Fourth Question: Working With Real Data

Arnau Abella

03/04/2020

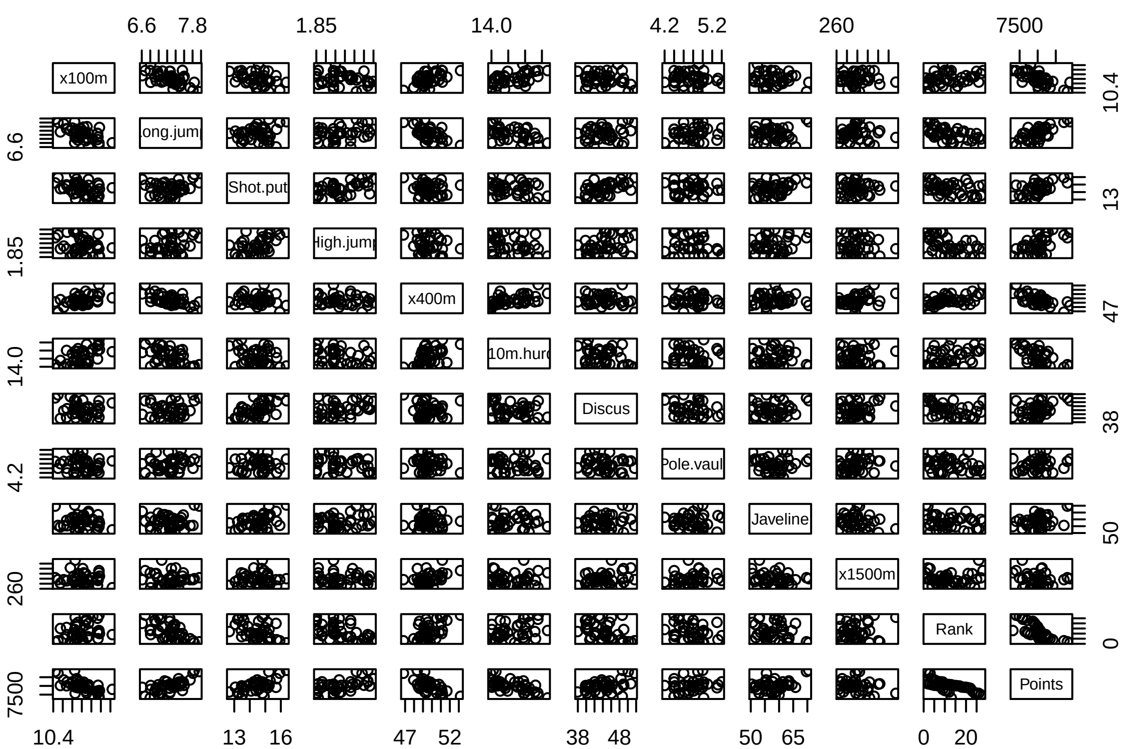
Decathlon Dataset

It is easy to see from the Variables factor map of the PCA that the points is inversly proportional to the rank i.e. the more points you get the lower rank you achieve (lower rank is better)._

From the chart we also see that either the variable has an effect to the rank or to the points, which, at the end of the day, is equivalent. Notice that some results do not have the same impact on the puntuation/rank such as 1500m.

