Assignment 3 Solution

Ted Ladas

Question 1

1a.

After a first exploration of the dataset we conclude that the percentage of missing cases on the `nhanes` dataset is `48%`

1b.

The proportions of variance due to the missing data for each parameter are given by the lambda term. Specifically:

`age: 0.6864` `hyp: 0.3504` `chl: 0.3041`

Therefore the `age` parameter seem to be most affected by the nonresponse.

```
# m = 5 (default value)
bmi_hat = pool(with(mice(nhanes, seed=1, printFlag=FALSE), lm(bmi ~ age + hyp + chl)))
bmi_hat[,3][c(1,3,10)]
```

```
##          term    estimate     lambda
## 1 (Intercept) 19.61789252 0.08938989
## 2         age -3.55287155 0.68640637
## 3         hyp  2.19701748 0.35043452
## 4         chl  0.05378081 0.30408063
```

```
# summary(bmi_hat)
```

1c.

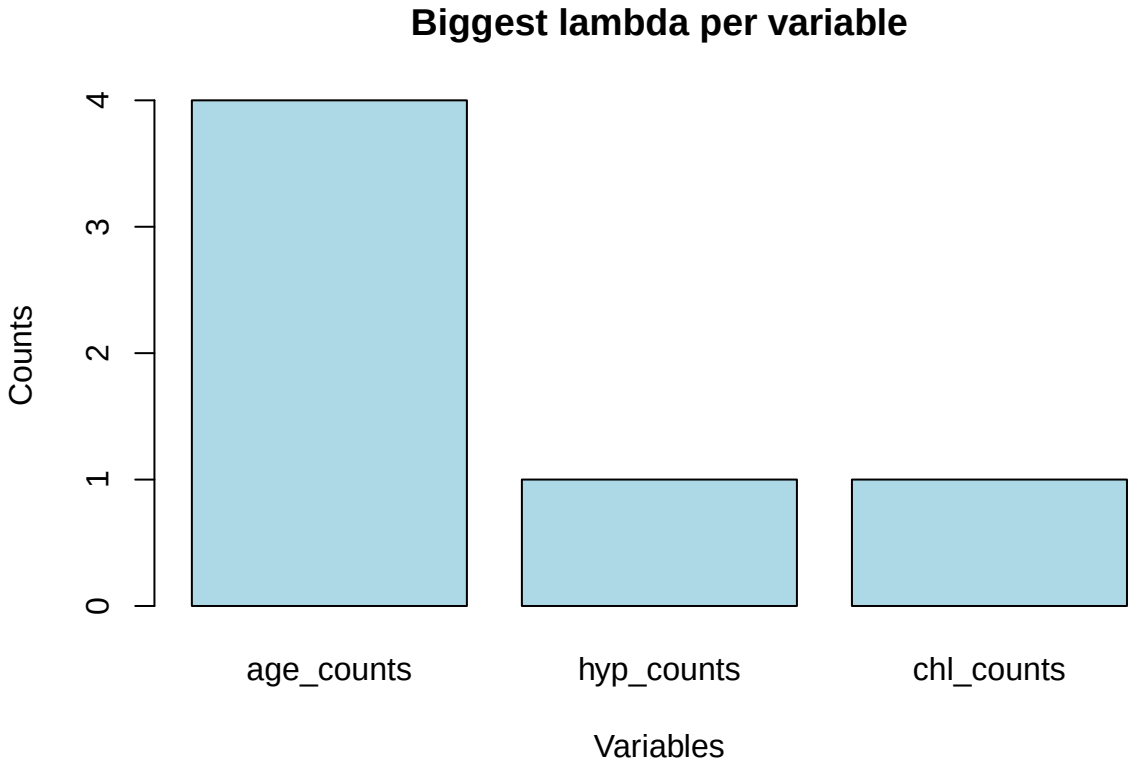
The script below tries to automate and visualize the process in order to be able to potentially scale it. For the six iteration of random seeds and the tree variables, we notice that on seeds `1`, `2`, `3` and `6` the `age` variable is the one with the biggest `lambda` parameter. On `seed = 4` the `chl` variable has the biggest `lambda` and on `seed = 5` it's the `hyp` parameter. On my tests up to `1e2` different seeds (not displayed here), the distribution was: `age: 46%`, `hyp: 26%`, `chl: 28%` So finally, the results are dependent on the seed.

```
analyze <- function(data, seeds, models=5){
  num_of_seeds = seeds
  dataset = data
  m = models
  # Initialize a dataframe to store results
```

```
  df <- data.frame(age = rep(0L, num_of_seeds),
                   hyp = rep(0L, num_of_seeds),
                   chl = rep(0L, num_of_seeds),
                   age_lambda = rep(0L, num_of_seeds),
                   hyp_lambda = rep(0L, num_of_seeds),
                   chl_lambda = rep(0L, num_of_seeds))
  max_lambda <- c(1:num_of_seeds)*0

  for (i in c(1:num_of_seeds)){
    bmi_hat = pool(with(mice(dataset, seed=i, printFlag=FALSE, m=m), lm(bmi ~ age + hyp + chl)))
    df[i,1:3] <- bmi_hat[2:4,3][1:3,3] # store estimates
    df[i,4:6] <- bmi_hat[2:4,3][1:3,10] # store lambda values

    # Not the cleanest code
    # Logic loop in order to find the maximum per row of the above dataframe of the lambda parameters s

    max_lambda[i] <- which.max(t(df[4:6])[,i])
    if (max_lambda[i]==1){
      max_lambda[i] <- 'age'
    }
    else if (max_lambda[i]==2){
      max_lambda[i] <- 'hyp'
    }
    else {
      max_lambda[i] <- 'chl'
    }
  }

  # Counts per variable.
  # Shows on num_of_seeds iterations which variable had the biggest lambda score and adds them up.
  counts <- data.frame(age_counts = length(which(max_lambda == 'age')),
                       hyp_counts = length(which(max_lambda == 'hyp')),
                       chl_counts = length(which(max_lambda == 'chl')))

  print(df)
  cat('\n')
  print(max_lambda)
  cat('\n')
  print(counts)

  barplot(as.matrix(counts),
          col='lightblue',
          main = 'Biggest lambda per variable',
          xlab = 'Variables',
          ylab = 'Counts')
}

analyze(data=nhanes, seeds=6)
```

```
##         age      hyp        chl age_lambda hyp_lambda chl_lambda
## 1 -3.552872 2.197017 0.05378081  0.6864064  0.3504345  0.3040806
## 2 -4.061509 1.530476 0.06283490  0.4033924  0.1430995  0.2959966
## 3 -3.857533 1.352812 0.05872834  0.5895051  0.4101152  0.5621346
```

```
## 4 -3.506034 2.750530 0.04920611  0.2189333  0.1961083  0.3305334
## 5 -3.496722 1.509548 0.06081272  0.4511896  0.5942866  0.2346065
## 6 -2.921414 1.224746 0.04949218  0.6549523  0.2960364  0.5196295
##
## [1] "age" "age" "age" "chl" "hyp" "age"
##
##   age_counts hyp_counts chl_counts
## 1          4          1          1
```



1d.

