

# Bodenatmung im Nationalpark Hainich

Erstellung eines statistischen Modells und Diskussion von  
Fehlentscheidungen im Rahmen der Variablen Selektion

Sichtling C

Loos D

Jajali R

7. September 2016

## PROJEKTARBEIT

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

Die Freisetzung von  $\text{CO}_2$  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 Variablen Selektion können statistische Unsicherheiten zu Fehlern führen. Dies wird im folgendem diskutiert.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Bodenatmung als geophysikalischer Prozess . . . . .	3
1.2	Statistische Grundlagen . . . . .	3
1.2.1	Fehlermaße . . . . .	3
1.2.2	Korrelation . . . . .	3
1.2.3	Informationskriterien . . . . .	4
1.2.4	Test auf Vorliegen einer Normalverteilung . . . . .	4
1.2.5	F-Test in geschachtelten Modellen . . . . .	4
<b>2</b>	<b>Erstellung eines Modells zur Bodenatmung</b>	<b>6</b>
2.1	Korrelationsanalyse . . . . .	6
2.2	Transformationen . . . . .	6
2.3	Variablen Selektion . . . . .	6
2.4	Ergebnis . . . . .	6
2.5	Umsetzung mit R . . . . .	6
<b>3</b>	<b>Simulation</b>	<b>8</b>
3.1	Vorgehensweise . . . . .	8
3.2	Auswertung . . . . .	8
3.3	Diskussion . . . . .	10
3.3.1	Mögliche Gründe . . . . .	10
3.3.2	Auswirkungen der Fehlentscheidungen . . . . .	11
<b>4</b>	<b>Code</b>	<b>12</b>
4.1	Korrelation . . . . .	12
4.1.1	Korrelation der Temperatur . . . . .	13
4.2	Variablenselektion . . . . .	13
4.3	Simulation . . . . .	16
4.3.1	Monte-Carlo . . . . .	16
4.3.2	Kreuzvalidierung . . . . .	19

# 1 Einleitung

## 1.1 Bodenatmung als geophysikalischer Prozess

[1]

## 1.2 Statistische Grundlagen

### 1.2.1 Fehlermaße

RSS (*Residual sum of Squares*). SPSE ist der summierte quadratische Fehler zwischen Modellvorhersage und Messwert auf unabhängigen Testdaten:

$$\widehat{SPSE} = \sum_{i=1}^n (y_i - \vec{\beta} * x_i^T)^2 \quad (1)$$

Das Modell wurde vorher mit anderen Trainingsdaten erstellt.

### 1.2.2 Korrelation

Unter *Korrelation* versteht man die „Wechselbeziehung“ zweier Zufallsgrößen. Dieser Zusammenhang kann entweder *linear* (Korrelationskoeffizient nach Pearson) oder lediglich *monoton* sein. Ist eine Korrelation nicht gegeben, so scheint die Zufallsgröße als ungeeigneter Prädiktor für die jeweils andere Variabel. Kausale Beziehungen erfordern Korrelation.

Der *lineare, empirische Korrelationskoeffizient nach Person* zwischen den Variablen  $X$  und  $Y$  wird definiert durch:

$$Kor_e(x, y) := \varrho_e(x, y) := r_{xy} := \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (2)$$

mit den empirischen Mittelwert  $\bar{x}$  aus  $n$  Messwerten von  $X$ . Möchte man lediglich ein *lineares* Modell erstellen, so ist die Pearson-Korrelation zu wählen.

