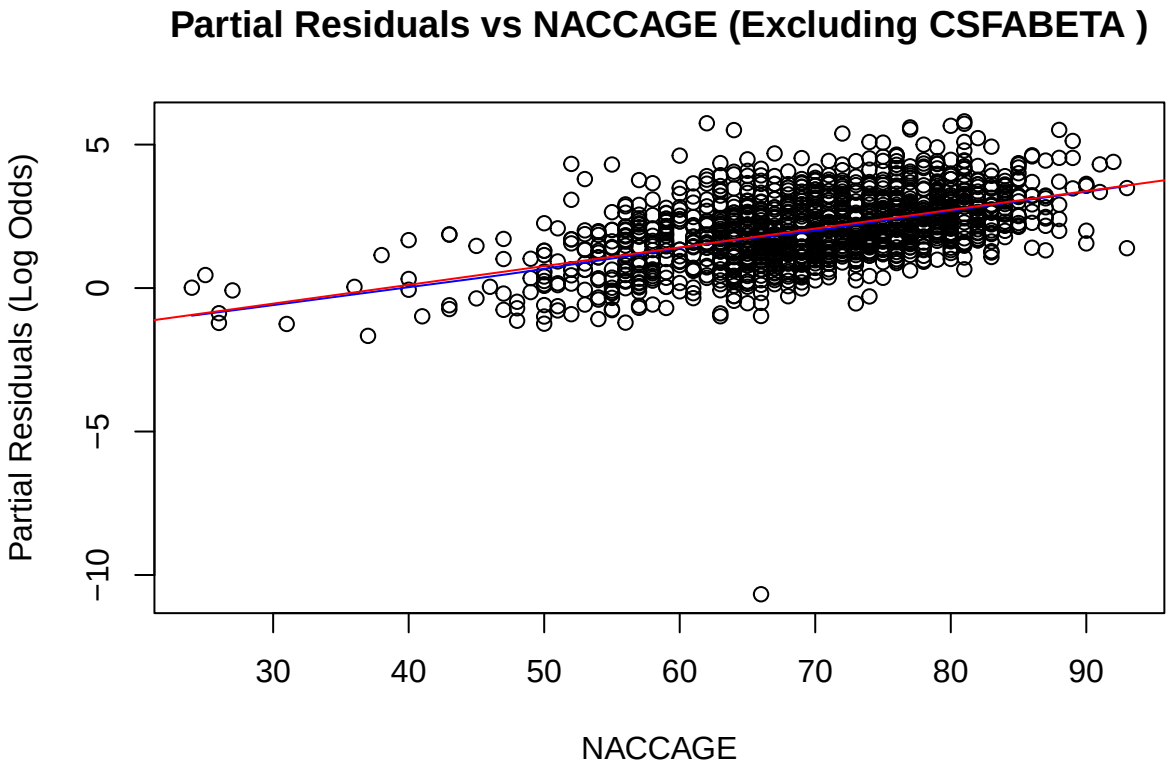



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

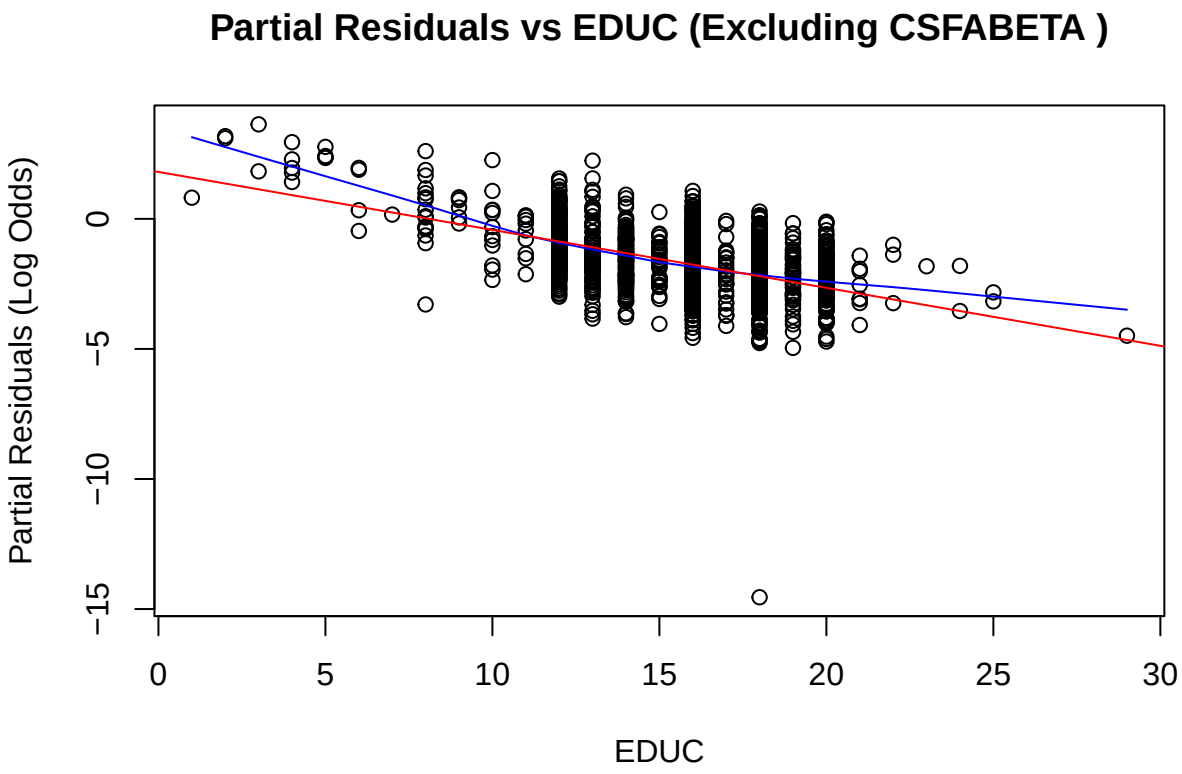
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



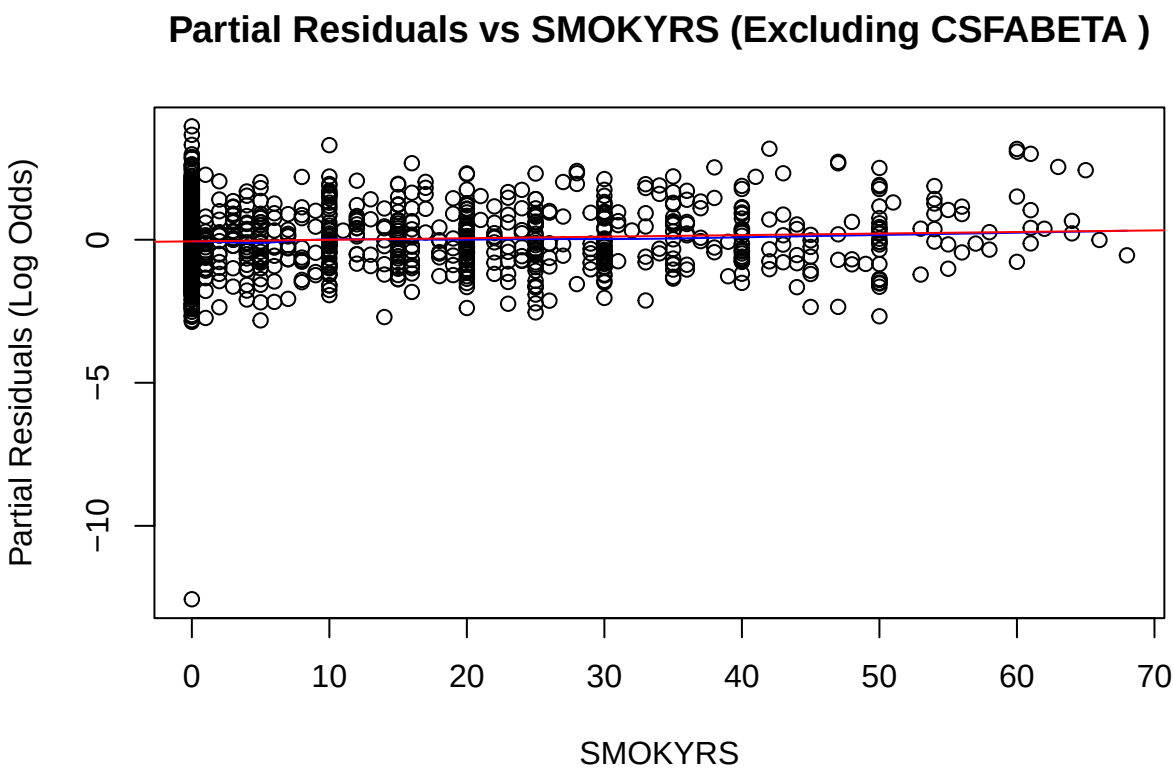

```
## [1] "Processing: NACCAGE"
```



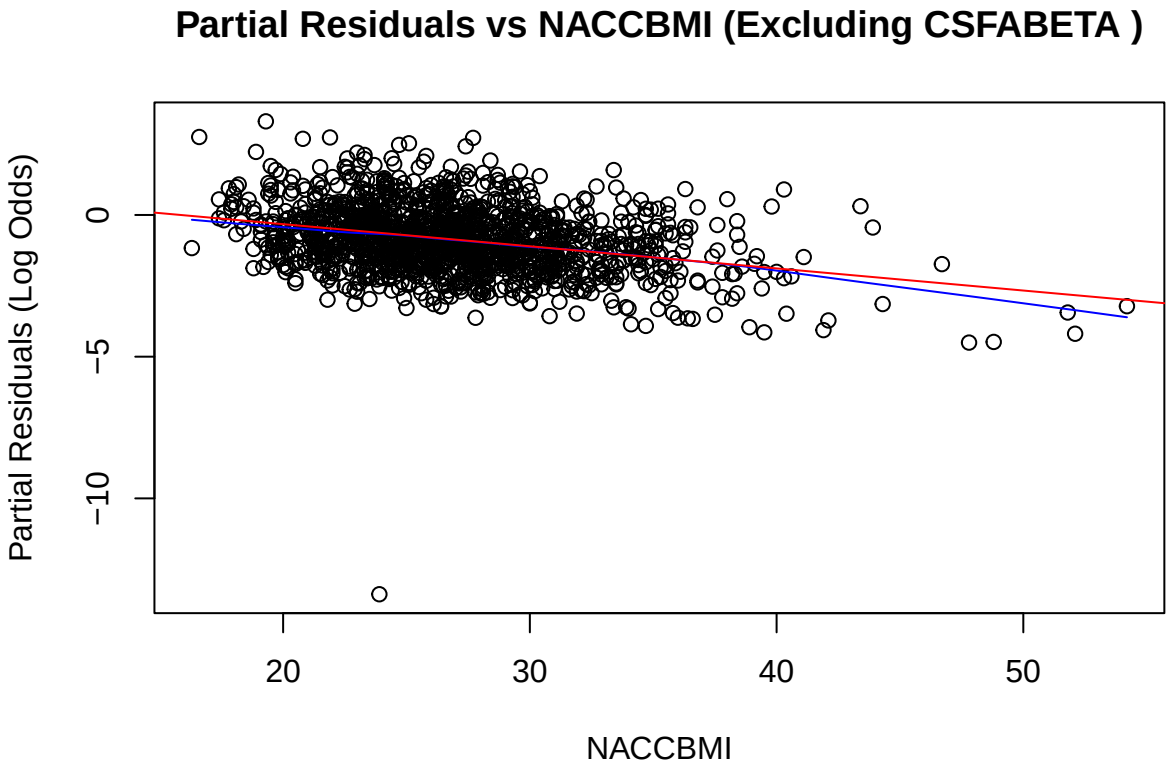

```
## [1] "Processing: EDUC"
```



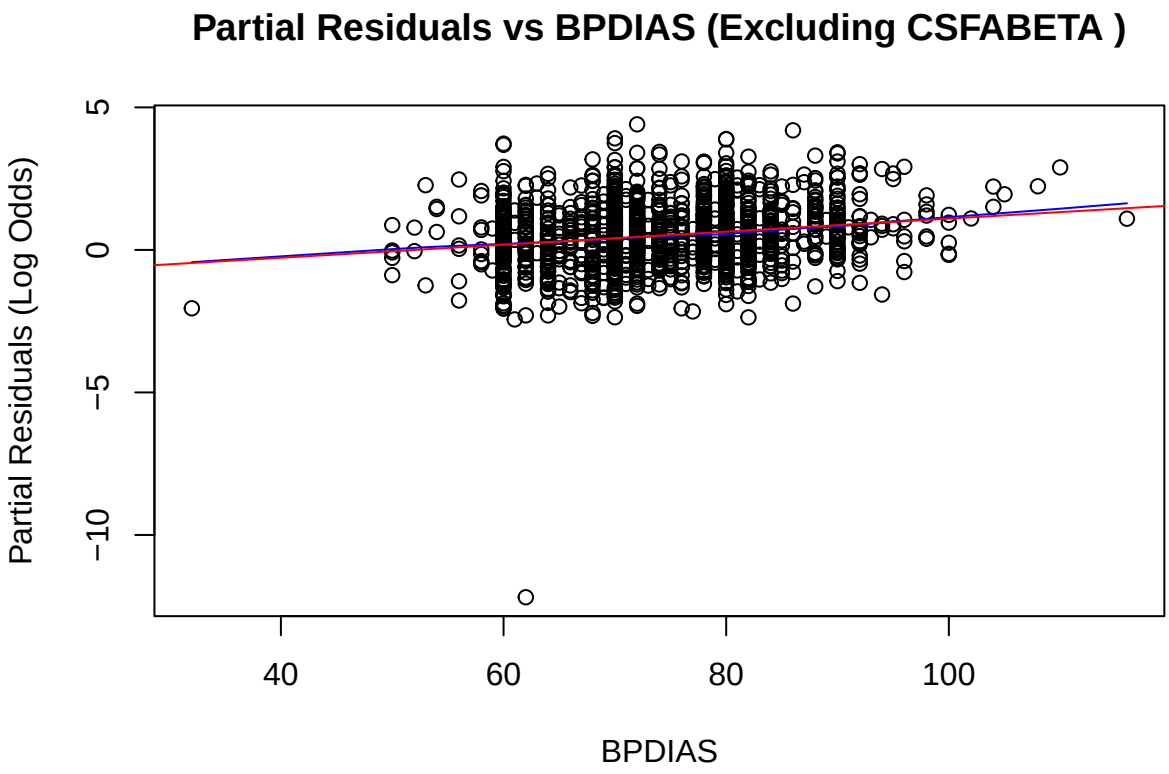

```
## [1] "Processing: SMOKYRS"
```



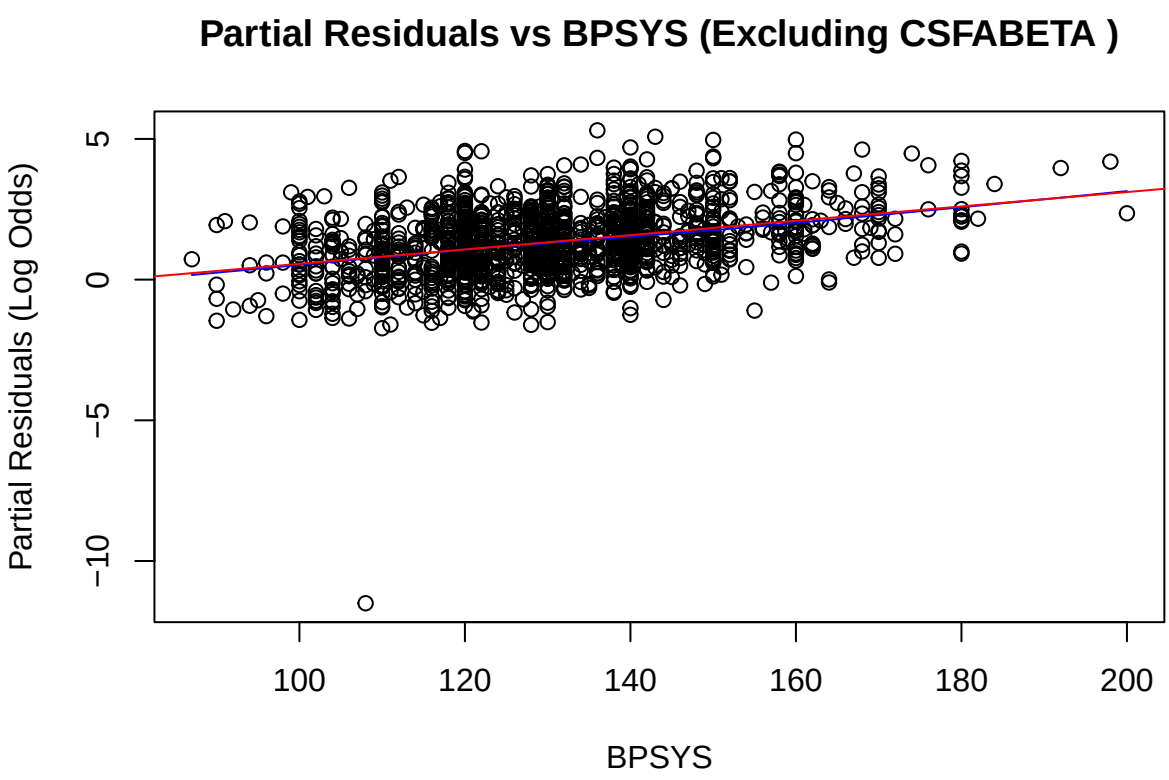

```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```

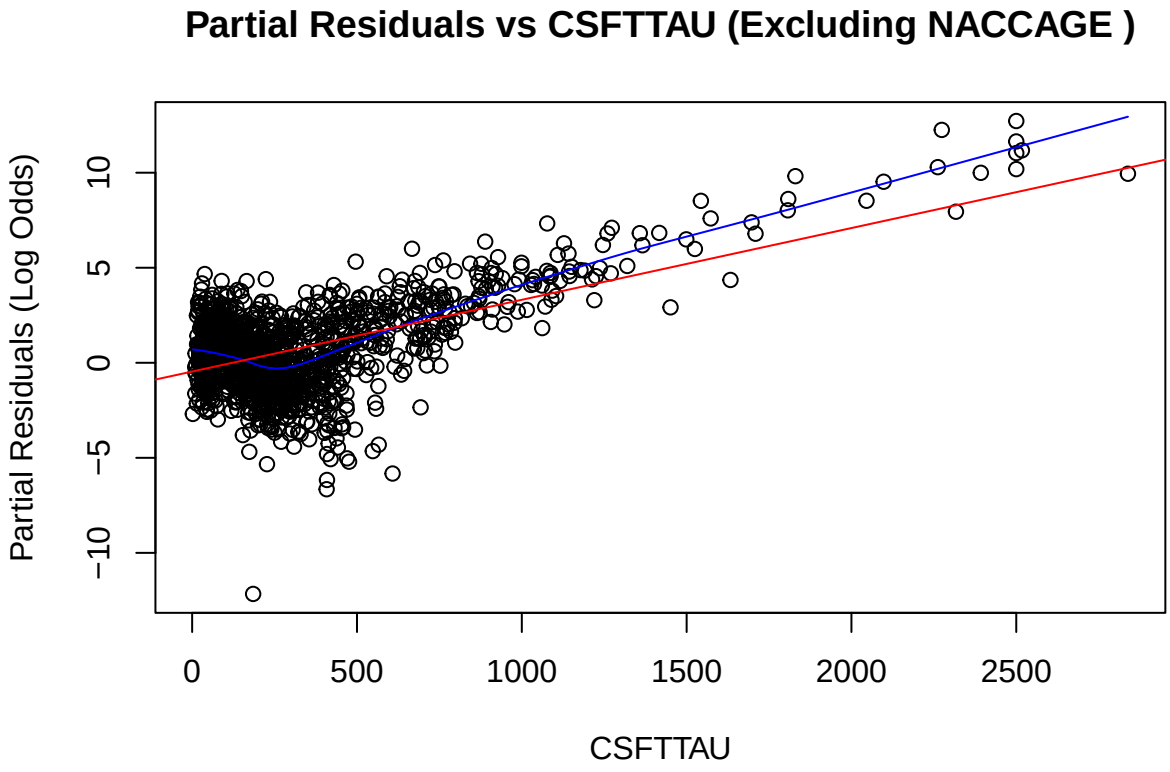



