Bodenatmung im Nationalpark Hainich

Erstellung eines statistischen Modells und Diskussion von Fehlentscheidungen im Rahmen der Variablenselektion

Sichtling C Loos D Jajali R

8. September 2016

PROJEKTARBEIT

im Rahmen des Moduls *Statistische Verfahren*
an der Friedrich-Schiller-Universität Jena

Die Freisetzung von CO_2 in die Atmosphäre ist ein fundamentaler Bestandteil von Modellen zur Berechnung klimatischer Phänomene. Die *Bodenatmung* ist hierbei der dominierende Prozess auf Landgebieten. Zur Modellierung dieses Prozesses wurde im Nationalpark Hainich Messdaten im Einfluss von ca. 30 Variablen erhoben. Daraus wird nun ein statistisches Modell erstellt. Bei der Variablenselektion können statistische Unsicherheiten zu Fehlern führen. Dies wird im folgendem diskutiert.

Inhaltsverzeichnis

1 Erstellung eines Modells zur Bodenatmung

Der Datensatz enthält sehr viele Features im Vergleich zur Anzahl an Messungen. Der Suchraum der Variablenselektion wäre viel zu groß.

1.1 Test auf Vorliegen einer Normalverteilung

Es werden nur normalverteilte (*Shapiro-Wilk*) und stark korrelierende (*Pearson*) Variablen in Betracht gezogen. Um *Pearson* als Korrelationskoeffizienten nutzen zu können, müssen die Messungen der Variablen normalverteilt sein. Um dies zu testen nutzen wir den *Shapiro-Wilk-Test*. Wenn die *p-Value* des *Shapiro-Wilk-Test* kleiner als 0.05 ist, dann ist die Variable normalverteilt. Wir in Abbildung 1 zu sehen ist, sind nur 14 der insgesamt 33 Variablen normalverteilt.

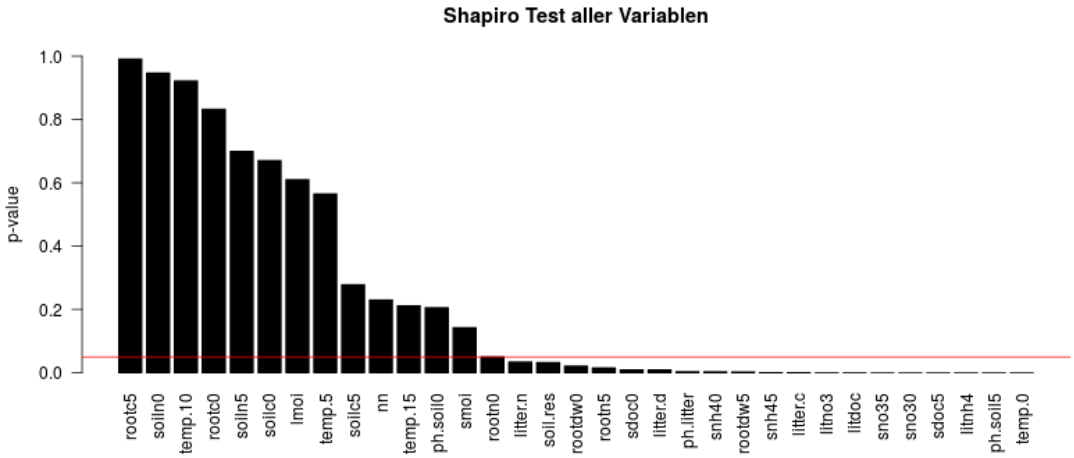

Abbildung 1: **P-Value des Shapiro-Wilk-Test aller Variablen.**

1.2 Korrelationsanalyse

Aus den normalverteilten Variablen werden die am stärksten korrelierenden Variablen ausgewählt (siehe Abbildung 2).

- *lmoi* relative Feuchte der Streuschicht (litter moisture)
- *temp15* Bodentemperatur in 15 cm Tiefe
- *smoi* relative Bodenfeuchte (soil moisture)
- *soiln0* Stickstoffgehalt des Bodens in 0-5 cm Tiefe

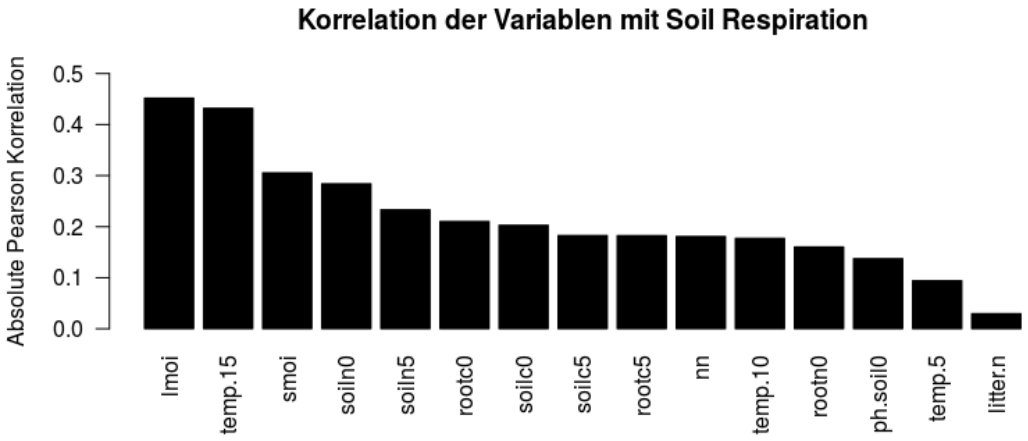

Abbildung 2: **Pearson-Korrelation** der normalverteilten Einflussgrößen (p-*Value* $>$ 0.32) mit der Bodenatmung.