Eine allgemeinere Beschreibung der Korrelation ist die nach *Spearsman*, welche auf Ranglisten beruht:

$$r_s = \frac{\sum_i (rg(x_i) - \overline{rg_x})(rg(y_i) - \overline{rg_y})}{\sqrt{\sum_i (rg(x_i) - \overline{rg_x})^2} \sqrt{\sum_i (rg(y_i) - \overline{rg_y})^2}} \quad (3)$$

$$= \frac{\frac{1}{n} \sum_i (rg(x_i)rg(y_i)) - \overline{rg_x} \overline{rg_y}}{s_{rg_x} s_{rg_y}} \quad (4)$$

$$= \frac{Cov(rg_x, rg_y)}{s_{rg_x} s_{rg_y}} \quad (5)$$

Somit lassen sich auch nicht-lineare Korrelationen in Datensätzen erkennen. Insbesondere im Betrachtung des exponentiellen Verhaltens in Abhängigkeit von der Temperatur ist diese Art der Korrelation wichtig. Zur Berechnung des Anteils an der *erklärten* Varianz in linearen Modellen allerdings kann diese Variante nicht verwendet werden.

### 1.2.3 Informationskriterien

BIC/AIC

### 1.2.4 Test auf Vorliegen einer Normalverteilung

Shapiro

### 1.2.5 F-Test in geschachtelten Modellen

Der *F-Test* überprüft, ob zwei verschachtelte Modelle mit den Featuremengen  $M_1 \subseteq M_2$  sich signifikant unterscheiden. Hierbei wird auf den gleichen Testdaten evaluiert. Die Statistik ist F-verteilt unter Annahme der Nullhypothese und abhängig von den Freiheitsgraden (Anzahl der Features im Modell). Es wird die Nullhypothese überprüft, ob die hinzugefügten Features des erweiterten Modells statistisch irrelevant sind ( $H_0 : \beta_i = 0$ ). Die F-Statistik wird gebildet durch:

$$F = \frac{\frac{RSS_1 - RSS_2}{p_2 - p_1}}{\frac{RSS_2}{n - p_2}} \quad (6)$$

mit den quadrierten Residuen  $RSS$ , den Anzahl an Freiheitsgraden  $p_i$  in den Modellen  $M_1$  und das um weitere Feature erweiterte Modell  $M_2$ . Ist der F-Wert je nach Signifikanzniveau hinreichend groß, kann die Nullhypothese abgelehnt werden: Das erweiterte Modell beschreibt die Daten nun signifikant genauer. Der Fehler  $RSS$  wird so stark verringert, dass es auch eine Erhöhung der Freiheitsgrade rechtfertigt.

Die R-Funktion `anova(model1, model2)` führt derartige Tests durch. Ist der p-Value (in `R F Pr(>F)`) kleiner als das Signifikanzniveau, so wird im Rahmen der Variablenselektion das erweiterte Modell favorisiert.

