# Assignment 6 – Factor Analysis

Project Name: Prediction of sales prices of houses
GitHub link: https://github.com/tnutalapati/prediction-of-sales-prices-of-houses

Team:
Tejaswini Nutalapati
Aditi Bhargava

Loading the libraries

```
library(ggplot2)
library(caret)
library(scales)
library(dummies)
library(fmsb)
library(randomForest)
library(corrplot)
library(ggrepel)
library(rlang)
library(gridExtra)
library(knitr)
library(Amelia)
library(mice)
library(dplyr)
library(fastDummies)
library(lattice)
library(pcaPP)
```

```
head(train,2)
```

```
head(train,2)
A tibble: 2 x 82
   Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour Utilities LotConfig LandSlope Neighborhood
<dbl>      <dbl> <chr>    <chr>         <dbl> <chr>  <fct> <chr>    <chr>       <chr>     <chr>     <chr>     <chr>
    1         60 RL       65             8450 Pave   NA    Reg      Lvl         AllPub    Inside    Gtl       CollgCr
    2         20 RL       80             9600 Pave   NA    Reg      Lvl         AllPub    FR2       Gtl       Veenker
… with 69 more variables: Condition1 <chr>, Condition2 <chr>, BldgType <chr>, HouseStyle <chr>, OverallQual <dbl>,
  OverallCond <dbl>, YearBuilt <dbl>, YearRemodAdd <dbl>, RoofStyle <chr>, RoofMatl <chr>, Exterior1st <chr>, Exterior2nd <chr>,
  MasVnrType <fct>, MasVnrArea <chr>, ExterQual <chr>, ExterCond <chr>, Foundation <chr>, BsmtQual <fct>, BsmtCond <fct>,
  BsmtExposure <fct>, BsmtFinType1 <fct>, BsmtFinSF1 <chr>, BsmtFinType2 <fct>, BsmtFinSF2 <chr>, BsmtUnfSF <chr>,
  TotalBsmtSF <chr>, Heating <chr>, HeatingQC <chr>, CentralAir <chr>, Electrical <chr>, `1stFlrSF` <dbl>, `2ndFlrSF` <dbl>,
  LowQualFinSF <dbl>, GrLivArea <dbl>, BsmtFullBath <chr>, BsmtHalfBath <chr>, FullBath <dbl>, HalfBath <dbl>, BedroomAbvGr <dbl>,
  KitchenAbvGr <dbl>, KitchenQual <chr>, TotRmsAbvGrd <dbl>, Functional <chr>, Fireplaces <dbl>, FireplaceQu <fct>,
  GarageType <fct>, GarageYrBlt <chr>, GarageFinish <fct>, GarageCars <dbl>, GarageArea <dbl>, GarageQual <fct>, GarageCond <fct>,
  PavedDrive <chr>, WoodDeckSF <dbl>, OpenPorchSF <dbl>, EnclosedPorch <dbl>, `3SsnPorch` <dbl>, ScreenPorch <dbl>,
  PoolArea <dbl>, PoolQC <fct>, Fence <fct>, MiscFeature <fct>, MiscVal <dbl>, MoSold <dbl>, YrSold <dbl>, SaleType <chr>,
  SaleCondition <chr>, SalePrice <dbl>, Remodel_flag <fct>
```

```
head(test,2)
```

```
head(test,2)
A tibble: 2 x 82
   Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour Utilities LotConfig LandSlope Neighborhood
<dbl>      <dbl> <chr>    <chr>         <dbl> <chr>  <fct> <chr>    <chr>       <chr>     <chr>     <chr>     <chr>
 1461         20 RH       80            11622 Pave   NA    Reg      Lvl         AllPub    Inside    Gtl       NAmes
 1462         20 RL       81            14267 Pave   NA    IR1      Lvl         AllPub    Corner    Gtl       NAmes
… with 69 more variables: Condition1 <chr>, Condition2 <chr>, BldgType <chr>, HouseStyle <chr>, OverallQual <dbl>,
  OverallCond <dbl>, YearBuilt <dbl>, YearRemodAdd <dbl>, RoofStyle <chr>, RoofMatl <chr>, Exterior1st <chr>, Exterior2nd <chr>,
  MasVnrType <fct>, MasVnrArea <chr>, ExterQual <chr>, ExterCond <chr>, Foundation <chr>, BsmtQual <fct>, BsmtCond <fct>,
  BsmtExposure <fct>, BsmtFinType1 <fct>, BsmtFinSF1 <chr>, BsmtFinType2 <fct>, BsmtFinSF2 <chr>, BsmtUnfSF <chr>,
  TotalBsmtSF <chr>, Heating <chr>, HeatingQC <chr>, CentralAir <chr>, Electrical <chr>, `1stFlrSF` <dbl>, `2ndFlrSF` <dbl>,
  LowQualFinSF <dbl>, GrLivArea <dbl>, BsmtFullBath <chr>, BsmtHalfBath <chr>, FullBath <dbl>, HalfBath <dbl>, BedroomAbvGr <dbl>,
  KitchenAbvGr <dbl>, KitchenQual <chr>, TotRmsAbvGrd <dbl>, Functional <chr>, Fireplaces <dbl>, FireplaceQu <fct>,
  GarageType <fct>, GarageYrBlt <chr>, GarageFinish <fct>, GarageCars <dbl>, GarageArea <dbl>, GarageQual <fct>, GarageCond <fct>,
  PavedDrive <chr>, WoodDeckSF <dbl>, OpenPorchSF <dbl>, EnclosedPorch <dbl>, `3SsnPorch` <dbl>, ScreenPorch <dbl>,
  PoolArea <dbl>, PoolQC <fct>, Fence <fct>, MiscFeature <fct>, MiscVal <dbl>, MoSold <dbl>, YrSold <dbl>, SaleType <chr>,
  SaleCondition <chr>, SalePrice <dbl>, Remodel_flag <fct>
```

```
SalePrice <- train$SalePrice
train2<-train[-81]
data <- rbind(train2, test)
id<-data[1]
```

```
sum(is.na(data))
```