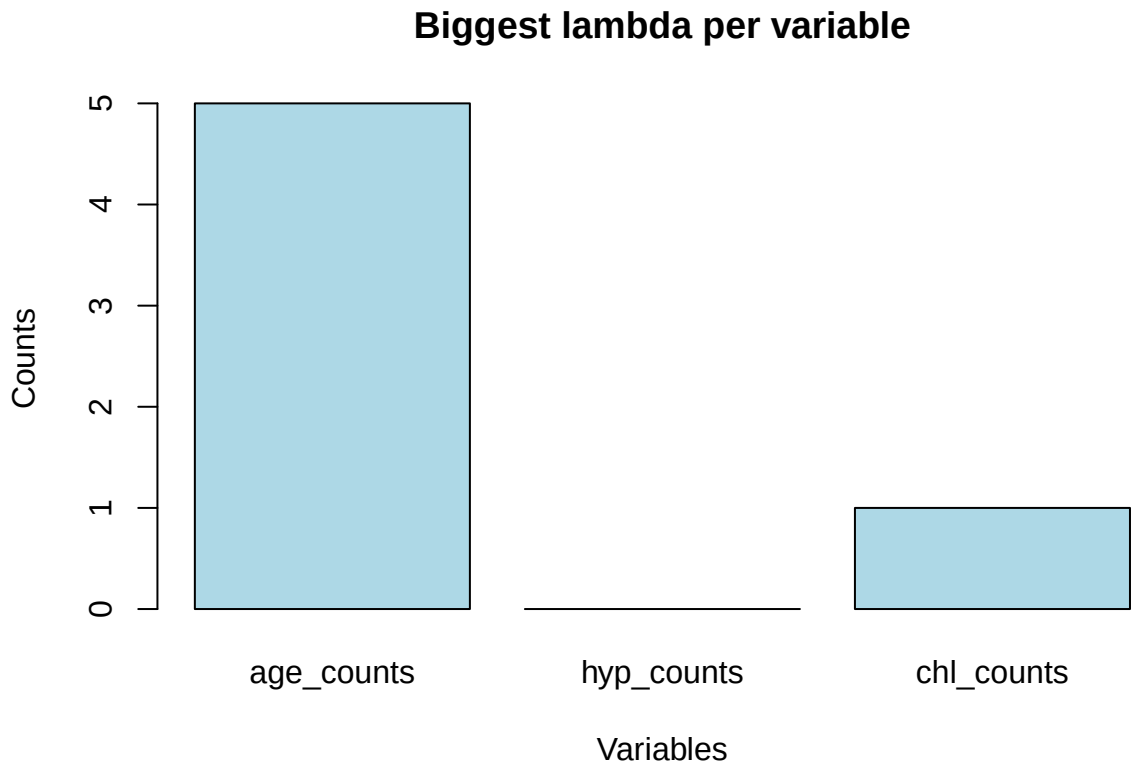
With the above work on the function, now we can just call the function `analyze` with the extra argument `models=10` and we have our results so we can focus on the interpretation.

For `M=100` we see exactly what we expected, which is the reduction of the variability since `age` is most effected by the nonresponse. However, even for a small number of seeds there is a noticeable increase in time until execution of the models, since we are now calculating 3 coefficients for 6 seeds and 100 models. The overall $\mathcal{O}$ is $\mathcal{O}(CSM)$ where `C: coefficients`, `S: number of seeds`,`M: number of models`, which is expensive. Our dataset is small so the calculations of all the coefficients don't take that much time, but in order to calculate the coefficients which are given from: $b = (X^T X)^{-1} X^T Y$ we need $\mathcal{O}(N^3)$ where `N: number or rows in dataset X` which can get be very expensive very quickly.

When comparing between the model with `M=5` and `M=100` for this specific dataset, I would prefer the `M=100`.

```
analyze(data=nhanes, seeds=6, models=100)
```

```
##         age      hyp        chl age_lambda hyp_lambda chl_lambda
## 1 -3.641255 1.685794 0.05449706  0.4324680  0.2915346  0.3217837
## 2 -3.633789 1.719915 0.05352120  0.4031077  0.2825108  0.2939693
## 3 -3.557061 1.557562 0.05445409  0.3093072  0.2425105  0.3281911
## 4 -3.648153 1.659470 0.05511595  0.3943223  0.2565132  0.2835232
## 5 -3.762972 1.802832 0.05534382  0.3322570  0.2893046  0.2461956
## 6 -3.593092 1.862196 0.05319323  0.4430300  0.2860700  0.3113085
##
## [1] "age" "age" "chl" "age" "age" "age"
##
##   age_counts hyp_counts chl_counts
## 1          5          0          1
```



Question 2

```
load('dataex2.Rdata')
```

Question 3

Question 4

Question 5

We want to find how weight is affected by gender, height and waist circumference from the NHANES2 dataset.