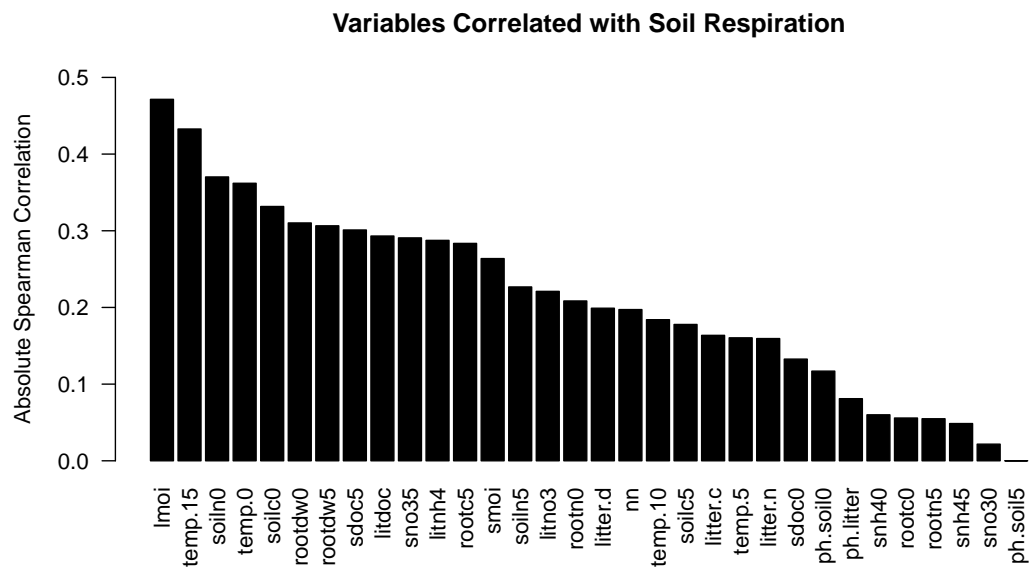


Abbildung 1: **Spearsman-Korrelation** der Einflussgrößen mit der Bodenatmung.

## 2 Erstellung eines Modells zur Bodenatmung

### 2.1 Korrelationsanalyse

### 2.2 Transformationen

$\exp(Tmp)$  anstatt  $Tmp$ ?

### 2.3 Variabelnselektion

### 2.4 Ergebnis

Qualität des gewählten Modells

### 2.5 Umsetzung mit R

```

1 ## correlation
2 #correlation from soil.res with all others
3 hainich.r <- abs(cor(hainich, method = "pearson"))["soil.res",-1]
4 hainich.r.ordered <- hainich.r[order(hainich.r, decreasing = T)]
5 barplot(hainich.r.ordered, las = 2, ylim = c(0,0.5), col = "black"
  ,

```

```
6 ylab = "Absolute_Pearson_Correlation", main = "Variables_
  Correlated_with_Soil_Respiration")
7
8 ## scatter plot
9 pairs(~ rootdw0 + smoi + nn,
10 data = hainich, main = "Scatterplot_Matrix")
```

### 3 Simulation

In Kapitel 2 wurde folgendes Modell ausgewählt:

$$soil.res = \vec{\beta} * (1, lmoi, temp15, smoi)^T \quad (7)$$

#### 3.1 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)` 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.

#### 3.2 Auswertung

In Abbildung 2 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. In diesem Falle wäre das *litter.d*. Betrachtet man allerdings den erwarteten Reproduktionsfehler  $\widehat{SPSE}$ , so führt das Hinzunehmen der Variable *ph.soil5* 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. 4)

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.