1.3 Transformationen

Transformationen bringen keine deutlichen Verbesserungen der Variablen. Für diese Analyse wurde ein Scatterplot genutzt (Abbildung 7 im Anhang).

1.4 Variablenselektion

Mit Hilfe des R-Pakets *leaps* wird das Modell mit dem geringsten Wert des *Bayesschem Informationskriteriums (BIC)* ausgewählt, welches das Kriterium der Modellqualität erfüllt. In Abbildung 3 sieht man, dass alle Variablen außer *soiln0* gewählt werden.

1.5 Ergebnis

$$soil.res = \vec{\beta} * (1, lmoi, temp15, smoi)^T$$

1.6 Umsetzung mit R

```
1 # Test auf Normalverteilung
2 hainich.shapiro <- mapply(function(x) shapiro.test(x)$p.value,
    hainich)
3 hainich.normal <- hainich[names(hainich.shapiro[hainich.shapiro >
    0.032])]
4 # Korrelation mit Pearson
5 hainich.pear <- abs(cor(hainich.normal))["soil.res",-16]
```

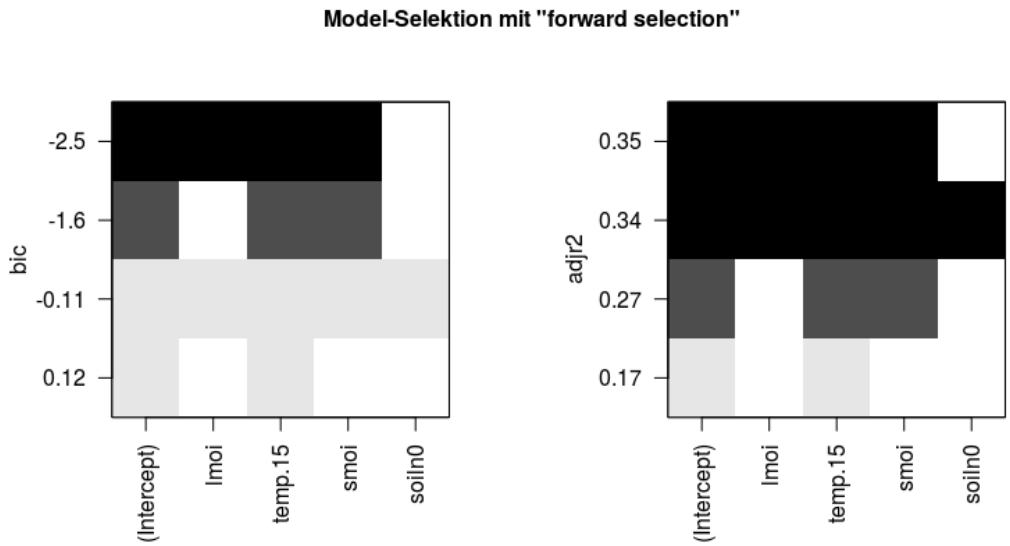

Abbildung 3: **Variablenselektion mittels Bayesschem Informationskriterium (bic) und korrigiertes Bestimmtheitsmaß $\bar{R}^2$ (adjr22)**. Im linken Bild sieht man die Variablenselektion anhand des *Bayesschem Informationskriteriums* und des *korrigierte Bestimmtheitsmaßes*, die beide das Modell mit den ersten drei Variablen (*lmoi, temp.15, smoi*) bevorzugt.

2 Simulation

In Kapitel 1 wurde folgendes Modell ausgewählt:

$$soil.res = \vec{\beta} * (1, lmoi, temp15, smoi)^T \tag{1}$$

2.1 Vorgehensweise

Die Daten wurden in 80% zum Training und 20% zum Testen aufgeteilt. Anschließend wurde das "wahre"Modell um jeweils ein weiteres Feature erweitert. Das erweiterte Modell $soil.res = \vec{\beta} * (1, lmoi, temp15, smoi, x)^T$ wurde mittels ANOVA mit dem "wahren"Modell verglichen. In R berechnete die Funktion `anova(realModel, nxtModel )$F[2]` den jeweiligen F-Wert. Wichtig hierbei war, dass beide Modelle auf den selben Trainingsdaten erstellt wurden und es sich um *verschachtelte* Featuremengen handelte. Die Partitionierung der Daten erfolgte sowohl mittels Kreuzvalidierung als auch mit einem Monte-Carlo-Ansatz. Bei der 4-fachen Kreuzvalidierung wurden die Daten in vier

möglichst gleich großen Partitionen aufgeteilt. Eine Partition wurde zum Testen des Reproduktionsfehlers verwendet, die Anderen zum Training. Beim Monte-Carlo-Ansatz wurden 100 mal zufällig 20% der Daten zum Testen ausgewählt. Dadurch kann die Wahrscheinlichkeit der Fehlentscheidungen im Bezug auf die Nullhypothese besser geschätzt werden. Die Ermittlung des Reproduktionsfehlers $\widehat{SPSE}$ erfolgte mittels unabhängigen Testdaten. Um zu überprüfen, ob sich das Modell überhaupt verbessert hat, wurde die Differenz $\Delta\widehat{SPSE}$ im Bezug auf das Ausgangsmodell berechnet.