There is a total of 13965missing values. Now, checking which variables have missing values

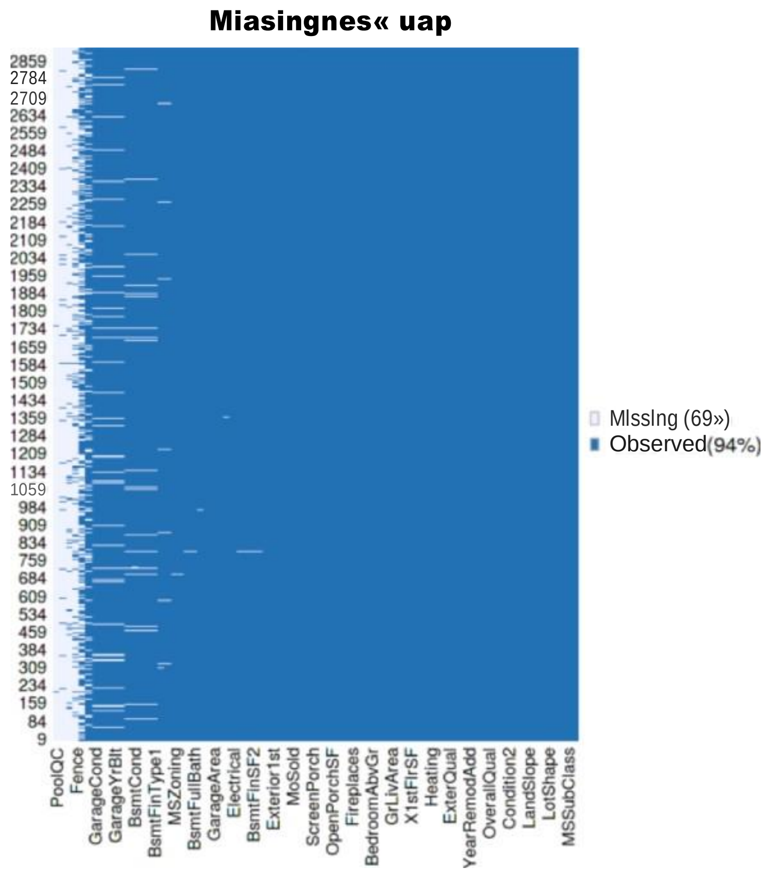
```
i <- 1
na <- 1
for (i in 1:length(data))
{
  na[i] <- ifelse(sum(is.na(data[i]))>0, sum(is.na(data[i])), 0)
  if (na[i]>0)
  {
    cat(names(data[i]), '=', na[i],' ')
  }
}
```

```
> MSZoning = 4
  LotFrontage = 486
  Alley = 2721
  Utilities = 2
  Exterior1st = 1
  Exterior2nd = 1
  MasVnrType = 24
  MasVnrArea = 23
  BsmtQual = 81
  BsmtCond = 82
  BsmtExposure = 82
  BsmtFinType1 = 79
  BsmtFinSF1 = 1
  BsmtFinType2 = 80
  BsmtFinSF2 = 1
  BsmtUnfSF = 1
  TotalBsmtSF = 1
  Electrical = 1
  BsmtFullBath = 2
  BsmtHalfBath = 2
  KitchenQual = 1
  Functional = 2
  FireplaceQu = 1420
  GarageType = 157
  GarageYrBlt = 159
  GarageFinish = 159
  GarageCars = 1
  GarageArea = 1
  GarageQual = 159
  GarageCond = 159
  PoolQC = 2909
  Fence = 2348
  MiscFeature = 2814
  SaleType = 1
```