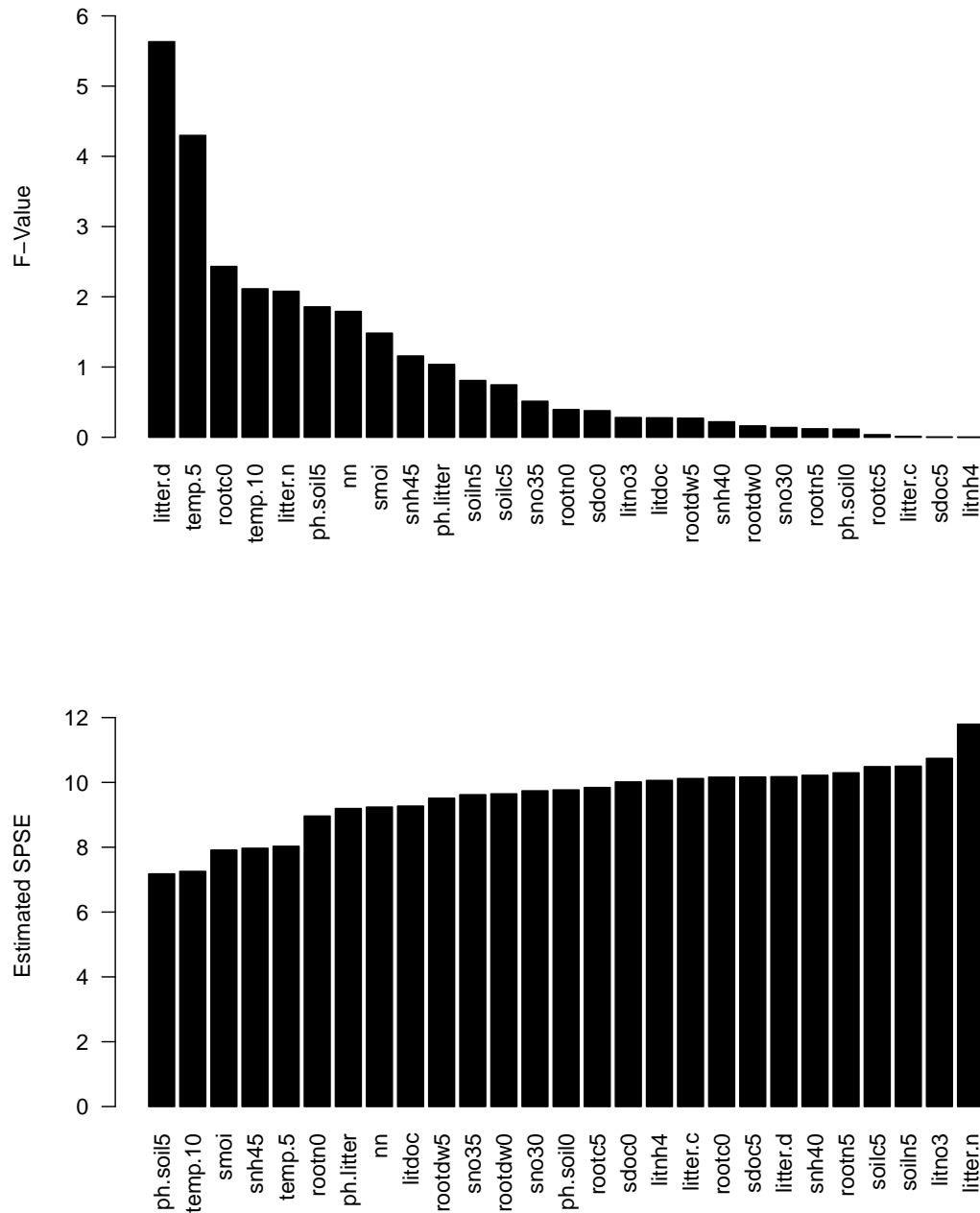


Abbildung 2: **Beispiel einer Simulation.** Das in Kapitel 2 erstellte "wahre" Modell wurde um jeweils ein Feature erweitert. Dargestellt ist der Fehler **SPSE** und der zugehörige **F-Wert** in Abhängigkeit von der gewählten zusätzlichen Variable in der ersten Runde der Kreuzvalidierung. Im Verlaufe der Variablenselektion wird das Modell mit dem höchsten F-Wert gewählt. Dies ist allerdings selten das Modell mit dem geringsten Fehler.

Unter der Nullhypothese (Erweitertes Modell ist nicht besser als das "wahre" Modell)

