ILAA Tutorial

Jose Tamez

2023-11-25

Table of Contents

# Introduction

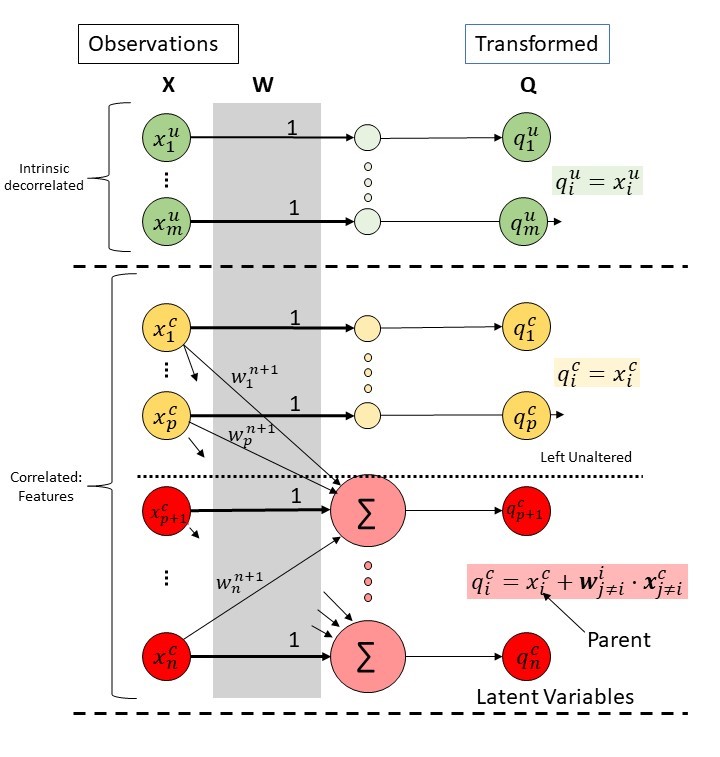


Figure 1: The UPLTM

Iterative Linear Association Analysis (ILAA) is a computational method that estimates the Exploratory Residualization Transform (ERT), an special case of the UPLTM as seen in Figure 1. ERT is estimated from a sample of multidimensional data. and mitigates muticollinearity issues via variable residualization.

The dataframe with reduced multicollinarity issues, , is estimated by:

where is the observed dataframe and is the Exploratory Residualization Transform (ERT) matrix.

The returned transformation matrix can be used to:

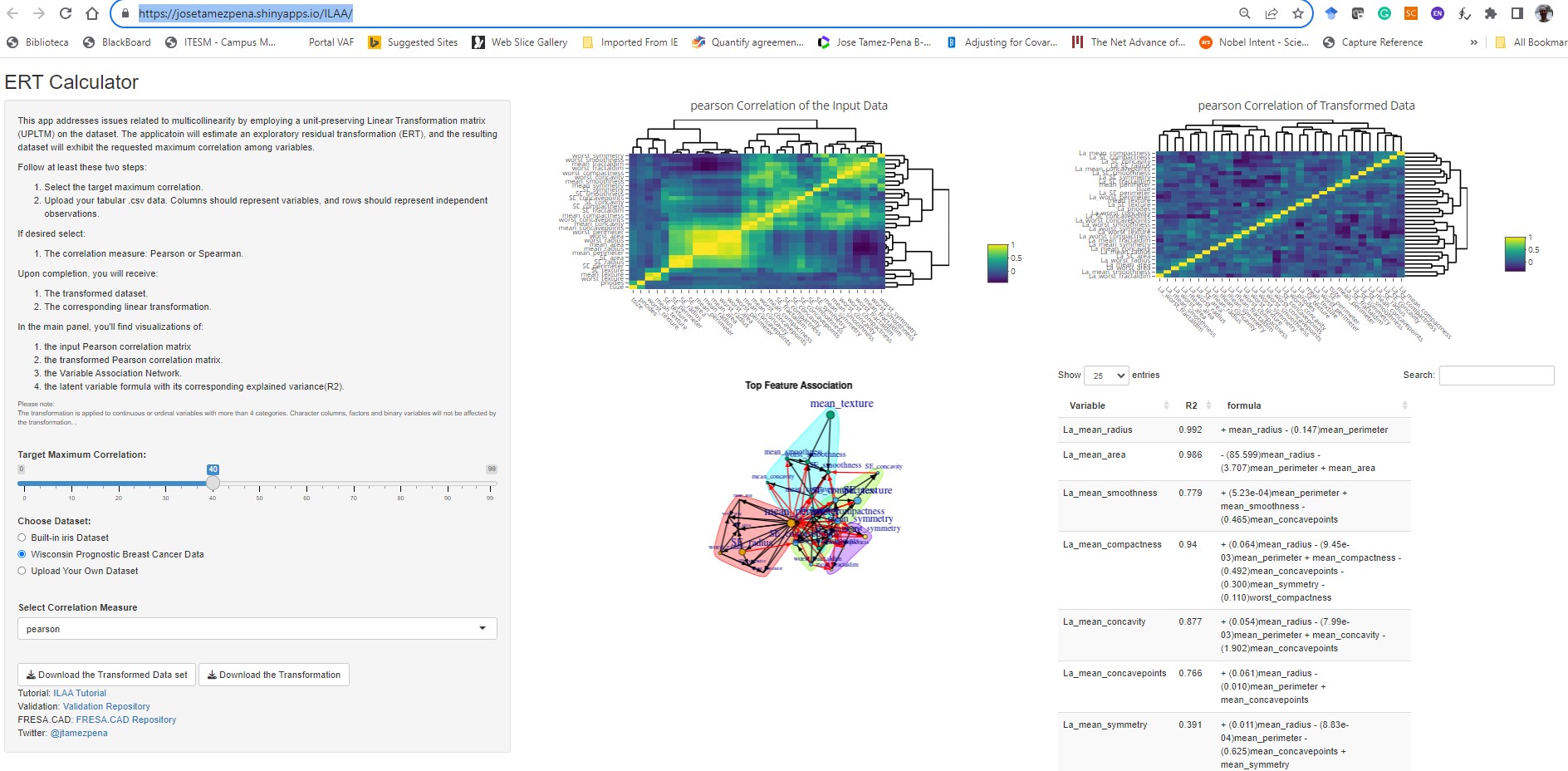
1. Do exploratory analysis of latent variables and their association to the observed variables
2. Do exploratory discovery of latent variables associated with a specific outcome-target
3. Addressing multicollinearity issues in linear regression models
   1. Better estimation and interpretation of model variables
   2. Improve linear model performance
4. Simplify the multidimensional search space for many ML algorithms

The objective of this tutorial is to guide users in using the ILAA to effectively accomplish the aforementioned tasks. The tutorial will showcase:

* Transform a data frame affected by data multicollinearity into a new a data frame with a maximum degree of data correlation among variables
* Visualize the transformation matrix
* Explore the returned formulas for each one of the returned latent variables
* Understand and interpret the returned latent variables
* Use ILAA as a pre-processing step to model a specific target outcome using linear models
  + Explore the model in the transformed space
  + Get the observed variables coefficients.

## Shiny App

Users can test ILLA application using the ERT calculator: [ILLA Shiny App](https://josetamezpena.shinyapps.io/ILAA/):

[](https://josetamezpena.shinyapps.io/ILAA/)

## The Libraries

ILAA is a wrapper of the more general method of data decorrelation algorithm (IDeA) implemented in R, and both are part of the FRESA.CAD 3.4.6 package.

## From git hub  
#First install package devtools  
#library(devtools)  
#install\_github("joseTamezPena/FRESA.CAD")  
  
## For ILAA  
library("FRESA.CAD")  
  
## For network analysis  
library(igraph)  
  
## For multicollinearity  
library(multiColl)  
library(car)  
library("colorRamps")

# Test Data

For this tutorial I’ll use the body-fat prediction data set. The data was downloaded from Kaggle:

<https://www.kaggle.com/datasets/fedesoriano/body-fat-prediction-dataset>

The Kaggle data disclaimer:

“Source The data were generously supplied by Dr. A. Garth Fisher who gave permission to freely distribute the data and use for non-commercial purposes.

Roger W. Johnson Department of Mathematics & Computer Science South Dakota School of Mines & Technology 501 East St. Joseph Street Rapid City, SD 57701

email address: [rwjohnso@silver.sdsmt.edu](mailto:rwjohnso@silver.sdsmt.edu) web address: <http://silver.sdsmt.edu/~rwjohnso>”

## Loading the Data

The BodyFat dataset contains the density information, a not direct measurement. In this tutorial, we will remove the density and we will try to model the body fat based on antrophometry.

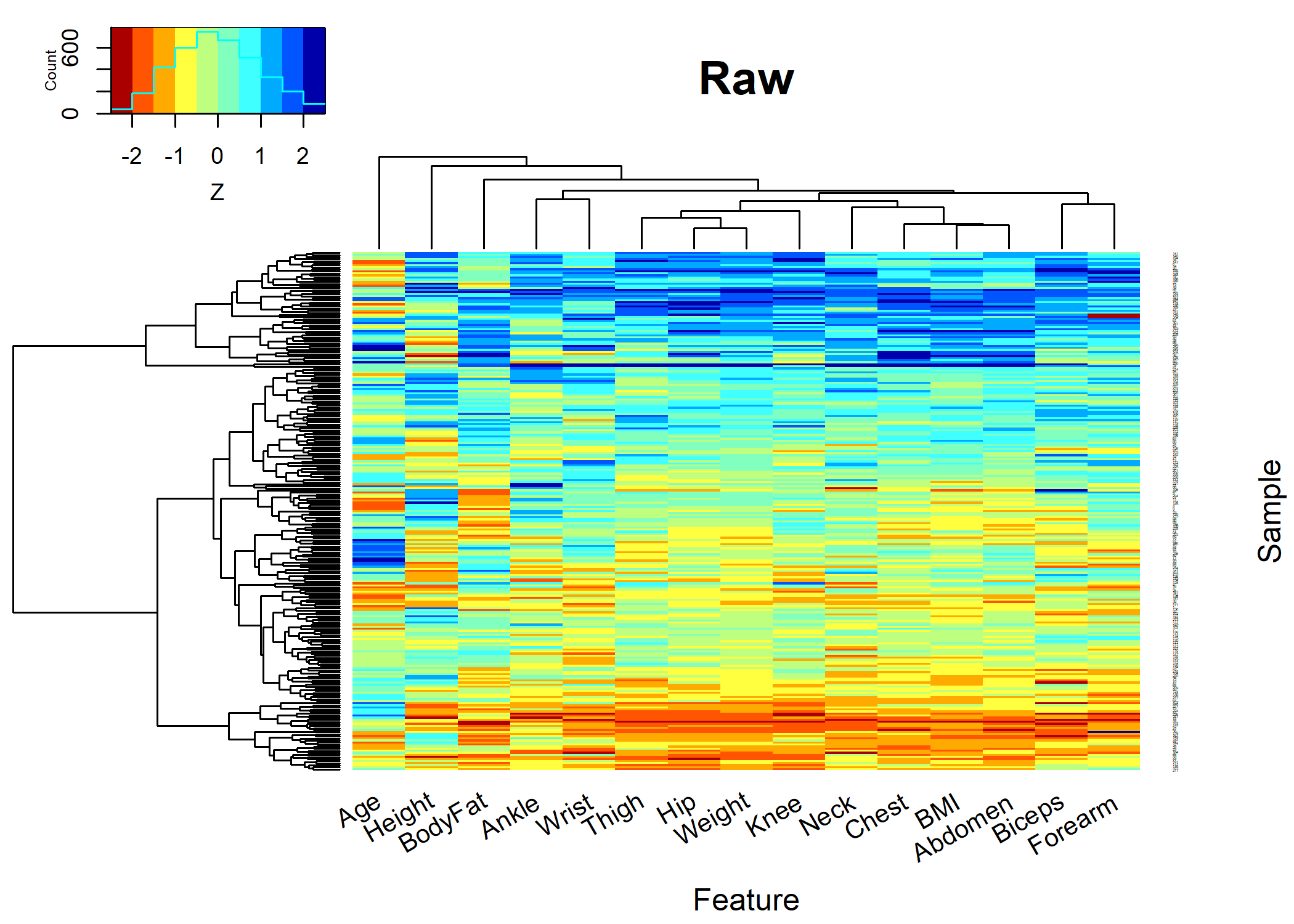
The following code snippet loads the data and removes the density information from the data. It also computes the Body Mass Index (BMI)

body\_fat <- read.csv("~/GitHub/LatentBiomarkers/Data/BodyFat/BodyFat.csv", header=TRUE)  
  
### Removing density as estimator  
body\_fat$Density <- NULL  
  
body\_fat$BMI <- 10000\*body\_fat$Weight\*0.453592/((body\_fat$Height\*2.54)^2)  
## Removing subjects with data errors  
body\_fat <- body\_fat[body\_fat$BMI<=50,]

### The Heatmap of the Raw Data

Now, here we show the heatmap of the dataframe:

bkcolors <-seq(-2.5, 2.5, by = 0.5)  
smap <- FRESAScale(body\_fat,method="OrderLogit")$scaledData  
 hm <- gplots::heatmap.2(scale(as.matrix(smap)),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(colorRamps::matlab.like(length(bkcolors)-1)),  
 breaks = bkcolors,  
 main = "Raw",  
 cexRow = 0.15,  
 cexCol = 1.00,  
 srtCol=30,  
 srtRow=0,  
 key.title=NA,  
 key.xlab="Z",  
 xlab="Feature", ylab="Sample")



par(op)

# ILAA Unsupervised Processing

The ILLA function is defined as follows:

decorrelatedData <- ILAA(data=NULL,  
 thr=0.80,  
 method=c("pearson","spearman"),  
 Outcome=NULL,  
 drivingFeatures=NULL,  
 maxLoops=100,  
 verbose=FALSE,  
 bootstrap=0  
 )

where:

* data: The source data-frame
* thr : The target correlation goal.
* method : Defines the correlation measure
* Outcome :The name of the target variable, and it is required for supervised learning
* drivingFeatures : Defines a set of variables that are aimed to be basis unaltered vectors
* maxLoops : The maximum number of iterations cycles
* verbose : Display the evolution of the algorithm.
* bootstrap : The number of bootstrap estimations. (True bootstrap when n>500, 5% bootstrap at n<=500)

At return of the ILLA function is a decorrelated dataframe that shares the same dimensions as the input dataframe. The dataframe has the following attributes:

RTM <- attr(decorrelatedData,"UPLTM")   
fscore <- attr(decorrelatedData,"fscore");  
drivingFeatures <- attr(decorrelatedData,"drivingFeatures");  
adjustedpvalue <- attr(decorrelatedData,"unipvalue")  
RCritical <- attr(decorrelatedData,"R.critical")  
EvolutionData <- attr(decorrelatedData,"IDeAEvolution")  
VarRatio <- attr(decorrelatedData,"VarRatio")

Attributes details:

* UPLTM: The UPLTM matrix that can be used to decorrelated or analyze variables associations
* fscore: A numeric vector with the final feature score of each analyzed variable. The fscore contains the number of times a variable was used as an independent variable minus the times it was a dependent variable.
* drivingFeatures : The ordered character vector indicating the hierarchy of the variables for tiebreak
* unipvalue : The adjusted p-values used to define a true variable-to-variable association inside the linear modeling
* R.critical : The pearson R critical value used to filter-out false association between variables.
* IDeAEvolution : A list with two elements:
  + Corr: The evolution of the maximum observed correlation.
  + Spar: The evolution of the matrix sparcity.
* VarRatio : A vector indicating the ratio of the observed variance explained by the latent variable model.

## ILLA Auxiliary Functions

FRESA.CAD provide the following auxiliary functions:

newTransformedData <- predictDecorrelate(decorrelatedData,NewData)  
theBetaCoefficientts <- getLatentCoefficients(decorrelatedData)  
fromLatenttoObserved <- getObservedCoef(decorrelatedData,latentModel)

* predictDecorrelate() Rotates any new data set based on the output of an ILAA transformed data set.
* getLatentCoefficients() Returns a list of all the beta coefficients for each one of the discovered latent variables. The attribute: “LatentCharFormulas” returns a list of the character string of the corresponding latent variable formula.
* getObservedCoef() returns the beta coefficients on the observed space of any linear model that was trained on the UPLTM space.

## Sample Usage

By default, the ILAA function will target a correlation lower than 0.8 using the Pearson correlation measure. But user has the freedom to chose between robust fitting with Spearman correlation measure, and/or set the level of feature association by lowering the threshold. The following snippet shows the different options.

# Default call  
body\_fat\_Decorrelated <- ILAA(body\_fat)  
varRatio\_D <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_D <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_D <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar  
  
# Explore the convergence metrics in verbose mode  
body\_fat\_Decorrelated <- ILAA(body\_fat,verbose=TRUE)

fast | LM | Weight BodyFat Age Weight Height Neck Chest 0.40000000 0.06666667 1.00000000 0.13333333 0.53333333 0.73333333

Included: 15 , Uni p: 0.01 , Base Size: 1 , Rcrit: 0.1467743

1 <R=0.944,thr=0.900>, Top: 2< 1 >[Fa= 2](2%20,%203%20,%200),<|><>Tot Used: 5 , Added: 3 , Zero Std: 0 , Max Cor: 0.888 2 <R=0.888,thr=0.800>, Top: 1< 5 >[Fa= 2](1%20,%205%20,%202),<|><>Tot Used: 9 , Added: 5 , Zero Std: 0 , Max Cor: 0.860 3 <R=0.860,thr=0.800>, Top: 1< 1 >[Fa= 2](1%20,%201%20,%202),<|><>Tot Used: 10 , Added: 1 , Zero Std: 0 , Max Cor: 0.959 4 <R=0.959,thr=0.950>, Top: 1< 1 >[Fa= 2](1%20,%201%20,%202),<|><>Tot Used: 10 , Added: 1 , Zero Std: 0 , Max Cor: 0.735 5 <R=0.735,thr=0.800> [ 5 ], 0.4782625 Decor Dimension: 10 Nused: 10 . Cor to Base: 7 , ABase: 15 , Outcome Base: 0

# Robust Linear Fitting with the Spearman correlation measure  
body\_fat\_Decorrelated <- ILAA(body\_fat,method="spearman")  
varRatio\_S <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_S <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_S <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar  
  
# Lowering the threshold  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.4)  
varRatio\_P\_40 <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_P\_40 <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_P\_40 <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar  
  
# Trying to achieve the maximum independence between variables, i.e., thr=0.0  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.0)  
varRatior\_P\_00 <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_P\_00 <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_P\_00 <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar  
  
# The BMI and BodyFat variables Driving the Decorrelation  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.2,drivingFeatures=c("BMI","BodyFat"))  
varRatior\_P\_20\_D <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_P\_20\_D <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_P\_20\_D <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar  
  
# The Bodyfat variable and its association will be Driving the Decorrelation process  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.2,Outcome="BodyFat",drivingFeatures="BodyFat")  
varRatior\_P\_20\_OD <- attr(body\_fat\_Decorrelated,"VarRatio")  
yCor\_P\_20\_OD <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Corr  
ySpar\_P\_20\_OD <- attr(body\_fat\_Decorrelated,"IDeAEvolution")$Spar

### Latent Models Variance Ratios

Every change in parameters will create different solutions of the ERT transform.

Here we will check the variance ratio of each latent model. Where the variance ratio can be interpreted as the percentage of the observed variance still present in the latent variable.

Note: Every time the variance ratio is 1, is an indication that the observed variable was not modeled by any other variable in the dataframe.

names(varRatio\_D) <- str\_remove\_all(names(varRatio\_D),"La\_")  
names(varRatio\_S) <- str\_remove\_all(names(varRatio\_S),"La\_")  
names(varRatio\_P\_40) <- str\_remove\_all(names(varRatio\_P\_40),"La\_")  
names(varRatior\_P\_00) <- str\_remove\_all(names(varRatior\_P\_00),"La\_")  
names(varRatior\_P\_20\_D) <- str\_remove\_all(names(varRatior\_P\_20\_D),"La\_")  
names(varRatior\_P\_20\_OD) <- str\_remove\_all(names(varRatior\_P\_20\_OD),"La\_")  
namesVar <- names(varRatio\_D)  
  
varratios <- rbind(Default=varRatio\_D,  
 Spearman=varRatio\_S[namesVar],  
 At\_40=varRatio\_P\_40[namesVar],  
 At\_0=varRatior\_P\_00[namesVar],  
 BMI\_BF=varRatior\_P\_20\_D[namesVar],  
 BodyFat\_Driven=varRatior\_P\_20\_OD[namesVar])  
pander::pander(varratios,caption="Unexplained variance ratio of latent models")

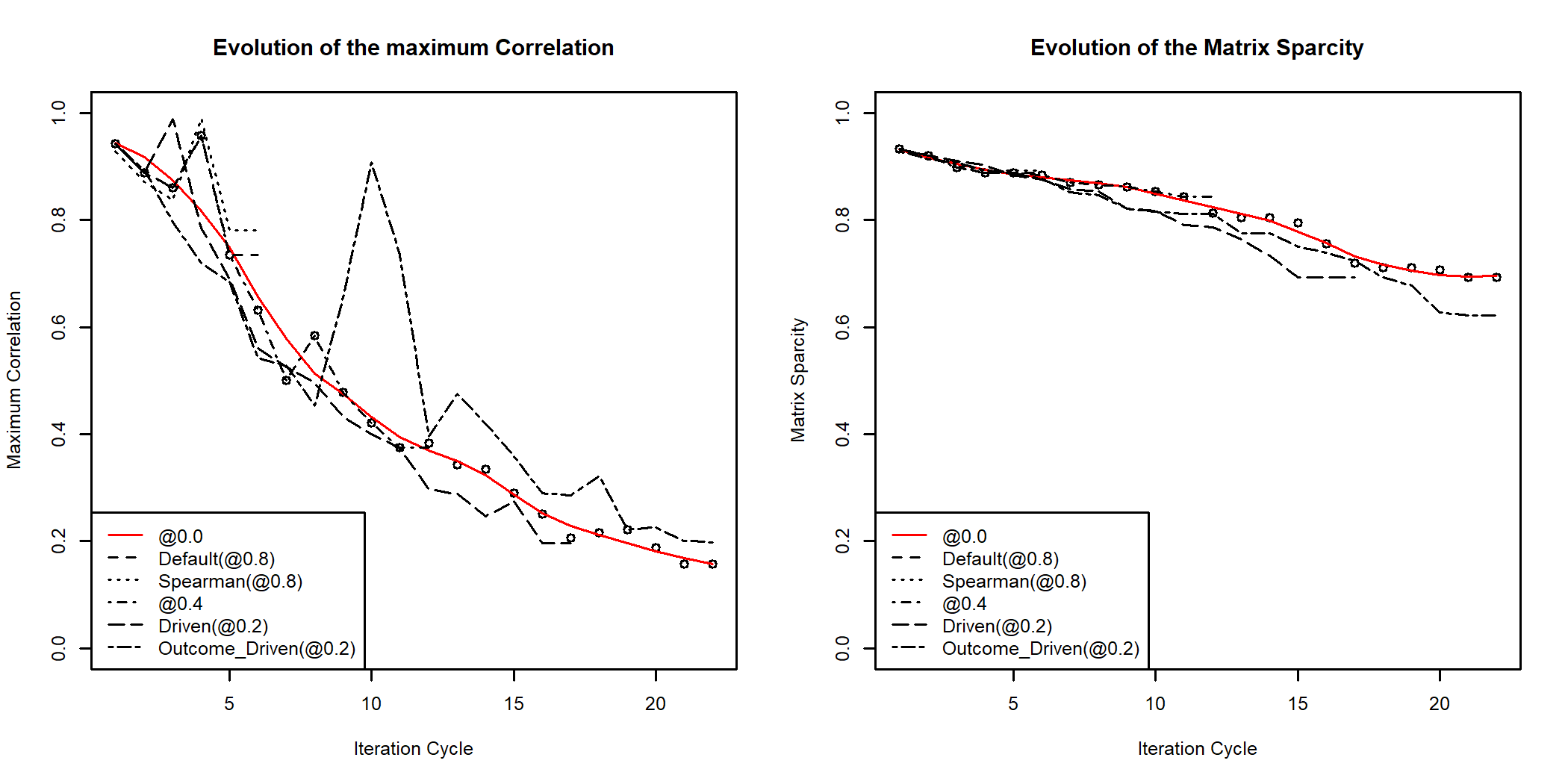
Unexplained variance ratio of latent models

|  | BodyFat | Age | Ankle | Forearm | Wrist | Weight | Biceps | Neck | Knee | Thigh | BMI | Chest | Abdomen | Hip | Height |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Default** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.0000 | 0.359 | 0.303 | 0.272 | 0.242 | 0.211 | 0.1710 | 0.1500 | 0.1096 | 0.0209 |
| **Spearman** | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.0000 | 1.000 | 0.303 | 0.273 | 0.242 | 0.212 | 0.1737 | 0.1500 | 0.1099 | 0.0221 |
| **At\_40** | 0.309 | 1.000 | 0.624 | 0.602 | 0.459 | 1.0000 | 0.359 | 0.303 | 0.272 | 0.194 | 0.211 | 0.1085 | 0.1500 | 0.1096 | 0.0209 |
| **At\_0** | 0.290 | 0.483 | 0.540 | 0.518 | 0.403 | 1.0000 | 0.320 | 0.269 | 0.228 | 0.183 | 0.211 | 0.1007 | 0.1277 | 0.0993 | 0.0209 |
| **BMI\_BF** | 0.475 | 0.491 | 0.583 | 0.517 | 0.386 | 0.2108 | 0.308 | 0.294 | 0.243 | 0.234 | 1.000 | 0.0980 | 0.0734 | 0.0779 | 0.0209 |
| **BodyFat\_Driven** | 1.000 | 0.494 | 0.565 | 0.513 | 0.383 | 0.0687 | 0.315 | 0.367 | 0.256 | 0.190 | 0.124 | 0.0984 | 1.0000 | 0.2344 | 0.0210 |

### Plotting the Evolution

Here we will use the attr(dataTransformed,"IDeAEvolution") to plot the evolution of the correlation measure and the evolution of the matrix sparsity.

par(mfrow=c(1,2),cex=0.5)  
  
# Correlation  
yval <- yCor\_P\_00  
xidx <- c(1:length(yval))  
plot(xidx,yval,  
 xlab="Iteration Cycle",  
 ylab="Maximum Correlation",  
 ylim=c(0,1.0),  
 main="Evolution of the maximum Correlation")  
 lfit <-try(loess(yval~xidx,span=0.5));  
 if (!inherits(lfit,"try-error"))  
 {  
 plx <- try(predict(lfit,se=TRUE))  
 if (!inherits(plx,"try-error"))  
 {  
 lines(xidx,plx$fit,lty=1,col="red")  
 }  
 }  
lines(xidx,yCor\_D[xidx],lty=2)  
lines(xidx,yCor\_S[xidx],lty=3)  
lines(xidx,yCor\_P\_40[xidx],lty=4)  
lines(xidx,yCor\_P\_20\_D[xidx],lty=5)  
lines(xidx,yCor\_P\_20\_OD[xidx],lty=6)  
legend("bottomleft",  
 legend=c("@0.0","Default(@0.8)","Spearman(@0.8)","@0.4","Driven(@0.2)","Outcome\_Driven(@0.2)"),  
 lty=c(1:6),  
 col=c("red","black","black","black","black","black"))  
   
# Sparsity   
yval <- ySpar\_P\_00  
  
plot(xidx,yval,  
 xlab="Iteration Cycle",  
 ylab="Matrix Sparcity",  
 ylim=c(0,1.0),  
 main="Evolution of the Matrix Sparcity")  
 lfit <-try(loess(yval~xidx,span=0.5));  
 if (!inherits(lfit,"try-error"))  
 {  
 plx <- try(predict(lfit,se=TRUE))  
 if (!inherits(plx,"try-error"))  
 {  
 lines(xidx,plx$fit,lty=1,col="red")  
 }  
 }  
lines(xidx,ySpar\_D[xidx],lty=2)  
lines(xidx,ySpar\_S[xidx],lty=3)  
lines(xidx,ySpar\_P\_40[xidx],lty=4)  
lines(xidx,ySpar\_P\_20\_D[xidx],lty=5)  
lines(xidx,ySpar\_P\_20\_OD[xidx],lty=6)  
legend("bottomleft",  
 legend=c("@0.0","Default(@0.8)","Spearman(@0.8)","@0.4","Driven(@0.2)","Outcome\_Driven(@0.2)"),  
 lty=c(1:6),  
 col=c("red","black","black","black","black","black"))