Variables with about or more than 50% missing data in train and test set is PoolQC, MiscFeature, Alley, Fence and FireplaceQu

Mapping the missingness Map:

```
missmap(data)
```




```
z‹ - c(WI ch(coin ees{dotog==" PoolgC" }, sht ch(co1nomes{doteg=="Ht scFeoture" }
        , nhichCco1nomesCdoto}=="41Iey"    , wht Ch(ColnorreS{ dotoJ==" Fence"
        , whtch(cotnomes£doto›--"Ft reptocegu" })


//Removt ng vor I obl es wtth mostly nt ssh ng values
doto< -data , c( - 73, - 75, -7, - 74, -38)a


#Predt ct tve meon matchi ng tregut ott on for numert c , btnory and ordered vort obt es
mt ce_zx>d < - mt ce(doto [ , c( "8smtg uol " , "BsM£ond" , "BsWExposure", "BsmtFt nType1" , " Bsmt FtnType2"
                                         , "E \ ectrt cot " , "GorogeType" , "GorogeFtn\ sh" , "Go regegual " , "GoregeCond"
                                         , "MosVnrType " , "MSZoni ng™ , "Uli T I tt es™ , " E xterl or 1st " , " E xter I or Znd"
                                         , "K tchenquol " , "Funct Sonal " , "SoleType " , "Lotr montage " , "forage YrBt I"
                                         , "NosVnrAreo" , "BsntF tnSF1" , "BsaitF I nSF2" , "BsmJ-Un£ SF" , "Totat BsmrSF"
                                         , "Bsmt Fu\.1Both", "BsnttHol fBoth" , "€orogeCars" , "€orogeAreo“g§ , meLhod- ' pms }
mi ce_conpt ete • - couple tetni ce_niod)
```

## Data set preparation

```
#Two sets - numeric and factor
data_factor <- select_if(data, is.factor)
data_numeric<- select_if(data, is.numeric)
data_numeric <- data_numeric[-1]
```

```
#Feature scaling numeric variables
#if PCA is not used there is no need to scale variables in regression tree model)
data_numeric_scale<-data.frame(scale(data_numeric))
```

```
#Creating dummy variables (if PCA is not used there is no need to make dummy variables)
data_factor_dummy <- dummy_cols(data_factor, remove_first_dummy = TRUE)
data_factor_dummy <- select_if(data_factor_dummy, is.numeric)
```

```
data2 <- cbind(id, data_numeric_scale, data_factor_dummy)
```

```
#removing highly correlated variables
datacor<-cor(data2)
f<-findCorrelation(datacor, cutoff = 0.9, verbose = T, names=F)

data2 <- data2[ ,-f]
```

```
Combination row 106 and column 108 is above the cut-off, value = -0.939
Flagging column 106
Combination row 122 and column 136 is above the cut-off, value = 0.983
Flagging column 122
Combination row 125 and column 139 is above the cut-off, value = 0.97
Flagging column 125
Combination row 129 and column 144 is above the cut-off, value = 0.978
Flagging column 129
Combination row 226 and column 233 is above the cut-off, value = 0.987
Flagging column 226
```

```
#train
train_s <- data2[1:1460,]
train_set<- cbind(train_s, SalePrice)
train_set <- train_set[-1]
#test
test_set <- data2[1461:2919,]
test_set<-test_set[-1]
```