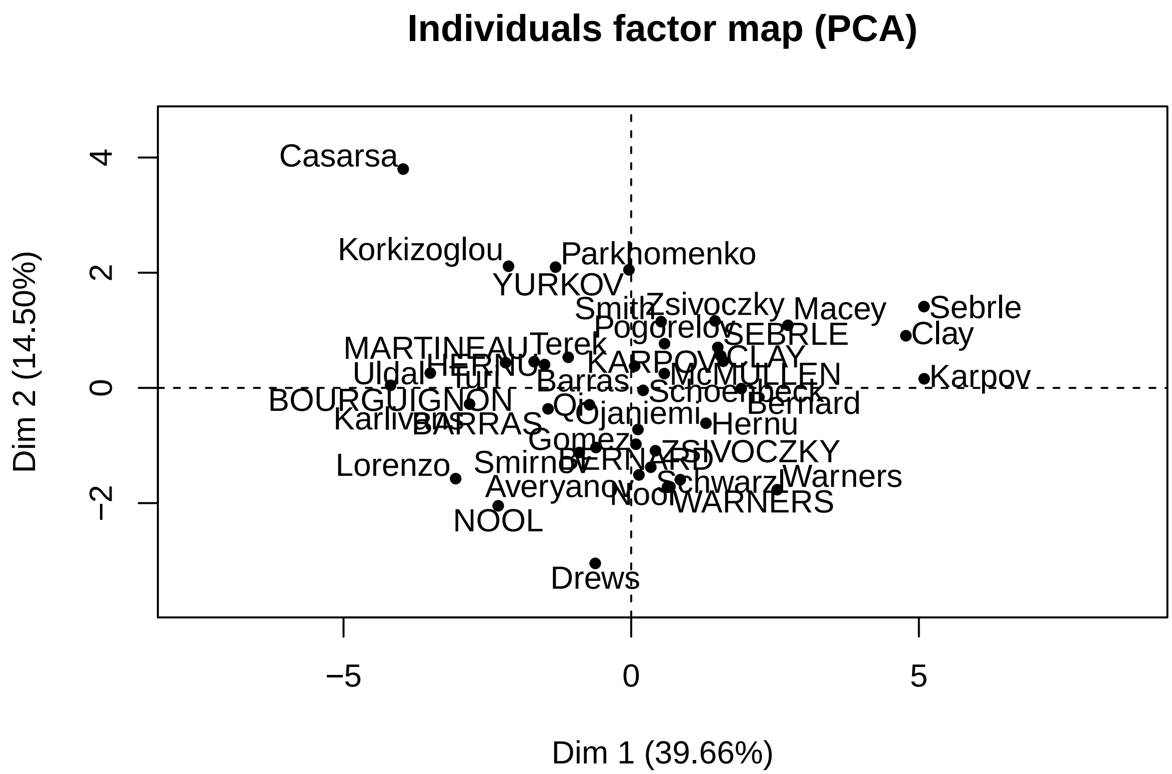
```
# Load the dataset and preprocess it.
data(decathlon)
colnames(decathlon)[c(1,5,6,10)]<-c("x100m","x400m","x110m.hurdle","x1500m")
colnames(decathlon)
```