For the next part of the tutorial I’ll set the correlation goal to 0.2

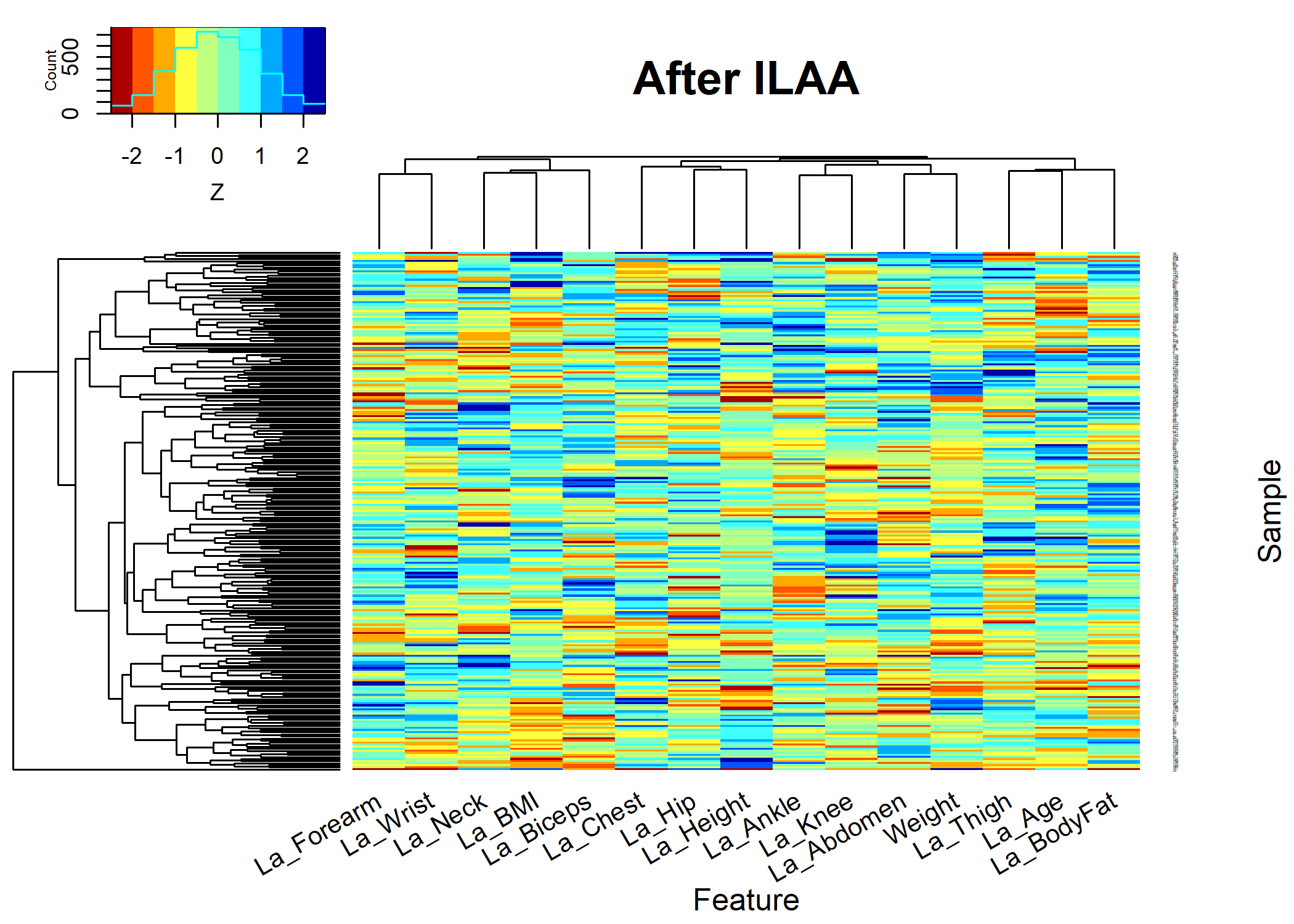
# Calling ILAA to achieve a final correlation of 0.2  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.2)  
pander::pander(attr(body\_fat\_Decorrelated,"VarRatio"))

| Weight | La\_Ankle | La\_Forearm | La\_Age | La\_Wrist | La\_Biceps | La\_BodyFat | La\_Neck | La\_Knee | La\_BMI | La\_Thigh | La\_Abdomen | La\_Chest | La\_Hip | La\_Height |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.555 | 0.518 | 0.483 | 0.403 | 0.32 | 0.29 | 0.279 | 0.228 | 0.211 | 0.183 | 0.132 | 0.104 | 0.0993 | 0.0209 |

## The Heatmap of the Transformed Data

Here we review the transformed data using a heatmap of the data

smap <- FRESAScale(body\_fat\_Decorrelated,method="OrderLogit")$scaledData  
 hm <- gplots::heatmap.2(scale(as.matrix(smap)),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(colorRamps::matlab.like(length(bkcolors)-1)),  
 breaks = bkcolors,  
 main = "After ILAA",  
 cexRow = 0.15,  
 cexCol = 1.00,  
 srtCol=30,  
 srtRow=0,  
 key.title=NA,  
 key.xlab="Z",  
 xlab="Feature", ylab="Sample")



par(op)

## Data Frame Attributes

The returned data matrix contains the following attributes

attr(body\_fat\_Decorrelated,"UPLTM") #The transformation matrix  
 attr(body\_fat\_Decorrelated,"fscore") #The score of each feature  
 attr(body\_fat\_Decorrelated,"drivingFeatures") #The list of driving features  
 attr(body\_fat\_Decorrelated,"R.critical") #The estimated minimum achieviable correlation  
 attr(body\_fat\_Decorrelated,"IDeAEvolution") #Evolution of the algorithm  
 attr(body\_fat\_Decorrelated,"VarRatio") #Variance Ratios: var(Latent)/Var(obs)

The main attributes is “UPLTM”. That stores the specific linear transformation matrix from observed variables to the latent variable.

The next relevant attribute is the “VarRatio", this attributive stores the fraction of the original feature variance that is still present in the latent variable. All non-altered variables return a”VarRatio” of 1.

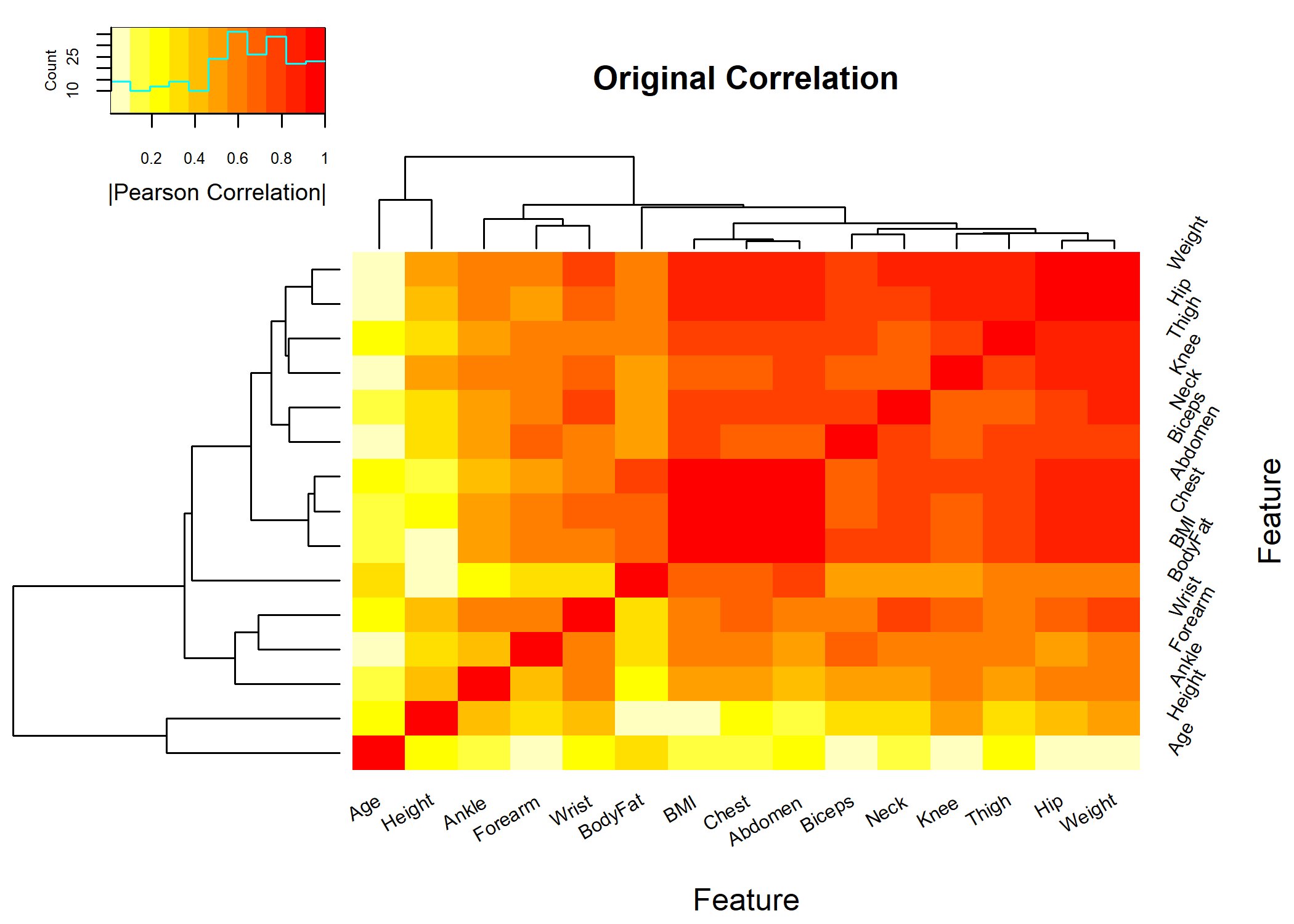
The “IDeAEvolution” attribute can be used to verify if the algorithm achieved the target correlation goal, and the sparsity of the returned matrix.

## The ILAA Transformed Data

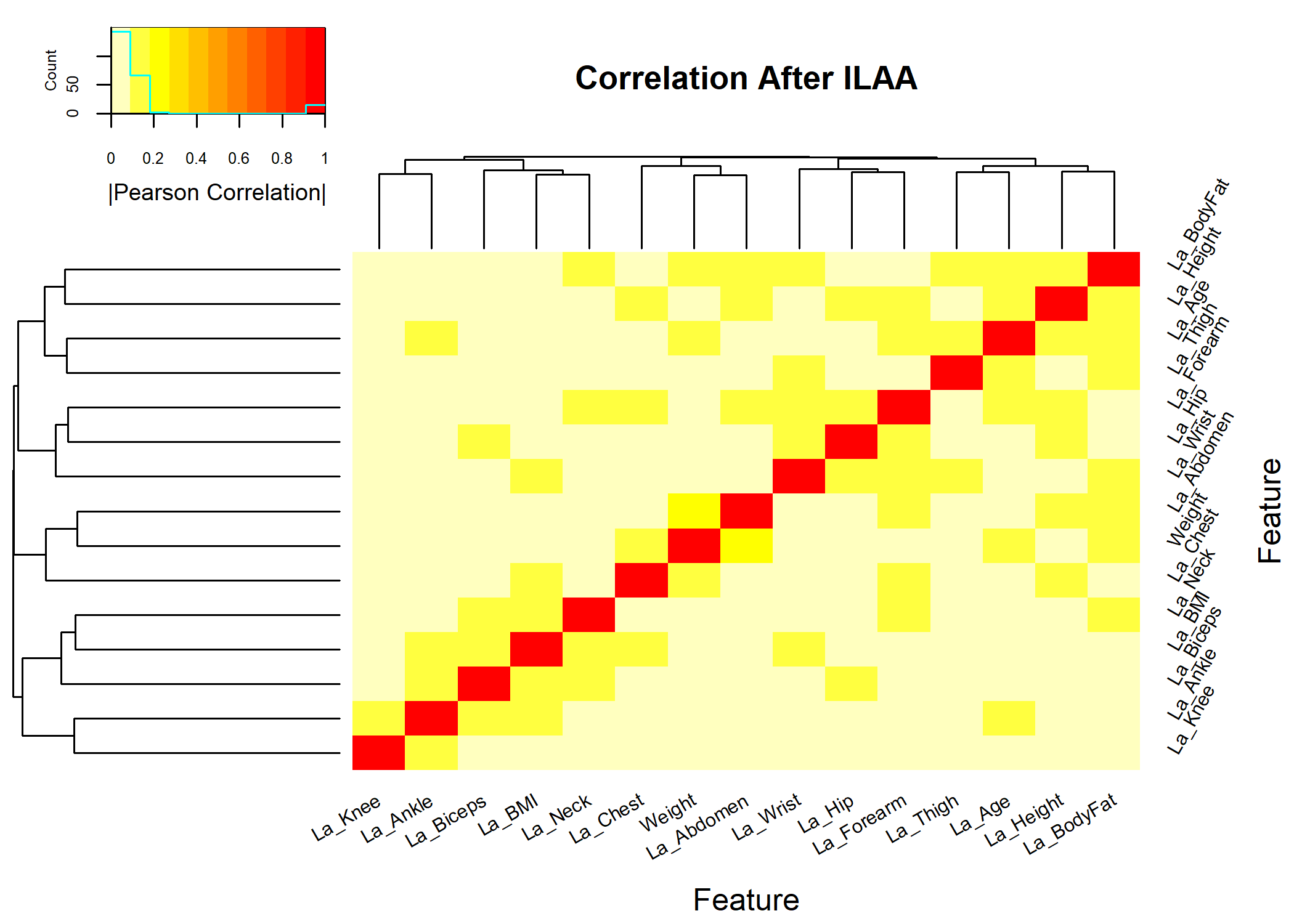
Before exploring into more detail, the properties of the ILAA results. Let us first verify that the returned matrix does not contain features with very high correlation among them.

Here I’ll plot the original correlation and the correlation of the returned data set.

# The original  
 par(cex=0.6,cex.main=0.85,cex.axis=0.7)  
 cormat <- cor(body\_fat,method="pearson")  
 gplots::heatmap.2(abs(cormat),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(heat.colors(11)),  
 main = "Original Correlation",  
 cexRow = 0.75,  
 cexCol = 0.75,  
 srtCol=30,  
 srtRow=60,  
 key.title=NA,  
 key.xlab="|Pearson Correlation|",  
 xlab="Feature", ylab="Feature")



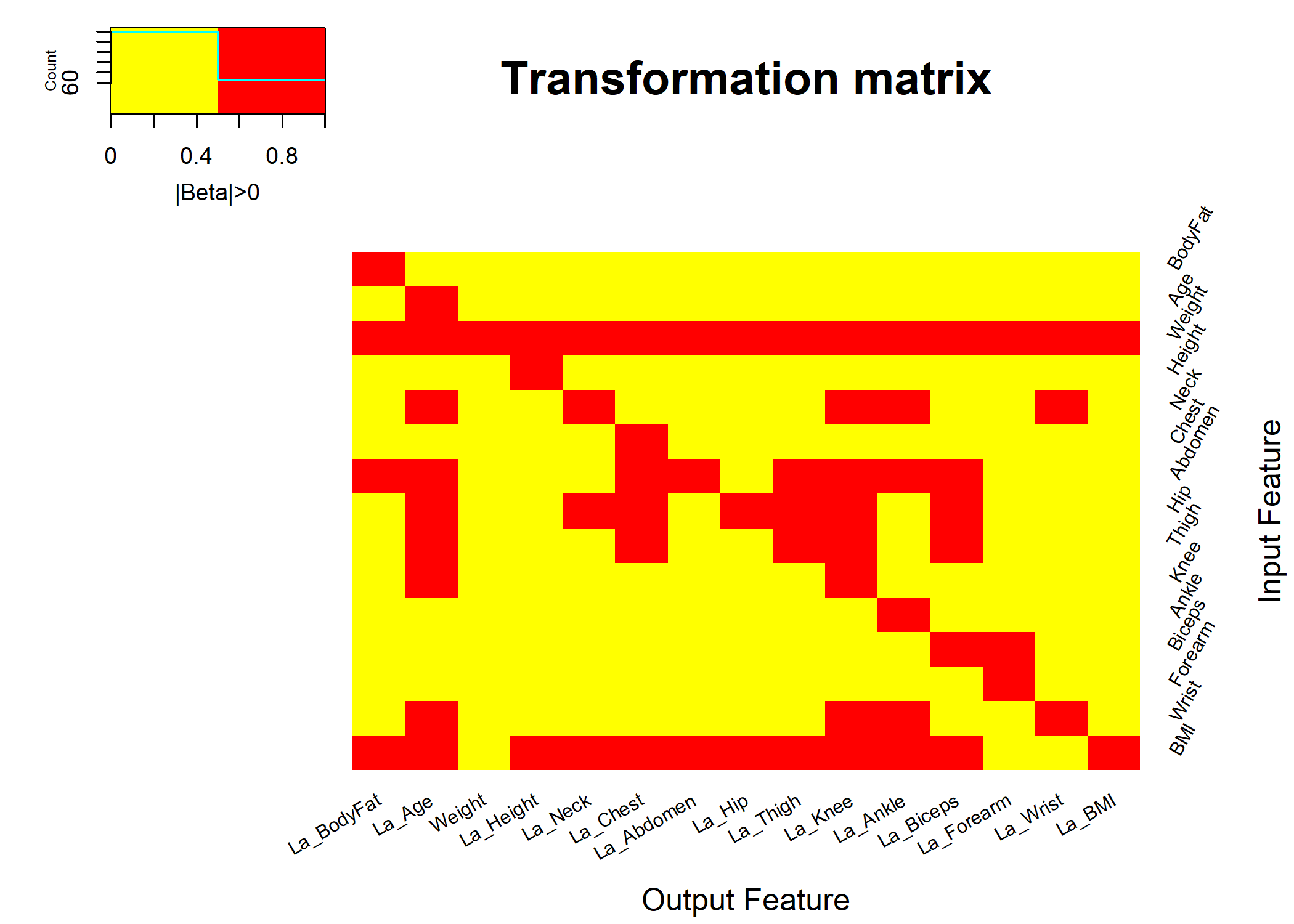
# The transformed  
 cormat <- cor(body\_fat\_Decorrelated,method="pearson")  
 gplots::heatmap.2(abs(cormat),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(heat.colors(11)),  
 main = "Correlation After ILAA",  
 cexRow = 0.75,  
 cexCol = 0.75,  
 srtCol=30,  
 srtRow=60,  
 key.title=NA,  
 key.xlab="|Pearson Correlation|",  
 xlab="Feature", ylab="Feature")



## Exploring the Transformation

The attr(body\_fat\_Decorrelated,"UPLTM") returns the transformation matrix. The UPLTM is sparse, here I show a heat map of the transformation matrix that shows which elements are different from zero.

UPLTM <- attr(body\_fat\_Decorrelated,"UPLTM")  
   
 gplots::heatmap.2(1.0\*(abs(UPLTM)>0),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(heat.colors(2)),  
 Rowv=NULL,  
 Colv="Rowv",  
 dendrogram="none",  
 main = "Transformation matrix",  
 cexRow = 0.75,  
 cexCol = 0.75,  
 srtCol=30,  
 srtRow=60,  
 key.title=NA,  
 key.xlab="|Beta|>0",  
 xlab="Output Feature", ylab="Input Feature")



## The Latent Formulas

The sparsity of the UPLTM matrix can be analyzed to get the formula for each one of the latent formulas. The getLatentCoefficients() and its attribute: attr(LatentFormulas,"LatentCharFormulas") can be used to display the formula of the latent variables.

# Get a list with the latent formulas' coefficients  
LatentFormulas <- getLatentCoefficients(body\_fat\_Decorrelated)  
  
# A string character with the formulas can be obtained by:  
charFormulas <- attr(LatentFormulas,"LatentCharFormulas")  
pander::pander(as.matrix(charFormulas))

|  |  |
| --- | --- |
| **La\_BodyFat** | + BodyFat + (0.120)Weight - (0.800)Abdomen - (0.480)BMI |
| **La\_Age** | + Age + (0.363)Weight - (0.636)Neck - (1.117)Abdomen - (8.09e-04)Hip + (2.273)Thigh - (1.732)Knee - (5.032)Wrist - (0.864)BMI |
| **La\_Height** | - (0.191)Weight + Height + (1.339)BMI |
| **La\_Neck** | - (0.100)Weight + Neck + (0.172)Hip - (0.074)BMI |
| **La\_Chest** | - (0.140)Weight + Chest - (0.363)Abdomen + (0.419)Hip + (0.265)Thigh - (1.082)BMI |
| **La\_Abdomen** | - (0.094)Weight + Abdomen - (1.865)BMI |
| **La\_Hip** | - (0.181)Weight + Hip - (0.430)BMI |
| **La\_Thigh** | - (0.056)Weight + (0.137)Abdomen - (0.489)Hip + Thigh - (0.256)BMI |
| **La\_Knee** | - (0.056)Weight + (0.067)Neck - (0.017)Abdomen - (0.046)Hip - (0.121)Thigh + Knee - (0.406)Wrist + (0.229)BMI |
| **La\_Ankle** | - (0.035)Weight + (0.098)Neck + (0.069)Abdomen + Ankle - (0.594)Wrist - (0.128)BMI |
| **La\_Biceps** | - (0.081)Weight + (0.075)Abdomen + (0.098)Hip - (0.200)Thigh + Biceps - (0.140)BMI |
| **La\_Forearm** | - (0.017)Weight - (0.323)Biceps + Forearm |
| **La\_Wrist** | - (0.012)Weight - (0.165)Neck + Wrist |
| **La\_BMI** | - (0.111)Weight + BMI |

## Latent Variable Interpretation

The ILAA returns the Unit Preserving Linear Transformation Matrix (UPLTM). This specific transformation is the combination of statistically significant linear association analysis between feature pairs. Each significant association is modeled by a linear equation; henceforth, the interpretation of each feature is as follows:

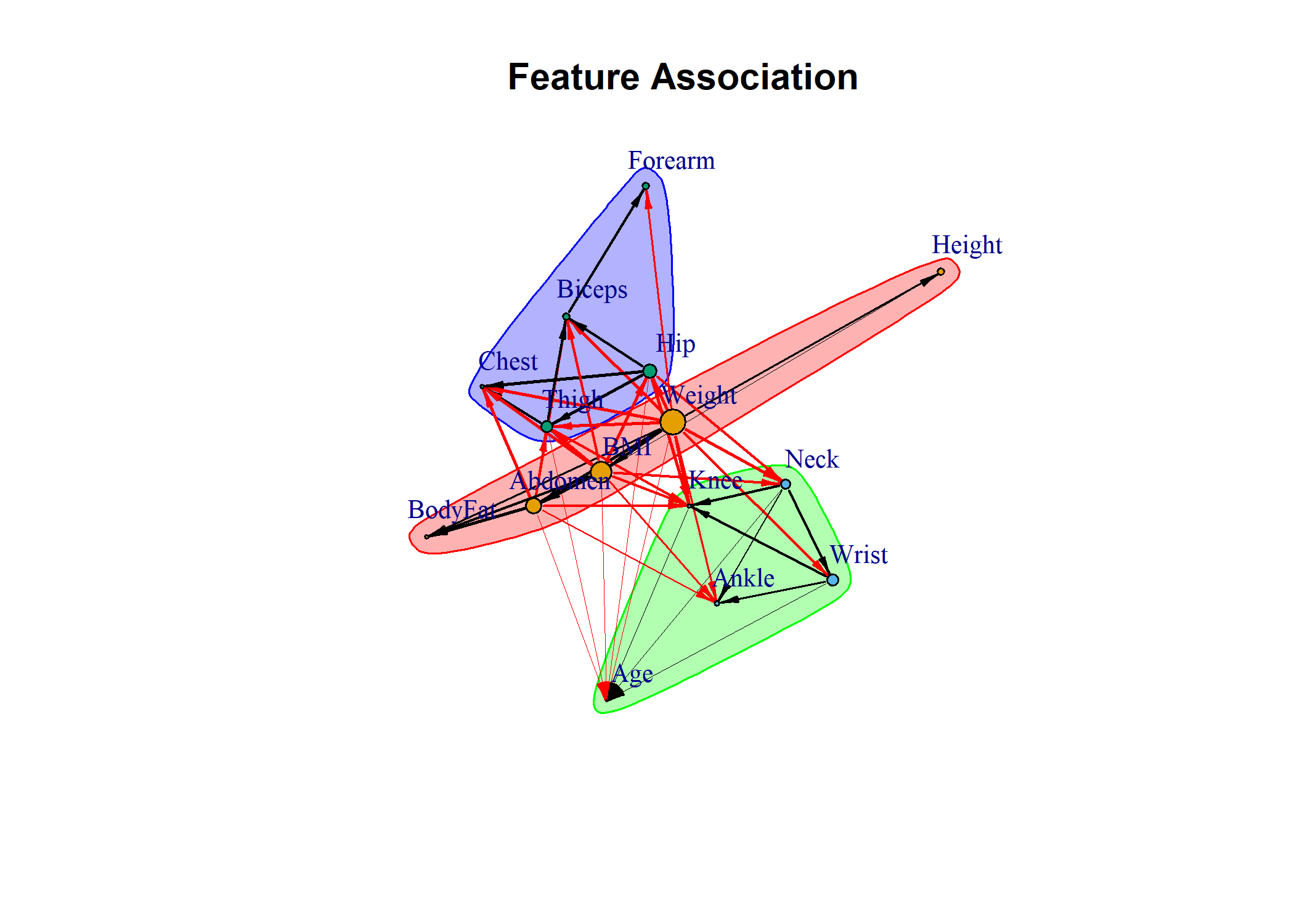
* Each discovered latent variable is the residual of the observed parent variable *vs.* the suitable model of the variables associated with the parent variable. For example:
* Describes that the is associated with the and . The latent variable is the amount of information in the not found by nor the .
* Therefore, the model of the is :

where is the bias term. It can be estimated using the difference between the mean of the raw observations and the mean of the model.

## The Formula Network

The graph\_from\_adjacency\_matrix() function from igraph can be used to visualize the association between variables.

par(op)  
  
transform <- attr(body\_fat\_Decorrelated,"UPLTM") != 0  
colnames(transform) <- str\_remove\_all(colnames(transform),"La\_")  
transform <- abs(transform\*cor(body\_fat[,rownames(transform)])) # The weights are proportional to the observed correlation  
  
  
VertexSize <- attr(body\_fat\_Decorrelated,"fscore") # The size depends on the variable independence relevance (fscore)  
names(VertexSize) <- str\_remove\_all(names(VertexSize),"La\_")  
VertexSize <- 10\*(VertexSize-min(VertexSize))/(max(VertexSize)-min(VertexSize)) # Normalization  
  
  
gr <- graph\_from\_adjacency\_matrix(transform,mode = "directed",diag = FALSE,weighted=TRUE)  
gr$layout <- layout\_with\_fr  
  
# The user can use any cluster method. Here we use the optimal clustering.  
fc <- cluster\_optimal(gr)  
plot(fc, gr,  
 edge.width=2\*E(gr)$weight,  
 edge.arrow.size=0.5,  
 edge.arrow.width=0.5,  
 vertex.size=VertexSize,  
 vertex.label.cex=0.85,  
 vertex.label.dist=2,  
 main="Feature Association")



par(op)

### ILAA Solution and Perturbations

ILLA solutions depends on the observed data. The provided function can add data perturbations aiming to improve the sensitivity to find multicollinearity issues.

#### Bootstrapping ILLA

To handle the data sensitivity to the input data, ILAA allows for bootstrapping estimation of the transformation matrix.