2.2 Auswertung

In Abbildung 4 sind die F-Werte und Fehler bei der Reproduktion von unabhängigen Testdaten gezeigt. Im Rahmen des Variablenselektionsverfahrens wird jeweils das Feature zum Modell hinzugenommen, welches um den größten F-Wert verfügt. Im diesem Falle wäre das *litter.d*. Betrachtet man allerdings den erwarteten Reproduktionsfehler $\widehat{SPSE}$, so führt das Hinzunehmen der Variable *ph.soil*5 zum Modell mit minimalen Fehlern. Hier ist der F-Wert aber um $\approx 3,5$ geringer. Die Nullhypothese wird hier fälschlicherweise abgelehnt und stattdessen *litter.d* zum Modell hinzugenommen. Auch kann es vorkommen, dass eine Zufallsvariable überwiegend Rauschen beinhaltet und sich das Modell dann durch hinzufügen ebendieser Variable lediglich verschlechtert (Vgl. Abb. 6)
Nach der Monte-Carlo-Simulation war das Modell mit dem geringsten $\widehat{SPSE}$ stets das Modell mit den 9. höchsten F-Wert. Während der 200 Simulationen verblieb der Wert unverändert.

Unter der Nullhypothese (Erweitertes Modell ist nicht besser als das "wahre"Modell) ist die F-Statistik eine F-verteilte Zufallsvariable. Die Dichtefunktion der Daten aus der Kreuzüberprüfung (approx. durch Kernel-Desity) ist nach Abb. 5 mitunter stark abweichend. Die Simulation möchte demnach das wahre Modell erweitern (H_0 wird abgelehnt). Dies führt zu Overfitting, da das Modell bereits als wahr angenommen wurde und demnach keine weiteren Features hinzugefügt werden sollten.

2.3 Diskussion

2.3.1 Mögliche Gründe

Dies kann mehrere Gründe haben: Gemessene Features müssen nicht zwangsläufig von der Bodenatmung statistisch abhängig sein. Je geringer die Korrelation dieser beiden Zufallsvariablen, desto höher ist die Wahrscheinlichkeit, dass es sich nur um Rauschen handelt. In diesem Fall sollte diese Variable nicht zum Modell hinzugefügt werden.

