PCA/PLS Regression

Aufgabe 1

a)

```
data = subset(prostate, select=c(lcavol,lweight,age,lbph,svi,lcp,gleason,pgg45))
cmat = cor(data)
kable(cmat, align = 'c', digits = 3)
```

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45
lcavol	1.000	0.281	0.225	0.027	0.539	0.675	0.432	0.434
lweight	0.281	1.000	0.348	0.442	0.155	0.165	0.057	0.107
age	0.225	0.348	1.000	0.350	0.118	0.128	0.269	0.276
lbph	0.027	0.442	0.350	1.000	-0.086	-0.007	0.078	0.078
svi	0.539	0.155	0.118	-0.086	1.000	0.673	0.320	0.458
lcp	0.675	0.165	0.128	-0.007	0.673	1.000	0.515	0.632
gleason	0.432	0.057	0.269	0.078	0.320	0.515	1.000	0.752
pgg45	0.434	0.107	0.276	0.078	0.458	0.632	0.752	1.000

Man erkennt, dass “pgg45” und “lcavol” am stärksten mit anderen Variablen korrelieren. Insbesondere “pgg45” und “gleason” weisen eine hoh positive Korrelation von 0.752 auf. Wenn wir Variablen mit hoher Korrelation als Prädiktor in das Modell hinzufügen, dann werden die Regressionskoeffizienten davon stark beeinflusst. Diese Multikollinearität ist insofern schlecht, da die Prognose dadurch in Richtungen gezerrt wird, die vom eigentlichen Ergebnis abweicht. Um dies zu vermeiden gibt es neben Variablen aus dem Modell nehmen verschiedene Verfahren. In Übung 4 nutze man dazu Ridge/Lasso Regression. Hier nun andere Methoden.

b)

```
data = subset(
  prostate,
  select = c(lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45, lpsa)
)
pcr = pcr(
  lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,
  data = data,
  validation = "CV",
  scale = TRUE
)
summary(pcr)
#> Data:    X dimension: 97 8
#>  Y dimension: 97 1
#> Fit method: svdpc
#> Number of components considered: 8
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV             1.16   0.8516   0.8569   0.7706   0.7683   0.7626   0.7656
#> adjCV          1.16   0.8504   0.8559   0.7685   0.7664   0.7609   0.7631
#>        7 comps  8 comps
#> CV      0.7493   0.7259
#> adjCV   0.7446   0.7227
```

```
#>
#> TRAINING: % variance explained
#>       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
#> X       42.01    62.61    74.81    82.71    88.75    94.28    97.56
#> lpsa    47.04    47.67    58.61    59.45    60.73    61.66    64.21
#>       8 comps
#> X      100.00
#> lpsa    66.34
```

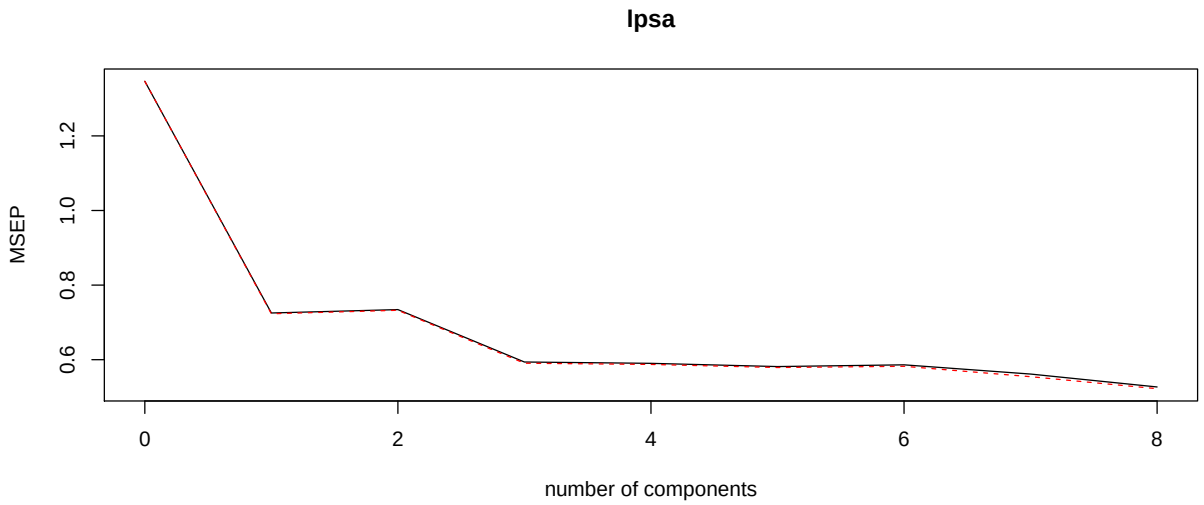
Wir sehen hier, dass wir mit 4 Hauptkomponenten schon 82.71% der Varianz erklären können. Danach steigt die zunahme der erklärten Varianz nur noch langsam. Bis zu 6 Hauptkomponenten könnte man noch sinnvoll in das Modell mit aufnehmen. Mehr würde dem eigentlich Sinn widersprechen die Anzahl an Prädiktoren zu senken, um Multikolinearität zu reduzieren.

c)

In der Summary bekommen wir zu den Cross-Validations den RMSEP (Root Mean Squared Error) mit ausgegeben. Wir erhalten den kleinsten Fehler mit 0.7583 für 8 Hauptkomponenten. Dies ist verständlich, da mit allen Hauptkomponenten auch 100% der Varianz erklärt werden. Dies heißt aber nicht, dass das Modell dann auf einem anderen (unter Umständten stark unterschiedlichen) Datensatz immer noch die besten Ergebnisse liefert.