Removing highly correlated variables

```
datacorforest<-cor(data_numeric)
findCorrelation(datacorforest, cutoff = 0.9, verbose = T, names=F)
```

```
> All correlations <= 0.9
```

```
train_sett<-data[1:1460,]
train_settt<- cbind(train_sett, SalePrice)
train_settt <- train_settt[-1]
#test no pca
test_settt <- data[1461:2919,]
test_settt<-test_settt[-1]
```

## PRINCIPAL COMPONENT ANALYSIS – PCA

Remove dependent variable from train set

```
dim(train_set)
```

1460. 228

Remove dependent variable from train set

```
train_set<-train_set[-228]
```

PCA works only on numeric variables. All values are numerical.
By default, it centers the variable to have mean equals to zero.
With parameter scale. = T, normalize the variables to have standard deviation equals to 1.

```
pca <- prcomp(train_set, scale. = T)
```

center and scale refer to mean and standard deviation of the variables

```
names(pca)
```

'sdev' 'rotation' 'center' 'scale' 'x'

Each column of rotation matrix contains the principal component loading vector. Principal components and first 3 rows

```
pca$rotation[1:3,1:4]
```

| | PC1 | PC2 | PC3 | PC4 |
|---|---|---|---|---|
| MSSubClass | -0.00133405 | 0.1272953 | -0.0404936 | 0.14172673 |
| LotFrontage | 0.06587685 | -0.1082165 | 0.1915191 | -0.03903983 |
| LotArea | 0.03631721 | -0.1118286 | 0.1874054 | -0.05478405 |

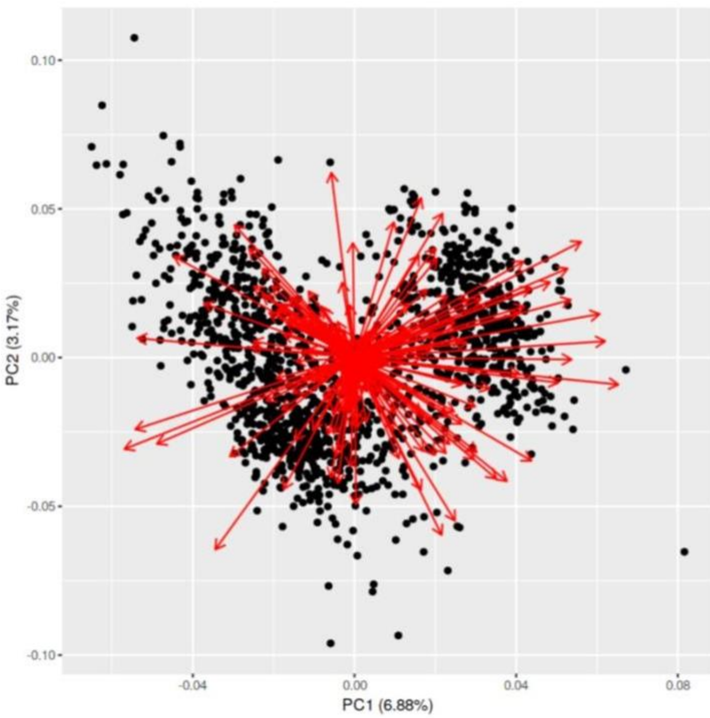
The matrix x has the principal component score vectors in a 1460 × 227 dimension.

```
dim(pca$x)
```

1460 227

**PC1 vs PC2 with loadings**

```
autoplot(pca, loadings = TRUE)
```



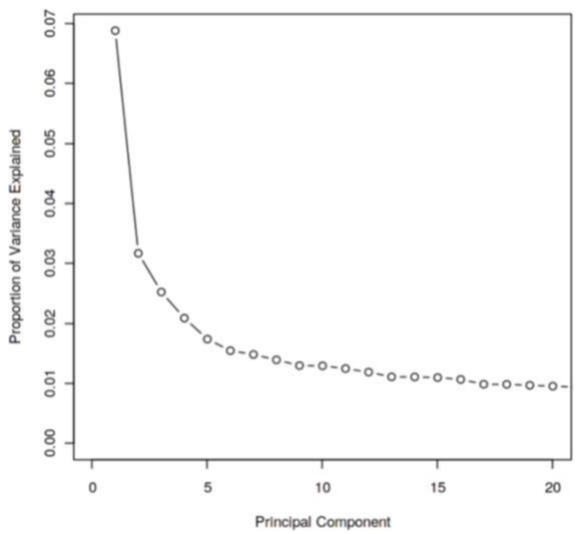

```
#standard deviation of each principal component
std_dev <- pca$sdev
#variance
variance <- std_dev^2
```