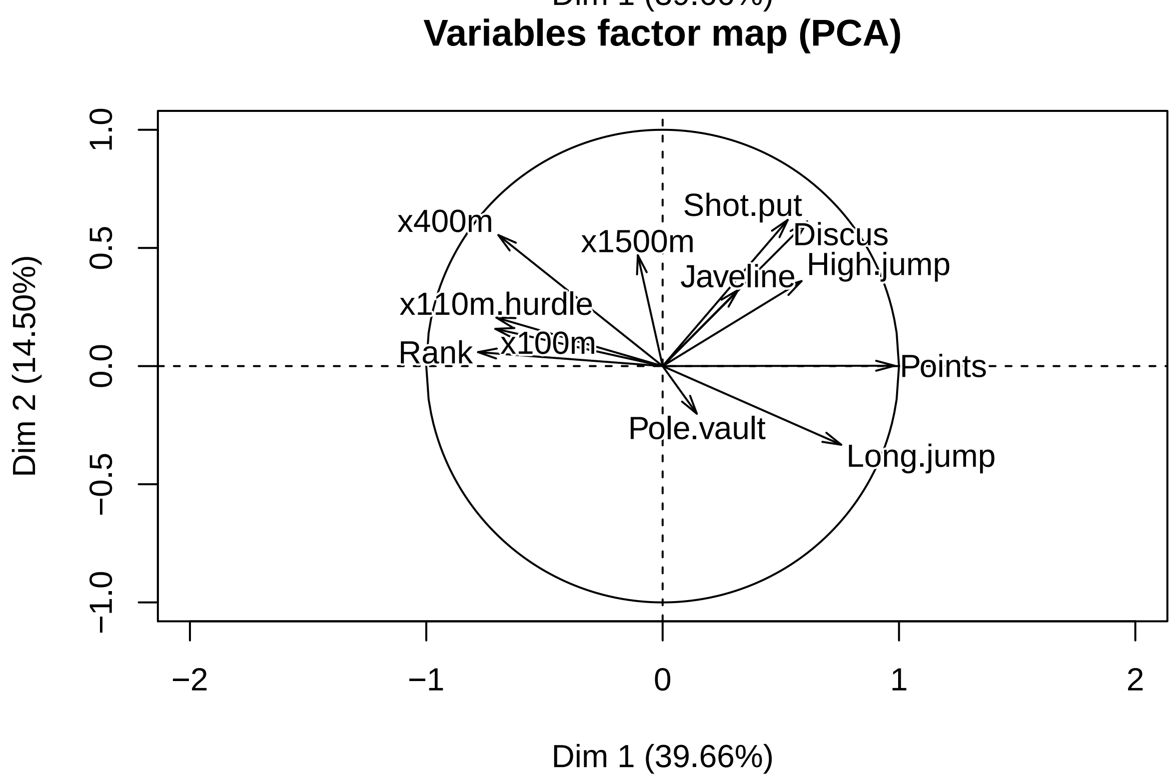
```
competition <- which(colnames(decathlon) == "Competition")
plot(decathlon[,-c(competition)])
```




```
# cor(decathlon[,-c(competition)])
```

```r
# Remove the dependent variable score.
pca<-PCA(decathlon[,-c(competition)])
```