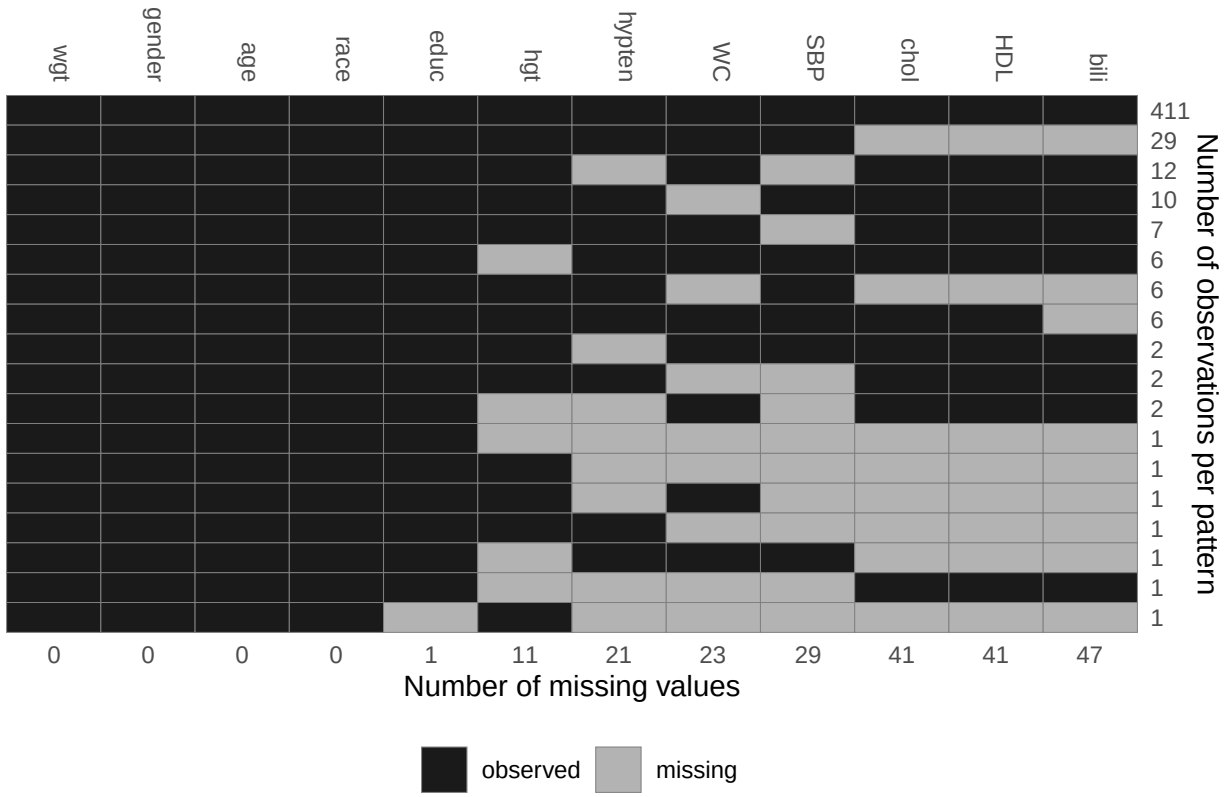
$\texttt{wht} = \beta_0 + \beta_1 \texttt{gender} + \beta_2 \texttt{age} + \beta_3 \texttt{hgt} + \beta_4 \texttt{WC} + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$

So at first we inspeact the data.

```
load('NHANES2.Rdata')
nhanes2 <- NHANES2
summary(nhanes2)
```

```
##       wgt            gender         bili             age             chol
##  Min.   : 39.01   male  :252   Min.   :0.2000   Min.   :20.00   Min.   : 2.07
##  1st Qu.: 65.20   female:248   1st Qu.:0.6000   1st Qu.:31.00   1st Qu.: 4.27
##  Median : 76.20                Median :0.7000   Median :43.00   Median : 4.86
##  Mean   : 78.25                Mean   :0.7404   Mean   :45.02   Mean   : 5.00
##  3rd Qu.: 86.41                3rd Qu.:0.9000   3rd Qu.:58.00   3rd Qu.: 5.64
##  Max.   :167.38                Max.   :2.9000   Max.   :79.00   Max.   :10.68
##                                NA's   :47                       NA's   :41
##       HDL             hgt                          educ
##  Min.   :0.360   Min.   :1.397   Less than 9th grade : 31
##  1st Qu.:1.110   1st Qu.:1.626   9-11th grade        : 69
##  Median :1.320   Median :1.676   High school graduate:115
##  Mean   :1.395   Mean   :1.687   some college        :148
##  3rd Qu.:1.590   3rd Qu.:1.753   College or above    :136
##  Max.   :3.130   Max.   :1.930   NA's                :  1
##  NA's   :41      NA's   :11
##                  race          SBP           hypten          WC
##  Mexican American  : 52   Min.   : 81.33   no  :354   Min.   : 61.90
##  Other Hispanic    : 58   1st Qu.:109.00   yes :125   1st Qu.: 84.80
##  Non-Hispanic White:182   Median :118.67   NA's: 21   Median : 95.00
##  Non-Hispanic Black:112   Mean   :120.05              Mean   : 96.07
##  other             : 96   3rd Qu.:128.67              3rd Qu.:104.80
##                           Max.   :202.00              Max.   :154.70
##                           NA's   :29                  NA's   :23
```

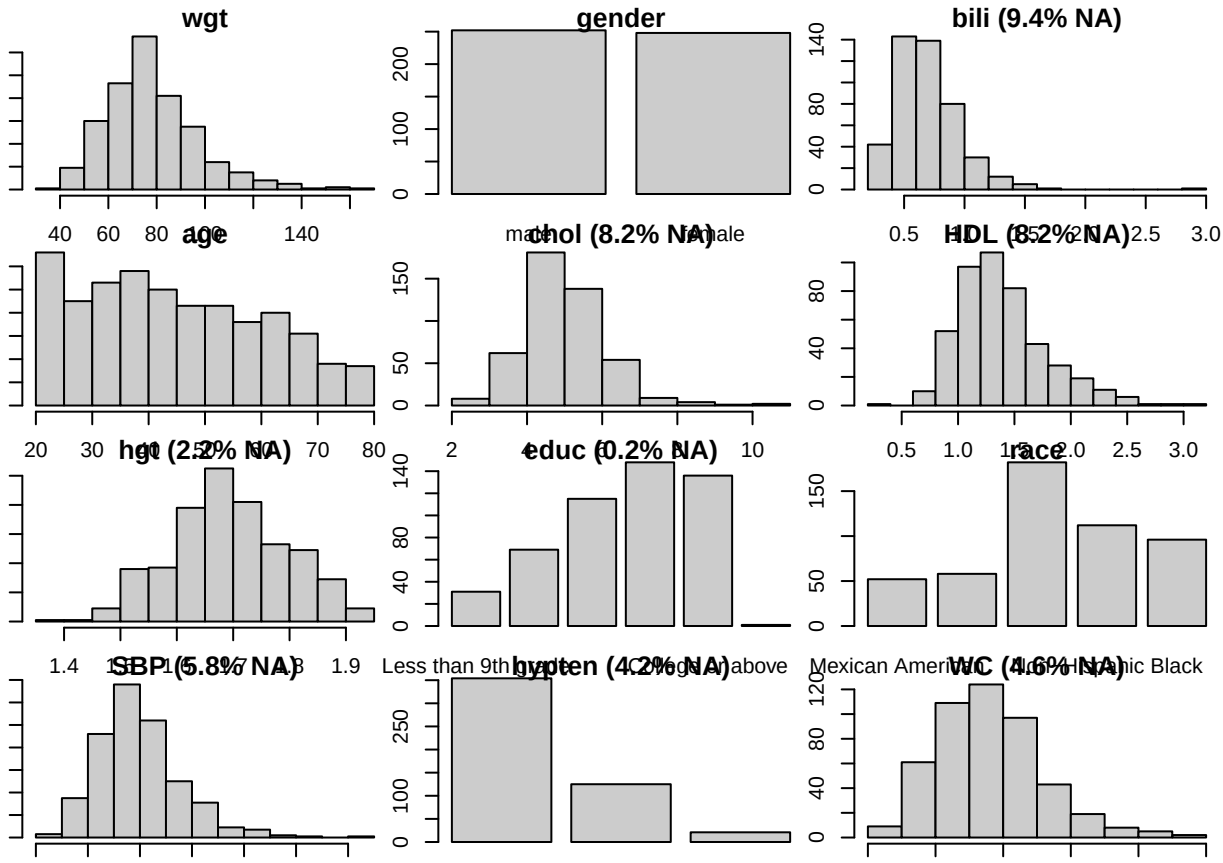
```
md_pattern(nhanes2, pattern = FALSE)
```



