Gudrun PREGARTNER

Design-of-Experiment
and
Statistical Modeling
of
Large Scale Lithium Ion Cells

MASTER THESIS

written to obtain the academic degree of a Master of Science (MSc)

Masterstudium Operations Research und Statistik

Graz University of Technology

Supervisor:
Univ.-Prof. Dipl.-Ing. Dr.techn. Ernst STADLOBER

Institute of Statistics

Graz, November 2012

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....

(Unterschrift)

ABSTRACT

This thesis was conducted in cooperation with the 'Kompetenzzentrum Das Virtuelle Fahrzeug Forschungsgesellschaft mbH (ViF)' research company, which is dedicated to the system optimization for full vehicles as well as the simulation and testing thereof.

This thesis is divided into three parts. After a short theoretical part giving the basics on linear regression as well as experimental design, it provides a short overview of useful `R` functions to be applied in part three, which consists of the analysis of a data set containing several measurements over time for 175 lithium-ion cells. We define sensible response variables and graphically visualize certain dependencies on a given set of factors. Furthermore, the design of experiments from which the data was basically obtained is analyzed. Special focus is hereby laid on the linear and quadratic influence of each factor as well as two-factor interactions. Finally, the insight gained from the foregoing analysis is used to find linear models for the response variables and to create a new design for an extended experiment.

ZUSAMMENFASSUNG

Diese Arbeit wurde in Zusammenarbeit mit der 'Kompetenzzentrum Das Virtuelle Fahrzeug Forschungsgesellschaft mbH (ViF)' erstellt, welche sich der Systemoptimierung ganzer Fahrzeuge, sowie der Simulation und dem Testen dieser Vorgänge widmet.

Diese Arbeit ist unterteilt in drei Kernbereiche. Anfangs werden die notwendigen theoretischen Grundlagen in linearer Regression und Versuchsplanung kurz abgehandelt. Der zweite Teil umfasst einen Überblick über hilfreiche Funktionen in `R`, welche im dritten Teil ihre Anwendung finden. Der dritte Teil umfasst dann schließlich die Analyse des zugrundeliegenden Datensatzes, welcher mehrere zeitversetzte Messungen von 175 Lithium-Ionen-Zellen beinhaltet. Hierfür werden zuerst sinnvolle Zielvariablen definiert. Eine graphische Analyse wird verwendet, um Abhängigkeiten von einer gewissen Menge an Faktoren zu erkennen. Außerdem wird der Versuchsplan, welcher die Basis für die Experimente und die Datengenerierung darstellte, eingehend analysiert. Hierbei wird vor allem Wert auf die linearen und quadratischen Einflüsse der einzelnen Faktoren, sowie auf Zweifach-Interaktionen gelegt. Zum Schluss werden die Ergebnisse der vorangehenden Analyse verwendet, um lineare Modelle für die Zielvariablen und einen erweiterten Versuchsplan zu erstellen.

ACKNOWLEDGEMENTS

First of all, I want to thank my supervising professor, Dr. Ernst Stadlober, for accepting this thesis and guiding me through my studies for the last few years.

I would also like to express my deepest gratitude to my supervisor DI Wenzel Prochazka (ViF) for his patience, time and help with this thesis.

A warm thank you goes to Dr. Nikolaus Haselgruber (AVL) for providing me with his personal code of his DDoE algorithm, which to our knowledge has not (yet) been made public.

Furthermore, I would like to thank Mr. Herwig Stütz for his ongoing emotional and technical support.

Last but not least, I wish to thank my parents who in the long run made this thesis possible by supporting me financially and emotionally throughout all these years.

Contents

Chapter 1

Introduction

1.1 Motivation

Caused by the heavily limited availability of mineral oil and an ever increasing general ecological awareness, a call for alternative power sources in the automotive industry is getting louder and finally heard. Around the globe, researchers focus intensely on the development of replenishable and affordable solutions. Lithium-Ion batteries seem to currently promise the greatest potential in the field of electric power and thus enjoy a great deal of attention, not least because they are able to (best) meet the high demands regarding safety issues and operating life.

1.2 Lithium-Ion Batteries

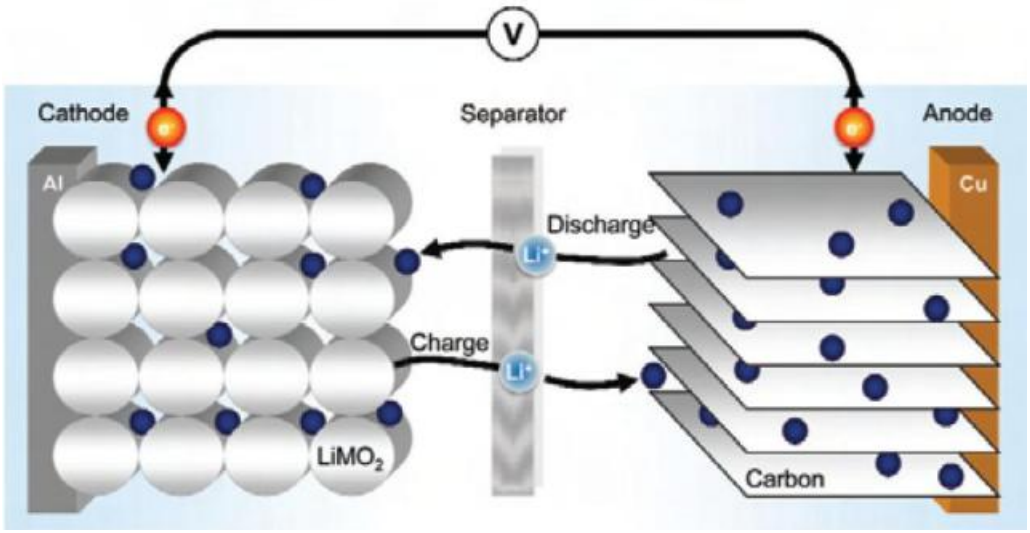

Figure 1.1: Schematic depiction of the charge and discharge process of a lithium ion battery. Graphic taken from `http://www.azom.com/article.aspx?ArticleID=5813`.

A lithium-ion battery consists of three main components, the positive and negative electrodes (anode and cathode) and the separator, which are all filled with electrolyte. During the discharging process the lithium ions move from the anode to the cathode and back again during the charging process.

However, due to several chemical side reactions occurring during this procedure, the battery loses capacity over time and number of operating cycles. The aging performance decay of lithium ion cells under various load conditions has become a major hindrance for their mass market introduction in automotive applications.

Exactly which factors contribute to this aging process to which content has so far been investigated in several publications and the following is intended to give but a short review of relevant research results in the area of identifying significant aging factors:

- Fischnaller et al., [13], discuss the difficulties of directly predicting the life-time.

  The battery's short time behavior depending on the factors temperature, state of charge and/or amplitude already seems to be relatively well known. As a consequence, focus is currently being laid on aging characteristics but some aspects, such as the complex structure of the dependencies and inevitable errors in the measuring process, severely complicate such a complete characterization.

  In order to describe the aging process accurately, a model must be developed. For this, the methods of statistical design of experiments are being deployed. Once a satisfactory model has been found, it is possible to avoid certain usage profiles that lead to fast degradation and thus increase the battery's life-time. At the moment it seems unrealistic to model the aging process of the complete system, so current efforts are being made to describe the *capacity* or *internal resistance* depending on temperature and aging factors which allows certain conclusions regarding residual energy and instantaneous power of the battery.

  Both of these factors are described as functions of the power throughput and thus allow an estimation of the expected life-time. Their results show an intensified degradation of both capacity and internal resistance at a high state of charge as well as for deep discharge profiles. Furthermore, they affirm the well-known effect of increased aging at high operating temperatures and state a smaller, yet possibly still significant, influence of the amplitude. Each of these influences has been tested at two different levels, a high and low level respectively. Note that these results only hold for the influences investigated uncoupled.

- Safari and Delacourt [54], investigate the aging behavior of a certain lithium-ion battery during both cycling and storage at different temperatures.

  They carry out a (not very statistically accurate) aging study of cells under different cycling and storage conditions at two different temperature levels. They come to the conclusion that at a given temperature level the cell's capacity loss is greater under cycling than under storage and that the higher temperature level results in greater capacity loss than the lower one. Furthermore, they state that the aging process mainly is subject to loss of capacity rather than to impedance increase, no matter of the cycling or storage conditions or the applied operating temperature. As

a main source of capacity loss they name the loss of cyclable lithium which might be triggered by the formation of a solid electrolyte inter-phase layer at the graphite electrode. However, numerous other chemical and mechanical aging impacts have already been identified in the past.

Note that much thought has been given to the right choice of measuring technique since it is sought to not lethally damage the cell in the process, which greatly restricts the possibilities.

Generally, we are of course above all interested in long-lasting batteries. The primary goal of this thesis is thus to explain the life span of a battery, in our case determined by measurable parameters such as cell capacity and interior cell resistance, as a function of several influential factors. Once such a model is established, it is possible to predict the outcome for fixed parameter settings. The four factors currently esteemed influential and considered in this thesis are the operating current `Curr`, the operating temperature `Temp`, the battery's momentary state of charge `SoC` and the fluctuation in state of charge `D-SoC`. It is, however, planned to increase the number of potential influences in future investigations in order to improve these predictions.

1.3 Schedule for this Thesis

While working on this thesis it became clear that several analyzing techniques which were initially planned would not merit the practitioner's insight into the matters to the hoped-for extent. Thus, we have decided to alter the course of action slightly and finally wound up with the following schedule.

1. The already existing data is to be analyzed according to standard **graphical statistical analysis** methods.
2. The underlying **design of experiment** is to be analyzed as such although it is in itself unbalanced.
3. The (interpolated) response variables are to be modeled as **ordinary linear models** as best possible by taking into account the factors mentioned above.
4. An **experimental design** is to be created for the acquisition of new data taking into account several additional influences.

It was also considered to analyze the data, which consists of several measurements over time per cell, as **longitudinal data** by means of linear mixed models additionally to the standard analysis of a time-free yet to be defined response. However, it turned out that the number of factors at hand makes such an analysis extremely difficult to interpret, especially for a non-mathematician. If such an analysis were intended for a new experiment it would also be advisable to measure cells at equidistant time points if at all possible.

The thesis is divided into three main parts. First, we start by giving the necessary theoretical preliminaries in the field of experimental design as well as linear models. The second part consists of a short overview of the according functionality in R, the statistic programming language chosen for the analysis throughout. We do not claim any originality in these two parts. All of our own work is then comprised in the third part, which consists of the actual analysis of the battery data set. The experiment itself will be explained to some more detail in this final part.

All computations, including the construction of graphics, were done with R (version 2.15.1), a language and environment for statistical computing and graphics available as open source package. In order to keep the appendix compact, most of this code may be found on the accompanying disc.

Part I

Theoretical Preliminaries

Chapter 2

Ordinary Linear Regression & Model Selection

Even though it must be clear that no model, no matter how complex, can correctly depict reality, considerable effort should be put into the selection of a 'good' model. Montgomery and Myers, [44], summarize the reasons for this very concisely as follows:

> *'Leaving out important regressors introduces bias into the parameter estimates, while including unimportant variables weakens the prediction or estimation capability of the model.'*

This chapter should give some insight into the general domain of linear regression methods as well as into the most commonly used model assessment techniques. It is widely based on lecture notes by Stadlober, [59], and Noble, [48].

2.1 Linear Regression

Regression aims to explain the relationship between a response variable y and one or several independent (or regressor) variables x. For linear regression it is thus sought to fit a straight line to the given data as best possible and thus a model of the following form is assumed:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \varepsilon_i \quad \text{for} \quad i = 1, \ldots, n.$$

Herein, the subscript i indicates the ith observation of the respective variable and n denotes the total number of observations, where $n > p = k + 1$ must hold.

ε_i denotes the error or noise term in the respective observation. This is an unobserved random variable with an expected mean of 0 and a variance of σ^2. Furthermore, the error terms are to be independent of each other.

If $k = 1$, we speak of simple linear regression, whereas in the case of $k > 1$ we are dealing with multiple linear regression. The term 'linear' indicates that the model is linear in the

so-called regression coefficients β_j, not necessarily in the regressor variables x_j for which any functional form is permitted. Note that multiple linear regression will be what we have in mind for the rest of this paper. Finally, the term β_0 is called the intercept term.

Equivalently, the above model can be written in matrix notation, which is often more practical when doing continuative calculations.

$$y = X\beta + \varepsilon \quad \text{with} \quad \varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 I),$$

$$\text{where } y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \; X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{pmatrix}, \; \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \; \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

In order to have a meaningful model for the prediction of the response variable for a given set of values for the regressor variables, the unknown regression parameter vector β needs to be estimated.

2.1.1 Method of Least Squares

The intention behind the method of least squares is to determine the regression coefficients in such a way that the error (or residual) sum of squares

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_j \right)^2$$

is minimized.

Using elementary matrix derivation, the optimal solution turns out to be the well-known least squares estimator

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

Note that $X^T X$ needs to be invertible for this estimator to exist. This can be guaranteed if the matrix X has full rank $p = k + 1$ and it thus holds that $|X^T X| \neq 0$.

Properties of the Least Squares Estimator

There are several good reasons why the least squares estimator is widely used for fitting linear regression models.

- One of the most important properties of $\hat{\beta}$ is that it is **unbiased**, i.e. $E[\hat{\beta}] = \beta$. This property can be derived straightforward when keeping in mind that $E[\varepsilon] = 0$.

  In fact, the least squares estimator is even the best linear unbiased estimator (BLUE).

- The covariance matrix of $\hat{\beta}$ can be determined as $Cov[\hat{\beta}] = E[(\hat{\beta} - E[\hat{\beta}])(\hat{\beta} - E[\hat{\beta}])^T]$ and turns out to be equal to $\sigma^2 (X^T X)^{-1}$.

  This also means that $Var[\hat{\beta}_j] = \sigma^2 (X^T X)^{-1}_{jj}$ and $Cov[\hat{\beta}_{j_1}, \hat{\beta}_{j_2}] = \sigma^2 (X^T X)^{-1}_{j_1 j_2}$.

- From $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 I)$, the least squares estimator also follows a normal distribution:

$$\hat{\beta} \sim N\left(\beta, \sigma^2 (X^T X)^{-1}\right).$$

- For independent and normally distributed error terms, the least squares estimator can be shown to coincide with the maximum likelihood estimator.

Confidence Intervals for the Least Squares Estimator

When estimating the response by means of the least square estimator, we speak of 'fitted' values and these are denoted by

$$\hat{y} = X\hat{\beta}.$$

As an estimator for the error ε, we get the so-called residual, which is defined to be the difference between the observed (actual) response vector and the fitted vector:

$$r = y - \hat{y}.$$

Note that other than ε, which is an unobservable error vector, r is observable and can be used to estimate the error variance σ^2 by means of the residual (or error) sum of squares

$$SSE = \sum_{i=1}^{n} r_i^2 = r^T r = (y - X\hat{\beta})^T (y - X\hat{\beta}) = \cdots = y^T y - \hat{\beta}^T X^T y.$$

For a model with $p = k + 1$ parameters (including the intercept term) to be estimated, the error sum of squares has $n - (k + 1)$ degrees of freedom and has an expectation of

$$E[SSE] = (n - (k + 1))\sigma^2.$$

An unbiased estimator for σ^2 is thus given by the mean squared error, which is the average error sum of squares obtained by dividing the error sum of squares by its degrees of freedom,

$$\hat{\sigma}^2 = MSE = \frac{SSE}{n - (k + 1)}.$$

Now, the covariance matrix of $\hat{\beta}$ can be estimated by replacing σ with its estimator $\hat{\sigma}$, thus allowing an estimate of the variance of the least squares estimators as $Var[\hat{\beta}_j] = (X^T X)^{-1}_{jj} \hat{\sigma}^2$.
Finally, this gives the following $(1 - \alpha)$-confidence intervals for the regression coefficients:

$$\beta_j \in \left[\hat{\beta}_j \pm t_{n-(k+1),1-\alpha/2} \cdot \sqrt{(X^T X)^{-1}_{jj} \hat{\sigma}^2}\right].$$

The constant $t_{n-(k+1),1-\alpha/2}$ hereby denotes the $(1-\alpha/2)$-quantile of Student's t-distribution with $n - (k + 1)$ degrees of freedom.

Confidence Intervals for the Fitted Values

From the distribution of $\hat{\beta}$ we can also obtain the distribution for the fitted variables. This leads to the following $(1-\alpha)$-confidence interval:

$$y_i \in \left[\hat{y}_i \pm t_{n-(k+1),1-\alpha/2} \cdot \sqrt{x_i^T (X^T X)^{-1} x_i \hat{\sigma}^2}\right],$$

where x_i denotes the vector $(1, x_{i1}, \ldots, x_{ik})^T$, that is the ith column vector of X.

Prediction Intervals for a new Observation

Whereas the point prediction for a new observation is simply given by

$$\hat{y}_{new} = \hat{\beta} x_{new},$$

where x_{new} denotes the vector of regressor variables for the new observation, a little more caution has to be put towards the development of the respective prediction intervals.

A prediction interval has to take into account the error from the fitted model as well as the error associated with the future observation. Therefore, a $(1-\alpha)$-prediction interval (or confidence interval for a new observation) is given by

$$y_{new} \in \left[\hat{y}_{new} \pm t_{n-(k+1),1-\alpha/2} \cdot \sqrt{(1 + x_{new}^T (X^T X)^{-1} x_{new}) \hat{\sigma}^2}\right].$$

2.1.2 Analysis of Variance

A famous result in statistics says that the total variation in a data set can be partitioned into two parts, the regression (corresponding to a regression model) and the residual (corresponding to the error thereof) fractions respectively:

$$\underbrace{\sum_{i=1}^{n} (y_i - \bar{y})^2}_{SST} = \underbrace{\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2}_{SSR} + \underbrace{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}_{SSE},$$

where SST denotes the total sum of squares, the squared deviation of each observation from the mean, SSR denotes the part of the deviation that can be explained by the regression model, and SSE denotes the part of the deviation associated with the residuals.

It needs to be pointed out that the above sum of squares will usually increase with the number of observations, as they are unscaled measures. For this reason, mean squares are often employed, which are simply the sum of squares divided by their respective degrees of freedom. The following table gives a depiction thereof:

Source	**Degrees of Freedom**	**Sum of Squares**	**Mean Squares**
Regression	$n-(k+1)$	SSR	$M_{SSR} = SSR/(n-(k+1))$
Residual	k	SSE	$M_{SSE} = SSE/k$
Total	$n-1$	SST	$M_{SST} = SST/(n-1)$

A second important result states that the error sum of squares divided by the true error variance follows a χ^2 distribution with the same amount of degrees of freedom as the respective sum of squares:

$$\frac{SSE}{\sigma^2} \sim \chi^2_{n-(k+1)}.$$

This now leads to a test statistic crucial to the model building process. Consider the following two models:

$$\text{Model 1:} \quad y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \varepsilon_i$$

$$\text{Model 2:} \quad y_i = \beta_0 + \sum_{j=1}^{l} \beta_j x_{ij} + \varepsilon_i, \quad \text{for} \quad l < k$$

It is of great interest to the statistician, whether (any of) the additional regressor variables in model 1 are significant, that is whether model 2 is a good enough description of the process or not. More precisely, this can be formulated in the words of statistical hypothesis testing as follows:

$$\underbrace{H_0 : \ \beta_{k+1} = \ldots = \beta_l = 0}_{\text{0-hypothesis}} \quad \text{vs.} \quad \underbrace{H_1 : \text{at least one of } \beta_{k+1}, \ldots, \beta_l \text{ is non-zero}}_{\text{alternative hypothesis}}.$$

Now, let SSE_1 denote the error sum of squares for model 1 and SSE_2 the error sum of squares for model 2 respectively. Since model 1 has more parameters than model 2 and thus $SSR_1 \geq SSR_2$, it holds that $SSE_1 \leq SSE_2$ (since $SST_1 = SST_2$). We can thus rewrite SSE_2 as follows: $SSE_2 = SSE_1 + (SSE_2 - SSE_1)$, where the second additive term is considered the excess sum of squares.

Now, if the null-hypothesis holds, the following are true:

$$\frac{SSE_1}{\sigma^2} \sim \chi^2_{n-(k+1)}, \quad \frac{SSE_2}{\sigma^2} \sim \chi^2_{n-(l+1)} \quad \text{and} \quad \frac{SSE_2 - SSE_1}{\sigma^2} \sim \chi^2_{k-l}.$$

Furthermore, under H_0 it holds that SSE_1 and $SSE_2 - SSE_1$ are independent of each other.

Putting these together, we get the following F-statistic to test the hypotheses at hand. If H_0 holds, then

$$F = \frac{(SSE_2 - SSE_1)/(k-l)}{SSE_1/(n-(k+1))} \sim F_{k-l,n-(k+1)}.$$

SSE_2 large implies that at least one of the parameters $\beta_{l+1}, \ldots, \beta_k$ is non-zero and thus significant. Therefore, one can reject H_0 for large values of F. The general rule here is to reject H_0 if $F > F_{k-l,n-(k+1);\alpha}$ for a chosen significance level α.

The `summary` command in `R` provides an F-value f and its corresponding realized significance $p = P(F > f)$, where $F \sim F_{k-l,n-(k+1)}$, as the last line of the output. This F-value represents the result of the above F-test for

$$H_0 : \ \beta_1 = \ldots = \beta_k = 0 \quad \text{vs.} \quad H_1 : \ \text{at least one parameter is non-zero.}$$

This means that if the null-hypothesis can be rejected, at least one of the regressors in the model has a significant influence on the response.