```
## [1] "Processing: BPSYS"
```

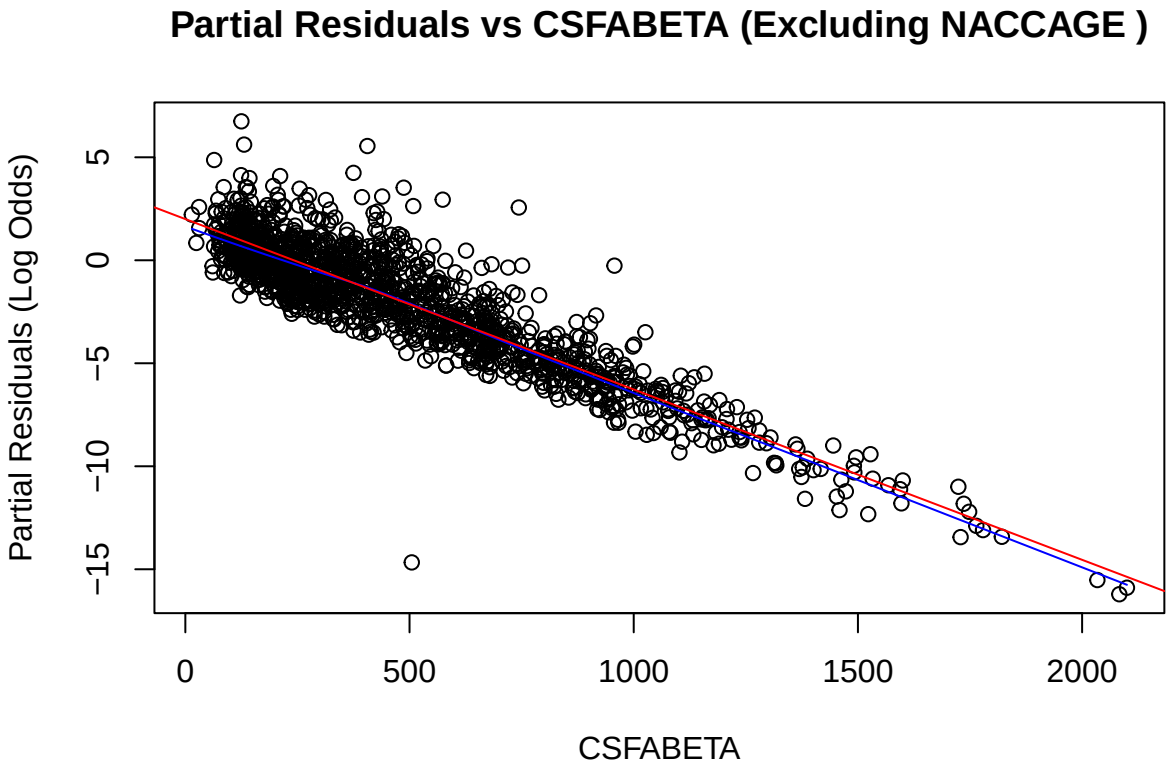



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

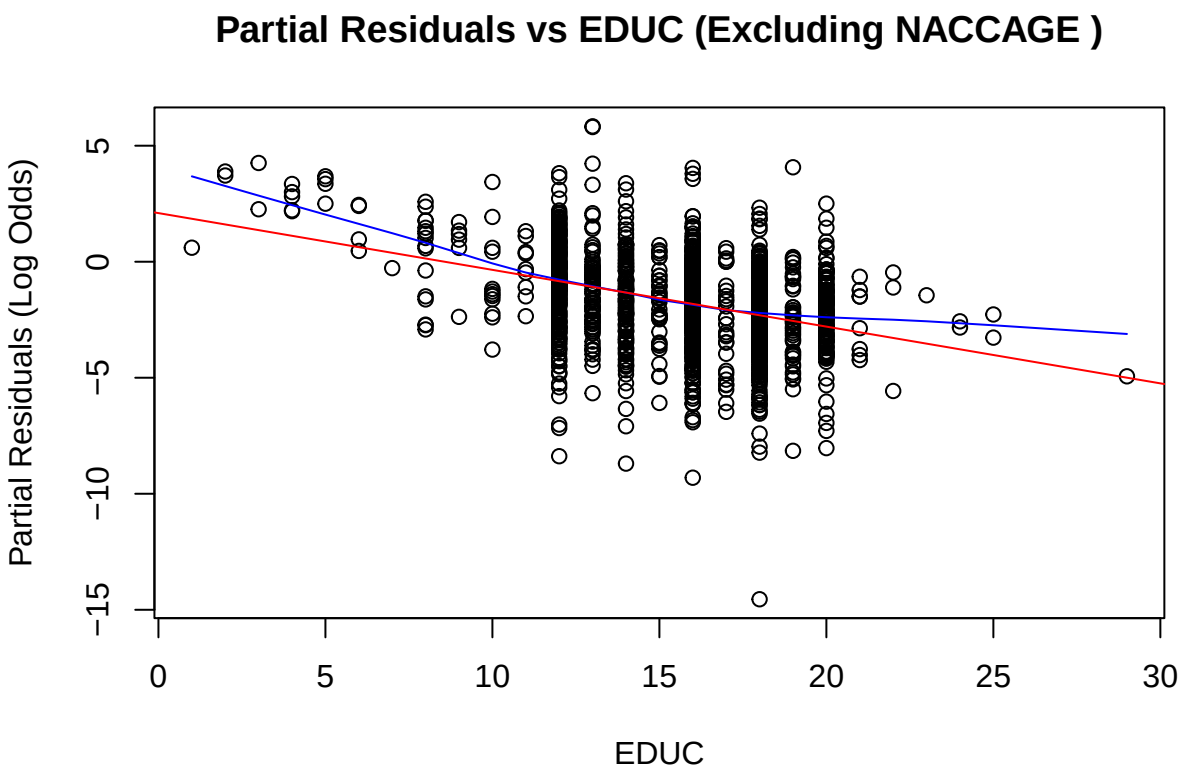
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



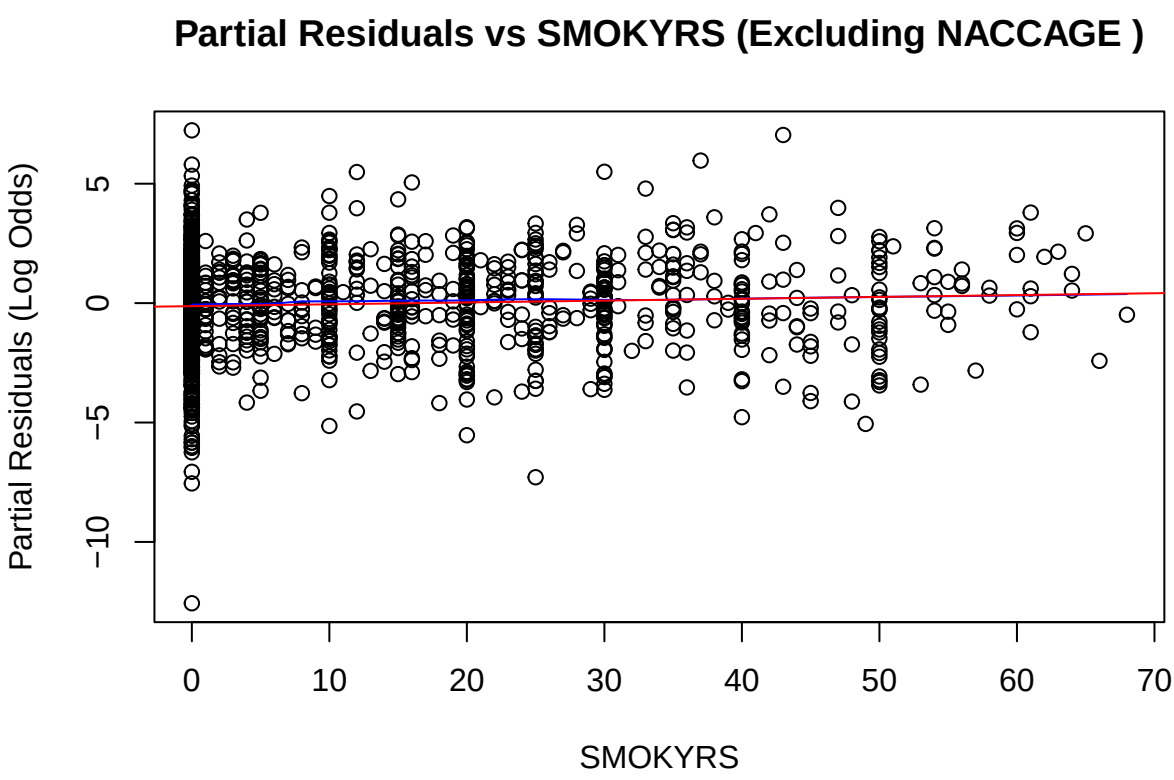

```
## [1] "Processing: CSFABETA"
```



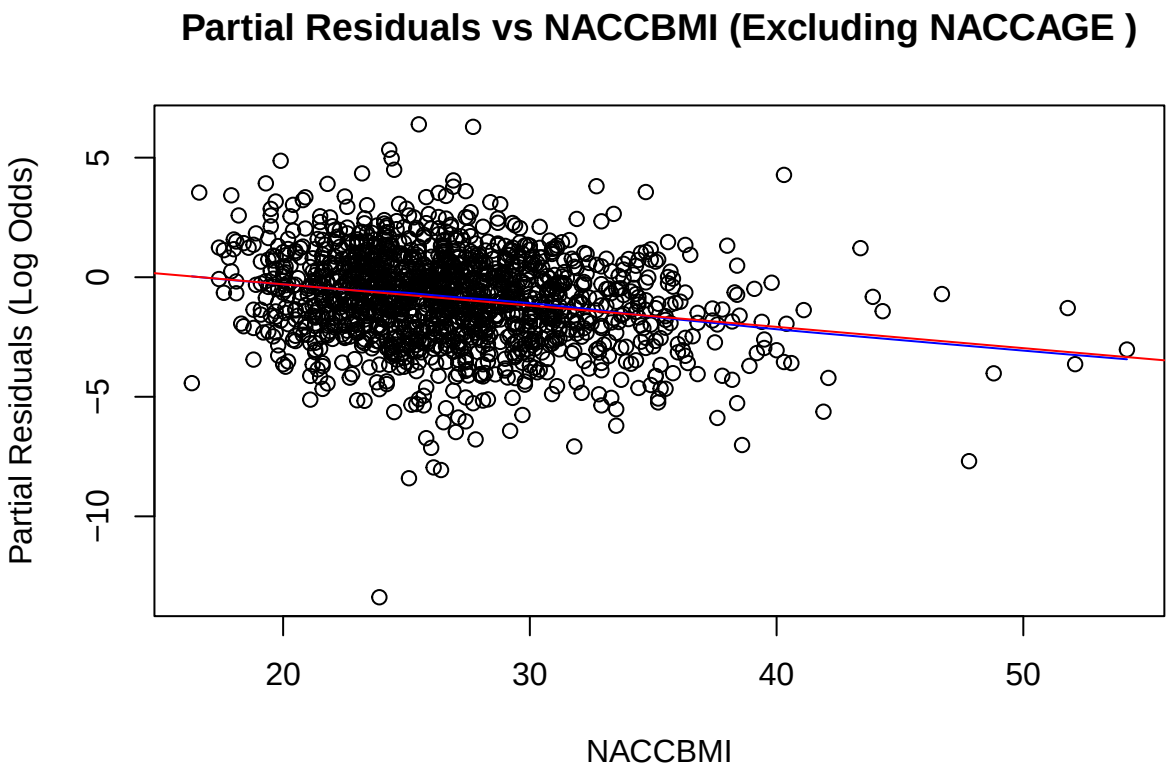