We quickly realize that we have a lot of missing values on the important variables so we need to impute them.

Firstly we will try to visualize the distributions of all the variables in order to get a first feeling for our dataset. We can extract some meaning full information form the graphs, such as `age` looks quite Uniform(20,80), maybe with a heave decreasing tail, `hgt` can be approximated by Normal around `1.7` `wgt` and `SBP` look like they follow some kind of shifted Beta distribution etc.

```
par(mar=c(1,1,1,1))
plot_all(nhanes2)
```



Before starting with imposing missing values, we need to inspect what methods are automatically attributed to the missing data values. We notice that `mice` has chosen `pmm` for the height variable while from our graphs we see that we could change that for a Normal.

```
imp0 <- mice(nhanes2, maxit=0)
imp0
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##      wgt   gender     bili      age     chol      HDL      hgt     educ
##       ""       ""    "pmm"       ""    "pmm"    "pmm"    "pmm"   "polr"
##     race      SBP   hypten       WC
##       ""    "pmm" "logreg"    "pmm"
## PredictorMatrix:
##        wgt gender bili age chol HDL hgt educ race SBP hypten WC
## wgt      0      1    1   1    1   1   1    1    1   1      1  1
## gender   1      0    1   1    1   1   1    1    1   1      1  1
## bili     1      1    0   1    1   1   1    1    1   1      1  1
## age      1      1    1   0    1   1   1    1    1   1      1  1
## chol     1      1    1   1    0   1   1    1    1   1      1  1
## HDL      1      1    1   1    1   0   1    1    1   1      1  1
```

Which is what we do with the following chunk of code.

```
methods <- imp0$method
methods['hgt'] <- 'norm'
methods
```

```
##      wgt   gender     bili      age     chol      HDL      hgt     educ
##       ""       ""    "pmm"       ""    "pmm"    "pmm"   "norm"   "polr"
##     race      SBP   hypten       WC
##       ""    "pmm" "logreg"    "pmm"
```

Next step to our analysis, is to set the minimum and the maximum for all the numeric variables that we will try to imputing On this specific example, I tried to automate the procedure by finding the min and the max for all the variables and then setting those values to be the barriers for each variable.

```
min_bili <- min(nhanes2$bili)
max_bili <- max(nhanes2$bili)
min_chol <- min(nhanes2$chol)
max_chol <- max(nhanes2$chol)
min_HDL <- min(nhanes2$HDL)
max_HDL <- max(nhanes2$HDL)
min_hgt <- min(nhanes2$hgt)
max_hgt <- max(nhanes2$hgt)
min_SBP <- min(nhanes2$SBP)
max_SBP <- max(nhanes2$SBP)
min_WC <- min(nhanes2$WC)
max_WC <- max(nhanes2$WC)

post <- imp0$post
post['bili'] <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_bili, max_bili)'
post['chol'] <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_chol, max_chol)'
post['HDL']  <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_HDL , max_HDL )'
post['hgt']  <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_hgt , max_hgt )'
post['SBP']  <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_SBP , max_SBP )'
post['WC']   <- 'imp[[j]][,i] <- sqeeze(imp[[j]][,i], c(min_WC  , max_WC  )'
```

Now our analysis can begin. On this part, we will try to impute the missing values by using 30 copies of the dataset and running them for 20 steps. This is a critical point in our analysis because in order to go further and still trust our result we need to somehow convene ourselves that our chains have converged. Unfortunately we know that we cannot *prove* convergence, we can only examine various plots and graphs in order to stop any hard patterns or trends. In this example we are going to examine the trace and histogram plots.

On this point I would like to spend a little time on the hyperparameters `maxit` and `m`, as well as the `seed`. A good way to set these hyperparameters would be through a grid search method, where we would loop through `maxit` $\in (10, 100)$, `m`$\in (10, 100)$ and repeat the process for some random seeds in order to exclude the possibility of the random seed affecting the results. On each iteration we would store for each chain the Brooks-Gelman-Rubin statistic, which measures the proportion of between to within variability. Then after the grid search we would average on all `seeds` the `BGR` statistic and finally we would pick the one where it's closest to one. This was out of the scope of the exercise but I wanted to mention it as a potentially improvement and automation of this algorithm.

Finally we see that the `loggedEvents` are NULL.