Note that this case corresponds to $l = 0$ in model 2 and the above F-test can be rewritten as a test for model 1 solely, which allows us to drop the index:

$$F = \frac{SSR/k}{SSE/(n-(k+1))} = \frac{M_{SSR}}{M_{SSE}} \sim F_{k,n-(k+1)}.$$

Whereas a large F-value now indicates that at least one of the regressors in the model are significant, it does not state how many of the independent variables are indeed necessary to explain the response satisfactorily.

It is also possible to test for the significance of a single influential variable x_j. Here, the hypotheses to be tested are

$$H_0 : \ \beta_j = 0 \quad \text{vs.} \quad H_1 : \ \beta_j \neq 0.$$

If H_0 cannot be rejected, the variable x_j can be removed from the model without greatly diminishing the explanatory power of the model. If the null-hypothesis holds, the test statistic in use follows a $t_{n-(k+1)}$-distribution,

$$t = \frac{\hat{\beta}_j}{\sqrt{\hat{\sigma}^2 (X^T X)^{-1}_{jj}}} \sim t_{n-(k+1)}.$$

If the test statistic is comparatively large, that is $|t| > t_{n-(k+1);\alpha/2}$, the null-hypothesis can be rejected and it can be concluded that regressor x_j is significant.

It has to be pointed out, though, that the t-test for one variable is not as meaningful as the F-test for a whole set of variables, as the regression coefficient $\hat{\beta}_j$ depends on the other regressors considered in the model.

Again, for each of the parameters considered in a linear model, these t-values and their respective significances p are given in the output of `R`'s `summary`-command.

2.2 Model Selection

There are several methods available that will help the practitioner to choose an adequate model, especially if there are many possibilities available. Particularly when the number of influential variables is unclear, a stepwise selection might prove helpful. There are three different main approaches to such a selection process:

- **Forward Selection:** Here, the initial model consists of the constant intercept term only and other variables are included one by one if they prove statistically significant. That is, the most significant variable will be added first, others thereafter if they deliver a significant improvement to the new current model.

- **Backward Elimination:** Here, the initial model consists of all candidate variables considered test-worthy and variables are deleted one by one in case they turn out to be statistically insignificant. Again, the most insignificant term is removed from the model first.
- The most common stepwise regression method is a combination of the above and can either drop or add variables in each step.

Of course, there needs to be a means of assessing the quality of the models produced in each of the steps. These can vary from one algorithm to another, however, the following provides an overview of the most common criteria.

2.2.1 Model Evaluation

These are several possible criteria for the evaluation of the quality of one or more models in consideration:

- **R^2-value** (uniquely defined for linear models only):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y}_i)^2} = 1 - \frac{\text{SSE}}{\text{SST}} = \frac{\text{SSR}}{\text{SST}} = \frac{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y}_i)^2},$$

  where $\hat{Y}$ denotes the fitted value, $\overline{Y}$ the average, SST the total sum of squares, SSE the residual (error) sum of squares and SSR the regression sum of squares.

  It takes on values between 0 (no fit) and 1 (perfect fit) and corresponds to the square of Pearson's correlation coefficient in the case of simple regression and to square of the multiple correlation coefficient in the case of multiple regression.

  This statistic shows the quality of the linear approximation, not however whether the model was specified correctly. Furthermore, it does not make an assertion about the statistical significance of the determined relationship. For this purpose, a special test needs to be conducted additionally. Last but not least, the R^2-value does not show whether or not a transformation of the data might give a better result, i.e. a more meaningful regression model.

  Since the R^2-value of a model with additional regression variables is always greater than that for the initial model, no matter if the additional variables are in fact able to explain the regressor more precisely, an adjusted R^2-value that takes into account the number of regression variables in the model is usually applied to compare several models with each other. This value is given as

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p}.$$

  In `R`, the `summary` function for a linear model (`lm`) gives, among others, both the R^2 and the adjusted R^2 value.

- **AIC** (Akaike's Information Criterion): This criterion can be used in the model selection process to compare models with each other, does however not give any information about the absolute goodness of fit of either of them. It is formally defined as

  $$AIC = 2 \cdot p - 2 \cdot \log(L),$$

  where p denotes the number of model parameters as usual, and $\log(L)$ is the model's maximized log-likelihood function.

  As the aim is to find the model with minimum AIC, the first term in the definition is a penalty term to avoid over-fitting, i.e. including model parameters to increase the goodness of fit irrespective of the actual relationship between the regression variables and the regressor.

  For small (finite) data sets a correction is usually applied which uses a stronger penalty for additional parameters. This is given as

  $$AIC_c = AIC + \frac{2 \cdot p(p+1)}{n-p-1}.$$

  Many statisticians advise to use the AIC_c instead of the AIC regardless the situation, unless n is many times larger than p^2, to avoid over-fitting.

- **BIC** (Bayesian Information Criterion), also called Schwarz Criterion: The BIC is partially based on the likelihood function as it is sought to increase the likelihood of the estimation by adding parameters. As this may lead to over-fitting of the data, a penalty term concerning the numbers of parameters in the model is included.

  It is defined as $BIC = p \cdot \log(n) - 2 \cdot \log(L)$.

  This criterion is closely related to the AIC but penalizes the use of too many regression variables more vigorously.

- **Mallow's C_p-value:** This criterion is commonly used as a stopping criterion in various stepwise regression procedures as it intends to guard against over-fitting. Mallow's C_p estimates the mean squared prediction error and is usually defined as

  $$C_p = \frac{SSE_p}{S^2} - n + 2 \cdot p,$$

  where p is the number of regressors (chosen from a possible set of $k > p$ variables) and SSE_p denotes the error sum of squares for the model with p regressors. S^2 stands for the residual mean square after a regression with the full k-parameter model and can be estimated by the mean square error. Note that here the notation for p has changed and $p \neq k + 1$.

  For the selection process, it is suggested that one should choose from a list of subsets ordered by an increasing number of regressors a set of p regression variables such that the corresponding C_p-value approaches p from above.

Since each of these criteria addresses only a single measure and takes it into account for the optimization process, one could use either one of them for a pre-selection and then assess

several resulting models to more detail using the model assessment techniques discussed in the following section. In this way, it can be ensured that the final choice does not suffer from some serious structural inadequacy or otherwise violates the model assumptions.

2.3 Checking the Modeling Assumptions

In the linear regression model several basic assumptions are made. It is crucial that these assumptions are tested for and checked before drawing any conclusions from the resulting model. Only if these assumptions are met, the model can be correctly interpreted.

The following four assumptions are made in the modeling process and need to be thoroughly investigated:

- The relationship between the response and the regressor variables is **linear**.
- The errors are **independent** of each other.
- The errors have a common **constant variance** (homoscedasticity).
- The errors are distributed **normally**.

In [47], the problems arising from ignoring any of the above points are addressed clearly as well as briefly:

"If any of these assumptions is violated [...], then the forecasts, confidence intervals, and economic insights yielded by a regression model may be (at best) inefficient or (at worst) seriously biased or misleading."

The most important tool for the model validation process is a graphical analysis of the residuals. This can provide insight into several different aspects of the model quality. However, there are also many different hypothesis-tests available to check specific properties of the model. The next sections are dedicated to these two methodologies respectively.

2.3.1 Graphical Assessment of Model Adequacy

The main focus of attention for the graphical analysis is hereby laid on the residuals of a linear regression fit. It is those residuals r_i that estimate the random error ε_i, which is supposedly identically and independently normally distributed with mean 0 and variance σ^2.

Often, the residuals are given in standardized or studentized form, which both have a variance normalizing effect. That is, both standardized and studentized residuals have the property that their variance is approximately 1. These transformed residuals are determined as

$$\hat{r}_i = \frac{r_i}{\hat{\sigma}} \text{ (standardized)}, \quad r_i^s = \frac{r_i}{\hat{\sigma}\sqrt{1-h_{ii}}} \text{ (studentized)},$$

where h_{ii} denotes the corresponding leverage or 'hat' value, that is the respective diagonal element of $H = X(X^TX)^{-1}X^T$ for the model matrix X.

The studentized residuals and leverage values also play an important rule in the detection of outliers. We shall later on use Cook's distance for such purposes. This difference is defined as

$$CD_i = \frac{(\hat{\beta}_i - \hat{\beta})^T (X^T X)(\hat{\beta}_i - \hat{\beta})}{p \cdot \hat{\sigma}^2} = \frac{r_i^2}{p} \left(\frac{h_{ii}}{(1 - h_{ii})^2} \right).$$

Here, $\hat{\beta}$ denotes the least squares estimator based on all n observations (as before) and $\hat{\beta}_i$ denotes the least squares estimator when observation i is removed from the total set of observations.

As a general rule of thumb, observations with a distance of greater than 1 should be examined closely under all circumstances and for observations with a distance greater 0.5 this action is suggested.

An extensive comparison of methods dedicated to that matter can be found in [1].

Next, we focus on checking the assumptions made concerning the distribution of ε and the relationship between the regressors and the response variable.

Checking the Linearity Assumption

In order to detect non-linearity in the relationship between the response and the regressors, one often-most looks at either the scatter plot depicting fitted against observed values or the scatter plot of residuals versus fitted values. The former should ideally show a straight diagonal line around which the points lie symmetrically, the latter should depict points being distributed randomly around the line $y = 0$.

If either of these plots does not look as expected, for example a distinctive curve visible in the residual plot, some kind of non-linear transformation (e.g. quadratic or logarithmic) to either the dependent or independent variables might be considered. For positive data a Box-Cox plot could be conducted indicating a suitable transformation. Alternatively, it could also prove useful to add a new regressor to the model that is a non-linear function in one of the original regressors. By all means, it should in such a case be abstained from using the original model for further considerations as the predictions are likely to be seriously wrong.

Checking the Independency Assumption

If this assumption is violated, then the sign of the residual r_i depends on the sign of the residual r_{i-1}. Such serial correlation is a serious issue in time series regression, however, it can also diminish the quality of a 'normal' regression model. Most often, small serial correlations in non-time-series regression indicate a poorly specified model with room for improvement.

The most common graphical means for detecting serial correlation in a time series data set are the lag plot and auto-correlation plot. For common linear regression this information is also derived from the residual plot (scatter plot of residuals versus fitted values).

Checking the Constant Variance Assumption

To detect non-constant variance of the error terms, again a residual plot proves most helpful. In case of a non-constant variance, the residual plot shows a distinctive structural component such as a cone.

Non-constant variance (heteroscedasticity) can be a by-product of some other assumption being violated. Consequently, it can likely be fixed by fixing the underlying problem.

Checking the Normality Assumption

If the normality assumption for the errors is not met, the residuals are most likely to take on a non-normal form. This can be detected by a normal probability plot, or so-called QQ-plot. This graphical aid plots the quantiles of the residual distribution against the theoretical quantiles of a normal distribution with the same mean and variance. The plot should show a sufficiently straight line without too much curvature.

An alternative (or additional) means of checking the normality assumption is the histogram. If the normality assumption holds, the histogram should take on roughly the shape of the Gaussian bell curve.

Violations to the normality assumptions could also be caused by a violation of the linearity assumption.

2.3.2 Non-graphical Assessment of Model Adequacy

There are many tests available to evaluate certain aspects of the model. These being special hypothesis tests, they do not give a general picture of the model quality but rather are limited to a single value derived from a test statistic. Whereas this might be useful when trying to confirm a conclusion drawn from the residual analysis, it is by no means an adequate substitute thereof. For this and other reasons, we shall not go into any more detail about these tests at this point, but we give a short list of tests available in various `R` packages in part two of this thesis.

We believe the following statement, taken from [47], makes an excellent final comment on the matter:

"Graphical methods have an advantage over numerical methods for model validation because they readily illustrate a broad range of complex aspects of the relationship between the model and the data."

Chapter 3

Experimental Design & Response Surface Methodology

Unless otherwise stated, this chapter is mainly based on books by Montgomery and Myers, [44], and Montgomery, [43], as well as on lecture notes by Stadlober, [58].

Since this research area is vast and still growing, we do not attempt to give a summary of these topics here but merely intend to focus on several aspects that are of particular interest to us.

3.1 Response Surface Methodology

This notion subsumes methods used for the statistical treatment of problems in which one or more response variables are to be explained in terms of influential variables and the aim is usually to maximize or minimize the response variable(s). However, beforehand, the true relationship between the response and influential variables is generally unknown and so the underlying situation is approximated by use of polynomials of low degree, usually linear or quadratic (in case of prevailing curvature).

Some of the most important tools of response surface methodology are the contour and response surface plots, which usually allow for a rather accurate graphical localization of factor settings that will result in an optimum response (given the currently investigated region by means of considered factors and their levels), be it minimizing or maximizing some effect or having it right on some target value. These plots prove especially useful if there are no more than three influential variables. When dealing with more variables, however, the interpretations of these plots become cumbersome and a more formal analysis is inevitable. This approach is called canonical analysis.

In addition to knowing the region of optimum response the practitioner should be interested in the nature of the system. It is invaluable to know not only where in the investigated region the stationary point is located but also whether it is a point of maximum or minimum response or a saddle point. To know the nature of the system is especially vital if it is impossible to operate at the detected point of optimum response, since then it is necessary to find an operable point that yields a response value not too far from the optimum, a

'compromise' point in a way. For this it is clearly essential to determine how sensitive the response system is to changes in the factor settings. This, too, can be investigated via a canonical analysis.

'The eventual objective of RSM is to determine the optimum operating conditions for the system or to determine a region of the factor space in which operating requirements are satisfied.' ([44])

Since response surface methodology (RSM) is usually applied for process optimizing purposes, it is a sequential procedure. A useful tool, which is usually employed at an early planning stage of the product development process in order to move the search near the vicinity of an optimum, is the so-called method of steepest ascent. It is a common practice to estimate a linear model (by means of, e.g., a (fractional) factorial design) in an early screening phase of the process in order to determine important influential variables and then employ more elaborate designs (e.g. central composite designs) in a main experiment so as to be able to accommodate for a quadratic model.

3.2 Analysis of Variance

It has to be understood that the data obtained from an experimental design contains some error due to the nature of fitting a model with the traditional method of least squares. There are two important assumptions made about this error term, denoted by ϵ, namely that it is normally distributed and independent of the influence variables. This provides that the resulting estimators for the model coefficients, $\hat{\beta}_j$, are unbiased.

It is of great interest to keep this error, and thus the variation of the response variable, in check. There are several measures that give some insight into the variance structure of the underlying system. Montgomery and Myers, [44], give methods to determine the standard error of the predicted response, confidence intervals for the location of the stationary points as well as confidence intervals for the eigenvalues of the matrix B used for the canonical analysis.

A useful tool for evaluating and comparing designs on the ground of their prediction variance, given by

$$Var[\hat{y}(x)] = \sigma^2 x^T (X^T X)^{-1} x,$$

are Variance Dispersion Graphs (VDG), which are also available in `R`. Taken directly from Montgomery, a 'VDG is a graph displaying the minimum, maximum, and average prediction variance for a specific design and response model versus the distance of the design point from the center of the region. The distance or radius usually varies from zero (the design center) to $\sqrt{k}$'. It is customary to also include the scaled prediction variance, which is the prediction variance multiplied by the number of runs and divided by the error variance, in a VDG.

The basic analysis of variance is habitually done via the `R`-command `aov`, however, several more elaborate methods are available in the `rsm` package. This package is specifically tailored to the purposes of response surface methodology and provides helpful functions

such as contour plots and methods determining the steepest ascent or canonical path as well as certain standard designs.

3.3 Classical Designs

Before going into more detail about the advantages and disadvantages of, as well as the general settings in which optimal designs, mostly computer-generated designs, are to be employed, we wish to review briefly the most commonly used standard designs and their most prominent characteristics. These designs are constructed to take on a certain (geometrical) form and as a natural consequence have special properties. It is a resulting regularity of these designs that sets them apart from optimal designs which will be discussed later.

The proper choice of experimental design depends crucially on the experimental region to be investigated and on the model used to fit the response surface.

Usually, the first step is to transform the natural data, which is given in some counting unit, into coded variables. Hereby, the coded values are chosen such that they are in the interval [-1,1]. Assuming that the region of exploration for a factor ξ is given by $[a, b]$, this transformation can be achieved by applying the following formula:

$$x = \frac{\xi - (a+b)/2}{(b-a)/2}.$$

3.3.1 Full Factorial Designs

One of the most important design families is that of the full factorial designs. Such designs contain all possible combinations of the factors' levels. If we consider an example with three factors A, B and C and assume that each of these factors should be tested for a, b and c different levels, then the resulting full factorial design will consist of $a \cdot b \cdot c$ experimental runs. It should be clear that too many different levels will make the design size impractical, for which reason 2^k or 3^k designs are most commonly used. Here, a 2^k denotes a design with k factors at two levels each and a 3^k follows the same principle. Note that 2^k or 3^k respectively refers to the resulting number of experimental runs of such designs. Again, for $k \geq 5$ factors this requires a number of experiments that might already be too large for many applications, even at only two levels.

3.3.2 Fractional Factorial Designs

In order to reduce the number or required experimental runs fractions of full factorial designs are often used. If all factors are set at two levels a 2^r fraction of a full 2^k factorial design is denoted by 2^{k-r}. The choice of the fraction needs to be considered carefully since a fractional design does not allow to estimate all factors and interactions separately. For a 2^{k-r} design there are r generators which determine which interactions are to be confounded, or aliased. Defining relations ('words', consisting of letters referring to factor interactions) allow to determine the complete alias structure of the design. The aliases are calculated

by multiplying an effect by the defining relation(s) 'modulo 2', meaning that even powers of a factor cancel out.

For example, for a 2^{3-1} and a single defining relation of $I = ABC$ the complete resulting alias structure is given by A = A(ABC) = BC, B = AC, C = AB . Note that in this case the main effects can be estimated but each are aliased with a two-factor interaction.

The measure of resolution gives some information as to how well main effects and interaction terms can be separated from one another. The resolution of a design, denoted by roman numerals, is the length of the shortest word among the defining relations. Generally, it is favorable to have designs of resolution IV or V because they allow to estimate main factors and even two-factor interactions, aliased by interactions of order three or higher.

The 2^k can estimate all k main effects as well as all interactions up to the k-factor interaction. However, an unreplicated 2^k design has no degrees of freedom left for the estimation of error.

Also, an unreplicated design with only two levels per factor is not capable of fitting quadratic or higher-order models in the factors (these models are still 'linear' models, since they are linear in the βs). In fact, $p+1$ levels are needed for factor X in order to estimate the model term X^p. As a remedial measure it is possible to include additional center runs into the two-level design, thus allowing to test for curvature (i.e. the significance of the regression coefficient corresponding to the model term X^2) in the response surface. As a positive side-effect, conducting several center runs also result in a lower prediction variance at the center.

3.3.3 Central Composite Designs

This design is a very useful tool in response surface methodology to fit a quadratic model. Usually, for spherical or cuboidal design regions this design should be among the practitioner's first choices.

Note that there are a total of $1 + 2k + k(k-1)/2 =: n$ parameters to be estimated for a full quadratic model of the form

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k} \sum_{j<i} \beta_{ij} x_i x_j + \epsilon.$$

This requires a design with at least n distinct design points and at least three levels on the factors to estimate pure quadratic error.

The CCD, which for several factors results in fewer runs than the alternative full 3^k design, consists of three main parts:

- A full 2^k or a fraction of resolution V for the factors at their high and low level, also called the factorial part of the design;
- $2k$ axial runs, one for each of the factors at levels α and $-\alpha$ (all other factors are taken at the center level, coded as 0);
- Several center runs, i.e. all factors are set to their center level.

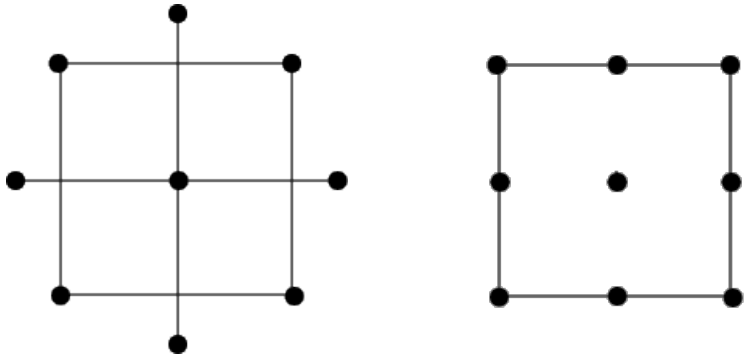

Figure 3.1: Comparing number of design runs for two factors and three levels each:
On the left the (spherical) CCD and on the right the full 3^2 factorial design (or cuboidal CCD).

This special structure makes the CCD well suited for the use in sequential experimentation. Each of the three parts contributes to the estimation of different effects: the factorial points contribute to the estimation of the linear and interaction terms, the axial points to the estimation of the quadratic terms and the center runs to the estimation of pure error.

Note that the number of center runs and the choice of the axial level α have significant influence on the resulting design and its properties. Usually, α is chosen around $\sqrt{k}$, resulting in a spherical design, or $\alpha = 1$, in which case the design becomes cuboidal (as shown in graphic 3.1). This choice should be based on the form of the region of interest.

Note that both designs depicted in figure 3.1 have the same number of runs which is a special case for only two factors, since the cuboidal CCD and the 3^2 coincide. Already for $k = 3$ the CCD needs considerably less experiments, namely only $8 + 6 + 1 = 15$ as compared to the 27 of a full 3^3 factorial design. Even though it is often preferable to have not only one but three or five center runs, the CCD is still a very economical design.

Including replications in or of a design allows the separation of the experimental error into its two components lack-of-fit error, which indicates that one or more important terms or factors are missing from the model, and pure (or random) error. While replication of any design point, and hence also of the center point, allows to estimate pure error, in order to estimate the lack-of-fit error the whole design needs to be replicated.