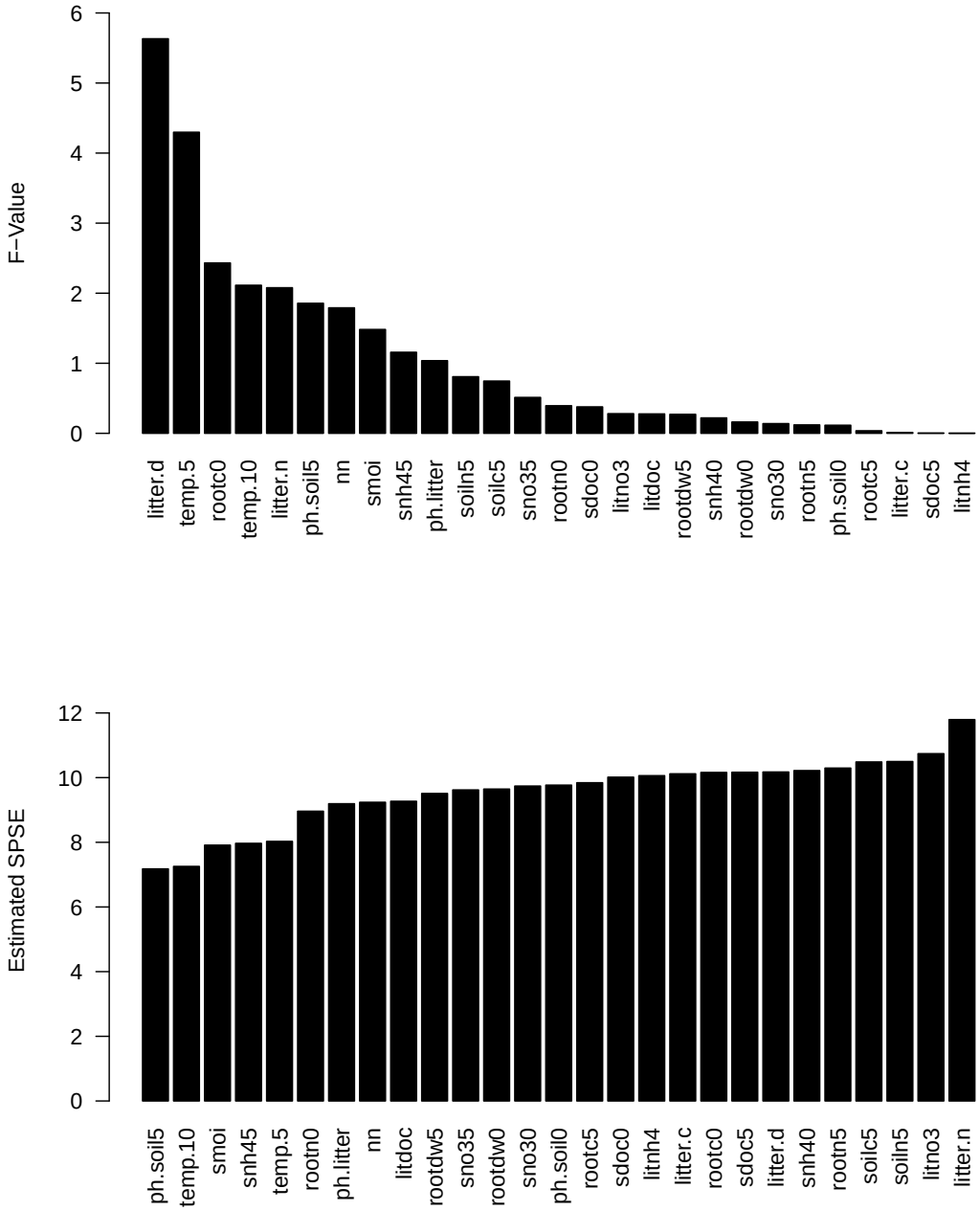

Abbildung 4: **Beispiel einer Simulation.** Das in Kapitel 1 erstellte "wahre"Modell wurde um jeweils ein Feature erweitert. Dargestellt ist der Fehler **SPSE** und der zugehörige **F-Wert** in Abhänigkeit von der gewählten zusätzlichen Variable in der ersten Runde der Kreuzvalidierung. Im Verlaufe der Variablenselektion wird das Modell mit dem höchsten F-Wert gewählt. Dies ist allerdings selten das Modell mit dem geringsten Fehler.

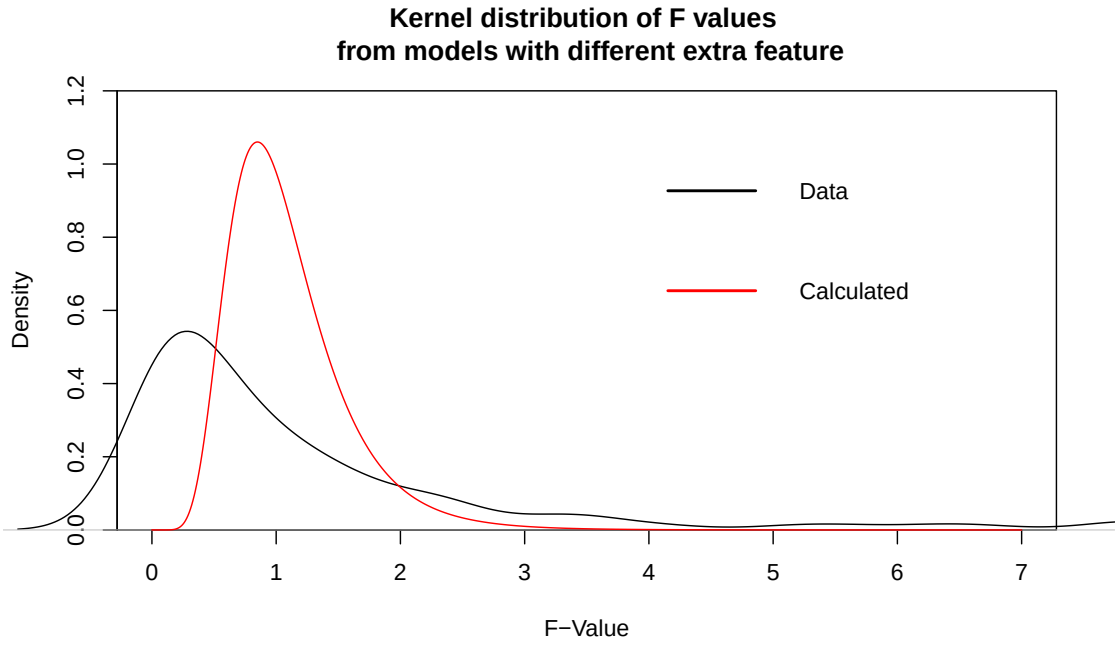

Abbildung 5: **Approximierte Dichtefunktionen der F-Werte.** Gezeigt ist die Kerneldichtefunktion der F-Werte der Simulationen und die berechnete F-Dichtefunktion mit den Freiheitsgraden der ANOVA-Tabelle unter Berücksichtigung der Nullhypothese. Der Unterschied der beiden Funktionen deutet auf fehlerhafte F-tests hin.

Zufälligerweise kann dies aber auch zu hohen F-Werten führen, welches zu fehlerhaften Entscheidungen bei den Hypothesentests führt. Ferner fordert der F-Test normalverteilte Variablen. Folgt eine Messgröße nicht dieser Verteilung, ist der Test nicht aussagekräftig. Auch eine nicht repräsentative Stichprobe fälscht das Ergebnis. Uns lagen lediglich 38 Observationen vor; dies könnte zu gering sein. Eine fehlerhafte Datenerhebung führt auch zu verfälschten Ergebnissen. Letztendlich kann der F-Test auch nur zufällig richtig sein. Ein geringer p-Value schließt keine Fehlentscheidungen aus; sie werden lediglich unwahrscheinlicher.