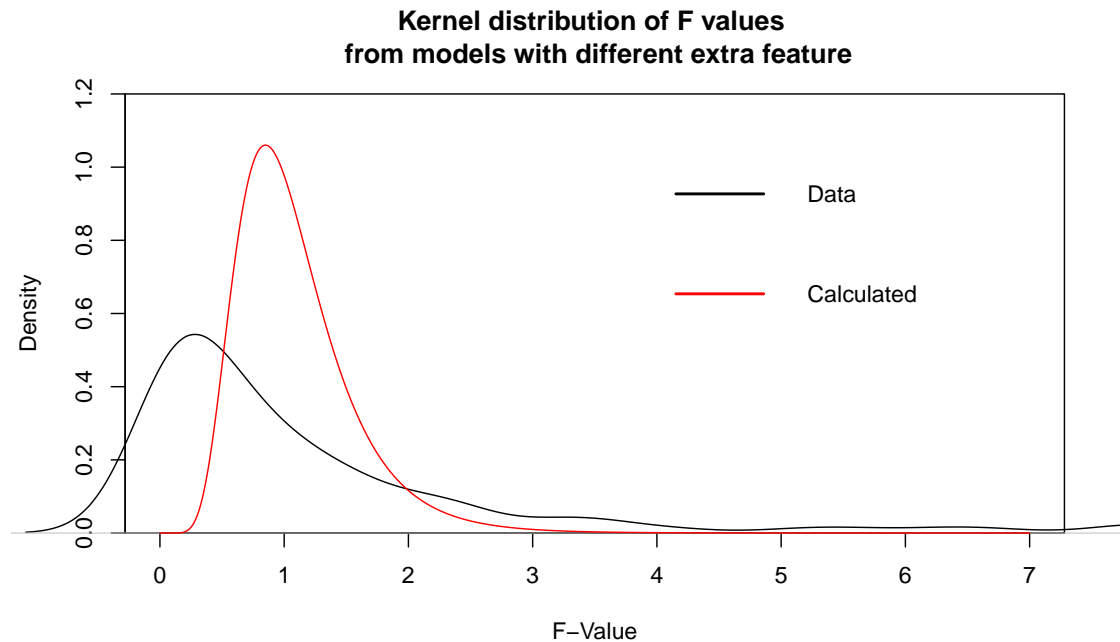


Abbildung 3: **Approximierte Dichtefunktionen der F-Werte.** Gezeigt ist die Kerneldichtefunktion der F-Werte der Simulationen und die berechnete F-Dichtefunktion mit den Freiheitsgraden der ANOVA-Tabelle unter Berücksichtigung der Nullhypothese. Der Unterschied der beiden Funktionen deutet auf fehlerhafte F-tests hin.

ist die F-Statistik eine F-verteilte Zufallsvariabel. Die Dichtefunktion der Daten aus der Kreuzüberprüfung (approx. durch Kernel-Desity) ist nach Abb. 3 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.

### 3.3 Diskussion

#### 3.3.1 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 Variabel nicht zum Modell hinzugefügt werden.

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.