3.4 Desirable Design Properties

The proper choice of an experimental design is the key to fitting and analyzing a response surface. Whereas several properties of a design may generally be considered desirable, it is unrealistic to expect that there exists a single design that is capable of facilitating all of them, especially since often they are conflicting.

Montgomery and Myers give the following list consisting of 10 desirable design properties:

- A design should result in a good fit of the model to the data.
- A design should give sufficient information to allow a test for lack of fit.
- It should allow models of increasing order to be constructed sequentially.

- A design should provide an estimate of 'pure' experimental error.
- A design should be insensitive (robust) to the presence of outliers in the data.
- A design should be robust to errors in control of design levels.
- A design should be cost-effective.
- A design should allow for experiments to be done in blocks.
- A design should provide a check on the homogeneous variance assumption.
- A design should provide a good distribution of $Var[\hat{y}(x)]/\sigma^2$.

Which of these properties are to be given priority in the selection process is highly dependent of the investigated application.

For first-order designs it is definitely desirable to have *orthogonality*. An orthogonal design is one for which (X^TX) is a diagonal matrix. The 2^k designs are orthogonal designs and since the covariance, which is in fact equal to the variance matrix is given by $\sigma^2(X^TX)^{-1}$, this implies that each of the main effects and the interactions are independent of each other.

Orthogonal designs that have all the factor levels set to ± 1 (in coded notation) minimize $Var(\hat{\beta}_i)$, the scaled variance of the regression coefficients. Two-level factorial designs and fractions thereof that have a resolution of at least III pose as an example for the pure first-order case, where no interactions of the factors are considered. If two-factor interactions are taken into account as well, a fraction of at least resolution V must be employed.

Note that while the addition of center runs does not harm the orthogonality property of a design, the resulting design is no longer variance-minimal due to the factors not being set at the ± 1 levels solely.

A desirable property for a design to fit a second-order response model is *rotatability*. This property states that the prediction variance $n \cdot Var[\hat{y}(x)]/\sigma^2$, where n denotes the number of design points, does not depend on the position of x in the design region directly, but only on its distance from the design center. The prediction variance is thus constant on spheres. Clearly, it is sought that the prediction variance be nearly stable throughout the whole experimental region. Rotatability (or near-rotatability) can often be achieved quite easily without having to compromise other important design properties. Montgomery and Myers give a characterization of rotatability for both first and second-order models by means of design moments.

Note that special spherical CCDs and any first-order orthogonal design are rotatable designs.

One of the desirable properties of a design is that it is conductible sequentially in order to include higher-order terms at a later date if necessary. However, if considerable time passes between the conduction of the two parts it is often impossible to ensure equivalent experimental conditions. Thus, *blocking* may become necessary. In such a case it is desirable to have a design block orthogonally, such that the block effect does not have any influence on the estimation of the response surface model, that is block effects are orthogonal to model coefficients.

For a first-order model, a full 2^k or fractional 2^{k-r} design can be arranged to block orthogonally. If there are additional center runs these are to be distributed equally among

the blocks. For a second-order model, there are two conditions that must be satisfied in order for the design to block orthogonally: each block must itself be a first-order orthogonal design and 'the fraction of the total sum of squares for each variable contributed by every block must be equal to the fraction of the total observations that occur in the block'. A CCD can always be constructed to block orthogonally in two blocks, with one block consisting of the factorial points plus a fraction of the center runs and the other block containing the axial points plus the other fraction of the center runs.

For the analysis of blocked designs, the model has to be augmented additively by the block effects and it is assumed that no interaction between blocks and other model terms are significant. If this assumption may likely be violated, it is best to fit separate response surfaces for each block, which, however, brings about some difficulties in interpreting the results.

After discussing many properties that are desirable in a 'good' design, it should not go unmentioned that there are also properties that are to be avoided if at all possible. One example of the latter are so-called *saturated* designs. These are designs with equal amounts of design points and model terms. Thus no degree of freedom is available for the estimation of lack of fit and therefore it cannot be tested whether the proposed model is indeed adequate.

There are certain, so-called 'small', designs available that are saturated or near-saturated. These can be applied if the cost factor is important and therefore it is absolutely impossible to conduct more experiments.

3.5 Optimal Designs

The areas of application for optimal designs, which are mainly computer-generated designs, are plenty, but mainly include all situations in which standard-designs are not the best choice, brought about by reasons such as:

- An irregular experimental region due to constraints on the design variables.
- Non-standard regression models such as an incomplete quadratic model for example.
- The occurrence of (several) qualitative variables among the design variables.
- Special requirements regarding the number of conducted experiments due to economical restrictions.

Note that this list is by no means exhaustive but merely states the settings for which the possible application of optimal designs for some optimality criterion immediately comes to mind of the experienced practitioner.

The typical approach when applying computer-generated designs is as follows:

1. Specification of a model,
2. Determination of the experimental region,
3. Specification of number of experiments to be conducted,
4. Selection of an optimality criterion,
5. Selection of experimental points from a candidate list by means of an algorithm.

3.5.1 Alphabetic Optimality Criteria

The following is a short introduction of the most important and most frequently applied optimality criteria, also referred to as alphabetic optimality criteria due to their nomination. Most of these criteria fall into one of two categories, namely aiming to minimize either the variance of the response factor or the regression variables. Note that these traits are closely related to the so called moment matrix, which is given by

$$M = \frac{X^T X}{n},$$

where n denotes the number of experiments conducted. The inverse of the moment matrix is directly related to the variances and covariances of the regression coefficients which are given by

$$Cov(\beta) = \sigma^2 (X^T X)^{-1}.$$

It shall be pointed out at this point that the unreflected use of optimal designs is not encouraged since the resulting designs by definition take into account only one important aspect and fail to meet a variety of the properties considered desirable in a good design. However, it is obvious that some situations make the application of computer-generated designs inevitable, the most common of which have been pointed out previously in this paper.

- **D-Optimality**: The aim is to find a design that minimizes $|(X^T X)^{-1}|$ or maximizes $|X^T X|$ respectively, where X denotes the design matrix. This criterion is equivalent to minimizing the volume of the joint confidence region of the regression coefficient vector. This, however, does usually not imply that the resulting design manages to minimize the variance of the response variable.

  In order to compare designs $Des1$ and $Des2$ with corresponding design matrices X_1 and X_2 with each other, *D-efficiency* is defined as

  $$D_{eff} = \left(\frac{|(X_1^T X_1)^{-1}|}{|(X_2^T X_2)^{-1}|} \right)^{1/p},$$

  where p denotes the number of model parameters. This measure gives the relative efficiency of $Des1$ when compared to $Des2$. Assuming a resulting $D_{eff} = t$, this implies that $Des1$ needs to be replicated $1/t$ times in order to achieve the same precision on the estimation of the regression coefficients as $Des2$. It is especially useful to choose a D-optimal design as a reference, i.e. as $Des2$. The aim then is to find a design which minimizes D_{eff}. Due to the scaling, the power of $1/p$ indicating the use of p model parameters, the concept of D-efficiency even allows for comparison of designs with different sample sizes.

  In the following discussion, we shall restrict our attention to the use of this optimality criterion solely.

- **A-Optimality**: The aim with this optimality criterion is the minimization of the sum of the variances of the regression coefficients, which corresponds to the minimization of $tr((X^T X)^{-1})$.

This criterion does not take into account any covariances among the coefficients, and like D-optimality is unable to consider the response (or prediction) variance.

- **G-Optimality**: The aim of this criterion is the minimization of the maximum scaled variance of the response variable over the whole experimental region. This means that it is sought to minimize $\max v(\mathbf{x})$, where

$$v(\mathbf{x}) = \frac{n \cdot Var\left[\hat{y}(\mathbf{x})\right]}{\sigma^2} = n \cdot \mathbf{x}^{(m)T}(X^T X)^{-1}\mathbf{x}^{(m)},$$

  where $\mathbf{x}^{(m)}$ not only denotes the position in the design space but also the order of the model.

  It is obvious from the notation that unlike the criteria mentioned before, G-optimality depends on the position in the design space $\mathbf{x}$.

  Note that there exists a method to create designs which attempt to stabilize $v(\mathbf{x})$. The inclusion of center runs usually results in a lowered response variance especially around the center of the design region.

  We have mentioned the notion of design efficiency before. As an alternative to the D-efficiency defined above one can also employ G-efficiency defined as

$$G_{eff} = \frac{p}{\max_{\mathbf{x}\in\mathbb{R}} v(\mathbf{x})},$$

  where p denotes the number of model parameters (including the intercept).

  When dealing with exact rather than with approximate designs, one has to keep in mind that the G-efficiency can be computed precisely whereas for the D-efficiency only lower bounds can be given. This will be of great importance to us in the practical part of this thesis.

- **Q-Optimality**: As mentioned before, prediction-based criteria have the disadvantage that they are depending on the position in the design space, therefore Q-optimality attempts to overcome this drawback by averaging the response variance over some region of interest, thus resulting in a single-value criterion much the same as D- or A-optimality. This is achieved by integration over the region of interest and scaling by the volume of this region which accounts for the alternative denotation of IV-optimality, namely integrated variance.

- **V-Optimality**: The aim with this optimality criterion is the minimization of the average variance of the response variable given a predetermined set of points as experimental region.

D-optimality enjoys great popularity, not least due to one significant advantage over, for example, A-optimality: the computational effort to update the determinant is much lower than to update the trace of a matrix. This is particularly vital for the implementation of an algorithm intended to find such an optimal design as most of the approaches include a stepwise exchange of design points and thus can make good use of an algebraic formula that allows for not having to calculate the complete determinant anew in each iteration.

3.6 Algorithms for D-optimal Designs

Note that in most cases the algorithmic determination of the actual optimal design would involve an exhaustive search over all possible designs in the experimental domain. Since this is computationally impossible, we now introduce some algorithms to find approximate D-optimal designs in a more efficient way.

Most such algorithms are so-called **exchange algorithms**. De Aguiar et al., [9], nicely summarize the basic idea behind such exchanges. Hereby, in each iteration of the algorithm one or more rows from the design model matrix X_n (n rows) are chosen and replaced by an equal number of candidates. There are some differences concerning how to determine a start matrix and whether the exchanges are executed sequentially or simultaneously. This process is then repeated until some level of satisfaction is reached, for example in the case of D-optimality such a criterion might be that there is no exchange possible that will result in a higher determinant of the information matrix.

Note that this is the very definition of a local exchange algorithm and the returned optimum might well be a local one. In general, with these methods there is no guarantee to find global optima and there are no known criteria for a local optimum to be globally optimal.

Also, keep in mind that all of these algorithms are tightly bound to the model one wants to find a good design for, as it is represented in the design matrix.

3.6.1 Mitchell's Algorithm

This algorithm is also known as the DETMAX algorithm and was published in 1974, [42]. Although published later, it is considered the first algorithm to find an approximate D-optimal design.

The exchange procedure for this algorithm is based on the variance function of each point in the experimental domain. In each step, the procedure selects a candidate point that has the largest variance function value since this point differs most from those already selected and adds it to the design matrix. The point with the smallest variance function value is in turn removed from the design matrix as it contributes the least information. We shall now give some more technical details, following [9].

Let X_{old} be a (design) matrix with n rows. Then X_{old} augmented by an additional row (or candidate point) x_j yields a new (design) matrix X_{new} with $n+1$ rows. The information matrices corresponding to X_{new} and X_{old} are linked as follows:

$$X_{new}^T X_{new} = X_{old}^T X_{old} + x_j \cdot x_j^T.$$

For the determinants thereof, it holds that

$$|X_{new}^T X_{new}| = |X_{old}^T X_{old}| \cdot (1 + d(x_j^T)),$$

where

$$d(x_j) = x_j^T (X_{old}^T X_{old})^{-1} x_j$$

is the variance function for the candidate point x_j. For the algorithm, x_j is chosen among all candidate points such that it yields the maximum increase in the determinant of the information matrix.

Now, let X'_{old} be the (design) matrix with $n+1$ rows resulting from the foregoing augmentation. Then again, deleting one experiment x_i from X'_{old} yields a new (design) matrix X'_{new} with n rows. The corresponding information matrices then are linked as follows:

$$X'^T_{new}X'_{new} = X'^T_{old}X'_{old} - x_j \cdot x_j^T.$$

For the determinants thereof, it holds that

$$|X'^T_{new}X'_{new}| = |X'^T_{old}X'_{old}| \cdot (1 - d(x_i^T)).$$

Here, of course, x_i is chosen such that its removal from the design matrix yields the minimum decrease in the determinant of the information matrix.

Mitchell's algorithm permits temporary deviations from the n-run design matrix, where a few adding or removal steps may be skipped, in order to increase the chance of finding better optima.

3.6.2 Fedorov Exchange Algorithm

Published in 1972, [12], this was one of the first algorithms for the given problem and has been modified and extended upon numerous times by now.

The main idea of this algorithm is to not consider the adding and removing of the points sequentially but instead do the exchange in one single step. If we replace experiment x_i from the design matrix by a candidate x_j, the relation between the original design matrix X_{old} and the updated design matrix X_{new} is as follows:

$$X^T_{new}X_{new} = X^T_{old}X_{old} - x_i \cdot x_i^T + x_j \cdot x_j^T.$$

Furthermore, the determinants of the two respective information matrices are linked by

$$|X^T_{new}X_{new}| = |X^T_{old}X_{old}| \cdot (1 + \Delta(x_i, x_j)),$$

where

$$\Delta(x_i, x_j) = d(x_j) - d(x_i) - \left[d(x_i)d(x_j) - (d(x_i, x_j)^2\right]$$

with

$$\begin{aligned} d(x_i) &= x_i^T(X^T_{old}X_{old})^{-1}x_i \\ d(x_j) &= x_j^T(X^T_{old}X_{old})^{-1}x_j \\ d(x_i, x_j) &= x_i^T(X^T_{old}X_{old})^{-1}x_j = x_j^T(X^T_{old}X_{old})^{-1}x_i \end{aligned} .$$

In each step this algorithm requires the calculations of $d(x_j)$ for all points x_j currently in the design matrix and $d(x_i$ for all x_i in the candidate set as well as $\Delta(x_i, x_j)$ for all possible couples (x_i, x_j). The couple leading to the maximum Δ-value is then chosen for the exchange. Theoretically, the algorithm stops as soon as there is no couple with a positive Δ to be found. Practically, reaching this stopping criterion might take a long time and a small threshold ϵ is usually employed so as to terminate the algorithm as soon as $\Delta < \epsilon$.

In 1980, Cook and Nachtsheim, [8], developed a modification of this basic version which breaks down each iteration of Fedorov's algorithm into n stages (one for each point in the current design matrix). At each stage one point in the design matrix is fixed, all possible Δ-values containing this point are computed and the best exchange is executed if it yields an increase in the determinant of the information matrix.

3.6.3 k-Exchange Algorithm

This algorithm, first introduced by Johnson and Nachtsheim in 1983, [24], can be seen as another modification to Fedorov's exchange algorithm.

Since it turned out that the points selected for removal from the design matrix by Fedorov's algorithm are usually not those with the lowest variance, the idea for this adaptation is to select the k points with the lowest variance for the exchange procedure. Again, for each of these points an iteration is conducted in which all corresponding Δ-values are computed and the best exchange is executed if it results in an improved determinant.

Note that this algorithm for $k = 1$ is similar to the basic version of Mitchell's algorithm (also called the Wynn-Mitchell algorithm) and for $k = n$ resembles the aforementioned modified exchange algorithm by Cook and Nachtsheim.

The choice for k is usually problem-dependent but the authors suggested $k = 3$ or $k = 4$ as a general rule-of-thumb.

3.6.4 kl-Exchange Algorithm

Despite its name this algorithm, which was published by Atkinson and Donev in 1989, [2], is a modification to the basic Fedorov algorithm rather than the k-exchange algorithm.

The algorithm attempts to reduce the number of exchange candidates by not computing the Δ-values for all possible couples but rather by choosing the k points with the lowest variance currently in the design matrix and the l points with the highest variance among the candidates. For each of the resulting couples the Δ-values are then computed and only a single best exchange is conducted.

Of course, there are many implementations and variants to be found for these basic algorithms. Much attention has also been put towards computational aspects such as reducing the size of the candidate set (e.g. [40]) and general fine-tuning as well as adding to the functionality.

Note that so far we have not dealt with the problem of finding a feasible initial design matrix. For this purpose, simple algorithms developed by Wynn in 1970, [73], or Dykstra in 1971, [10], can be employed. Both algorithms build a feasible design from scratch by sequentially adding those points with the highest variance until the desired design size is reached. These algorithms usually yield poor results but nonetheless accomplish to find feasible start designs.

A good summary of the exact algorithms described in this section can be found in [61].

3.6.5 Haselgruber's DDoE-Algorithm

This fairly new dynamic method, which has been published in [21] (2007) and [22] (2008), is based on a modular algorithm which allows for the replacement of 'failed' experiments and the preliminary fixing or partly (component-wise) fixing of experiments. Furthermore, the algorithm is also apt to incorporate already conducted experiments into the final design and the range of a factor can be modified 'on the go', hence the D for 'dynamic' in the algorithm's name. In addition, this method grants the possibility of a design extension by additional experiment runs or variables, making it a very powerful and versatile tool.

For details on parameters and other specifics please refer to the original publication of this algorithm.

The algorithm is partitioned into the following four modules, which usually (but not mandatorily) are applied sequentially and iteratively.

- **Construction of a Candidate List:** First, one has to specify a mesh density ρ_i for each of the factors x_i, that is to say the number of operating levels to be used. Then a full factorial design containing all $\prod_{i=1}^{m} \rho_i$ possible combinations of factor settings is constructed, each row representing one experiment. Next, all points violating some constraint are removed from the design. If there are requirements about the inclusion of points with partly fixed factor settings then for each of these points all combinations of the variable factors are added to the design. The same is to be done for points with completely fixed factor settings.

  After setting up the so-called 'alternative group vector' and 'status vector', the last step of this module is extending the candidate set according to the model such that each model term is represented by a column.

- **Construction of an Initial Design Matrix:** The aim of this module is to find an initial design with a non-singular moment matrix $M = X^T X$. For this purpose, the model matrix X is being built from scratch by first including all mandatory experiments and one randomly chosen candidate from each alternative group, as well as a certain amount of randomly chosen exchangeable points. Unless the corresponding moment matrix M is regular, the set of randomly chosen exchangeable experiments is replaced by a new (possibly larger) set. Finally, one needs to find the candidate point x_k which maximizes $x_k^T M^{-1} x_k$ and then add it to model matrix. This is repeated until X comprises exactly n experiments.

- **Determination of a D-optimal Design:** This module uses single point exchange steps to arrive at a hopefully close to D-optimal design. The main instrument here is the exchange delta function given by

  $$\Delta(x_i, x_o) = x_i^T u_i - x_o^T u_o + (x_i^T u_o)^2 - x_i^T u_i x_o^T u_o,$$

  where x_o denotes the point to be replaced by x_i and $u = M^{-1}x$ for each of the indices i and o respectively. The objective in each iteration is to find points x_o currently in the design and x_i currently in the candidate list that maximize this exchange delta function. This can be achieved by calculating an information factor $w_i = x_i^T M^{-1} x_i$

for every exchangeable experiment. The whole iterative procedure terminates as soon as $\Delta(x_i, x_o) < \epsilon$ for some small predetermined constant $\epsilon > 0$.

As a final remark on the previous modules it should be pointed out that it is generally favorable to repeat the initialization and optimization of the design several times in order to avoid poor results caused by the random selection of the initial design points.

- **Extension/adaptation of the Design:** With this module it is possible to extend an existing design in a way that ensures that the resulting design is D-optimal with respect to the original design and the candidate list or to replace an experiment, for example because it failed. Candidate points that are either appropriate or inappropriate as an extension or replacement are indicated. The rest of this module is similar to the previous one.

The most significant advantage of the modular approach is that the algorithm can be started with each of the modules.

Next, we are going to introduce a different kind of algorithms to find D-optimal designs, namely **approximate algorithms**. A good summary of these algorithms can be found in [75]. The big difference to the exact exchange algorithms mentioned before is that the upcoming algorithms put a probability measure on all possible support points (i.e. the set of candidates) and then iteratively update these weights until some convergence criterion is met. These algorithms themselves differ in the choice of a starting measure and the update routine. For this kind of algorithms, the general equivalence theorem by Kiefer and Wolfowitz, [27], can be used to obtain an optimality criterion.

Let w denote the probability measure, i.e. $w_i \geq 0$, $\sum_{i=1}^{N} w_i = 1$, on the set of experiments $\{x_1, \ldots, x_N\}$ in the (finite) design space. Then Kiefer, [26], called w an *approximate design*.

Assume that a model with m terms is to be considered. Then $M(w) = \sum_{i=1}^{N} w_i x_i x_i^T$ is a matrix proportional to the information matrix of the parameter vector. We further define $d(i, j, w) = x_i^T M^{-1}(w) x_j$ and $d(i, w) = d(i, i, w)$.

The general equivalence theorem then states that w^* is a D-optimal design if and only if

$$\frac{1}{m} \max_{1 \leq i \leq N} d(i, w) = 1.$$

3.6.6 Multiplicative Algorithm

This algorithm was published by Silvey et al. in 1978, [57].

It chooses strictly positive values for the start design and follows an update rule of

$$w_i^{(k+1)} = w_i^{(k)} \frac{1}{m} d(i, w^{(k)}) \quad \text{for } i = 1, \ldots, N.$$

The algorithm generally converges rather slowly as it focuses priorly on theoretical properties. There, however, exist various modifications and improvements on this algorithm.

3.6.7 Vertex Direction Method

This algorithm by Fedorov which dates back to 1972, [12], is designed to update w in the direction in which the directional derivative of $\log(|M(w)|)$, corresponding to $d(i, w)$, is the greatest. In this sense the algorithm is a steepest ascent method.