```
## [1] "Processing: EDUC"
```



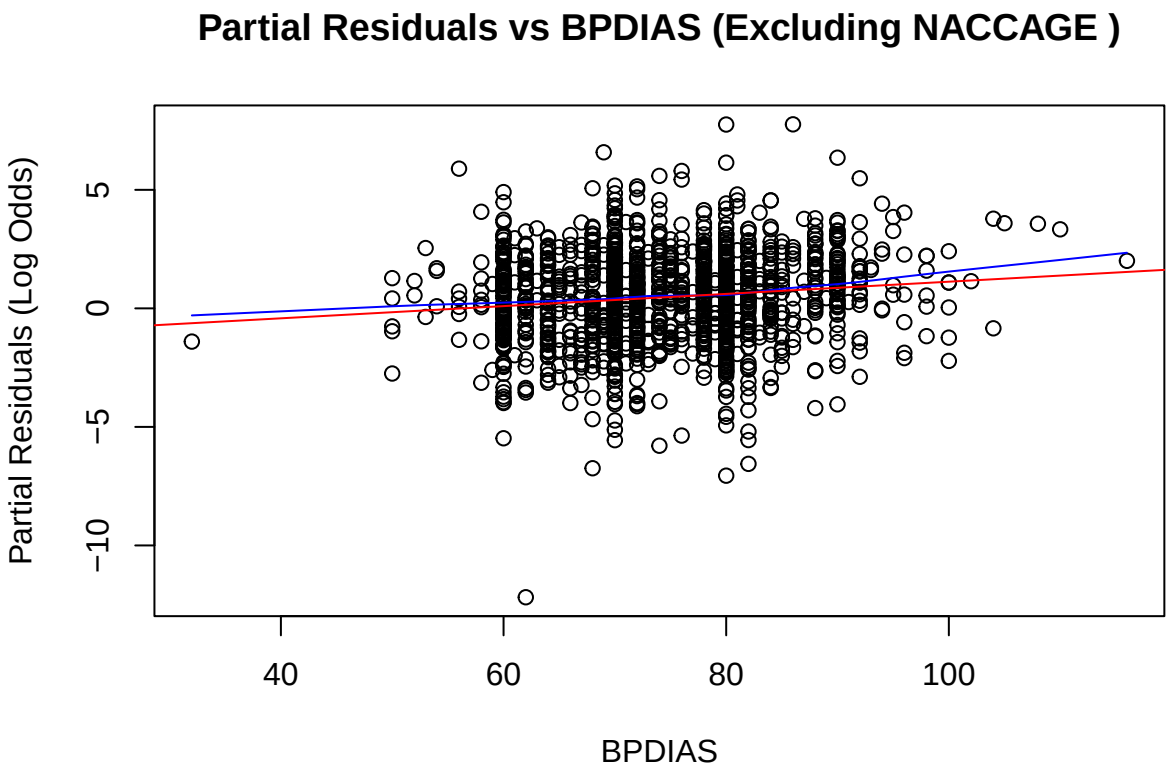

```
## [1] "Processing: SMOKYRS"
```



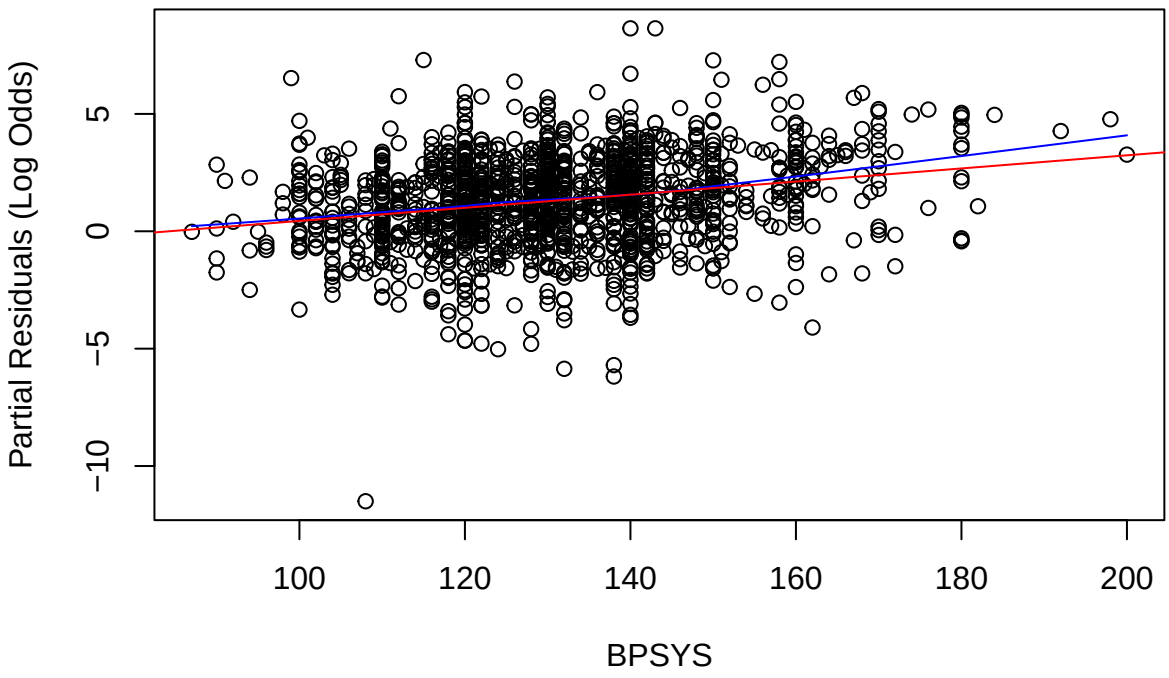

```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```

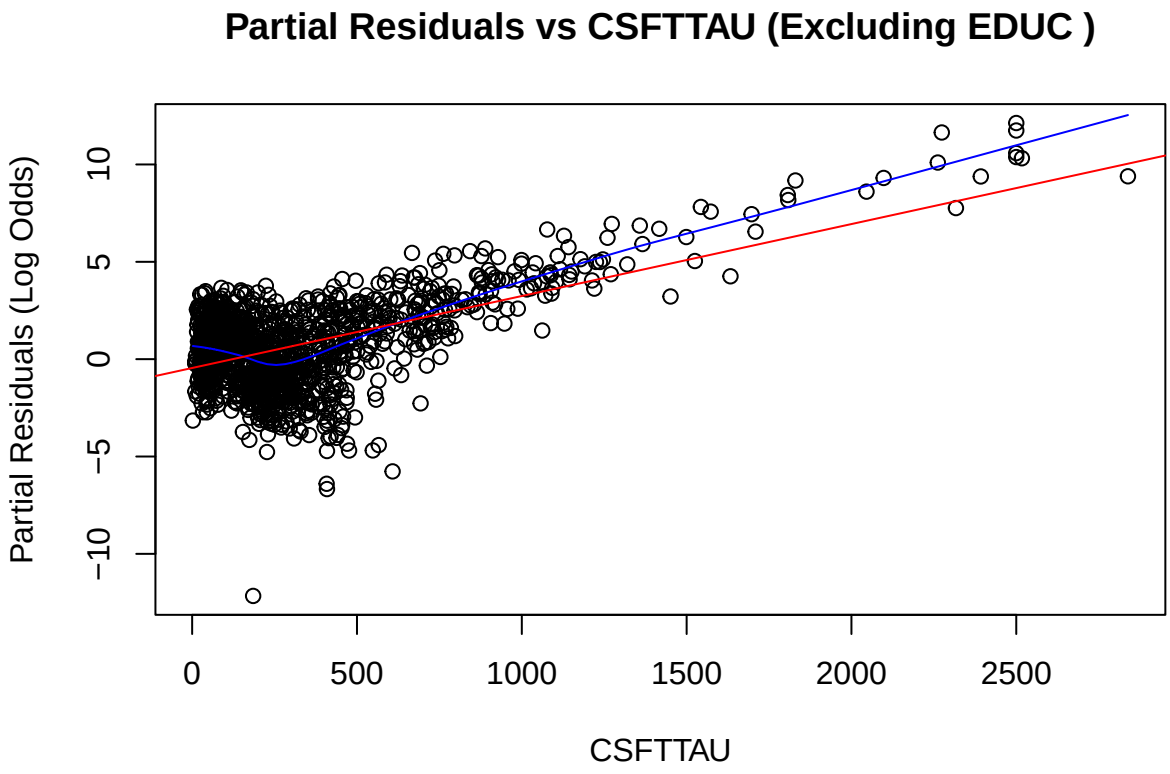



```
## [1] "Processing: BPSYS"
```

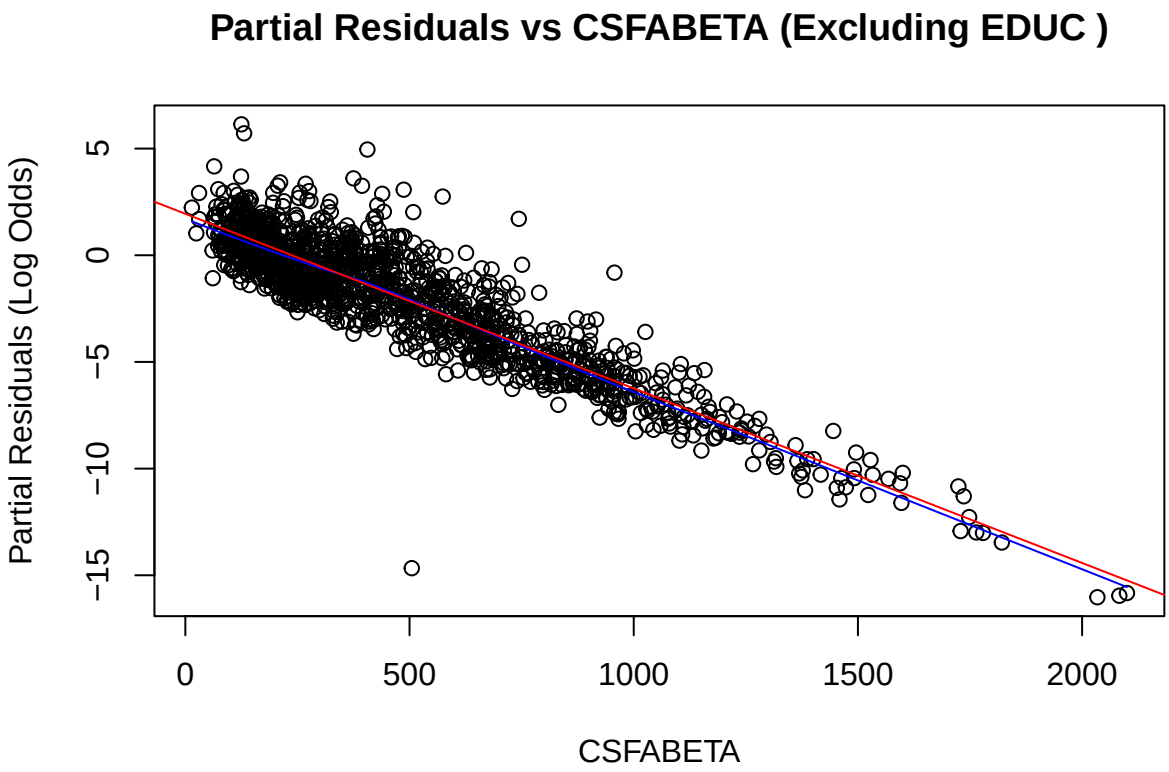



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

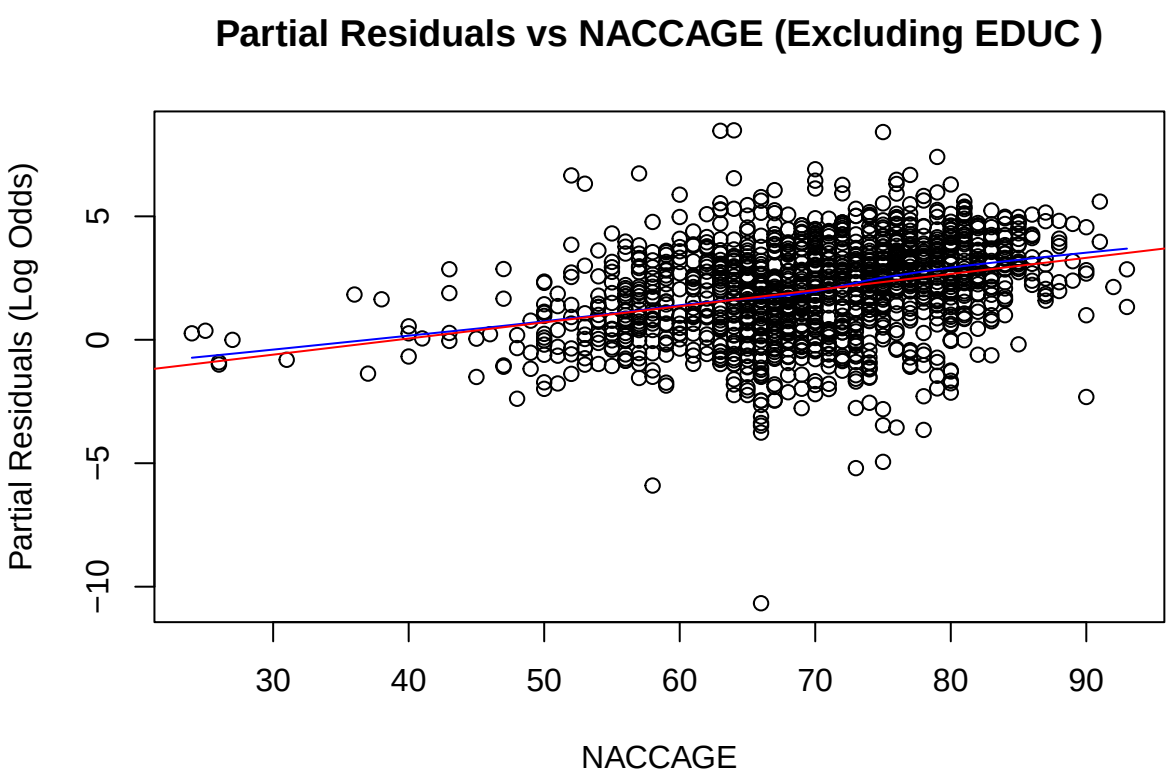
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



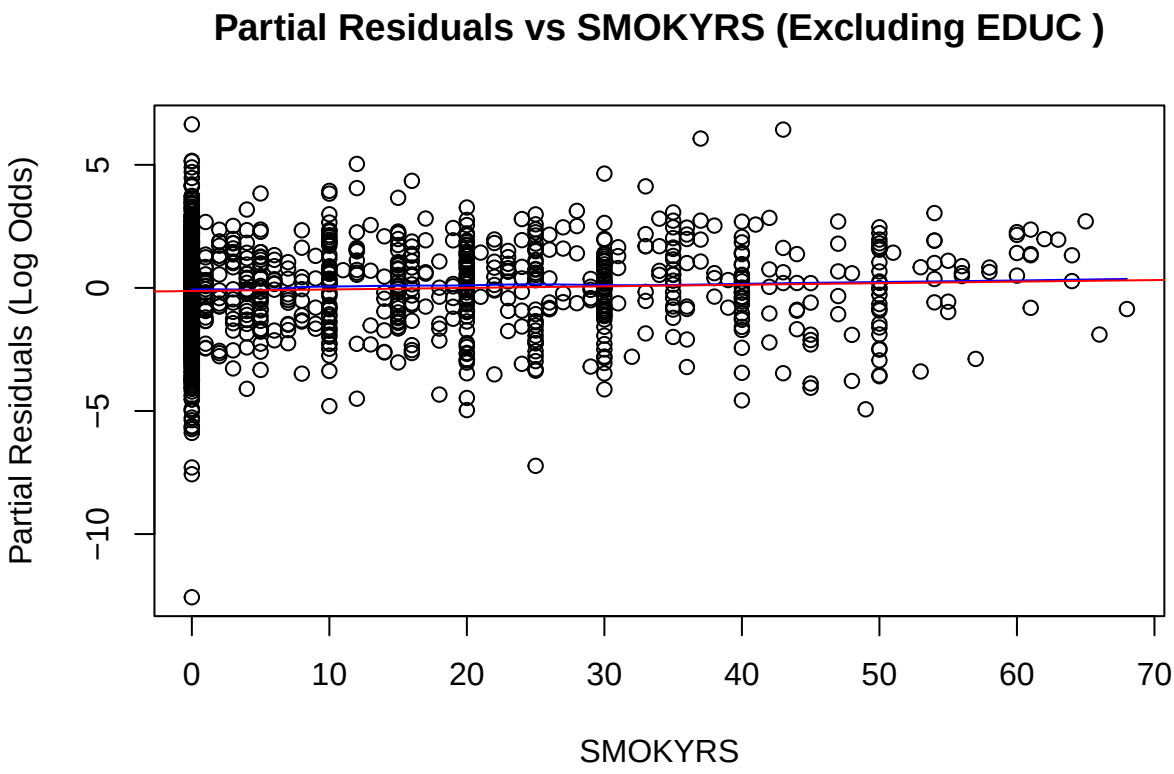

```
## [1] "Processing: CSFABETA"
```



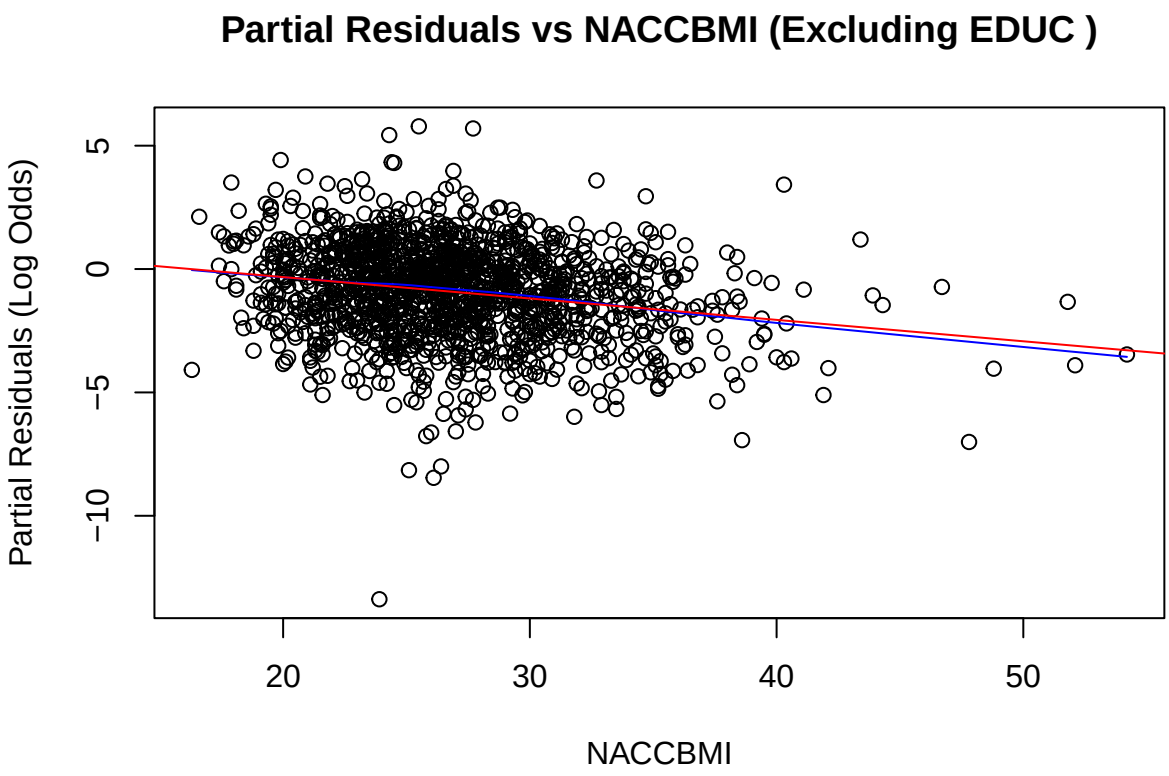

```
## [1] "Processing: NACCAGE"
```



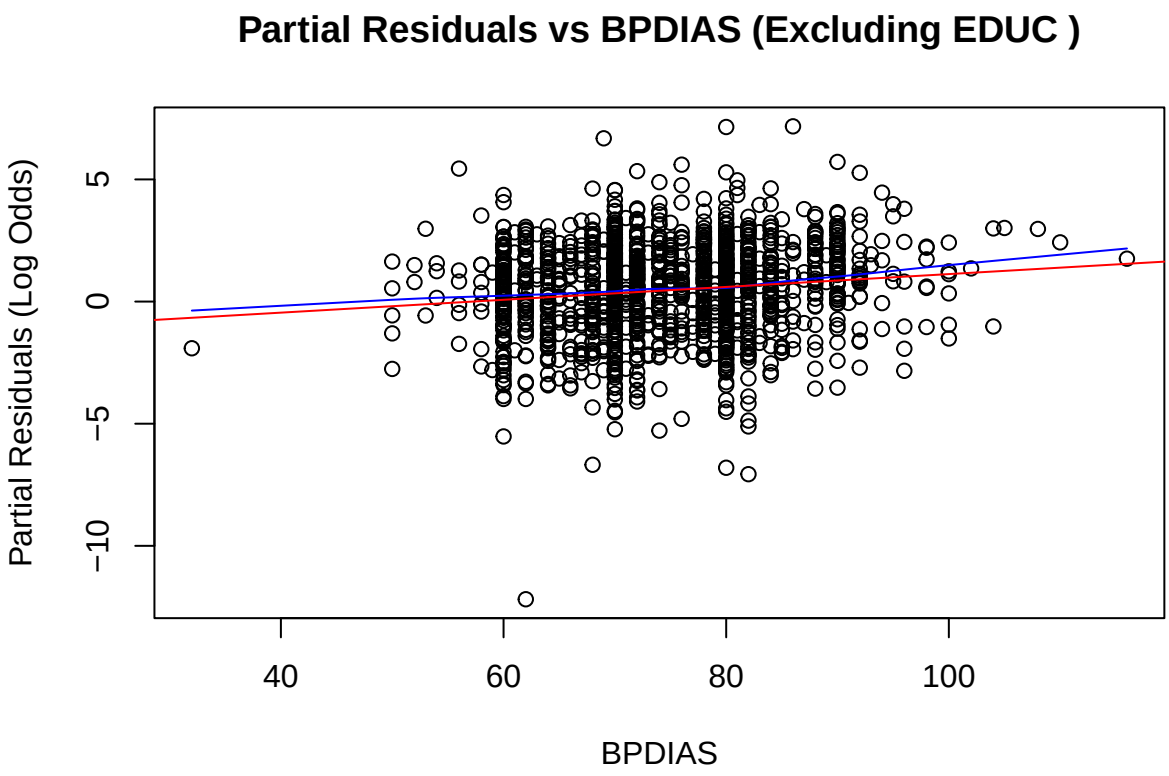

```
## [1] "Processing: SMOKYRS"
```



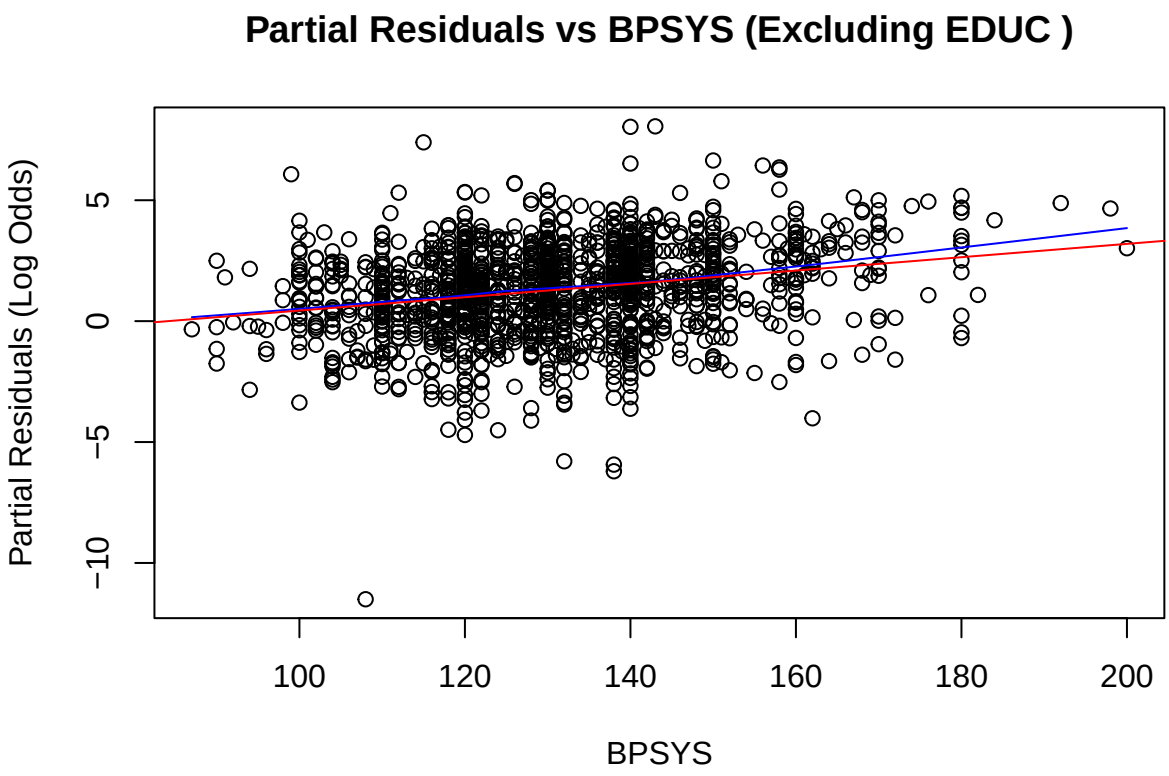

```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```

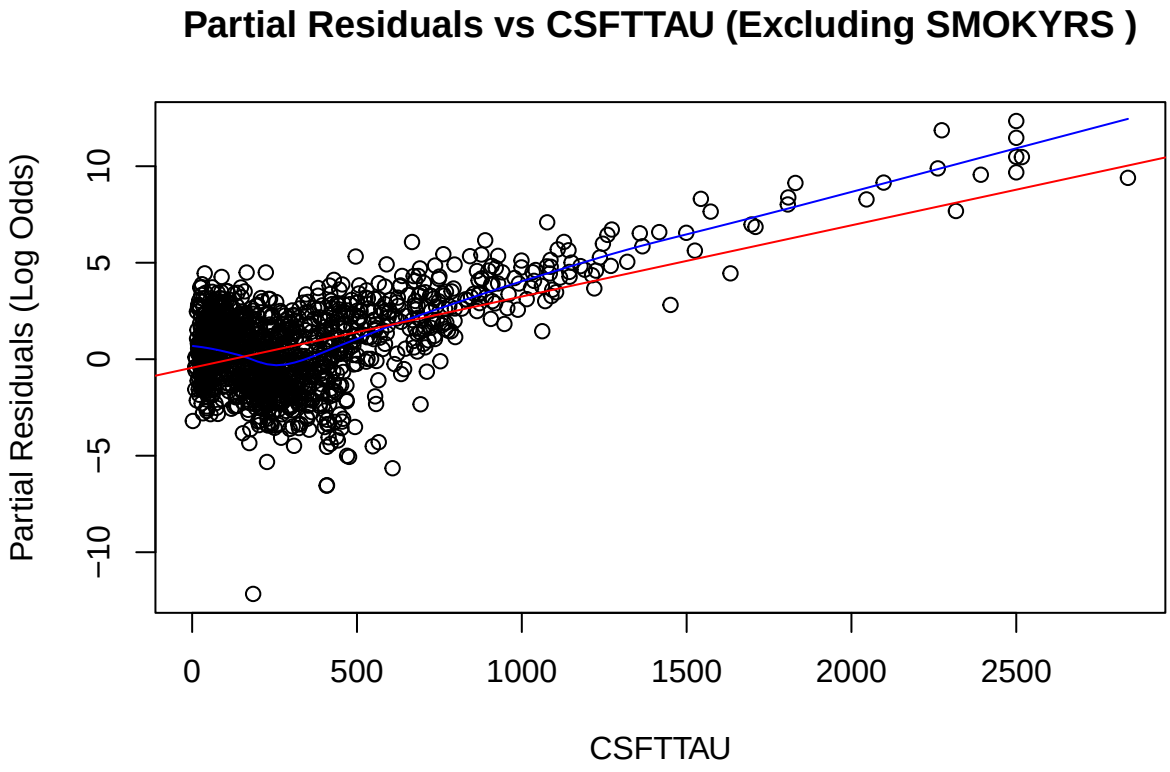