### 3.3.2 Auswirkungen der Fehlentscheidungen

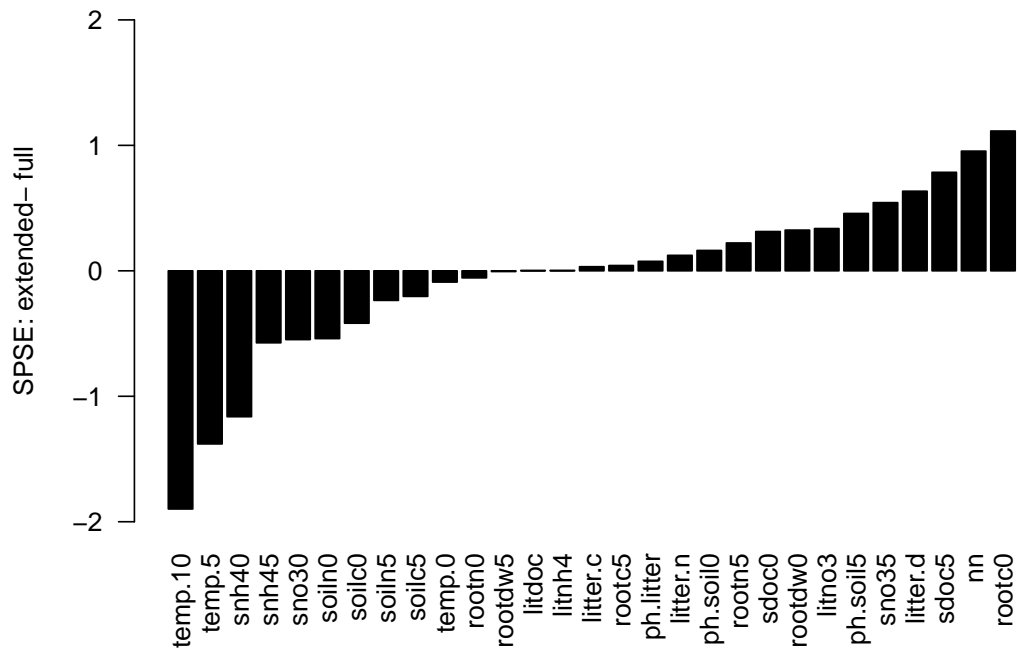


Abbildung 4: **Verbesserung der Modelle.** Dargestellt ist die Differenz  $\Delta \widehat{SPSE}$  aus dem „wahren“ und dem erweiterten Modell. Positive Werte deuten darauf hin, dass die hinzugefügte Variable dem Modell lediglich Rauschen beifügt.

## 4 Code

### 4.1 Korrelation

```
1 hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
2
3 # spearman (monoton)
4 hainich.spear <- abs(cor(hainich, method = "spearman"))["soil.res"
5   ,-1]
6 hainich.spear.ordered <- hainich.spear[order(hainich.spear,
7   decreasing = T)]
8
9 # barplot(hainich.spear.ordered, las = 2, ylim = c(0,0.5), col = "
10   black",
11   ylab = "Absolute_Spearman_Correlation", main = "Variables_
12     Correlated_with_Soil_Respiration")
13
14 # pearson (linear)
15 hainich.pear <- abs(cor(hainich, method = "pearson"))["soil.res"
16   ,-1]
17 hainich.pear.ordered <- hainich.pear[order(hainich.pear,
18   decreasing = T)]
19 barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "
20   black",
21   ylab = "Absolute_Pearson_Correlation", main = "Variables_
22     Correlated_with_Soil_Respiration")
23
24 # p.value < 0.05 bedeutet nicht normalverteilt
25 shapiro.test(hainich[,1])$p.value
26
27 hainich.pear <- abs(cor(hainich, method = "pearson"))["soil.res"
28   ,-1]
29 hainich.pear.ordered <- hainich.pear[order(hainich.pear,
30   decreasing = T)]
31 barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "
32   black",
33   ylab = "Absolute_Pearson_Correlation", main = "Variables_
34     Correlated_with_Soil_Respiration")
35
36 # Vergleich der Barplots
37 spearpear = t(data.frame(hainich.spear, hainich.pear))
38 spearpear.ordered <- spearpear[,order(apply(spearpear, c(2),FUN=
39   mean), decreasing = T)]
```

```

25 barplot(as.matrix(spearpear.ordered), las = 2, beside=TRUE)
26
27 # Shapiro normalverteilt?
28 hainich.shapiro <- mapply(function(x) shapiro.test(x)$p.value,
    hainich)
29 hainich.shapiro.ordered <- hainich.shapiro[order(hainich.shapiro,
    decreasing = T)]
30 barplot(hainich.shapiro.ordered, las = 2, ylim = c(0,1), col = "
    black",
31         ylab = "Shapiro_Test_p-value", main = "Variables")
32 abline(h=0.05, col="red")
33 shapiro.test(hainich[,14])
34
35 names(hainich.shapiro.ordered)
36
37 # Basic Scatterplot Matrix
38 ## Top 15 Spearman Variablen
39 pairs(~ lmoi + temp.15 + soiln0 + temp.0 + soilc0 + rootdw0 +
    rootdw5 + sdoc5 + soil.res,data=hainich ,main="Simple_
    Scatterplot_Matrix")
40 ## Top 15 Pearson Variablen
41 pairs(~ lmoi + temp.15 + litdoc + litter.d + smoi + rootdw0 +
    temp.0 + sdoc5 + soil.res,data=hainich ,main="Simple_
    Scatterplot_Matrix")