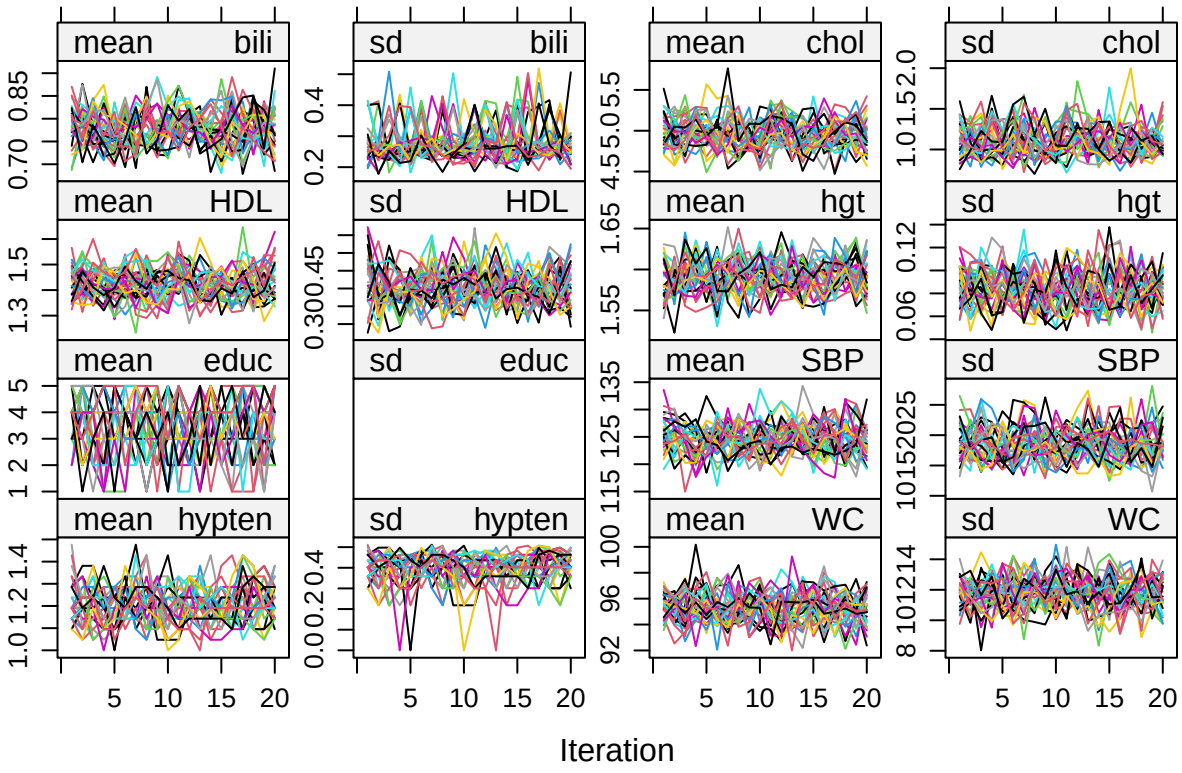
```
# analysis
imp <- mice(nhanes2,
            method=methods,
            maxit=20,
            m=30,
            seed=1,
            printFlag=FALSE)
imp$loggedEvents # NULL
```

```
## NULL
```

```
# TODO: grid search
```

So by looking at the chains we don't spot anything too out of the ordinary. We might be able to see some strong patterns on the `educ` variable, but that it to be expected since it's a categorical variable with only a few levels and only 0.2% of their entries missing. So for the moment every chain looks to be normal and well behaved.

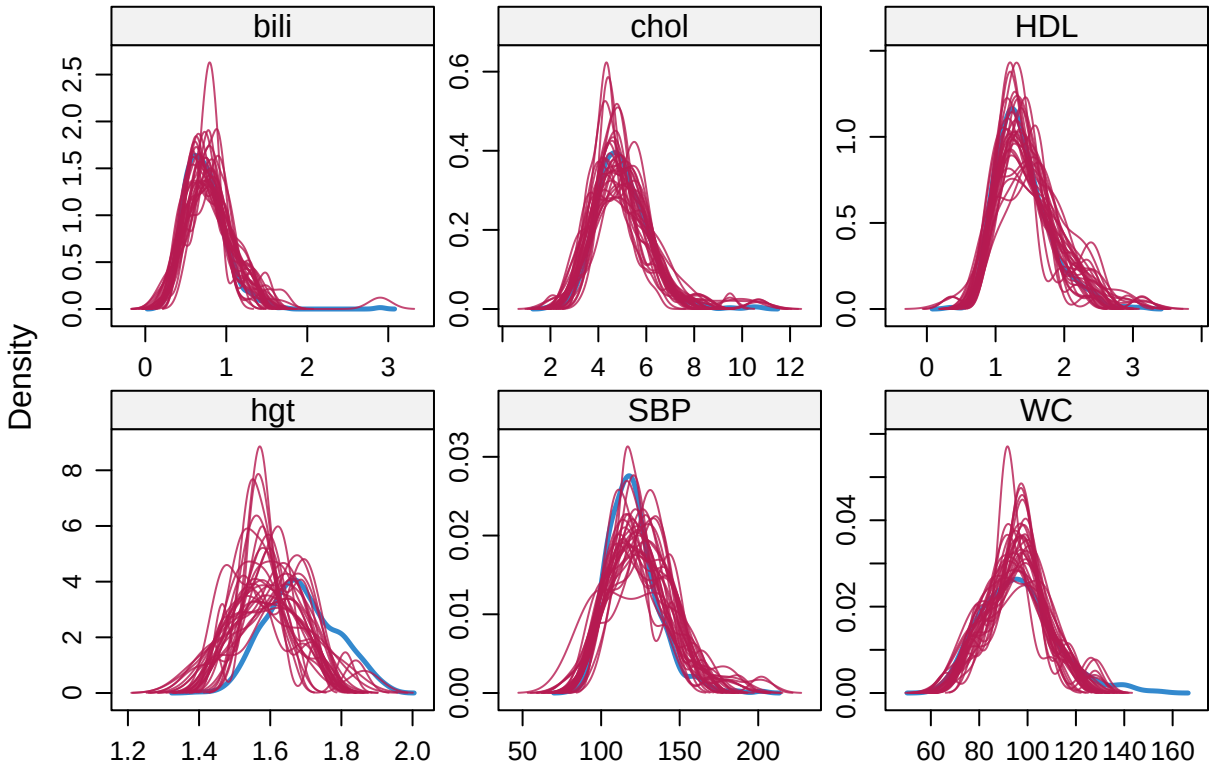
```
# checking convergence
plot(imp, layout=c(4,4))
```



However, after seeing the Histograms we start to see that something might be wrong with the `hgt` variable after all. We can clearly see that the imposed lines are not following very well the blue line, indicating that the imposed values don't have the same distribution as the complete cases on the original dataset. Furthermore, we see that the blue line looks like a mixture model of two Normals, or a *Binormal* distribution, which

makes sense, because the distribution of the whole population consists males and females and we can assume that they have statistically significant average heights. Our imposed data not only missed the mean of the distribution, but they also missed the structure of the distribution. `hgt` was the only variable that we changed the imposing method, so it's clear not that we should change it back.