d)

```
validationplot(pcr, val.type = "MSEP")
```



Sinnvoll ist es eine Anzahl an Hauptkomponenten zu wählen, wo sich nicht mehr viel ändert. Dies kann man anhand eines Plot mittels Elbow-Methode erreichen. Sprich wir schauen, wo auf der Kurve etwa der Ellbogen liegt und nehmen dann den Wert an dieser stelle. Schauen wir uns den Plot an, so erscheint es sinnvoll sich für 3 Hauptkomponenten zu entscheiden.

e)

```
train = prostate[prostate$train == TRUE, ]
test = prostate[prostate$train == FALSE, ]
pcr = pcr(
  lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,
  data = train,
  scale = TRUE
)

y.true = test$lpsa
y.pred = predict(pcr, test, ncomd = 3)

MSE = mean((y.true - y.pred)^2)
MSE = round(MSE, 3)
MSE
#> [1] 0.538
```

Wir erhalten einen MSE von 0.538. Vergleichen wir den Wert mit den Ergebnissen aus der letzten Übung, so stellen wir fest, dass der MSE etwa auf gleichem Niveau ist, wie der von Ridge/Lasso-Regression. Mittels Parametertuning lässt sich bei den anderen Regressionsverfahren ein besseres Ergebnis erreichen, aber dafür ist dafür auch in deutlich höhrer Aufwand nötig. Die Ergebnisse der PCA sind für den gegebenen Aufwand sehr gut.

A2

a)

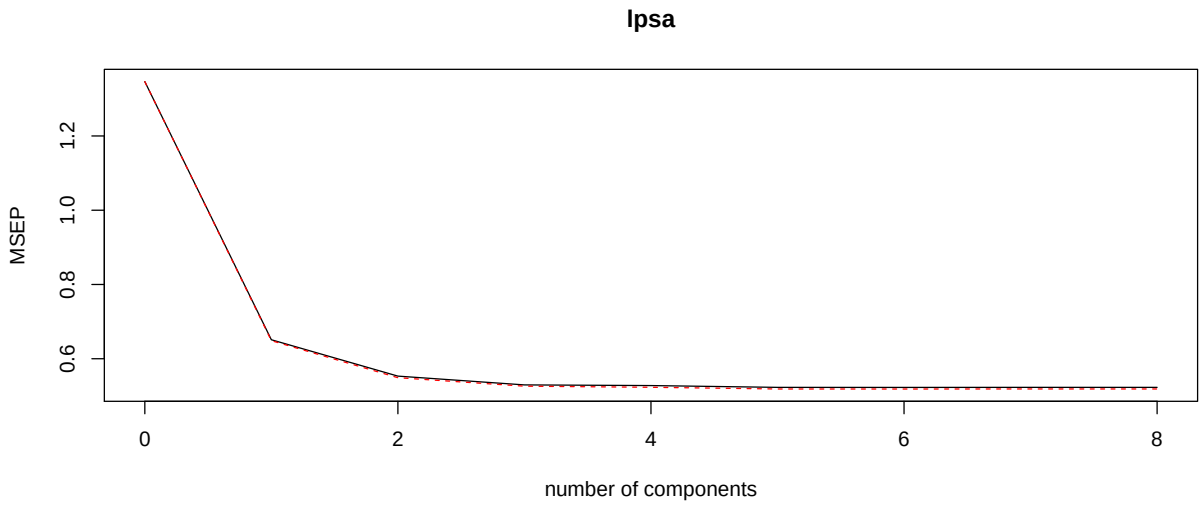
```
data = subset(
  prostate,
  select = c(lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45, lpsa)
)
plsr = plsr(
  lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,
  data = data,
  validation = "CV",
  scale = TRUE
)
summary(plsr)
#> Data:    X dimension: 97 8
#>  Y dimension: 97 1
#> Fit method: kernelpls
#> Number of components considered: 8
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV            1.16   0.8069   0.7438   0.7277   0.7265   0.7231    0.723
#> adjCV         1.16   0.8054   0.7409   0.7253   0.7233   0.7201    0.720
#>        7 comps  8 comps
#> CV      0.7231   0.7231
#> adjCV   0.7201   0.7201
#>
#> TRAINING: % variance explained
#>       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
#> X       41.30    53.96    66.37    78.64    84.35    90.34    94.36
#> lpsa    54.62    63.55    65.31    66.12    66.32    66.34    66.34
#>       8 comps
#> X      100.00
#> lpsa    66.34
```

b)

Hier sinkt der Fehler schneller im Vergleich zur PCA. Der geringste Fehler ist 0.7411 für 3 Komponenten.

c)

```
validationplot(plsr, val.type = "MSEP")
```



Nach der Elbow-Methode bieten sich hier 2 Komponenten an.

d)

```
train = prostate[prostate$train == TRUE, ]
test = prostate[prostate$train == FALSE, ]
plsr = plsr(
  lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,
  data = train,
  scale = TRUE
)

y.true = test$lpsa
y.pred = predict(plsr, test, ncomd = 2)

MSE = mean((y.true - y.pred)^2)
MSE = round(MSE, 3)
MSE
#> [1] 0.509
```

Wir erhalten einen MSE von 0.509. Dieser Wert ist besser als das Ergebnis von Aufgabe 1 und näher an den Ridge/Lasso-Regression Werten aus der vorherigen Übung aber nicht besser als diese. Das macht auch Sinn, da PLSR vom Aufwand her zwischen PSA und Ridge/Lasso-Regression liegt.