```

#### 4.1.1 Korrelation der Temperatur

```

1 #
2 # Has Temperature an non linear effect on soil-resp?
3 #
4
5 hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
6 plot(hainich$temp.0,hainich$soil.res,
7      xlim = c(11,13.5),
8      ylab = "Soil_Respiration", xlab = "Soil_Temperature")
9 points(hainich$temp.5,hainich$soil.res, col = "red")
10 points(hainich$temp.10,hainich$soil.res, col = "blue")
11 legend("topright",c("0cm","5cm","10cm"), lwd=2,
12      col=c("black","red","blue"), bty = "n")

```

#### 4.2 Variablenselektion

```

1 hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
2
3 # Correlation mit Pearson
4 hainich.pear <- abs(cor(hainich))["soil.res",-1]
5 hainich.pear.ordered <- hainich.pear[order(hainich.pear,
      decreasing = T)]
6 png(file="../doc/fig/correlation-pearson.png",width=800,height
      =360)
7 barplot(hainich.pear.ordered, las = 2, ylim = c(0,0.5), col = "
      black",
8         ylab = "Absolute_Pearson_Korrelation", main = "Korrelation
      _der_Variablen_mit_Soil_Respiration")
9 dev.off()
10
11 names(hainich.pear.ordered)
12 # fig/scatterplot-pearson-top8.png
13 png(file="../doc/fig/scatterplot-pearson-top8.png",width=800,
      height=800)
14 pairs(~ lmoi + temp.15 + litdoc + litter.d + smoi + rootdw0 +
      temp.0 + soiln0, data=hainich ,main="Scatterplot_der_Top_8_
      korrelierenden_Variablen_nach_Pearson")
15 dev.off()
16
17 # Shapiro normalverteilt?
18 hainich.shapiro <- mapply(function(x) shapiro.test(x)$p.value,
      hainich)
19 hainich.shapiro.ordered <- hainich.shapiro[order(hainich.shapiro,
      decreasing = T)]
20 png(file="../doc/fig/normalverteilung-shapiro.png",width=800,
      height=360)
21 barplot(hainich.shapiro.ordered, las = 2, ylim = c(0,1), col = "
      black",
22         ylab = "p-value", main = "Shapiro_Test_aller_Variablen")
23 abline(h=0.05, col="red")
24 dev.off()
25
26 hainich.normal <- hainich[names(hainich.shapiro.ordered[hainich.
      shapiro.ordered > 0.05])]
27
28
29

```

```

30 # fig/scatterplot-pearson-top8-normalverteilt.png
31 png(file="../doc/fig/scatterplot-pearson-normalverteilt.png",width
    =800,height=800)
32 dev.off()
33 #pairs(~ lmoi + log(temp.15) + smoi + soiln0, data=hainich ,main="
    Simple Scatterplot Matrix")
34
35 #shapiro.test(hainich$soiln0)$p.value
36 #shapiro.test(log(hainich$soiln0))$p.value
37
38 #pairs(~ soil.res + exp(temp.15) + exp(temp.10) + exp(temp.5) +
    exp(temp.0), data=hainich[2:38,])
39
40 # sampling training data
41 set.seed(1337)
42 testFraction <- 0.2 #Fraction of data used for testing (estimate
    for integer sample size)
43 sampleCount <- dim(hainich)[1]
44 index <- sample(1:sampleCount, round(testFraction * sampleCount))
45 hainich.test <- hainich[index,]
46 hainich.train <- hainich[-index,]
47
48 null <- lm(soil.res ~ 1, data = hainich.train)
49 # Top 15 Variablen
50 #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)
51 # Top 6 Variablen
52 #full <- lm(soil.res ~ 1 + lmoi + temp.15 + litdoc + litter.d +
    smoi + rootdw0, data = hainich.train)
53 # Top 8 nur normalverteilte Variablen = 4
54 full <- lm(soil.res ~ 1 + lmoi + temp.15 + smoi + soiln0, data=
    hainich.train)
55
56 step(null, scope=list(lower=null, upper=full), direction="forward"
    )
57
58 #info: http://www.stat.columbia.edu/~martin/W2024/R10.pdf
59 library("leaps")
60 hainich.leaps <- regsubsets(soil.res ~ 1 + lmoi + temp.15 + smoi +
    soiln0,