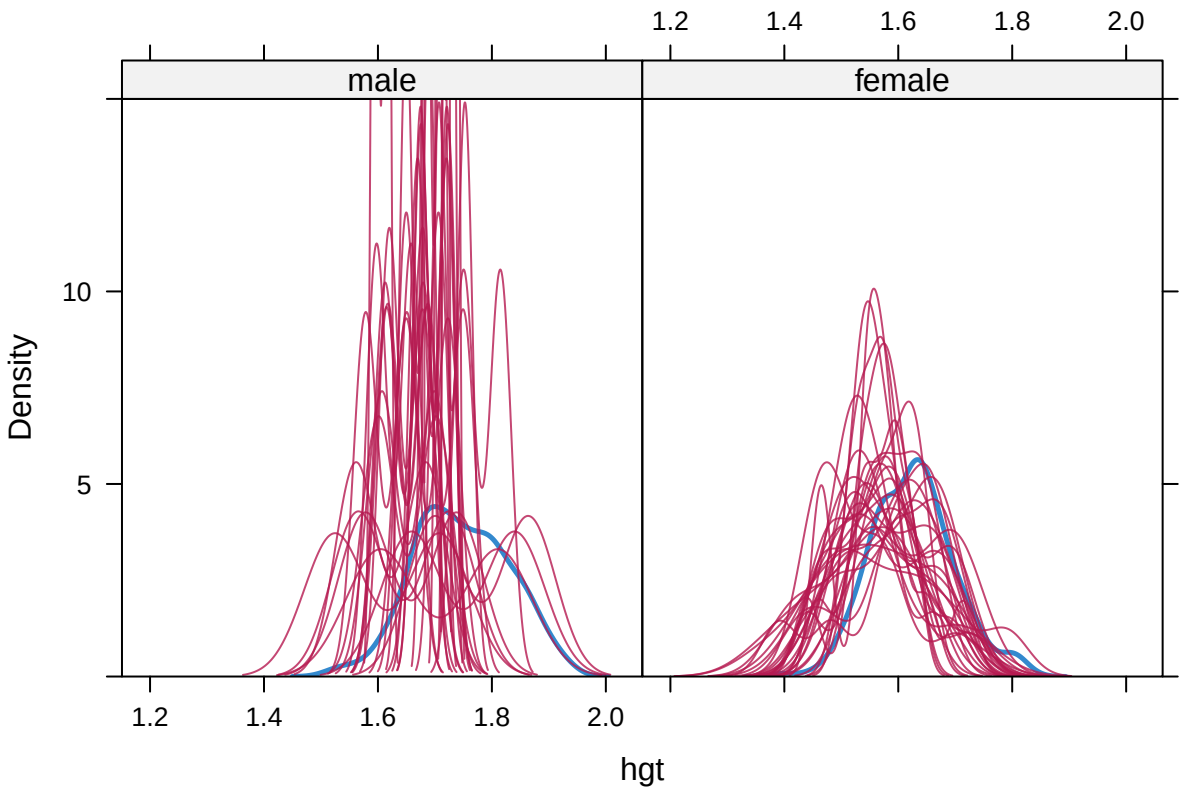
```
# maxit = 20, m=30 no strong patterns visible

densityplot(imp)
```




```
# height's actual distribution appears to be biNormal which makes sense since height has a big correlat
# it appears as if the imputed values for height are a bit biased towards lower numbers, however they d
# we should not be to worries about this.
```

Taking a more granular look we see the problem clearly on the males. The imposed distributions for height of the `M=30` datasets when `gender=male` are very poorly approximating the complete cases.

```
densityplot(imp, ~hgt|gender, ylim=c(0,15))
```



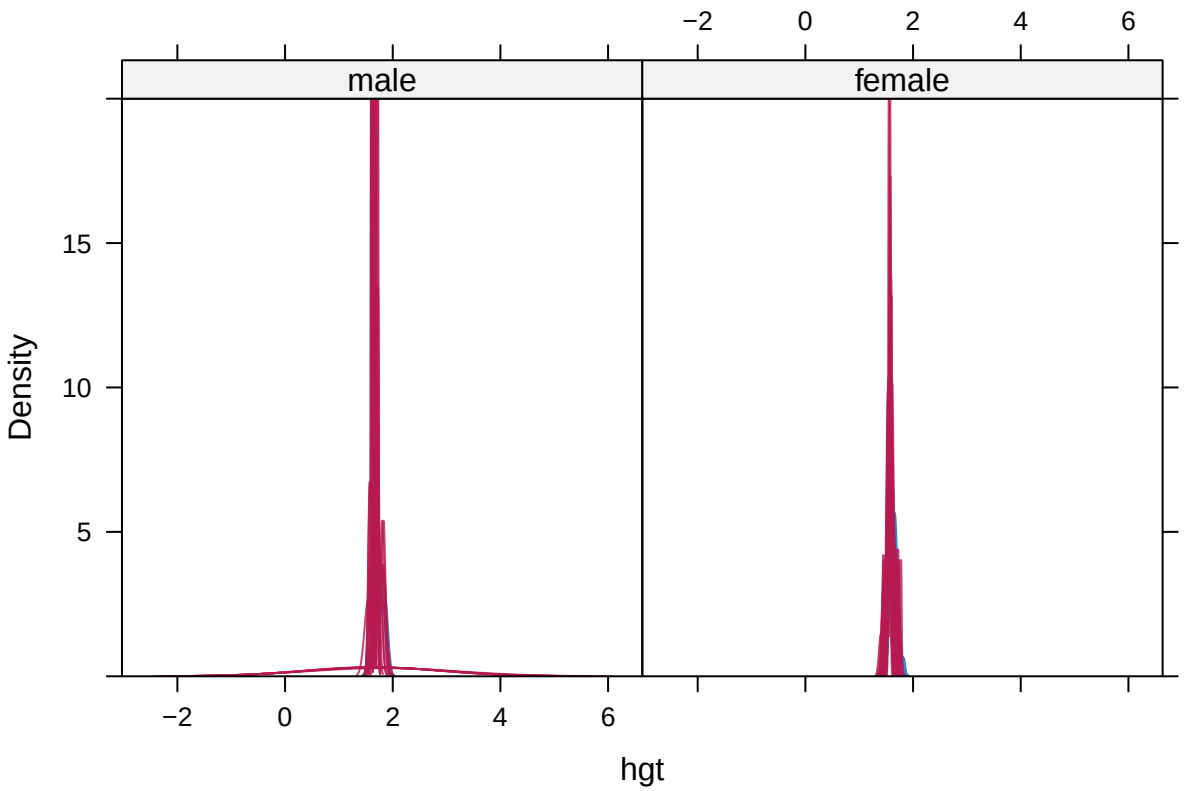

```
#here we can see more clearly where the problem on the above diagram comes from. It comes from the esti
# it doesn't really cover very well the spectrum since it over attributes the median ~1.7...
```

