ILAA Tutorial

Jose Gerardo Tamez-Peña

2024-06-19

Contents

1 Introduction

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

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

$$Q = WX,$$

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

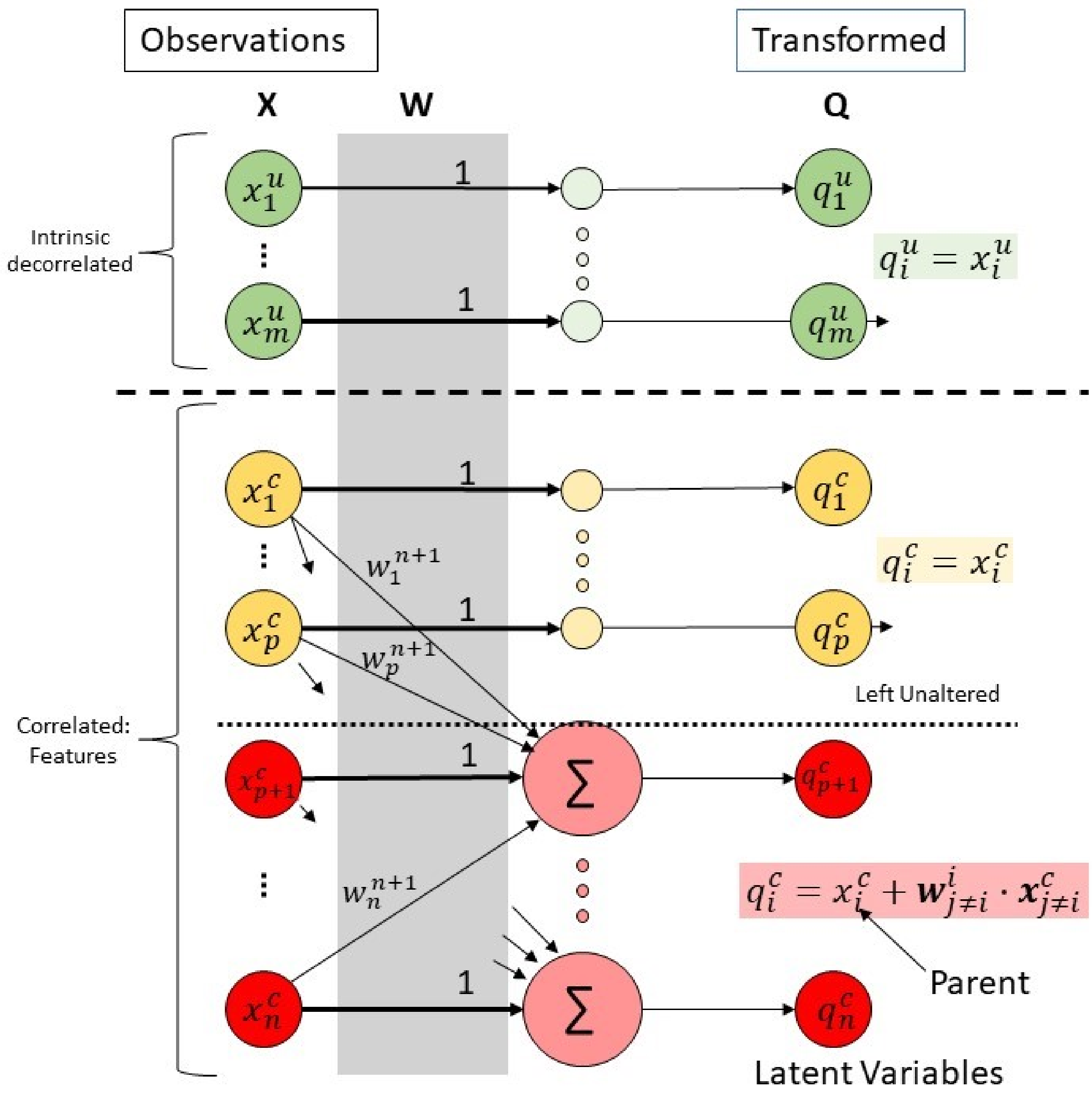


Figure 1: Figure 1: The UPLTM: any linear transformation matrix that does not reduce data dimensionality and has a one-to-one association between units of measurement, i.e., it is preserving the space metric. The ERT is special form of UPLTM that is preserving the metric, and aims to create latent variables with controlled association between them.

The returned transformation matrix can be used to:

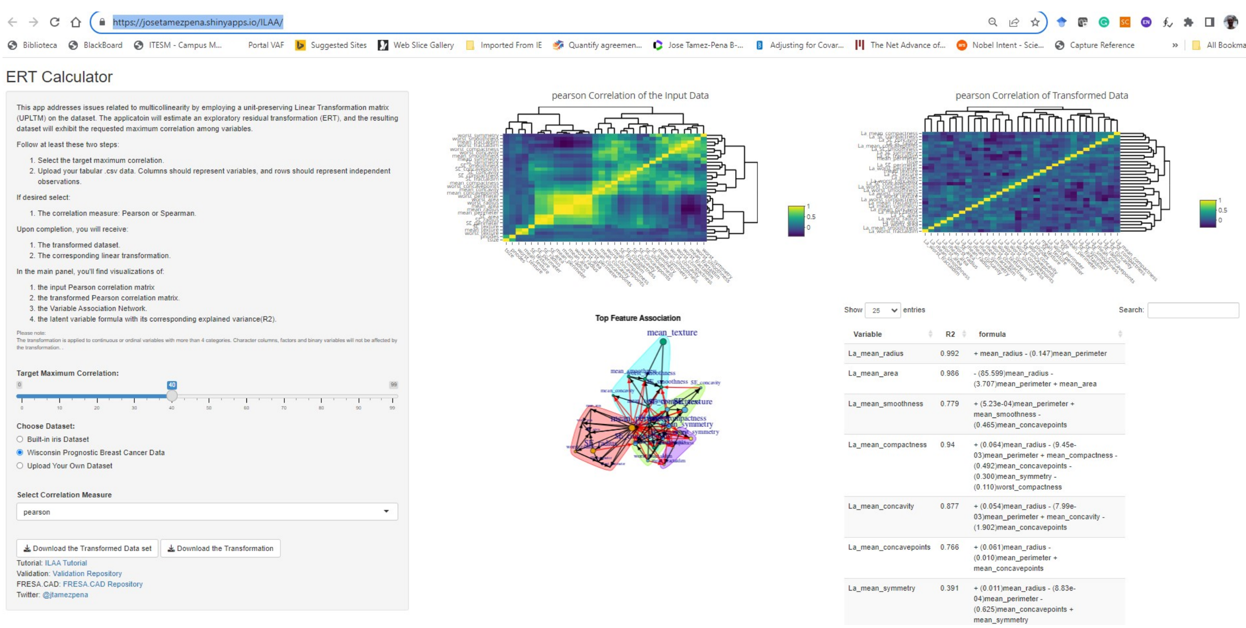
1. Do an exploratory analysis of latent variables and their association to the observed variables
2. Do exploratory discovery of latent variables associated with a specific outcome-target
3. Addressing multicollinearity issues in regression models
    1. Better estimation and interpretation of model 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.

1.1 Shiny App

Users can test ILLA application using the ERT calculator: ILLA Shiny App:



1.2 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.

```r
## 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")
```

2 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 web address: http://silver.sdsmt.edu/~rwjohnso"

2.1 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)

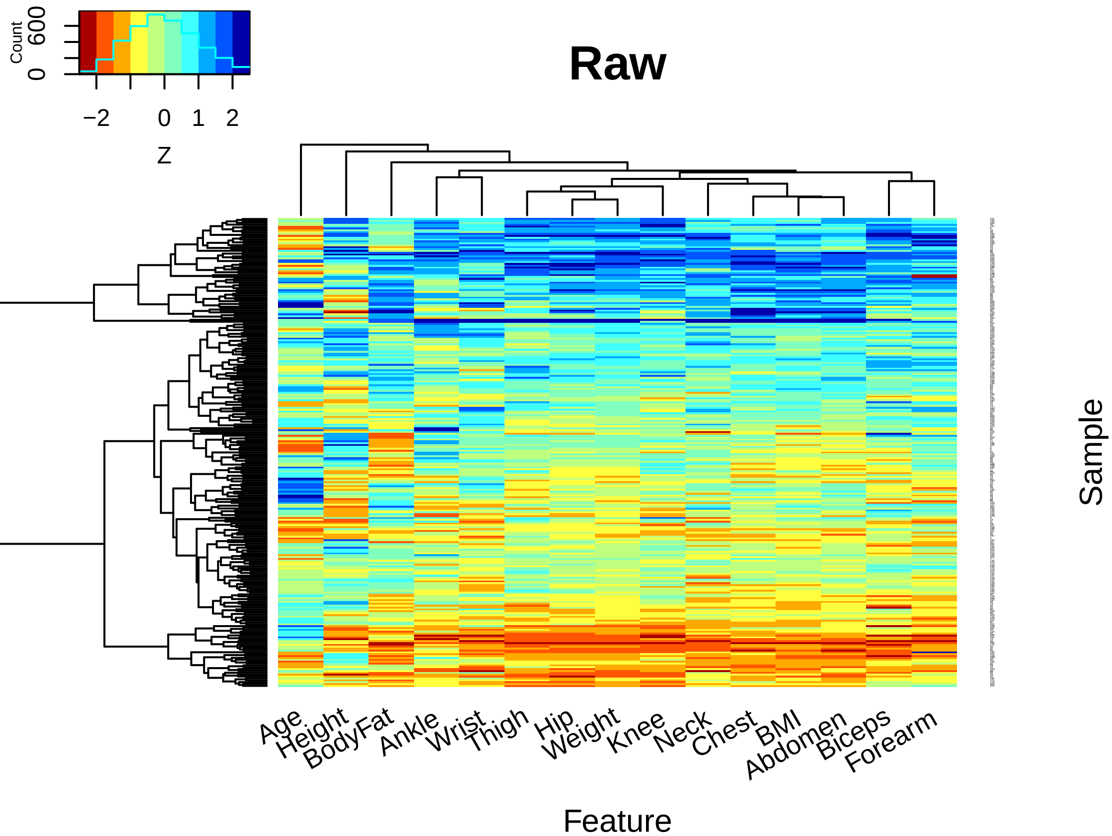
```r
body_fat <- read.csv("~/GitHub/LatentBiomarkers/Data/BodyFat/BodyFat.csv",
                     header=TRUE)

### 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,]
```

2.1.1 The Heatmap of the Raw Data

Now, here we show the heatmap of the dataframe:

```r
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)
```

3 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% constrained resampling 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.

3.1 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.

3.2 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.

```r
# 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=0.890,thr=0.900>, Top: 2< 1 >Fa= 2,<|><>Tot Used: 5 , Added: 3 , Zero Std: 0 , Max Cor: 0.888
2 <R=0.861,thr=0.800>, Top: 1< 5 >Fa= 2,<|><>Tot Used: 9 , Added: 5 , Zero Std: 0 , Max Cor: 0.860
3 <R=0.860,thr=0.800>, Top: 1< 1 >Fa= 2,<|><>Tot Used: 10 , Added: 1 , Zero Std: 0 , Max Cor: 0.959
4 <R=0.959,thr=0.950>, Top: 1< 1 >Fa= 2,<|><>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

```r
# 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
```

3.2.1 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")
```

Table 1: Unexplained variance ratio of latent models (continued below)

	BodyFat	Age	Ankle	Forearm	Wrist	Weight
Default	1.000	1.000	1.000	1.000	1.000	1.0000
Spearman	1.000	1.000	1.000	1.000	1.000	1.0000
At_40	0.309	1.000	0.624	0.602	0.459	1.0000
At_0	0.290	0.483	0.540	0.518	0.403	1.0000
BMI_BF	0.475	0.491	0.583	0.517	0.386	0.2108
BodyFat_Driven	1.000	0.494	0.565	0.513	0.383	0.0687

Table 2: Table continues below

	Biceps	Neck	Knee	Thigh	BMI	Chest	Abdomen
Default	0.359	0.303	0.272	0.242	0.211	0.1710	0.1500
Spearman	1.000	0.303	0.273	0.242	0.212	0.1737	0.1500
At_40	0.359	0.303	0.272	0.194	0.211	0.1085	0.1500
At_0	0.320	0.269	0.228	0.183	0.211	0.1007	0.1277
BMI_BF	0.308	0.294	0.243	0.234	1.000	0.0980	0.0734
BodyFat_Driven	0.315	0.367	0.256	0.190	0.124	0.0984	1.0000

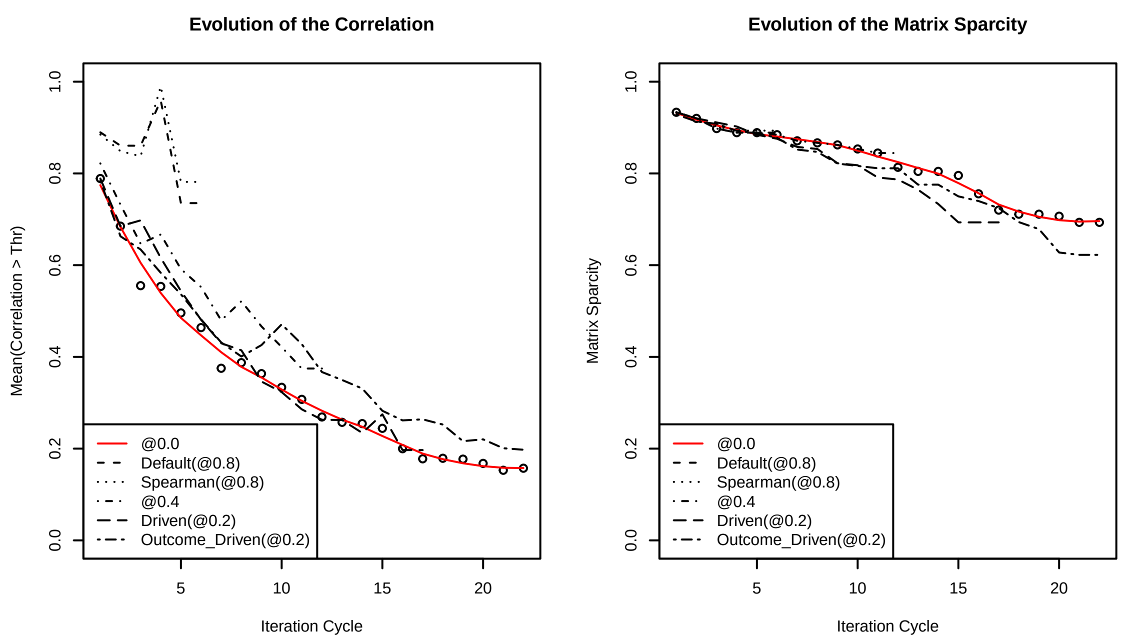
	Hip	Height
Default	0.1096	0.0209
Spearman	0.1099	0.0221
At_40	0.1096	0.0209
At_0	0.0993	0.0209
BMI_BF	0.0779	0.0209
BodyFat_Driven	0.2344	0.0210

3.2.2 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="Mean(Correlation > Thr)",
     ylim=c(0,1.0),
     main="Evolution of the 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"))
```

```r
# 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

```r
# 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"))
```

Table 4: Table continues below

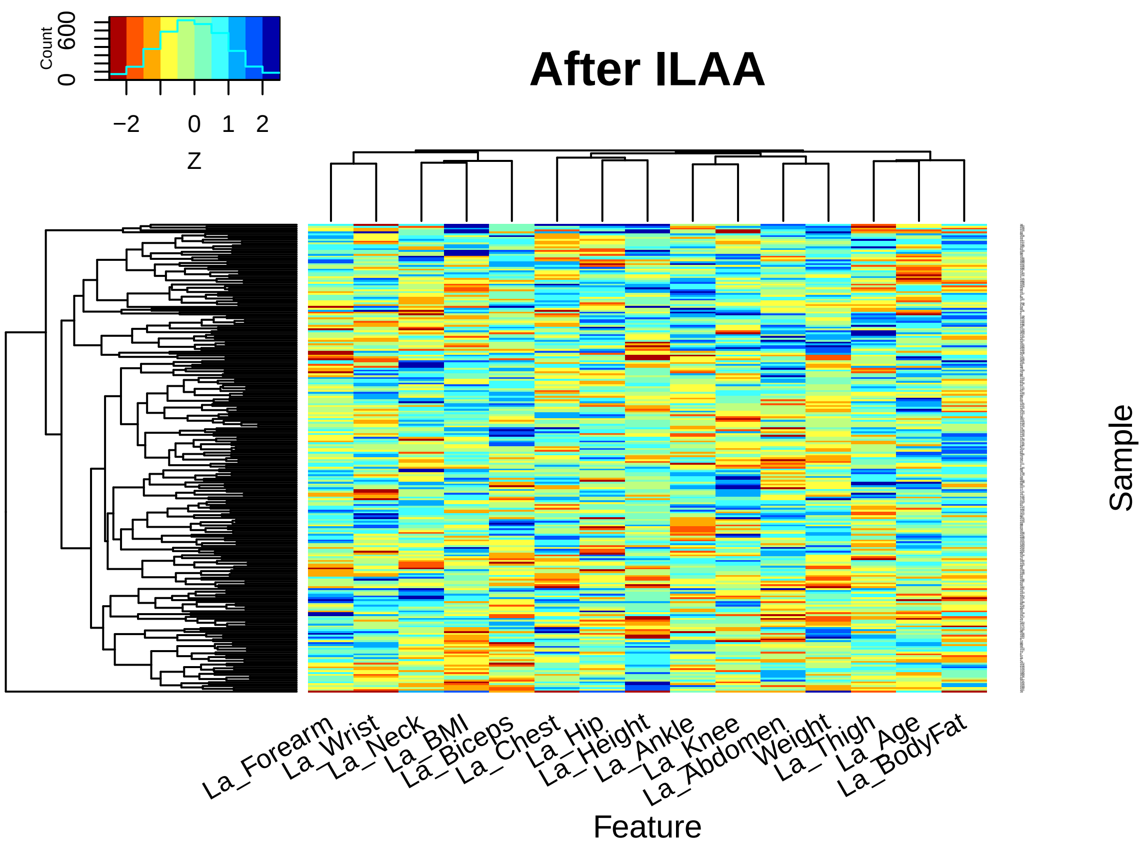
Weight	La_Ankle	La_Forearm	La_Age	La_Wrist	La_Biceps	La_BodyFat
1	0.555	0.518	0.483	0.403	0.32	0.29