```
## [1] "Processing: BPSYS"
```

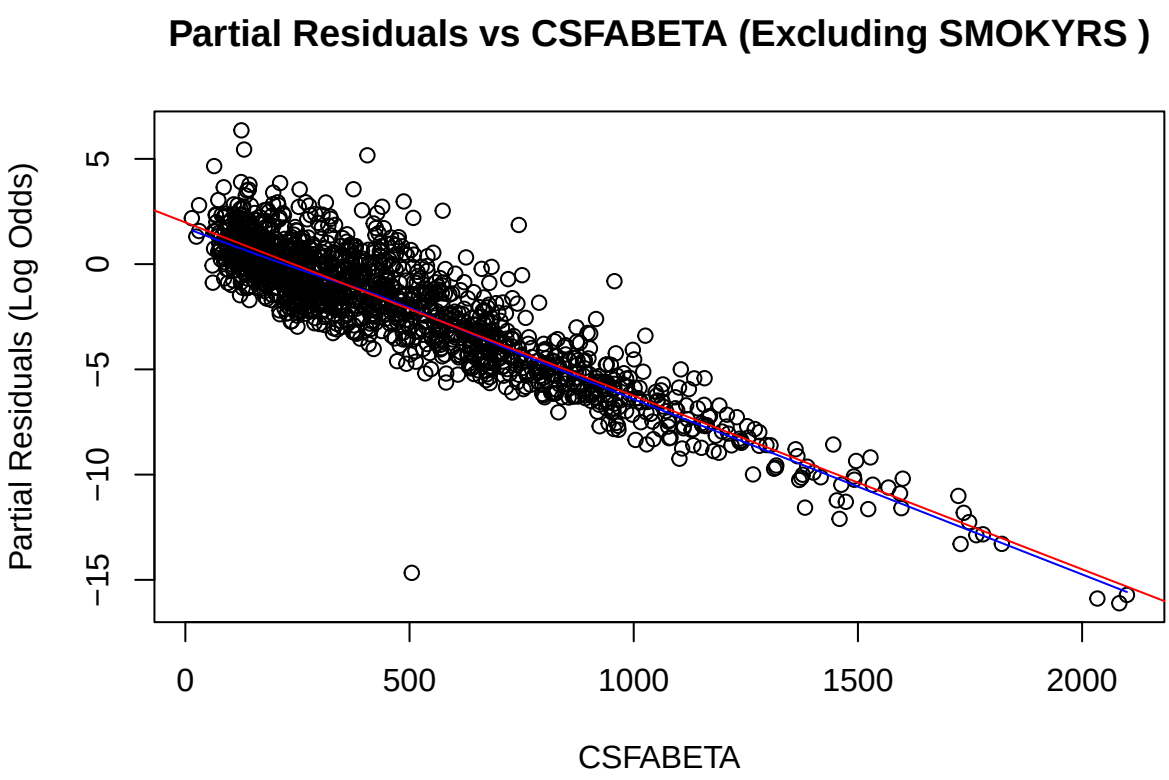



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

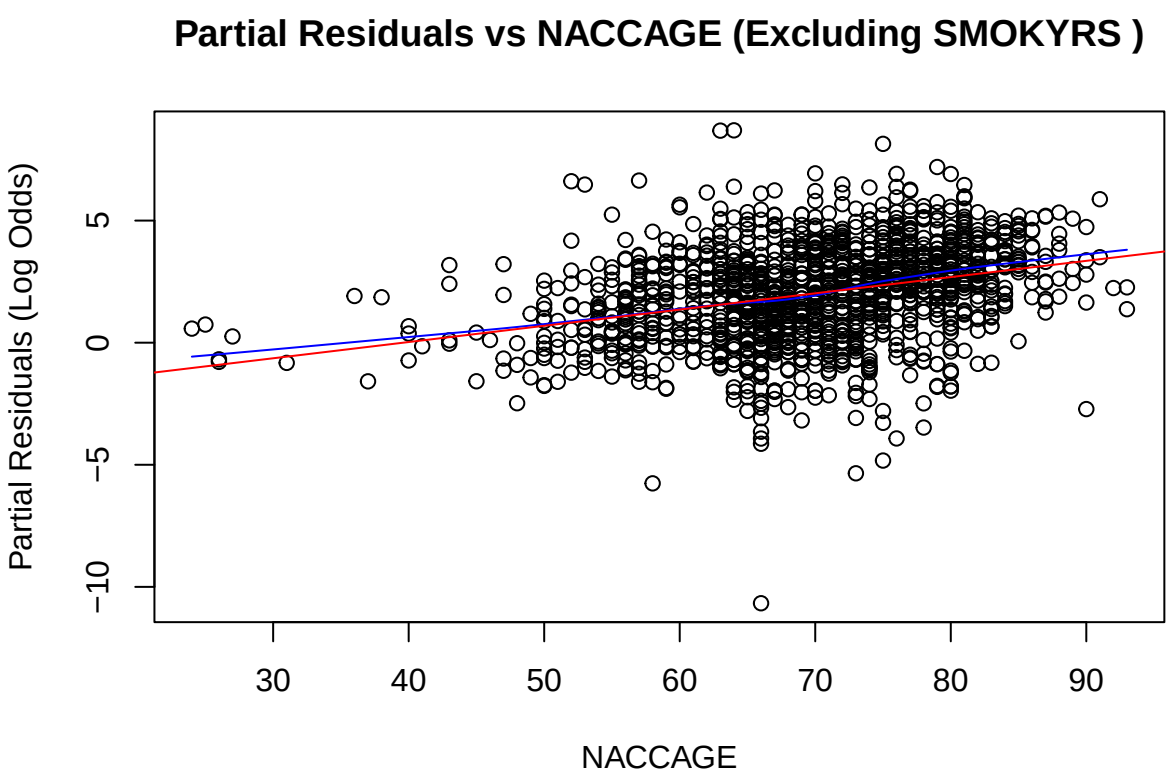
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



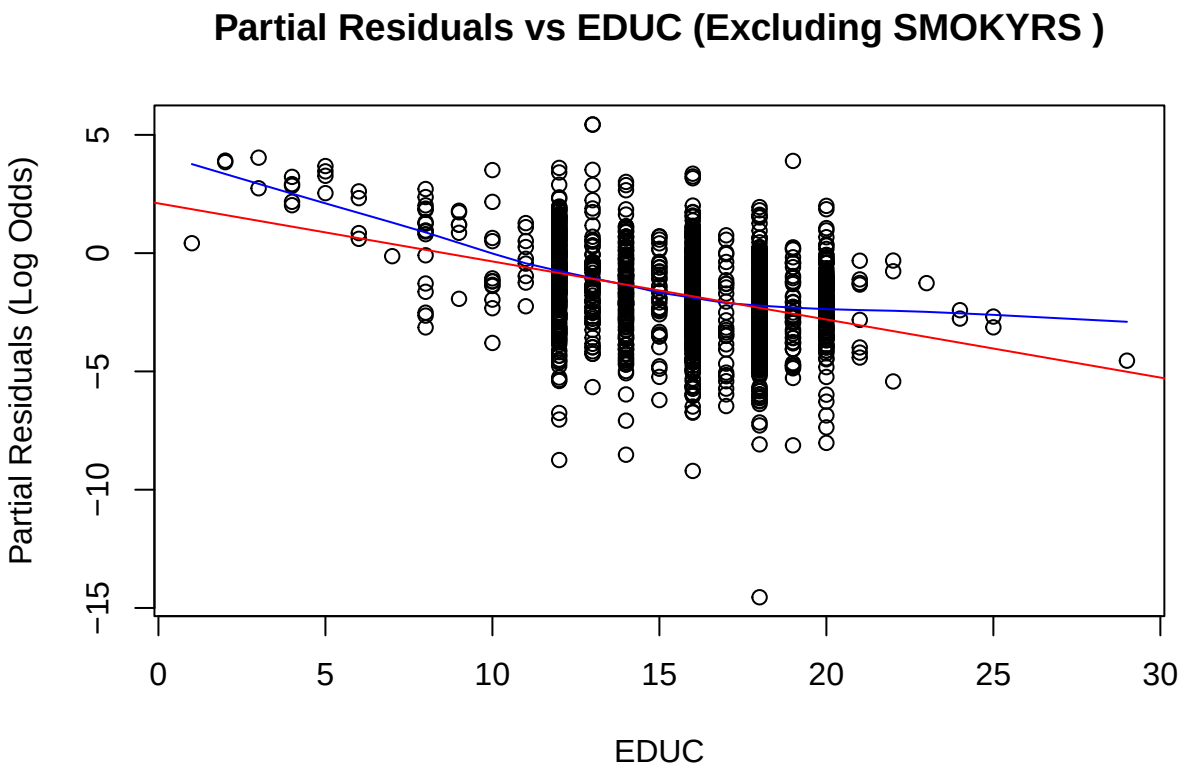

```
## [1] "Processing: CSFABETA"
```



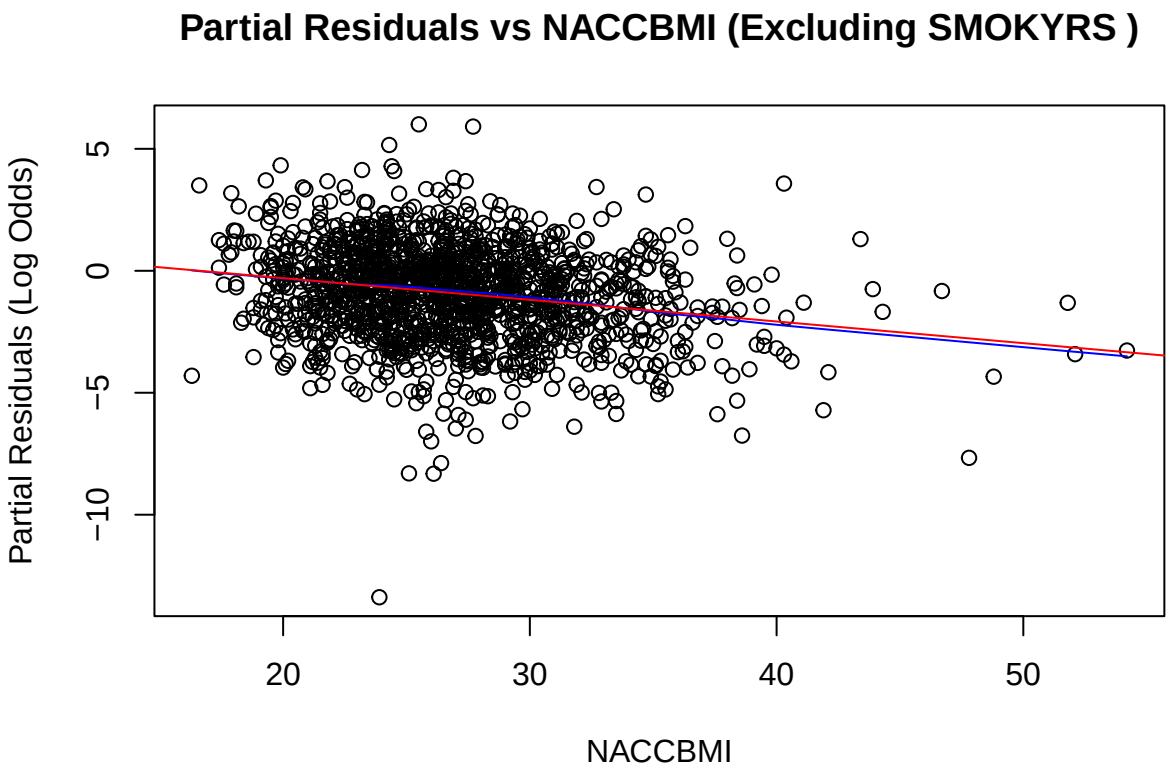