**divide the variance by sum of total variance -> to compute the proportion of variance explained by each component**

```
variance_prop <- variance/sum(variance)
#first principal component explains 6.98% of the variance, second 3.2%, third 2.5%
variance_prop[1:10]
```

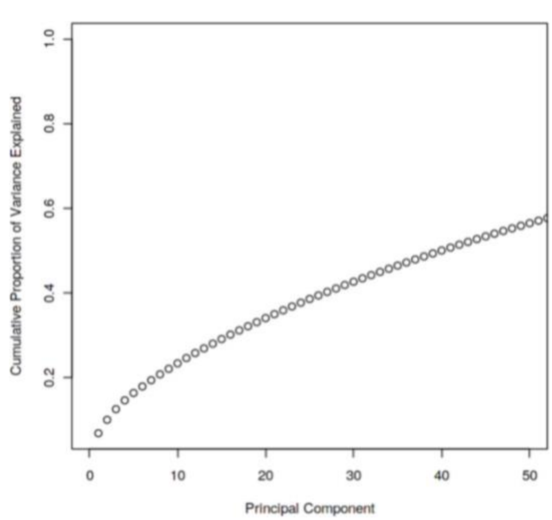
0.0688016033899863  0.0316744090851641  0.0252170776030978  0.0208644460252106
0.0173608341666115  0.0154386797614029  0.0147949460535524  0.0139073769471355
0.0129561906693511  0.0129149176043199

```
#scree plot - the percentage of variance explained by each principal component
plot(variance_prop, xlab = "Principal Component", ylab = "Proportion of Variance Explained"
     , type = "b", xlim=c(0, 20))
```




**Cumulative variance plot**

```
# ~ 50 components explains around 60% variance in the data set.
plot(cumsum(variance_prop), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     type = "b", xlim=c(0, 50))
```

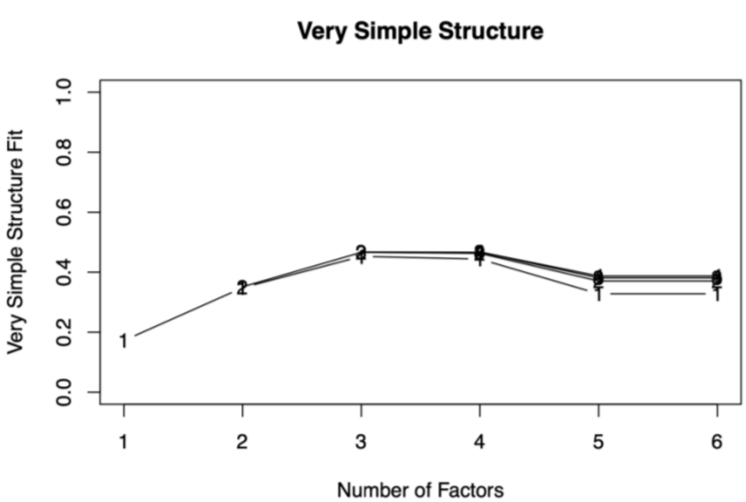


```
#add a column
test_set$SalePrice <- 1
#new training set with principal components
train_set_pca <- data.frame(SalePrice = train$SalePrice, pca$x)
# first 50 PCAs
train_set_pca <- train_set_pca[,1:51]
```

| SalaPrice | PC 1 | PC2 | PC 3 | PC4 | PC 3 | PCB | PC 7 | PCB |
|---|---|---|---|---|---|---|---|---|
| •int• | < dbl• | •dbl> | • 0D\|> | •d bl > | • dbl• | •dbl> | < dbl• | •dbl> |
| 70 8 500 | 4.<2877471 | 1.6140:i7 | •1.3619S3 | 1.5 3<4 467 | •0.68D6910 | •0.0S508328 | I .2 17208 | 1.2 78a 7 655 |
| 2 18 1500 | 0.4 9S675d 2 | -0.O67224 | - 0.2$ 3 76 B | - 0.38 2 3131 | - 0.7698 8 64 | -1.22 037 33 7 | -0.IB 331d | -0.O6163059 |