La_Neck	La_Knee	La_BMI	La_Thigh	La_Abdomen	La_Chest	La_Hip	La_Height
0.279	0.228	0.211	0.183	0.132	0.104	0.0993	0.0209

3.3 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)
```

3.4 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.

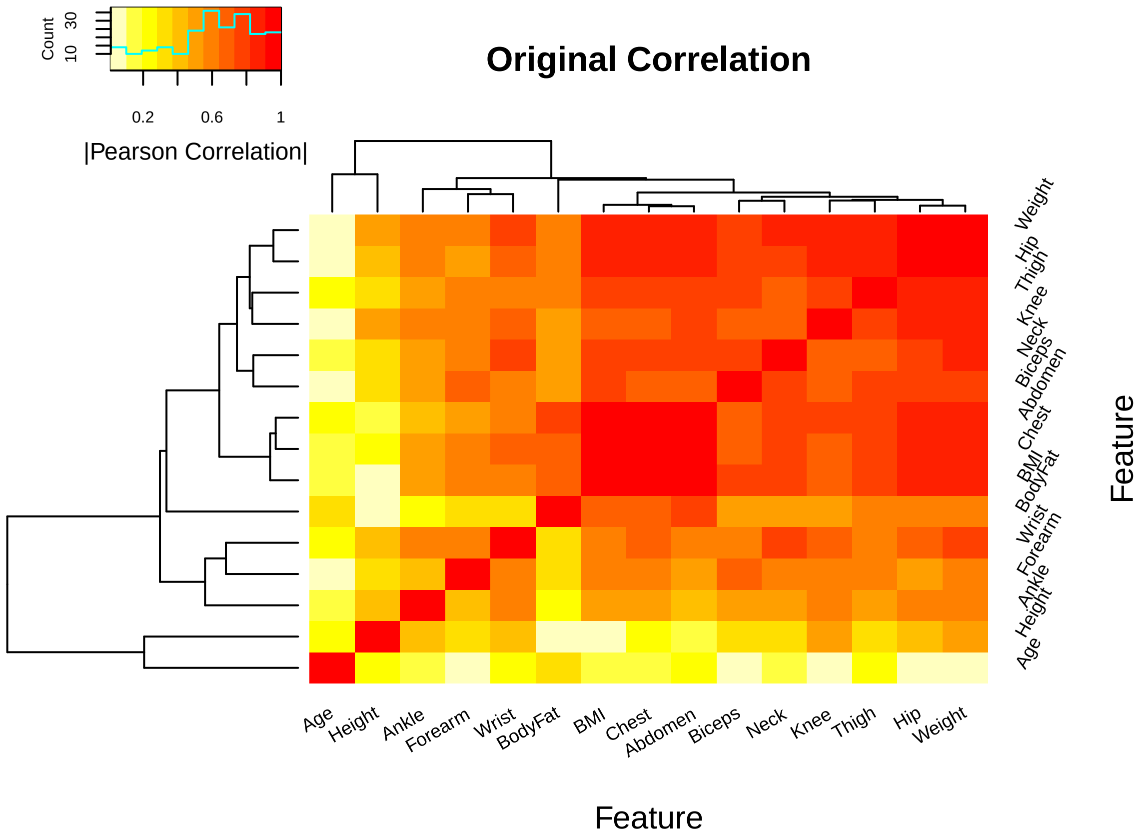
The "`IDeAEvolution`" attribute can be used to verify if the algorithm achieved the target correlation goal, and the sparsity of the returned matrix.

3.5 The ILAA Transformed Data

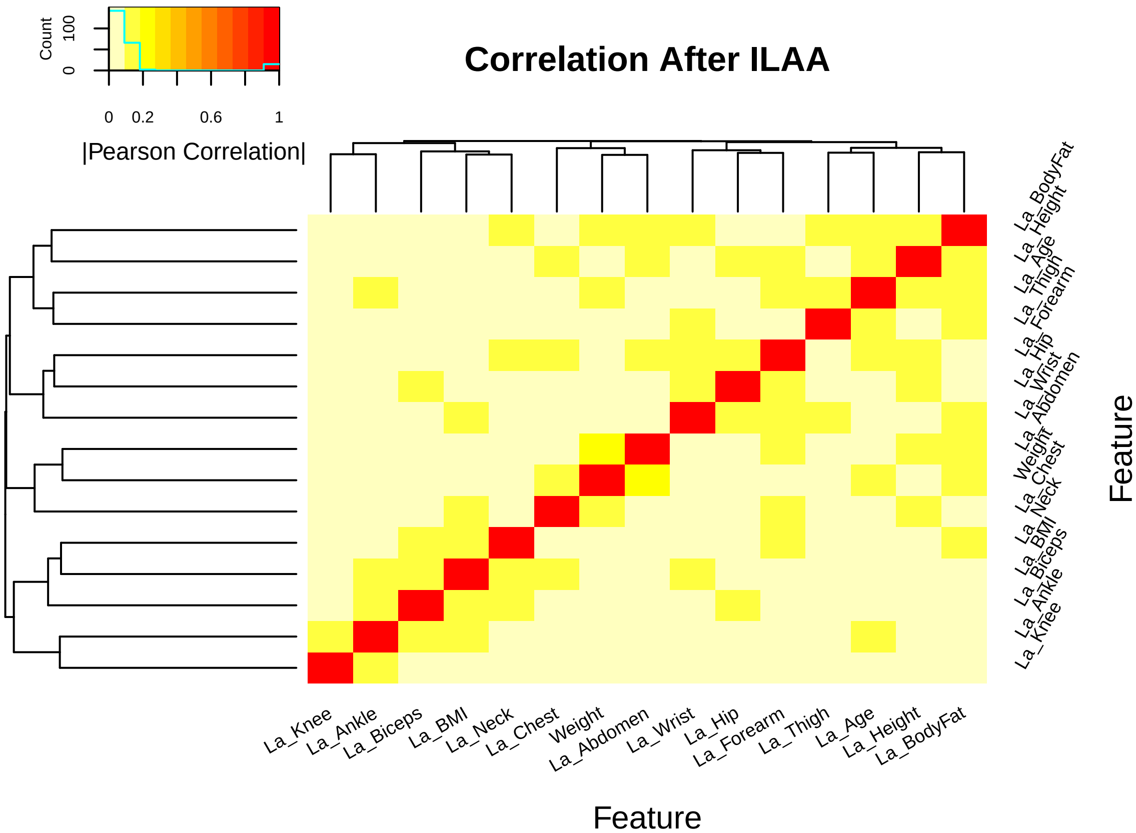
Before exploring into more detail, the properties of the `ILAA` results. Let us first verify that the returned matrix does not contain features with very high correlation among them.

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")
```



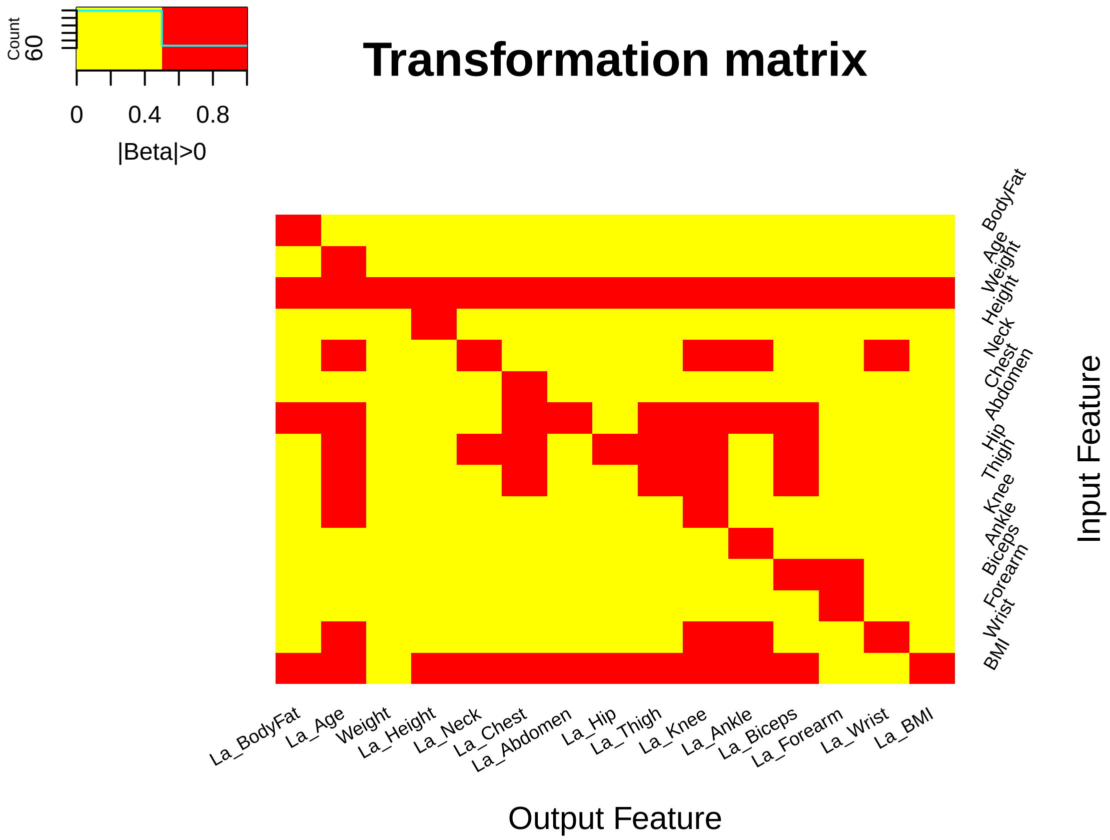

```r
# The transformed
  cormat <- cor(body_fat_Decorrelated,method="pearson")
  gplots::heatmap.2(abs(cormat),
                    trace = "none",
                    mar = c(5,5),
                    col=rev(heat.colors(11)),
                    main = "Correlation After ILAA",
                   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



3.6 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")
```



3.7 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

3.8 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:

$$LaWrist = Wrist - 0.012Weight - 0.165Neck.$$

  Describes that the $Wrist$ is associated with the $Weight$ and $Neck$. The latent variable $LaWrist$ is the amount of information in the $Wrist$ not found by $Weight$ nor the $Neck$.

- Therefore, the model of the $Wrist$ is :

$$Wrist = +0.012Weight + 0.165Neck + b_o,$$

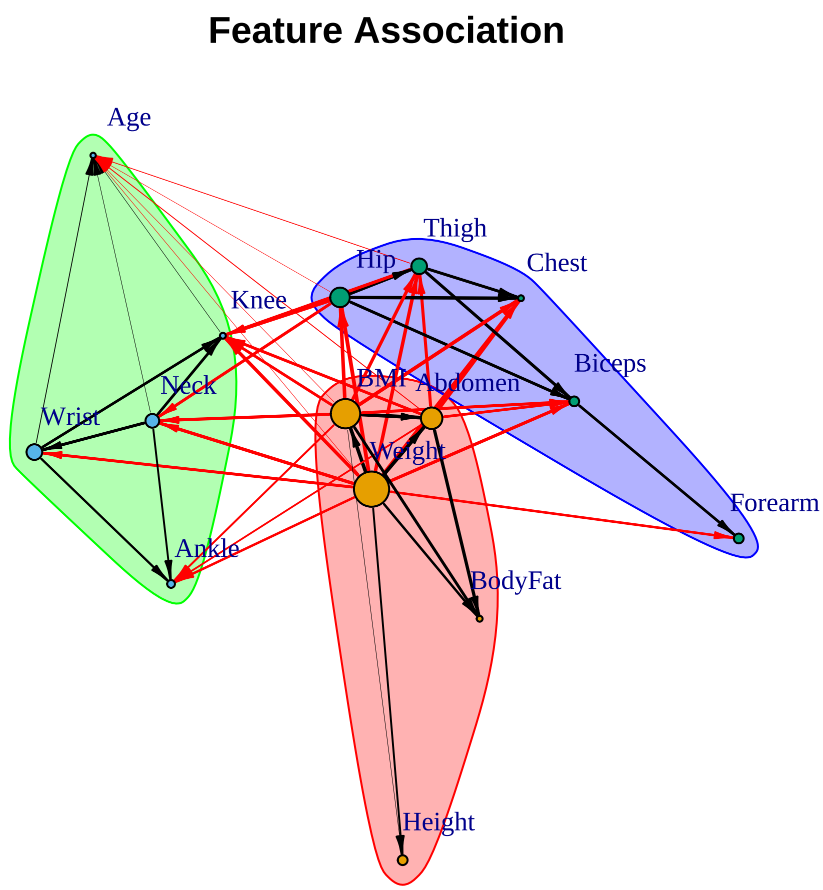
where b_o is the bias term. It can be estimated using the difference between the mean of the raw observations and the mean of the model.

3.9 The Formula Network

The `graph_from_adjacency_matrix()` function from `igraph` can be used to visualize the association between variables.

```r
par(op)

transform <- attr(body_fat_Decorrelated,"UPLTM") != 0
colnames(transform) <- str_remove_all(colnames(transform),"La_")
# The weights are proportional to the observed correlation
transform <- abs(transform*cor(body_fat[,rownames(transform)]))


# The size depends on the variable independence  relevance (fscore)

VertexSize <- attr(body_fat_Decorrelated,"fscore")

names(VertexSize) <- str_remove_all(names(VertexSize),"La_")
# Normalization
VertexSize <- 10*(VertexSize-min(VertexSize))/(max(VertexSize)-min(VertexSize))
```

```r
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")
```




```r
par(op)
```

3.9.1 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.

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

```r
## 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"))
```

Table 7: Table continues below

Weight	La_Ankle	La_Forearm	La_Age	La_Wrist	La_Biceps	La_BodyFat
1	0.551	0.517	0.484	0.402	0.32	0.291

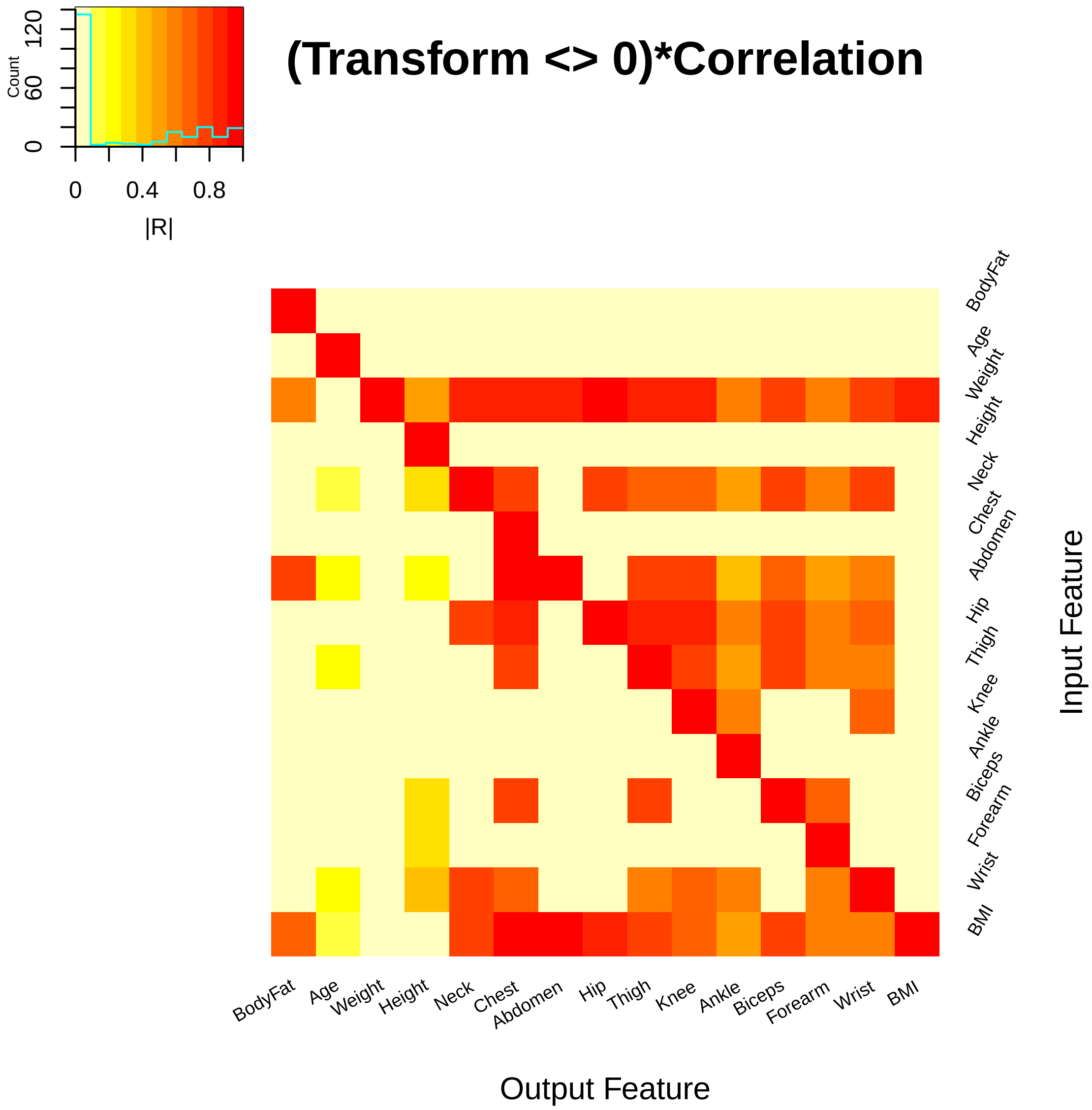
La_Neck	La_Knee	La_BMI	La_Thigh	La_Abdomen	La_Chest	La_Hip	La_Height
0.276	0.241	0.211	0.182	0.13	0.103	0.0992	0.0208

```r
## Getting the formulas
LatentFormulas <- getLatentCoefficients(body_fat_Decorrelated)
charFormulas <- attr(LatentFormulas,"LatentCharFormulas")
pander::pander(as.matrix(charFormulas))
```