```
## [1] "Processing: NACCAGE"
```



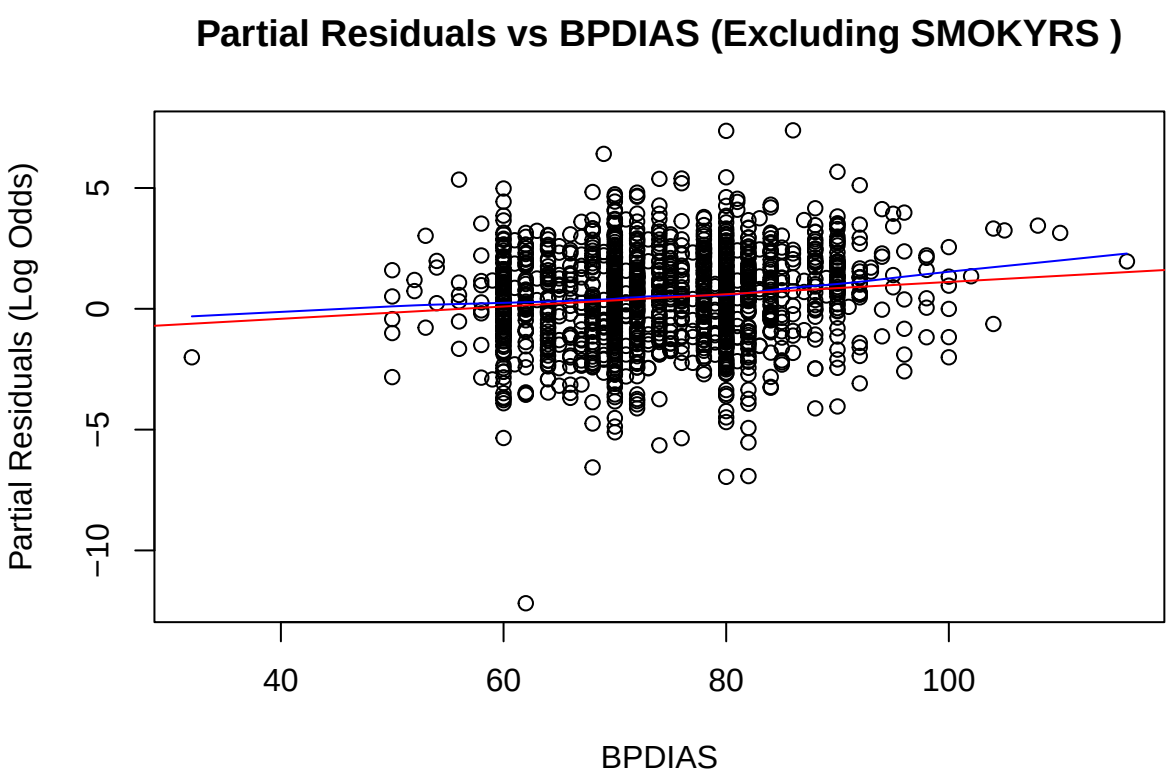

```
## [1] "Processing: EDUC"
```



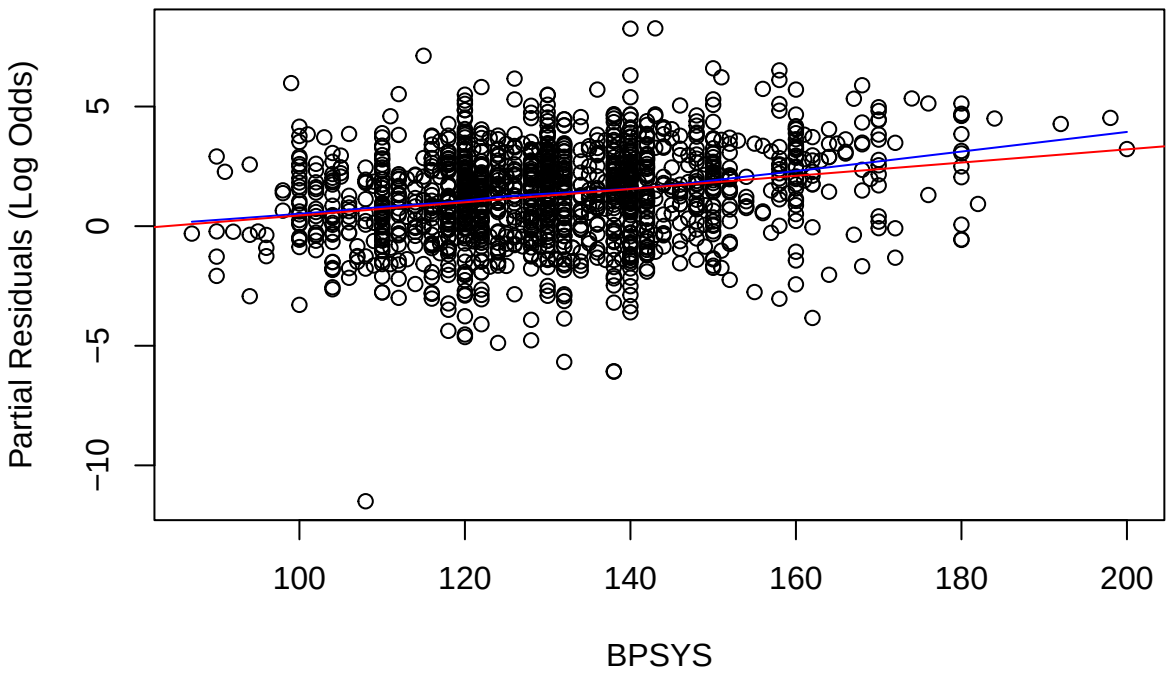

```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```

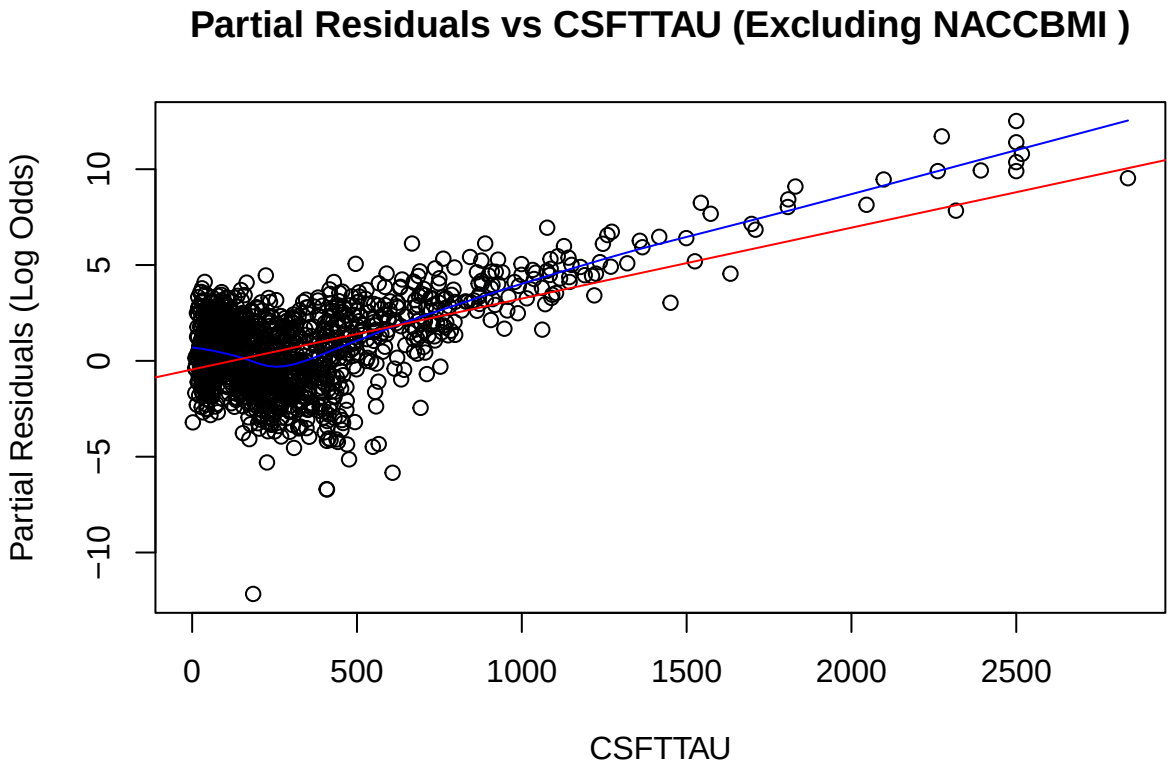



```
## [1] "Processing: BPSYS"
```

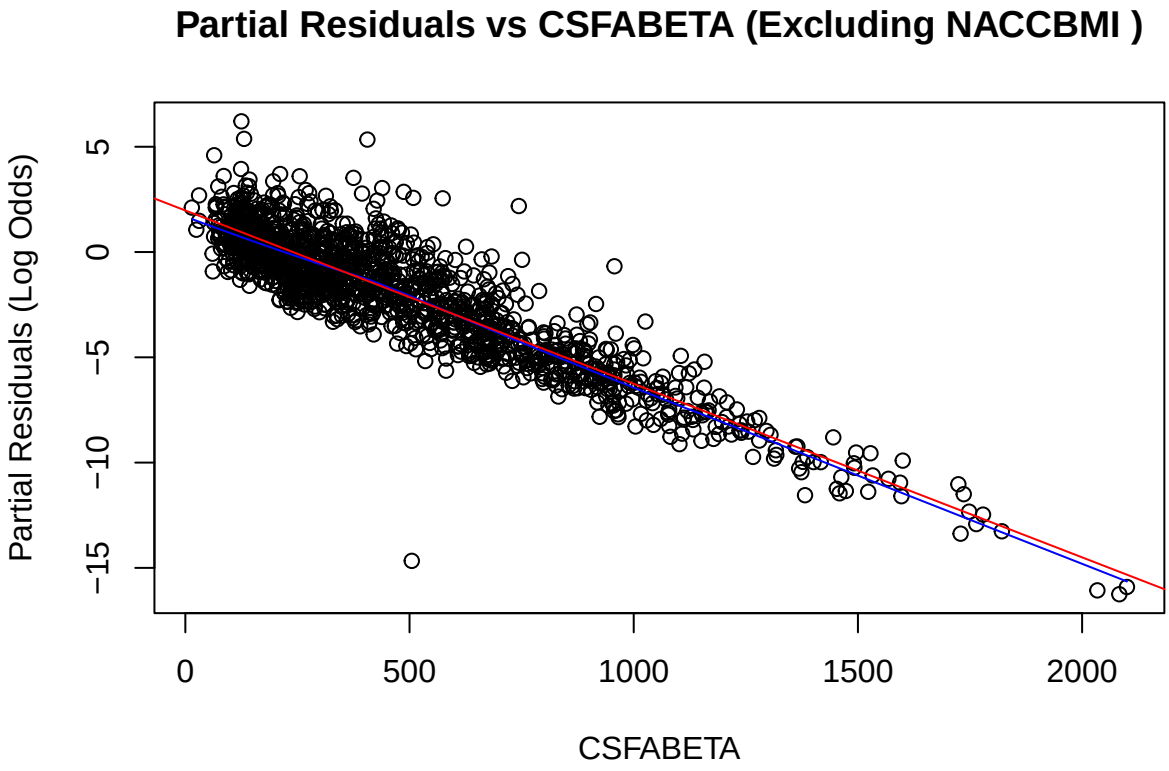



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

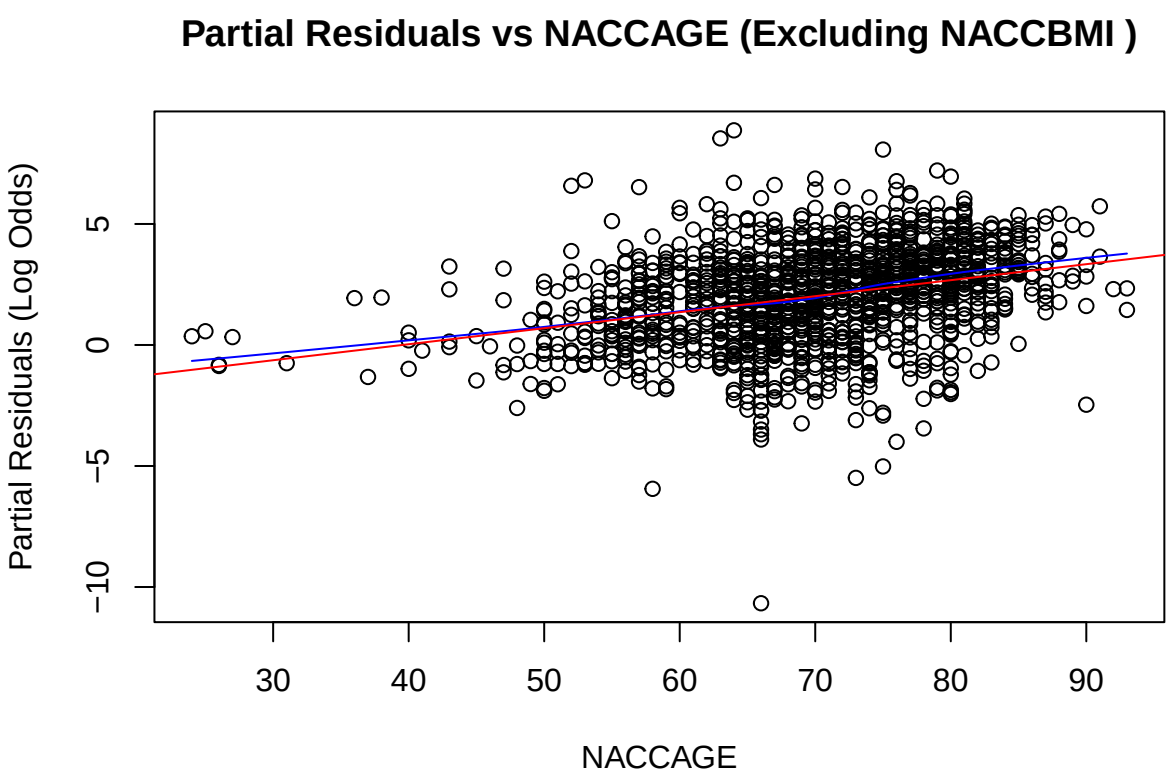
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



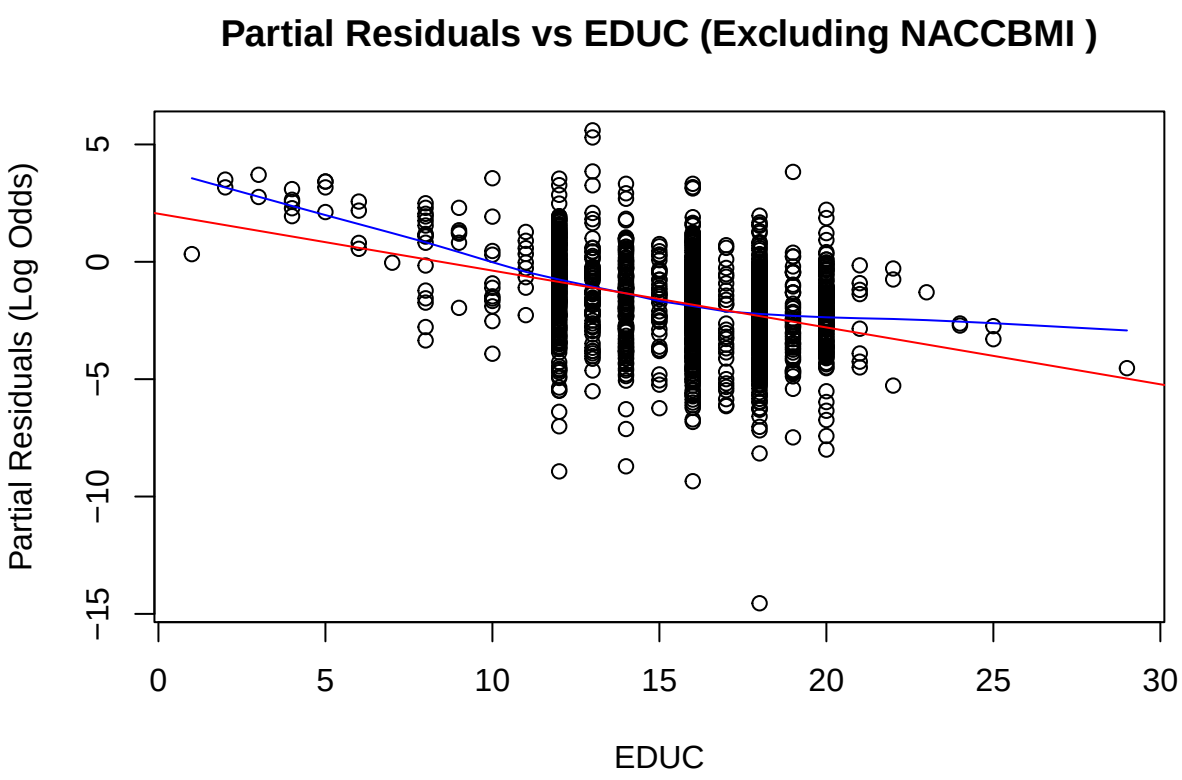

```
## [1] "Processing: CSFABETA"
```



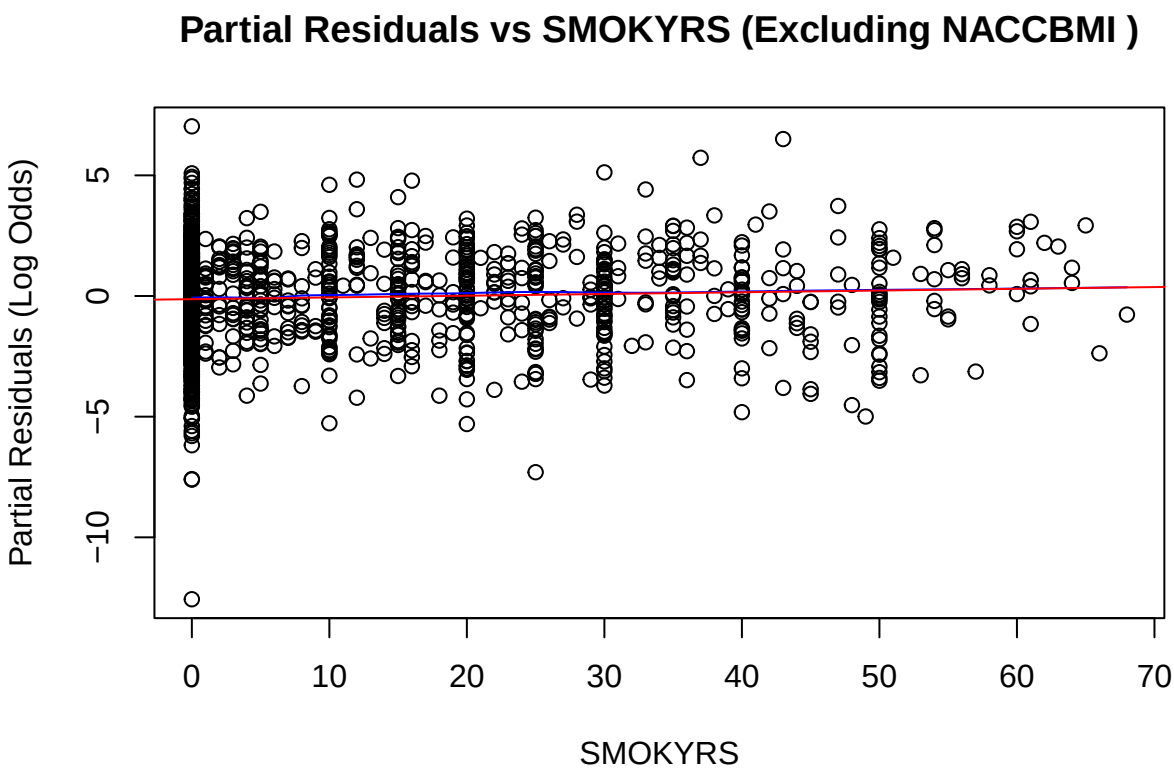

```
## [1] "Processing: NACCAGE"
```



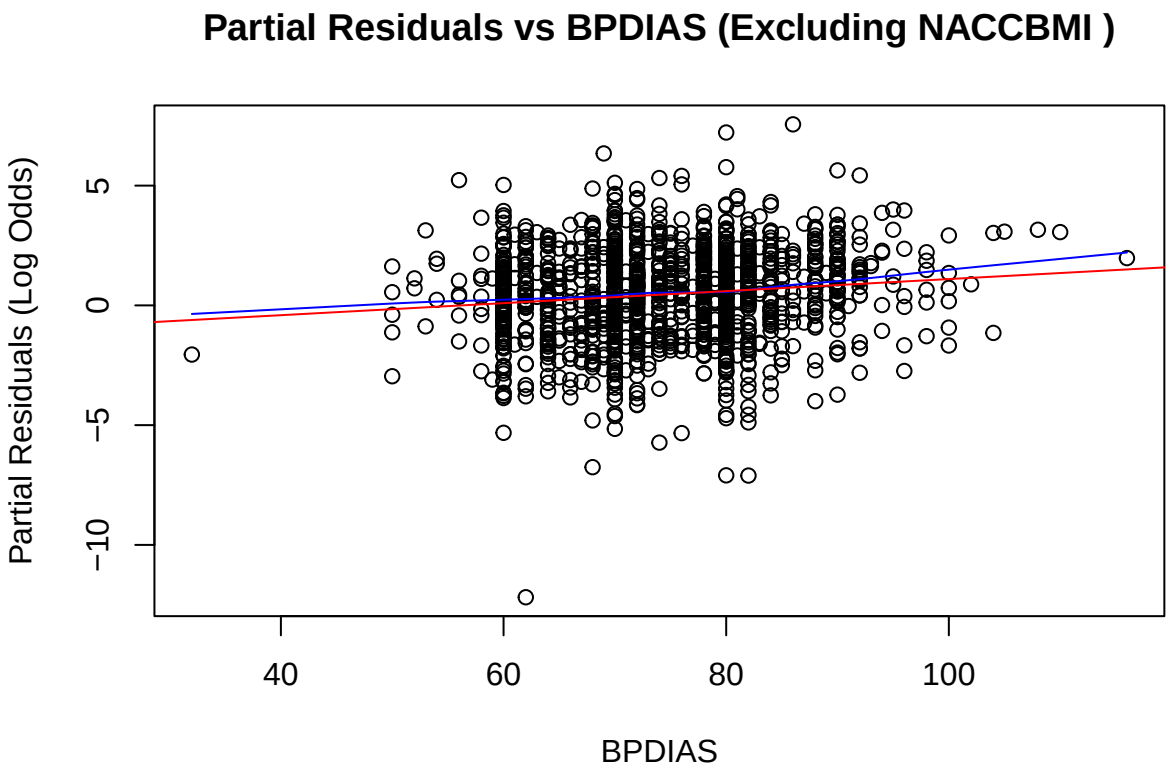

```
## [1] "Processing: EDUC"
```



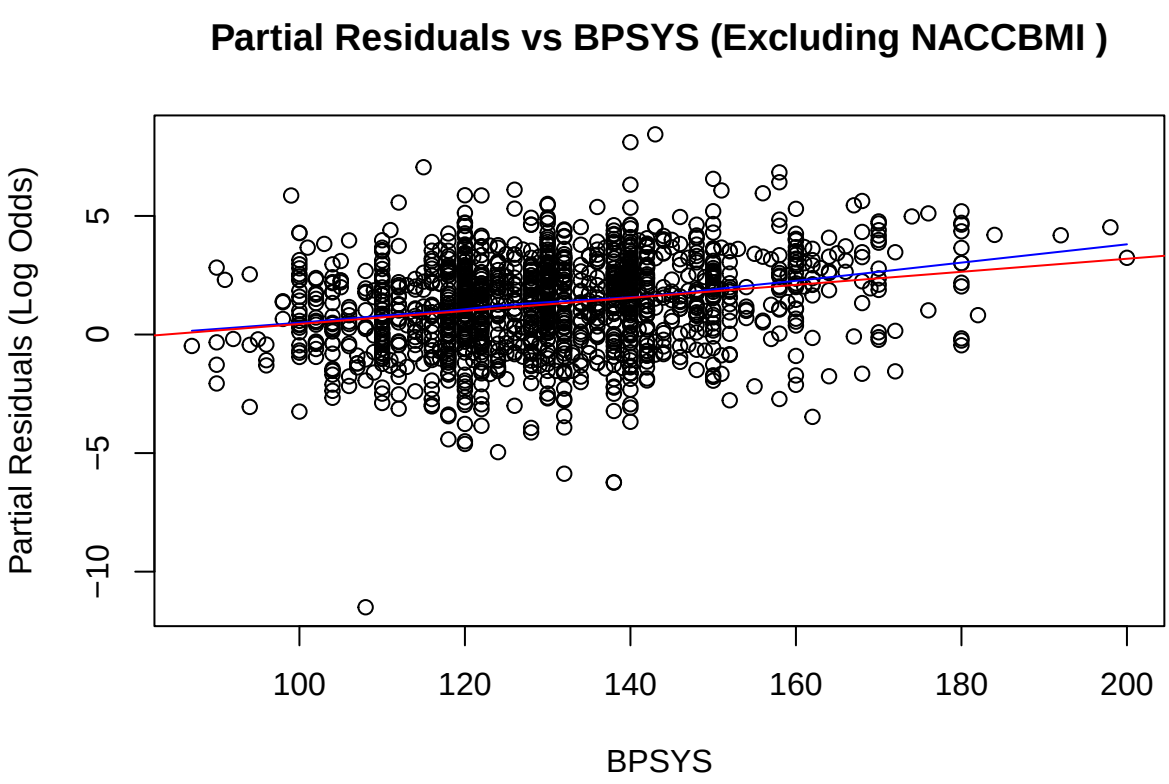

```
## [1] "Processing: SMOKYRS"
```




```
## [1] "Processing: BPDIAS"
```