The vertex direction method searches an index $i_{max} \in \{1, \ldots, N\}$ such that $d(i_{max}, w) = \max_{1 \le i \le N} d(i, w)$.

In each iteration the design is then updated according to the following formula:

$$w_i^{(k+1)} = \begin{cases} (1-\delta) w_i^{(k)} & \text{for } i \neq i_{max}, \\ (1-\delta) w_i^{(k)} + \delta, & \text{for } i = i_{max}, \end{cases} \quad \text{where } \delta \in [0, 1].$$

Note that δ is chosen such that $|M(w^{(k+1)}|$ is maximized.

3.6.8 Vertex Exchange Method

This algorithm, which was published by Böhning in 1986, [4], is closely related to the vertex direction method.

The basis of this algorithm is formed by the following general exchange step for any two support points x_a, x_b with $a \neq b$:

$$w_i^{(k+1)} = \begin{cases} w_i^{(k)} & \text{for } i \notin \{a, b\}, \\ w_i^{(k)} - \delta & \text{for } i = a, \\ w_i^{(k)} + \delta & \text{for } i = b, \end{cases} \quad \text{where } \delta \in [-w_a, w_b].$$

δ is again chosen such that $|M(w^{(k+1)}|$ is maximized.

The vertex exchange method chooses an initial design $w^{(0)}$ such that $|M(w^{(0)}| > 0$. For the update two indices i_{min} and i_{max} have to be determined such that

$$d(i_{min}, w^{(k)}) = \min_{1 \le i \le N} \left\{ d(i, w^{(k)}) : \; w_i^{(k)} > 0 \right\}, \quad d(i_{max}, w^{(k)}) = \max_{1 \le i \le N} d(i, w^{(k)}).$$

These two support points are then used as points for the exchange.

3.6.9 Yu's Cocktail Algorithm

This fairly new algorithm was published by Yu in 2011, [75], and is a combination of the well-known multiplicative algorithm, the vertex direction method and a nearest neighbor exchange strategy. This algorithm can be shown to globally converge linearly and its main advantage is that it numerically performs much better than each of its components taken separately.

One big problem concerning the convergence speed of the multiplicative algorithm is that it may take many iterations until the weight of points adjacent to support points in the actually optimal design is reduced sufficiently. The cocktail algorithm thus uses a nearest

neighbor exchange in order to avoid this problem. Hereby a natural ordering of the points is crucial. For each support point in the current design, i.e. x_i such that $w_i^{(k)} > 0$, the support point with a higher index that minimizes the absolute distance to that point is chosen for a vertex exchange step. The nearest neighbor method can help keep the number of support points small as it never readmits points to this set.

For the cocktail algorithm, an initial design $w^{(0)}$ is chosen such that $|M(w^{(0)})| > 0$. The update rule then consists of first performing an iteration of the vertex direction method, followed by a nearest neighbor step and finally an iteration of the multiplicative algorithm. Numerically there seems to be no reason why the vertex direction method is to be preferred over the vertex exchange method in the first part of the update procedure but this choice is crucial to the proof of the theoretical convergence result.

This algorithm's implementation in `R` is freely available on the developer's homepage.

Let us add a remark about the discretization of approximate designs. It is generally desired to have a design consisting of n runs rather than a probability distribution over the whole set of support points. Pukelsheim and Rieder, [50], introduced a method called efficient rounding to round approximate designs into exact design for a given sample size. They showed that this method has the smallest loss of efficiency for the D-optimality criterion (among others).

An alternative approach to finding optimal designs are so-called **genetic algorithms**. These shall not be considered at this point but the interested reader is referred to, for example, [53].

3.7 Issues Concerning the Analysis of Unbalanced Designs

This section deals with a practical problem to keep in mind as a practitioner as well as a statistician. In many cases difficulties with the analysis of the data can be avoided when all involved parties are aware of the respective issues from the beginning.

Wynn observed in [74] that the replication of a design 'is useful as it supplies a direct measure of error variance'. However, unless the original design was an indeed optimum design, one is unavoidably going to lose efficiency in the replication process. Moreover, if the design is unevenly replicated, meaning that not all design points are replicated the same amount of times, this will further add to the loss of efficiency.

Of course, such uneven replications resulting in an unbalanced design do not necessarily have to stem from desire but rather from a premature failure of an experiment. Data that was obtained without the use of an experimental design is also almost habitually unbalanced. However, there are certain problems arising from said unbalancedness. One such drawback is that factors may no longer be independent of each other, even if they set out to be in the unreplicated design. If this is the case then the order in which the terms (or factors) are entered into the respective model does indeed make a difference in

the analysis of variance as the sums of squares corresponding to the factors may vary. It is thus suggested to change the order of the factors and repeat the analysis several times to see how much the results for each factor vary, [28].

Note that while unbalancedness does not necessarily cause unorthogonality of the respective designs, not every design has to be orthogonal either. Unorthogonal designs, where the factors are not independent of each other, have another undesirable property, namely that the composition of total sum of squares into regression and residual parts does not work anymore because of overlapping sums of squares for the regression factors.

Another problem arising from unequal cell sizes (where a cell is a combination of settings) in the design is that they might 'give rise to heterogeneity of variance across cells, and make problems for valid standard error estimates', [19].

Note that when we are talking about the analysis of variance we are implicitly referring to type I sums of squares. For the analysis of unbalanced data, however, the use of type II or type III sums of squares is preferable, [34]. The reason this never comes up is that all three types of sums of squares coincide for balanced data.

3.7.1 Types of Sums of Squares

We are quickly going to state the different types of squares as well as some of their most prevailing properties, both positive and negative. This list is according to [7], which in turn summarizes the contents of [34].

Type I (Sequential) Sums of Squares

Here, the sum of squares for each factor states the decrease of the residual sum of squares when that factor is added to the model and all terms appearing before that factor are already contained in the model.

For this type of SS the usual decomposition of the total sum of squares into regression and residual sum of squares holds, no matter whether the data is balanced or not.

However, as already mentioned, the order in which the factors are added to the model matters for this type of SS. Furthermore, a small p-value corresponding to a factor does not necessarily mean that the factor is significant. Clearly, this property limits the applicability of this type.

Type II (Hierarchical or Partially Sequential) Sums of Squares

In a type II analysis each effect is adjusted for all other terms except model terms that contain the effect being tested, [34]. This type of SS can also be obtained by considering several type I analyses, the number of such analyses depending on the amount of effects to be tested.

Type II sums of squares are not usually applied because of one undesirable assumption, namely that interactions are negligible or don't even exist. However, this type of SS is invariant to the order in which the effects are entered into the model.

Another major drawback of this type of SS is the hypothesis they are actually testing, which usually are hard to interpret.

Furthermore, this approach is not appropriate for factorial designs.

Type III (Marginal or Orthogonal) Sums of Squares

For this type, the sum of squares for a factor is equal to the type I SS if the factor is entered last into the model, meaning that all other factors have already been accounted for before.

The main advantage of this type of SS lies in the use with unbalanced designs, namely that 'effect estimates are not a function of the frequency of observations in any group. When there are no missing cells in the design, these sub-population means are least squares means.'

A drawback of this approach is that it is not appropriate for designs with missing cells, which optimal designs are almost exclusively, because of the complex hypotheses tested in such cases.

In [34] the issue of which type of SS (II or III) is to be preferred. The author favors type II SS, whereas most statistical packages by default use type III SS.

Finally, this section should have pointed out that the analysis of unbalanced designs is not as straightforward as that of balanced designs and there is much controversy as to how such situations can be handled. A good summary of the possibilities can be found in either [62] or [19].

3.8 Final Remarks

If the levels of a factor are hard to change or to control exactly and the practitioner wishes to conduct all experiments with this factor set at a certain level consecutively rather than randomly, **split plot designs**, divided into whole plots and subplots, can be used. Two kinds of error have to be considered for such designs, often denoted as inner and outer errors.

'Robust parameter design', an attempt at choosing the levels of the influential factors such that the mean response is at a desired level while ensuring that the variability around this target is small, is often employed to design processes that are relatively sensitive to small variations in the factor settings, as these are often difficult to control precisely. Taguchi designs can be a very useful tool for such kinds of problems.

It is usually assumed that the response y is normally distributed, allowing a good parameter estimation via least squares. However, if the normality assumption is violated and a distribution with heavier tails is expected, *robust regression* methods should be applied in order to mitigate the effect of outliers.

If the underlying situation requires the simultaneous handling of multiple response factors,

there are several possible ways of doing so:

- The first possibility consists of the construction of separate models for each of the response factors. The resulting contour plots are then superposed in order to allow identification of promising points in the design space. Note that this method is really only applicable if there are no more than three influencing factors since the results are not interpretable otherwise. This basically excludes this method for our purposes.

- An alternative approach is to formulate the problem as a constrained optimization problem. For this purpose, one response factor is chosen as the main factor of interest whereas the other ones are merely bounded. This leads to the general notion of non-linear programming. There are several options to facilitate this approach in `R`, such as functions `constrOptim` and `donlp2` (from package `Rdonlp2`, [60]).

- Finally, one could use the approach via desirability functions. Here, every response factor y_i is coded into a desirability function d_i satisfying $0 \leq d_i \leq 1$, where $d_i = 1$ indicates that y_i already lies in the desired region, whereas $d_i = 0$ indicates that y_i is currently violating feasibility. The aim is to choose the influencing factors (variables) such that the overall desirability given by $D = (d_1 \cdot d_2 \cdot \ldots \cdot d_m)^{1/m}$, where m denotes the number of response factors, is maximized.

  There exists an entire `R`-package designated to this approach, conveniently entitled `desirability`, [30].

Part II

R-Functionality

Chapter 4

Linear Regression & Model Building

Linear Regression is a topic well covered in R. The standard version has readily incorporated most procedures that will ever be of interest to the practitioner concerning analysis of variance and residual analysis. However, there are many additional packages available and we will mention but a few of those as we shall have need for their contents. The main package to find additional or enhanced plots in as well as methods to represent or summarize the models is 'car', [14], short for 'Companion to Applied Regression', which indeed it is.

4.1 Core Functions

The functions described in this section are all available in R's core packages, [52]. They are well documented in R directly and in the online documentation.

Here, we are not so much interested in the various options but rather in the basic use. However, we shall give specifics if they are esteemed viable to follow the code given in the upcoming practical part.

`lm()` (linear models) and `glm()` (generalized linear models)

The two main functions for model fitting are `lm` and `glm`. Whereas `glm` can be seen as a generalization of `lm`, the latter is often sufficient and also computationally more desirable as is described in [18]:

> *'The `glm` function fits by default normally distributed data. If the data are normally distributed and the model has an identity link one can use the `lm` function which is numerically more efficient. For other link functions one has to use the `glm` function.'*

Both functions are provided with a formula, given as $y \sim x_1 + \ldots + x_p$. This of course represents the model

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i x_i.$$

The model does not necessarily need to be specified term by term (separated by a '+') but can instead be given using several additional operators. For example, '`a*b*c`' is a short way of writing '`a+b+c+a:b+a:c+b:c+a:b:c`', where the operator '`:`' is used to define interactions between the terms `a` and `b`. Note, however, that `lm` (and `glm`) will ignore '`a^2`' or the likes for a single term. This operator has a different meaning than one might intuitively expect. '`(a+b+c)^2`' is a short way of declaring '`a+b+c+a:b+a:c+b:c`'. If the square of a term is to be considered in the model, one instead has to write '`I(a^2)`', where `I()` symbolizes that the function inside is to be considered as a single additional term. Note that neither '`a*a`' nor '`a:a`' will yield the desired result here.

Of course, there are many options that could be specified for either function but we shall hardly need anything but the standard procedure in the following and shall therefore not give any more details on these functions here.

`aov()` (analysis of variance)

This function is actually a wrapper function to `lm`. However, it differs from that function in one key point, namely the way the `summary` function operates on the produced model objects.

`summary()`

The standard output that the aforementioned functions, and others, produce does not contain all the information one is usually looking for. Therefore, a nice wrapper function named `summary` is available, which, as the name might suggest, provides a more detailed summary of the given model objects.

For a model fitted with `lm` (or `glm`) the summary consists of the model call, a summary of the residuals (minimum, 1st quartile, median, 3rd quartile and maximum values) as well as a table containing detailed information about the estimated coefficients. This table includes the estimate, its standard error and the according t- and p-values. Furthermore, the residual standard error and the quality measures multiple R^2 and adjusted R^2 are included in the output of the summary, as are the F statistic and the according p-value.

The summary for a call to `glm` also contains other information, of which the alternative quality measure AIC is the most interesting for us.

For a model fitted with `aov` such a summary consists of a table containing each model term (including interactions and higher order terms) as well as the according degrees of freedom, sums of squares, mean squares and F- and p-value. Additionally, the degree of freedom, sum of squares and mean squares are also given for the residuals. Note that the model call and the residual standard error are included in the standard `aov` command, however not in its summary.

`resid()`, `coef()` and `fitted()`

All models fitted with either of the functions mentioned above contain, among others, a 'residuals', 'coefficients' and a 'fitted.values' attribute, which can be extracted using either `modelobject$resid` or (the full function call) `residuals(modelobject)`, which can be abbreviated to `resid(modelobject)`. Of course, the call for the coefficients and fitted values follows the same notational principle. However, note that for the coefficients the short calls are either `coef(modelobject)` or `modelobject$coeff`, mind the difference in spelling.

We mention these extractions because they are often needed for the graphical representation and quality assessment of a model.

`predict`

`predict` or the subroutine thereof `predict.lm` can be used to predict response values according to a linear model object. If the function is called on the data set which was used for the model fitting, then clearly the predicted values coincide with the fitted ones. However, `predict.lm` allows to define an alternative data set containing the same variables as in the model and their respective values which are then used for the prediction.

Furthermore, with this function it is possible to determine the prediction or confidence intervals of the predicted or fitted values by specifying the respective method in option `interval`. The default for this option is "none".

`step()`

`step` is a function that is very useful when attempting to find an adequate model based on a given set of candidate terms (terms that are expected to be significant), provided as a model formula. The function then sequentially removes the most insignificant terms according to the AIC criterion and stops iterating once no improvement on the AIC can be found.

It is also possible to provide both an upper and a lower bound for the search, each provided as model formulae. The procedure then both adds and removes terms until a model with a locally best AIC is found.

Furthermore, one can choose similar evaluation criteria such as the BIC as well by providing the according value k in the formula.

We will use this function in the modeling chapter to get a first idea of which terms (out of the complete quadratic model) might be important influences for the responses.

AIC and Variants Thereof

Whereas function `AIC` from the core package can handle both the normal AIC and the BIC value, which basically just differ by the choice of the penalty function, for the corrected AIC (AICc) the package 'AICcmodavg' needs to be installed. Function `AICc` in that package

can then calculate both the AIC and the AICc (for which option `second.ord` has to be set to TRUE).

4.2 Important Plots

Here, we shall only focus on plots specifically intended for the analysis of regression models. It should be clear that `R` contains a vast amount of different standard plots, which we shall not even mention in this chapter.

`plot()`

`R`'s all-round function `plot` can be used to generate scatter plots for many different applications. Of course, we could use this function in its most basic way, namely stating two vectors of our choice to plot against each other (and we will). However, for a model object fitted by one of the functions introduced before, this command can also give a graphical analysis of the residuals automatically. The output then consists of four subplots depicting the (standardized) residuals versus the leverage values, the residual values versus the fitted values, a normal QQ-plot of the residuals including the QQ-line and finally a scale-location plot. In these plots outliers are identified by their index (position) in the underlying data for the model fitting.

`qqnorm()` **and** `qqline()`

`qqnorm` plots the normal QQ-plot, or normal quantile-quantile plot, for a provided vector. Such a plot is a graphical means to compare two probability distributions with each other. In our case the first distribution, which serves as a reference, is fixed to be the the normal distribution against which the residuals of a model object are tested. The name stems from the fact that the quantiles of the distributions are compared with each other.

`qqline` is used to add a line which passes through the first and third quartiles to the QQ-plot. If the residuals are indeed normally distributed, all plotted values should be very close to that line.

Note that the more intuitive command `qqplot` also exists but is not the right one to use in our situation as it produces a QQ-plot of two data sets.

4.2.1 Package 'car'

The 'car' package contains a function `Boxplot` which is a wrapper function for the standard `boxplot` but also allows to modify the look. The same holds for `qqPlot` and its core analogon `qqnorm`.

With this package it is also possible to quickly identify outliers graphically by the use of a function called `infIndexPlot`. This so-called influence index plot provides four subplots depicting Cook's distance, the Studentized residuals, the corresponding Bonferroni p-values

for an outlier test and finally the leverage values ('hat' values) for each observation of a regression.

Alternatively, one could use the influence plot produced by `influencePlot`. This function creates a 'bubble' plot of the Studentized residuals versus the 'hat' values. The areas of the circles hereby represent Cook's distance.

With the 'car' package it is also possible to easily add a regression line to scatter plots by means of the function `regLine`.

4.3 Transformation of Data

Generally, the assumed linearity assumption is not likely to hold for any arbitrary set of response and regressor variables. Therefore, a transformation of some of the independent or dependent variable(s) is often needed. Some sort of power transformation, that is replacing a variable by a certain power of itself, does often suffice.

In `R`, the function `boxcox` can be used to find a suitable power for such a transformation of the response variable.

The Box-Cox Transformation of the Response

The Box-Cox transformation is formally defined as

$$z_i = y_i^{(\lambda)} = \begin{cases} \frac{y_i^{\lambda}-1}{\lambda}, \text{if } \lambda \neq 0 \\ \log(y_i), \text{if } \lambda = 0 \end{cases}$$

The `boxcox`-function computes and optionally plots profile log-likelihoods for the parameter λ of the Box-Cox power transformation. The plot also includes a 95% confidence interval for the theoretical value of λ. For example, if 0 is to lie within this confidence interval, a log-transformation might prove very useful. Note that for $\lambda = 1$ the resulting transformation is in fact a shift of the data by one.

A slight generalization thereof is to be found in the 'car' package, named `boxCox`. This function allows for other families of transformations to be used as well. Each of the possibilities can be called directly via the following functions as well: `basocPower` can perform simple power transformations, `bcPower` the standard Box-Cox transformation and `yjPower` the so-called Yeo-Johnson transformation to a given data vector.

The function `boxTidwell` from package 'car' performs the Box-Tidwell power transformations of the predictors in a linear model.

The function `symbox` from package 'car' can also be used to select a suitable power for a power transformation. It first transforms the input vector to each of a series of selected powers in such a way that each transformation is standardized to have a mean of 0 and standard deviation of 1. The function also plots the resulting data in box plots which allows

for a comparison among the tested transformations. The goal is to find a transformation that results in a symmetric distribution.

4.4 Hypothesis Testing

Tests for the normality assumption of data (not necessarily limited to regression residuals) are available in `R`'s base installation. However, for univariate data (such as the residuals of a linear model) there seems to be one to favor:

- **Shapiro-Wilk-Test** (`shapiro.test`): Normality test for univariate data.

For more specific hypothesis tests there are several packages available in `R`, a few of which we are going to name in the following.

'lmtest' and **'fRegression'** are both dedicated to hypothesis tests for common linear as well as time series regression models. The latter being a wrapper package for the former it aims at increasing the usability. These packages contain a great amount of tests, including tests for linearity (Harvey-Collier and Rainbow) and tests for or against heteroscedasticity (Breusch-Pagan, Goldfeldt-Quandt and Harrison-McCabe) as well as a test for the functional form of the regression (Ramsey's RESET).

Additionally, several tests can be found in the package **'car'**, maybe the most useful of which are the Bonferroni outlier test and the score test for non-constant error variance.

As mentioned in a previous chapter, the use of these tests should be considered carefully. It cannot replace a thorough residual analysis as a means of model quality assessment.

Chapter 5

Experimental Design & Response Surface Methodology

5.1 Core Functions

Different to linear regression (or regression generally), R's core packages are not really tailored for the analysis of data created by an experimental design.

However, many packages have been developed lately as can be checked on the CRAN task view concerning the topic 'Design of Experiments (DoE) & Analysis of Experimental Data', [15].

There are also several tutorials available on how to use R for the analysis of experiments. [67], [33] (an unfinished R-companion to [43]) and [32] provide good insight into the matter. Note that the last of these is not dedicated to experimental design alone but rather comprises an introduction to R in general which includes useful chapters on the topic at hand.

Next, we shall quickly state a few core functions that can prove very helpful when analyzing designs. Possibly the most important functions therefore are once again `lm` and `aov`, which have already been covered in the previous chapter.

`alias()`

This function allows the user to find aliases (or linearly dependent terms) in a specified linear model object fitted by `lm` or `aov`. The output of this function can be very messy with highly unorthogonal data. Also, the alias structure depends on the contrasts in use.

`replications()`

The function `replications` prints a table of counts for each factor and combination of factors according to a specified model for a given data frame. In a way, this function is a collection of calls to `table`, which also gives counts for one or several specified vectors,

in the latter case including the counts of the pairwise combinations. However, `table` can also handle non-factors.

`contrasts()` and `C()`

These function are used to view and set contrasts associated with a factor. `R` can handle several different contrasts, each of which can be set as an attribute to a factor using function either function `contrasts` or `C` with the respective option. The possibilities therefore are (taken from the documentation):

- `contr.helmert`: contrasts which contrast the second level with the first, the third with the average of the first two, and so on.
- `contr.poly`: contrasts based on orthogonal polynomials.
- `contr.sum`: 'sum to zero contrasts'.
- `contr.treatment`: contrasts which contrast each level with the baseline level (to be specified) and the baseline level is omitted.