2.3.2 Auswirkungen der Fehlentscheidungen

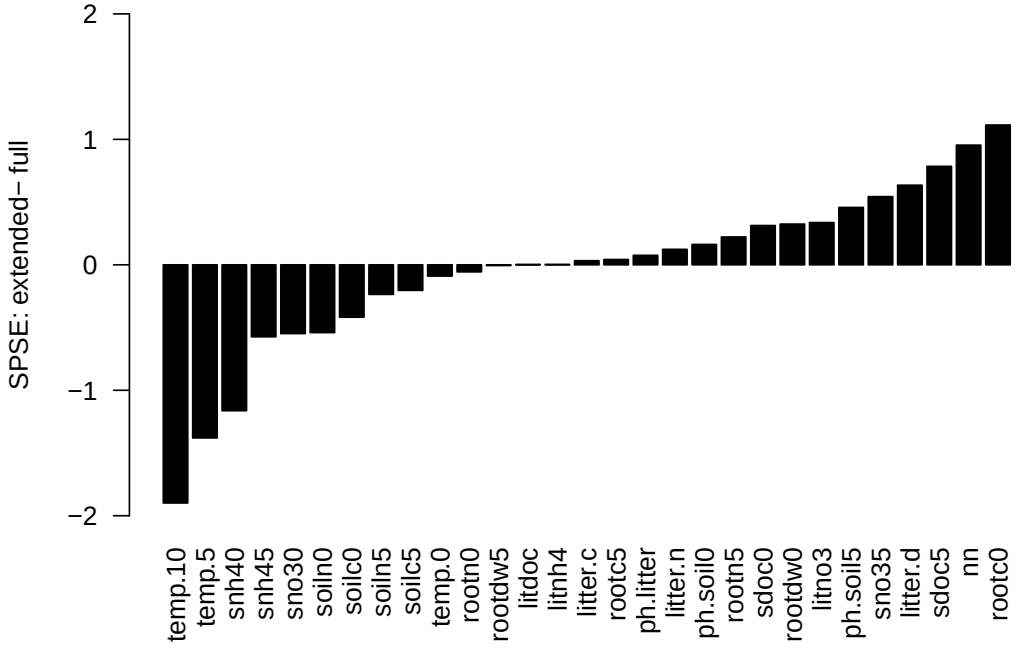

Abbildung 6: **Verbesserung der Modelle.** Dargestellt ist die Differenz $\Delta\widehat{SPSE}$ aus dem „wahren“ und dem erweiterten Modell. Positive Werte deuten darauf hin, dass die hinzugefügte Variable dem Modell lediglich Rauschen beifügt.

3 Code

3.1 Korrelation

```
hainich <- read.csv("hainich.csv", sep = ";", dec = ".")

# spearman (monoton)
hainich.spear <- abs(cor(hainich, method = "spearman"))["soil.res",-1]
hainich.spear.ordered <- hainich.spear[order(hainich.spear, decreasing = T)]
barplot(hainich.spear.ordered, las = 2, ylim = c(0,0.5), col = "black",
        ylab = "Absolute Spearman Correlation", main = "Variables Correlated with Soil Respiration")

# pearson (linear)
hainich.pear <- abs(cor(hainich, method = "pearson"))["soil.res",-1]
hainich.pear.ordered <- hainich.pear[order(hainich.pear, decreasing = T)]
barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "black",
        ylab = "Absolute Pearson Correlation", main = "Variables Correlated with Soil Respiration")
# p.value < 0.05 bedeuted nicht normalverteilt
shapiro.test(hainich[,1])$p.value

hainich.pear <- abs(cor(hainich, method = "pearson"))["soil.res",-1]
hainich.pear.ordered <- hainich.pear[order(hainich.pear, decreasing = T)]
barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "black",
        ylab = "Absolute Pearson Correlation", main = "Variables Correlated with Soil Respiration")

# Vergleich der Barplots
spearpear = t(data.frame(hainich.spear, hainich.pear))
spearpear.ordered <- spearpear[,order(apply(spearpear, c(2),FUN=mean), decreasing = T)]
```

```r
barplot(as.matrix(spearpear.ordered), las = 2, beside=TRUE)

# Shapiro normalverteilt?
hainich.shapiro <- mapply(function(x) shapiro.test(x)$p.value,
    hainich)
hainich.shapiro.ordered <- hainich.shapiro[order(hainich.shapiro,
    decreasing = T)]
barplot(hainich.shapiro.ordered, las = 2, ylim = c(0,1), col = "
    black",
        ylab = "Shapiro Test p-value", main = "Variables")
abline(h=0.05, col="red")
shapiro.test(hainich[,14])

names(hainich.shapiro.ordered)

# Basic Scatterplot Matrix
## Top 15 Spearman Variablen
pairs(~ lmoi + temp.15 + soiln0 + temp.0 + soilc0 + rootdw0 +
    rootdw5 + sdoc5 + soil.res,data=hainich ,main="Simple
    Scatterplot Matrix")
## Top 15 Pearson Variablen
pairs(~ lmoi + temp.15 + litdoc + litter.d  + smoi + rootdw0 +
    temp.0 + sdoc5 + soil.res,data=hainich ,main="Simple
    Scatterplot Matrix")
```

3.1.1 Korrelation der Temperatur

```r
#
# Has Temperature an non linear effect on soil-resp?
#

hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
plot(hainich$temp.0,hainich$soil.res,
     xlim = c(11,13.5),
     ylab = "Soil Respiration", xlab = "Soil Temperature")
points(hainich$temp.5,hainich$soil.res, col = "red")
points(hainich$temp.10,hainich$soil.res, col = "blue")
legend("topright",c("0cm","5cm","10cm"), lwd=2,
       col=c("black","red","blue"), bty = "n")
```