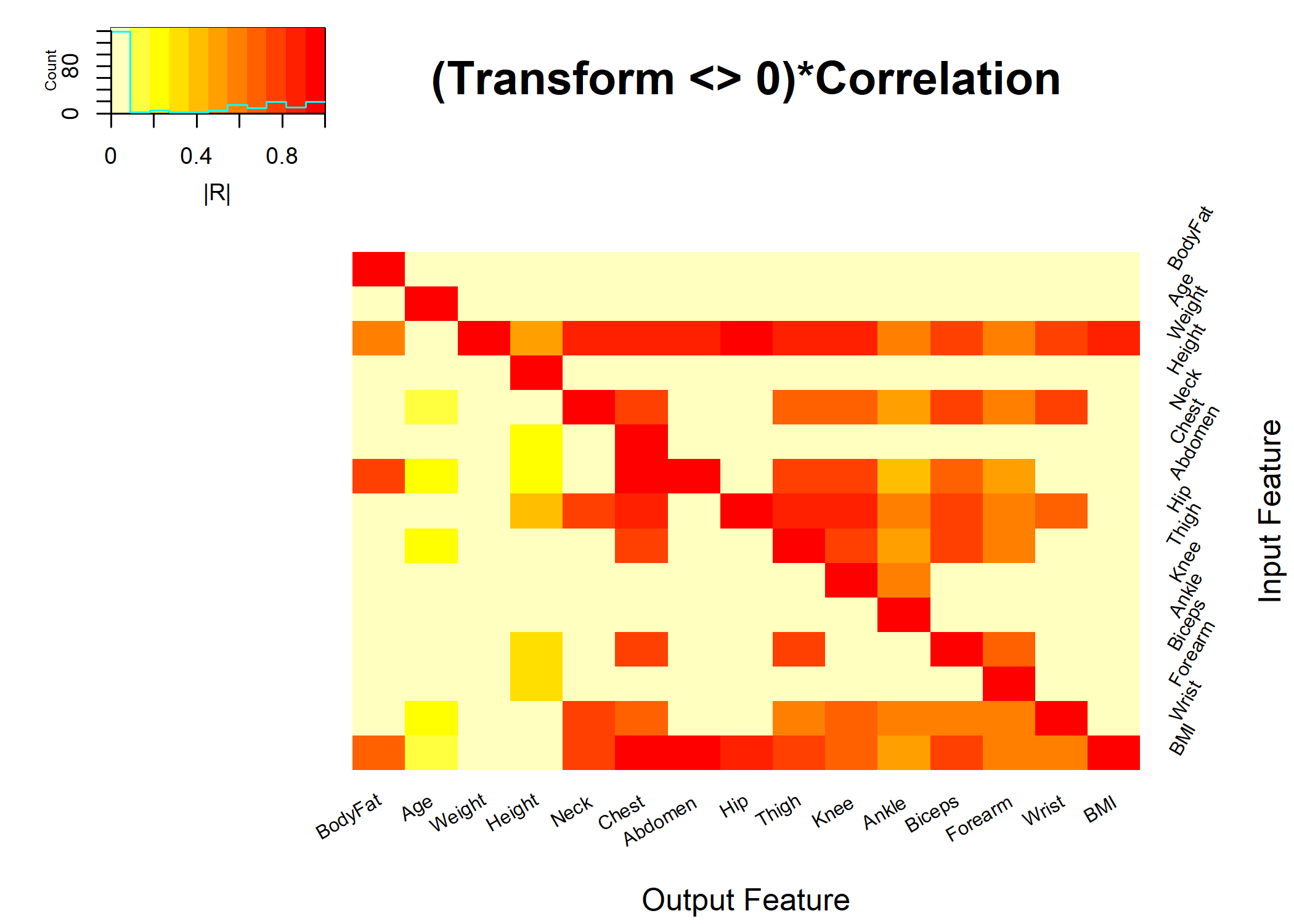
## Here we petrubate only 5% of the data  
body\_fat\_Decorrelated <- ILAA(body\_fat,thr=0.2,bootstrap=100)  
  
  
pander::pander(attr(body\_fat\_Decorrelated,"VarRatio"))

| Weight | La\_Ankle | La\_Forearm | La\_Age | La\_Wrist | La\_Biceps | La\_BodyFat | La\_Neck | La\_Knee | La\_BMI | La\_Thigh | La\_Abdomen | La\_Chest | La\_Hip | La\_Height |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.557 | 0.519 | 0.482 | 0.403 | 0.321 | 0.29 | 0.275 | 0.239 | 0.211 | 0.183 | 0.131 | 0.103 | 0.0993 | 0.0209 |

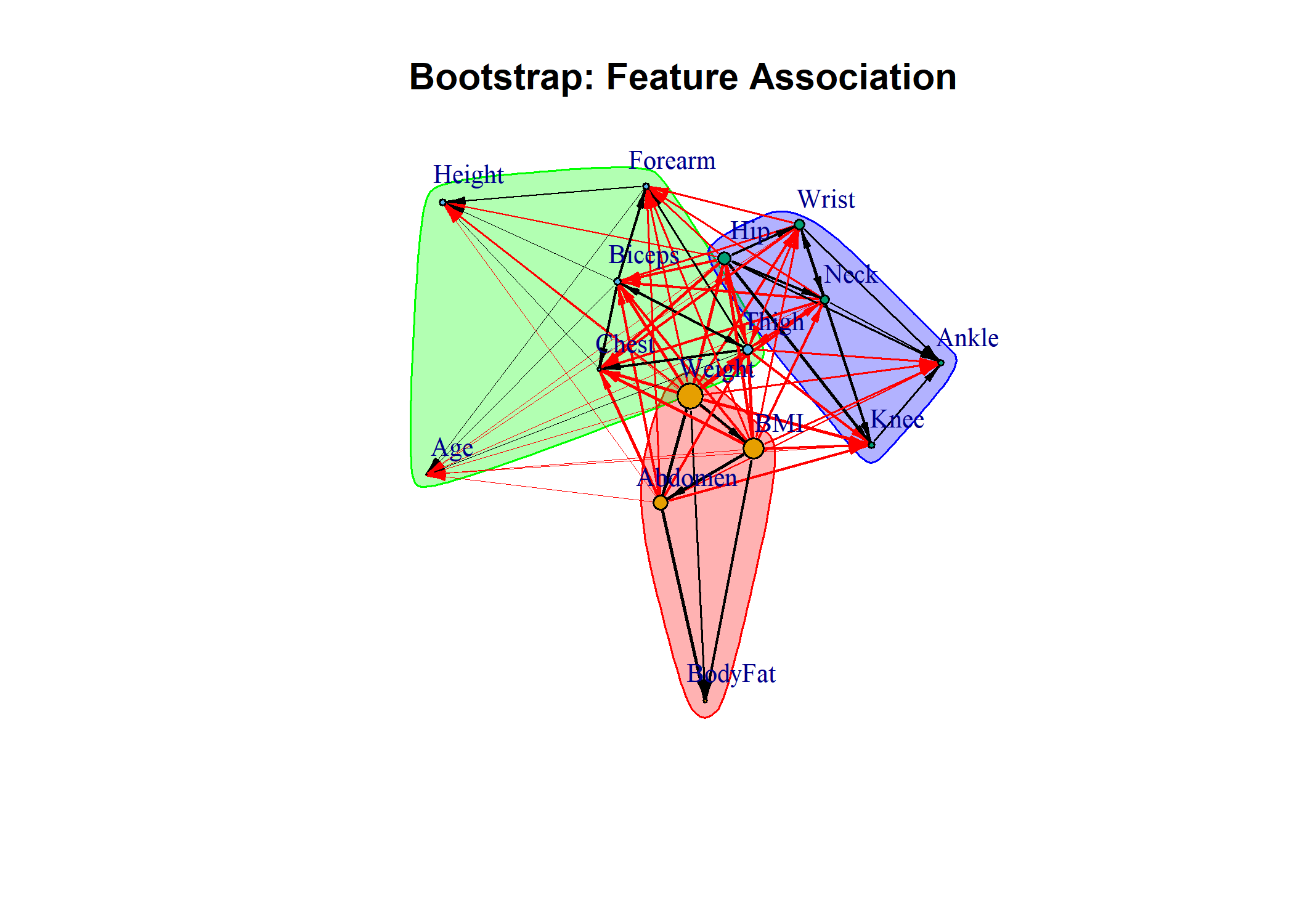
## Getting the formulas  
LatentFormulas <- getLatentCoefficients(body\_fat\_Decorrelated)  
charFormulas <- attr(LatentFormulas,"LatentCharFormulas")  
pander::pander(as.matrix(charFormulas))

|  |  |
| --- | --- |
| **La\_BodyFat** | + BodyFat + (0.121)Weight - (0.799)Abdomen - (0.482)BMI |
| **La\_Age** | + Age + (0.381)Weight - (0.591)Neck - (1.135)Abdomen - (0.115)Hip + (2.157)Thigh - (1.460)Knee - (3.90e-03)Biceps + (0.012)Forearm - (5.553)Wrist - (0.662)BMI |
| **La\_Height** | - (0.192)Weight + Height - (1.13e-04)Chest + (3.59e-05)Abdomen - (5.97e-05)Hip + (6.41e-05)Biceps - (1.90e-04)Forearm + (1.341)BMI |
| **La\_Neck** | - (0.099)Weight + Neck + (0.173)Hip - (0.024)Wrist - (0.085)BMI |
| **La\_Chest** | - (0.139)Weight - (1.45e-04)Neck + Chest - (0.370)Abdomen + (0.439)Hip + (0.206)Thigh - (4.56e-04)Biceps + (8.63e-04)Wrist - (1.036)BMI |
| **La\_Abdomen** | - (0.097)Weight + Abdomen - (1.863)BMI |
| **La\_Hip** | - (0.180)Weight + Hip - (0.431)BMI |
| **La\_Thigh** | - (0.056)Weight - (4.53e-04)Neck + (0.136)Abdomen - (0.488)Hip + Thigh - (2.77e-03)Biceps + (2.70e-03)Wrist - (0.254)BMI |
| **La\_Knee** | - (0.062)Weight + (0.038)Neck - (8.71e-03)Abdomen - (0.028)Hip - (0.063)Thigh + Knee - (0.232)Wrist + (0.120)BMI |
| **La\_Ankle** | - (0.033)Weight + (0.094)Neck + (0.051)Abdomen - (2.75e-04)Hip + (3.66e-04)Thigh - (4.53e-03)Knee + Ankle - (0.585)Wrist - (0.094)BMI |
| **La\_Biceps** | - (0.079)Weight + (9.67e-05)Neck + (0.054)Abdomen + (0.102)Hip - (0.208)Thigh + Biceps - (5.76e-04)Wrist - (0.101)BMI |
| **La\_Forearm** | - (0.018)Weight + (3.59e-03)Neck - (6.55e-05)Abdomen - (0.011)Hip + (0.023)Thigh - (0.321)Biceps + Forearm - (0.021)Wrist + (1.20e-04)BMI |
| **La\_Wrist** | - (0.012)Weight - (0.160)Neck + (2.17e-04)Hip + Wrist - (9.93e-05)BMI |
| **La\_BMI** | - (0.110)Weight + BMI |

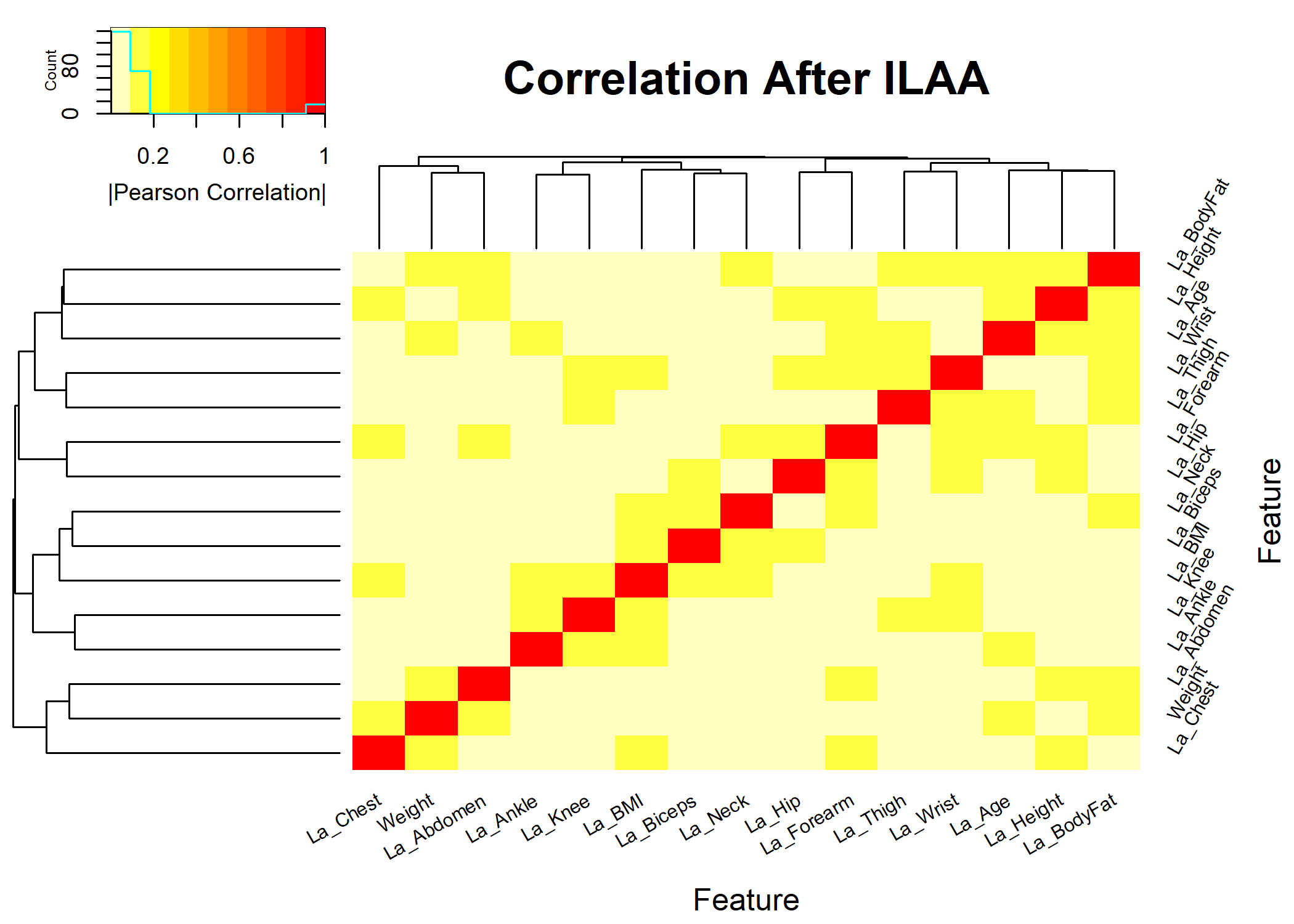
## The transformation  
par(op)  
  
transform <- attr(body\_fat\_Decorrelated,"UPLTM") != 0 # The non-zero coefficients  
colnames(transform) <- str\_remove\_all(colnames(transform),"La\_") # For network analysis  
transform <- abs(transform\*cor(body\_fat[,rownames(transform)])) # The weights are proportional to the observed correlation  
  
  
gplots::heatmap.2(transform,  
 trace = "none",  
 mar = c(5,5),  
 Rowv=NULL,  
 Colv="Rowv",  
 dendrogram="none",  
 col=rev(heat.colors(11)),  
 main = "(Transform <> 0)\*Correlation",  
 cexRow = 0.75,  
 cexCol = 0.75,  
 srtCol=30,  
 srtRow=60,  
 key.title=NA,  
 key.xlab="|R|",  
 xlab="Output Feature", ylab="Input Feature")  
  
 par(op)



## Network analysis  
# The vertex size will be proportional to the fscore of the IDeA procedure.  
   
VertexSize <- attr(body\_fat\_Decorrelated,"fscore") # The size depends on the variable independence relevance (fscore)  
VertexSize <- 10\*(VertexSize-min(VertexSize))/(max(VertexSize)-min(VertexSize)) # Normalization  
  
  
  
gr <- graph\_from\_adjacency\_matrix(transform,mode = "directed",diag = FALSE,weighted=TRUE)  
gr$layout <- layout\_with\_fr  
  
fc <- cluster\_optimal(gr)  
plot(fc, gr,  
 edge.width=2\*E(gr)$weight,  
 edge.arrow.size=0.5,  
 edge.arrow.width=0.5,  
 vertex.size=VertexSize,  
 vertex.label.cex=0.85,  
 vertex.label.dist=2,  
 main="Bootstrap: Feature Association")



par(op)  
  
## Here we plot the final degree of correlation among output features  
 cormat <- cor(body\_fat\_Decorrelated,method="pearson")  
 gplots::heatmap.2(abs(cormat),  
 trace = "none",  
 mar = c(5,5),  
 col=rev(heat.colors(11)),  
 main = "Correlation After ILAA",  
 cexRow = 0.75,  
 cexCol = 0.75,  
 srtCol=30,  
 srtRow=60,  
 key.title=NA,  
 key.xlab="|Pearson Correlation|",  
 xlab="Feature", ylab="Feature")



par(op)  
diag(cormat) <- 0  
pander::pander(max(abs(cormat)))