  Note that this does not produce 'contrasts' as defined in the standard theory for linear models as they are not orthogonal to the intercept.
- `contr.SAS`: a wrapper for `contr.treatment` that sets the base level to be the last level of the factor.

However, it is also possible to reset the default option for contrasts which is then applied to all factors subsequently, which is probably easier in many cases. To do so, one simply defines the desired contrasts in the options as follows:

```
options(contrasts=c("contr.treatment", "contr.poly"))
```

Note that the contrasts stated are the default settings, so this code can be used to restore the default once it has been tempered with.

`make.contrasts` from package 'gregmisc', [68], also allows to construct a contrast matrix that is user-specified. Such a matrix can then be used as an argument to the methods mentioned before.

Furthermore, there exists an entire package dedicated to simplify and standardize the use of contrasts for several linear model producing functions, namely the 'contrast' package, [31].

When searching for contrasts on the internet, it seems as if sum-to-zero contrasts are often considered the 'best' contrast type. However, their advantage is only inherent for completely balanced designs for which they imply orthogonality of the columns of the design matrix. For unbalanced designs they are not so useful, because they often simply yield wrong results.

There is a lot of literature covering the topic of contrasts in R to be found, especially in lecture notes. In [64] the use of the different contrast types is explained to more detail. Another good source for additional information (although not R-specific) is [6], which gives some mathematical background as well.

Testing for Orthogonality of Contrasts

Taken from the R-help, [65], function `crossprod` can be used to find out whether a given model matrix, which can be created using `model.matrix`, is orthogonal or not. The necessary condition for the contrasts to be orthogonal is that all between-term off-diagonal elements of the output from this function are zero.

Note that the results are again depending on the contrasts used as well as the coding of the factors.

5.1.1 Package 'rsm'

This package allows to generate response-surface designs, fit first- and second- order response-surface models, make surface plots, obtain the path of steepest ascent and do canonical analysis, [36].

Coding and Decoding

The package contains several functions that allow the quick and easy coding and decoding of data, such as `coded.data`, `decode.data` or `val2code` and `code2val`. These functions can be used to code data quite freely, allowing to provide an own formula for the coding.

`rsm()`

`rsm` is a function intended to 'fit a linear model with a response-surface component and produce appropriate analyses and summaries'. Response surface components are specified using the functions `FO` (first order), `TWI` (two-way interactions), `PQ` (pure quadratic) and `SO` (second order).

The output of this function is a special linear model object, including additional values such as the order of the model and the first- and second-order response-surface coefficients.

When calling the `summary` command for an object created by `rsm`, the following additional parameters are given: the path of steepest ascent for first-order models or a canonical analysis for second-order models, an ANOVA table including a test for lack-of-fit and the coding formulas.

5.2 Useful Plots

`interaction.plot()`

This plot is used to show interactions between factors by graphically depicting the means (or other function) of a given response (plotted on the y-axis) for the respective two-way combinations of two given factors. One factor is hereby given on the x-axis, the other one is called a 'trace' factor and the plot includes one curve for each value of this factor.

Note that this function plots the levels of the factor on the x-axis in an equidistant manner, which does not necessarily have to be in accordance with the real intervals between values.

Furthermore, combinations of the factors that are not available are omitted, resulting in gaps between the lines which are normally drawn to join the levels of the factor on the x-axis so as to better separate the values of the trace factor visually. Such missing values can make the plot somewhat confusing, especially when there are only few combinations present and the plot therefore does not contain any lines.

5.2.1 Package 'graphics'

The following plots are all available in the 'graphics' package, [51], and are very useful tools for the graphical investigation of experimental designs or response surface models.

`plot.design()`

This function shows the effect of each factor in the design by showing the difference in means for each level of that factor. The factors are hereby plotted on the x-axis (next to each other) and the response is plotted on the y-axis. For each factor a vertical line is given, connecting the respective means for the factor's levels. The length of this line gives an estimate of how much influence this factor has on the response (when compared to the other factors).

`plot.factor()`

This function is an extension for factors to the generic `plot` function. It takes one or two input vectors and produces different types of plots according to the class of those vectors.

For only one input vector, a bar plot is produced. Otherwise, if the second vector is numeric the output consists of a boxplot and if the second vector is again a factor then a spine plot (a stacked bar plot) is given.

`contour()`

This function can either be used to create a contour plot or to add contour lines to an existing plot. A contour plot is a two-dimensional plot giving two factors on the axes and contour lines to illustrate the area of the plot in which the response is equal, thus clearly showing minima, maxima and ridges.

This function can be combined with another, called `image`, which simply creates a colored grid across the plot area with different colors indicating different values of the response. It is a common practice to add contour lines to such an image plot.

Alternatively, one can also use the function `filled.contours`, which basically does the same as `contour` and `image` combined, it adds color to the contour plot. Different values of the response are hereby indicated by different colors in addition to the contour lines and a legend that maps the colors to these values is also included.

`persp()`

This function allows to create perspective plots, which are three-dimensional plots showing a response surface above the two-dimensional factor plane.

Although very helpful in applications with only two factors, such plots are not always usable.

5.2.2 Package 'lattice'

Among many other useful plots, the 'lattice' package, [55], also contains analogons to the plots from the 'graphics' package we just introduced.

`contourplot` clearly produces contour plots and `levelplot` can be seen as the counterpart to `image`.

Note that the way the factors and the response have to be given to the function differ from the use in the respective 'graphics' plots.

5.2.3 Package 'rsm'

There exists an own vignette, which is intended as a companion to the 'rsm' vignette, for plots that can be found in the 'rsm' package, [37].

Although the general plots available in this package are the same as in the 'graphics' package, namely a contour- and a perspective plot, they are enhanced and allow for more functionality.

5.2.4 Package 'Vdgraph'

As the name might suggest, this package is dedicated to the creation of variance dispersion graphs. The package mainly consists of a port of such graphs from FORTRAN to `R`, [35].

The main function of interest is called `Vdgraph`. The single input parameter to this function is a matrix containing the design in coded or uncoded variables, each column of which is to contain a factor and each row a design run.

A variance dispersion graph, as explained in [44], is a plot depicting the scaled prediction variance versus the distance from the origin (design center) on the axes and including three curves, namely the minimum, average and maximum prediction variance as a function of

the radius. Furthermore, a horizontal line at the number of parameters in the design is included as a measure of optimality.

In the Minitab documentation on the topic, [41], the field of application of a variance dispersion graph is described as follows:

'These graphs show the scaled prediction variance of a DoE across the design space and are typically used to compare response surface designs. However, variance dispersion graphs can also be used to compare the performance of multiple designs for a specific model, such as a linear model, a linear model with interaction terms, a linear model with quadratic terms, or a full quadratic model.'

5.3 Algorithms and Packages for D-Optimal Designs

Most commonly used commercial statistical software packages (Minitab, SPSS, Design-Expert, Statistica, Matlab, ...) contain routines that generate optimal designs. The fundamental algorithms used in these packages are mainly based on local point exchange steps, which is well suited for the incorporation of for example (partly) fixed design points, i.e. points that have to be included in the final design.

It should be pointed out that most implementations use approximations, as explained in the first part of this thesis, to reduce calculation time. In most cases, the final design consisting of n not necessarily distinct design points with the highest weights is selected by means of rounding, thus resulting in a discrete design fulfilling the requirement of n runs to be conducted.

One needs to be aware of the fact that due to the pure nature of the point exchange approach there is no guarantee that the output design is an optimal design since the result might lie in the vicinity of a local optimum rather than a global one. Unless one is willing to perform an exhaustive search over the entire design space, which is in most cases not only very unpractical but merely impossible due to calculation time, such a guarantee cannot be given but most implementations perform rather well, especially when the search is repeated several times with different starting designs.

In the following, we shall focus on the available `R`-variants for creating D-optimal designs, namely the package `AlgDesign` maintained by Wheeler, [69], and an extension thereof by Grömping, named `DoE.wrapper`, [16].

5.3.1 Package 'AlgDesign'

This package is fully incorporated into the `R`-repository and contains numerous procedures, among which are two different algorithms capable of creating exact and approximate D- and A-optimal designs. Furthermore, the package contains functions for the evaluation of these designs, both blocked and unblocked. Also, a procedure to generate a model matrix from a given model formula is provided. The two main functions of interest, however, are `optFederov` and `optMonteCarlo` which shall be explained to some more detail here.

`optFederov` is based on Fedorov's exchange algorithm, one of the most famous algorithms for creating optimal designs. This routine can be used to create approximate or exact optimal designs according to one of three optimality criteria (D, A or I, which is the same as the Q-criterion described above) from scratch or to augment an existing design. The procedure can be sped up by choosing to use only a fraction of the design points in the search of an exchange candidate.

`optMonteCarlo` is also based on Fedorov's exchange algorithm but as the key difference to the aforementioned `optFederov` uses only a random subset of the points in the candidate list. As a consequence, it is possible to handle huge designs with this algorithm since the candidate list does not need to be stored physically as a whole. One further significant advantage over `optFederov` is the ability to incorporate constraints on the variables directly.

5.3.2 Package 'DoE.wrapper'

This package has recently been published, allowing to create standard designs for industrial applications as well as to apply the algorithms from `AlgDesign` in a more convenient way. Unfortunately, this package is under heavy construction and some functionality may not yet be available.

The functions of interest to us are `Dopt.design`, which uses `optFederov` but adds flexibility to the routine, and `Dopt.augment`.

We dedicate a whole chapter in part three to the investigation of functionality of the algorithms contained in these two packages.

Part III

Practical Applications

Chapter 6

Description of the Experiment

Before every experiment, it is vital to clarify several aspects for both the practitioner (experimenter) and the statistician. These include:

- Model Specification
- Experimental Region
- Response Variable(s)
- Expectations Concerning the Results of the Experiment
- Determination of Phase of Experiment (Screening, ...)

The screening phase is, among other things, supposed to determine possible limits of the settings of influence factors. In our case the screening was done by a literature search. All of these experiments consisted of varying only one influence factor at a time. One of the main interests of this experiment is thus the investigation of possible interactions between these factors.

The general goal of the experiment is to determine the optimal settings of the key exterior influence parameters on the two target values *cell capacity decrease* and *resistance increase*. These multiple responses of interest did not influence the design or the overall planning of the experiment.

The experiment at hand was planned by the practitioners at the ViF research company together with the statistician Nikolaus Haselgruber (AVL, CIS). It was decided to consider four influential variables (to be explained shortly) and to conduct the experiment according to a computer-generated D-optimal design. The model in use was to be the full quadratic model with interactions in those four variables. Each test point, i.e. run in the design matrix, was to be repeated at least *three times* so as to obtain statistically valid results. Unfortunately, as table 6.2 shows, this was not achieved in the end.

The Factors Considered in the Experiment

The four factors considered in the experiment are the operating temperature (`Temp`), operating current (`Curr`), state of charge (`SoC`) and delta state of charge (`dSoC`) of the cells in use. The following table gives an overview of the factors' level settings in both natural and in coded values. The table also contains the factor-entities.

	Levels	Entity	Natural Variables	Coded Variables
`Temp`	5	$^\circ$ C	$-10, 10, 30, 50, 70$	$-1, -0.5, 0, 0.5, 1$
`SoC`	5	mAh	$10, 20, 50, 80, 90$	$-1, -0.75, 0, 0.75, 1$
`dSoC`	5	%	$0, 10, 40, 70, 80$	$-1, -0.75, 0, 0.75, 1$
`Curr`	3	rel. to Cap.	$0, 1, 5$	$-1, -0.6, 1$

For the current we use the generally accepted definition of a c-rate, which is the current corresponding to the cell capacity. The measurement of the c-rate is done by a 0.2C discharge after constant voltage charging at $30°C$.

The Imposed Factor Restrictions

Since not all combinations of settings are technically achievable, the following restrictions on the experimental region, given for the coded variables, have been known beforehand:

$$\begin{aligned}
\texttt{dSoC} &\leq 2 \cdot \texttt{SoC} + 1 \\
\texttt{dSoC} &\leq -2 \cdot \texttt{SoC} + 1 \\
\texttt{dSoC} &\leq 5 \cdot \texttt{Curr} + 4 \\
\texttt{dSoC} &\geq 0.125 \cdot \texttt{Curr} - 0.875 \\
\texttt{Temp} &\geq \tfrac{25}{64} \cdot \texttt{Curr} - \tfrac{49}{64}
\end{aligned}$$

It soon became clear that an additional constraint would have needed to be considered as these settings too were impossible to achieve:

$$\texttt{dSoC} \leq -\tfrac{32}{15} \cdot \texttt{Curr} + \tfrac{46}{3}$$

The Underlying Experimental Design

Table 6.1 shows the 32-run (including repetitions) D-optimal design according to which the data was obtained. This design was created using Haselgruber's DDoE-algorithm and considered the *first five constraints* and *four factors* introduced before.

The Cells in Use

The practitioners used two types of cell chemistry, namely LFP (featuring lithium-iron-phosphate cathodes and high power graphite anodes) and NCA (featuring nickel-aluminum-cobalt-oxide cathodes and high energy graphite anodes), throughout the experiment. For

	Temp	SoC	dSoC	Curr
1	0.0	-0.75	-0.75	1.0
2	0.0	0.75	-0.75	1.0
3	1.0	0.00	1.00	1.0
4	-0.5	0.00	1.00	-0.6
5	1.0	1.00	-1.00	-1.0
6	1.0	-1.00	-1.00	-1.0
7	0.0	0.00	-1.00	-1.0
8	0.0	0.00	1.00	1.0
9	-1.0	1.00	-1.00	-1.0
10	-0.5	0.00	1.00	-0.6
11	1.0	0.00	1.00	-0.6
12	-1.0	-1.00	-1.00	-1.0
13	1.0	0.00	1.00	1.0
14	0.0	-0.75	-0.75	1.0
15	1.0	0.75	-0.75	1.0
16	1.0	-1.00	-1.00	-1.0
17	-0.5	0.00	1.00	-0.6
18	0.0	1.00	-1.00	-1.0
19	0.0	0.00	-1.00	-1.0
20	0.0	0.75	-0.75	1.0
21	-1.0	1.00	-1.00	-1.0
22	1.0	-0.75	-0.75	1.0
23	1.0	0.00	1.00	-0.6
24	1.0	1.00	-1.00	-1.0
25	1.0	-0.75	-0.75	1.0
26	0.0	0.00	1.00	1.0
27	-1.0	0.00	-1.00	-1.0
28	1.0	0.75	-0.75	1.0
29	0.0	-1.00	-1.00	-1.0
30	-1.0	-1.00	-1.00	-1.0
31	1.0	0.00	1.00	-0.6
32	1.0	0.00	-1.00	-1.0
	4	5	3	3

Table 6.1: The D-optimal design that forms the foundation of the data acquisition for the experiment. The last row indicates the number of levels per factor in the design.

reasons of energy consumption and security of the experiment a small cell size was selected for the cells. The cells were, pouch foil covered and of around 55mAh capacity (nominal 50 mAh for LFP and 60 mAh for NCA), made up from three square electrode sheets. During the whole experiment the cells were pressurized by clamps sitting on the outside of the pouch foil, in analogy to the straps commonly used in automotive applications, to hold the electrodes in place firmly. The cells were especially designed and built for this experiment, using base electrode materials from Südchemie AG and cell building know-how from GAIA Akkumulatorenwerke GmbH.

The Reference Test Procedure

The reference test procedure was designed to be a measurement sequence which allows for a calculation of *capacity decrease*, *resistance increase* and their respective estimation errors as well as additional analysis data sets, such as differential capacity and differential voltage. It consists of an initial segment in which the cell is preconditioned to a full SoC. Then a slow discharge is made for measuring actual capacity. This capacity is used as base for the following c-rate scan with 0.5, 1, 2, and 5 C charge and discharge each including constant voltage full charges. The third section is a galvanostatic intermittent titration test which has 5 pulses and 4 rest periods at 0.5C in discharge and in charge direction. This section is the base for calculation of the actual resistance, and eventually pulse impedance, of the cell. In the last part of the reference test procedure a full charge and a controlled 50% discharge set the cell to be ready to start a new load cycle. The initialization of the process is done by hand as well as the load cycling restart, because the new capacity has to be manually decreased in the control program of the cycling equipment.

The Sub-Experiments per Chemistry

For several reasons these two cell chemistries need to be regarded separately for an analysis. For example, it was known from the beginning that it is impossible to age NCA cells at extreme temperatures or at a high current. Furthermore, the factors under consideration are expected to influence the two cell chemistries differently and it is therefore preferable to view the chemistries completely separated, instead of introducing a categorical variable.

Since the cell-to-run appointment seems to have been done without a certain plan, the resulting sub-experiments for the two cell chemistries do no longer follow the initial design and it is therefore to be expected that each sub-design has a by far worse efficiency than the original design did. Table 6.2 shows the number of cells that were run for each combination of factor settings (in the following referred to as 'DoE-points'). There, the columns labeled 'LFP' and 'NCA' respectively state how many cells of the corresponding chemistry were aged at the indicated conditions. One can see that not for all DoE-points cells from both chemistries were conducted (at all or) at an even number. Note that NCA cells could not be run at a current of 5C as intended, but were instead aged at 2C thus resulting merely in a different name of the corresponding DoE-points. Also, LFP cells that were designed to be aged at $-15°$ centigrade were instead aged at $-10°$ centigrade.

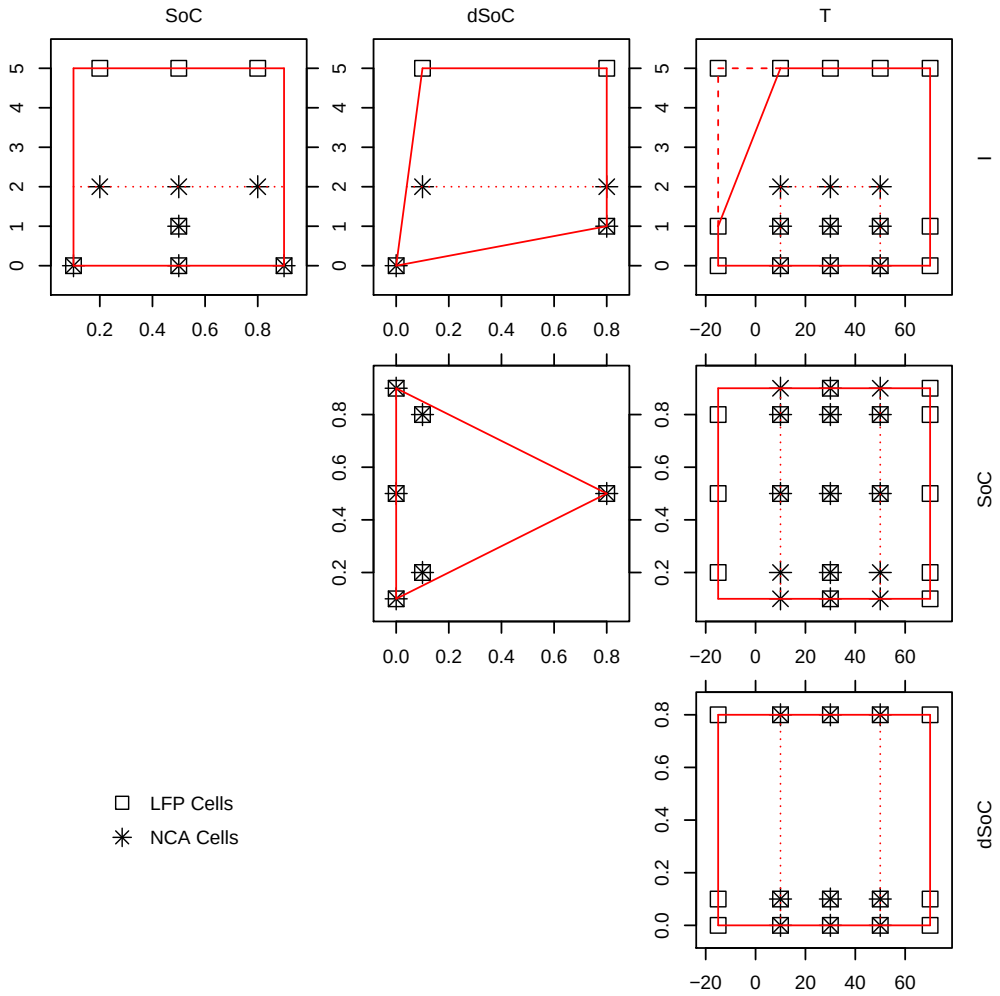

Figure 6.1: Plot visualizing the intended experiments, divided into the two cell chemistries.

Plot 6.1 summarizes the used DoE-points for each chemistry and and also shows the constraints on the factor settings.

The Obtained Data Set

The data set we are going to be dealing with contains measurement data of 175 cells in total, divided into 98 LFP and 77 NCA cells. Each of these cells was measured periodically over time until it was broken by test definition. All in all there were 1597 measurements. In the next chapter we shall focus on defining one-dimensional (time-free) response variables.

Given the situation, that is difficulties to receive exact measurements and non-constant profiles over time, it was clear from the beginning that the intended factor settings might not be realistically achievable in all cases. The practitioner was aware of the fact that the measurement procedure has an influence on the settings of `Curr` as well as `SoC` and `dSoC`.