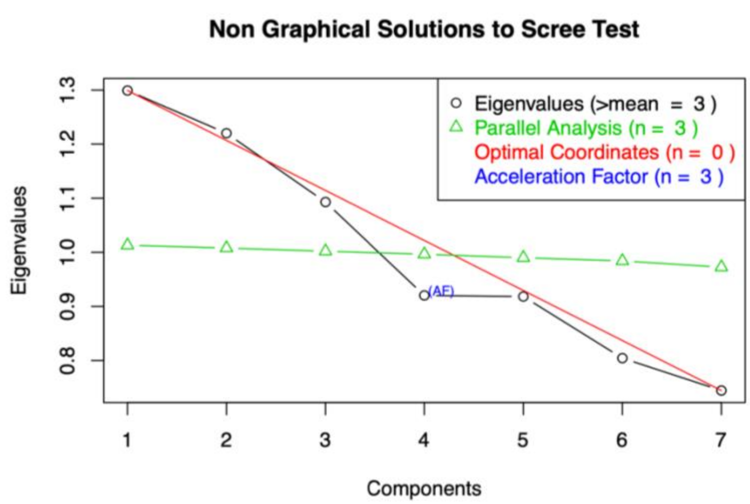
## Factor analysis

```
(eigen_data <- round(euroemp_pca$data^2,2))
names(eigen_data) <- paste("PC",1:9,sep="")
eigen_data

sumlambdas <- sum(eigen_data)
sumlambdas

cumvar_data <- cumsum(propvar)
propvar <- round(eigen_data/sumlambdas,2)
propvar

cumvar_data <- cumsum(propvar)
cumvar_data

matlambdas <- rbind(eigen_data,propvar,cumvar_data)
matlambdas

rownames(matlambdas) <- c("Eigenvalues","Prop. variance","Cum. prop. variance")
rownames(matlambdas)

eigvec.emp <- data_pca$rotation
print(data_pca)
```



```
# Taking the first four PCs to generate linear combinations for all the variables with four factors
pcafactors.emp <- eigvec.emp[,1:4]
pcafactors.emp


# Multiplying each column of the eigenvector's matrix by the square-root of the corresponding eigenvalues
unrot.fact.emp <- sweep(pcafactors.emp,MARGIN=2,euroemp_pca$sdev[1:4],`*`)
unrot.fact.emp


# Computing communalities
communalities.emp <- rowSums(unrot.fact.emp^2)
communalities.emp


# Performing the varimax rotation.
#The default in the varimax function is norm=TRUE thus, Kaiser normalization is carried out
rot.fact.emp <- varimax(unrot.fact.emp)
View(unrot.fact.emp)
rot.fact.emp

# The print method of varimax omits loadings less than abs(0.1).
#In order to display all the loadings, it is necessary to ask explicitly the contents of the object $loadings
fact.load.emp <- rot.fact.emp$loadings[1:9,1:4]
fact.load.emp
```

```
# Computing the rotated factor scores for the 30 European Countries.
#Notice that signs are reversed for factors F2 (PC2), F3 (PC3) and F4 (PC4)
scale.data <- scale(data[-1])
scale.data
as.matrix(scale.emp)%*%fact.load.emp%*%solve(t(fact.load.emp)%*%fact.load.emp)
```




```
fit.pc <- principal(data[-1], nfactors=4, rotate="varimax")
fit.pc
round(fit.pc$values, 3)
fit.pc$loadings

# Loadings with more digits
for (i in c(1,3,2,4)) { print(fit.pc$loadings[[1,i]])}

# Communalities
fit.pc$communality

# Rotated factor scores, Notice the columns ordering: RC1, RC3, RC2 and RC4
fit.pc$scores

# Play with FA utilities
fa.parallel(data[-1])
fa.plot(fit.pc)
fa.diagram(fit.pc)
vss(data[-1])
```