*0.176*

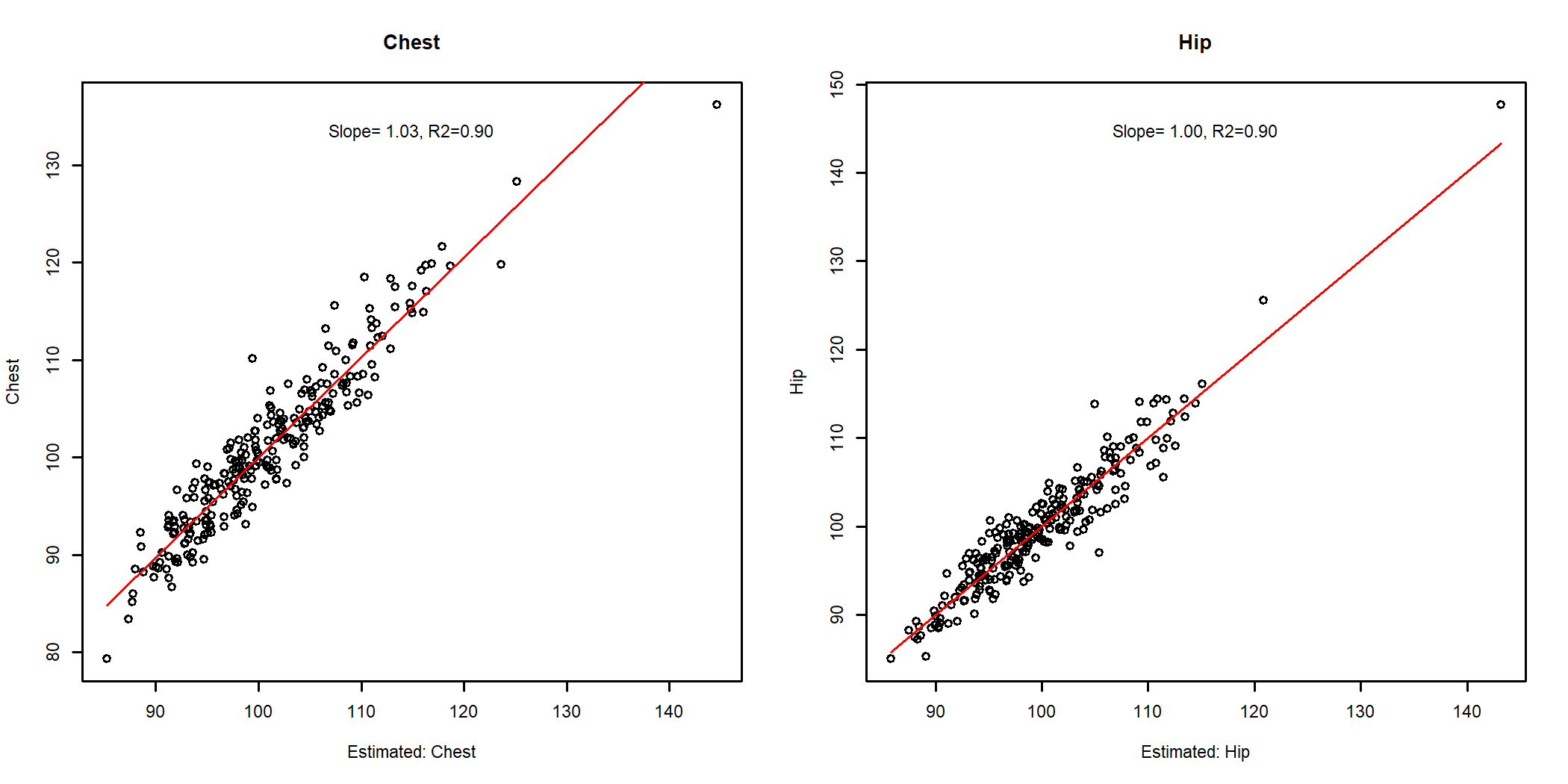
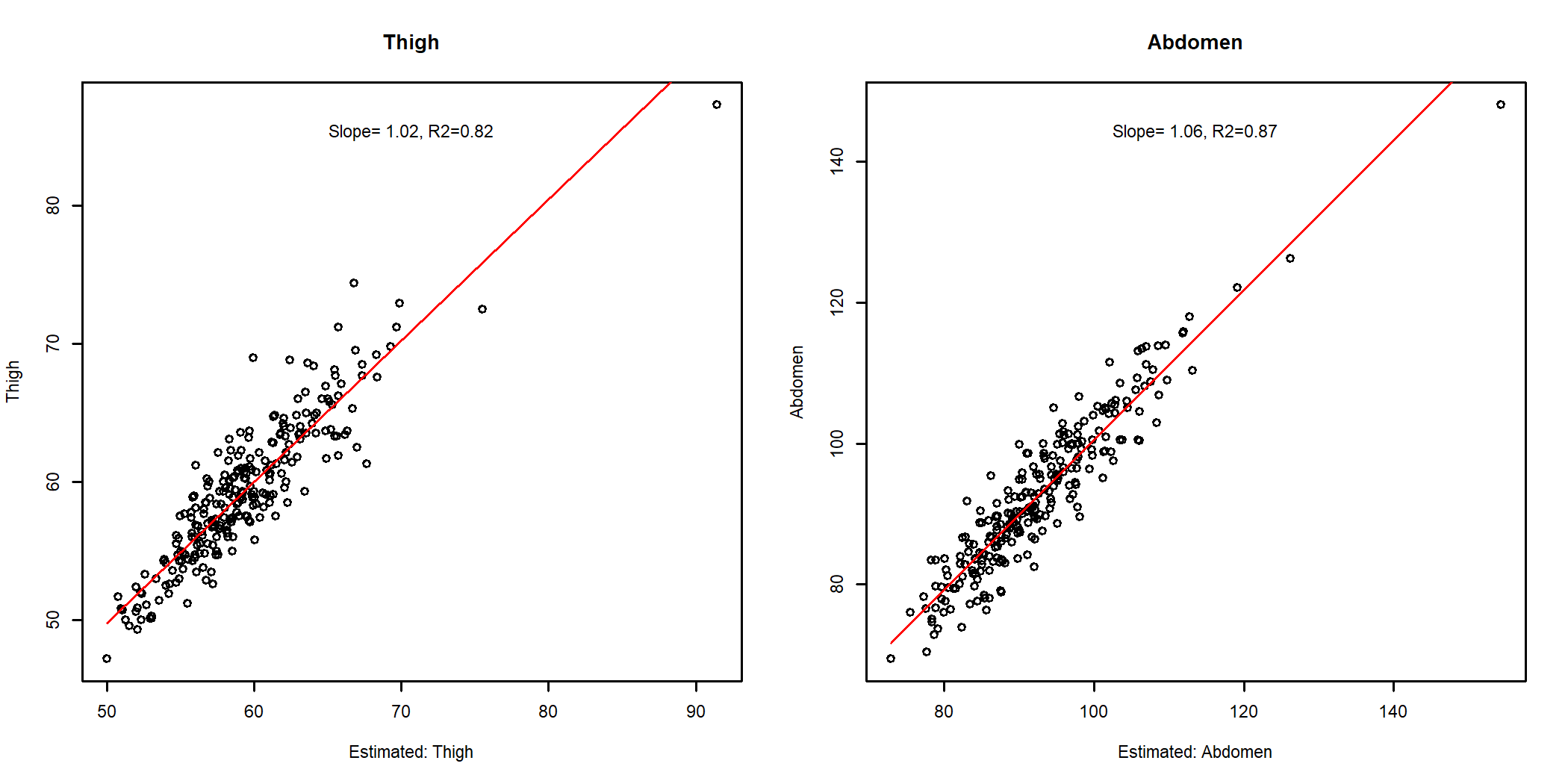
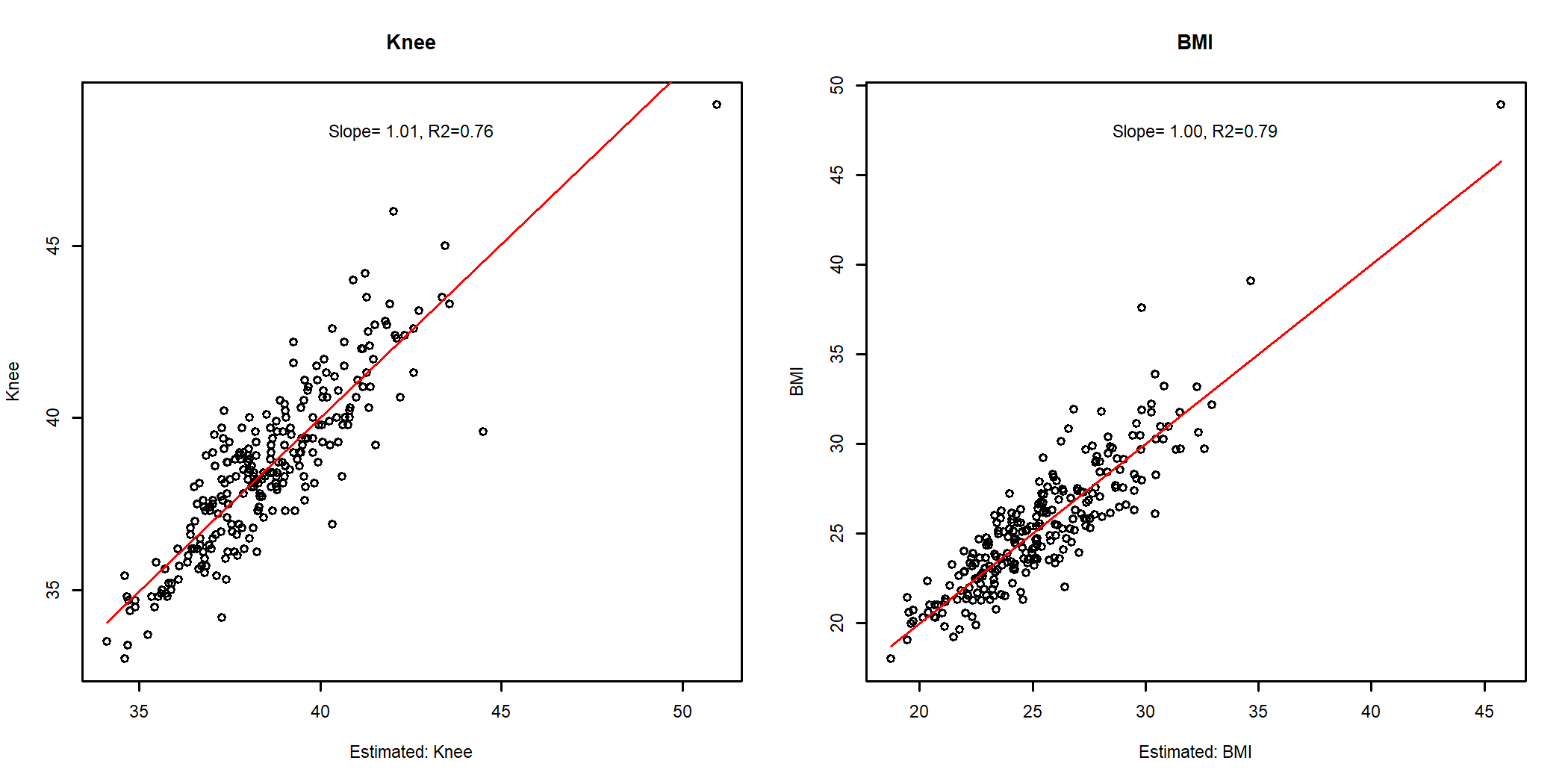
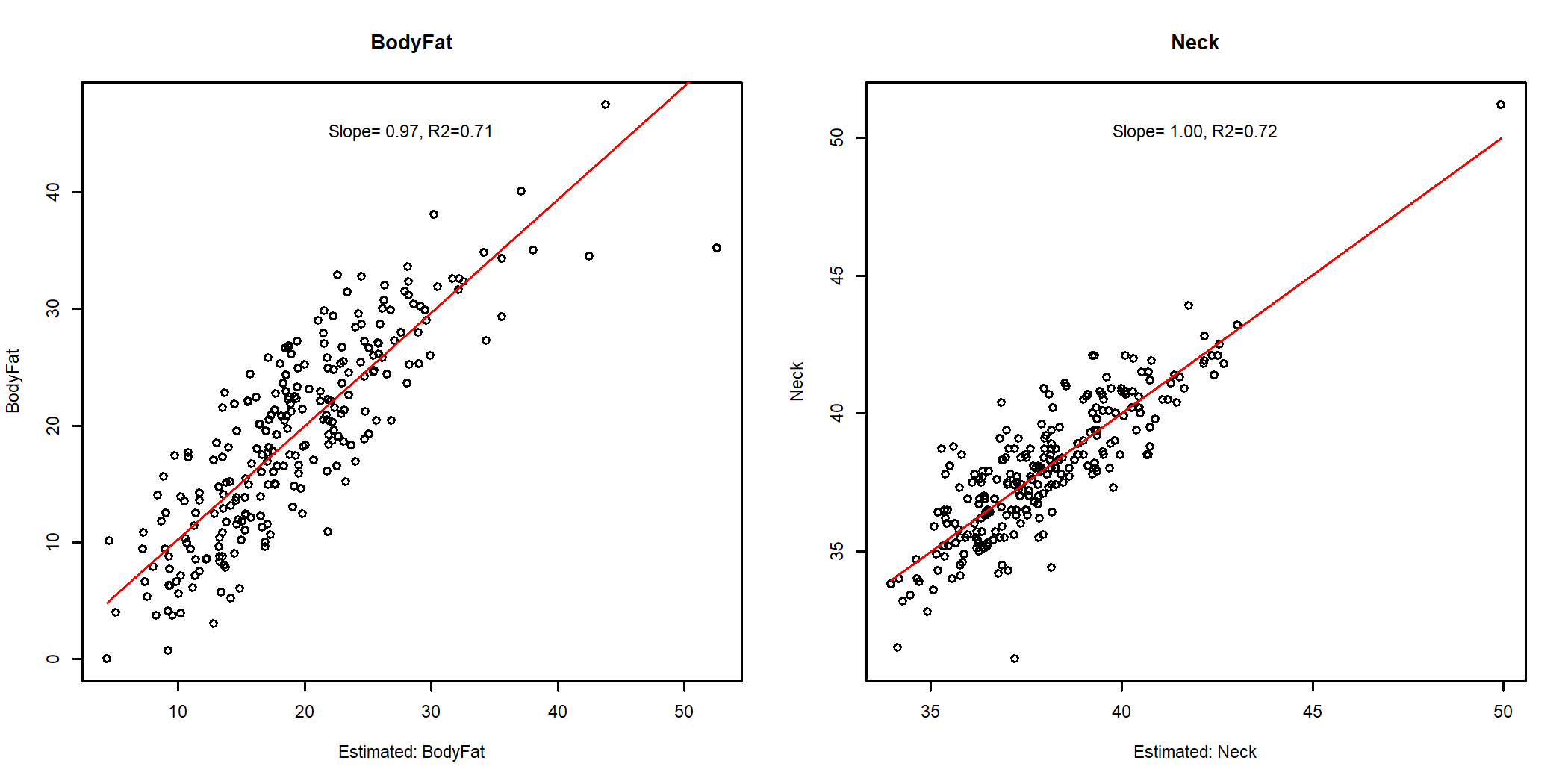
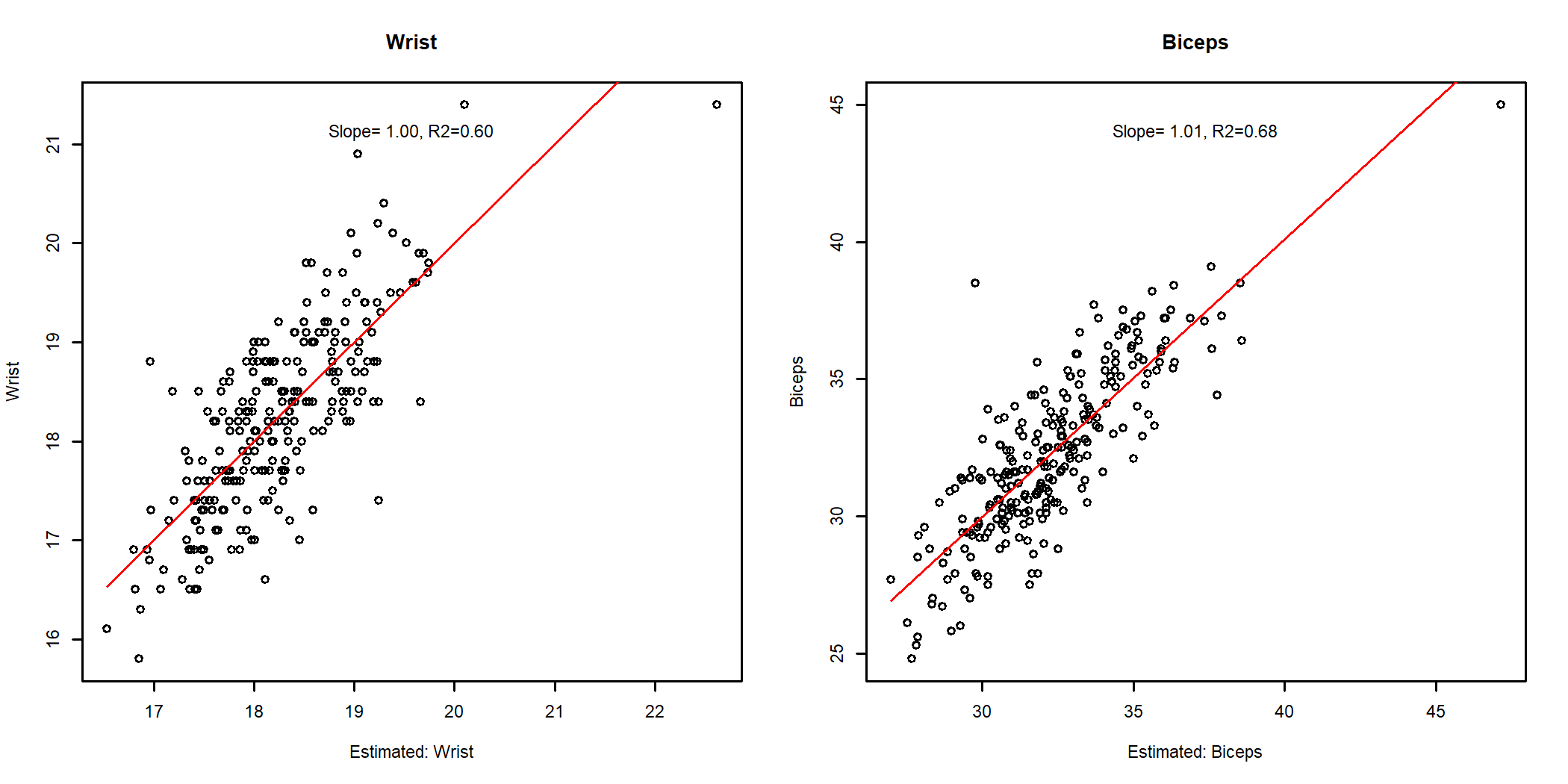
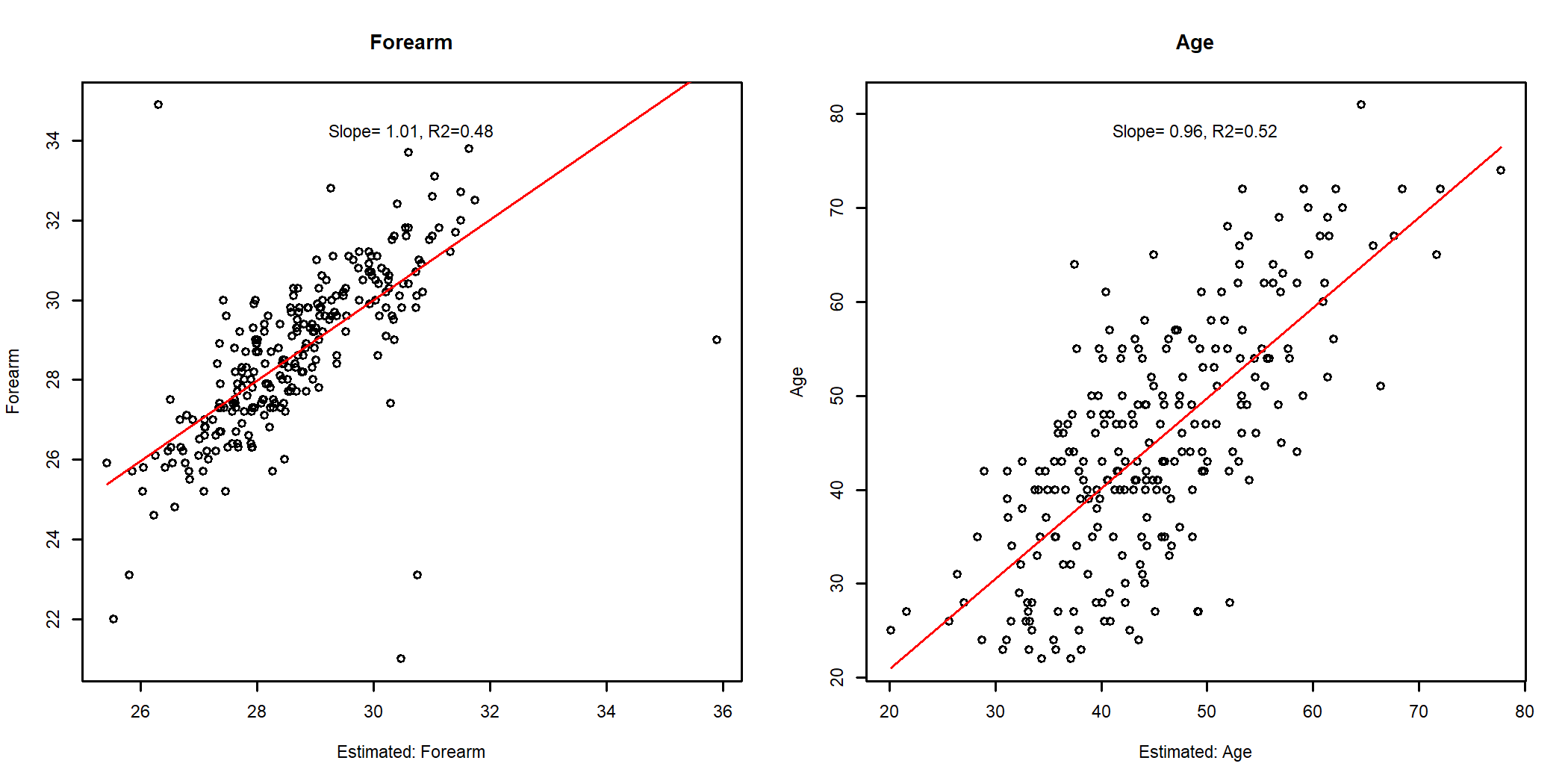
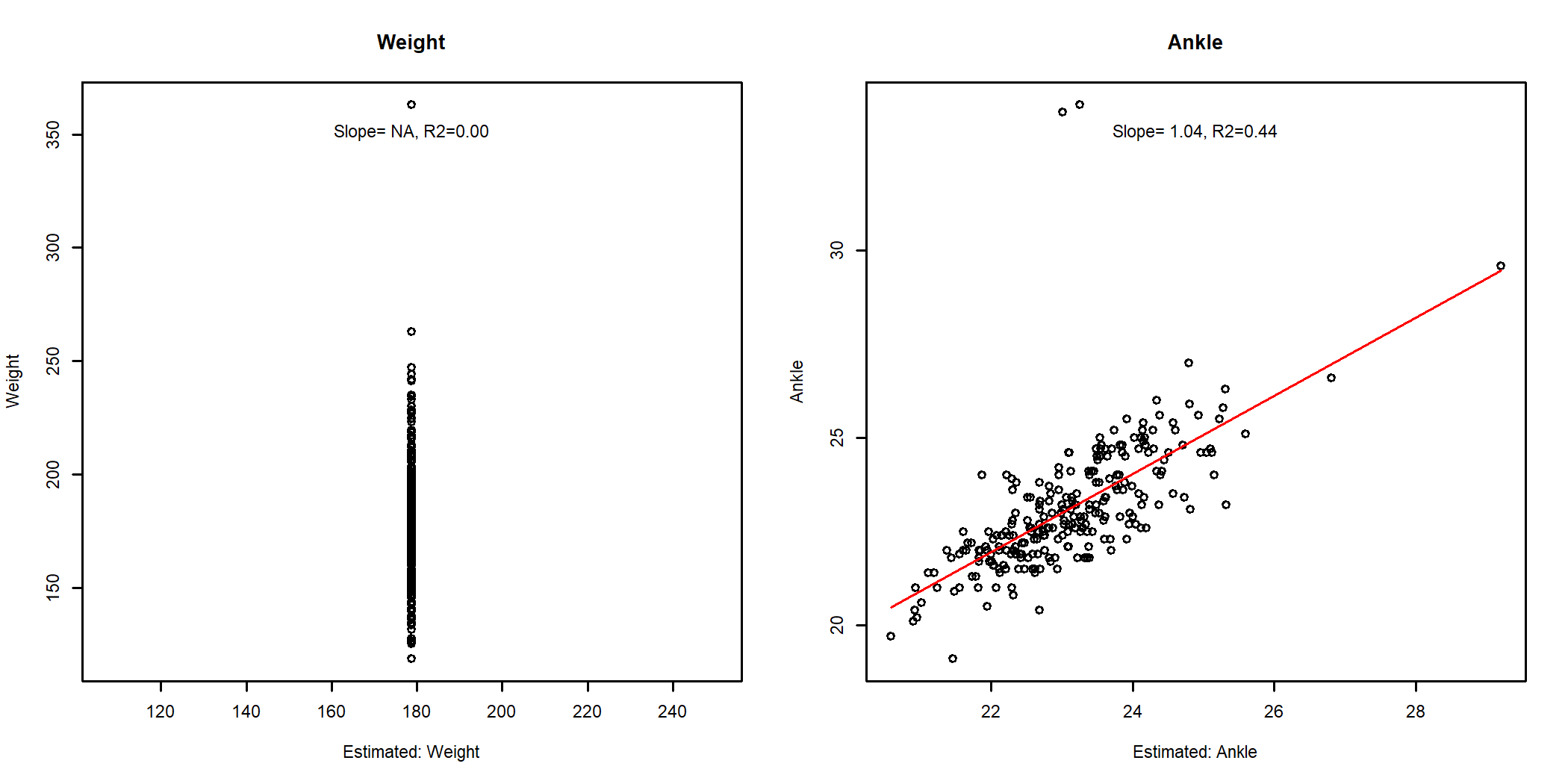
### Association Plots

#### Direct Transform Estimation

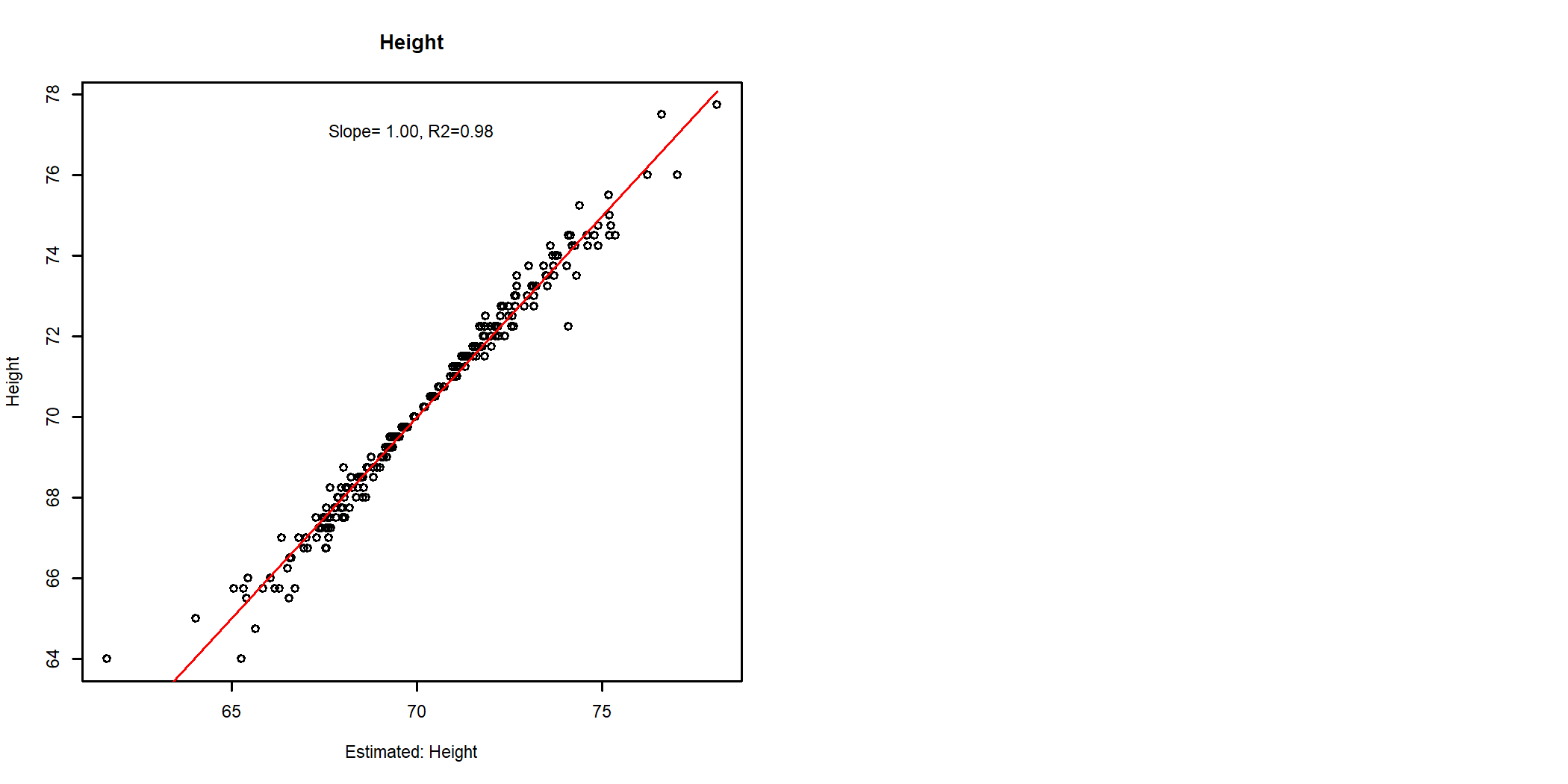
The transformation matrix can be used to get estimation of each variable from the latent models.

To to this just set the diagonal of the transformation to zero, then rotate the input matrix, multiply by -1, and the the output is the estimated observation from the independent variables. The bias term is estimated by computing the observed mean minus the transformed mean.

transform <- attr(body\_fat\_Decorrelated,"UPLTM")  
varratio <- attr(body\_fat\_Decorrelated,"VarRatio")  
# Set the diagonal to zero  
diag(transform) <- 0  
  
#Estimating the observation  
obsestim <- -1\*as.data.frame(as.matrix(body\_fat[,rownames(transform)]) %\*% transform)  
  
#Bias estimation  
bias <- apply(body\_fat[,rownames(transform)],2,mean) - apply(obsestim[,colnames(transform)],2,mean)  
  
#Plotting  
par(mfrow=c(1,2),cex=0.45)  
for (vn in names(varratio))  
{  
 oname <- str\_remove\_all(vn,"La\_")  
 plot(obsestim[,vn] + bias[oname],body\_fat[,oname],xlab=paste("Estimated:",oname),ylab=oname,main=oname)  
 indx <- obsestim[,vn]+bias[oname]  
 lmtvals <- lm(body\_fat[,oname] ~ indx )  
 xvals <- c(min(obsestim[,vn]+ bias[oname]),max(obsestim[,vn]+ bias[oname]))   
 pred <- lmtvals$coefficients[1] + lmtvals$coefficients[2] \* xvals  
 lines(x=xvals,y=pred,col="red")  
 ylim <- c(min(body\_fat[,oname]),max(body\_fat[,oname]))  
  
 text(xvals[1]+(xvals[2]-xvals[1])/2,0.95\*(ylim[2]-ylim[1])+ylim[1],  
 sprintf("Slope= %.2f, R2=%3.2f",lmtvals$coefficients[2],1.0-varratio[vn])  
 )  
}



par(op)



The visual inspection of the above-displayed figures shows that some latent variables are not associated with the original parent variable, but their model is fully correlated to the observed parent variable. A clear example is the last plot.

# ILAA for Supervised Learning

The rerecorded use of ILAA transformation in supervised learning is to split the data into training and validation sets. Henceforth, the next lines of code will split the data into training (75%) and testing (25%)

## Split into Training Testing Sets

# 75% for training 25% for testing   
set.seed(2)  
trainsamples <- sample(nrow(body\_fat),3\*nrow(body\_fat)/4)  
  
trainingset <- body\_fat[trainsamples,]  
testingset <- body\_fat[-trainsamples,]

## Data Train Analysis and Prediction of the Test Set

By default, ILAA() transforms are blind to outcome associations. but in supervised learning the user is free to specify a target outcome to drive the shape of the transformation matrix. Outcome-driven transformations try to keep unaltered features strongly associated with the target.

The predictDecorrelate() function can be used to predict any new dataset from an ILAA transformed object.

The next code snippet shows the process of transforming the training set and then using the returned object to transform the testing set using both outcome-blind and outcome-driven transformations.

## Outcome-blind  
body\_fat\_Decorrelated\_train <- ILAA(trainingset,  
 thr=0.2,  
 Outcome="BodyFat")  
pander::pander(attr(body\_fat\_Decorrelated\_train,"drivingFeatures"))

*Weight*, *Hip*, *BMI*, *Chest*, *Abdomen*, *Thigh*, *Knee*, *Neck*, *Biceps*, *Wrist*, *Forearm*, *Ankle*, *Height* and *Age*

body\_fat\_Decorrelated\_test <- predictDecorrelate(body\_fat\_Decorrelated\_train  
 ,testingset)  
  
## Outcome-driven transformation  
body\_fat\_Decorrelated\_trainD <- ILAA(trainingset,  
 thr=0.2,  
 Outcome="BodyFat",  
 drivingFeatures="BodyFat")  
  
pander::pander(attr(body\_fat\_Decorrelated\_trainD,"drivingFeatures"))

*Abdomen*, *BMI*, *Chest*, *Hip*, *Weight*, *Thigh*, *Knee*, *Neck*, *Biceps*, *Forearm*, *Wrist*, *Ankle*, *Age* and *Height*

body\_fat\_Decorrelated\_testD <- predictDecorrelate(body\_fat\_Decorrelated\_trainD  
 ,testingset)

## Train a Regression Model for Body Fat Prediction

Once we have a transformed training and testing set, we can proceed to train a linear model of the body fat content. For this example we will use the LASSO\_MIN() function of the FRESA.CAD package to model the using all the variables in the transformed training set.

## Outcome-Blind  
modelBodyFatRaw <- LASSO\_MIN(BodyFat~.,trainingset)  
pander::pander(as.matrix(modelBodyFatRaw$coef),caption="Raw Coefficients")

Raw Coefficients

|  |  |
| --- | --- |
| **(Intercept)** | 3.04784 |
| **Age** | 0.04975 |
| **Height** | -0.39330 |
| **Neck** | -0.14960 |
| **Chest** | -0.15167 |
| **Abdomen** | 0.80580 |
| **Thigh** | 0.10851 |
| **Ankle** | 0.10517 |
| **Biceps** | 0.13697 |
| **Forearm** | 0.00417 |
| **Wrist** | -1.39800 |

## Outcome-Blind  
modelBodyFat <- LASSO\_MIN(BodyFat~.,body\_fat\_Decorrelated\_train)  
pander::pander(as.matrix(modelBodyFat$coef),caption="Outcome-Blind Coefficients")

Outcome-Blind Coefficients

|  |  |
| --- | --- |
| **(Intercept)** | -56.0734 |
| **La\_Age** | 0.0372 |
| **Weight** | 0.1769 |
| **La\_Height** | 0.5626 |
| **La\_Neck** | -0.2083 |
| **La\_Chest** | 0.1769 |
| **La\_Abdomen** | 0.9393 |
| **La\_Hip** | 0.2382 |
| **La\_Thigh** | 0.1103 |
| **La\_Knee** | 0.0819 |
| **La\_Ankle** | 0.0997 |
| **La\_Biceps** | 0.1537 |
| **La\_Wrist** | -0.9001 |
| **La\_BMI** | 1.9868 |

## Outcome-Driven  
modelBodyFatD <- LASSO\_MIN(BodyFat~.,body\_fat\_Decorrelated\_trainD)  
pander::pander(as.matrix(modelBodyFatD$coef),caption="Outcome-Driven Coefficients")

Outcome-Driven Coefficients

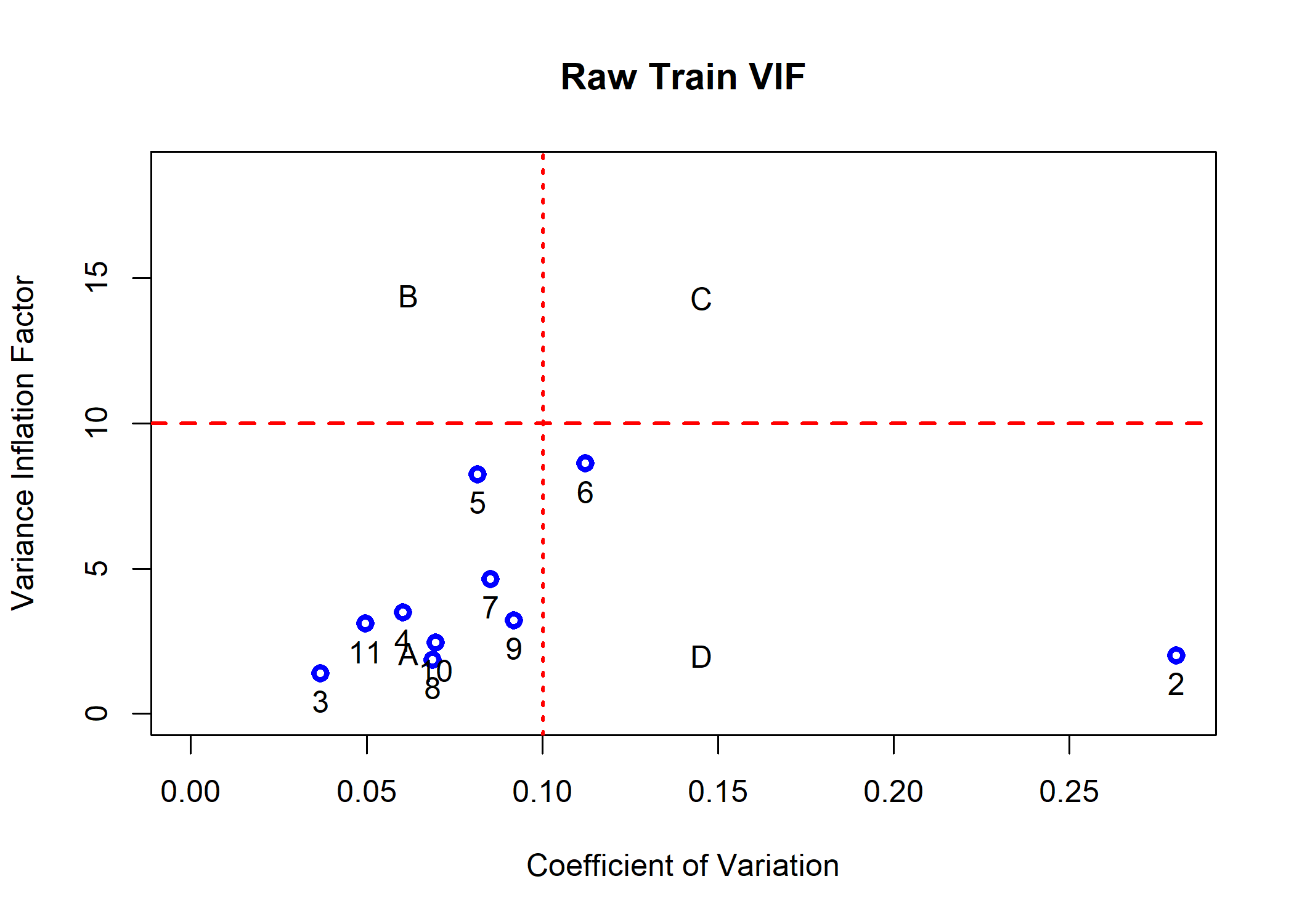
|  |  |
| --- | --- |
| **(Intercept)** | -29.7164 |
| **La\_Age** | 0.0172 |
| **La\_Weight** | -0.0989 |
| **La\_Neck** | -0.5088 |
| **La\_Chest** | -0.1315 |
| **Abdomen** | 0.6441 |
| **La\_Hip** | -0.1942 |
| **La\_Thigh** | 0.1177 |
| **La\_Ankle** | 0.0920 |
| **La\_Biceps** | 0.1394 |
| **La\_Wrist** | -0.5890 |

The printed beta coefficients of the models show that the LASSO models are different between the Outcome-driven and outcome-blind ILAA methods.