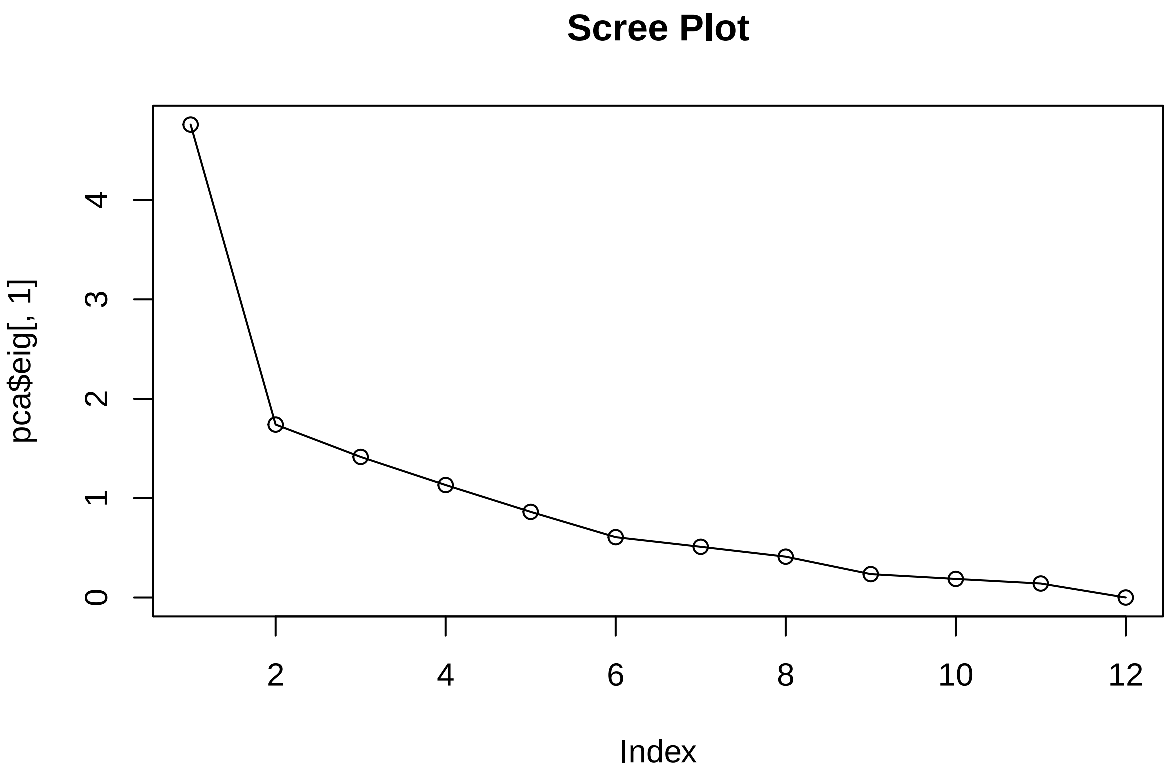


Principal Component Regression

From the cummulative percentage of variance, we need at least 4 principal components to have an accumulative variance $\geq \frac{2}{3}$.

```
library(FactoMineR)
competition <- which(colnames(decathlon) == "Competition")

pca$eig
plot(pca$eig[,1], type="o", main="Scree Plot")
```




```
summary(pca)

decathlon$PC1<-pca$ind$coord[,1]
decathlon$PC2<-pca$ind$coord[,2]
decathlon$PC3<-pca$ind$coord[,3]
decathlon$PC4<-pca$ind$coord[,4]
reg_pc<-lm(Points~PC1 + PC2 + PC3 + PC4, data=decathlon)
```

```
summary(reg_pc)
```

```
##
## Call:
## lm(formula = Points ~ PC1 + PC2 + PC3 + PC4, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -77.522 -35.990   5.294  33.767  89.454
##
## Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept) 8005.3659     6.9336 1154.574  < 2e-16 ***
## PC1          152.1889     3.1784   47.882  < 2e-16 ***
## PC2            0.3998     5.2561    0.076  0.93979
## PC3          -17.8926     5.8290   -3.070  0.00406 **
## PC4           41.6555     6.5175    6.391 2.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.4 on 36 degrees of freedom
## Multiple R-squared:  0.9849, Adjusted R-squared:  0.9832
## F-statistic: 585.7 on 4 and 36 DF,  p-value: < 2.2e-16
```

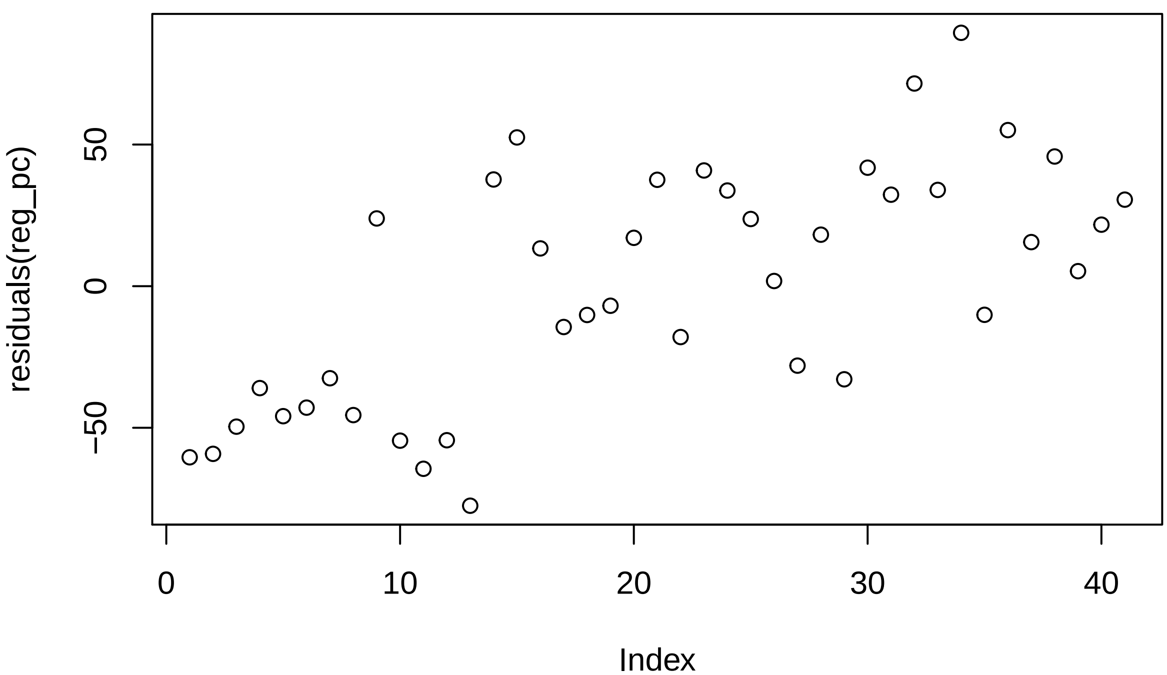
Testing the assumptions of the regression model