After changing the method back to `pmm` from `norm` we see a much better coverage hense it should stay like that.

```
methods['hgt'] <- 'pmm'
imp <- mice(nhanes2,
            method=methods,
            maxit=20,
            m=30,
            seed=1,
            printFlag=FALSE)
imp$loggedEvents # NULL
```

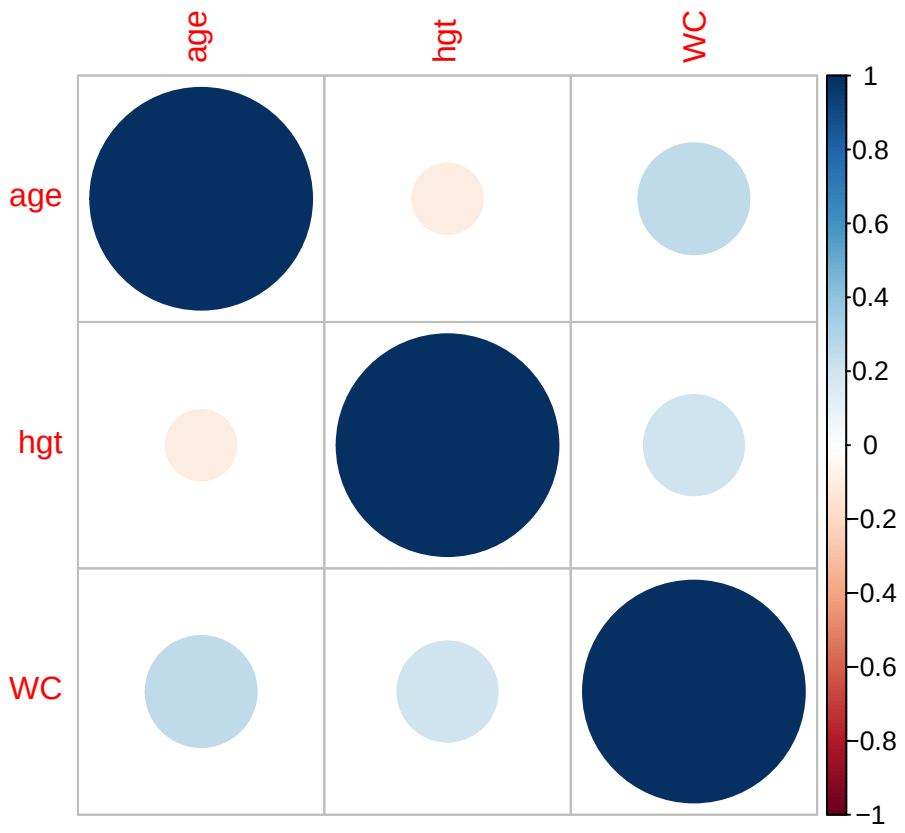
```
## NULL
```

```
densityplot(imp, ~hgt|gender, ylim=c(0,20))
```



So after changing the method back, we can start our analysis and see our model. Before that I would like to see the correlation plot for our dependent variables in order have a feeling of multicolinearity. It's very evident that the `WC` is correlated to `age` and to `hgt`. From the theory of OLS we know that the OLS method will give unbiased estimators in the presence of multicolinearity, but not efficient ones. Hence the R^2 produced will be slightly overestimated. Whit that in mind the final model shows that Weight is mostly influenced by height since our $\hat{\beta}_3 = 52$, which makes total sense. All other variables coefficients have relatively little influence on the weight. An important observation is also that genders coefficient, has a `p-values` of `0.122` which is statistically insignificant. That result combined with the multicolinearity we observed from the correlation diagram could indicate that a simpler model, one without gender as an independent variable could be a better one.

```
comp1 <- complete(imp, 1)

# proceed to analysis
model_data <- comp1[,c(4,7,12)]
M <- cor(model_data)
corrplot(M)
```




```
fit <- with(imp, lm(wgt ~ gender + age + hgt + WC))
summary(fit$analyses[[1]])
```

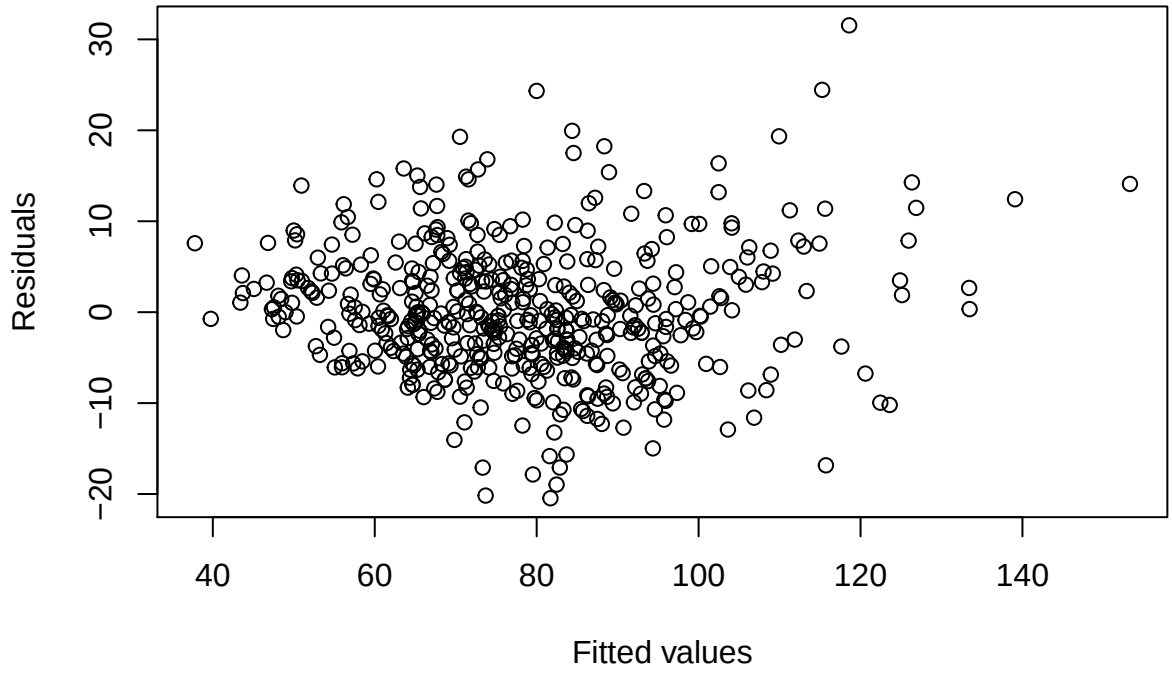
```
##
## Call:
## lm(formula = wgt ~ gender + age + hgt + WC)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.4638  -4.5537  -0.4955   3.8854  31.5403
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -100.51035    7.50652 -13.390  < 2e-16 ***
## genderfemale   -1.26815    0.81952  -1.547    0.122
## age            -0.15827    0.02085  -7.590  1.6e-13 ***
## hgt            52.15392    4.29615  12.140  < 2e-16 ***
## WC              1.02795    0.02213  46.452  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.179 on 495 degrees of freedom
## Multiple R-squared:  0.8575, Adjusted R-squared:  0.8563
## F-statistic: 744.6 on 4 and 495 DF,  p-value: < 2.2e-16
```