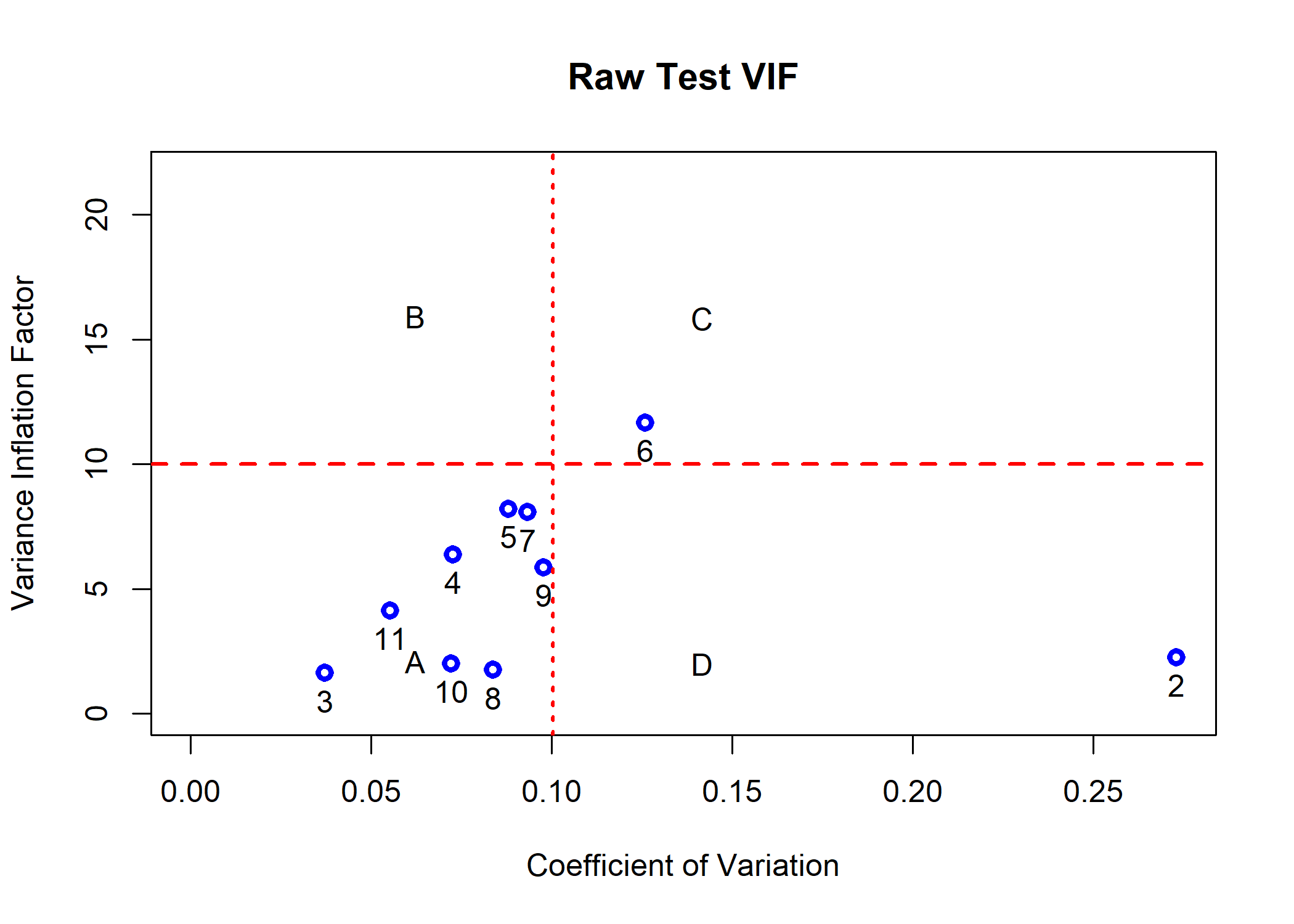
#### Muticollinear Analysis

Here we check the Variance inflation factor (VIF) on the train and testing sets

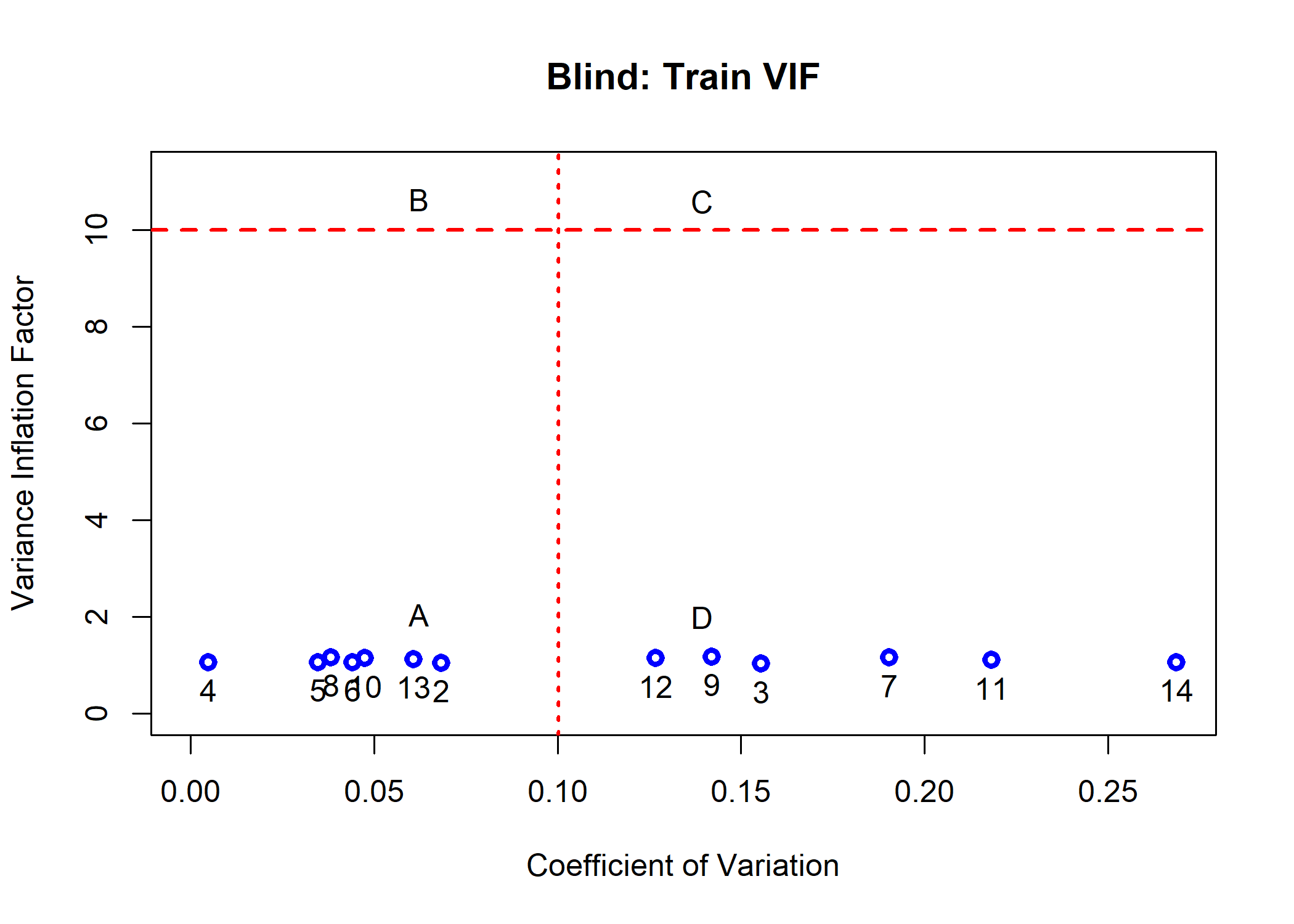
frm <- paste("BodyFat~",str\_flatten(modelBodyFatRaw$selectedfeatures," + "))  
  
X <- model.matrix(formula(frm),trainingset);  
mc <- multiCol(X)  
vifd <- VIF(X)  
vifx <-vif(lm(formula(frm),trainingset))  
title("Raw Train VIF")



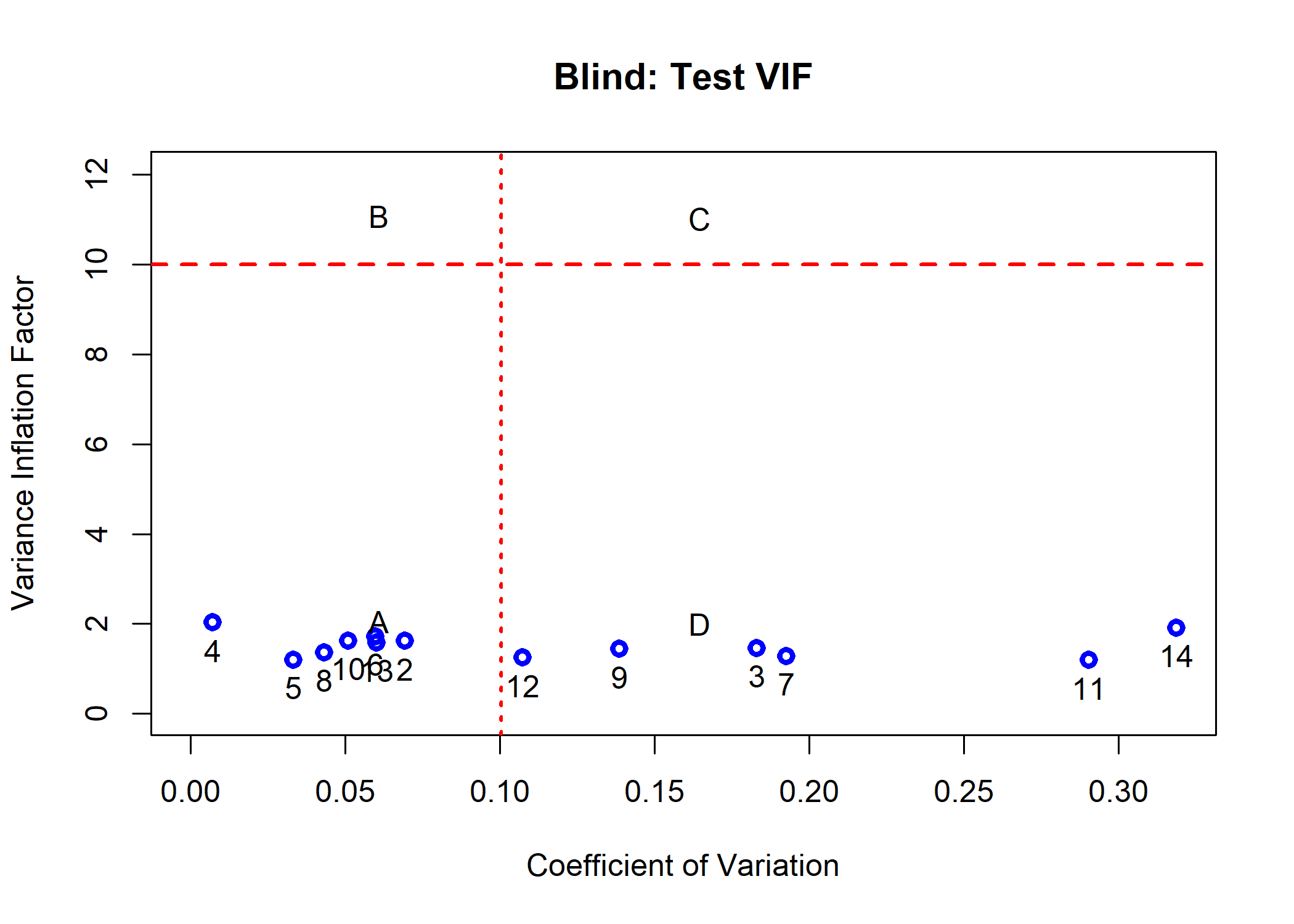
X <- model.matrix(formula(frm),testingset);  
mc <- multiCol(X)  
vifd <- VIF(X)  
vifx <-vif(lm(formula(frm),testingset))  
title("Raw Test VIF")



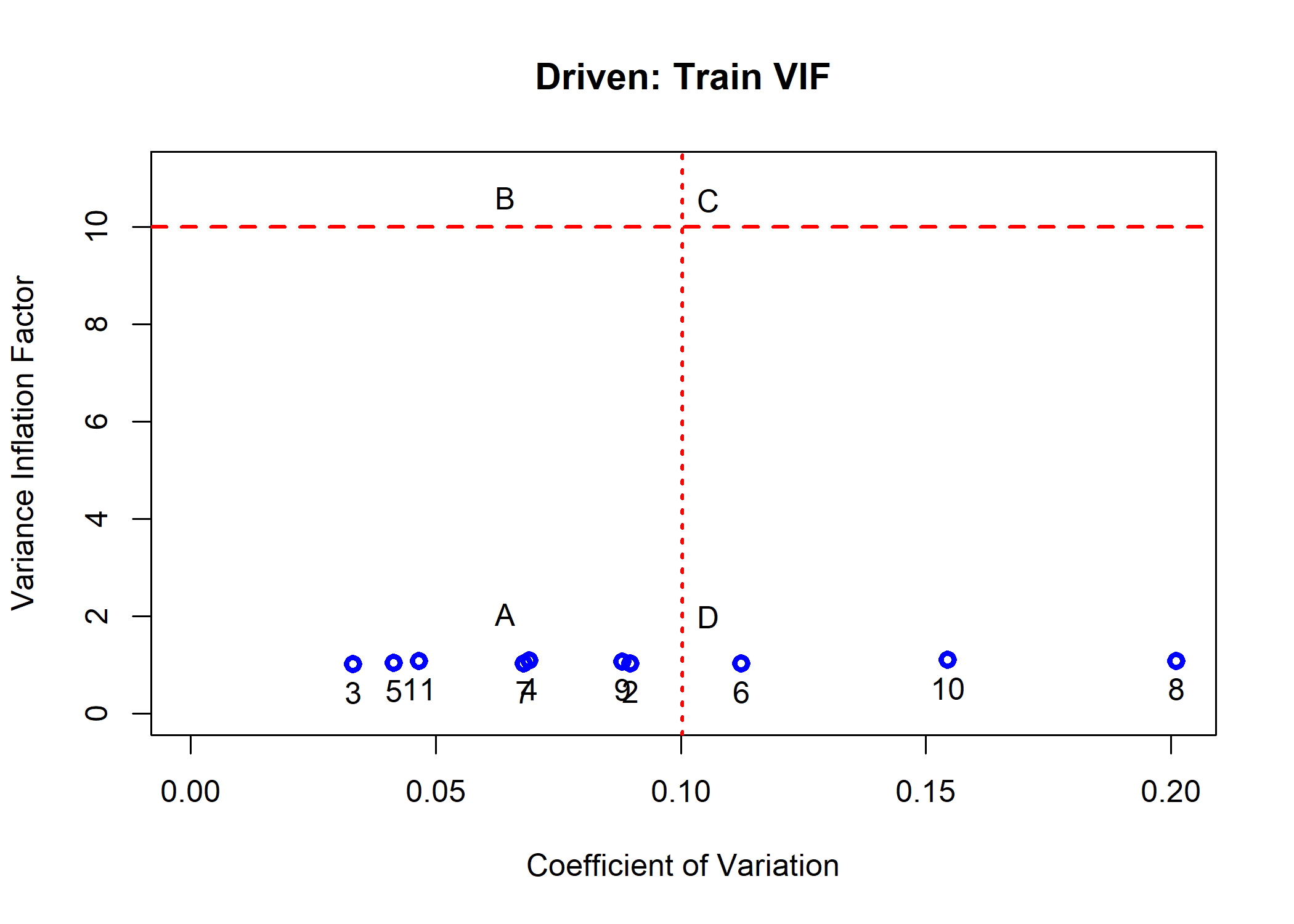
frm <- paste("BodyFat~",str\_flatten(modelBodyFat$selectedfeatures," + "))  
  
X <- model.matrix(formula(frm),body\_fat\_Decorrelated\_train);  
mc <- multiCol(X)  
vifd <- VIF(X)  
vifx <-vif(lm(formula(frm),body\_fat\_Decorrelated\_train))  
title("Blind: Train VIF")



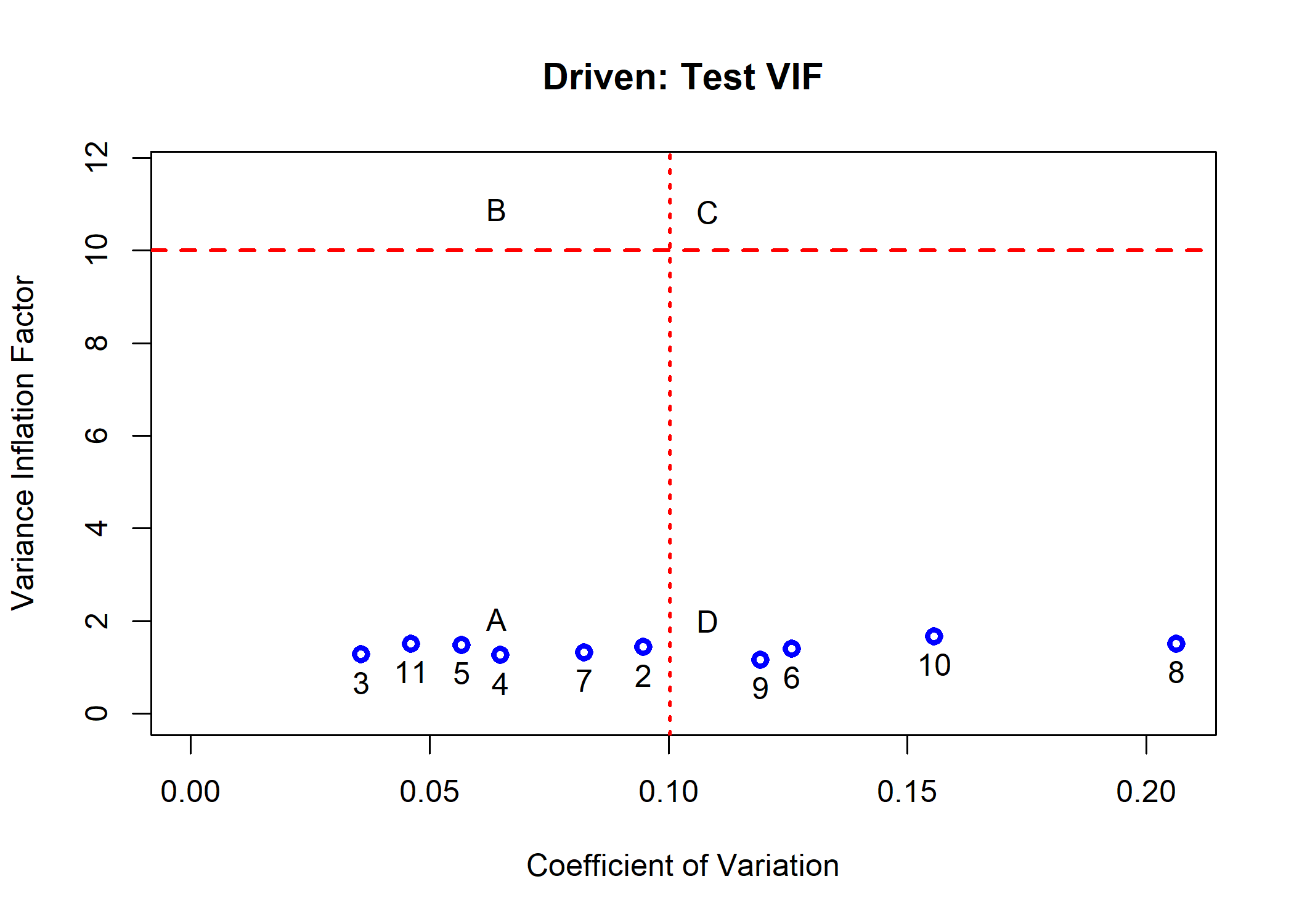
X <- model.matrix(formula(frm),body\_fat\_Decorrelated\_test);  
mc <- multiCol(X)  
title("Blind: Test VIF")



frm <- paste("BodyFat~",str\_flatten(modelBodyFatD$selectedfeatures," + "))  
X <- model.matrix(formula(frm),body\_fat\_Decorrelated\_trainD);  
mc <- multiCol(X)  
title("Driven: Train VIF")



X <- model.matrix(formula(frm),body\_fat\_Decorrelated\_testD);  
  
mc <- multiCol(X)  
title("Driven: Test VIF")



The plots clearly indicate that both models do not have colinearity issues

### The Model Coefficients in the Observed Space

The FRESA.CAD package provides a handy function, getObservedCoef()m to get the linear beta coefficients from the transformed object. The next code shows the procedure.

# Get the coefficients in the observed space for the outcome-blind  
observedCoef <- getObservedCoef(body\_fat\_Decorrelated\_train,modelBodyFat)  
pander::pander(as.matrix(observedCoef$coefficients),caption="Blind Coefficients")

Blind Coefficients

|  |  |
| --- | --- |
| **(Intercept)** | -56.07340 |
| **Age** | 0.03722 |
| **Weight** | -0.18055 |
| **Height** | 0.56259 |
| **Neck** | -0.07292 |
| **Chest** | -0.28360 |
| **Abdomen** | 0.89781 |
| **Hip** | -0.14482 |
| **Thigh** | 0.15616 |
| **Knee** | -0.00665 |
| **Ankle** | 0.09967 |
| **Biceps** | 0.15371 |
| **Wrist** | -1.19584 |
| **BMI** | 1.36021 |

# The outcome-driven coefficients  
observedCoefD <- getObservedCoef(body\_fat\_Decorrelated\_trainD,modelBodyFatD)  
pander::pander(as.matrix(observedCoefD$coefficients),caption="Driven Coefficients")

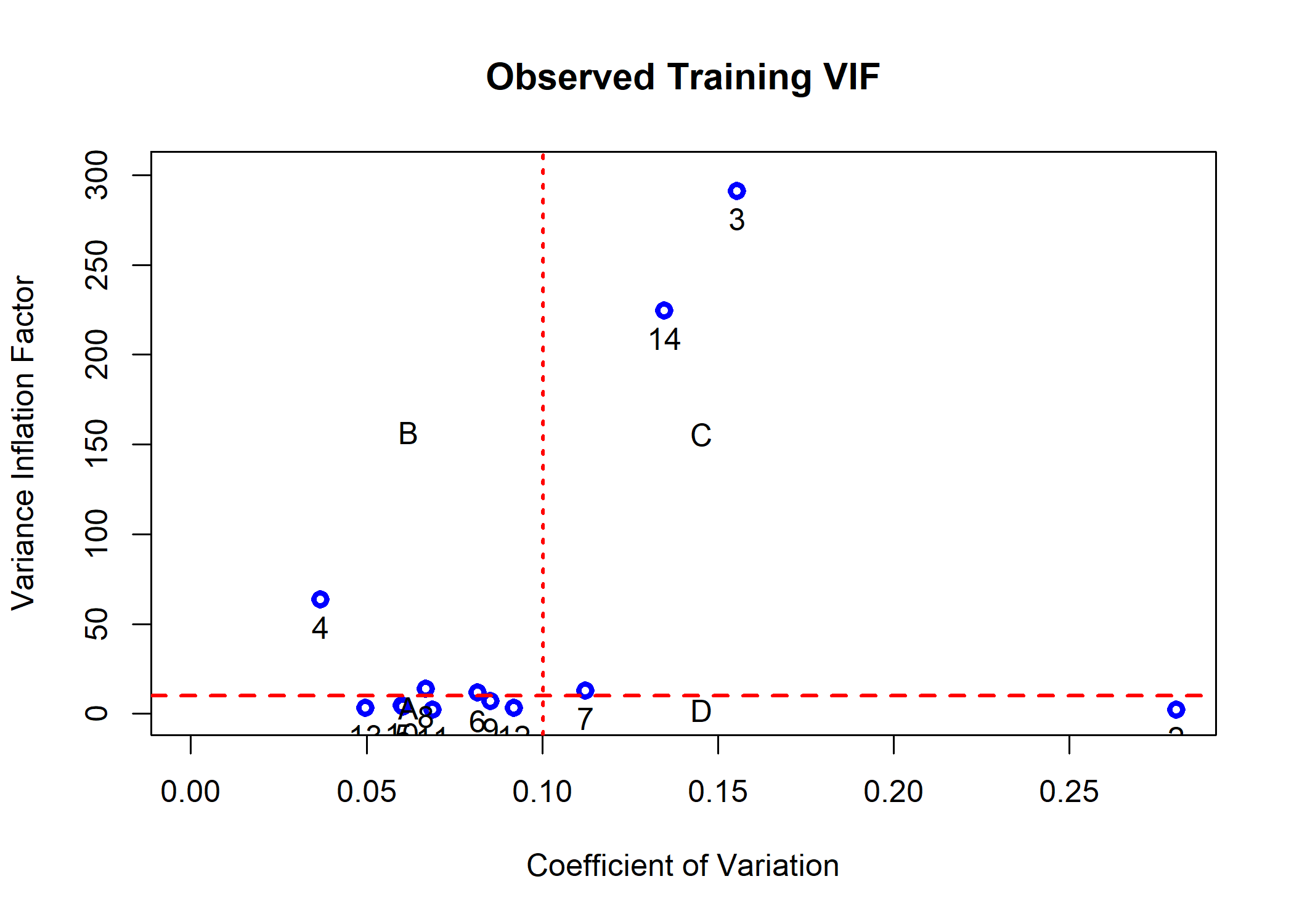
Driven Coefficients

|  |  |
| --- | --- |
| **(Intercept)** | -29.7164 |
| **Age** | 0.0172 |
| **Weight** | -0.0790 |
| **Neck** | -0.1669 |
| **Chest** | -0.1315 |
| **Abdomen** | 0.8662 |
| **Hip** | -0.0162 |
| **Thigh** | 0.1124 |
| **Knee** | -0.0417 |
| **Ankle** | 0.0920 |
| **Biceps** | 0.1394 |
| **Wrist** | -0.7604 |
| **BMI** | 0.2150 |