- Normality: the error term does follow a normal distribution which is desired.
- Homogenity: the variance is homogeneous.
- Independence of errors: the errors do have correlation which may affect the results.

```r
# Normality
shapiro.test(residuals(reg_pc))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(reg_pc)
## W = 0.96366, p-value = 0.2108
```

```r
# Homogenity
plot(residuals(reg_pc))
```




```r
bptest(reg_pc)
```

```
##
##  studentized Breusch-Pagan test
##
```

```
## data:  reg_pc
## BP = 13.223, df = 4, p-value = 0.01024
```

```r
# Independence of errors
dwtest(reg_pc, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  reg_pc
## DW = 1.0195, p-value = 0.0004396
## alternative hypothesis: true autocorrelation is not 0
```

Predicting the points of an athelete using the regression model

Nota bene, the RMSE is small compared to the points scale so we can conclude that the model is accurate "enough".

```r
n <- nrow(decathlon)
train.sample <- sample(1:n, round(0.67*n))

train.set <- decathlon[train.sample, ]
test.set <- decathlon[-train.sample, ]

train.model <- lm(Points ~ PC1+PC2+PC3+PC4 , data = train.set)
summary(train.model)
```

```
##
## Call:
## lm(formula = Points ~ PC1 + PC2 + PC3 + PC4, data = train.set)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -60.83 -33.70 -13.57  34.43  65.69
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7998.1407     8.5430 936.223  < 2e-16 ***
## PC1          152.0412     4.4297  34.323  < 2e-16 ***
## PC2            0.5286     8.0539   0.066  0.94826
## PC3          -25.6946     7.0097  -3.666  0.00136 **
## PC4           42.7321     7.4789   5.714 9.54e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.97 on 22 degrees of freedom
## Multiple R-squared:  0.9826, Adjusted R-squared:  0.9795
## F-statistic:   311 on 4 and 22 DF,  p-value: < 2.2e-16
```

```r
yhat<-predict(train.model, test.set, interval="prediction")
yhat
```

```
##                fit      lwr      upr
## MARTINEAU 7776.188 7680.956 7871.420
## NOOL      7708.842 7604.981 7812.704
## Sebrle    8851.531 8741.777 8961.286
```

```
## Karpov    8702.490 8594.922 8810.058
## Warners   8338.320 8236.283 8440.358
## Zsivoczky 8298.769 8199.168 8398.369
## Schwarzl  8047.820 7950.861 8144.778
## Smith     8055.547 7954.041 8157.053
## Ojaniemi  8035.316 7941.154 8129.478
## Qi        7902.937 7807.772 7998.101
## Drews     7838.609 7729.806 7947.412
## Terek     7779.868 7682.123 7877.613
## Lorenzo   7582.013 7478.281 7685.745
## Casarsa   7366.519 7248.579 7484.459
```

```r
y<-test.set$score

error<-cbind(yhat[,1,drop=FALSE],y,(yhat[,1]-y)^2)
sqr_err<-error[,1]
sse<-sum(sqr_err)

# Root Mean Square Error = sqrt(SSE/N)
RMSE<-sqrt(sse/(nrow(test.set)))
RMSE
```

```
## [1] 89.55635
```