3.2 Variablenselektion

```r
hainich <- read.csv("hainich.csv", sep = ";", dec = ".")

# Correlation mit Pearson
hainich.pear <- abs(cor(hainich))["soil.res",-1]
hainich.pear.ordered <- hainich.pear[order(hainich.pear,
    decreasing = T)]
png(file="../doc/fig/correlation-pearson.png",width=800,height
    =360)
barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "
    black",
        ylab = "Absolute Pearson Korrelation", main = "Korrelation
            der Variablen mit Soil Respiration")
dev.off()

names(hainich.pear.ordered)
# fig/scatterplot-pearson-top8.png
png(file="../doc/fig/scatterplot-pearson-top8.png",width=800,
    height=800)
pairs(~ lmoi + temp.15 + litdoc + litter.d  + smoi + rootdw0 +
    temp.0 + soiln0, data=hainich ,main="Scatterplot der Top 8
    korrelierenden Variablen nach Pearson")
dev.off()

# Shapiro normalverteilt?
hainich.shapiro <- mapply(function(x) shapiro.test(x)$p.value,
    hainich)
hainich.shapiro.ordered <- hainich.shapiro[order(hainich.shapiro,
    decreasing = T)]
png(file="../doc/fig/normalverteilung-shapiro.png",width=800,
    height=360)
barplot(hainich.shapiro.ordered, las = 2, ylim = c(0,1), col = "
    black",
        ylab = "p-value", main = "Shapiro Test aller Variablen")
abline(h=0.05, col="red")
dev.off()

hainich.normal <- hainich[names(hainich.shapiro.ordered[hainich.
    shapiro.ordered > 0.05])]
```

```r
# fig/scatterplot-pearson-top8-normalverteilt.png
png(file="../doc/fig/scatterplot-pearson-normalverteilt.png",width=800,height=800)
dev.off()
#pairs(~ lmoi + log(temp.15) + smoi + soiln0, data=hainich ,main="Simple Scatterplot Matrix")

#shapiro.test(hainich$soiln0)$p.value
#shapiro.test(log(hainich$soiln0))$p.value

#pairs(~ soil.res + exp(temp.15) + exp(temp.10) + exp(temp.5) + exp(temp.0), data=hainich[2:38,])

# sampling training data
set.seed(1337)
testFraction <- 0.2 #Fraction of data used for testing (estimate for integer sample size)
sampleCount <- dim(hainich)[1]
index <- sample(1:sampleCount, round(testFraction * sampleCount))
hainich.test <- hainich[index,]
hainich.train <- hainich[-index,]

null <- lm(soil.res ~ 1, data = hainich.train)
# Top 15 Variablen
#full <- lm(soil.res ~ 1 + lmoi + temp.15 + litdoc + litter.d + smoi + rootdw0 + temp.0 + soiln0 + sno35 + soiln5 + rootc0 + rootdw5 + soilc0 + sdoc5 + soilc5, data = hainich.train)
# Top 6 Variablen
#full <- lm(soil.res ~ 1 + lmoi + temp.15 + litdoc + litter.d + smoi + rootdw0, data = hainich.train)
# Top 8 nur normalverteilte Variablen = 4
full <- lm(soil.res ~ 1 + lmoi + temp.15 + smoi + soiln0, data=hainich.train)

step(null, scope=list(lower=null, upper=full), direction="forward")

#info: http://www.stat.columbia.edu/~martin/W2024/R10.pdf
library("leaps")
hainich.leaps <- regsubsets(soil.res ~ 1 + lmoi + temp.15 + smoi + soiln0,
```

```
                                data=hainich.train, method = "forward
                                    ")

summary(hainich.leaps)
png("../doc/fig/variablenselektion-bic-adjr2.png", width = 400,
    height = 270)
par(mfrow=c(1,2))
plot(hainich.leaps, scale="bic")
plot(hainich.leaps, scale="adjr2")
mtext("Model-Selektion mit \"forward selection\"", side=3, outer=
    TRUE, line=-3.5, cex=1, font = 2)
dev.off()
summary(hainich.leaps)$bic
```