La_BodyFat	+ BodyFat + (0.119)Weight - (0.795)Abdomen - (0.492)BMI
La_Age	+ Age + (0.365)Weight - (0.678)Neck - (1.136)Abdomen - (0.115)Hip + (2.159)Thigh - (1.331)Knee - (5.499)Wrist - (0.571)BMI
La_Height	- (0.192)Weight + Height - (1.24e-05)Neck - (1.22e-04)Abdomen + (6.60e-04)Biceps - (1.91e-03)Forearm + (7.88e-05)Wrist + (1.342)BMI
La_Neck	- (0.098)Weight + Neck + (0.170)Hip - (3.87e-03)Wrist - (0.093)BMI
La_Chest	- (0.142)Weight - (1.30e-04)Neck + Chest - (0.371)Abdomen + (0.457)Hip + (0.195)Thigh - (9.67e-04)Biceps + (7.56e-04)Wrist - (1.023)BMI
La_Abdomen	- (0.102)Weight + Abdomen - (1.850)BMI
La_Hip	- (0.182)Weight + (1.37e-03)Neck + Hip - (0.419)BMI
La_Thigh	- (0.054)Weight - (3.69e-04)Neck + (0.136)Abdomen - (0.499)Hip + Thigh - (3.37e-03)Biceps + (2.15e-03)Wrist - (0.253)BMI
La_Knee	- (0.060)Weight + (0.035)Neck - (8.58e-03)Abdomen - (0.027)Hip - (0.062)Thigh + Knee - (0.210)Wrist + (0.098)BMI
La_Ankle	- (0.032)Weight + (0.094)Neck + (0.051)Abdomen + (1.08e-04)Hip + (2.26e-04)Thigh - (0.032)Knee + Ankle - (0.575)Wrist - (0.094)BMI
La_Biceps	- (0.081)Weight - (3.95e-03)Neck + (0.067)Abdomen + (0.096)Hip - (0.194)Thigh + Biceps - (0.124)BMI

La_Forearm	- (0.018)Weight + (2.97e-03)Neck + (2.45e-03)Abdomen - (5.05e-03)Hip + (0.010)Thigh - (0.321)Biceps + Forearm - (0.018)Wrist - (4.53e-03)BMI
La_Wrist	- (0.012)Weight - (0.163)Neck + (2.73e-04)Abdomen - (1.02e-03)Hip + (1.79e-03)Thigh - (3.05e-04)Knee + Wrist - (5.04e-04)BMI
La_BMI	- (0.110)Weight + BMI

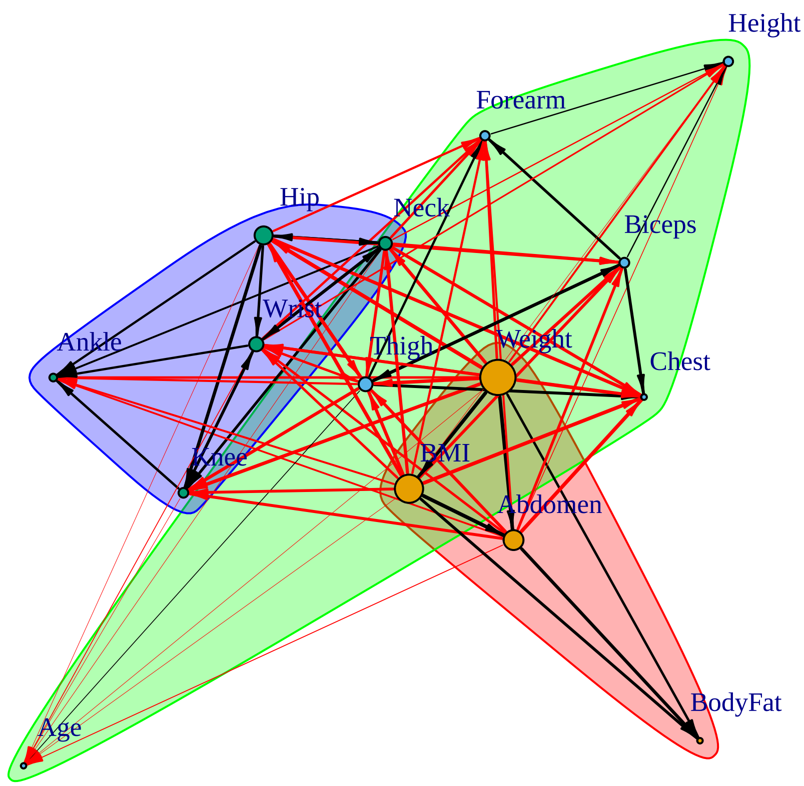
```r
## The transformation
par(op)

# The non-zero coefficients
transform <- attr(body_fat_Decorrelated,"UPLTM") != 0
# For network analysis
colnames(transform) <- str_remove_all(colnames(transform),"La_")
# The weights are proportional to the observed correlation
transform <- abs(transform*cor(body_fat[,rownames(transform)]))


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)
```



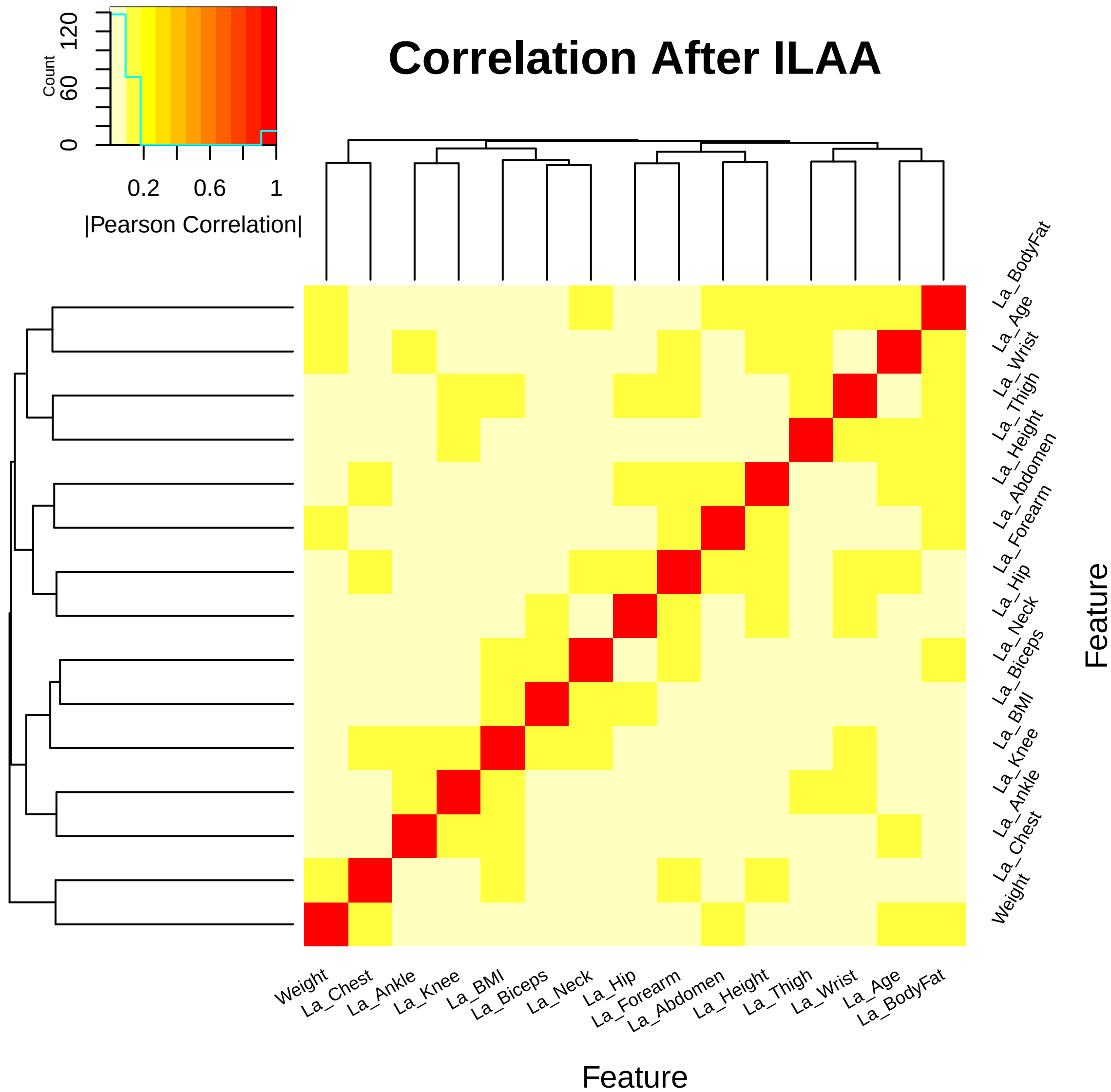

```r
## Network analysis
# The vertex size will be proportional to the fscore of the IDeA procedure.

# The size depends on the variable independence relevance (fscore)
VertexSize <- attr(body_fat_Decorrelated,"fscore")
# Normalization
VertexSize <- 10*(VertexSize-min(VertexSize))/(max(VertexSize)-min(VertexSize))



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")
```

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)))
```

0.166

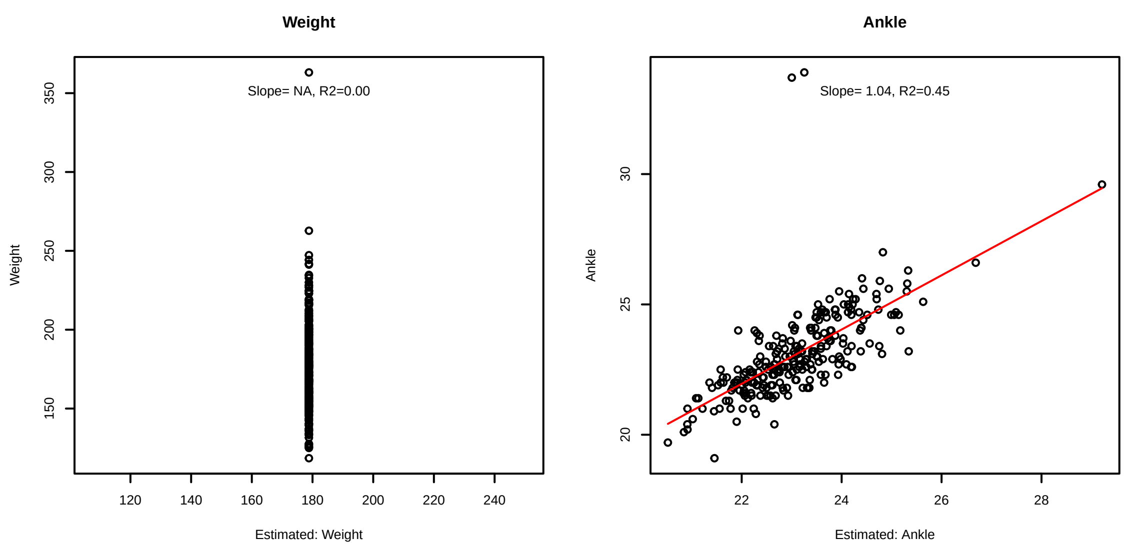
3.9.2 Association Plots

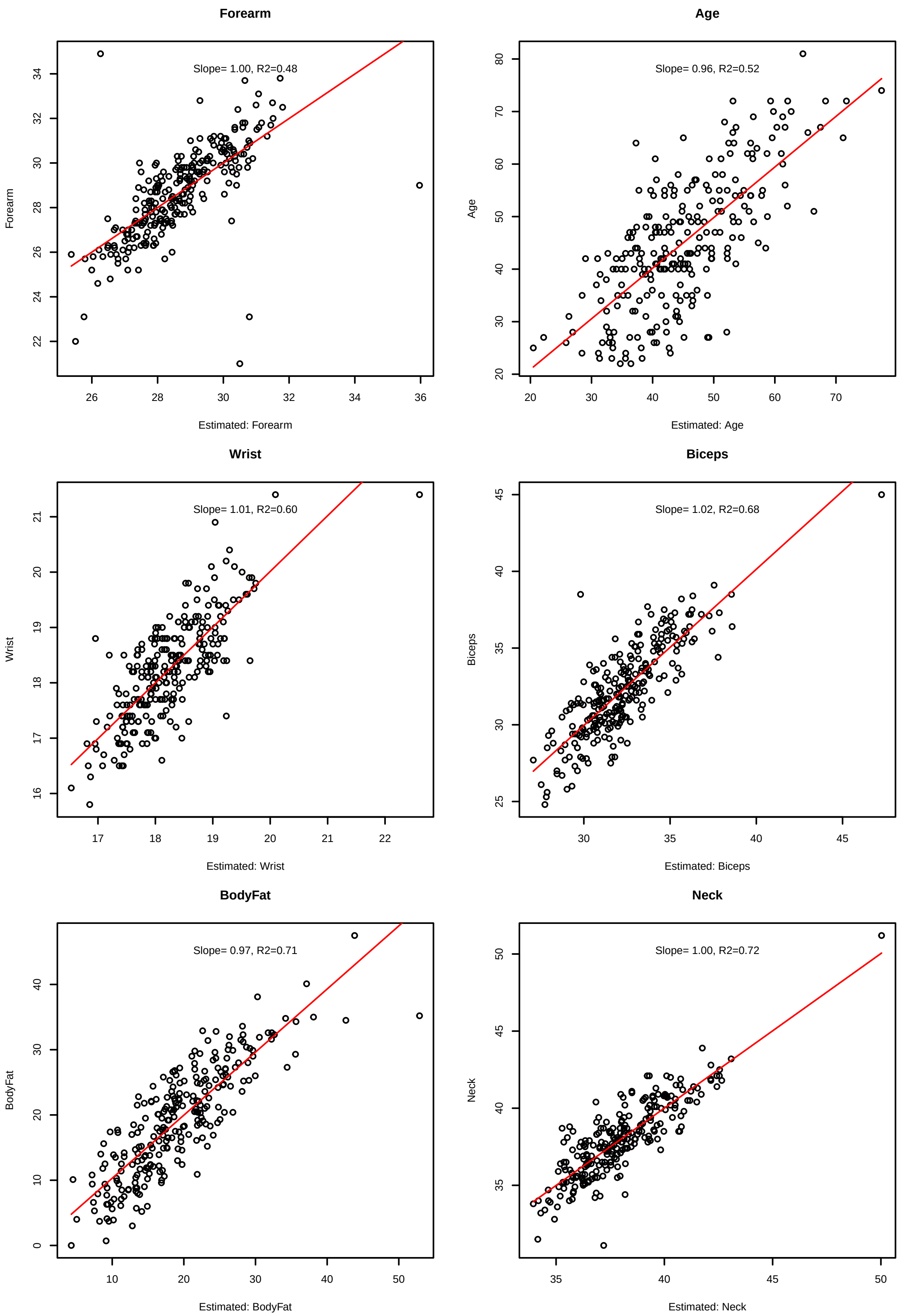
3.9.2.1 Direct Transform Estimation The transformation matrix can be used to get estimation of each variable from the latent models.

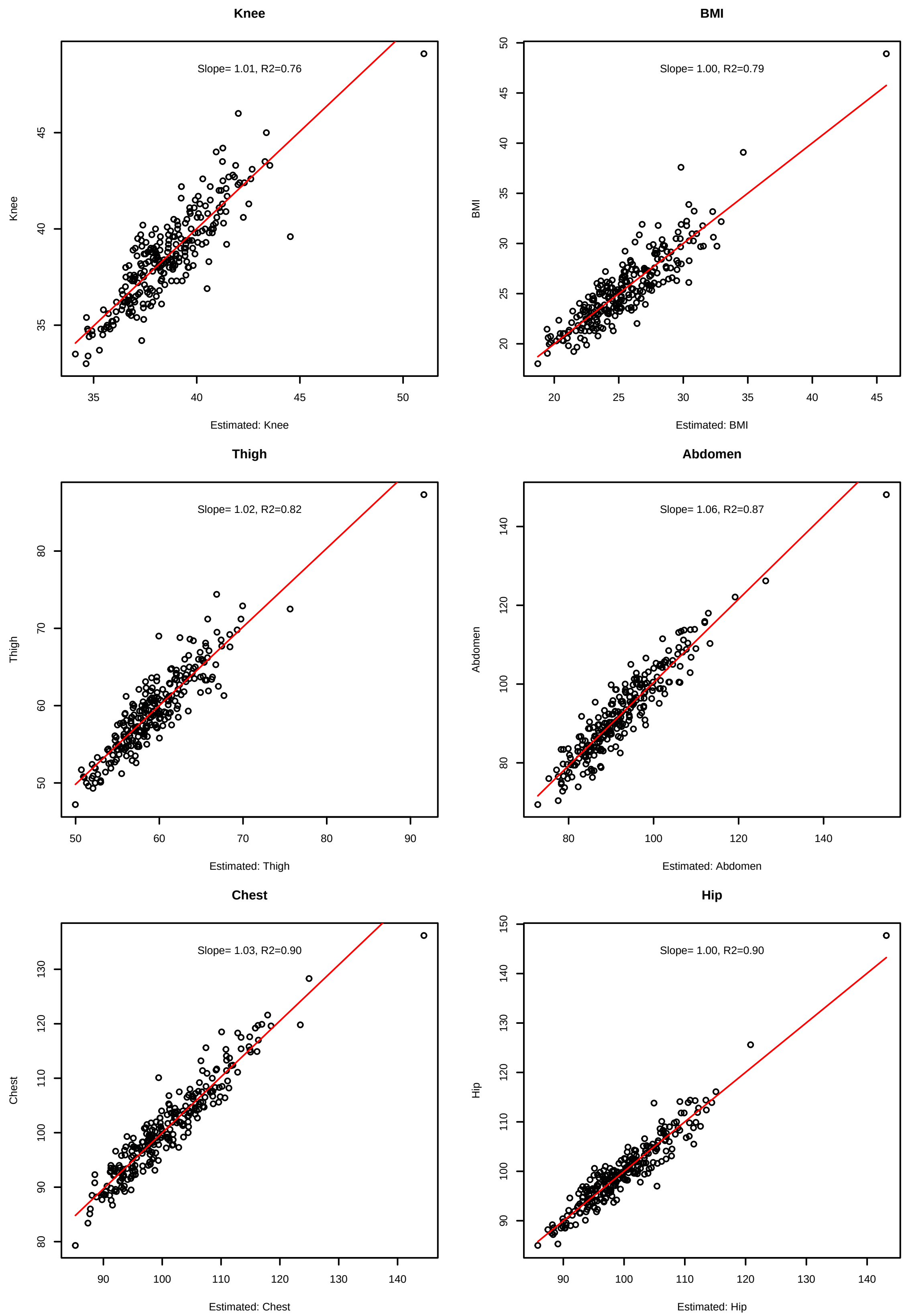
To to this just set the diagonal of the transformation to zero, then rotate the input matrix, multiply by -1, and the the output is the estimated observation from the independent variables. The bias term is estimated by computing the observed mean minus the transformed mean.