```

```

61         data=hainich.train, method = "forward
        ")
62
63 summary(hainich.leaps)
64 png("../doc/fig/variablenselektion-bic-adjr2.png", width = 400,
        height = 270)
65 par(mfrow=c(1,2))
66 plot(hainich.leaps, scale="bic")
67 plot(hainich.leaps, scale="adjr2")
68 mtext("Model-Selektion_mit_\\"forward selection\\", side=3, outer=
        TRUE, line=-3.5, cex=1, font = 2)
69 dev.off()
70 summary(hainich.leaps)$bic

```

## 4.3 Simulation

### 4.3.1 Monte-Carlo

```

1 spseHat <- function(model, test){
2   return(sum((test$soil.res - predict(model, newdata=test))^2))
3 }
4
5 getFtable <- function(train, test){
6   ## Simulation F values
7   realModelStr <- "soil.res~_l_lmoi+_temp.15+_smoi"
8   realModel <- lm(realModelStr, data = train)
9   # next model will have another feature in addition
10  excludeVarsRegEx <- "(soil.res|lmoi|temp.15|smoi)"
11  nextVars <- names(hainich)[
12    ! grepl(excludeVarsRegEx,names(hainich))]
13
14  #create result data frame
15  simulRes <- NULL
16  spseRealModel <- spseHat(realModel, test) # spse of full model
17  for(x in nextVars){
18    #create formula for next Model by adding current variable
19    nxtModelStr <- paste(realModelStr, "+", x)
20    nxtModel <- lm(nxtModelStr, data = train)
21
22    #evaluate nextModel
23    an <- anova(realModel, nxtModel)
24    f <- an$F[2] # F-value

```



```

25     p <- an$'Pr(>F)'[2] # p-value
26     spse <- spseHat(nxtModel, test) # error in new model
27     deltaSpse <- spse - spseRealModel # gained error diff
28     simulRes <- rbind(simulRes, c(spse,x,f,p,deltaSpse))
29 }
30 simulRes <- data.frame(spse=as.double(simulRes[,1]),
31                        addedFeature=simulRes[,2] , F=as.double(
32                        simulRes[,3]),
33                        p=as.double(simulRes[,4]), deltaSpse=as.
34                        double(simulRes[,5]),
35                        stringsAsFactors = T)
36
37     return(simulRes)
38 }
39
40 #n = number of monte carlo simulations
41 #nTrain <- number of data points for training
42 monte <- function(data, n, nTrain){
43     # 1 == train, 0 == test
44     index <- sample(c(rep(1,nTrain),rep(0,dim(data)[1] - nTrain)))
45
46     simulRes <- data.frame()
47     for (i in 1:n){
48         train <- hainich[index==1,]
49         test <- hainich[index==0,]
50
51         res <- getFtable(train,test)
52         res$run <- rep(i, dim(res)[1])
53         #concat new results
54         simulRes <- rbind(simulRes,res)
55     }
56
57     return(simulRes)
58 }
59
60 set.seed(1337)
61 hainich <- read.csv("hainich.csv", sep = ";", dec = ".")
62 n <- 100 # number of monte carlo simulations
63 nTrain <- 25 # number of training data points
64 simulRes <- monte(hainich,n,nTrain)
65

```

```

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)

```

### 4.3.2 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(
33                           simulRes[,3]),
34                         p=as.double(simulRes[,4]), deltaSpse=as.
35                           double(simulRes[,5]),
36                         stringsAsFactors = T)

```

```

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:"
74                 , idMaxF == idMinErr ))
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")

```

## Literatur

[1] N. Jr. My article, 2006.