3.3 Simulation

3.3.1 Monte-Carlo

```
spseHat <- function(model, test){
  return(sum((test$soil.res - predict(model, newdata=test))^2))
}

getFtable <- function(train, test){
  ## Simulation F values
  realModelStr <- "soil.res ~ 1 + lmoi + temp.15 + smoi"
  realModel <- lm(realModelStr, data = train)
  # next model will have another feature in addition
  excludeVarsRegEx <- "(soil.res|lmoi|temp.15|smoi)"
  nextVars <- names(hainich)[
    ! grepl(excludeVarsRegEx,names(hainich))]

  #create result data frame
  simulRes <- NULL
  spseRealModel <- spseHat(realModel, test) # spse of full model
  for(x in nextVars){
    #create formula for next Model by adding current variable
    nxtModelStr <- paste(realModelStr, "+", x)
    nxtModel <- lm(nxtModelStr, data = train)

    #evaluate nextModel
    an <- anova(realModel, nxtModel)
    f <- an$F[2] # F-value
```

```
25      p <- an$`Pr(>F)`[2] # p-value
26      spse <- spseHat(nxtModel, test) # error in new model
27      deltaSpse <- spse - spseRealModel # gained error diff
28      simulRes <- rbind(simulRes, c(spse,x,f,p,deltaSpse))
29    }
30    simulRes <- data.frame(spse=as.double(simulRes[,1]),
31                           addedFeature=simulRes[,2] , F=as.double(
                                 simulRes[,3]),
32                           p=as.double(simulRes[,4]), deltaSpse=as.
                                 double(simulRes[,5]),
33                           stringsAsFactors = T)
34
35    return(simulRes)
36  }
37
38  #n = number of monte carlo simulations
39  #nTrain <- number of data points for training
40  monte <- function(data, n, nTrain){
41    # 1 == train, 0 == test
42    index <- sample(c(rep(1,nTrain),rep(0,dim(data)[1] - nTrain)))
43
44    simulRes <- data.frame()
45    for (i in 1:n){
46      train <- hainich[index==1,]
47      test <- hainich[index==0,]
48
49      res <- getFtable(train,test)
50      res$run <- rep(i, dim(res)[1])
51      #concat new results
52      simulRes <- rbind(simulRes,res)
53    }
54
55    return(simulRes)
56  }
57
58  set.seed(1337)
59  hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
60  n <- 100 # number of monte carlo simulations
61  nTrain <- 25 # number of training data points
62  simulRes <- monte(hainich,n,nTrain)
63
```

```
64
65 ## evaluation
66 # gets a list of features with maximal F (count: topN)
67 # returns true if min(SPSE) is in this list
68 takenFeatures <- data.frame()
69 minSPSEinTopF <- function(round, topN){
70   run <- simulRes[simulRes$run == round,]
71   run <- run[order(run$F, decreasing = T),] # order by F
72   run <- head(run, topN) # get top N with max F
73
74   minSPSE <- simulRes[simulRes$spse == min(simulRes$spse)
75                       & simulRes$run == round,]
76   maxF <- simulRes[simulRes$F == max(simulRes$F)
77                    & simulRes$run == round,]
78
79   #return true if best feature by SPSE is in top list by F value
80   if(minSPSE$addedFeature %in% run$addedFeature){
81     return(TRUE)
82   }
83   else{
84     return(FALSE)
85   }
86 }
87
88 probDecisionWrong <- function(topN){
89   rounds <- max(simulRes$run)
90   wrongRoundsCount <- 0
91
92   #for every round
93   for(i in 1:rounds){
94     if(!minSPSEinTopF(i,topN)){
95       wrongRoundsCount <- wrongRoundsCount + 1
96     }
97   }
98
99   return(wrongRoundsCount / rounds)
100 }
101
102
103 #probability that F test dont get min SPSe in thier top 5 list
104 probWrongTop8 <- probDecisionWrong(8)
```

```
105 probWrongTop9 <- probDecisionWrong(9)
106 probWrongTop10 <- probDecisionWrong(10)
```

3.3.2 Kreuzvalidierung

```
1  spseHat <- function(model, test){
2    # SPSE.hat1
3    return(sum((test$soil.res - predict(model, newdata=test))^2))
4  }
5
6  getFtable <- function(train, test){
7    ## Simulation F values
8    realModelStr <- "soil.res ~ 1 + lmoi + temp.15 + smoi"
9    realModel <- lm(realModelStr, data = train)
10   # next model will have another feature in addition
11   excludeVarsRegEx <- "(soil.res|lmoi|temp.15|smoi)"
12   nextVars <- names(hainich)[
13     ! grepl(excludeVarsRegEx,names(hainich))]
14
15   #create result data frame
16   simulRes <- NULL
17   spseRealModel <- spseHat(realModel, test) # spse of full model
18   for(x in nextVars){
19     #create formula for next Model by adding current variable
20     nxtModelStr <- paste(realModelStr, "+", x)
21     nxtModel <- lm(nxtModelStr, data = train)
22
23     #evaluate nextModel
24     an <- anova(realModel, nxtModel)
25     f <- an$F[2] # F-value
26     p <- an$`Pr(>F)`[2] # p-value
27     spse <- spseHat(nxtModel, test) # error in new model
28     deltaSpse <- spse - spseRealModel # gained error diff
29     simulRes <- rbind(simulRes, c(spse,x,f,p,deltaSpse))
30   }
31   simulRes <- data.frame(spse=as.double(simulRes[,1]),
32                          addedFeature=simulRes[,2] , F=as.double(
                               simulRes[,3]),
33                          p=as.double(simulRes[,4]), deltaSpse=as.
                               double(simulRes[,5]),
34                          stringsAsFactors = T)
35
```

```
36    return(simulRes)
37  }
38
39  crossValPart <- function(data, fold){
40    index <- rep(1:fold, length.out=dim(hainich)[2])
41    index <- sample(index) # mixing
42
43    simulRes <- data.frame()
44    for (i in 1:fold){
45      train <- hainich[index!=i,]
46      test <- hainich[index==i,]
47
48      res <- getFtable(train,test)
49      res$run = rep(i, dim(res)[1])
50      simulRes <- rbind(simulRes,res)
51    }
52
53    return(simulRes)
54  }
55
56  hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
57  set.seed(1337)
58  #result by cross validation (random part against rest validation)
59  fold <- 4
60  simulRes <- crossValPart(hainich,fold)
61
62
63  ## evaluation
64  for(i in seq(fold)){
65    #select run
66    r <- simulRes[simulRes$run == i,]
67    #which model with additional feature was the real winner (SPSE)?
68    idMinErr <-  which(r$spse == min(r$spse))
69    #which model won but was not the best one (F)?
70    idMaxF <- which(r$F == max(r$F))
71
72    print(paste("In␣run", i,
73                "min␣SPSE␣and␣max␣F␣picked␣same␣additional␣feature:"
                    , idMaxF == idMinErr ))
74  }
75
```

```
76 ## plotting
77 xs <- seq(0,7,by = 0.01)
78 freal <- density(simulRes$F)
79 fNominal <- df(xs,25,24)
80 plot(freal, xlim=c(0,7), ylim=c(0,1.2),
81      xlab = "F-Value",
82      main="Kernel distribution of F values \n from models with
         different extra feature")
83 points(xs, fNominal, col="red", type="lines") #F distribution
84 legend("topright",c("Data","Calculated"), lwd=2, col=c("black","
    red"), bty = "n")
85
86 #values of F and spse in run 1
87 run1 <- simulRes[simulRes$run == 1,]
88 run1 <- run1[with(run1, order(F, decreasing = T)), ] # order by F
89 barplot(run1$F, names.arg = run1$addedFeature, ylim = c(0,6),
90         las=2, col="black", ylab = "F-Value")
91
92 run1 <- run1[with(run1, order(spse)), ] # order by spse
93 barplot(run1$spse, ylab = "Estimated SPSE", ylim = c(0,13),
94         names.arg = run1$addedFeature, las=2, col="black")
95
96 run1 <- run1[with(run1, order(deltaSpse)), ] # order by spse
    difference
97 barplot(run1$deltaSpse, ylab = "SPSE: extended - full", ylim = c
    (-2,2),
98         names.arg = run1$addedFeature, las=2, col="black")
```