```
transform <- attr(body_fat_Decorrelated,"UPLTM")
varratio <- attr(body_fat_Decorrelated,"VarRatio")
# Set the diagonal to zero
diag(transform) <- 0
```

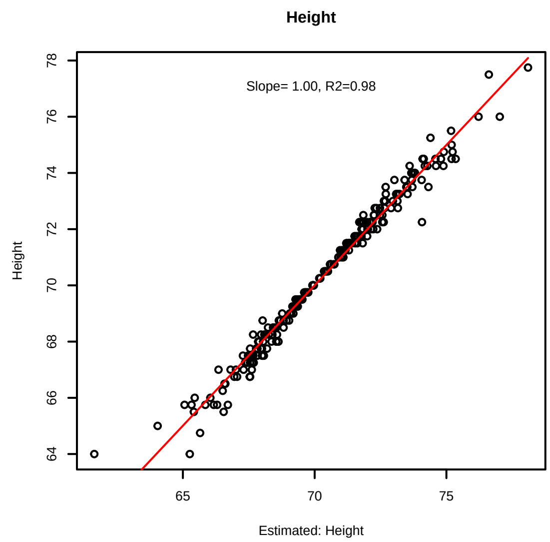
```
#Estimating the observation
obsestim <- -1*as.data.frame(
  as.matrix(body_fat[,rownames(transform)]) %*% transform)

#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])
     )
}
```




Forearm
Slope= 1.00, R2=0.48
Forearm
Estimated: Forearm
Age
Slope= 0.96, R2=0.52
Age
Estimated: Age
Wrist
Slope= 1.01, R2=0.60
Wrist
Estimated: Wrist
Biceps
Slope= 1.02, R2=0.68
Biceps
Estimated: Biceps
BodyFat
Slope= 0.97, R2=0.71
BodyFat
Estimated: BodyFat
Neck
Slope= 1.00, R2=0.72
Neck
Estimated: Neck


Knee
Slope= 1.01, R2=0.76
Knee
Estimated: Knee
BMI
Slope= 1.00, R2=0.79
BMI
Estimated: BMI
Thigh
Slope= 1.02, R2=0.82
Thigh
Estimated: Thigh
Abdomen
Slope= 1.06, R2=0.87
Abdomen
Estimated: Abdomen
Chest
Slope= 1.03, R2=0.90
Chest
Estimated: Chest
Hip
Slope= 1.00, R2=0.90
Hip
Estimated: Hip

```r
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.

4 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%)

4.1 Split into Training Testing Sets

```r
# 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,]
```

4.2 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.

```r
## 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)
```

4.3 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 *BodyFat* 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")
```

Table 10: 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")
```

Table 11: 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

```r
## Outcome-Driven
modelBodyFatD <- LASSO_MIN(BodyFat~.,body_fat_Decorrelated_trainD)
pander::pander(as.matrix(modelBodyFatD$coef),caption="Outcome-Driven Coefficients")
```

Table 12: Outcome-Driven Coefficients

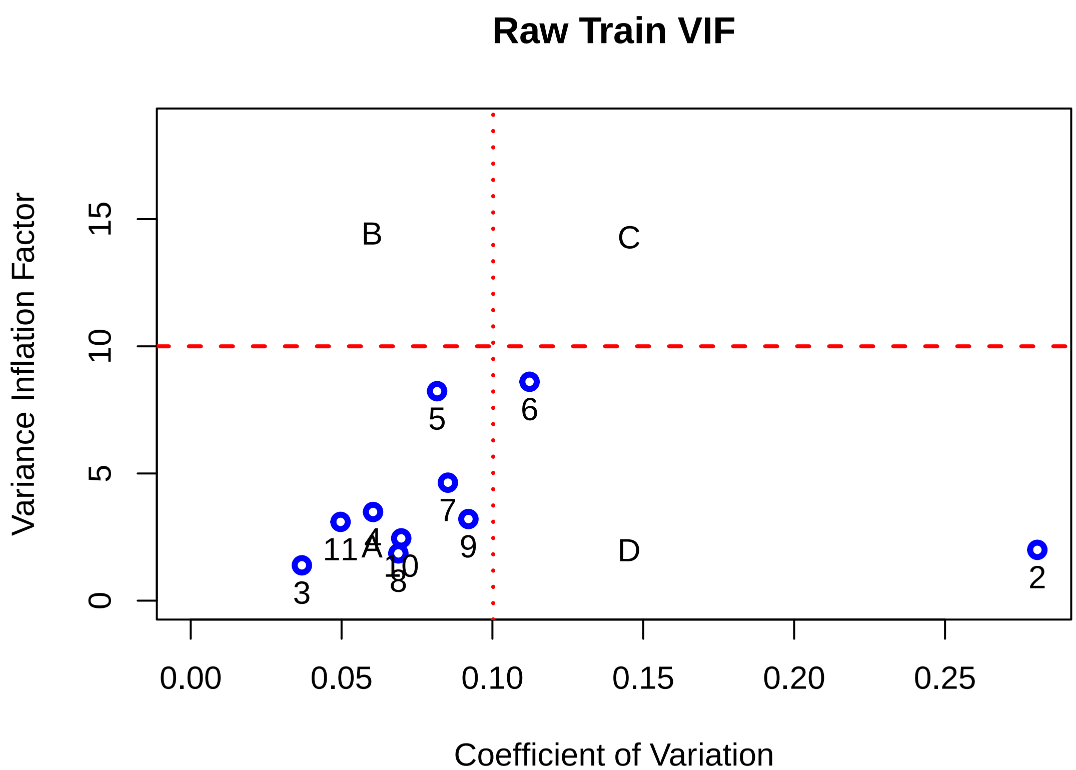
(Intercept)	-29.7164
La_Age	0.0172
La_Weight	-0.0989
La_Neck	-0.5088
La_Chest	-0.1315
Abdomen	0.6441
La_Hip	-0.1942
La_Thigh	0.1177
La_Ankle	0.0920
La_Biceps	0.1394
La_Wrist	-0.5890

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

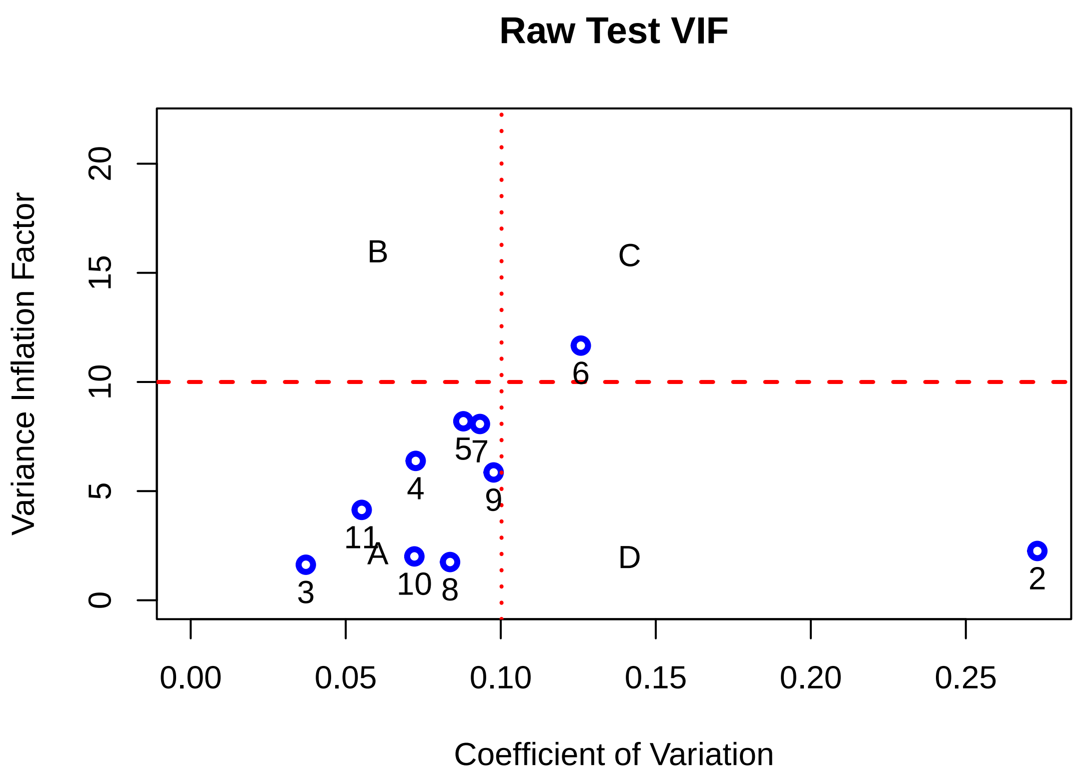
4.3.0.1 Multicollinear Analysis

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

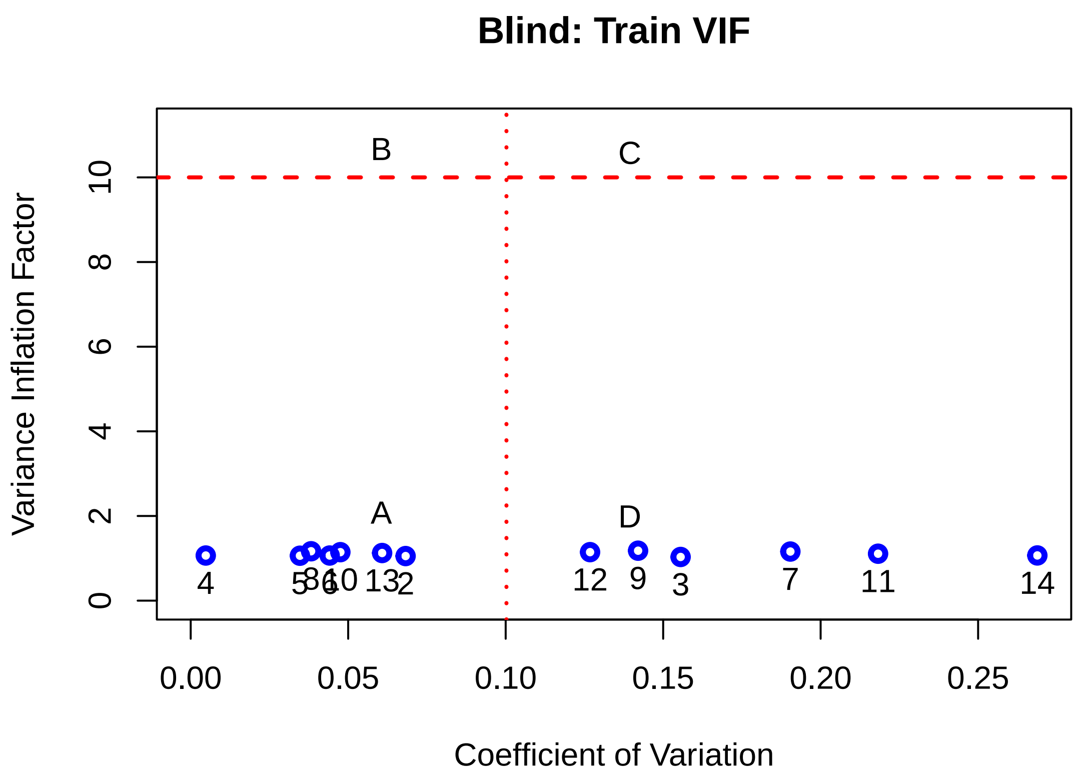
```r
frm <- paste("BodyFat~",str_flatten(modelBodyFatRaw$selectedfeatures," + "))

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



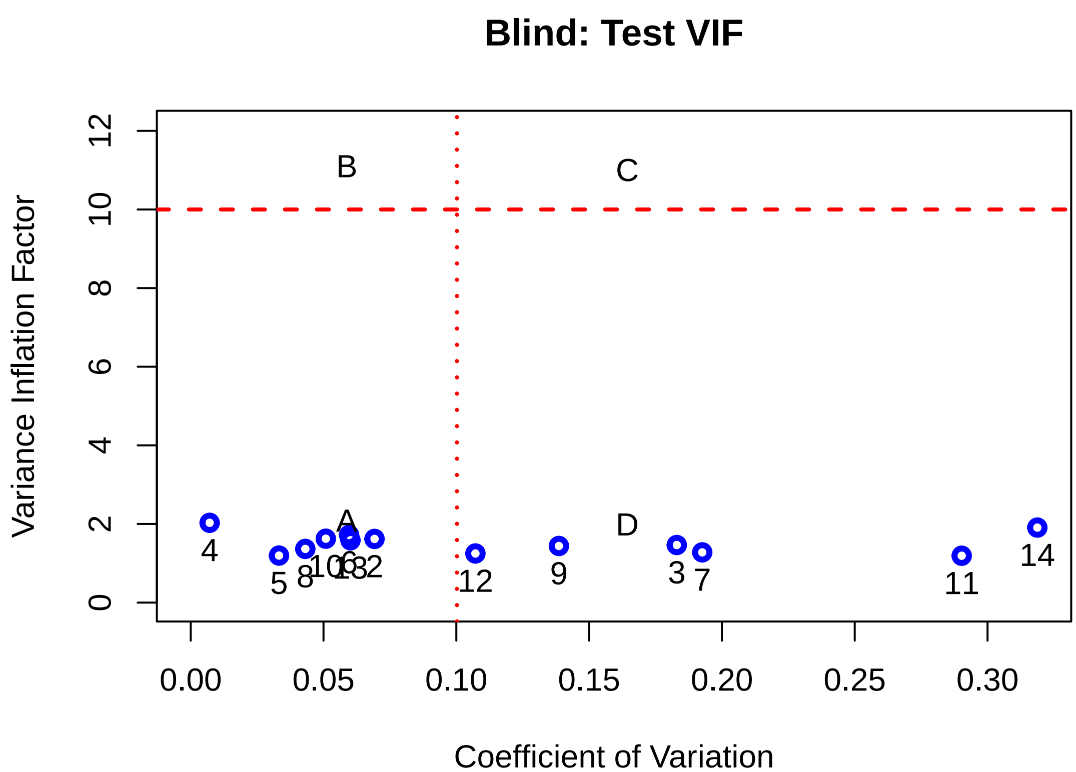