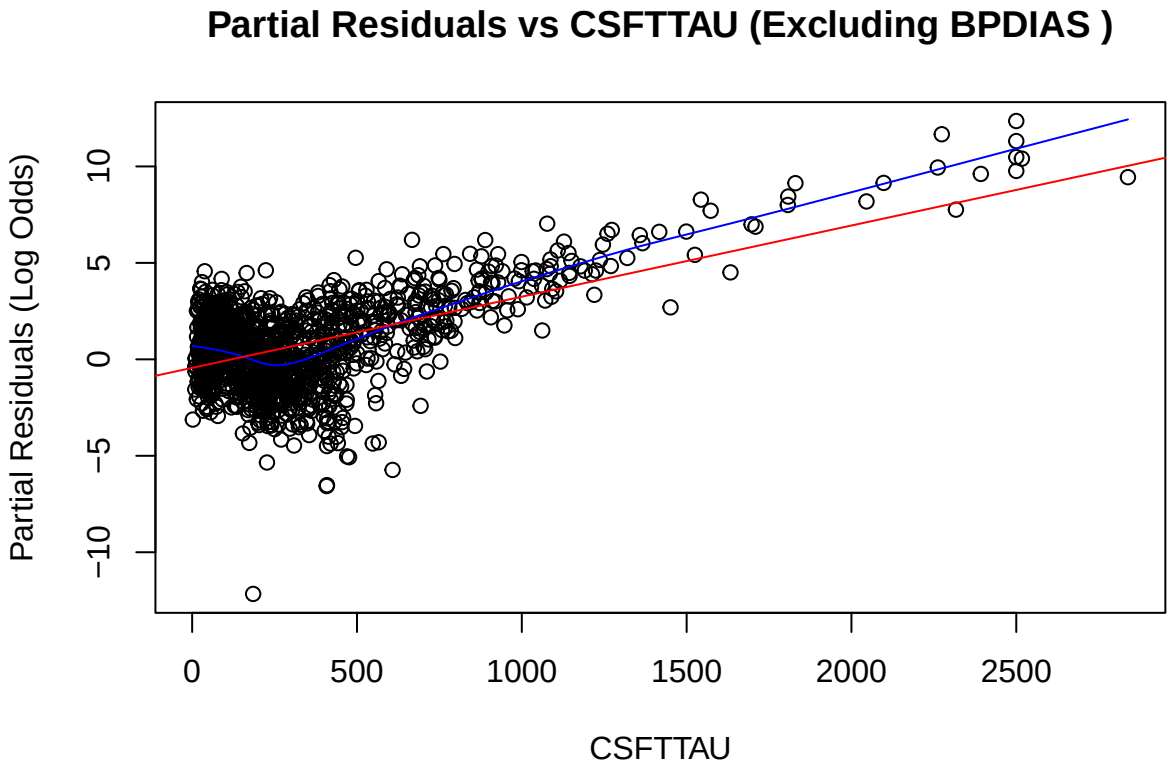



```
## [1] "Processing: BPSYS"
```

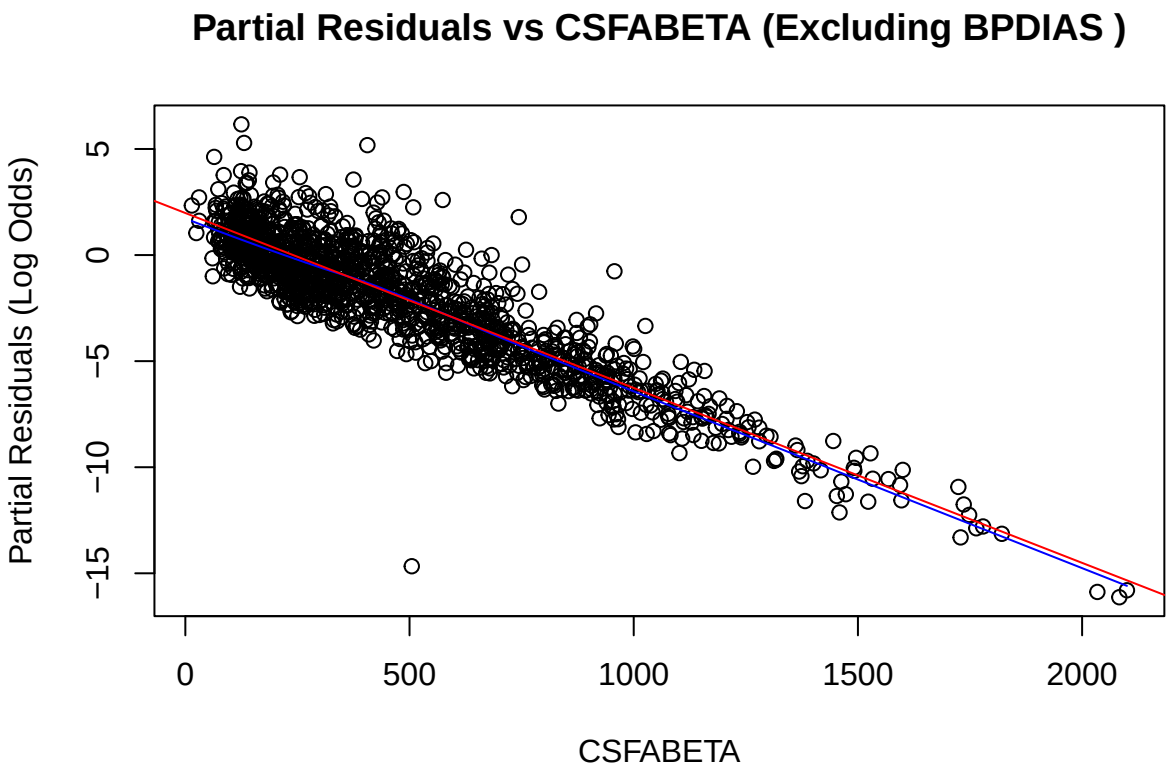



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

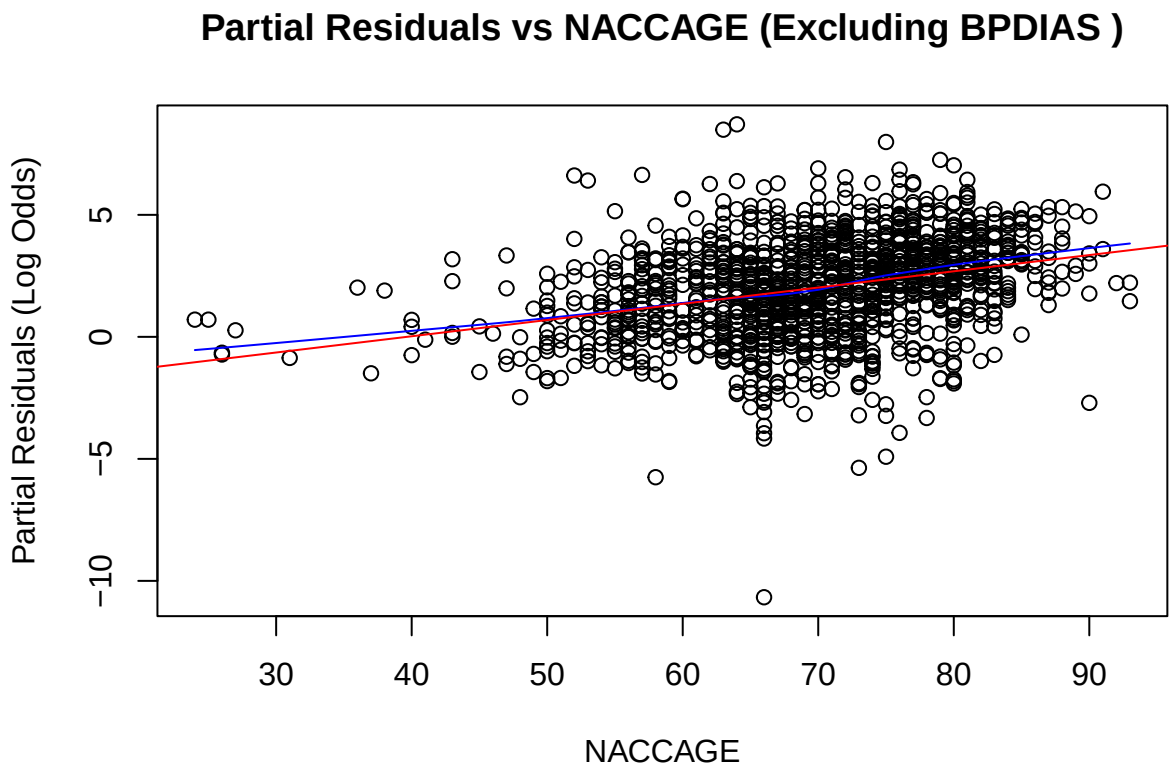
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



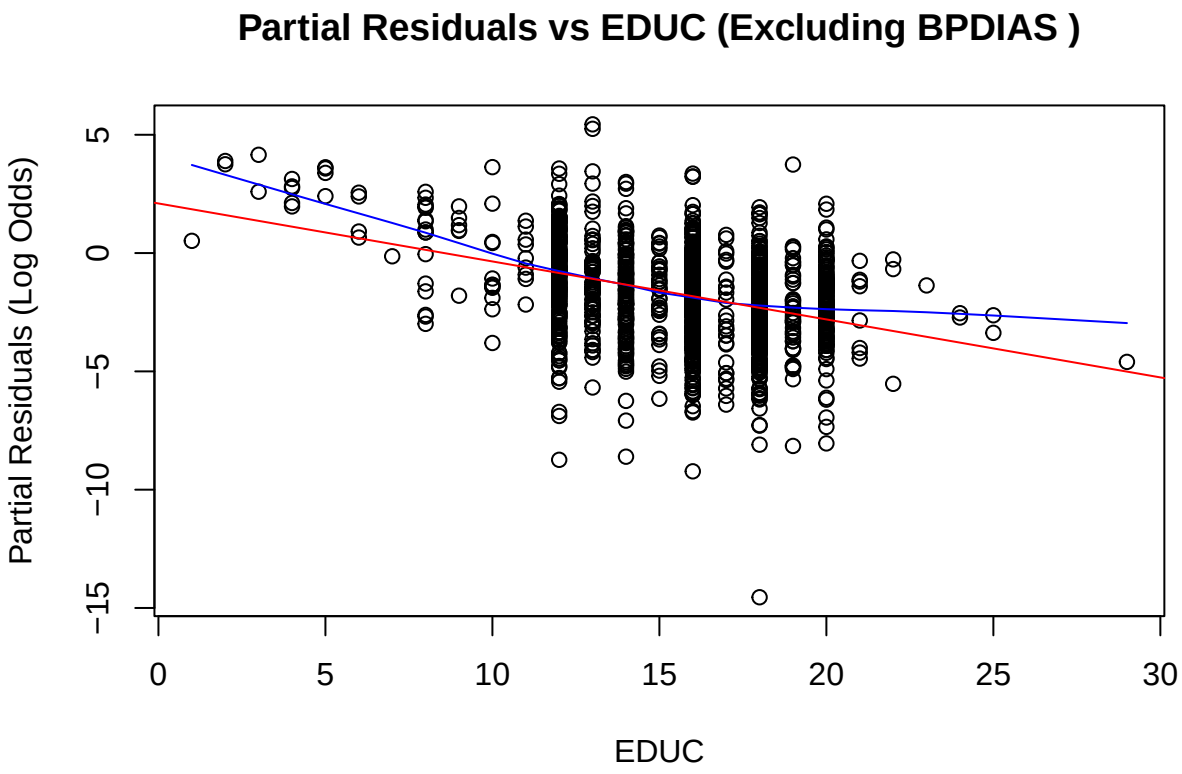

```
## [1] "Processing: CSFABETA"
```



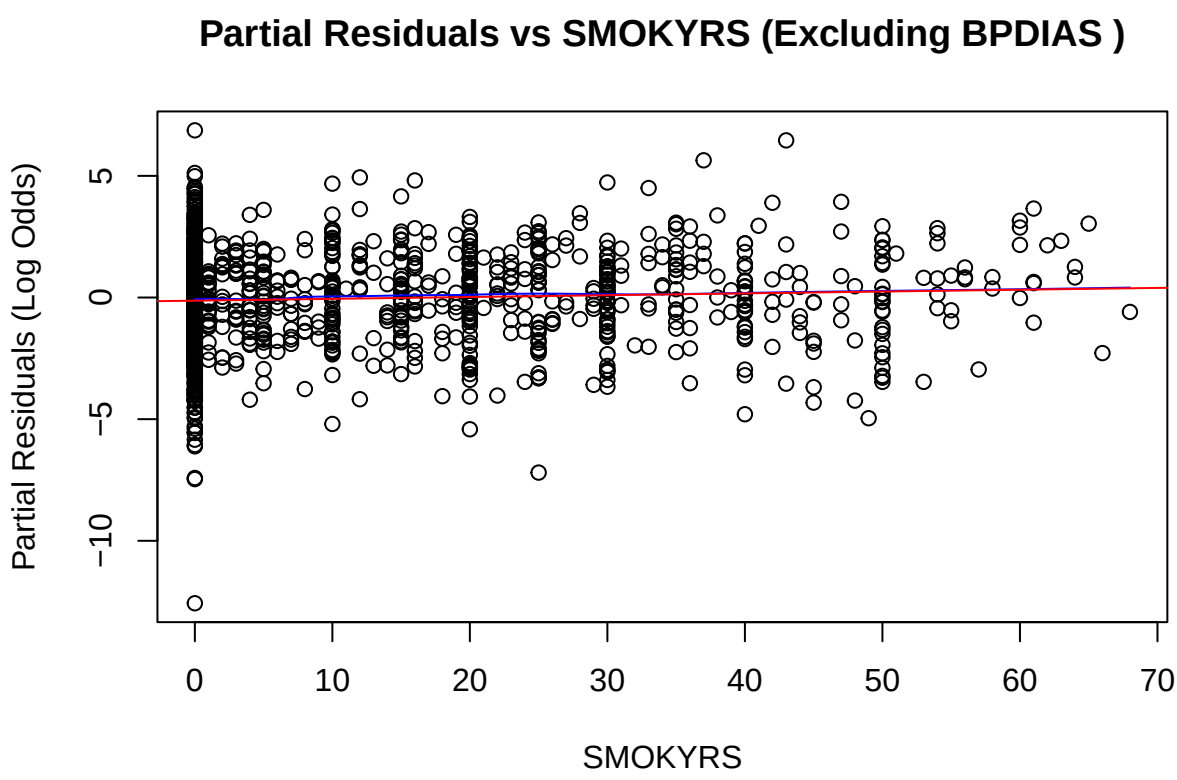

```
## [1] "Processing: NACCAGE"
```



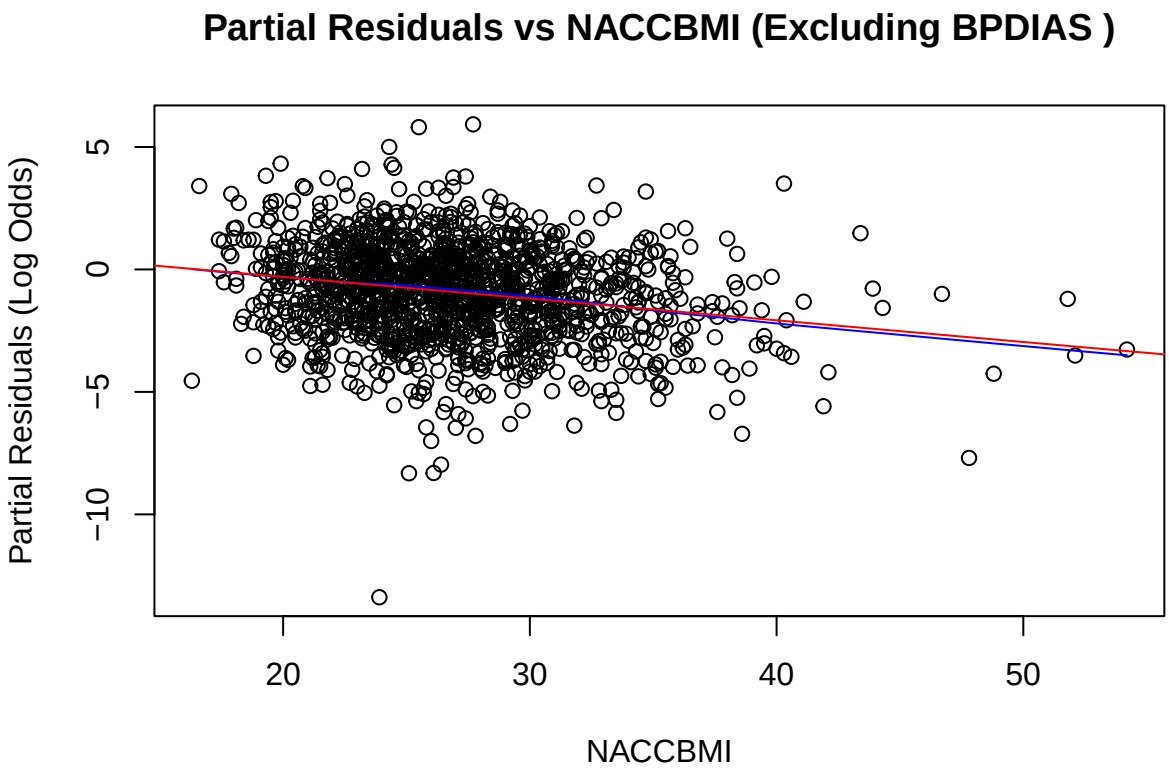

```
## [1] "Processing: EDUC"
```



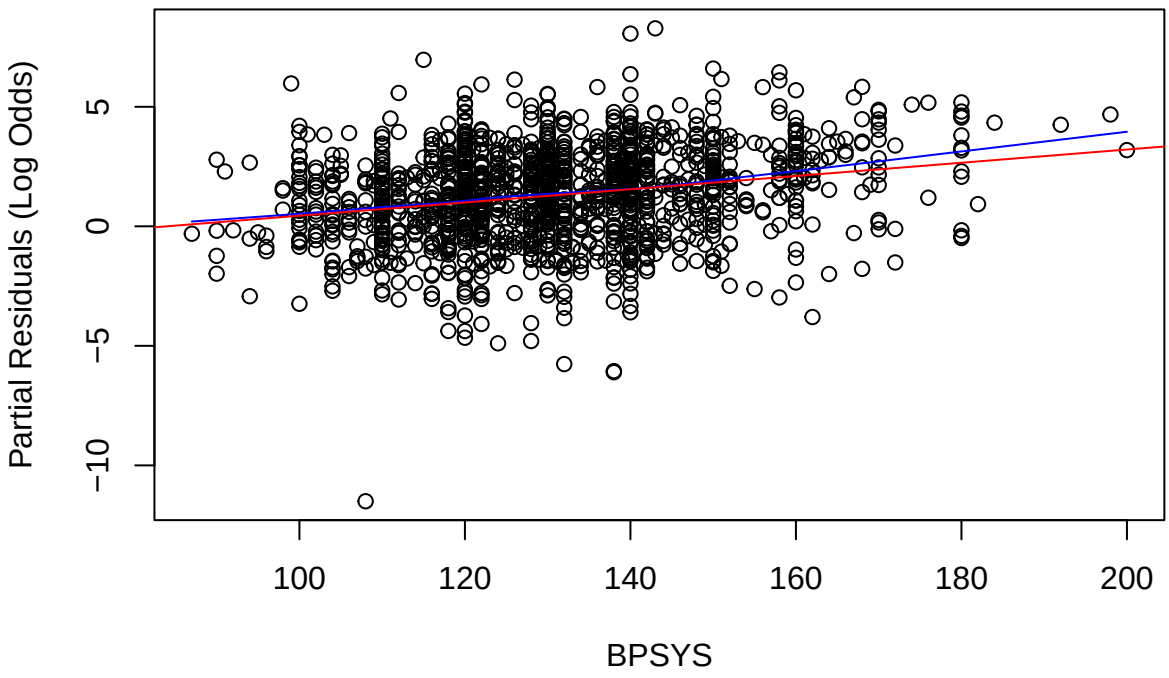

```
## [1] "Processing: SMOKYRS"
```




```
## [1] "Processing: NACCBMI"
```