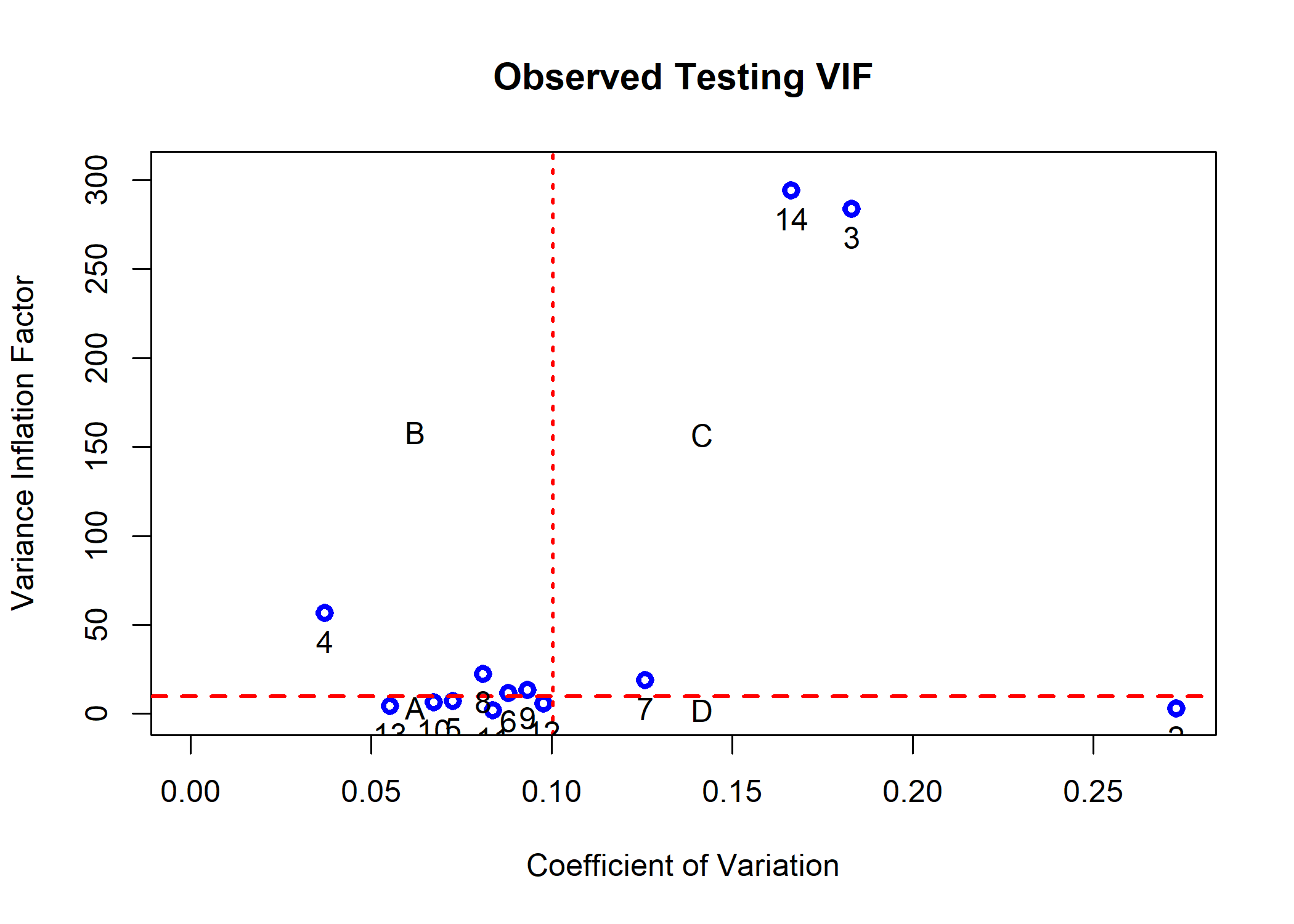
#### Muticollinear Analysis on the observed space

Here we check the Variance inflation factor (VIF) on the train and testing sets using the observed variables

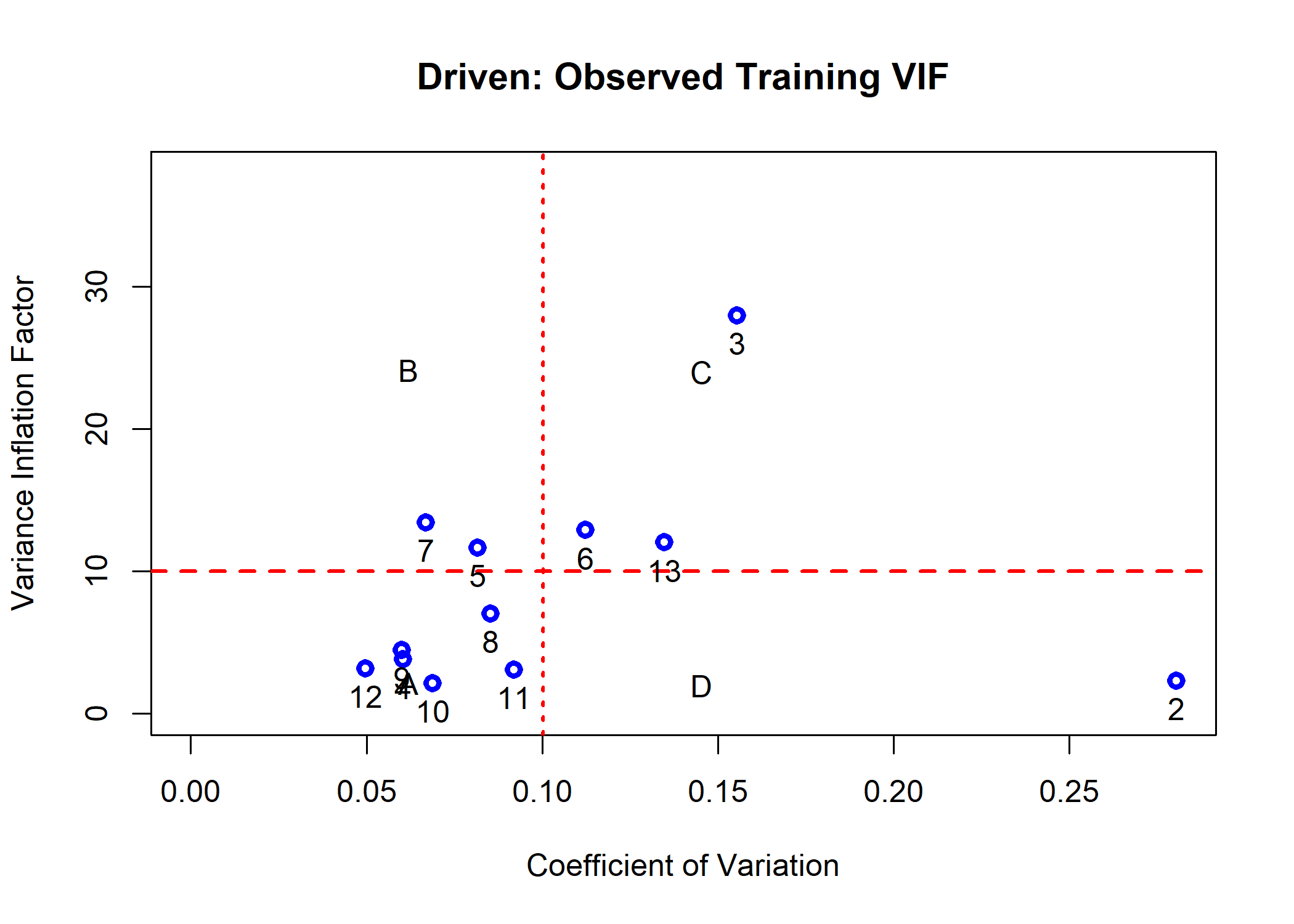
X <- model.matrix(formula(observedCoef$formula),trainingset);  
mc <- multiCol(X)  
title("Observed Training VIF")



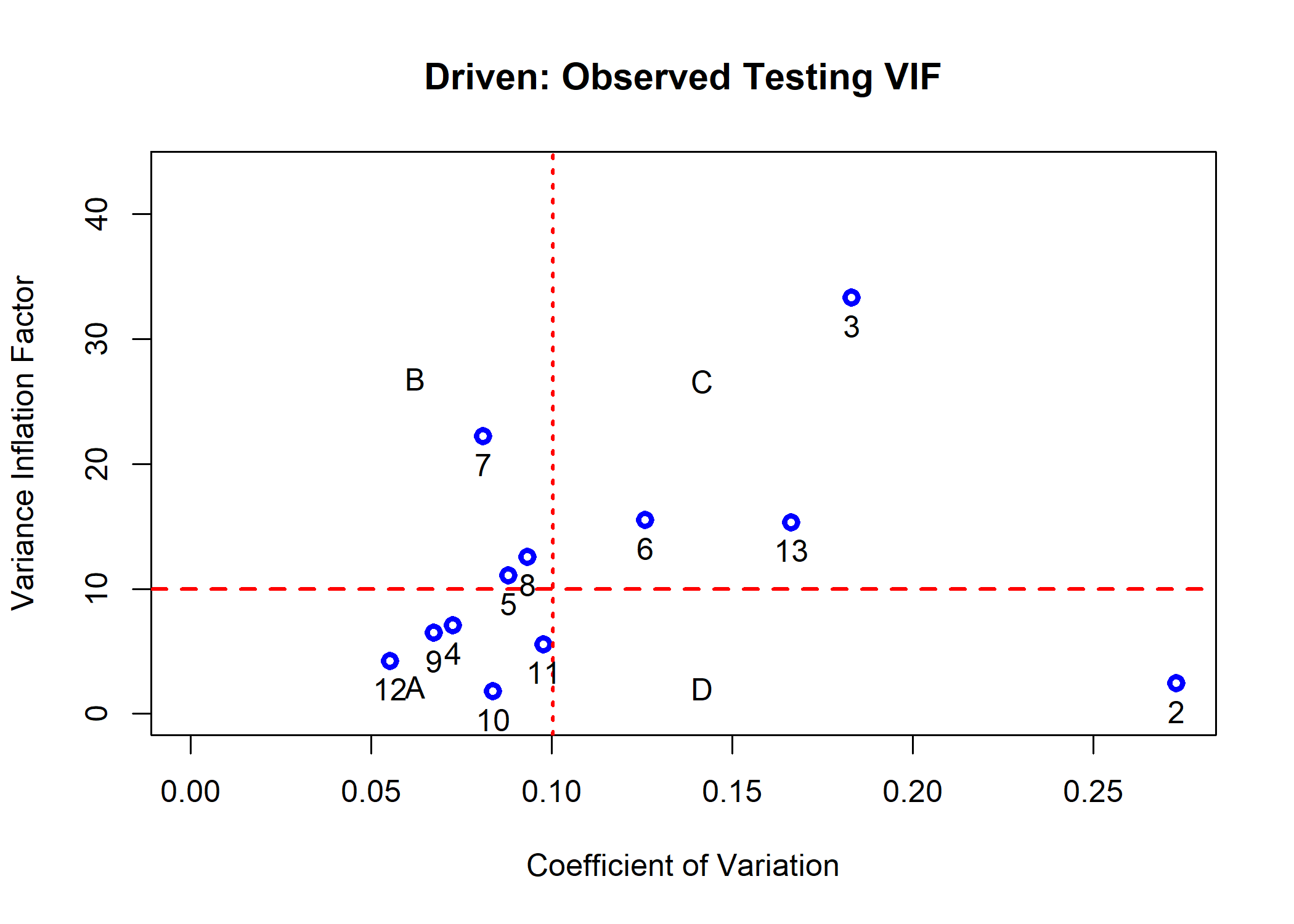
X <- model.matrix(formula(observedCoef$formula),testingset);  
mc <- multiCol(X)  
title("Observed Testing VIF")



X <- model.matrix(formula(observedCoefD$formula),trainingset);  
mc <- multiCol(X)  
title("Driven: Observed Training VIF")



X <- model.matrix(formula(observedCoefD$formula),testingset);  
mc <- multiCol(X)  
title("Driven: Observed Testing VIF")

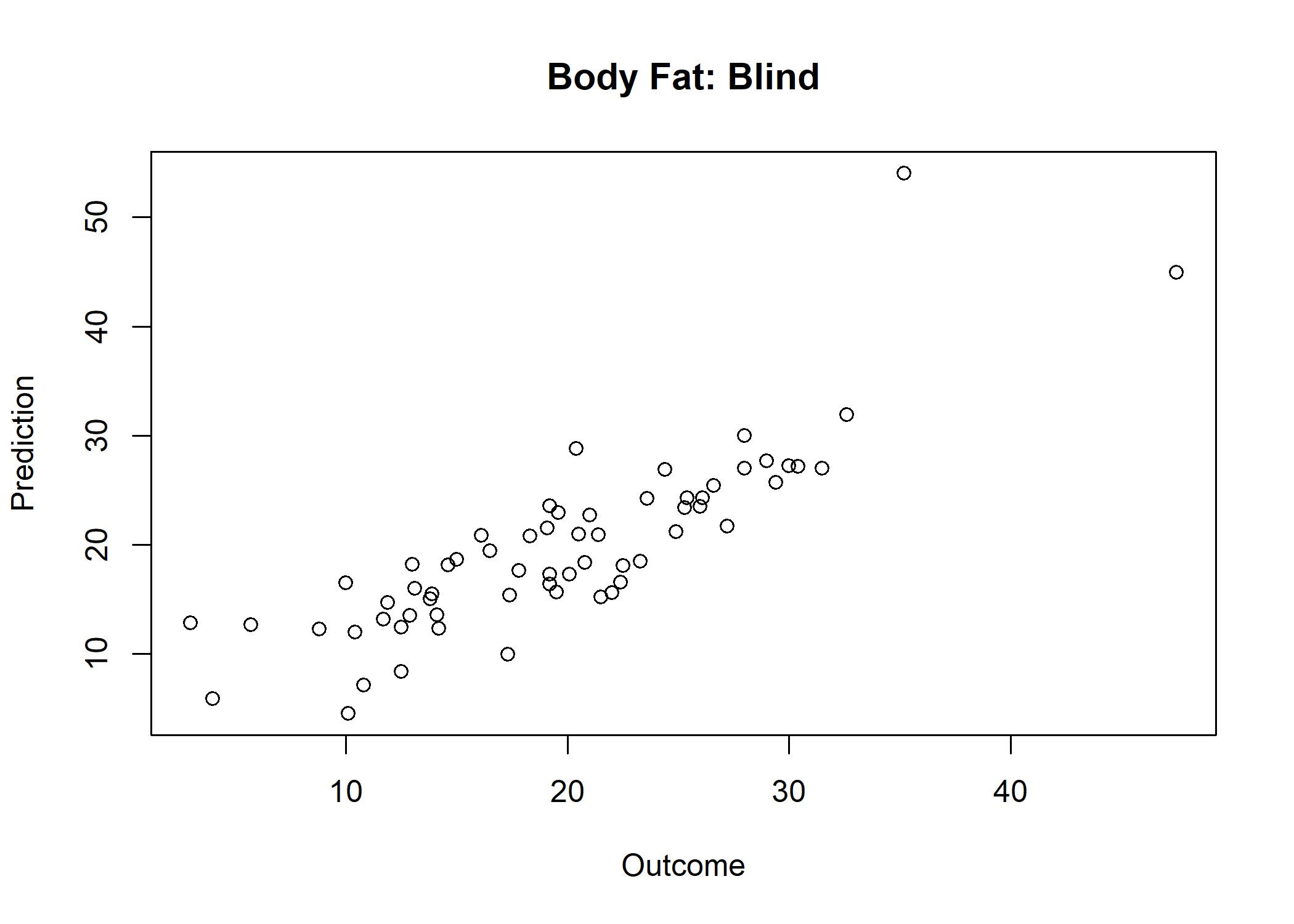


The results indicate that the models created using the observed variables have strong collinearity issues.

### Predict Using the Transformed Data-Set

The user can predict the BodyFat content using the handy predict() function. After that we can measure the testing performance using the predictionStats\_regression() function.

## OUtcome-Blind   
predicBodyFat <- predict(modelBodyFat,body\_fat\_Decorrelated\_test)  
rmetrics <- predictionStats\_regression(cbind(testingset$BodyFat,  
 predicBodyFat),  
 "Body Fat: Blind")

Body Fat: Blind 

pander::pander(rmetrics)

* **corci**:

| cor |  |  |
| --- | --- | --- |
| 0.848 | 0.76 | 0.905 |

* **biasci**: *0.0434*, *-1.0731* and *1.1599*
* **RMSEci**: *4.43*, *3.78* and *5.37*
* **spearmanci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.86 | 0.759 | 0.922 |

* **MAEci**:

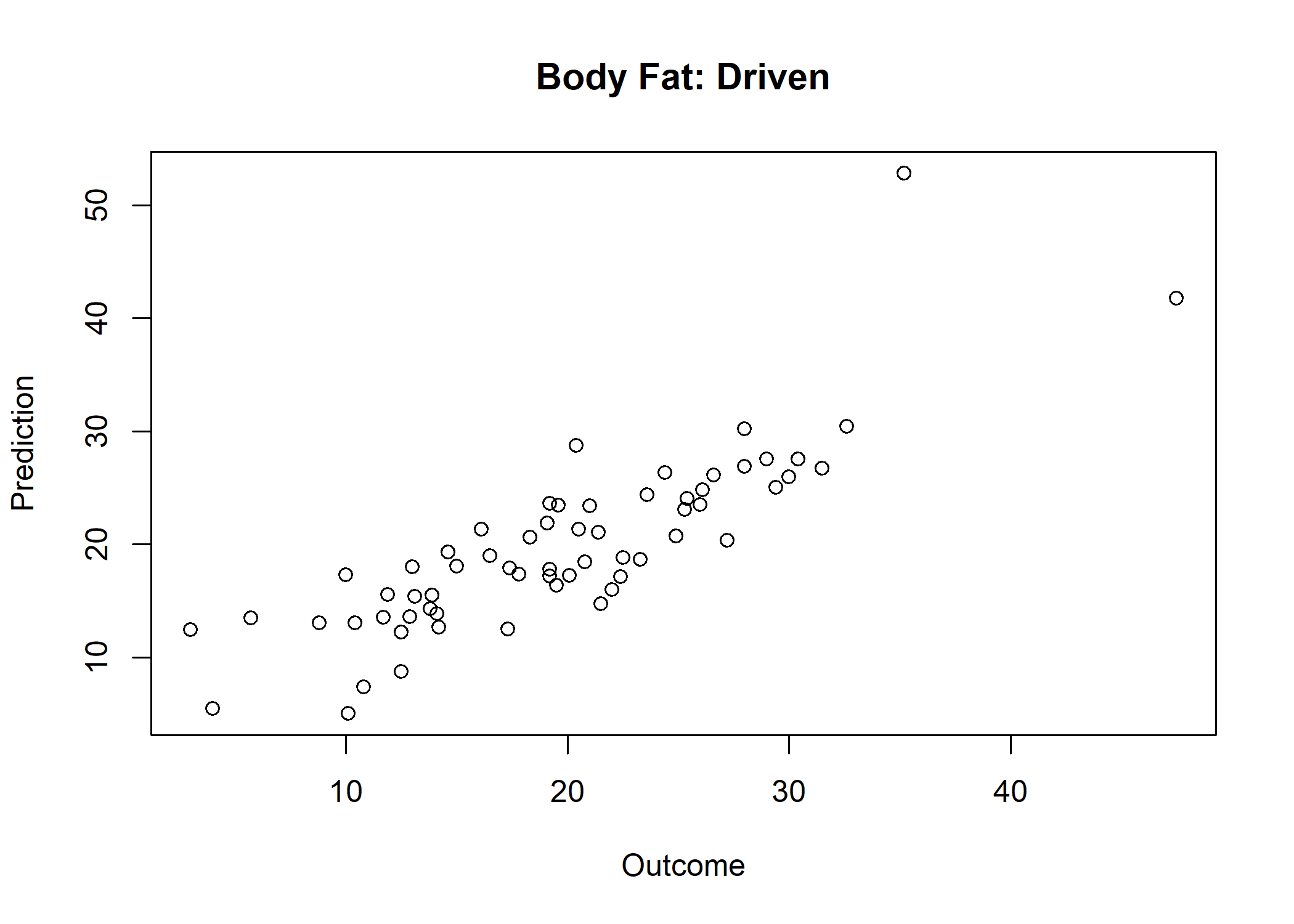
| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 3.37 | 2.77 | 4.14 |

* **pearson**:

Pearson’s product-moment correlation: predictions[, 1] and predictions[, 2]

| Test statistic | df | P value | Alternative hypothesis | cor |
| --- | --- | --- | --- | --- |
| 12.5 | 61 | 1.84e-18 \* \* \* | two.sided | 0.848 |

## Outcome-Driven  
predicBodyFatD <- predict(modelBodyFatD,body\_fat\_Decorrelated\_testD)  
rmetrics <- predictionStats\_regression(cbind(testingset$BodyFat,  
 predicBodyFatD),  
 "Body Fat: Driven")

Body Fat: Driven 

pander::pander(rmetrics)

* **corci**:

| cor |  |  |
| --- | --- | --- |
| 0.842 | 0.751 | 0.902 |

* **biasci**: *0.139*, *-0.972* and *1.249*
* **RMSEci**: *4.41*, *3.76* and *5.34*
* **spearmanci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.849 | 0.745 | 0.912 |

* **MAEci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 3.39 | 2.76 | 4.16 |

* **pearson**:

Pearson’s product-moment correlation: predictions[, 1] and predictions[, 2]

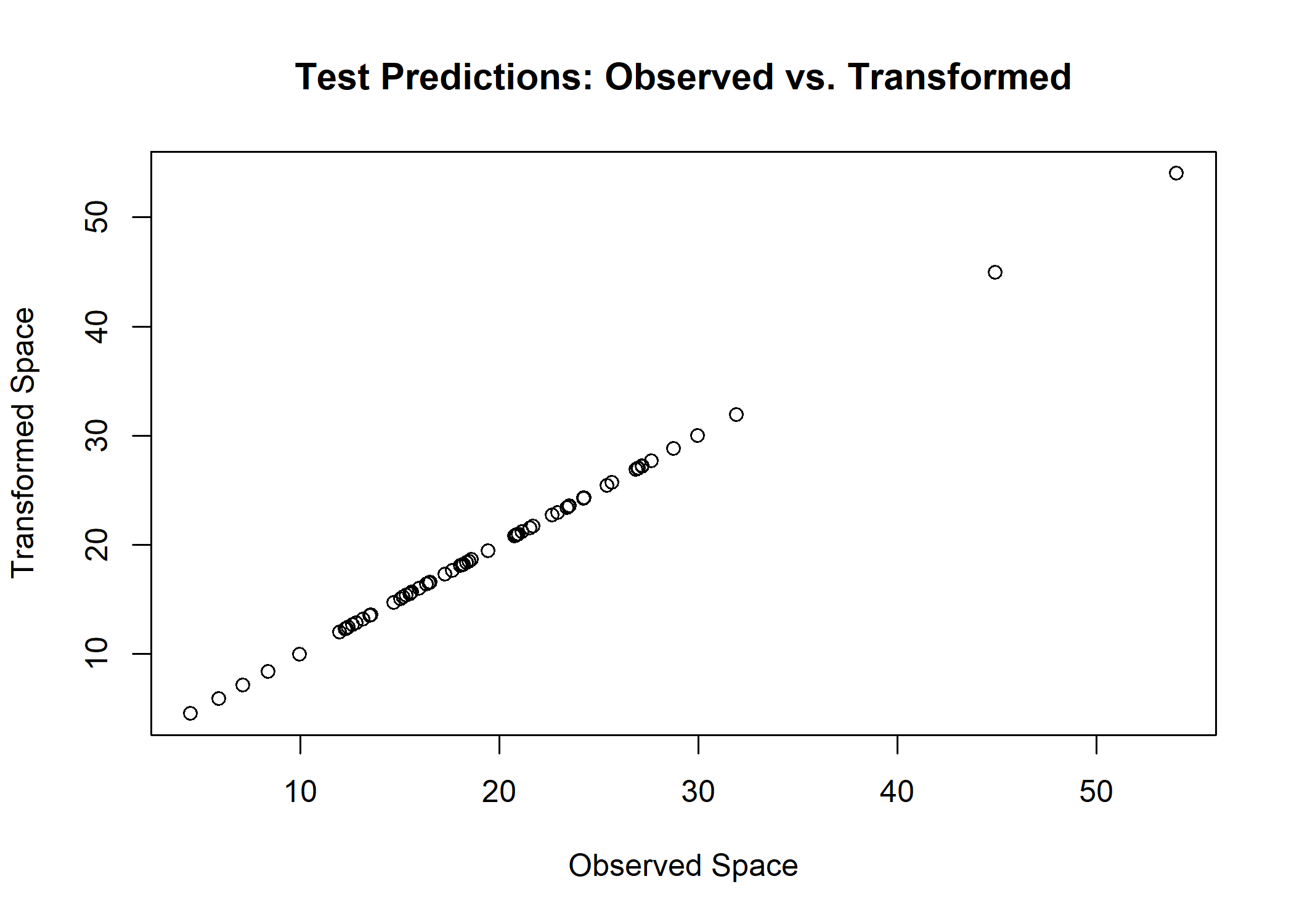
| Test statistic | df | P value | Alternative hypothesis | cor |
| --- | --- | --- | --- | --- |
| 12.2 | 61 | 5.56e-18 \* \* \* | two.sided | 0.842 |

The reported metrics indicated that the model predictions are highly correlated to the real

### Prediction Using the Observed Features

An ILAA user has the option to predict the content from the observed testing set using the computed beta coefficients. The next lines of code show how to do the prediction using model.matrix() R function and the dot product %\*% :

predicBodyFatObst <- model.matrix(formula(observedCoef$formula),testingset) %\*% observedCoef$coefficients  
  
plot(predicBodyFatObst,  
 predicBodyFat,  
 xlab="Observed Space",  
 ylab="Transformed Space",  
 main="Test Predictions: Observed vs. Transformed")



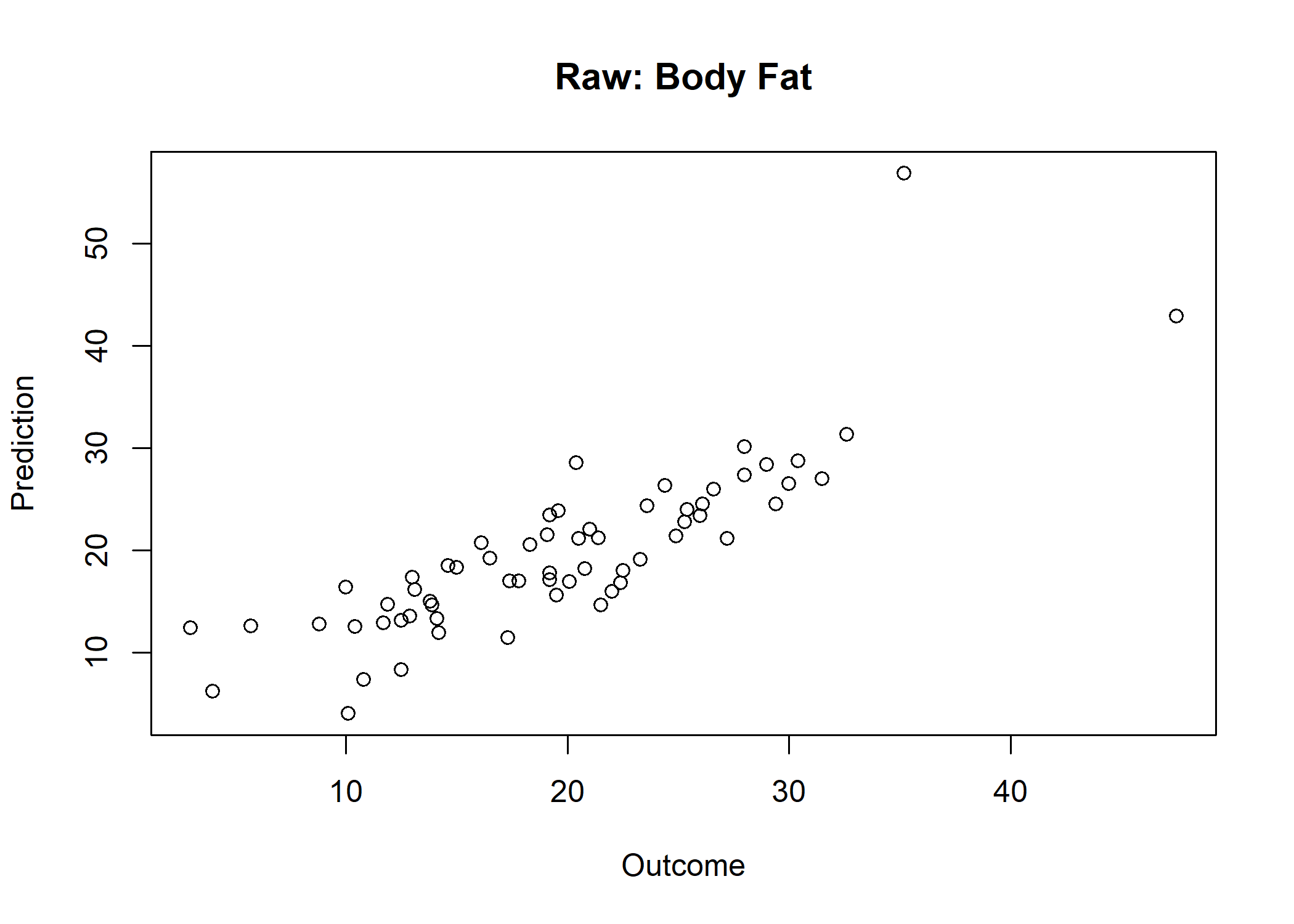
The last plot shows the expected result: that both predictions are identical.