```
X <- model.matrix(formula(frm),testingset);
mc <- multiCol(X)
vifd <- VIF(X)
vifx <-vif(lm(formula(frm),testingset))
title("Raw Test VIF")
```



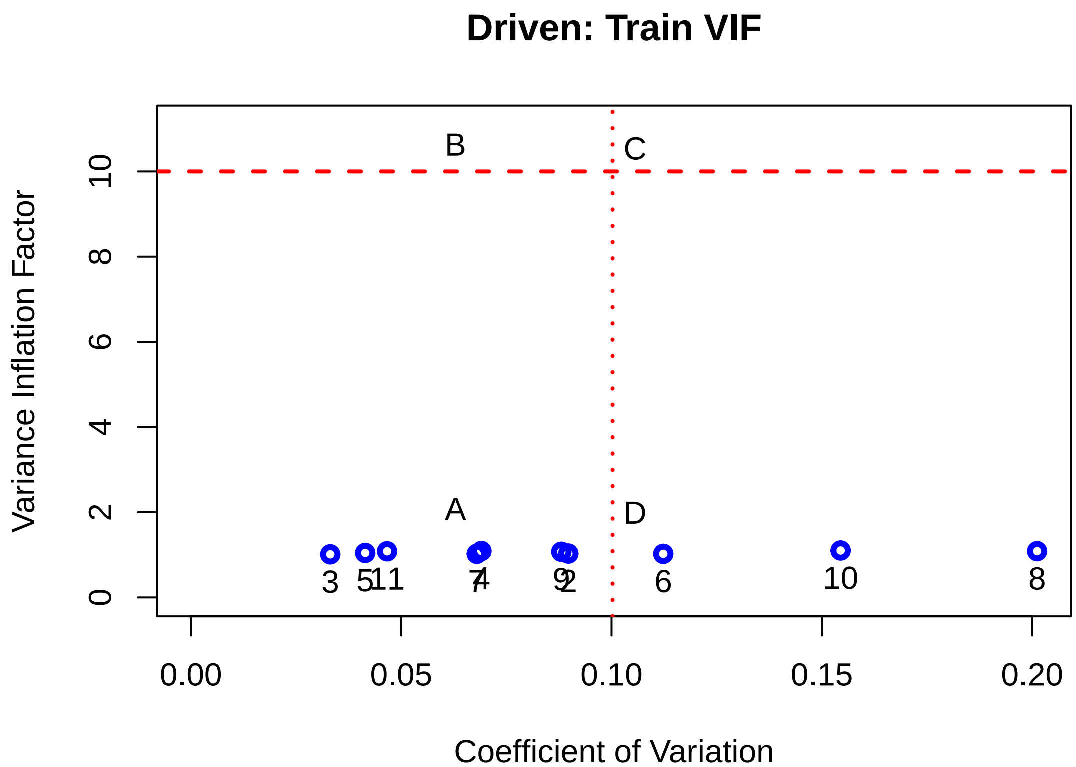

```
frm <- paste("BodyFat~",str_flatten(modelBodyFat$selectedfeatures," + "))

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")
```



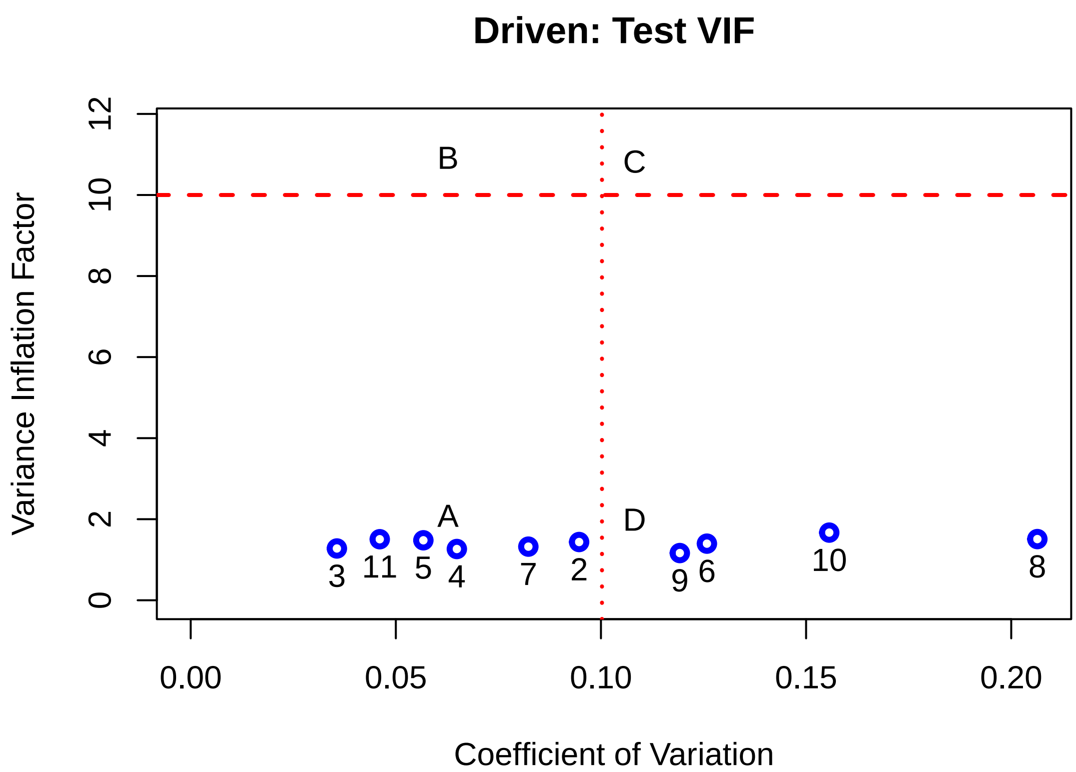

```
X <- model.matrix(formula(frm),body_fat_Decorrelated_test);
mc <- multiCol(X)
title("Blind: Test VIF")
```



```
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

4.3.1 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")
```

Table 13: 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

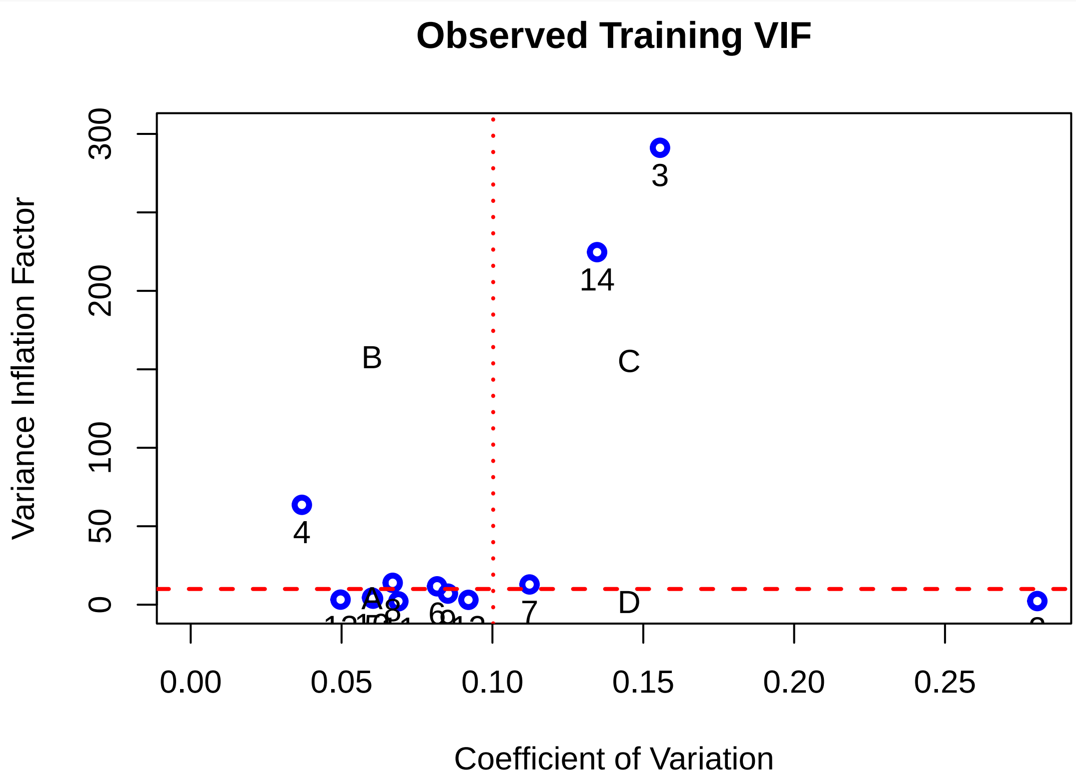
```r
# The outcome-driven coefficients
observedCoefD <- getObservedCoef(body_fat_Decorrelated_trainD,modelBodyFatD)
pander::pander(as.matrix(observedCoefD$coefficients),caption="Driven Coefficients")
```

Table 14: Driven Coefficients

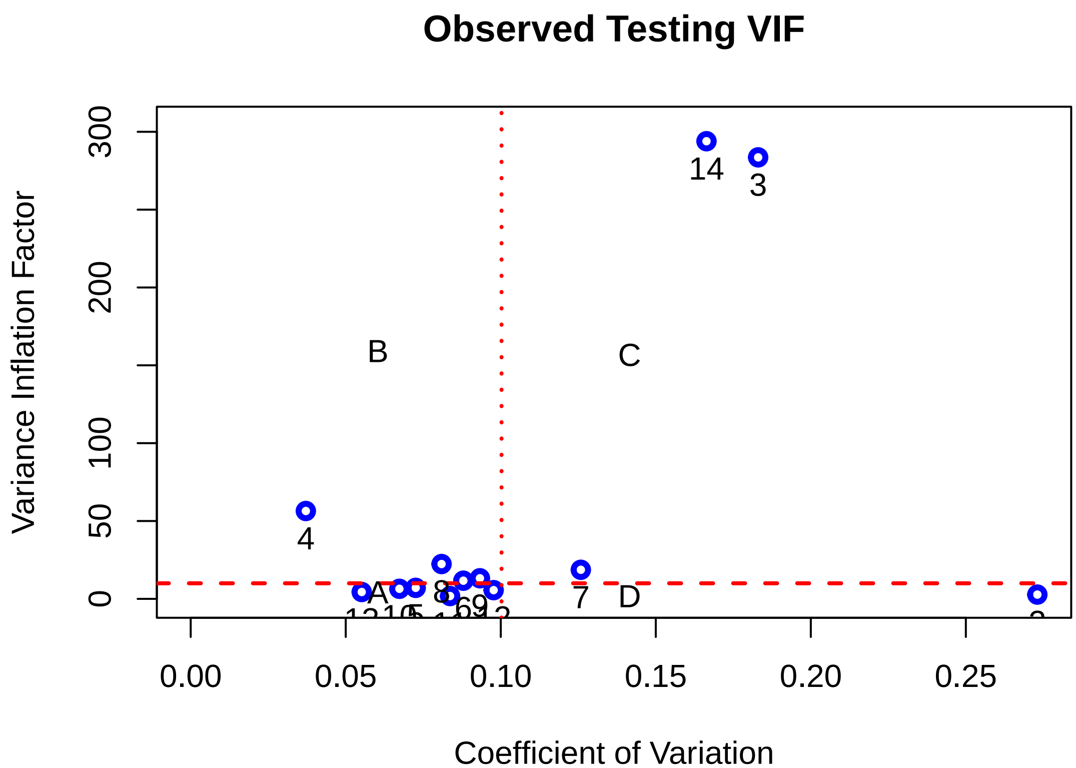
(Intercept)	-29.7164
Age	0.0172
Weight	-0.0790
Neck	-0.1669
Chest	-0.1315
Abdomen	0.8662
Hip	-0.0162
Thigh	0.1124
Knee	-0.0417
Ankle	0.0920
Biceps	0.1394
Wrist	-0.7604
BMI	0.2150

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

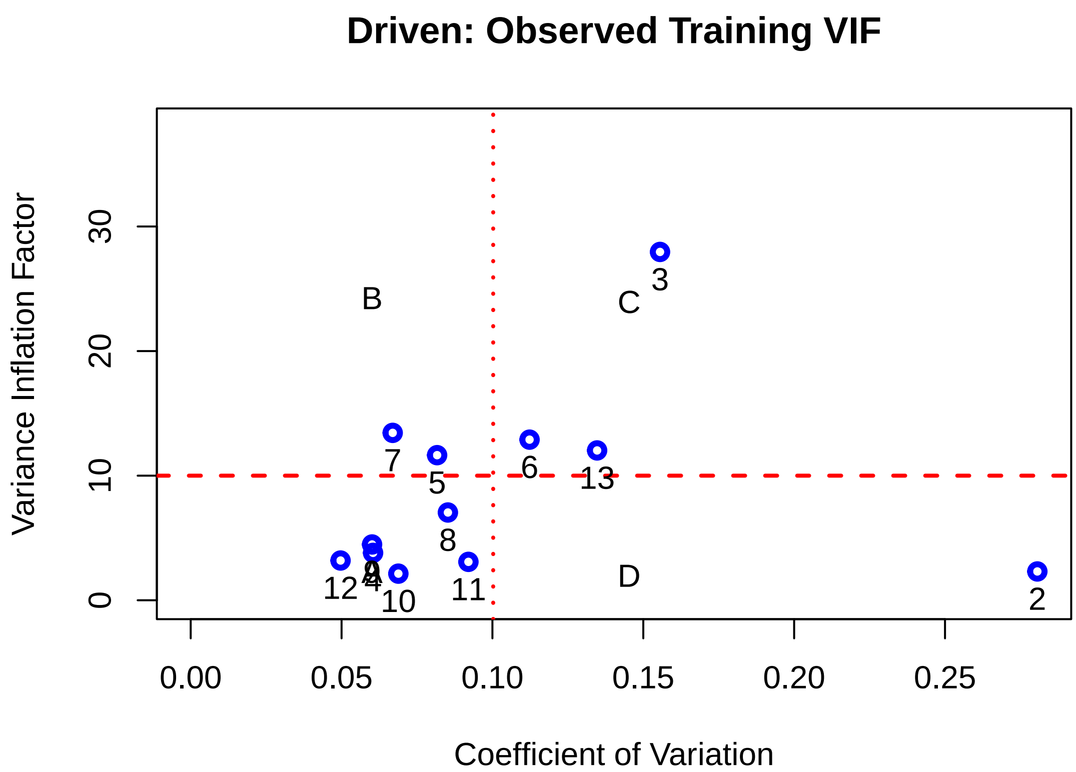
```r
X <- model.matrix(formula(observedCoef$formula),trainingset);
mc <- multiCol(X)
title("Observed Training VIF")
```



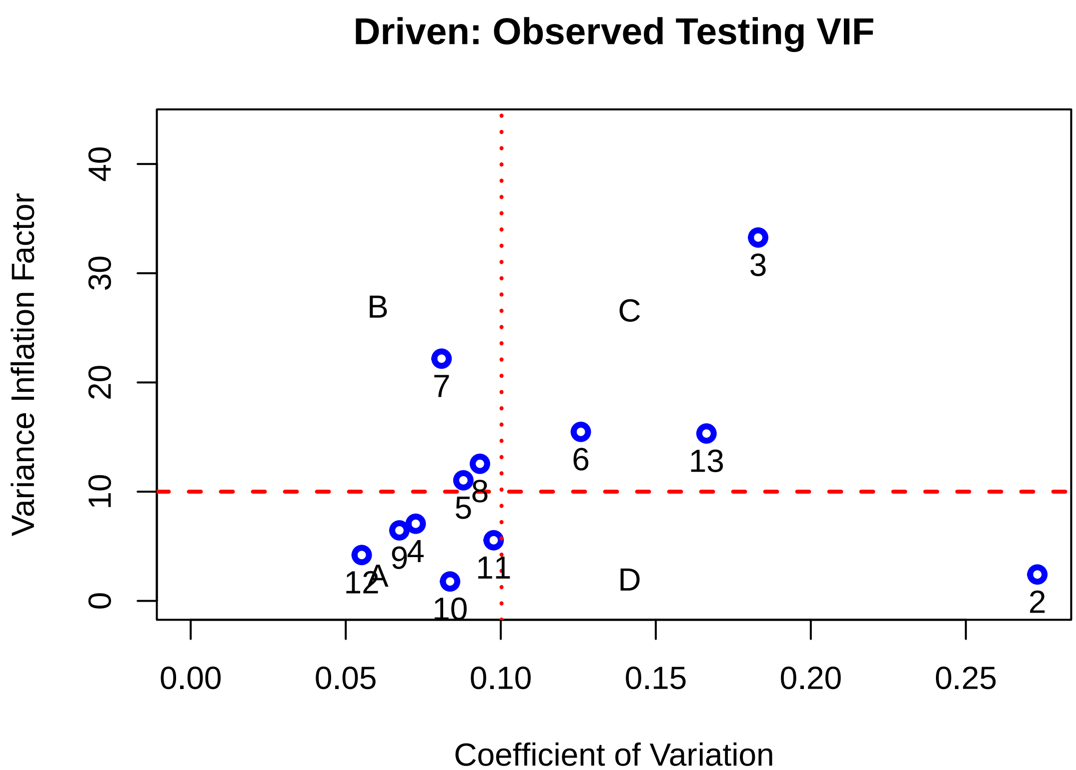
```r
X <- model.matrix(formula(observedCoef$formula),testingset);
mc <- multiCol(X)
title("Observed Testing VIF")
```




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




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



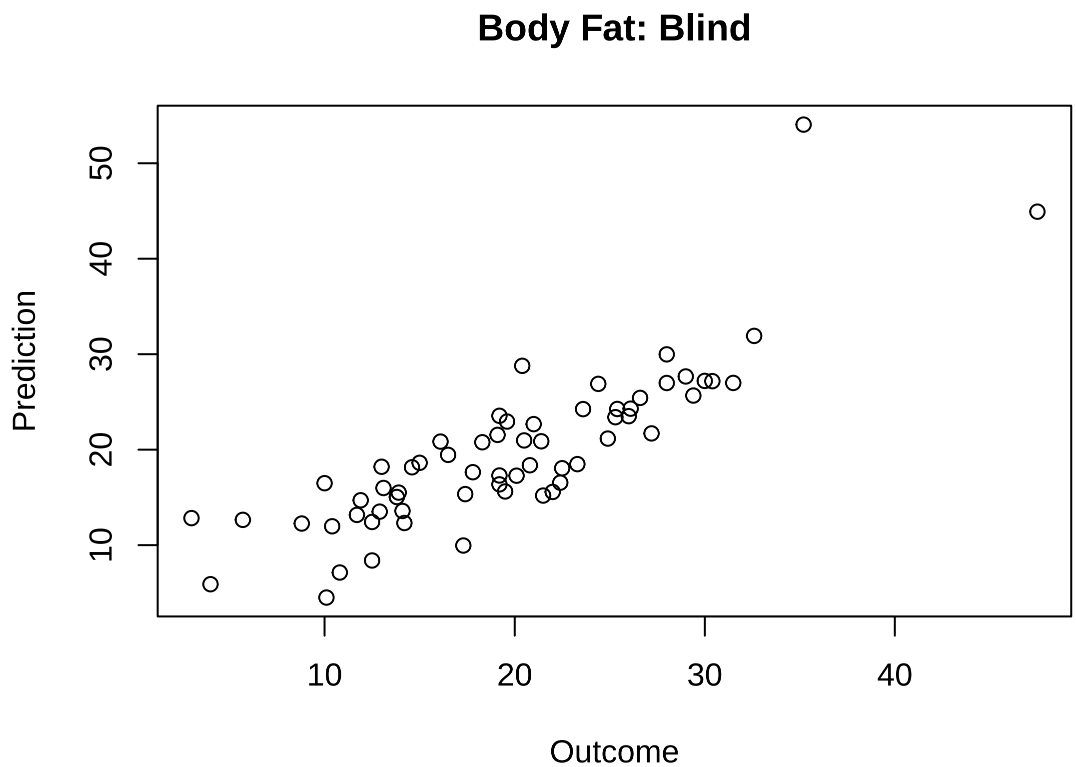
The results indicate that the models created using the observed variables have strong collinearity issues.

4.3.2 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

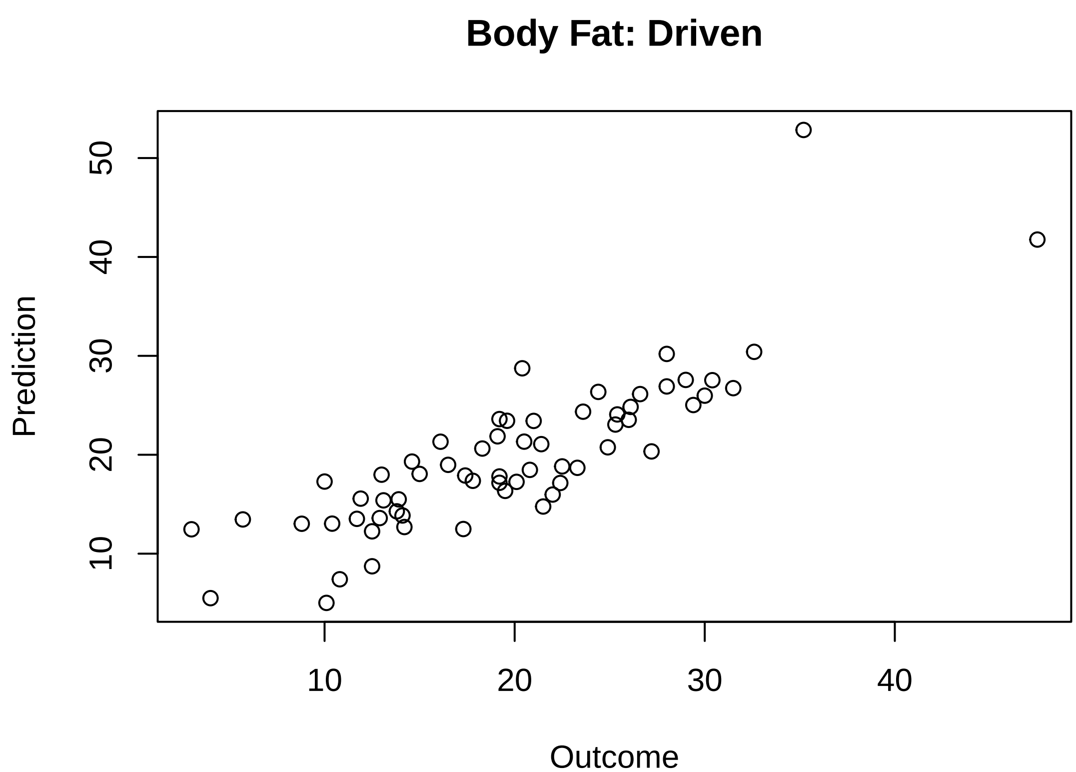
- **pearson**:

Table 18: 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**:

Table 22: Pearson's product-moment correlation: `predictions[, 1]` and `predictions[, 2]`

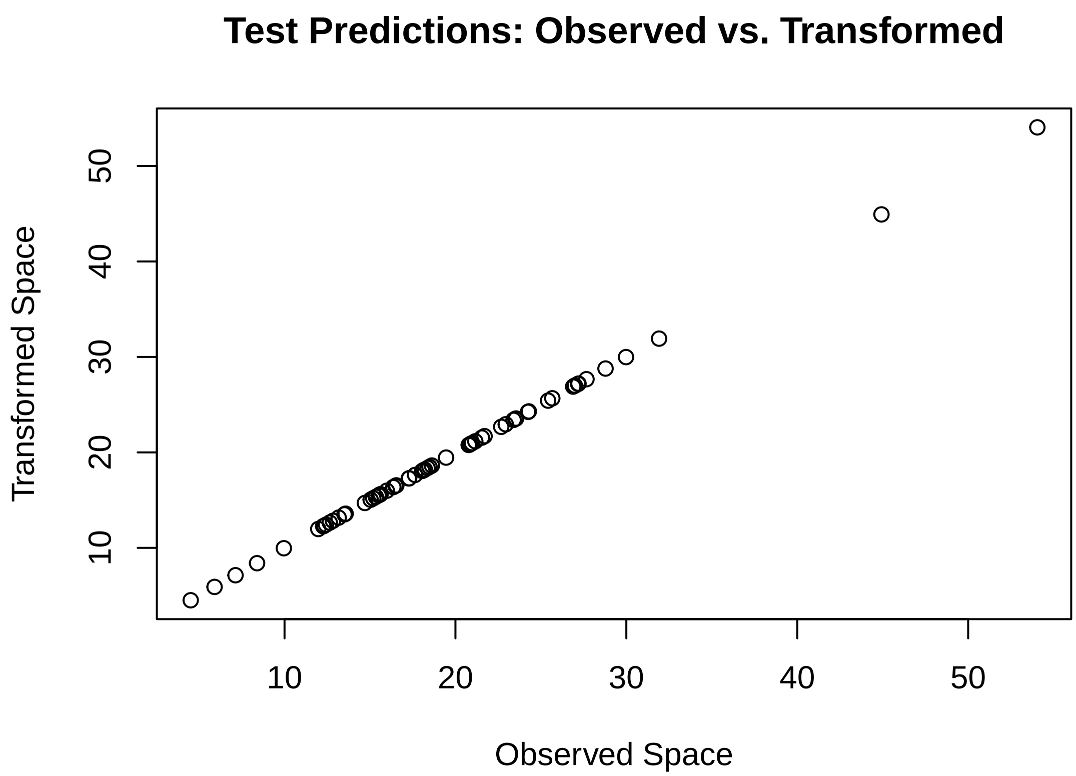
Test statistic	df	P value	Alternative hypothesis	cor
12.2	61	5.56e-18 * * *	two.sided	0.842

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

4.3.3 Prediction Using the Observed Features

An ILAA user has the option to predict the $BodyFat$ 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.

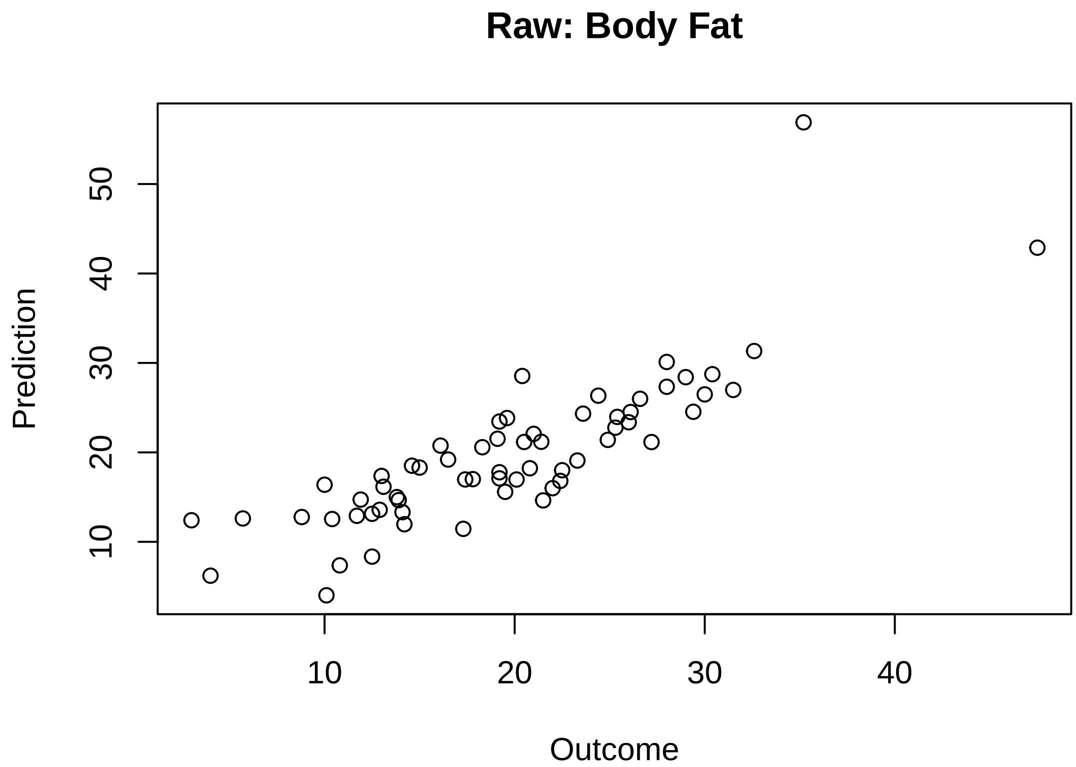
4.3.4 Comparison to Raw Model

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