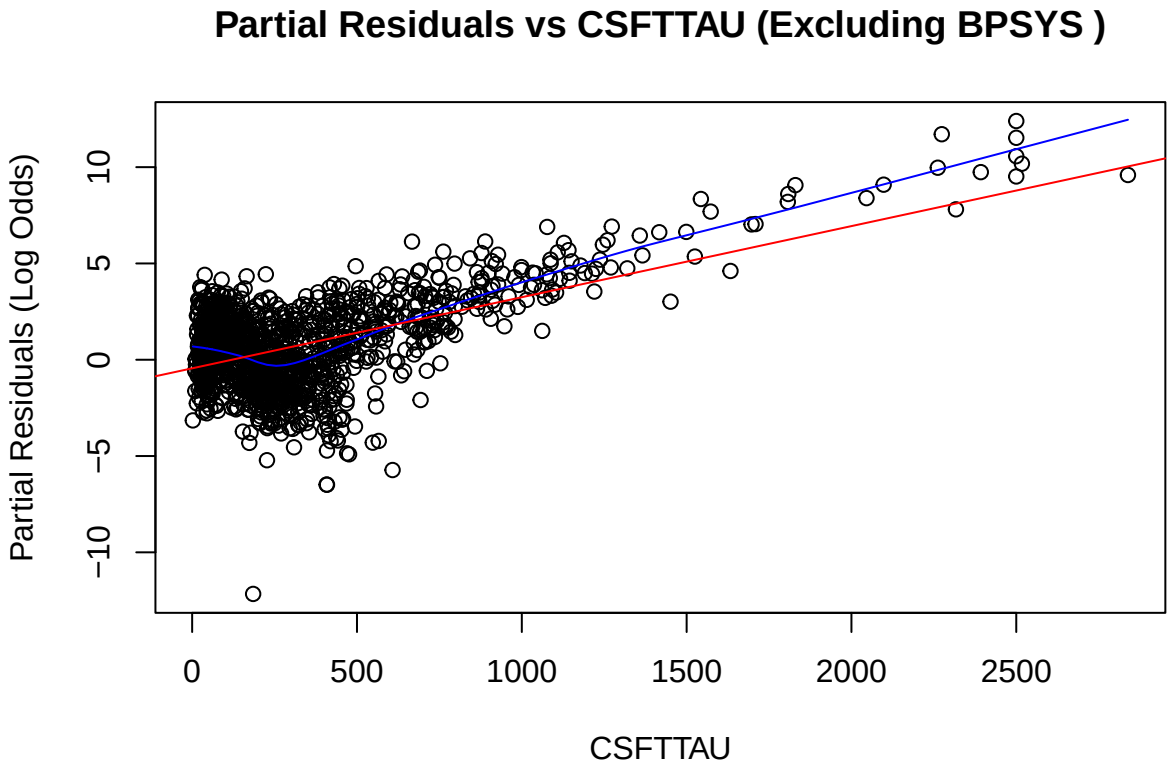



```
## [1] "Processing: BPSYS"
```

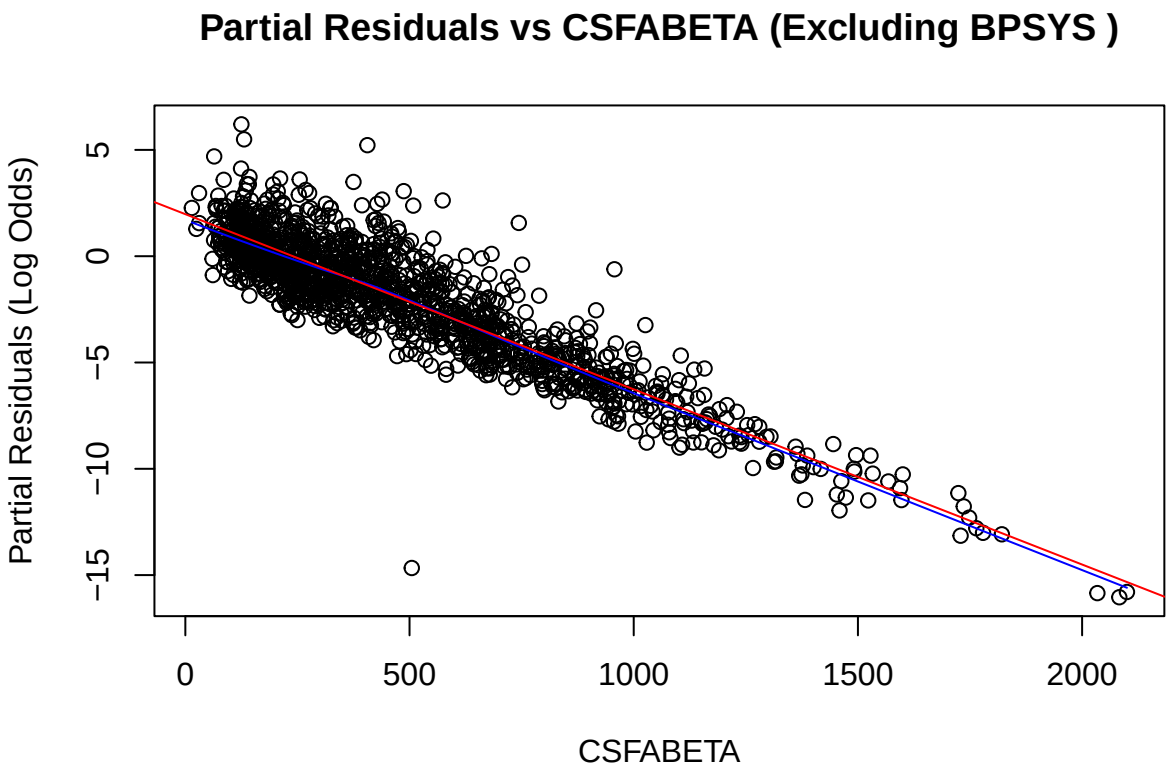



```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

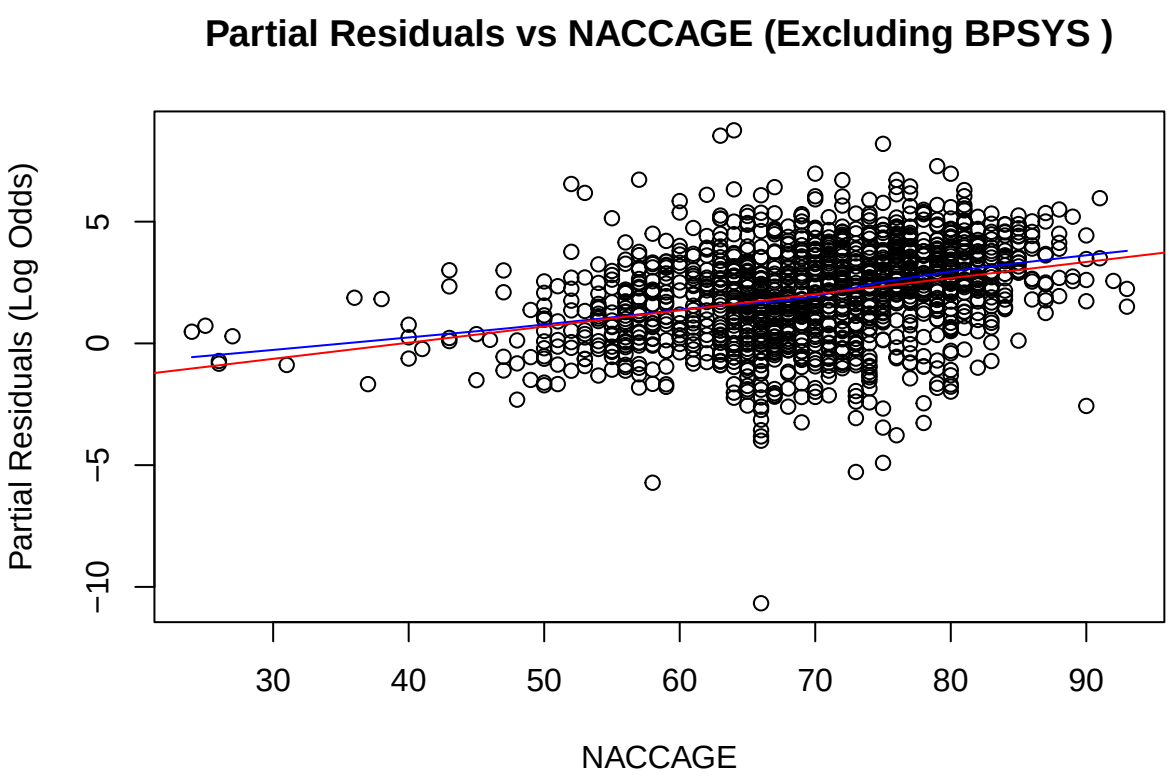
```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
## [1] "Processing: CSFTTAU"
```



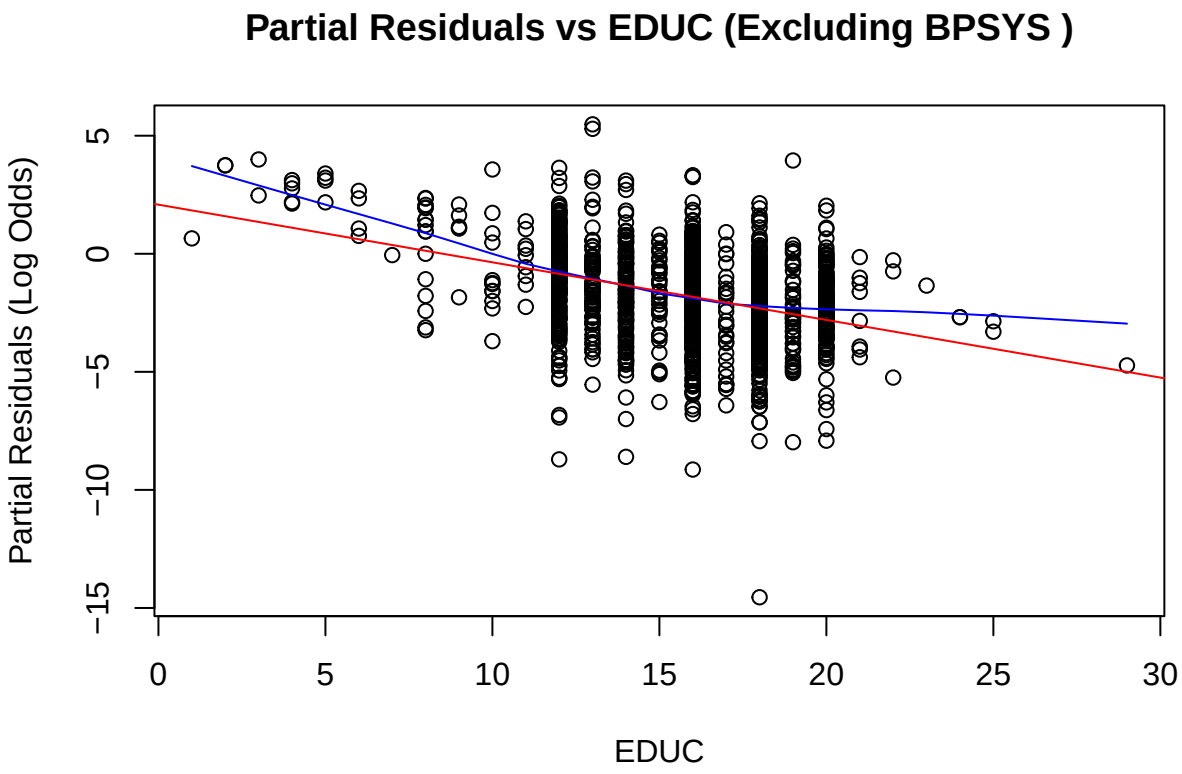

```
## [1] "Processing: CSFABETA"
```



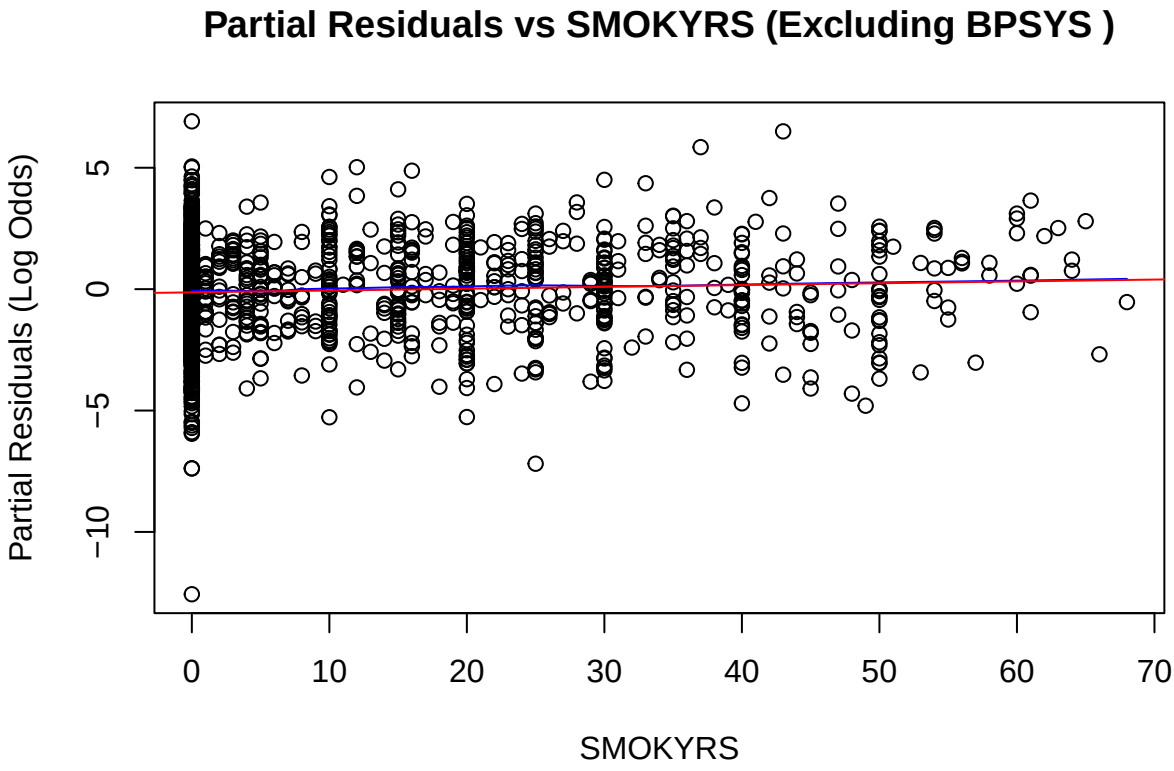

```
## [1] "Processing: NACCAGE"
```



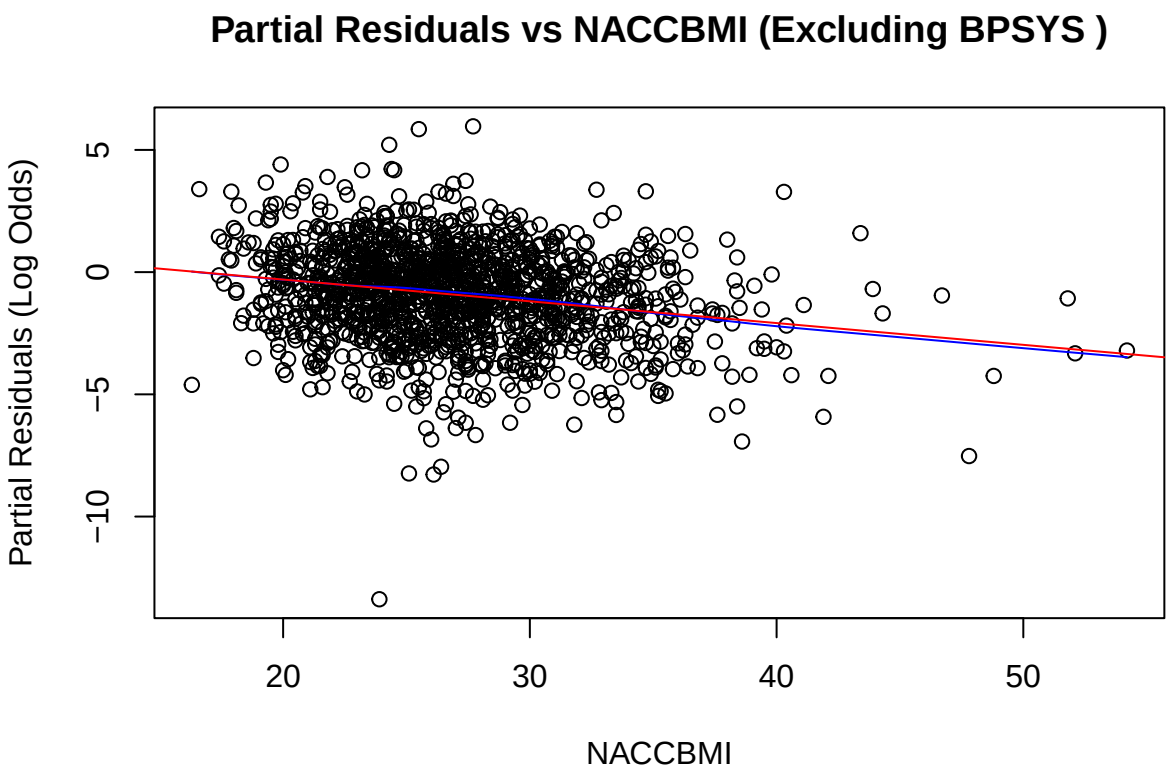

```
## [1] "Processing: EDUC"
```



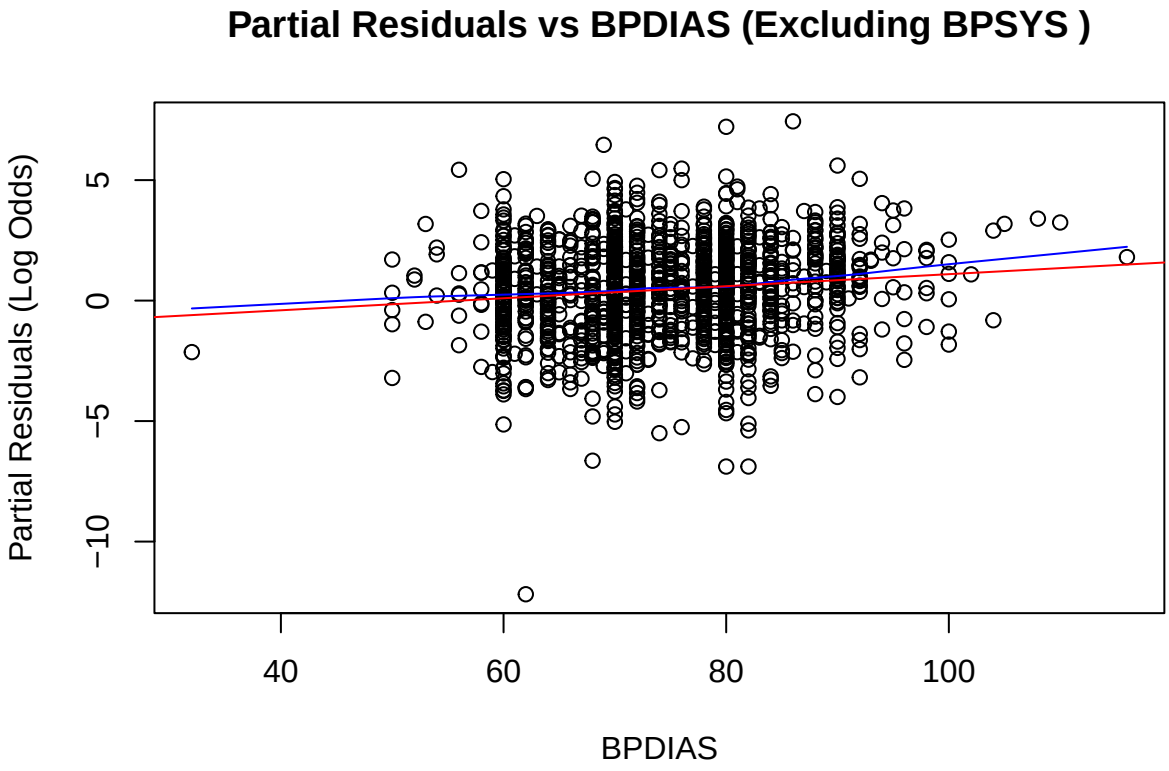

```
## [1] "Processing: SMOKYRS"
```




```
## [1] "Processing: NACCBMI"
```




```
## [1] "Processing: BPDIAS"
```




```
## [1] "Processing: HXHYPER1"
## [1] " HXHYPER1 "
## [1] "Processing: HXSTROKE1"
## [1] " HXSTROKE1 "
## [1] "Processing: DEP2YRS1"
## [1] " DEP2YRS1 "
## [1] "Processing: HISPANIC1"
## [1] " HISPANIC1 "
## [1] "Processing: SEX1"
## [1] " SEX1 "
## [1] "Processing: ALCOHOL1"
## [1] " ALCOHOL1 "
## [1] "Processing: HYPERCHO1"
## [1] " HYPERCHO1 "
## [1] "Processing: CVHATT1"
## [1] " CVHATT1 "
## [1] "Processing: CVCHF1"
## [1] " CVCHF1 "
## [1] "Processing: DIABETES1"
## [1] " DIABETES1 "
## [1] "Processing: NACCNIHR2"
## [1] " NACCNIHR2 "
## [1] "Processing: NACCNIHR4"
## [1] " NACCNIHR4 "
## [1] "Processing: NACCNIHR5"
## [1] " NACCNIHR5 "
```