### Comparison to Raw Model

A last experiment is to compare the differences between a LASSO model created from the observed features to the model created from the transformed observations.

The next lines of code compute the linear model using LASSO from the original observed data. Then, it computes the predicted performance.

#rawmodelBodyFat <- LASSO\_MIN(BodyFat~.,trainingset)  
#pander::pander(rawmodelBodyFat$coef)  
  
rawmodelBodyFat <- modelBodyFatRaw;  
  
  
rawpredicBodyFat <- predict(rawmodelBodyFat,testingset)  
rmetrics <- predictionStats\_regression(cbind(testingset$BodyFat,  
 rawpredicBodyFat),"Raw: Body Fat")

Raw: Body Fat 

pander::pander(rmetrics)

* **corci**:

| cor |  |  |
| --- | --- | --- |
| 0.837 | 0.743 | 0.898 |

* **biasci**: *0.0882*, *-1.0731* and *1.2494*
* **RMSEci**: *4.61*, *3.93* and *5.59*
* **spearmanci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.858 | 0.755 | 0.919 |

* **MAEci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 3.38 | 2.7 | 4.23 |

* **pearson**:

Pearson’s product-moment correlation: predictions[, 1] and predictions[, 2]

| Test statistic | df | P value | Alternative hypothesis | cor |
| --- | --- | --- | --- | --- |
| 11.9 | 61 | 1.27e-17 \* \* \* | two.sided | 0.837 |

The evaluation of the testing results indicates that the observed model predictions have a correlation of 0.875. Slightly superior, but not statistically significant, to the one observed from the model estimated from the transformed space: ( vs.  )

### Comparing the Feature Significance on the Models

The main advantage of the ILAA transformation is that the returned latent variables are not colinear hence the statistical significance of the beta coefficients are not affected by multicolinearity. The next code snippet shows how to get the beta coefficients using the lm() , and summary.lm() functions.

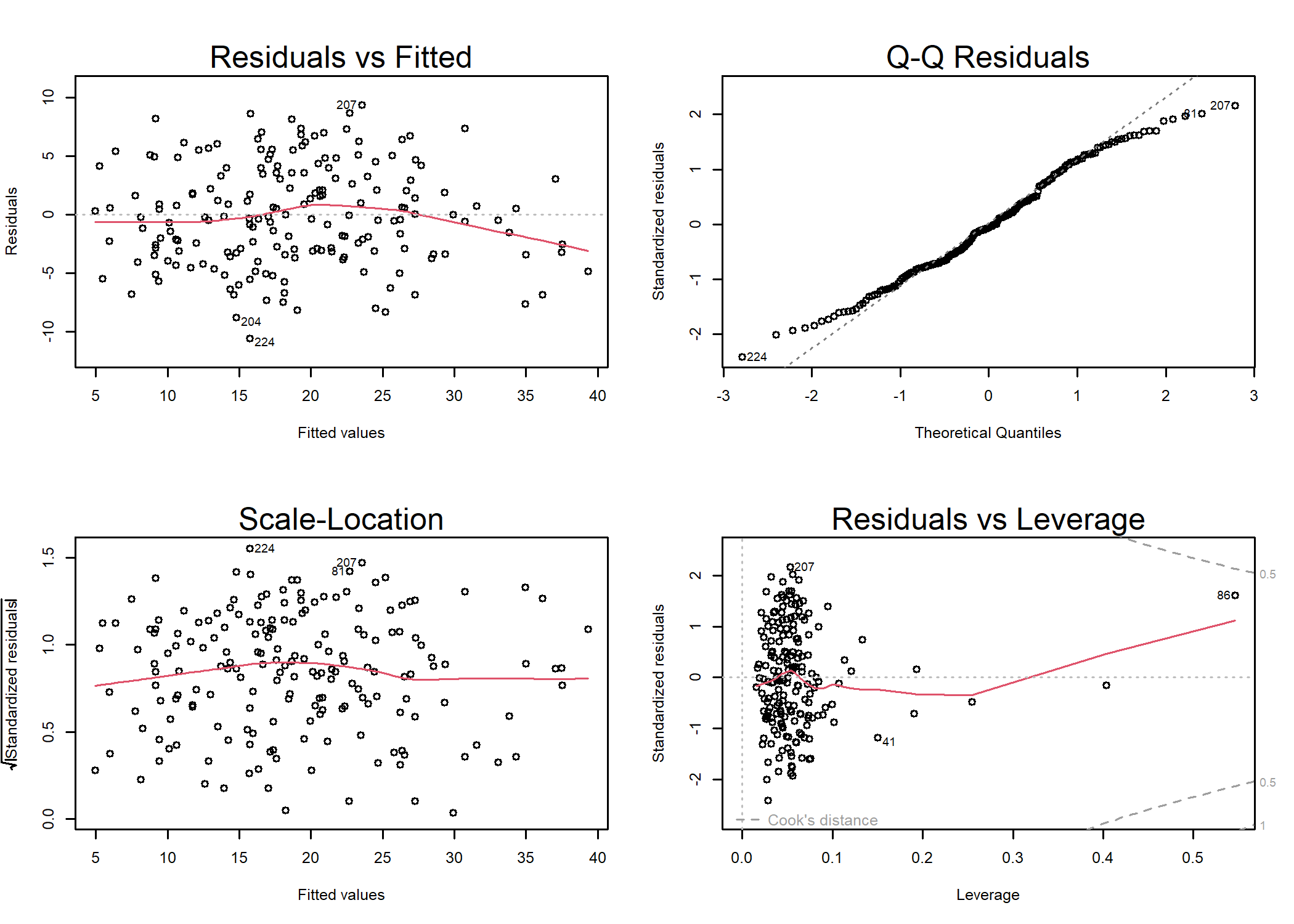
The inspection of the summary results clearly shows that most of the beta coefficients on the transformed data set are significant.

## Raw Model  
par(mfrow=c(2,2),cex=0.5)  
rawlm <- lm(BodyFat~.,  
 trainingset[,c("BodyFat",names(rawmodelBodyFat$coef)[-1])])  
pander::pander(rawlm,add.significance.stars=TRUE)

Fitting linear model: BodyFat ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | 7.7499 | 10.4950 | 0.738 | 4.61e-01 |  |
| **Age** | 0.0619 | 0.0357 | 1.733 | 8.48e-02 |  |
| **Height** | -0.4207 | 0.1476 | -2.851 | 4.87e-03 | \* \* |
| **Neck** | -0.2136 | 0.2639 | -0.810 | 4.19e-01 |  |
| **Chest** | -0.2765 | 0.1133 | -2.442 | 1.56e-02 | \* |
| **Abdomen** | 0.8843 | 0.0919 | 9.627 | 6.62e-18 | \* \* \* |
| **Thigh** | 0.1278 | 0.1383 | 0.924 | 3.57e-01 |  |
| **Ankle** | 0.2219 | 0.2788 | 0.796 | 4.27e-01 |  |
| **Biceps** | 0.2260 | 0.1956 | 1.155 | 2.49e-01 |  |
| **Forearm** | 0.1075 | 0.2538 | 0.423 | 6.72e-01 |  |
| **Wrist** | -1.6853 | 0.6294 | -2.678 | 8.11e-03 | \* \* |

plot(rawlm)

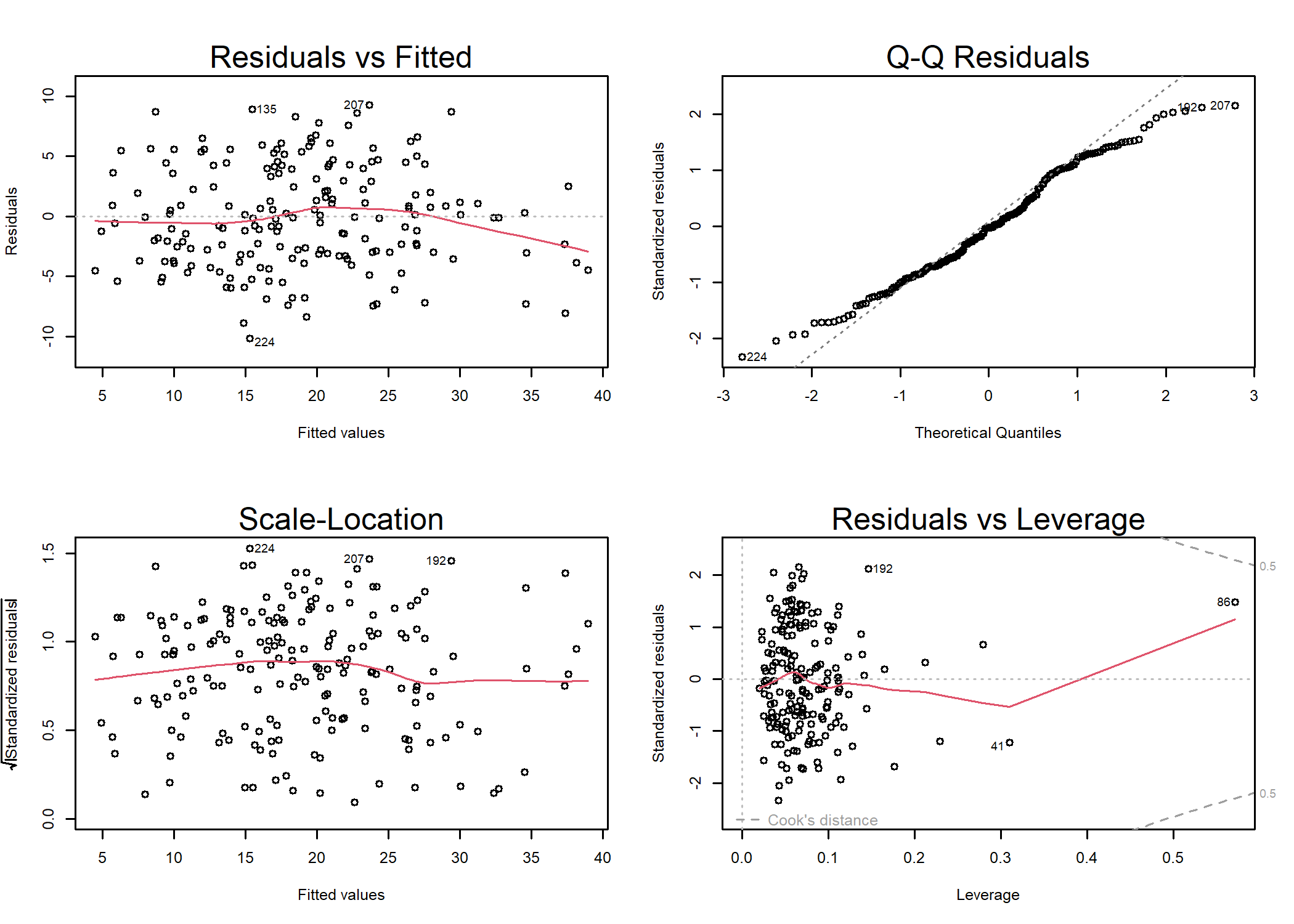


## Outcome-Blind  
par(mfrow=c(2,2),cex=0.5)  
Delm <- lm(BodyFat~.,body\_fat\_Decorrelated\_train[,c("BodyFat",names(modelBodyFat$coef)[-1])])  
pander::pander(Delm,add.significance.stars=TRUE)

Fitting linear model: BodyFat ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | -90.7305 | 70.5488 | -1.286 | 2.00e-01 |  |
| **La\_Age** | 0.0497 | 0.0384 | 1.294 | 1.97e-01 |  |
| **Weight** | 0.1812 | 0.0119 | 15.167 | 1.17e-33 | \* \* \* |
| **La\_Height** | 0.9638 | 0.9999 | 0.964 | 3.36e-01 |  |
| **La\_Neck** | -0.3390 | 0.2571 | -1.318 | 1.89e-01 |  |
| **La\_Chest** | 0.2466 | 0.1225 | 2.014 | 4.56e-02 | \* |
| **La\_Abdomen** | 0.9979 | 0.0995 | 10.032 | 5.63e-19 | \* \* \* |
| **La\_Hip** | 0.3015 | 0.1619 | 1.862 | 6.42e-02 |  |
| **La\_Thigh** | 0.1665 | 0.1582 | 1.052 | 2.94e-01 |  |
| **La\_Knee** | 0.2044 | 0.2819 | 0.725 | 4.69e-01 |  |
| **La\_Ankle** | 0.2925 | 0.2991 | 0.978 | 3.29e-01 |  |
| **La\_Biceps** | 0.2946 | 0.1921 | 1.534 | 1.27e-01 |  |
| **La\_Wrist** | -1.1125 | 0.5793 | -1.921 | 5.64e-02 |  |
| **La\_BMI** | 2.0726 | 0.2075 | 9.990 | 7.34e-19 | \* \* \* |

plot(Delm)

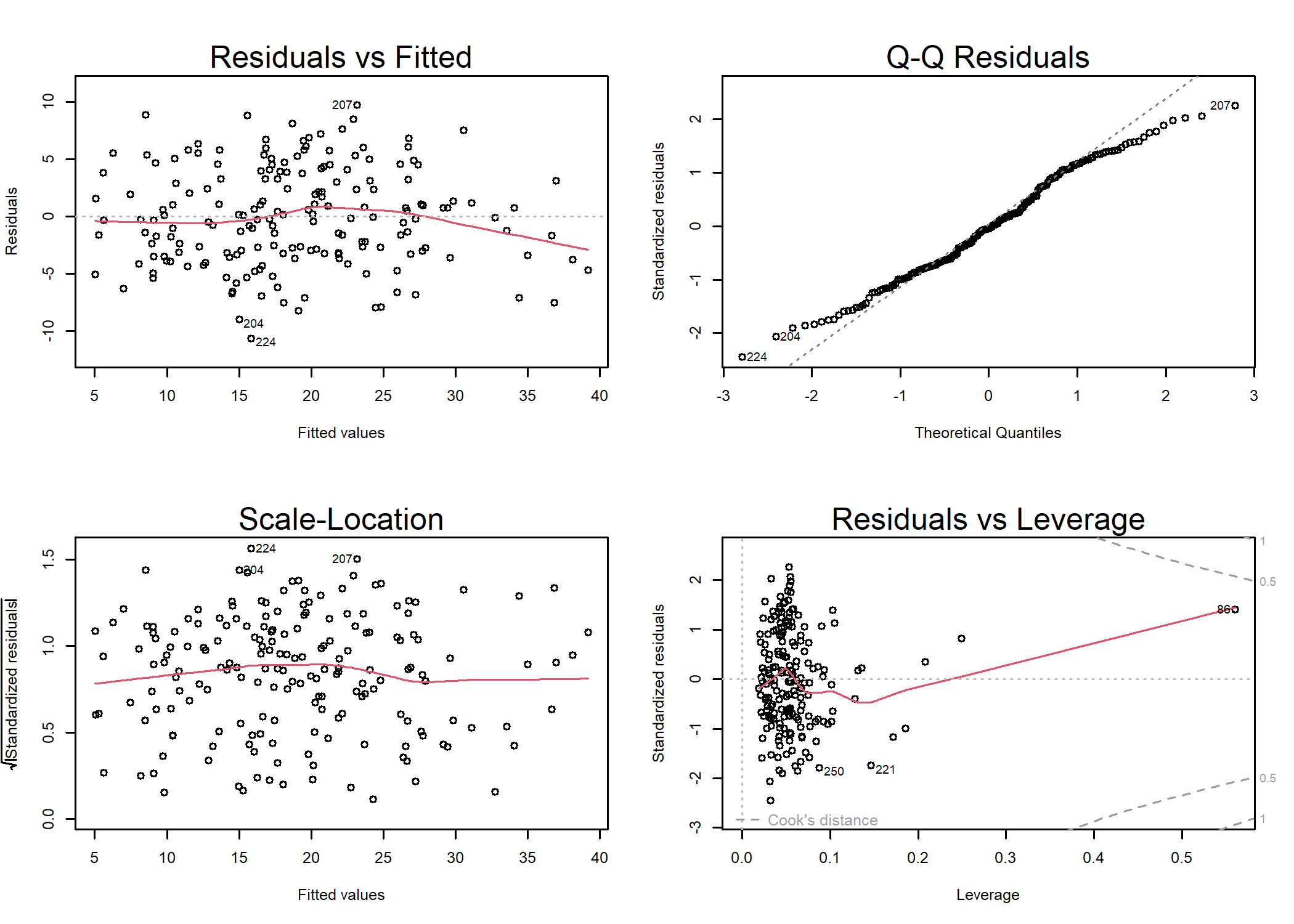


## Outcome-Driven  
par(mfrow=c(2,2),cex=0.5)  
Delm <- lm(BodyFat~.,  
 body\_fat\_Decorrelated\_trainD[,c("BodyFat",names(modelBodyFatD$coef)[-1])])  
pander::pander(Delm,add.significance.stars=TRUE)

Fitting linear model: BodyFat ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | -24.3110 | 18.3582 | -1.32 | 1.87e-01 |  |
| **La\_Age** | 0.0493 | 0.0381 | 1.29 | 1.97e-01 |  |
| **La\_Weight** | -0.1317 | 0.0432 | -3.05 | 2.65e-03 | \* \* |
| **La\_Neck** | -0.6998 | 0.2252 | -3.11 | 2.20e-03 | \* \* |
| **La\_Chest** | -0.2334 | 0.1341 | -1.74 | 8.36e-02 |  |
| **Abdomen** | 0.6707 | 0.0316 | 21.25 | 1.39e-50 | \* \* \* |
| **La\_Hip** | -0.2604 | 0.1007 | -2.59 | 1.05e-02 | \* |
| **La\_Thigh** | 0.1999 | 0.1503 | 1.33 | 1.85e-01 |  |
| **La\_Ankle** | 0.3420 | 0.2892 | 1.18 | 2.39e-01 |  |
| **La\_Biceps** | 0.2839 | 0.1909 | 1.49 | 1.39e-01 |  |
| **La\_Wrist** | -0.9602 | 0.5653 | -1.70 | 9.12e-02 |  |

plot(Delm)



par(op)

## Train a Logistic Model for Overweight Prediction

This last experiment showcases the effect of data transformation on logistic modeling. This experiment starts by creating a data-frame that does not includes the , , and variables. The target outcome is to identify if the person is Overweight or normal. (BMI>=25). The next lines of code compute the new data frames and remove the above mentioned variables.

### Data Conditioning

First Remove Height and Weight from Training and Testing Sets

trainingsetBMI <- trainingset[,!(colnames(trainingset) %in% c("Weight","Height"))]  
testingsetBMI <- testingset[,!(colnames(trainingset) %in% c("Weight","Height"))]  
trainingsetBMI$Overweight <- 1\*(trainingsetBMI$BMI>=25)  
testingsetBMI$Overweight <- 1\*(testingsetBMI$BMI>=25)  
trainingsetBMI$BMI <- NULL  
testingsetBMI$BMI <- NULL  
  
# The number of subjects  
pander::pander(table(trainingsetBMI$Overweight),caption="Training Distribution")

Training Distribution

| 0 | 1 |
| --- | --- |
| 96 | 92 |

pander::pander(table(testingsetBMI$Overweight),caption="Testing Distribution")

Testing Distribution

| 0 | 1 |
| --- | --- |
| 29 | 34 |

## The outcome-blind transformation  
OW\_Decorrelated\_train <- ILAA(trainingsetBMI,  
 thr=0.2,  
 Outcome="Overweight")  
  
OW\_Decorrelated\_test <- predictDecorrelate(OW\_Decorrelated\_train,testingsetBMI)  
  
## The outcome-driven transformation  
  
OW\_Decorrelated\_trainD <- ILAA(trainingsetBMI,  
 thr=0.2,  
 Outcome="Overweight",  
 drivingFeatures="Overweight")  
  
OW\_Decorrelated\_testD <- predictDecorrelate(OW\_Decorrelated\_trainD,testingsetBMI)

The last code snippet transforms the observed features using ILLA and setting a target variable and setting the convergence not to be affected by the target outcome.

### The Logistic Model

LASSO\_MIN with a binomial family is used to compute the logistic model of overweight.

## Outcome-blind  
modelOverweight <- LASSO\_MIN(Overweight~.,  
 OW\_Decorrelated\_train,  
 family="binomial")  
pander::pander(as.matrix(modelOverweight$coef),caption="Training: Blind")

Training: Blind

|  |  |
| --- | --- |
| **(Intercept)** | -60.8766 |
| **La\_BodyFat** | 0.0276 |
| **Chest** | 0.6239 |
| **La\_Abdomen** | 0.1753 |
| **La\_Hip** | 0.1748 |
| **La\_Thigh** | 0.0502 |
| **La\_Knee** | -0.0198 |
| **La\_Ankle** | 0.0707 |
| **La\_Biceps** | 0.0363 |
| **La\_Wrist** | 0.2129 |

## Outcome-driven  
modelOverweightD <- LASSO\_MIN(Overweight~.,  
 OW\_Decorrelated\_trainD,  
 family="binomial")  
pander::pander(as.matrix(modelOverweightD$coef),caption="Training: Driven")

Training: Driven

|  |  |
| --- | --- |
| **(Intercept)** | -49.36391 |
| **La\_BodyFat** | 0.00335 |
| **Chest** | 0.51300 |
| **La\_Abdomen** | 0.12957 |
| **La\_Hip** | 0.10793 |
| **La\_Ankle** | 0.01892 |
| **La\_Biceps** | 0.01039 |
| **La\_Wrist** | 0.01784 |

### The Model Coefficients in the Observed Space

Once the logistic model is created in the transformed space, we can compute the beta coefficients for each one of the observed variables.

# Get the coefficients in the observed space  
observedCoef <- getObservedCoef(OW\_Decorrelated\_train,modelOverweight)  
pander::pander(as.matrix(observedCoef$coefficients),caption="Observed Coefficients")

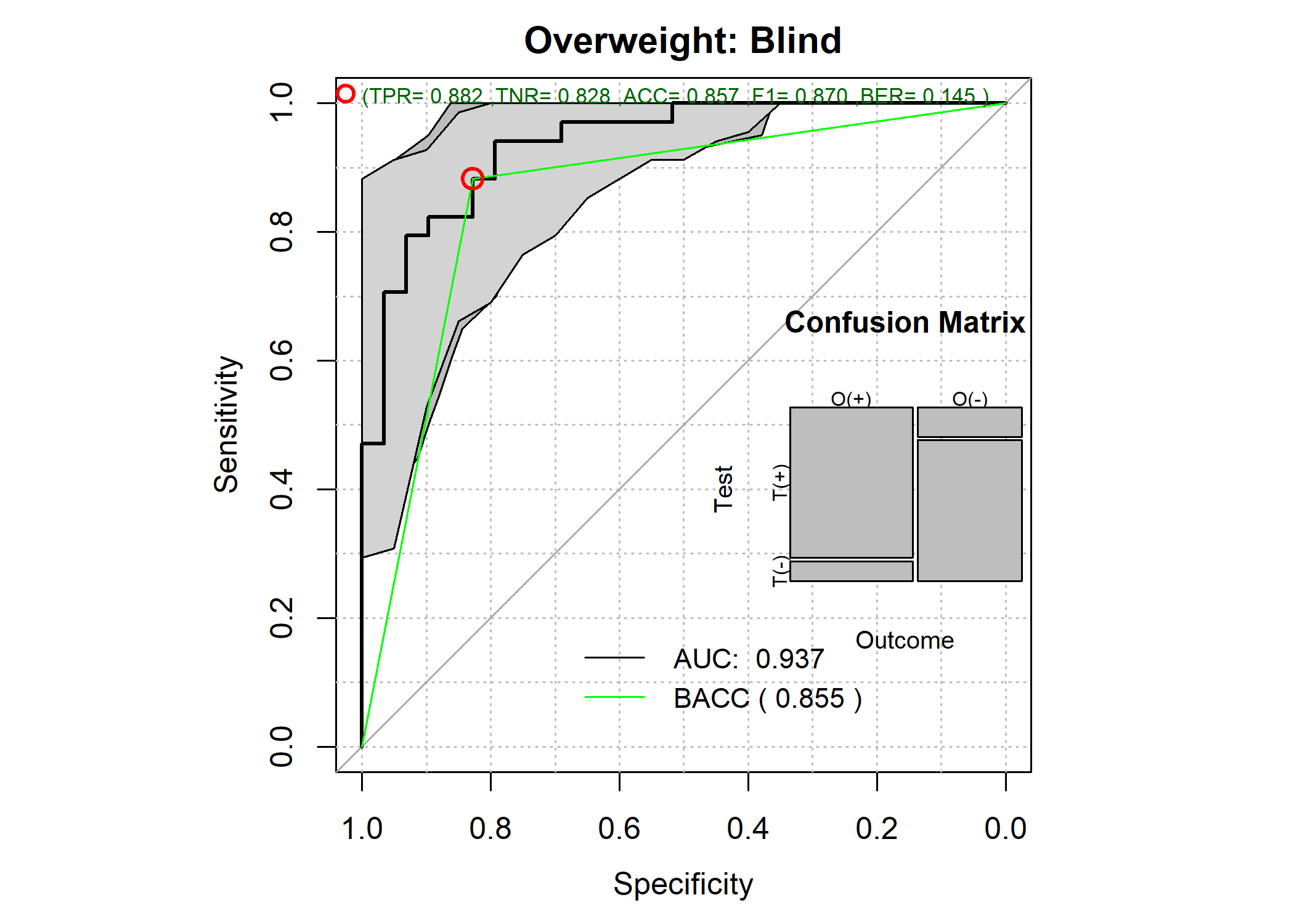
Observed Coefficients

|  |  |
| --- | --- |
| **(Intercept)** | -60.8766 |
| **BodyFat** | 0.0276 |
| **Neck** | -0.0539 |
| **Chest** | 0.3680 |
| **Abdomen** | 0.1495 |
| **Hip** | 0.0518 |
| **Thigh** | 0.0405 |
| **Knee** | -0.0376 |
| **Ankle** | 0.0707 |
| **Biceps** | 0.0363 |
| **Wrist** | 0.1712 |

### Predict Using the Transformed Data Set

The predictions of the testing set can be done using the handy predict() function. The evaluation of the testing results can be evaluated using the predictionStats\_binary() function.