```r
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**:

Table 26: 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: ($\rho_t = 0.863$ vs. $\rho_o = 0.875$)

4.3.5 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.

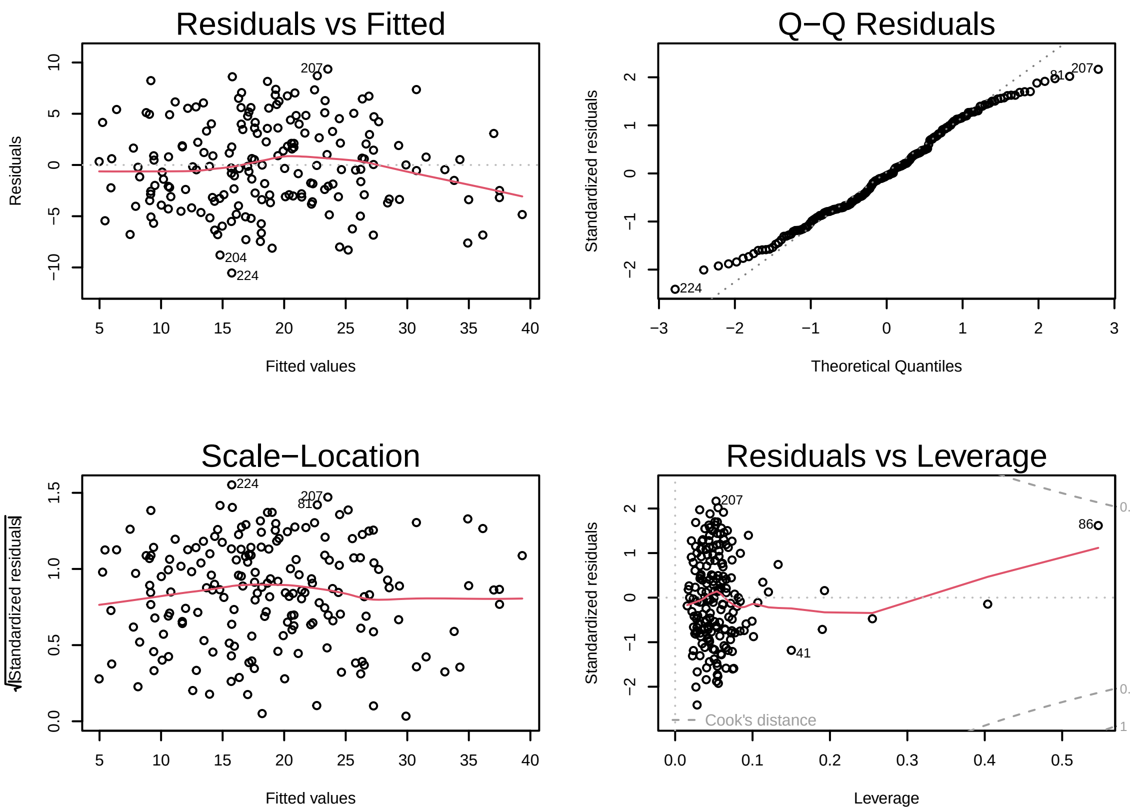
The inspection of the summary results clearly shows that most of the beta coefficients on the transformed data set are significant.

```
## 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)
```

Table 27: 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)
```

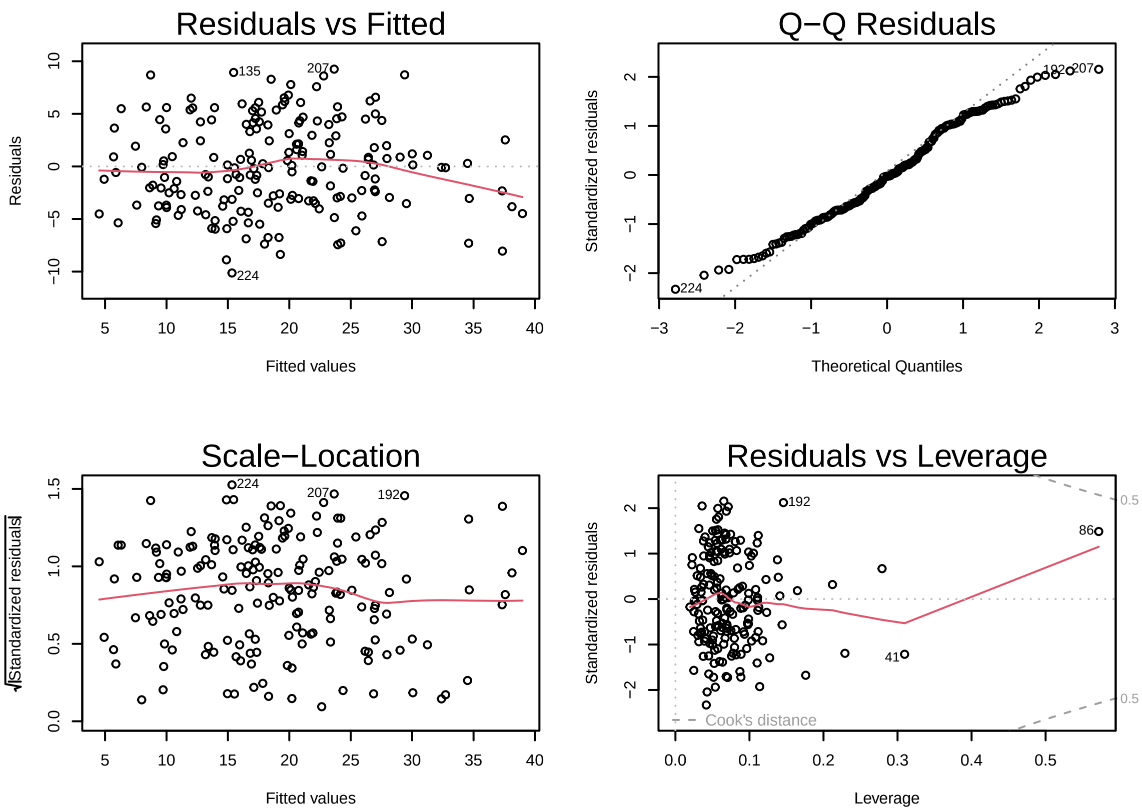



```r
## Outcome-Blind
par(mfrow=c(2,2),cex=0.5)
Delm <- lm(BodyFat~.,body_fat_Decorrelated_train[,c("BodyFat",names(modelBodyFat$coef)[-1])])
pander::pander(Delm,add.significance.stars=TRUE)
```

Table 28: 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	* * *

```r
plot(Delm)
```

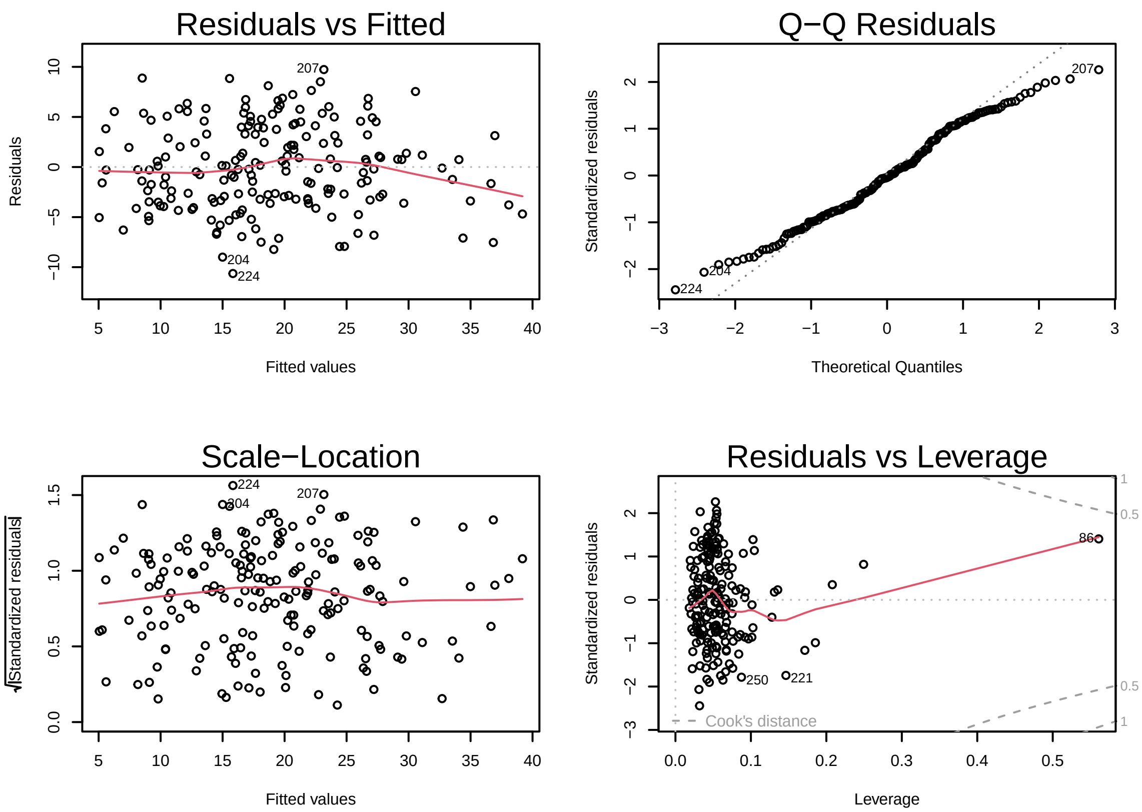



```r
## Outcome-Driven
par(mfrow=c(2,2),cex=0.5)
Delm <- lm(BodyFat~.,
           body_fat_Decorrelated_trainD[,c("BodyFat",
                                           names(modelBodyFatD$coef)[-1])])
pander::pander(Delm,add.significance.stars=TRUE)
```

Table 29: 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	

```r
plot(Delm)
```




```r
par(op)
```

4.4 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 BMI, $Height$, and $Weight$ 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.

4.4.1 Data Conditioning

First Remove Height and Weight from Training and Testing Sets

```r
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")
```

Table 30: Training Distribution

0	1
96	92

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

Table 31: Testing Distribution

0	1
29	34

```r
## 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.