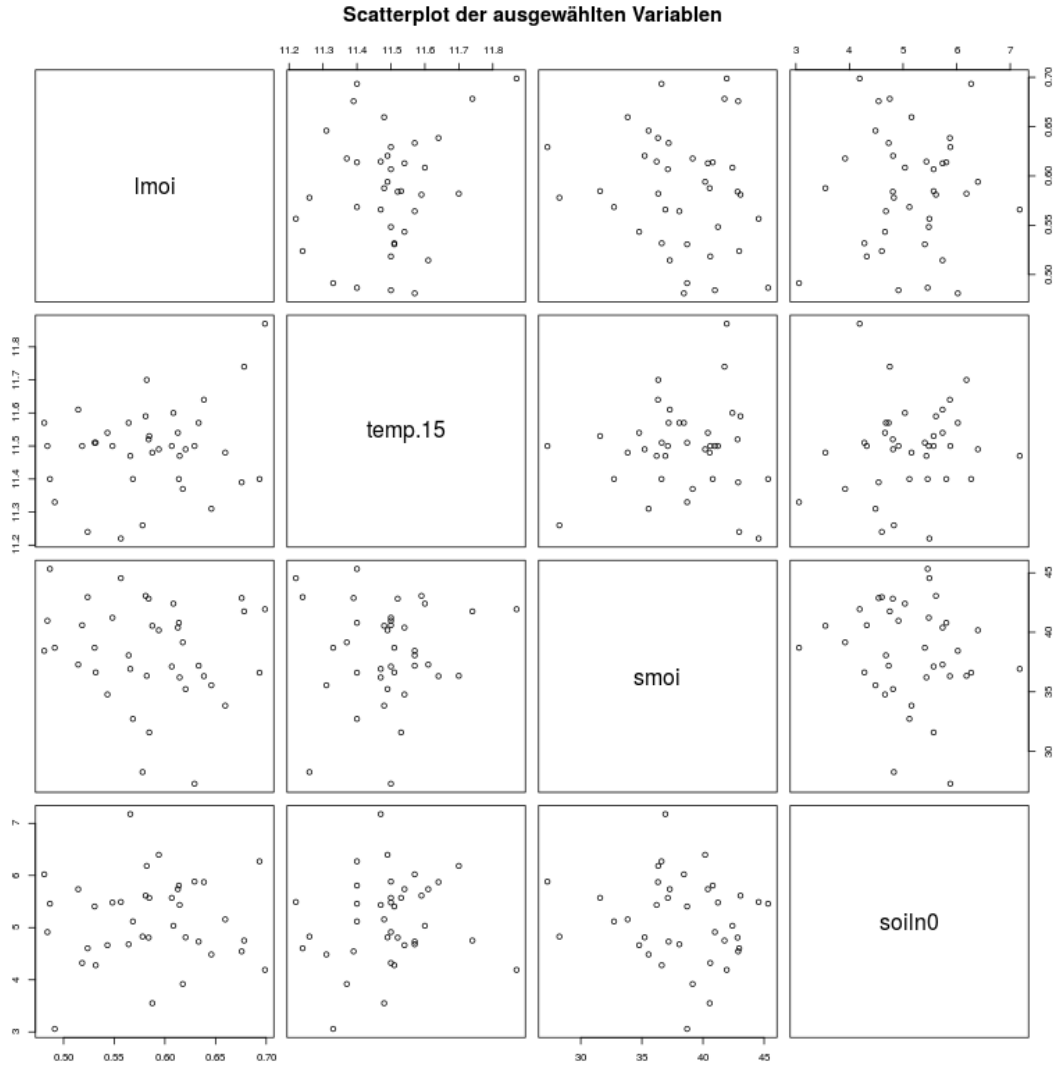

Abbildung 7: Scatterplot zur Betrachtung der Korrelation der ausgewählten Variablen