	Levels				Counts	
	5	4	5	3		
Number	**Temp**	**Curr**	**SoC**	**dSoC**	**LFP**	**NCA**
1	-15	0	0.5	0.0	5	–
2	-15	1	0.5	0.8	4	–
3	-15	5	0.2	0.1	4	–
4	-15	5	0.5	0.8	4	–
5	-15	5	0.8	0.1	4	–
6	10	0	0.1	0.0	0	3
7	10	0	0.5	0.0	4	1
8	10	0	0.9	0.0	0	2
9	10	1	0.5	0.8	4	3
10	10	2	0.2	0.1	–	2
11	10	2	0.5	0.8	–	3
12	10	2	0.8	0.1	–	3
13	10	5	0.8	0.1	4	–
14	30	0	0.1	0.0	3	6
15	30	0	0.5	0.0	6	6
16	30	0	0.9	0.0	3	5
17	30	1	0.5	0.8	7	7
18	30	2	0.2	0.1	–	5
19	30	2	0.5	0.8	–	6
20	30	2	0.8	0.1	–	5
21	30	5	0.2	0.1	3	–
22	30	5	0.5	0.8	4	–
23	30	5	0.8	0.1	3	–
24	50	0	0.1	0.0	0	3
25	50	0	0.5	0.0	4	2
26	50	0	0.9	0.0	0	3
27	50	1	0.5	0.8	4	3
28	50	2	0.2	0.1	–	3
29	50	2	0.5	0.8	–	3
30	50	2	0.8	0.1	–	3
31	50	5	0.8	0.1	4	–
32	70	0	0.1	0.0	1	–
33	70	0	0.5	0.0	6	–
34	70	0	0.9	0.0	1	–
35	70	1	0.5	0.8	4	–
36	70	5	0.2	0.1	4	–
37	70	5	0.5	0.8	4	–
38	70	5	0.8	0.1	4	–

Table 6.2: Table giving the counts for each intended DoE-point for both chemistries. Zeros in the LFP counts indicate that no experiment was conducted at the certain setting although it was intended by the design.

Comparing the Intended and Actual Factor Settings

We shall compare the actual values with the intended ones by means of histograms and scatter plots. The way these actual values are determined will also be covered in the next chapter. Since they represent an average value over time, for now it is only important to keep in mind that these values depend on the response, which in all cases is a time span until an end-of-life criterion is met. In order to not depict too many almost identical plots here, we shall give only one example, namely the histograms and scatter plots for LFP cells in figure 6.2.

`Temp` **settings:** The temperature factor was the easiest to control and set exactly. This seems to be no issue.

`Curr` **settings:** It is clear to see that although expected to be able to handle them, LFP cells too were unable to reach the high intended current of 5C, which was a known fact for NCA cells. Furthermore, note that generally average currents of above 1C are a rare sight.

`SoC` **settings:** Even though there seems to be quite a large statistical spread for this factor, with an obvious exception of 0.2 the `SoC` settings have been reached satisfactorily by most cells. Much the same can be said about the `dSoC` **settings**.

It is to be expected that several cells experienced an unpredictable cause for an early end-of-life and would thus have to be removed from the analysis. We shall investigate several such causes in a later chapter as well and eliminate certain extraordinary cells from the modeling process.

Issues to Keep in Mind

The following will keep coming up throughout the rest of the thesis as special care is to be taken with these situations.

- The data was obtained from a D-optimal design because of constraints on the factor settings.
- The resulting design was unbalanced.
- We are dealing with multiple responses. However, we shall consider them completely separately.
- Due to the difficulties in getting exact measurements on the response variables and keeping exact factor settings, we will be fitting models with the actual data as best we can determine them.
- The response variables need to be defined properly by means of a linear interpolation.

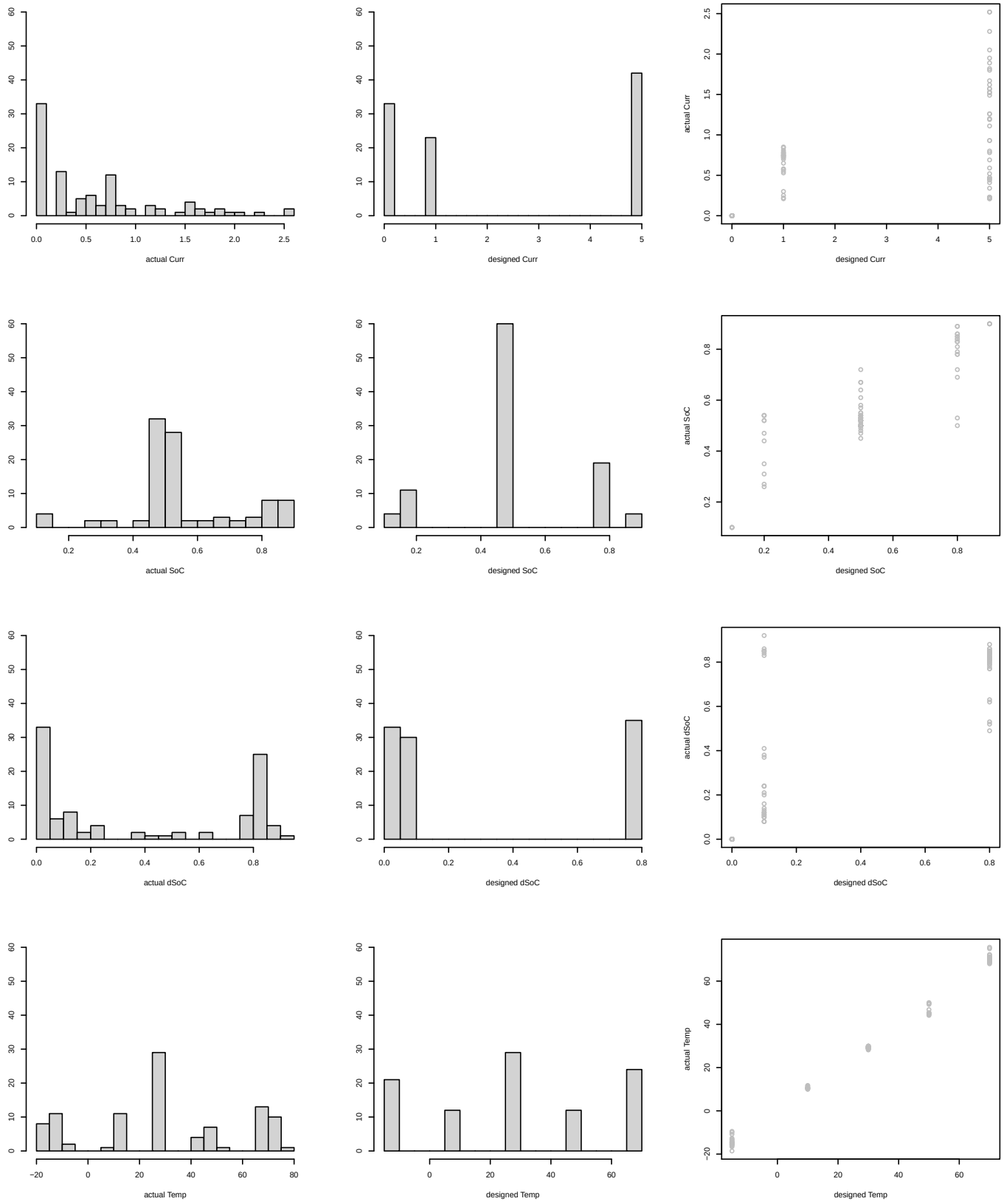

Figure 6.2: Histograms of the actual factor settings (left) and the intended factor settings (middle) as well as the scatter plots comparing them (right).

Chapter 7

End-of-Life Determination

Since the research at hand is aiming at the automotive industry, it is of course not sensible to declare a cell 'broken' only once it does absolutely not provide any energy to the user. Instead, there are several criteria that, once met, determine the cell's end-of-life. Below we consider the following two to some more detail because many aging effects manifest themselves through them: **capacity fade** and **internal resistance increase**.

We consider a cell that has experienced a capacity fade of 20% compared to its initial capacity 'broken'. It is thus of interest to estimate the average span until such a life-terminating state occurs. Here, we consider three different measures to determine this life span. These are either the *time* (in hours) or the *total number of cycles* it takes until the cell's capacity drops to 80% of its initial status, or the cell's *accumulated energy* respectively. We shall give a thorough investigation of the time criterion in the next chapter and deal with the other two criteria rather briefly. First, however, we give some explanations on the data at hand and the preparations thereof needed to determine the life spans according to the respective criteria.

7.1 Determining a Cell's End-of-Life

Since the cells under investigation were measured at irregular intervals and the exact time at which a cell's capacity has dropped to 80% or at which its resistance has increased to 300% of its initial value is unknown, we determine the approximate time of 'death'. Generally, this is done by an interpolation between the two points of measurements that border the interval in which the capacity falls below 80% or the resistance rises above 300% for the last time. Taking the last such an occurrence is necessary since due to the reference measurements at room temperatures the characteristic capacity or resistance curves are not usually monotonic over time.

The main function for the preparation of the data analysis to come is a procedure called `EoL_crit`. This function is tailored to our specific needs and might not be widely applicable, however, it is written in such a way that the life span determination can be varied slightly. That is to say, the time until one of the two end-of-life criteria was reached is not the only

response it can handle. Since time and date objects need special treatment, it has to be specified though whether or not one is dealing with time (default). In case of `time=FALSE` a user-specified response variable is needed. Here, any vector with absolute values (such as the cumulated energy or the total number of cycles considered later on) will work fine. However, these vectors should not already be standardized to the initial value per cell as the procedure does this internally and would not work correctly for such input.

Since this function is the base for all our investigations to follow, its use and functionality is explained in detail and attemptedly in a style following the `R` documentation in the appendix.

Let us now take a closer look at how the function actually works: After determining which case we are dealing with, for each cell we first do an index search to find out between which two measurements the cell's end-of-life criterion is met for the last time. For this purpose we revert the respective subset of values and therefore search for the first such occasion. However, there were many cells that did not meet this criterion. For these cells their end-of-life time (or other response) was then prudely set to the total elapsed time until the last measurement (or the value at the last measurement respectively). For option `time=TRUE` we consider two cases depending on whether such a cell had a short circuit or not, if such an indication vector was provided. If the cell was not terminated prematurely by a short circuit, its end-of-life is determined the same way as described above. However, if the cell experienced a short circuit, its end-of-life time is extrapolated by determining the straight line between the first and last measurement and extending it to the point of end-of-life according to the according criterion. Finally, if `remTime`, the parameter indicating a minimum life-time, is provided and `time=TRUE`, the end-of-life times that are below `remTime` $*$ 24 are replaced by NAs. This multiplicative factor is needed since the response values are given in hours and `remTime` denotes days. Note that the according cells could not simply be removed because a cell's identity is preserved by the position in the vector only.

Figures 7.1 and 7.2 show the trellis plots for each cell, visualizing the respective normed capacity and resistance at the times of measurement as well as the calculated end-of-life time.

Several cells were clearly malfunctioning from the beginning and should thus be removed from further analysis. What cannot be seen from the normed responses shown here but became obvious upon a closer inspection of the respective cells is that most of these malfunctioning cells already had a very low initial capacity. The practitioner felt comfortable with disregarding these cells completely since their extremely short lifetimes were not triggered by conditions brought about by the experiment. Such cells would grossly bias (and likely falsify) any linear model and of course the group means considered for the analysis of the experimental design.

For the resistance criterion some additional cells need to be excluded from the analysis as these were short circuit cells with a negative rise between their first and last measurement. Such cells would have a negative determined life-span.

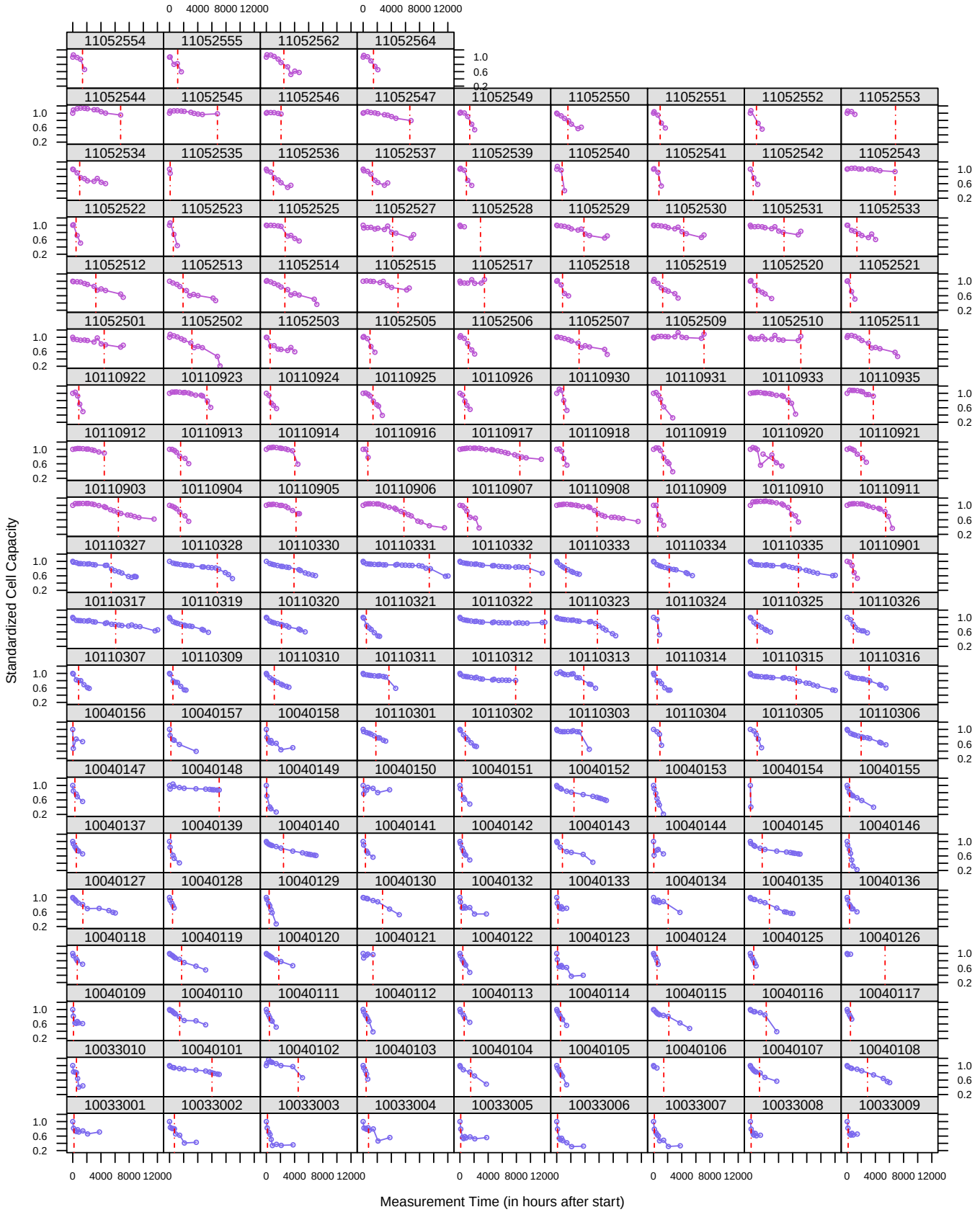

Figure 7.1: Trellis plot showing the normed capacity curve for each cell, including the determined end-of-life time. Purple colored cells are NCA, blue colored cells are LFP.

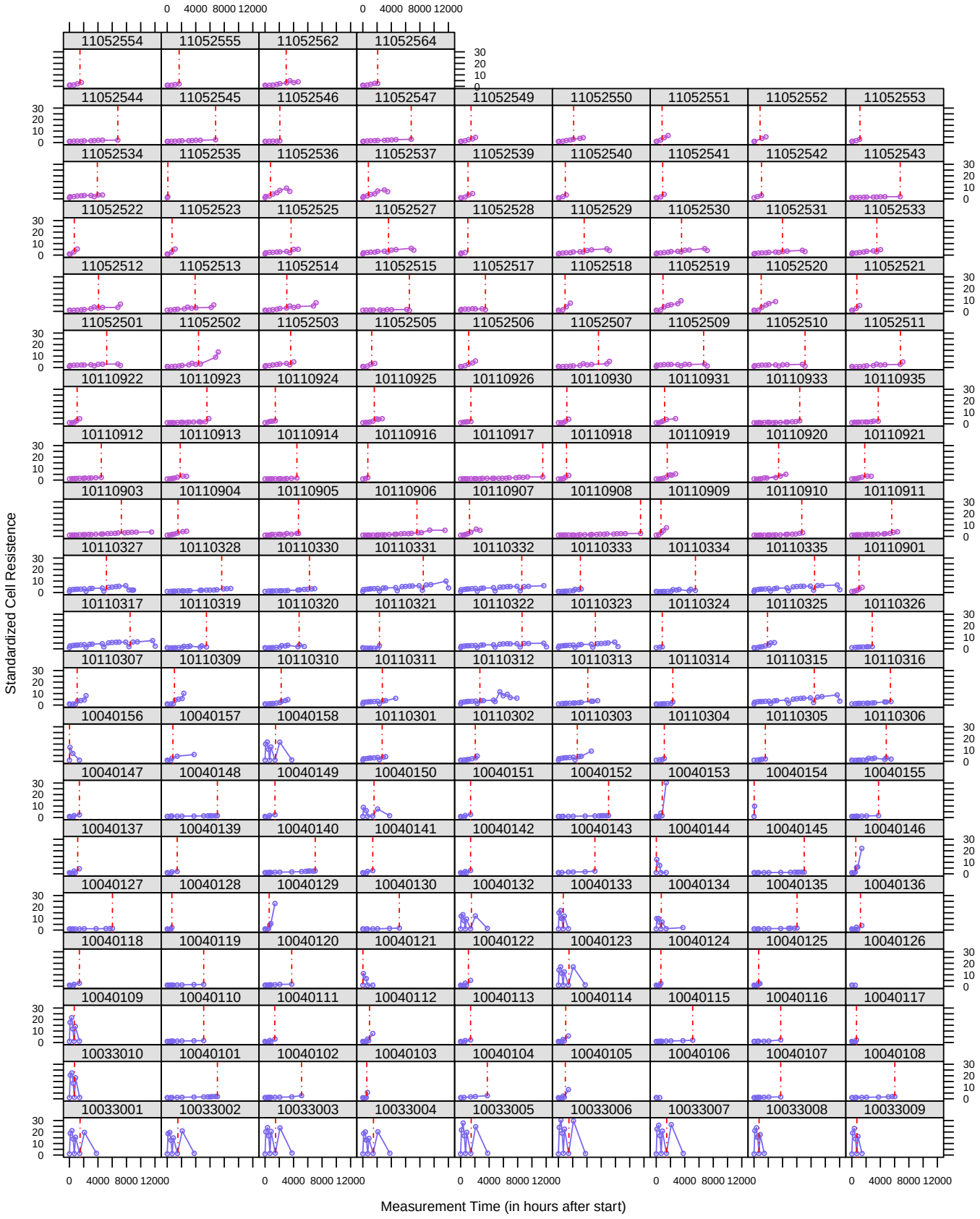

Figure 7.2: Trellis plot showing the normed resistance curve for each cell, including the determined end-of-life time. Purple colored cells are NCA, blue colored cells are LFP.

7.2 Further Preparational Information

One important measure to be introduced is the factor `Leak`, which was also recorded by the practitioner and states whether a cell was visibly broken upon opening (`Leak = 1`), whether it was expected to have experienced some sort of premature life-terminating event (`Leak = 0.5`) or whether it was sound (`Leak = 0`).

Furthermore, it was already pointed out that designed and actual values differed greatly for several settings. It is of course sensible to analyze the actual values in the upcoming chapters. However, since these are continuous variables, which renders their use in several plots pointless, we have to make an attempt at restoring certain groups. Table 7.1 gives the limits for these groups (note that the names are kept mostly the same as for the initially intended settings although they do not always fit nicely) and is inspired by the graphics in figure 6.2 and their omitted analogons for the resistance criterion.

Cap80.LFP							
	group	interval	group	interval	group	interval	**Levels**
Temp	-15 50	[-20, -5] [40, 55]	10 70	[5, 15] [65, 80]	30	[25, 30]	5
Curr	0	[0, 0.2]	1	(0.2, 1]	2	(1, 3]	3
SoC	0.1 0.8	[0.1, 0.15] (0.6, 0.85)	0.2 0.9	[0.25, 0.35] [0.85, 0.9]	0.5	[0.4, 0.6]	5
dSoC	0	[0, 0.05]	0.1	(0.05, 0.6]	0.8	(0.6, 0.95]	3

Cap80.LFP							
	group	interval	group	interval	group	interval	**Levels**
Temp	10	[5, 20]	30	[25, 35]	50	[45, 55]	3
Curr	0	[0, 0.2]	1	(0.2, 1]	2	(1, 2.5]	3
SoC	0.1 0.9	[0, 0.1] (0.85, 0.95]	0.5	[0.35, 0.6]	0.8	(0.6, 0.85]	4
dSoC	0	[0, 0.05]	0.1	[0.1, 0.4]	0.8	[0.5, 85]	3

Res300.LFP							
	group	interval	group	interval	group	interval	**Levels**
Temp	-15 50	[-20, -5] [40, 55]	10 70	[5, 15] [65, 80]	30	[25, 30]	5
Curr	0	[0, 0.2]	1	(0.2, 1]	2	(1, 3.5]	3
SoC	0.1 0.8	[0.1, 0.15] (0.6, 0.85)	0.2 0.9	[0.25, 0.4] [0.85, 0.95]	0.5	(0.4,0.6]	5
dSoC	0	[0, 0.05]	0.1	(0.05, 0.3]	0.8	[0.55, 0.9]	3

Res300.NCA							
	group	interval	group	interval	group	interval	**Levels**
Temp	10	[5, 20]	30	[25, 35]	50	[45,55]	3
Curr	0	[0, 0.1]	1	(0.1, 1]	2	(1, 3]	3
SoC	0.1 0.8	[0.1, 0.15] (0.6, 0.85]	0.2 0.9	[0.25, 0.4] (0.85, 0.9)	0.5 0.9	(0.4, 0.6] [0.9, 0.95]	6
dSoC	0	[0, 0.05]	0.1	(0.05, 0.3]	0.8	[0.55, 0.9]	3

Table 7.1

Chapter 8

Investigating a Cell's Life Span

It has to be expected that cells with different cell chemistry do not deliver comparable results. For this purpose we have divided the data into the according parts and shall do all consequent investigations for the two chemistry types, LFP and NCA respectively, separately.