## Outcome-blind  
predicOverweight <- predict(modelOverweight,OW\_Decorrelated\_test)  
pr <- predictionStats\_binary(cbind(OW\_Decorrelated\_test$Overweight,  
 predicOverweight),"Overweight: Blind")



pander::pander(pr$ClassMetrics)

* **accci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.857 | 0.762 | 0.937 |

* **senci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.856 | 0.763 | 0.935 |

* **aucci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.856 | 0.763 | 0.935 |

* **berci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.144 | 0.0651 | 0.237 |

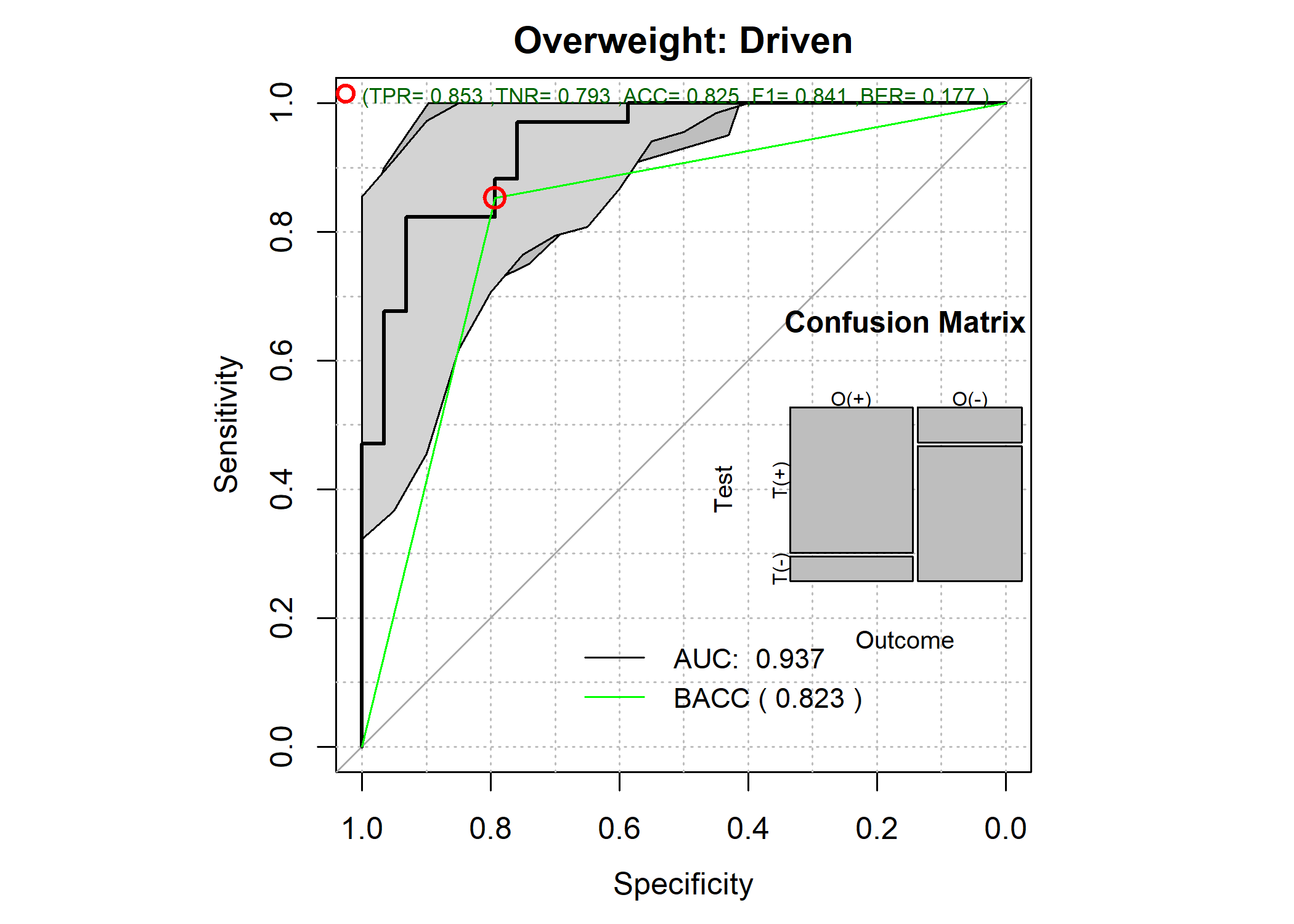
* **preci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.857 | 0.764 | 0.938 |

* **F1ci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.855 | 0.76 | 0.935 |

## Outcome-Driven  
predicOverweightD <- predict(modelOverweightD,OW\_Decorrelated\_testD)  
pr <- predictionStats\_binary(cbind(OW\_Decorrelated\_test$Overweight,  
 predicOverweightD),"Overweight: Driven")



pander::pander(pr$ClassMetrics)

* **accci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.825 | 0.73 | 0.905 |

* **senci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.826 | 0.732 | 0.91 |

* **aucci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.826 | 0.732 | 0.91 |

* **berci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.174 | 0.0903 | 0.268 |

* **preci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.827 | 0.73 | 0.914 |

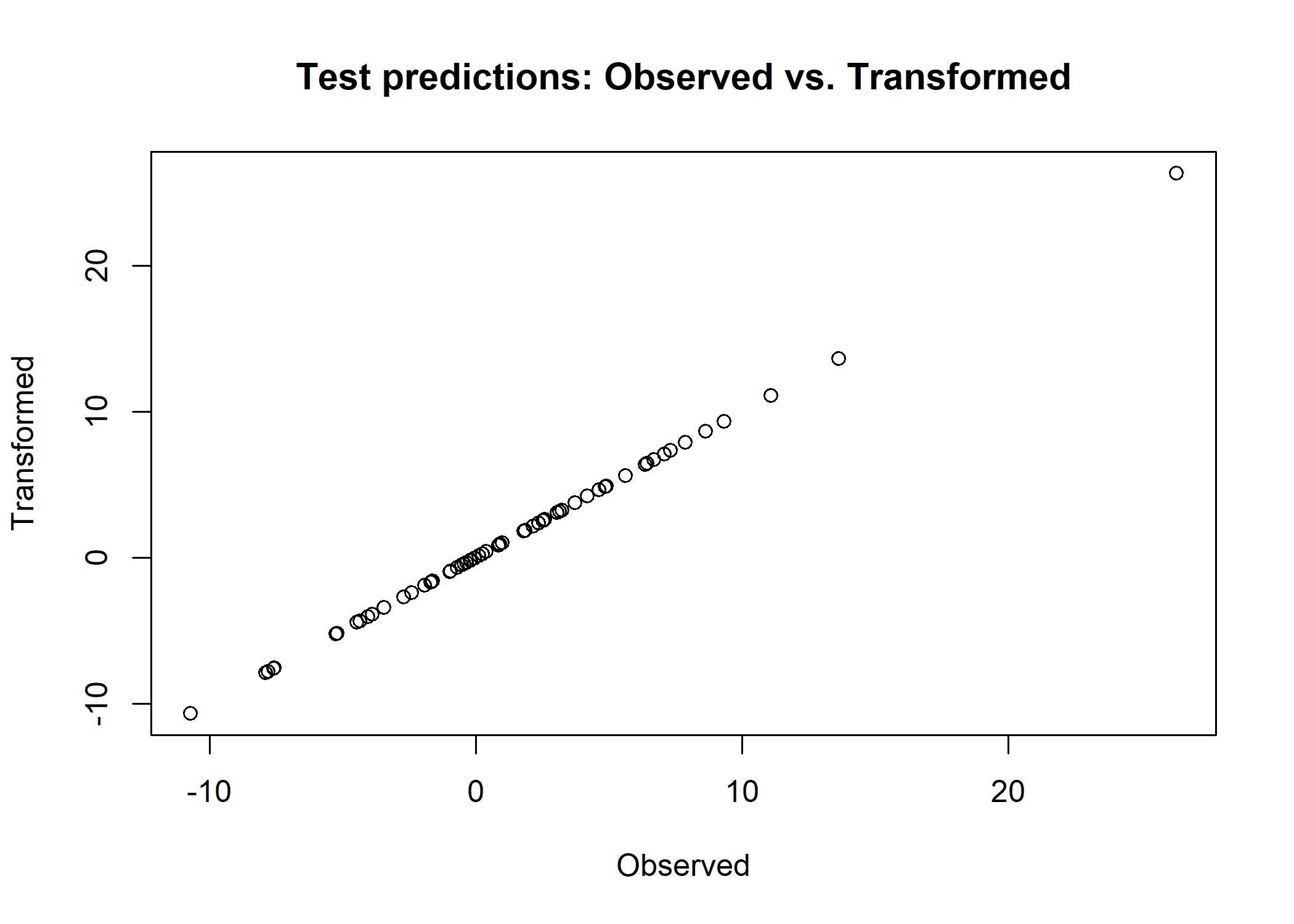
* **F1ci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.825 | 0.729 | 0.905 |

### Prediction Using the Observed Features

The predict of the testing set can be done using the model.matrix() and the dot product %\*%.

predicOverweightObst <- model.matrix(formula(observedCoef$formula),testingsetBMI) %\*% observedCoef$coefficients  
#predicOverweightObst <- 1.0/(1.0 + exp(-predicOverweightObst));  
  
plot(predicOverweightObst,predicOverweight,  
 xlab="Observed",  
 ylab="Transformed",  
 main="Test predictions: Observed vs. Transformed")



The last plot shows the expected result: both predictions are identical.

### Comparison to Raw Model

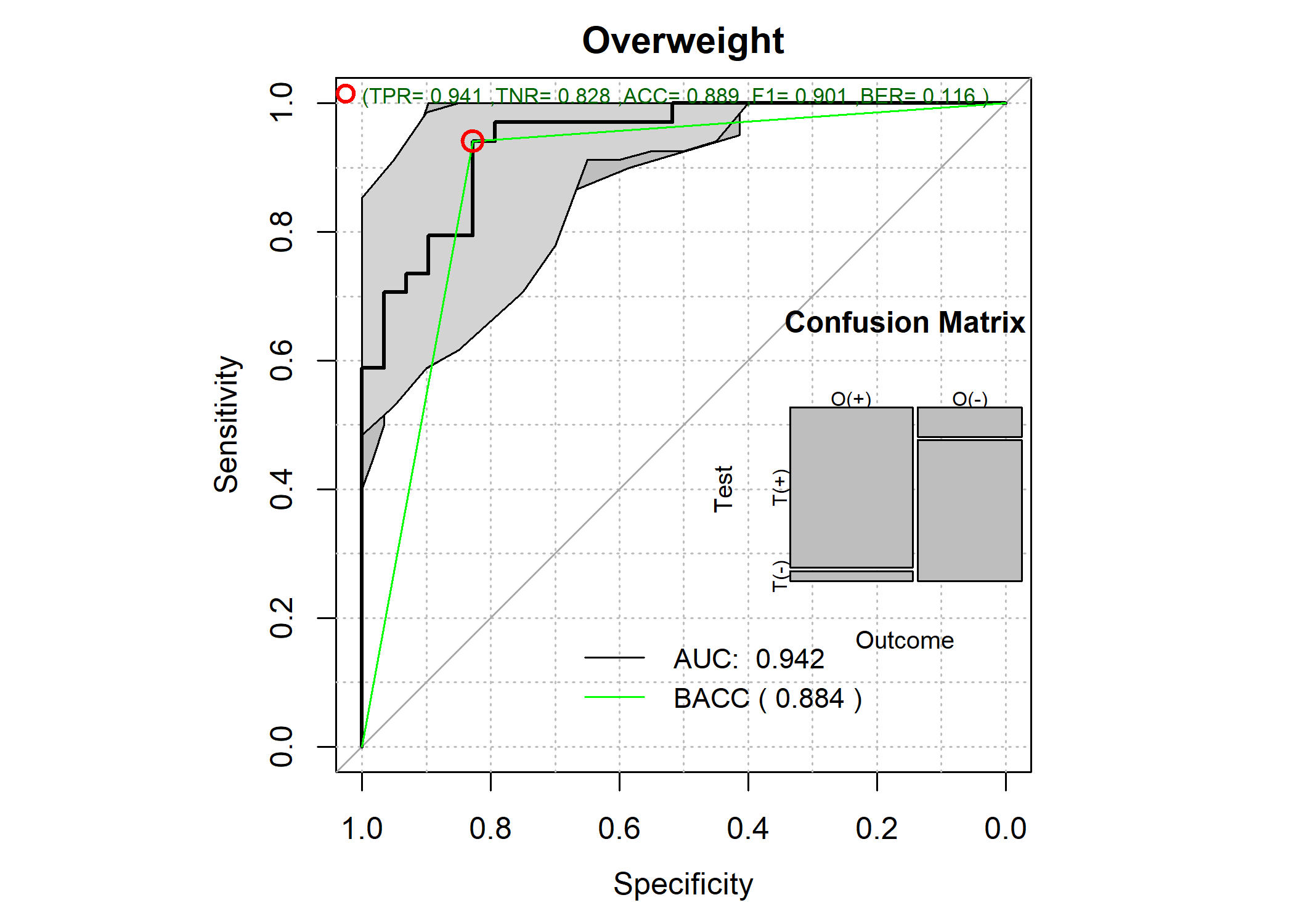
To showcase the advantage of transformed modeling *vs.* raw modeling, here I’ll estimate the logistic model from the observed variables and contrast to the model generated from the transformed space.

The next lines of code compute the logistic model and display its testing performance:

##Training  
rawmodelOverweight <- LASSO\_MIN(Overweight~.,  
 trainingsetBMI,  
 family="binomial")  
pander::pander(rawmodelOverweight$coef)

| (Intercept) | BodyFat | Chest | Abdomen | Hip | Thigh | Ankle | Biceps | Wrist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -66.5 | 0.044 | 0.331 | 0.188 | 0.0226 | 0.0946 | 0.132 | 0.0818 | 0.106 |

## Predict  
rawpredicOverweight <- predict(rawmodelOverweight,testingsetBMI)  
pr <- predictionStats\_binary(cbind(testingsetBMI$Overweight,  
 rawpredicOverweight),"Overweight")



pander::pander(pr$ClassMetrics)

* **accci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.889 | 0.81 | 0.952 |

* **senci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.888 | 0.797 | 0.955 |

* **aucci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.888 | 0.797 | 0.955 |

* **berci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.112 | 0.0454 | 0.203 |

* **preci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.896 | 0.813 | 0.964 |

* **F1ci**:

| 50% | 2.5% | 97.5% |
| --- | --- | --- |
| 0.888 | 0.793 | 0.952 |

The model created from the observed data has an ROC AUC that is not statistically significant to the transformed model

### Comparing the Feature Significance on the Models

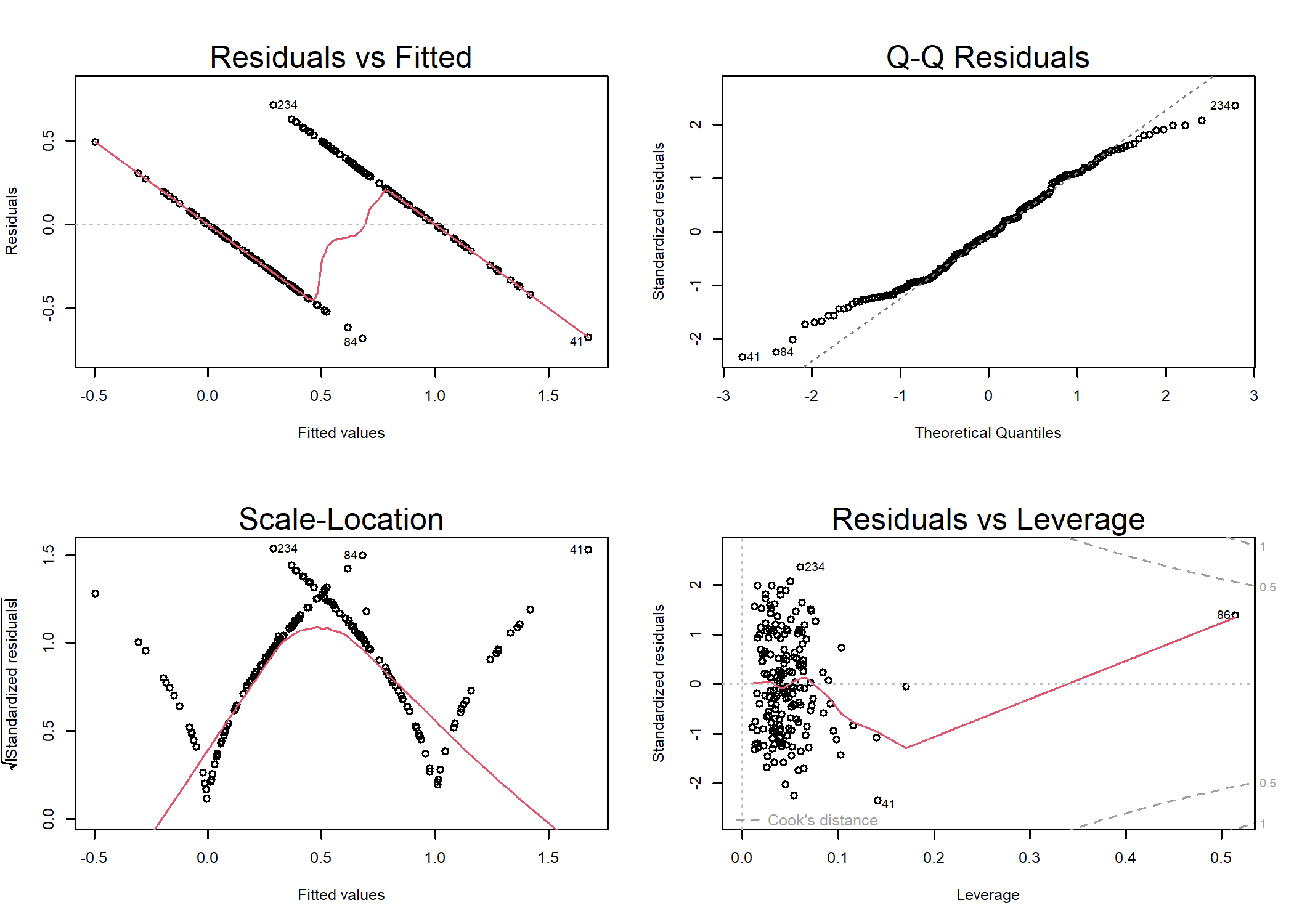
This last lines of code will compute the significance of the beta coefficients for both the observed model and the latent-based model. The user can clearly see that all the betas of the latent-based model are statically significant. An effect that is not seen in the logistic observed model.

par(mfrow=c(2,2),cex=0.5)  
  
## Raw model  
rawlm <- lm(Overweight~.,trainingsetBMI[,c("Overweight",names(rawmodelOverweight$coef)[-1])])  
pander::pander(rawlm,add.significance.stars=TRUE)

Fitting linear model: Overweight ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | -3.9632 | 0.55256 | -7.172 | 1.88e-11 | \* \* \* |
| **BodyFat** | 0.0060 | 0.00511 | 1.173 | 2.42e-01 |  |
| **Chest** | 0.0156 | 0.00783 | 1.993 | 4.78e-02 | \* |
| **Abdomen** | 0.0176 | 0.00830 | 2.116 | 3.57e-02 | \* |
| **Hip** | -0.0142 | 0.01013 | -1.402 | 1.63e-01 |  |
| **Thigh** | 0.0087 | 0.01024 | 0.849 | 3.97e-01 |  |
| **Ankle** | 0.0200 | 0.01927 | 1.039 | 3.00e-01 |  |
| **Biceps** | 0.0268 | 0.01271 | 2.112 | 3.60e-02 | \* |
| **Wrist** | 0.0401 | 0.03859 | 1.039 | 3.00e-01 |  |

plot(rawlm)

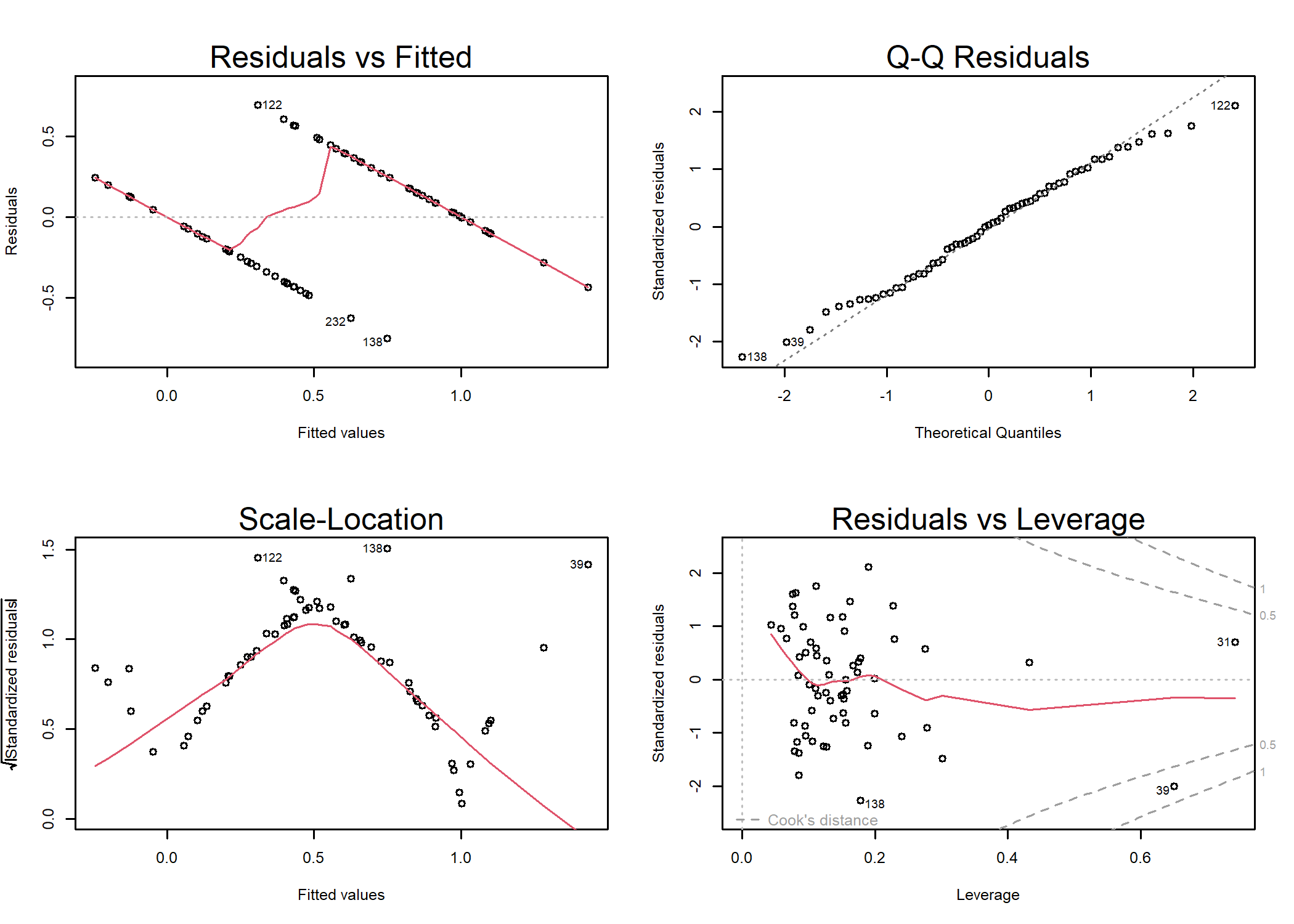


## Outcome-blind  
par(mfrow=c(2,2),cex=0.5)  
Delm <- lm(Overweight~.,OW\_Decorrelated\_test[,c("Overweight",names(modelOverweight$coef)[-1])])  
pander::pander(Delm,add.significance.stars=TRUE)

Fitting linear model: Overweight ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | -1.923103 | 0.97461 | -1.97320 | 5.37e-02 |  |
| **La\_BodyFat** | -0.002238 | 0.01292 | -0.17315 | 8.63e-01 |  |
| **Chest** | 0.038521 | 0.00541 | 7.12645 | 2.82e-09 | \* \* \* |
| **La\_Abdomen** | 0.015506 | 0.01297 | 1.19583 | 2.37e-01 |  |
| **La\_Hip** | -0.006350 | 0.01331 | -0.47691 | 6.35e-01 |  |
| **La\_Thigh** | 0.069865 | 0.02348 | 2.97580 | 4.39e-03 | \* \* |
| **La\_Knee** | -0.000289 | 0.03721 | -0.00777 | 9.94e-01 |  |
| **La\_Ankle** | -0.042165 | 0.03006 | -1.40281 | 1.67e-01 |  |
| **La\_Biceps** | 0.012316 | 0.03435 | 0.35855 | 7.21e-01 |  |
| **La\_Wrist** | -0.004661 | 0.08887 | -0.05245 | 9.58e-01 |  |

plot(Delm)

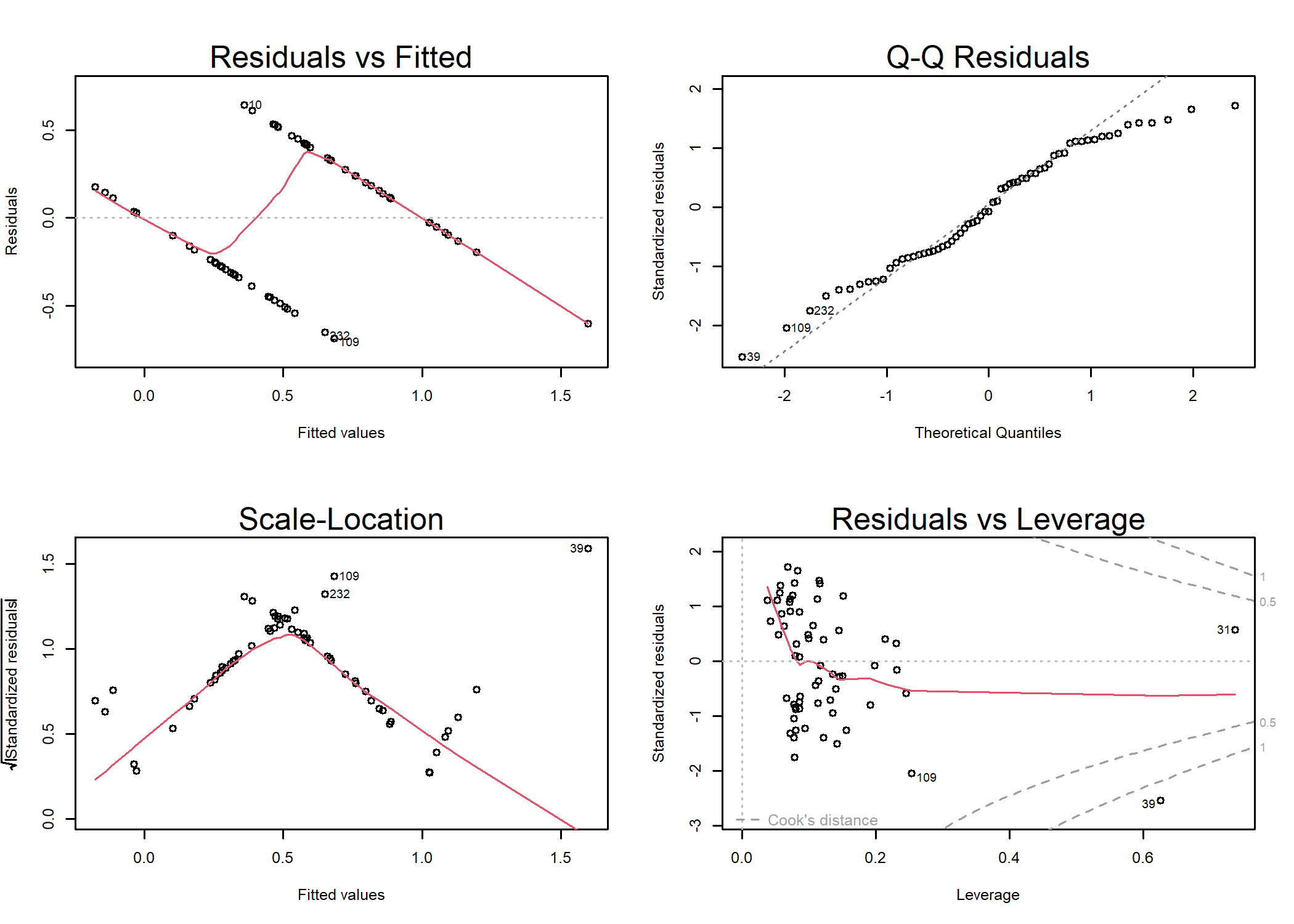


## Outcome-Driven  
par(mfrow=c(2,2),cex=0.5)  
Delm <- lm(Overweight~.,OW\_Decorrelated\_testD[,c("Overweight",names(modelOverweightD$coef)[-1])])  
pander::pander(Delm,add.significance.stars=TRUE)

Fitting linear model: Overweight ~ .

|  | Estimate | Std. Error | t value | Pr(>|t|) |  |
| --- | --- | --- | --- | --- | --- |
| **(Intercept)** | -1.80519 | 1.00514 | -1.796 | 7.80e-02 |  |
| **La\_BodyFat** | -0.00573 | 0.01364 | -0.420 | 6.76e-01 |  |
| **Chest** | 0.03517 | 0.00561 | 6.269 | 5.91e-08 | \* \* \* |
| **La\_Abdomen** | 0.01632 | 0.01374 | 1.187 | 2.40e-01 |  |
| **La\_Hip** | -0.00797 | 0.01388 | -0.574 | 5.68e-01 |  |
| **La\_Ankle** | -0.05489 | 0.03159 | -1.738 | 8.79e-02 |  |
| **La\_Biceps** | 0.04162 | 0.03492 | 1.192 | 2.38e-01 |  |
| **La\_Wrist** | -0.02560 | 0.09185 | -0.279 | 7.81e-01 |  |

plot(Delm)



# Conclusion

In conclusion, ILAA (Iterative Linear Association Analysis), stands as an unsupervised computer-based methodology adept at estimating linear transformation matrices. These matrices enable the conversion of datasets into a fresh latent-based space, offering a user-controlled degree of correlation. This report has effectively demonstrated the practical application of ILAA, providing comprehensive insights into its functions for estimating, predicting, and scrutinizing transformations. Such capabilities hold significant promise in supervised learning scenarios, encompassing regression and logistic models.