Finally we need to calculate our confidence intervals for our coefficients and compute the *adj* R^2. so we need

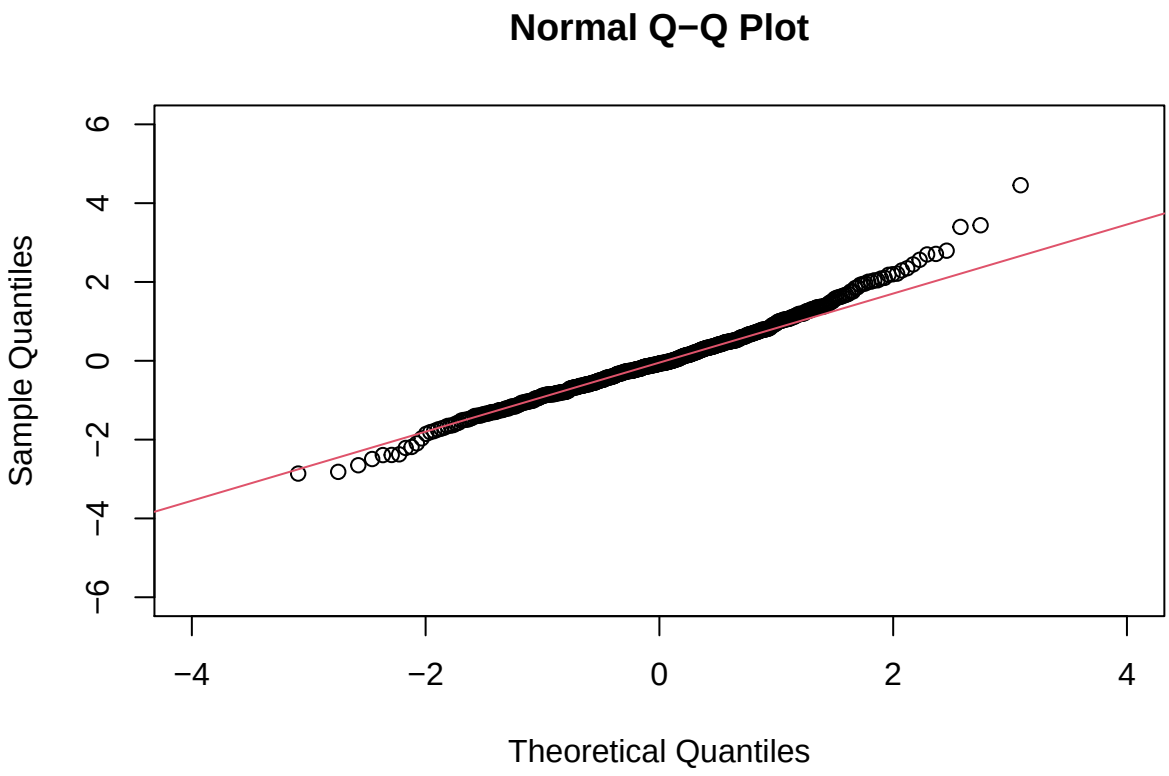
to make our final checks in order to trust the analysis. We need to check the second big disease a linear regression model might have, heteroskedasticity.

By plotting the residuals against the fitted values, we see that they are centered around `0` which is a good thing. However we also notice that the variance doesn't remain the same as it get's bigger the furthers right we move on the x-axis. It looks mostly like a cloud though which is a good sign so I would say that no further action is required. Similar conclusions can be reached when checking the QQ-plot with the tails being a bit of which is always acceptable to that extend.

```
plot(fit$analyses[[1]]$fitted.values, residuals(fit$analyses[[1]]),
xlab = "Fitted values", ylab = "Residuals")
```




```
# QQ plot
qqnorm(rstandard(fit$analyses[[1]]), xlim = c(-4, 4), ylim = c(-6, 6))
qqline(rstandard(fit$analyses[[1]]), col = 2)
```




```
# pooling
pooled_ests <- pool(fit)
summary(pooled_ests, conf.int = TRUE)
```

```
##           term     estimate  std.error  statistic       df      p.value
## 1  (Intercept) -100.8520702 7.67289217 -13.143945 449.6454 0.000000e+00
## 2 genderfemale   -1.3850796 0.83448095  -1.659810 469.9080 9.761988e-02
## 3          age   -0.1576829 0.02141424  -7.363458 451.6777 8.562040e-13
## 4          hgt   52.4292200 4.39636603  11.925581 444.3719 0.000000e+00
## 5           WC    1.0260613 0.02232811  45.953795 481.3369 0.000000e+00
##          2.5 %      97.5 %
## 1 -115.9312510 -85.7728894
## 2   -3.0248557   0.2546965
## 3   -0.1997668  -0.1155990
## 4   43.7889680  61.0694720
## 5    0.9821887   1.0699340
```

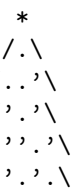
```
cat('\n\n')
```

```
pool.r.squared(pooled_ests, adjusted = TRUE)
```

```
##               est     lo 95     hi 95 fmi
## adj R^2 0.8559941 0.8304596 0.8779613 NaN
```

We present here the final confidence intervals per variable coefficient and the final $adj\ R^2 = 85.6$ which is a good indication that our model is a good fit to the data. However of course as said earlier we have to keep in mind that we have observed some multicolinearity on the model and potentially some heteroskedasticity, which both have an effect on the $adj\ \ R^2$

Happy Christmas and a happy new year



“ ‘ """“/‘;‘;’ ”” ’ ”’” jgs ^^[_]^^