4.4.2 The Logistic Model

LASSO_MIN with a binomial family is used to compute the logistic model of overweight.

```r
## Outcome-blind
modelOverweight <- LASSO_MIN(Overweight~.,
                             OW_Decorrelated_train,
                             family="binomial")
pander::pander(as.matrix(modelOverweight$coef),caption="Training: Blind")
```

Table 32: 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")
```

Table 33: 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

4.4.3 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")
```

Table 34: Observed Coefficients

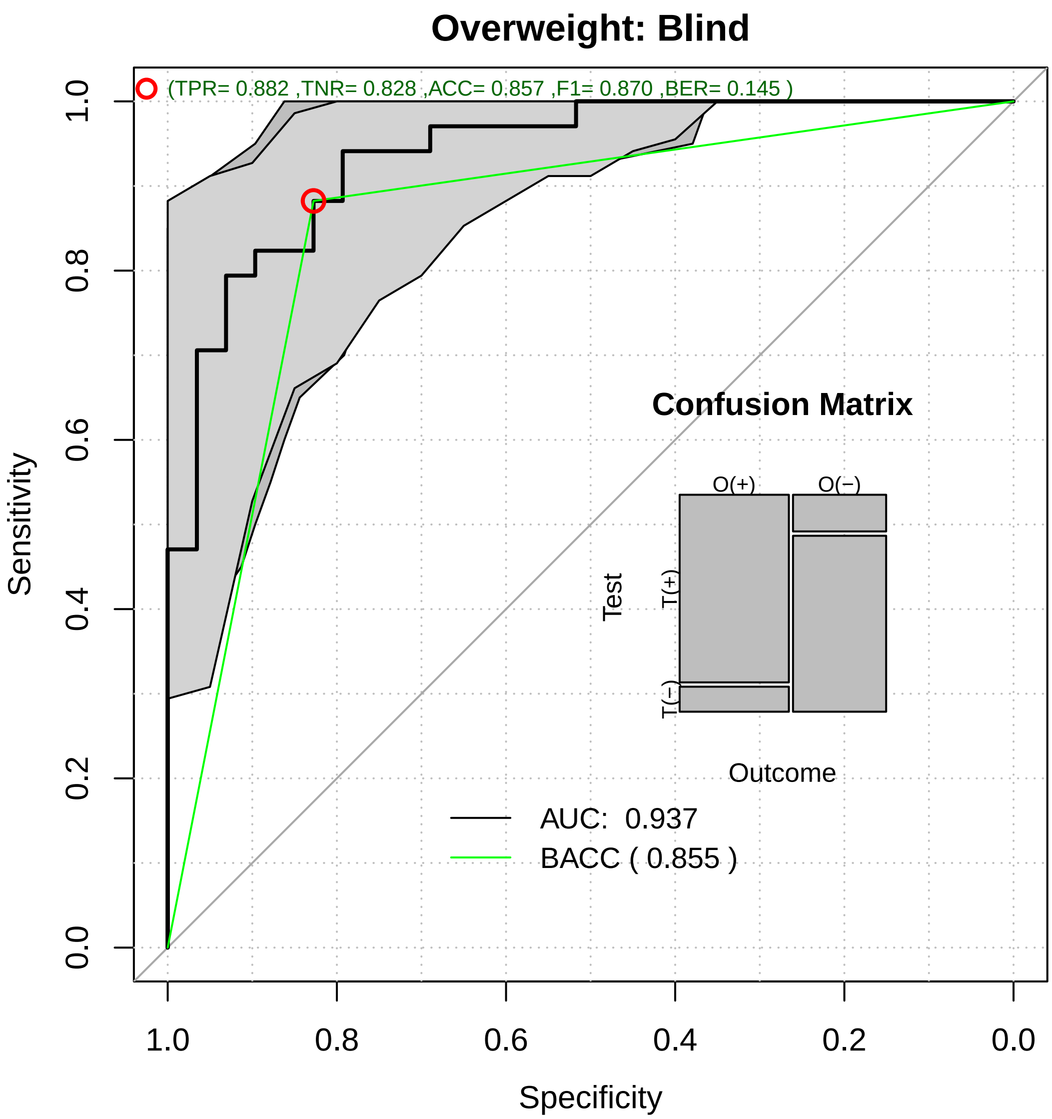
(Intercept)	-60.8766
BodyFat	0.0276
Neck	-0.0539
Chest	0.3680
Abdomen	0.1495
Hip	0.0518
Thigh	0.0405
Knee	-0.0376
Ankle	0.0707
Biceps	0.0363
Wrist	0.1712

4.4.4 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)
```

```r
pr <- predictionStats_binary(cbind(OW_Decorrelated_test$Overweight,
                                   predicOverweight),"Overweight: Blind")
```




```r
pander::pander(pr$ClassMetrics)
```

- **accci**:

50%	2.5%	97.5%
0.857	0.762	0.937

- **senci**:

50%	2.5%	97.5%
0.886	0.763	0.973

- **speci**:

50%	2.5%	97.5%
0.828	0.667	0.96

- **aucci**:

50%	2.5%	97.5%
0.856	0.762	0.936

- **berci**:

50%	2.5%	97.5%
0.144	0.0639	0.238

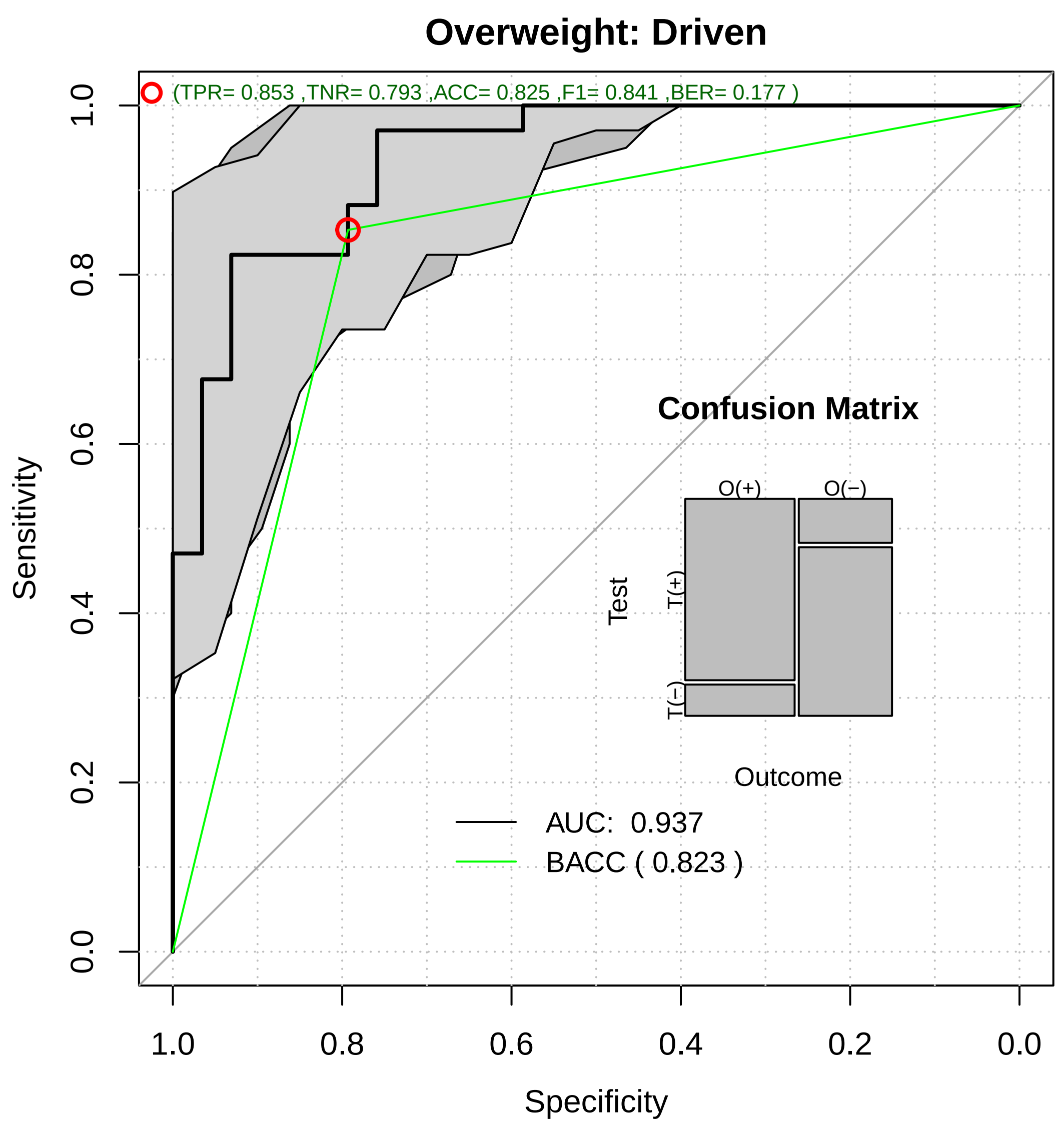
- **preci**:

50%	2.5%	97.5%
0.861	0.73	0.969

- **F1ci**:

50%	2.5%	97.5%
0.871	0.774	0.944

```r
## Outcome-Driven
predicOverweightD <- predict(modelOverweightD,OW_Decorrelated_testD)
pr <- predictionStats_binary(cbind(OW_Decorrelated_test$Overweight,
                                   predicOverweightD),"Overweight: Driven")
```




```r
pander::pander(pr$ClassMetrics)
```

- **accci**:

50%	2.5%	97.5%
0.825	0.73	0.921

- **senci**:

50%	2.5%	97.5%
0.853	0.719	0.969

- **speci**:

50%	2.5%	97.5%
0.8	0.64	0.929

- **aucci**:

50%	2.5%	97.5%
0.824	0.722	0.917

- **berci**:

50%	2.5%	97.5%
0.176	0.0833	0.278

- **preci**:

50%	2.5%	97.5%
0.829	0.694	0.941

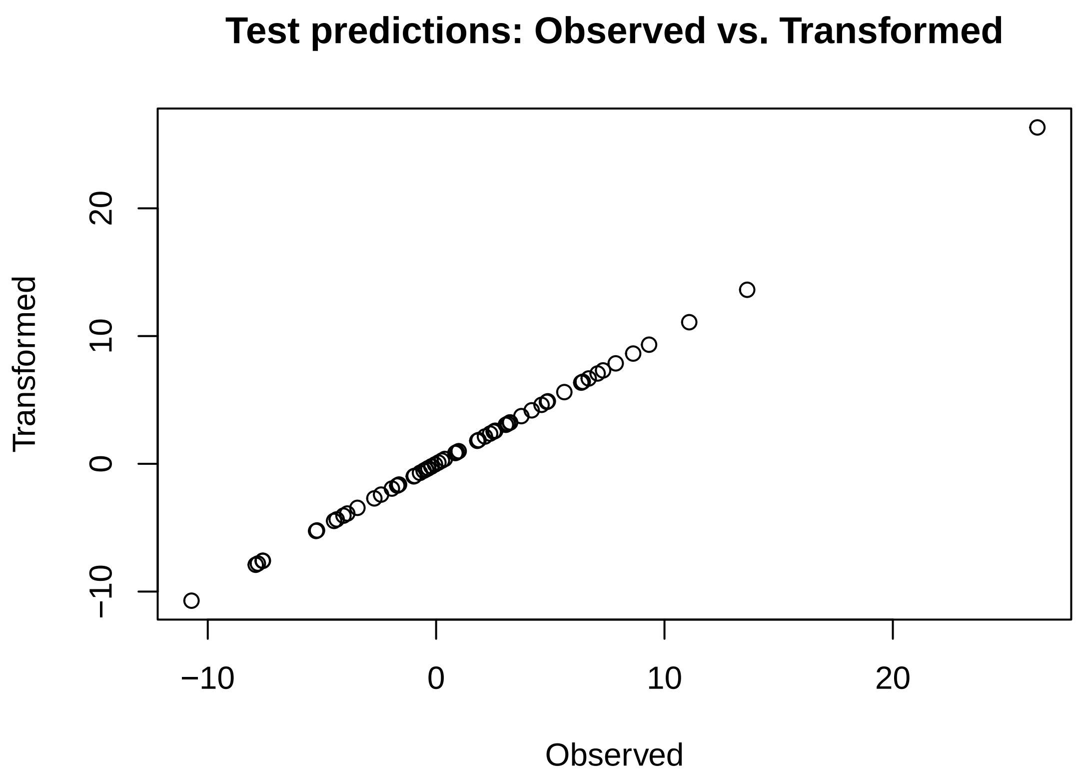
- **F1ci**:

50%	2.5%	97.5%
0.842	0.73	0.923

4.4.5 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.

4.4.6 Comparison to Raw Model

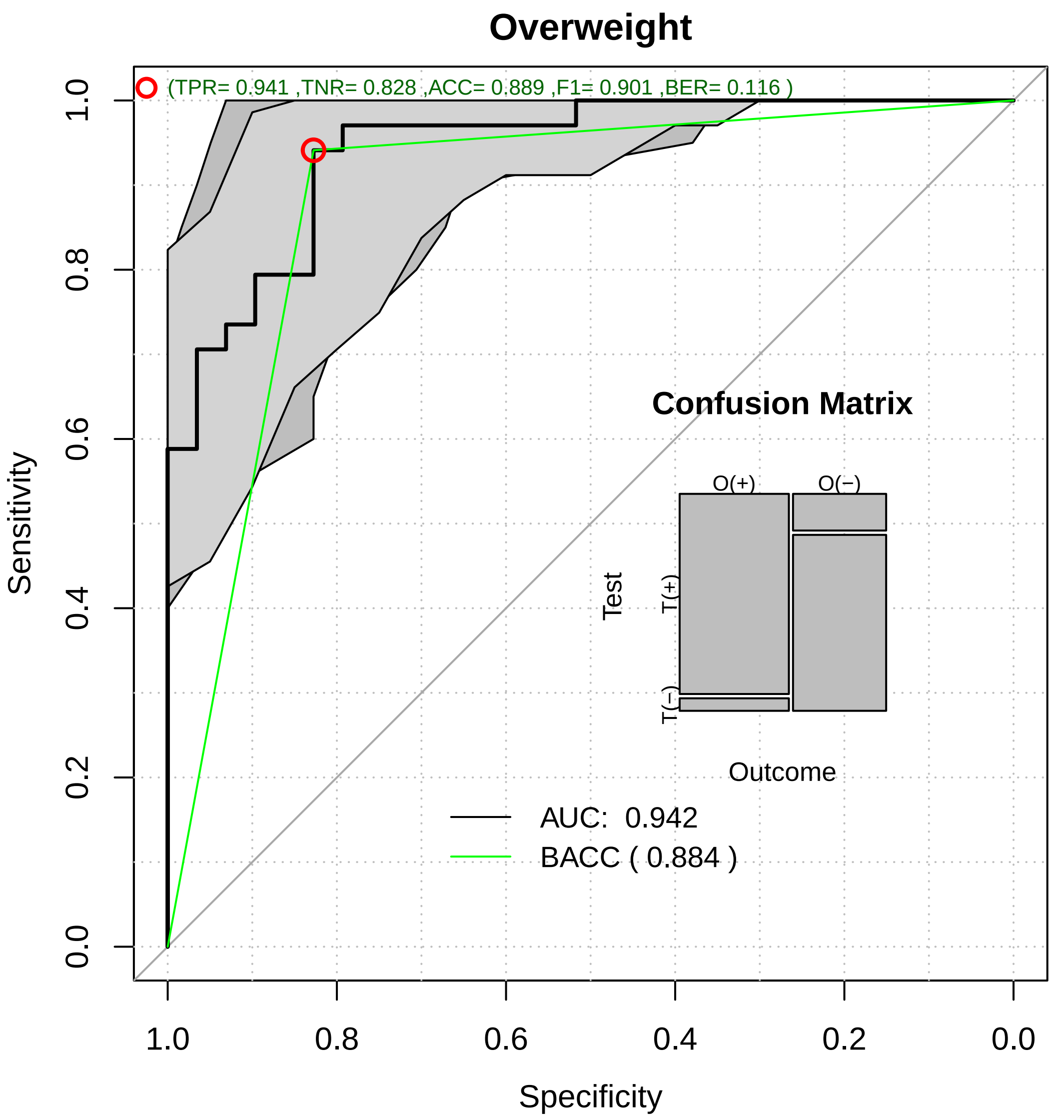
To showcase the advantage of transformed modeling *vs.* raw modeling, here I'll estimate the logistic model from the observed variables and contrast to the model generated from the transformed space.

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	0.968

- **senci**:

50%	2.5%	97.5%
0.943	0.853	1

- **speci**:

50%	2.5%	97.5%
0.833	0.686	0.963

- **aucci**:

50%	2.5%	97.5%
0.889	0.803	0.963

- **berci**:

50%	2.5%	97.5%
0.111	0.0366	0.197

- **preci**:

50%	2.5%	97.5%
0.871	0.75	0.971

- **F1ci**:

50%	2.5%	97.5%
0.904	0.821	0.97

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

4.4.7 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.

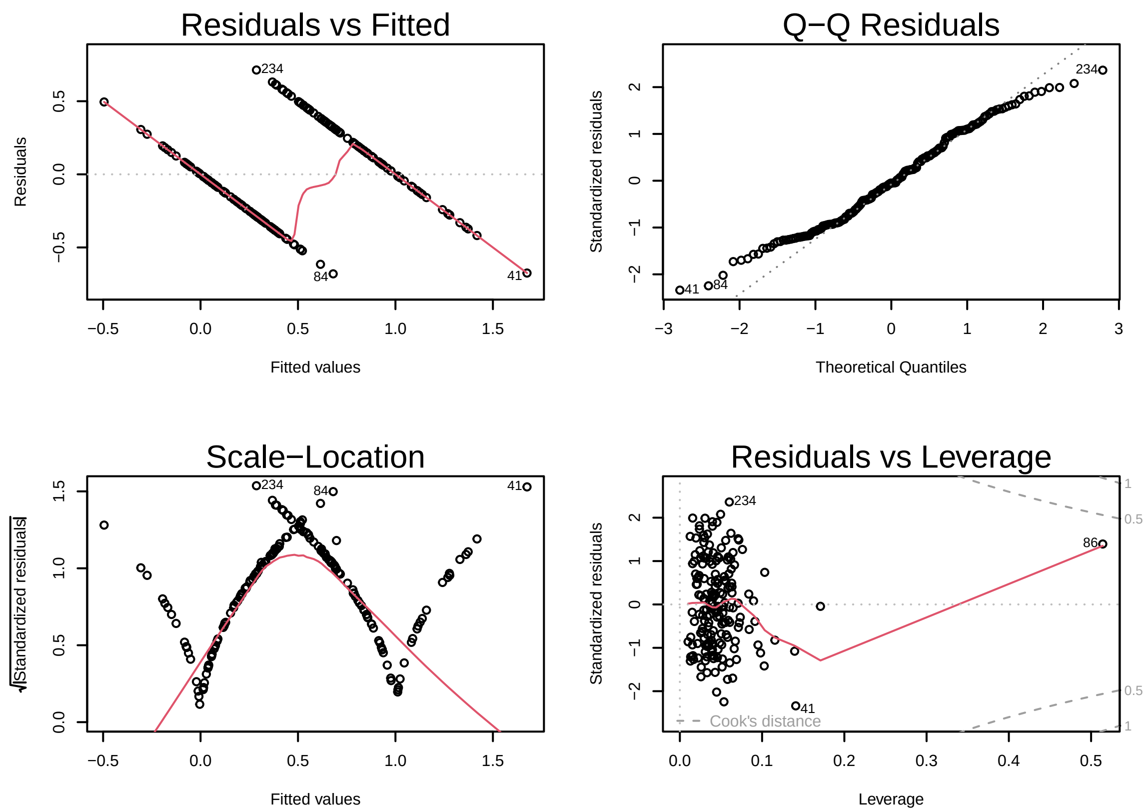
```
par(mfrow=c(2,2),cex=0.5)