In the following we do a graphical investigation of the time in hours that passed until the cells reach their end-of-life criterion. For this purpose, several different means of displaying this measure are employed. These comprise general summaries to give a quick overview of some influence, a more thorough analysis with boxplot series and histograms and of course various scatter plots to show certain dependencies and interactions. Summaries can be very useful to identify extraordinary cells and with these the bad cells for the resistance criterion became obvious.

Furthermore, it needs to be investigated whether the factor `Leak` had some influence on the responses as well. We shall therefore include this factor in the graphical analysis.

This chapter focuses entirely on the time until the EoL-criteria have been met, a similar analysis for other responses is possible but omitted at this point.

8.1 Elapsed Time until EoL

Before starting a differentiated analysis, we do a quick check on how reasonable the EoL-determination works by comparing the two criteria with each other.

We see that for most points the criteria are in accordance. Some cells seem to do really well for one criterion, however, but considerably worse for the other. This might be because we consider the last time a cell 'crosses' below the 80% capacity or above 300% resistance line, which for several cells happened quite early although the cell then continued on well above 80% capacity or well below 300% resistance. However, this was the way we defined the end-of-life condition, so this is acceptable.

Note that for LFP cells generally the capacity criterion is reached earlier than the resistance criterion, although the maximum value is higher for the capacity criterion.

For NCA cells it seems as if the criteria match each other very well.

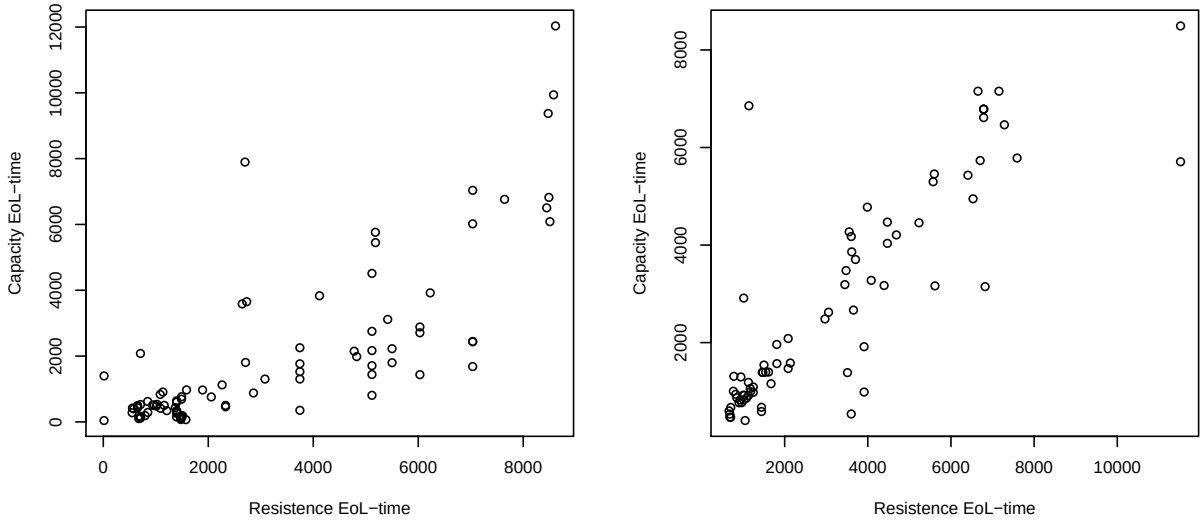

Figure 8.1: Scatter plots showing that the two end-of-life criteria for the most part are in good accordance. LFP cells are shown in the left plot, NCA cells are given on the right-hand side.

8.1.1 Summaries

We start the investigation of the elapsed time until the cells met their end-of-life criteria by printing several summaries, where we first print the results for the capacity criterion and then proceed with the resistance criterion.

Capacity Criterion

The first summary we are interested in is that of the complete and uncleaned data.

```
> summary(Cap80.LFP$Cap80Time)
    Min.     1st Qu.     Median       Mean     3rd Qu.       Max.
    26.0       339.8      786.5     1889.0      2246.0    12030.0

> summary(Cap80.NCA$Cap80Time)
    Min.     1st Qu.     Median       Mean     3rd Qu.       Max.
    88.0       941.0     1582.0     2770.0      4272.0     8492.0
```

First of all, note that the minimum value for LFP is indeed very small, which means that some cells did merely survive a day. Since we are working with time here, we could of course remove all cells that effectively reached 80% of their initial capacity before a certain amount of days was over. Let us assume we are only interested in cells that lasted for at least two weeks. We can then observe how the mean and median values change.

```
> summary(Cap80.LFP$Cap80Time[Cap80.LFP$Cap80Time > 24*14])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
   339.0      550.0    1445.0    2453.0     3058.0   12030.0


> summary(Cap80.NCA$Cap80Time[Cap80.NCA$Cap80Time > 24*14])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
   402.0      970.2    1749.0    2806.0     4318.0    8492.0
```

One thing that is not included in the summary is the total count of observations. We therefore give these numbers separately. A total of 24 (out of 98) LFP cells and only one (out of 77) NCA cell did not last two weeks. This explains why the mean and median for LFP cells are shifted a lot whereas for NCA cells they remain almost the same.

Furthermore, we can clearly see that the means and medians of the two chemistries are now rather close. Additionally it is noteworthy that most NCA cells were started after their LFP counterparts and that the extreme values (maxima) can and should thus not be compared directly.

Next, it is also of interest to investigate the factor `Leak` more closely. As mentioned before, cells that were visibly broken were marked with a leak of 1 and cells that are suspected to have been broken were marked with a leak of 0.5. We shall now see how the three specifications of this factor (0, 0.5, 1) influence the life time of the cells.

```
> summary(Cap80.LFP$Cap80Time[Cap80.LFP$Leak==1])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
    41.0      436.2     584.0    1301.0     1608.0    5375.0

> summary(Cap80.LFP$Cap80Time[Cap80.LFP$Leak==0.5])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
    30.0      278.0     517.0    1085.8     1440.0    7896.0

> summary(Cap80.LFP$Cap80Time[Cap80.LFP$Leak==0])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
    26.0      483.5    1806.0    3006.0     5138.0   12030.0

------------------------------------------------------------------------

> summary(Cap80.NCA$Cap80Time[Cap80.NCA$Leak==1])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
   587.0      887.8    1264.0    2214.0     3506.0    5457.0

> summary(Cap80.NCA$Cap80Time[Cap80.NCA$Leak==0.5])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
    88.0      862.0    1089.0    1785.0     1582.0    6855.0

> summary(Cap80.NCA$Cap80Time[Cap80.NCA$Leak==0])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
   402.0     1328.0    3180.0    3489.0     5518.0    8492.0
```

Again, we have to give the counts for the cells in the respective groups separately. Note that this distribution will later on become clearer in the plots. We state the ratio between sound (`Leak = 0`), suspicious (`Leak =0.5`) and leaked (`Leak = 1`) cells.

$$\begin{aligned} \text{LFP} \quad & \text{sound : suspicious : leaked} = 39 : 41 : 18 \\ \text{NCA} \quad & \text{sound : suspicious : leaked} = 42 : 45 : 10 \end{aligned}$$

When comparing the means and medians of the respective groups, it becomes clear that both broken and suspicious (or unknown) cells perform much worse than sound cells. We can, however, not remove those two groups from the further investigation as we would be omitting half of the results obtained and thus introduce a considerable bias.

We, however, should definitely consider removing those cells that were malfunctioning from the start. The following summaries show how this changes the distributions of the EoL-times.

```
> summary(Cap80.LFPclean$Cap80Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
    41.0      375.0     834.0    1948.0     2343.0   12030.0

> summary(Cap80.NCAclean$Cap80Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
   402.0      970.2    1749.0    2806.0     4318.0    8492.0
```

For both chemistries the lowest value has been removed. Note that for NCA there was only one cell removed and it seems to have been by far the worst cell. For LFP three cells were removed but the general distribution has not changed much.

Nonetheless, in the following we shall give all plots and do all further investigations on the cleaned data sets 'Cap80.LFPclean' and 'Cap80.NCAclean'.

Resistance Criterion

Here, again, we start our investigation with the complete data set containing even cells that should best be excluded.

```
> summary(Res300.LFP$Res300Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
  -11250       1038      1542      3102       5118     28200

> summary(Res300.NCA$Res300Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
      88       1081      2084      3200       4470     11520
```

This shows clearly why we have to remove two additional cells for the investigation of the resistance criterion (when compared to the capacity criterion). The extrapolation for two cells experiencing a short circuit deliver simply unbelievable results. First of all, a life time should by no means be negative. This was caused by a lower resistance at the final measurement than at the first one. The negative resistance slope between those two measurements causes an impossible negative estimated time of when that cell would reach 300% of its initial capacity. The slope for the second cell was barely positive, in fact so little so that the end-of-life time here too is ridiculous, although in the opposite direction.

Giving the summaries for the cleaned data frames should get rid of this problem.

```
> summary(Res300.LFPclean$Res300Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
      14       1091      1586      3071       5118      8615

> summary(Res300.NCAclean$Res300Time)
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
     663       1116      2110      3241       4524     11520
```

We see that removing, among others, the two extremes for LFP has made the distribution much more moderate and believable. Note that for NCA cells, much the same as it has been for the capacity criterion, the removal of only one cell has increased the minimum life time by a stern 24 days. Two NCA cells are tied for the maximum life time and although the unusually high values seem to stem from an unlikely extrapolation none of these cells is a short circuit cell.

Furthermore, note that although the longest-living LFP cell survived around 12000 hours according to the capacity criterion, whereas NCA cells only reached around 8000 hours, the situation is turned around for the resistance criterion.

Before moving on to a graphical investigation, we also check the influence of the factor `Leak` on the resistance criterion.

```
> summary(Res300.LFPclean$Res300Time[Res300.LFPclean$Leak==0])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
     556       1480      5118      4483       7037      8615

> summary(Res300.LFPclean$Res300Time[Res300.LFPclean$Leak==0.5])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
      15        893      1397      2240       3745      7037

> summary(Res300.LFPclean$Res300Time[Res300.LFPclean$Leak==1])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
      14        708      1397      1905       2732      5118

# ------------------------------------------------------------

> summary(Res300.NCAclean$Res300Time[Res300.NCAclean$Leak==0])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
     663       1707      3634      4240       6689     11520
```

```
> summary(Res300.NCAclean$Res300Time[Res300.NCAclean$Leak==0.5])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
     681       1012      1244      1902       1878      6408

> summary(Res300.NCAclean$Res300Time[Res300.NCAclean$Leak==1])
    Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
     822       1055      1443      2260       3299      5599
%
```

This looks as expected. Sound cells show a distribution that is clearly shifted to the right (positive) compared to both other groups. Not only are the maxima highest there but so are the means, medians and minima. The upcoming plots should be able to give a better visualization thereof.

8.1.2 Histograms

This part is dedicated to the investigation of only two aspects, namely the influences of the factor leak and that of extreme temperatures (only for LFP cells) on the distribution of the life expectancy of the cells.

Factor Leak

Figure 8.2 shows more clearly than the summary for the capacity criterion would have suggested that most leaked LFP cells were broken after a very short time. For NCA, there were a few broken cells that survived an unexpectedly long time.

Due to their greater number the fast end-of-life for suspicious cells is almost more striking as the greater part of this group did not live to see 2000 hours. However, note one peculiar peak of suspicious cells at around 5000 hours in the LFP capacity part of the plot.

However, the question remains if there exists a different non-observed and unmonitored factor that somehow triggered a large amount of leakage.

We will further investigate the influence of this factor by means of box plots in the next section.

Factor Extreme Temperatures

Note again that NCA cells could not be aged at extreme temperatures due to technical restrictions. This gives us all the more reason not to compare the various distributions of the two cell chemistries with each other, simply because the underlying conditions were not identical.

Figure 8.3 confirms that extreme aging temperatures cause a very early death of cells, regardless of the EoL-criterion. Furthermore, we can also observe the aforementioned phenomenon that several cells aged at extreme temperatures survived comparatively longer according to the resistance criterion than to the capacity criterion here.

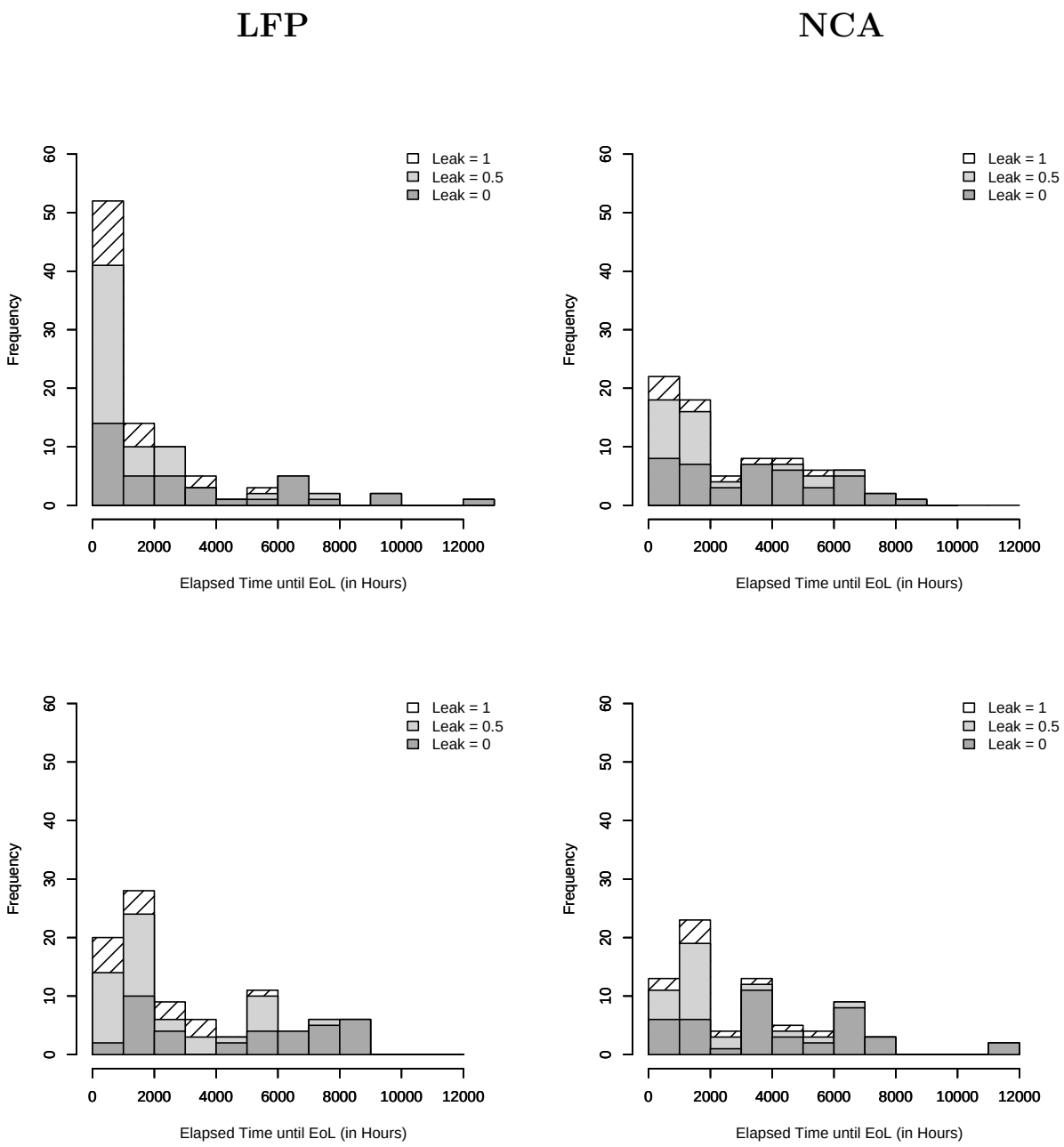

Figure 8.2: Histograms showing the life time distributions for each value of the **factor** `Leak`. The top plots visualize the capacity criterion, whereas the bottom plot show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

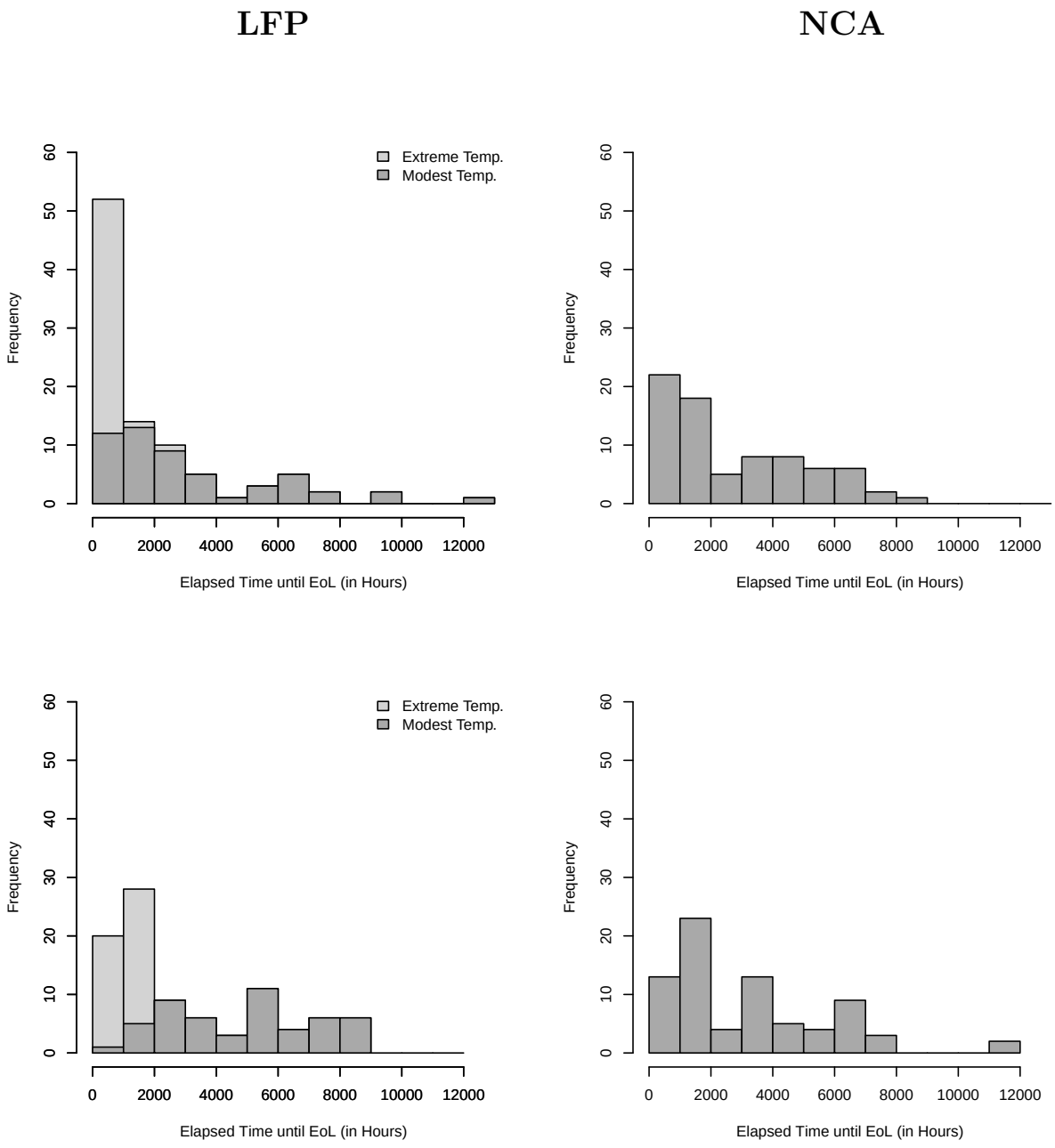

Figure 8.3: Histograms showing the influence of **extreme temperatures**. Note that although not aged at such conditions, NCA cells are included in the plots to show their general distribution as well. The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

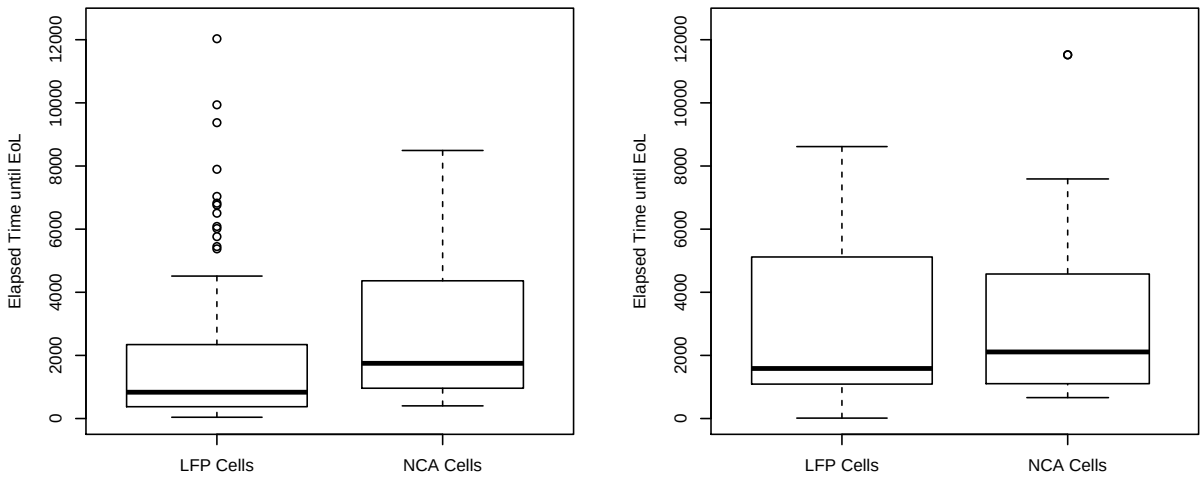

Figure 8.4: The left plot shows the **capacity criterion**, whereas the right plot shows the **resistance criterion**.