```
## [1] "Processing: NACCNIHR6"
## [1] " NACCNIHR6 "
## [1] "Processing: B12DEF1"
## [1] " B12DEF1 "
## [1] "Processing: MARISTAT1"
## [1] " MARISTAT1 "
## [1] "Processing: CVAFIB1"
## [1] " CVAFIB1 "
```

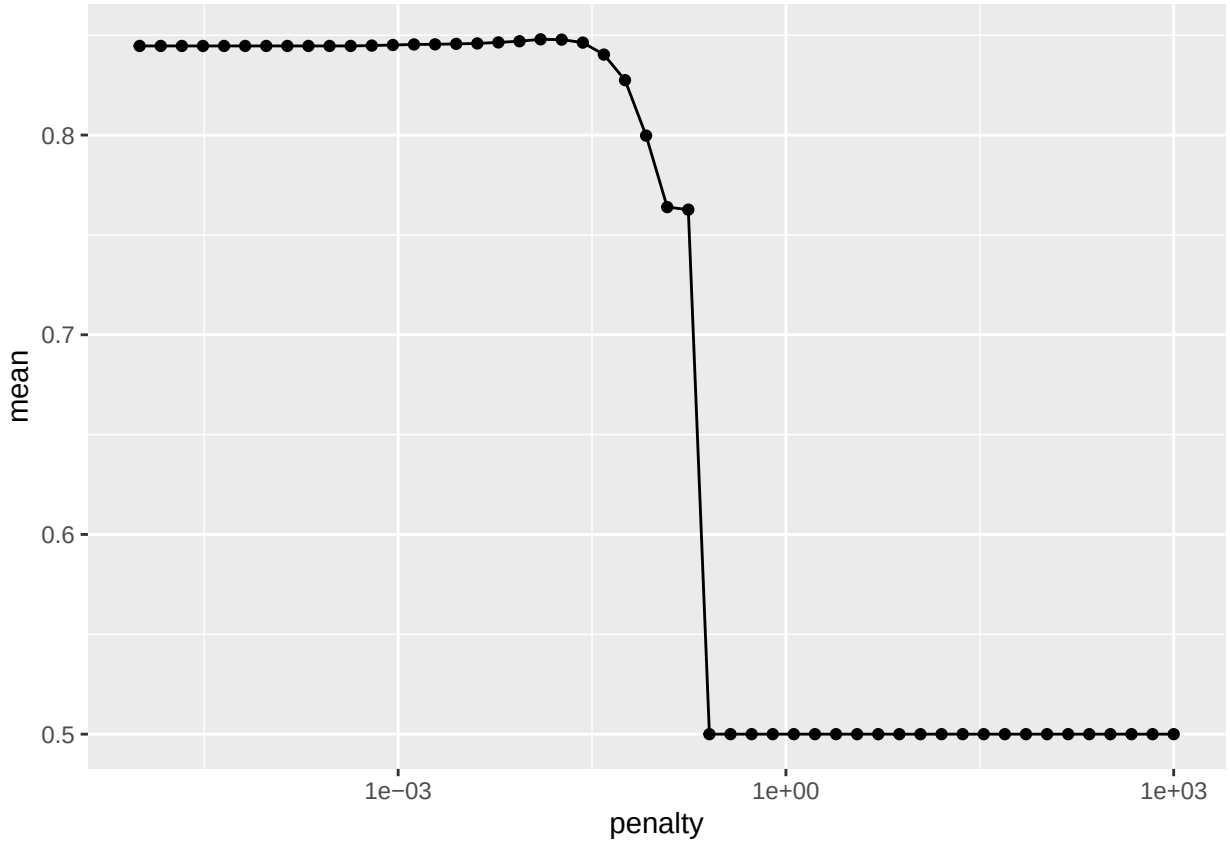
Linearity assumptions seems to pass. However, when I originally did this check, I was not aware of my need to check the linearity of the log-odds (logits) of each variable using partial residuals, so I falsely assumed that the linearity assumptions was violated. So, the following code is under the assumption that linearity was violted. Under that pretense, I employed bootstrapping.

```
df.lasso_model <- logistic_reg(mode = "classification", engine = "glmnet",
                           penalty = tune(), # let's tune the lambda penalty term
                           mixture = 1) #pure lasso regression

lasso_wflow <- workflow() |>
  add_model(df.lasso_model)
```

```
lasso_recipe <- recipe(
  NACCALZD ~ CSFTTAU + CSFABETA + NACCAGE + BPDIAS + BPSYS + HXHYPER + HXSTROKE + NACCBMI + DEP2YRS
  + SMOKYRS + HISPANIC + EDUC + SEX + ALCOHOL + HYPERCHO+ CVHATT + CVCHF
  + DIABETES + NACCNIHR + B12DEF + MARISTAT + CVAFIB, #response ~ predictors
  data = df
) |>
  step_normalize(all_numeric_predictors()) |>
  step_dummy(all_nominal_predictors())

lasso_wflow <- lasso_wflow |>
  add_recipe(lasso_recipe)
```

```
set.seed(1332)
df_cv <- vfold_cv(df, v = 10)

# auto.cv_grid <- expand.grid(penalty = ) dont think i need this...

lasso_tune1 <- tune_grid(df.lasso_model,
                     lasso_recipe,
                     resamples = df_cv,
                     grid = grid_regular(penalty(range = c(-5, 3)), levels = 50))
```

```
lasso_tune1 |>
  collect_metrics() |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = penalty, y = mean)) + geom_point() + geom_line() + scale_x_log10()
```




```
lasso_best <- lasso_tune1 |>
  select_by_one_std_err(
    metric = "roc_auc",
    desc(penalty) # order penalty from largest (highest bias = simplest model) to smallest
)
lasso_best
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1  0.0391 Preprocessor1_Model23
```

```
lasso_wflow_final <- lasso_wflow |>
  finalize_workflow(parameters = lasso_best)
```

```
lasso_fit <- lasso_wflow_final |>
  fit(data = df)
lasso_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
```

```
## 2 Recipe Steps
##
## * step_normalize()
## * step_dummy()
##
## -- Model ---------------------------------------------------------------------
##
## Call:  glmnet::glmnet(x = maybe_matrix(x), y = y, family = "binomial",      alpha = ~1)
##
##    Df  %Dev   Lambda
## 1   0  0.00 0.210900
## 2   1  2.18 0.192100
## 3   1  4.01 0.175100
## 4   1  5.56 0.159500
## 5   1  6.88 0.145300
## 6   1  8.01 0.132400
## 7   1  8.98 0.120700
## 8   2 10.36 0.110000
## 9   2 11.58 0.100200
## 10  3 12.81 0.091280
## 11  3 14.38 0.083170
## 12  3 15.75 0.075780
## 13  4 16.98 0.069050
## 14  5 18.28 0.062920
## 15  5 19.57 0.057330
## 16  6 20.80 0.052240
## 17  6 21.92 0.047590
## 18  6 22.89 0.043370
## 19  7 23.80 0.039510
## 20  8 24.62 0.036000
## 21  8 25.36 0.032810
## 22  8 25.99 0.029890
## 23 10 26.56 0.027240
## 24 10 27.09 0.024820
## 25 10 27.54 0.022610
## 26 11 27.93 0.020600
## 27 11 28.28 0.018770
## 28 12 28.58 0.017100
## 29 12 28.87 0.015590
## 30 12 29.11 0.014200
## 31 13 29.33 0.012940
## 32 13 29.52 0.011790
## 33 14 29.70 0.010740
## 34 15 29.85 0.009788
## 35 17 29.99 0.008918
## 36 18 30.11 0.008126
## 37 19 30.21 0.007404
## 38 20 30.31 0.006746
## 39 20 30.39 0.006147
## 40 20 30.46 0.005601
## 41 21 30.52 0.005103
## 42 21 30.58 0.004650
## 43 21 30.62 0.004237
## 44 23 30.66 0.003861
```

```
## 45 23 30.69 0.003518
## 46 23 30.72 0.003205
##
## ...
## and 21 more lines.
```

```
lasso_coef <- lasso_fit |>
  broom::tidy()
lasso_coef
```

```
## # A tibble: 26 x 3
##    term         estimate penalty
##    <chr>           <dbl>   <dbl>
##  1 (Intercept) -0.115     0.0391
##  2 CSFTTAU      0.325     0.0391
##  3 CSFABETA    -0.885     0.0391
##  4 NACCAGE      0.123     0.0391
##  5 BPDIAS       0         0.0391
##  6 BPSYS        0.000895  0.0391
##  7 NACCBMI      0         0.0391
##  8 SMOKYRS      0         0.0391
##  9 EDUC        -0.0808    0.0391
## 10 HXHYPER_X1   0         0.0391
## # i 16 more rows
```

```
# Check the first few rows and column names of df to confirm structure
print(names(df))
```

```
##  [1] "CSFTTAU"  "CSFABETA" "NACCAGE"  "BPDIAS"   "BPSYS"    "NACCBMI"
##  [7] "SMOKYRS"  "EDUC"     "HXHYPER"  "HYPERCHO" "HXSTROKE" "CVHATT"
## [13] "CVCHF"    "CVAFIB"   "DIABETES" "B12DEF"   "DEP2YRS"  "ALCOHOL"
## [19] "NACCNIHR" "HISPANIC" "SEX"      "MARISTAT" "NACCALZD"
```

```
print(head(df))
```

```
## # A tibble: 6 x 23
##   CSFTTAU CSFABETA NACCAGE BPDIAS BPSYS NACCBMI SMOKYRS  EDUC HXHYPER HYPERCHO
##     <dbl>    <dbl>   <int>  <int> <int>   <dbl>   <int> <int> <fct>   <fct>
## 1   135       177       72     72   123    29        10    14 0       1
## 2    54       461       66     70   132    30        10    20 1       1
## 3    42.2     301.      84     82   168    24.3      27    16 1       1
## 4    28.5     198.      75     86   128    28        47    12 0       1
## 5    25       189       79     90   150    25.6      61    20 0       0
## 6   186.      139.      65     78   132    26.9       2    20 0       1
## # i 13 more variables: HXSTROKE <fct>, CVHATT <fct>, CVCHF <fct>, CVAFIB <fct>,
## #   DIABETES <fct>, B12DEF <fct>, DEP2YRS <fct>, ALCOHOL <fct>, NACCNIHR <fct>,
## #   HISPANIC <fct>, SEX <fct>, MARISTAT <fct>, NACCALZD <fct>
```

```
library(tidymodels)
library(dplyr)
library(purrr)
```

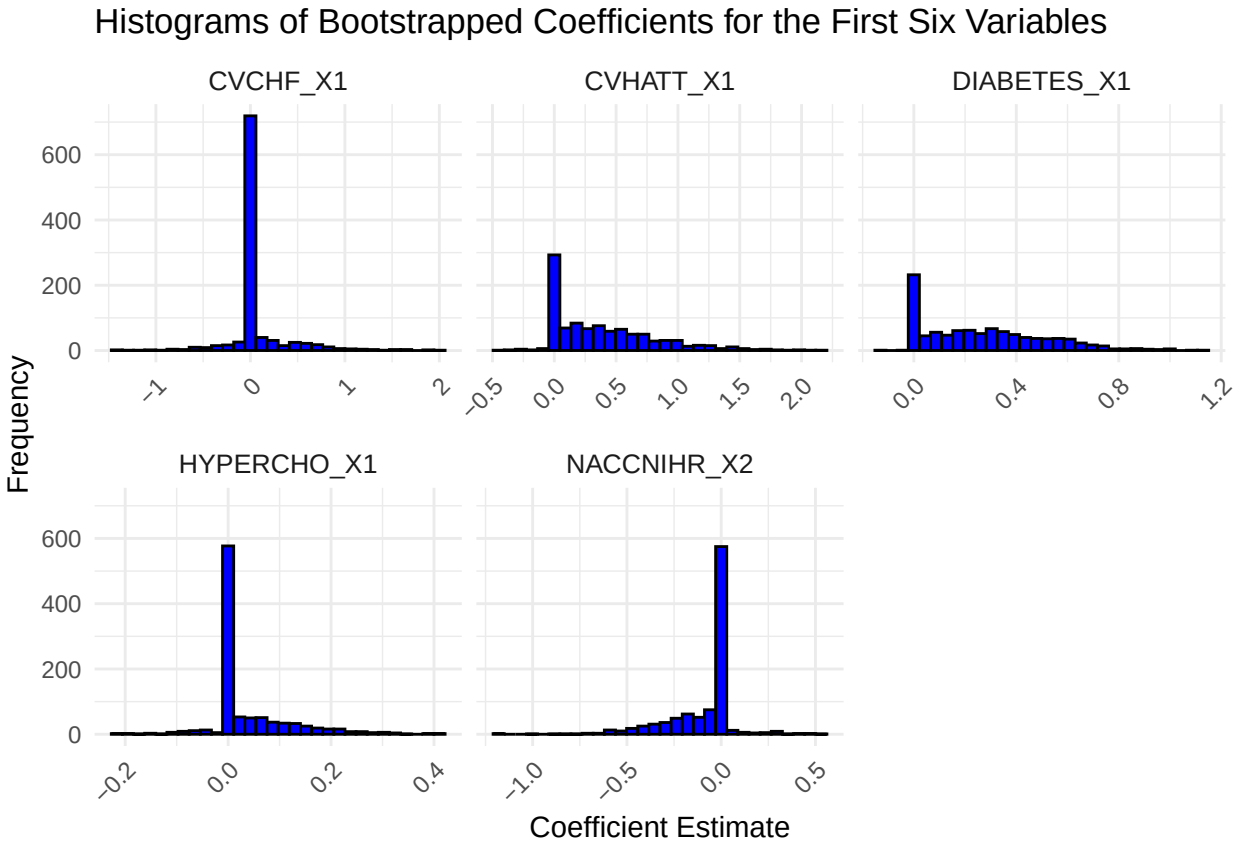
```r
library(broom)
library(glmnet)

time_result <- system.time({

# Set up LASSO model and workflow
df.lasso_model <- logistic_reg(mode = "classification", engine = "glmnet",
                           penalty = tune(), mixture = 1) # Pure LASSO

lasso_recipe <- recipe(
  NACCALZD ~ CSFTTAU + CSFABETA + NACCAGE + BPDIAS + BPSYS + HXHYPER + HXSTROKE + NACCBMI + DEP2YRS
  + SMOKYRS + HISPANIC + EDUC + SEX + ALCOHOL + HYPERCHO + CVHATT + CVCHF
  + DIABETES + NACCNIHR + B12DEF + MARISTAT + CVAFIB,
  data = df
) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

lasso_wflow <- workflow() %>%
  add_model(df.lasso_model) %>%
  add_recipe(lasso_recipe)

# Create 1000 bootstrap samples
set.seed(1332)
boot_samples <- bootstraps(df, times = 1000)

# Define the lambda grid for tuning (adjust if necessary)
lambda_grid <- grid_regular(penalty(range = c(-5, 3)), levels = 50)

# Function to fit and tune model on each sample
fit_model <- function(boot) {
  # Fit the model on the resampled data
  lasso_tuned <- tune_grid(
    lasso_wflow,
    resamples = bootstraps(analysis(boot), times = 1), # Single resample
    grid = lambda_grid
  ) %>%
    select_best(metric = "roc_auc")

  # Fit the final model using the selected best lambda
  final_fit <- finalize_workflow(
    lasso_wflow,
    parameters = lasso_tuned
  ) %>%
    fit(data = analysis(boot))

  # Extract coefficients using the best lambda
  tidy(final_fit, effects = "fixed", parameters = lasso_tuned)
}

# Apply the function to each bootstrap sample
boot_results <- map_df(boot_samples$splits, fit_model, .id = "bootstrap_id")
})
```

```r
print(time_result)
```

```
##    user  system elapsed
## 223.385   3.095 228.158
```

check shape of bootstrap distribution of coefficients to determine the best confidence interval to use

```r
library(ggplot2)
library(dplyr)

# Assuming boot_results has the structure with 'term' and 'estimate' columns

# Identify the first six unique terms/variables in the data frame
first_six_terms <- unique(boot_results$term)[16:20]

# Filter the data for only the first six variables and plot the histograms
boot_results %>%
  filter(term %in% first_six_terms) %>%
  ggplot(aes(x = estimate)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  facet_wrap(~ term, scales = "free_x") + # Creates a separate histogram for each variable
  labs(title = "Histograms of Bootstrapped Coefficients for the First Six Variables",
       x = "Coefficient Estimate",
       y = "Frequency") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Enhances readability of x-axis labels
        strip.text.x = element_text(size = 10))  # Adjust size for readability of facet labels
```



#Becasue some of these show some skewness, we will use normal theory confidence intervals

```
# Calculate the mean, standard error, and normal theory confidence intervals of the coefficients
coef_summary <- boot_results %>%
  group_by(term) %>%
  summarise(
    mean = mean(estimate),
    std_error = sd(estimate), # Calculate the standard error
    lower_ci = mean - qt(0.975, df = n() - 1) * std_error, # Calculate the lower bound of the CI
    upper_ci = mean + qt(0.975, df = n() - 1) * std_error  # Calculate the upper bound of the CI
  )
coef_summary
```

```
## # A tibble: 26 x 5
##    term           mean std_error lower_ci upper_ci
##    <chr>         <dbl>     <dbl>    <dbl>    <dbl>
##  1 (Intercept) -0.409     0.213   -0.827   0.00871
##  2 ALCOHOL_X1   0.0678    0.208   -0.340   0.476
##  3 B12DEF_X1   -0.222     0.319   -0.847   0.403
##  4 BPDIAS       0.0261    0.0523  -0.0766  0.129
##  5 BPSYS        0.110     0.0831  -0.0527  0.273
##  6 CSFABETA    -1.21      0.186   -1.58   -0.849
##  7 CSFTTAU      0.586     0.154    0.283   0.889
##  8 CVAFIB_X1    0.125     0.251   -0.368   0.617
##  9 CVCHF_X1     0.0635    0.320   -0.564   0.691
## 10 CVHATT_X1    0.386     0.419   -0.437   1.21
## # i 16 more rows
```

#Build a neural network using the varables that have the most predictive power