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

Table 57: 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	

	Estimate	Std. Error	t value	Pr(>|t|)	
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	

```r
plot(rawlm)
```

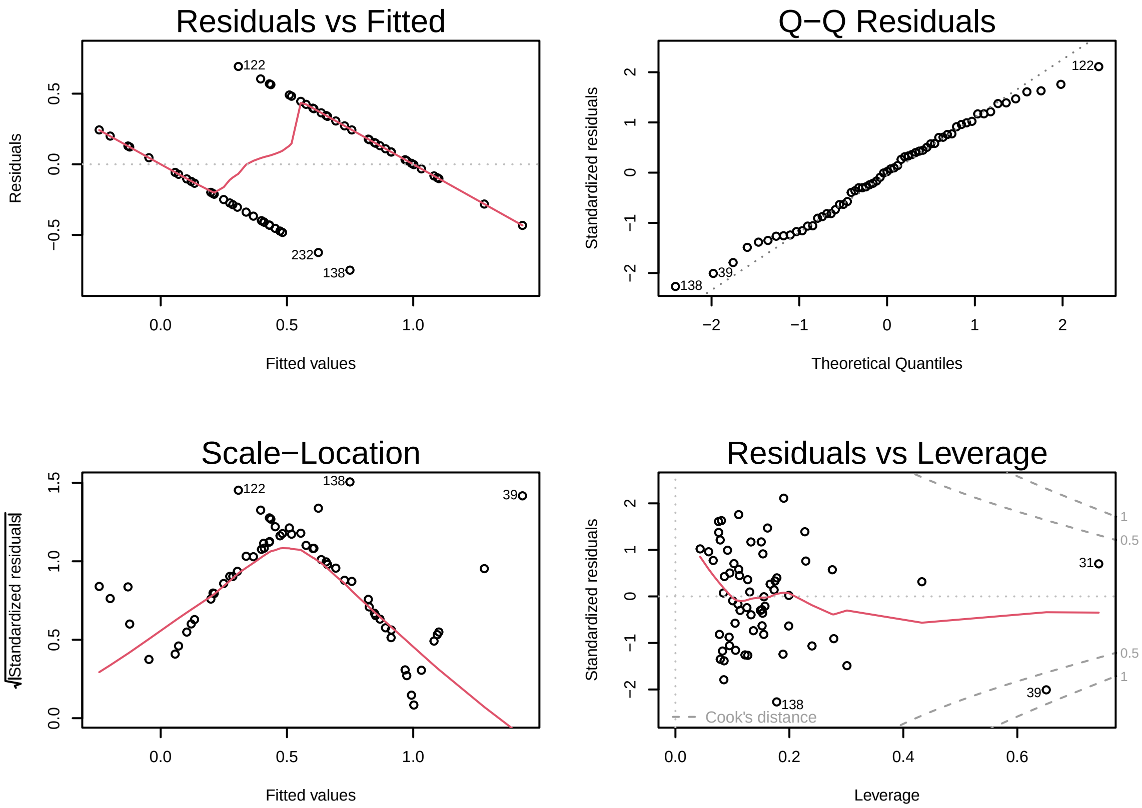



```r
## Outcome-blind
par(mfrow=c(2,2),cex=0.5)
Delm <- lm(Overweight~.,OW_Decorrelated_test[,c("Overweight",names(modelOverweight$coef)[-1])])
pander::pander(Delm,add.significance.stars=TRUE)
```

Table 58: 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	

```r
plot(Delm)
```

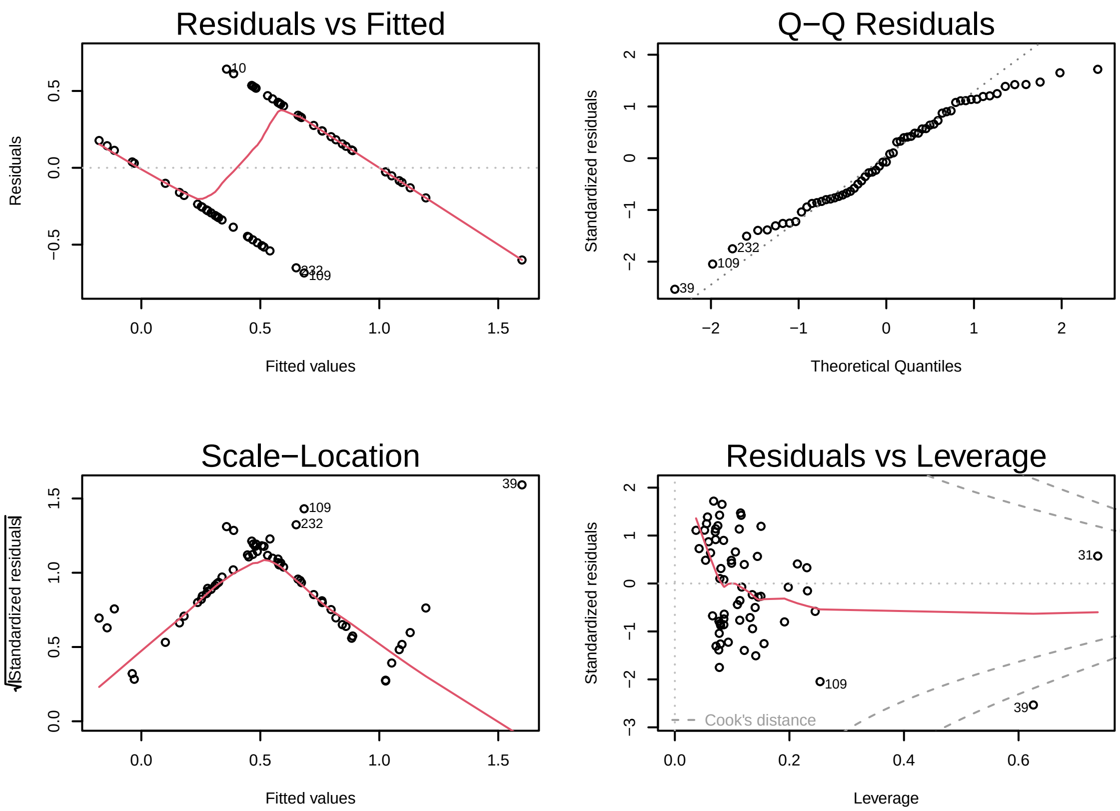



```r
## Outcome-Driven
par(mfrow=c(2,2),cex=0.5)
Delm <- lm(Overweight~.,OW_Decorrelated_testD[,c("Overweight",names(modelOverweightD$coef)[-1])])
pander::pander(Delm,add.significance.stars=TRUE)
```

Table 59: 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	

```r
plot(Delm)
```



5 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.

6 Appendix A

6.1 The ILAA Algorithm

There are many transformation matrices W that moves a the observed feature space X affected by collinearity into new space Q, with controled collinearity i.e.,:

$$Q = WX,$$

and they can be found in many different ways. The ILAA algorithm aims to find a single realization based on variable residualization, and unit preserving transformations. i.e., it aims to find the Exploratory Residualization Transform (ERT).

The algorithm iterates by estimating the residuals of the linear fitting between observed feature pairs (x_i, x_j):

$$x_j = \beta_{i,j} x_i + b_{i,j} + \epsilon_{i,j}.$$

Hence the residual variable:

$$q_j = x_j - \beta_{i,j} x_i, \quad \forall (i,j) \in \mathbb{B},$$

is a latent variable with zero correlation between the "independent" feature i and the "dependent" feature j.

The residualization is done on all the features pairs with significant association. Then, to address multiple collinearity the algorithm must be iterated.

The iteration:

$$W^n = W^{n-1}(I - \boldsymbol{\beta}^n),$$

where $\boldsymbol{\beta}^n$ is the matrix of the beta coefficients of all independent variables to the dependent variables, will estimate the final shape of the ERT transformation until the desired maximum absolute correlation on Q is achieved.

The main issue with iterated realization is the heuristic decision of which feature is "independent" and who is the "dependent" feature. The ILAA function decides the hierarchy based on the degree of correlation and/or association to variables.

If we define $\mathbb{B}$ as the basis vector of the observed matrix X, and we aim to return a Q matrix whose maximum correlation is lower than T, then the following steps estimate a single realization of W:

1. Set $Q^1 = X$, and $W^1 = I$ as the initial solutions.
2. Compute the absolute of the correlation matrix of Q^1: $R^1 = |corr(Q)|$ then set $R^1_{j,j \in \mathbb{B}} = 0$.
3. Extract the vector of the maximum correlation of each feature:

$$\boldsymbol{\rho} = \max(R^1_{k,:}) : \forall k \in \mathbb{B}$$

4. Extract the set of unaltered bases: $\mathbb{U}$:

Set $\mathbb{K} = \mathbb{B}$, then sort $\mathbb{K}$ from the most relevant feature to the least relevant then iterate trough all sorted features:

$$\mathbb{U}^{m+1} = \mathbb{U}^m \cup \{\mathbb{K}_m \mid R^1_{\mathbb{K}_m \times \mathbb{U}^m} < T_0\},$$

where T_0 is the critical value of significant association between variables, and $\mathbb{U}^1 = \emptyset$.

5. At iteration n, define the $\mathbb{I}$ (independent) and the $\mathbb{J}$ (dependent) paired set: $\mathbb{I} \times \mathbb{J}$.
   First, get all the correlation pairs above the correlation goal:

$$\mathbb{H}^n = \{(i,j) \quad | \quad R^n_{i,j} \geq T : \forall (i,j) \in \mathbb{B}\}.$$

   Second, sort the relevance of the pairs.

Third, interactively join all the pairs:

$$\mathbb{I}^m \times \mathbb{J}^m = \mathbb{I}^{m-1} \times \mathbb{J}^{m-1} \cup \{(i,j) \quad | \quad \rho_i \geq \rho_j : \forall (i,j) \in \mathbb{H}^n \wedge i \notin \mathbb{J}^{m-1} \wedge j \in \mathbb{T}\};$$

where

$$\mathbb{T} = \{k \in \mathbb{B} \setminus (\mathbb{U} \cup \mathbb{I}^{m-1})\},$$

and

$$\mathbb{I}^1 = \mathbb{J}^1 = \emptyset.$$

6. Compute the $\boldsymbol{\beta}^n$ matrix loading based on the $\mathbb{I} \times \mathbb{J}$ pairs. Estimate the slope coefficients $\beta_{i,j}$ via linear fitting
$$x_j = \beta_{i,j} x_i + a_j, \quad \forall (i,j) \in \mathbb{I} \times \mathbb{J}$$
Then, set all non-statistically significant $\beta_{i,j}$ values to zero. Statically significant values are defined by $p(\beta_{i,j}) < p_{th}$; where p_{th} is the maximum probability of acceptance, and $p(\beta_{i,j})$ is the probability that the slope coefficient is zero.

7. Update
$$W^n = W^{n-1}(I - \boldsymbol{\beta}^n),$$
then compute $Q^n = W^n X$.

8. Update the correlation matrix: $R^n = |corr(Q^n)|$, then set $R^n_{j,j} = 0 : \ \forall j \in \mathbb{B}$, and the maximum correlation vector: $\boldsymbol{\rho} = \max(R^n_{k,:})$.

9. Repeat steps 5 to 8 until the maximum value of R^n is less than T, i.e., $\max(R^n) < T$.

6.1.1 Implementation Notes:

1. The correlation metric could be Person, or Spearman.

2. The critical value T_0 at step 4 is the critical Pearson's correlation found using the t-distribution approximation. It is found by setting a significant q-value adjusted for multiple comparisons. $q = 0.15/|\mathbb{B}|$;

```r
q= 0.15/ncol(data)
ndf <- nrow(data)-2
tvalue <- qt(1.0 - q,ndf)
T_0 <- tvalue/sqrt(ndf + tvalue^2)
```

3. The FRESA.CAD::ILAA() linear fitting of step 6 is standard linear fitting for Pearson's Correlation, or Robust fitting for Spearman's rank correlation.

4. The p_{th} for statistical significance uses a q-value of 0.15 and it is adjusted for multiple comparisons using the Bonferroni correction: $p_{th} = 0.15/|B|$.

5. The ILAA implementation in FRESA.CAD uses relaxed iterations. i.e., the threshold values changes from an initial value of 0.95 until it reaches the user supplied threshold: T

6. In FRESA.CAD::ILAA the sorting of step 4 can be set to association to a user defined target (Driven) or the user can provide their own ranked variables.

7. In FRESA.CAD::ILAA the sorting of step 5 can be selected from maximum correlation or weighted correlation.

8. The correlation matrix R depends on the data. FRESA.CAD::ILAA() can be set to aggregate estimations of W with bootstrapped perturbations of the data.