8.1.3 Boxplot Series

A boxplot can be seen as a graphical representation of the `summary`-command in `R`. Hereby, the median of the data at hand is displayed as a bold horizontal line framed in a box showing the inter-quartile range (iqr), i.e. the distance between the 1^{st} and 3^{rd} quartile. The bars of the boxplot by default extend to both sides of the box to a length of $1.5 \cdot iqr$, all other data is shown as a circle and referred to as an outlier. Throughout this section, the sample size of the respective categories in the box plots are reflected in the widths of the boxes as these are proportional to the square root of the number of observations. This makes it easier to compare several groups while at the same time not drawing any faulty conclusions from several extraordinary observations.

Factor Chemistry

Given the obvious negative effect of extreme operating temperatures that were verified by the histogram and the fact that all NCA cells were run at more modest conditions, it is to be expected that the boxplot of NCA cells looks structurally different than that of LFP cells.

In figure 8.4 one can clearly see that for the capacity criterion NCA cells generally have a higher life expectancy than their LFP counterparts but that there are several LFP cells that lasted an unusually long time. Note, however, that no information can be inferred from these outliers due to the fact that NCA cells were generally started with an offset, as stated before.

Interestingly but in accordance with the initially displayed scatter plots, the cell chemistries

seem to behave more similarly for the resistance criterion.

Also note that the box widths state that there are slightly more LFP than NCA cells. The exact counts are 95 (or 93 for resistance) LFP versus 76 NCA cells.

Factor Leak

As we have already seen in the general summary section and the histogram, the `Leak` factor has a great influence on the life time of a cell. Therefore, we shall investigate this further by the use of appropriate box plots.

Both LFP plots in figure 8.5 seem to suggest that the two groups `Leak = 1` and `Leak = 0.5` could be considered as one group since both result in similarly bad life span values. For the resistance criterion at least, unknown cells fare slightly better than leaked ones although the medians are equal. Also note the big difference in life span medians between sound cells according to the two EoL-criteria. This is very interesting given the results of the scatter plots comparing the criteria earlier.

From the NCA plots in figure 8.5 it is also clear that the `Leak` factor has a negative effect on the cells' life span. Here, however, note that the box of the leaked cells extends by far wider into the desired direction than that of the unknown cells. Only a few outliers in the group of the unknown cells performed considerably better. As expected from above, the plots for the two different EoL-criteria here look very similar.

As a final remark, note that NCA cells have the higher group means throughout for the capacity criterion but that the same does absolutely not hold for the resistance criterion.

In the following, we also explore the influences of the factors chosen for the experimental design. It has to be pointed out, though, that these are only single-factor influences and that the interactions thereof will have to be considered additionally later on.

Note that for these plots the grouped actual factor settings are used, so please refer to table 7.1 for the translation between group name and interval.

Factor Temperature

Figure 8.6 indeed confirms the vast negative effect of extreme temperatures as the according boxes are hardly recognizable. For LFP cells the influence of the temperature could likely be modeled well centering the values around some modest temperature. This, however, will be part of the upcoming modeling chapter.

Since it was technically not possible to age NCA cells at the extreme temperature settings, it is clear that an effect similar to that for LFP cells cannot be observed here. A little surprising though, the structure of the NCA box plots in figure 8.6 does not even resemble the more moderate (middle) part of the LFP box plots. Even though the group medians are also highest for $+10°C$, the rest of the box plots suggest that NCA cells perform better at a slightly higher temperature.

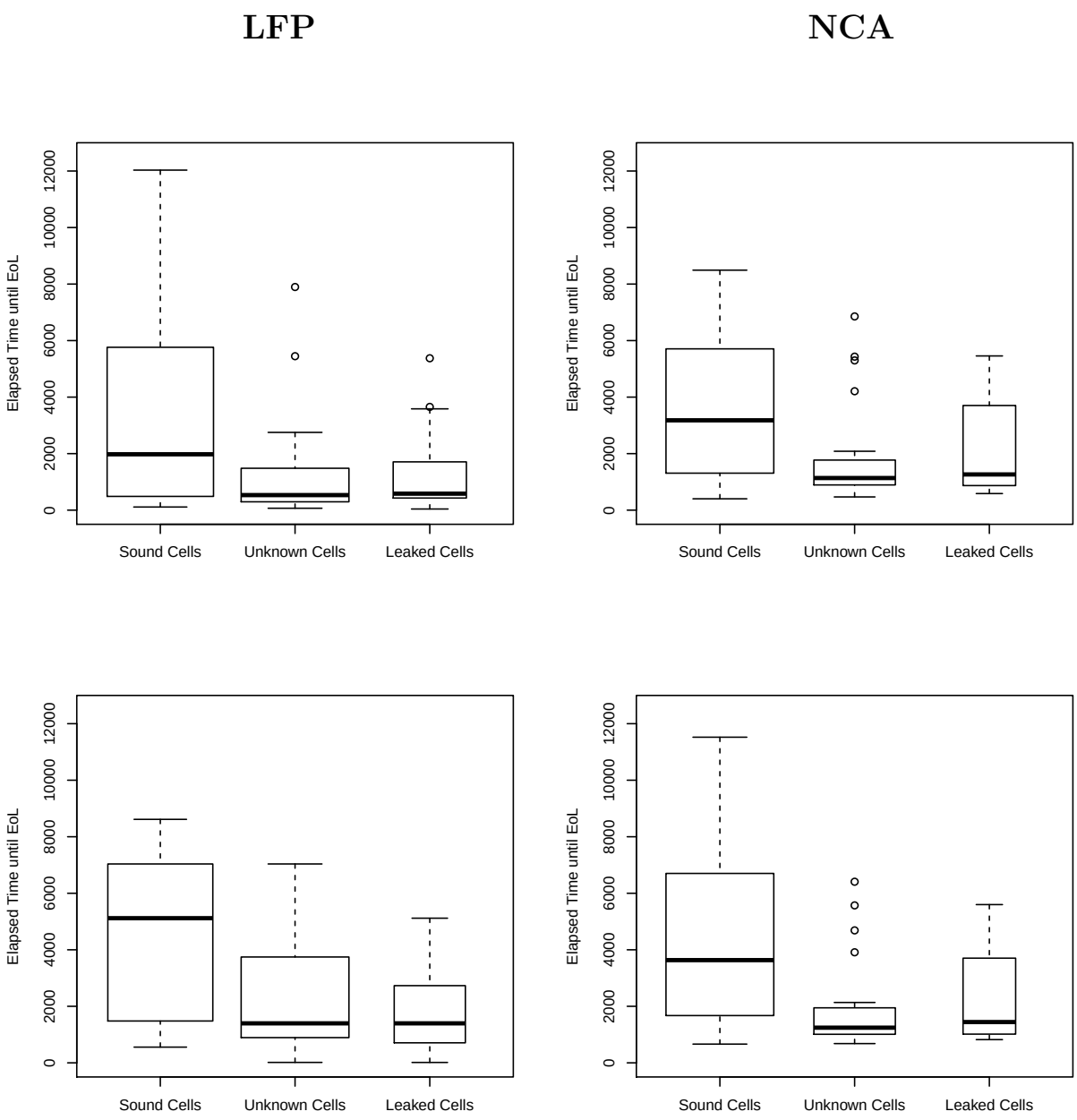

Figure 8.5: Box plots showing the influence of the **factor** `Leak`.
The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

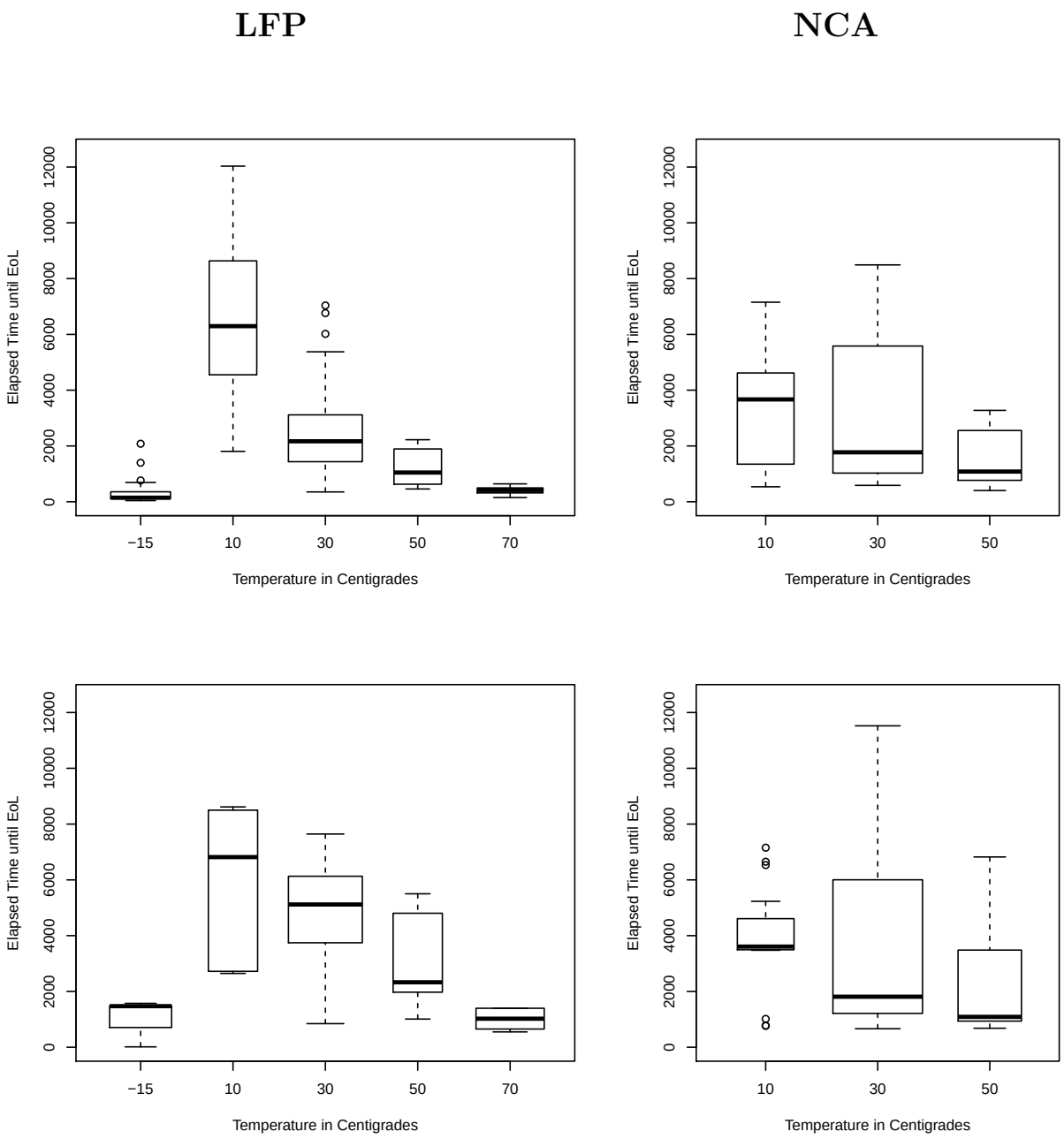

Figure 8.6: Box plots showing the influence of the **factor** `Temp`.

The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

Factor State of Charge

The first thing that is particularly striking for LFP cells in figure 8.7 is the comparatively uneven distribution over the possible settings for this factor. For NCA cells one intended factor setting, namely 0.2, could not be observed at all and was therefore removed completely.

For the setting `SoC`=0.9 in the LFP capacity plot a very interesting box form can be observed. Note that it has by far the highest group median and the greatest *iqr*. Note that the resistance plot also shows a large box for `SoC`=0.9 but the median is highest for `SoC`=0.1 here. The boxplot of `SoC`=0.5 is also noteworthy in the LFP capacity as it shows the highest overall values in its outliers. This is not very surprising since that group contains almost two thirds of all cells. Generally, the LFP plots for the two criteria look similar but show significant differences.

The plots for NCA cells again paint a completely different picture. Here, `SoC`=0.1 has the highest group medians for both criteria although the boxes for `SoC`=0.9 are a close second. Also, note that the outliers from `SoC`=0.5 are again the longest-lasting cells, however, they do not exceed the others by much, except for one extraordinary outlier for the resistance criterion. Note that only two fifths of NCA cells were aged at `SoC`=0.5 and the overall distribution is much more evenly spread.

Factor Delta State of Charge

Figure 8.8 shows that whereas this factor seems to have considerable influence on NCA cells, it does not make such a big difference for LFP cells.

Even though the medians of LFP storage cells (`dSoC`=0) are also the highest group medians and the corresponding boxes extend slightly further than the others, the structure of each of the pairs of three box plots is comparable. For the resistance criterion it is interesting that the lowest value is included in the best group, maybe suggesting some other influence here.

For NCA cells it is clear to see that storage cells have the highest medians. These plots suggests that for NCA cells storage is the only aging condition promising favorable cell life spans. This behavior has to be kept in mind throughout the rest of the investigations as it is likely to weaken other assertions made for this cell chemistry type.

Factor Current

It needs to be pointed out that cells with a operating current of `Curr`=0 coincide with the cells that were aged at `dSoC`=0 since these are storage cells. Thus, for both cell chemistries the box plots for the storage cells shown for this factor are the very same as those seen for factor dSoC.

Generally, we have to be careful when drawing conclusions here from the single-factor investigation because these two factors are interlinked closely and only few combinations were considered by the underlying experimental design. The exact distribution can be seen in the scatter plot section.

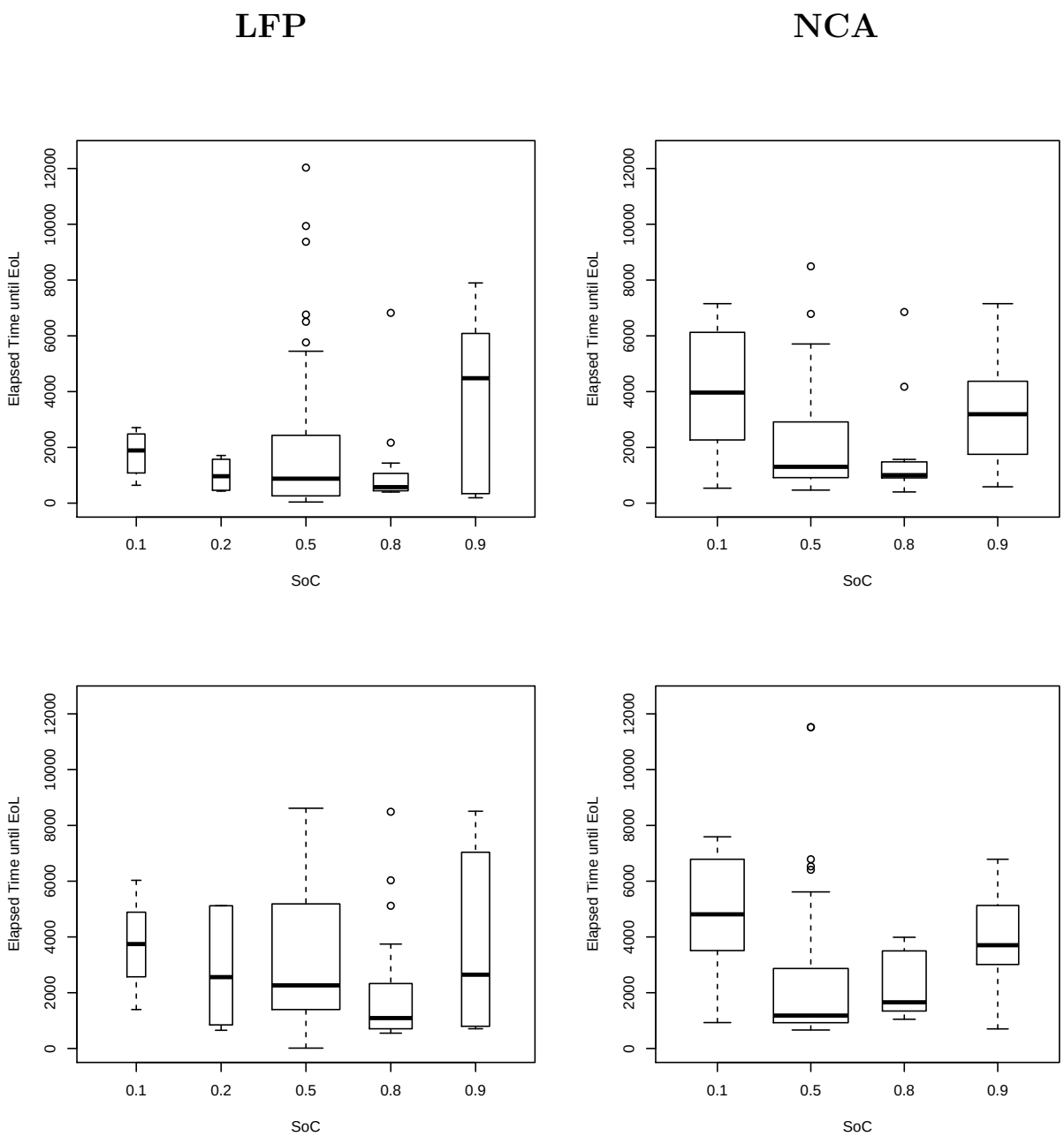

Figure 8.7: Box plots showing the influence of the **factor** `SoC`.

The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

Note that here the box widths for the various plots are not comparable.

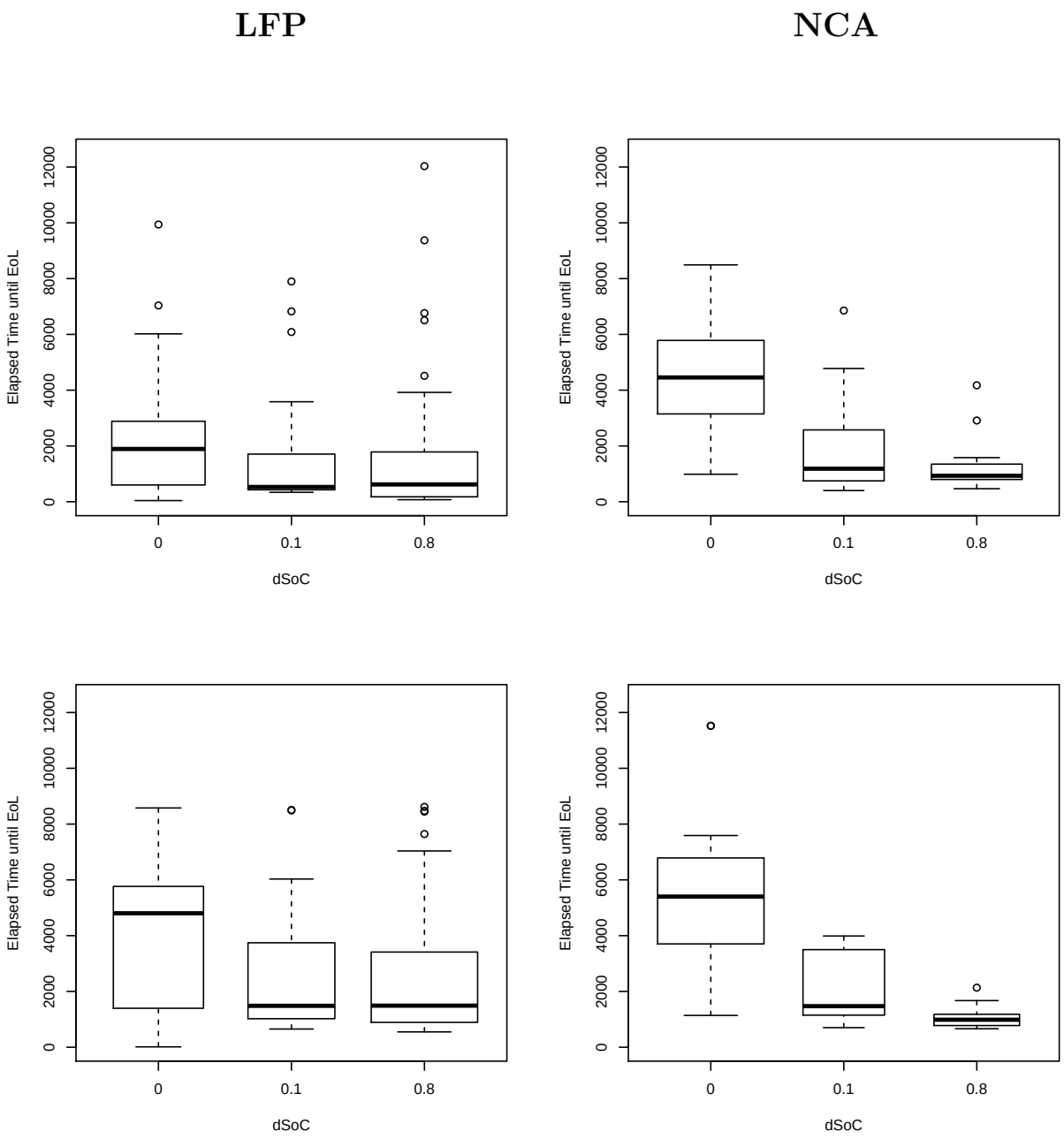

Figure 8.8: Box plots showing the influence of the **factor** `dSoC`.

The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

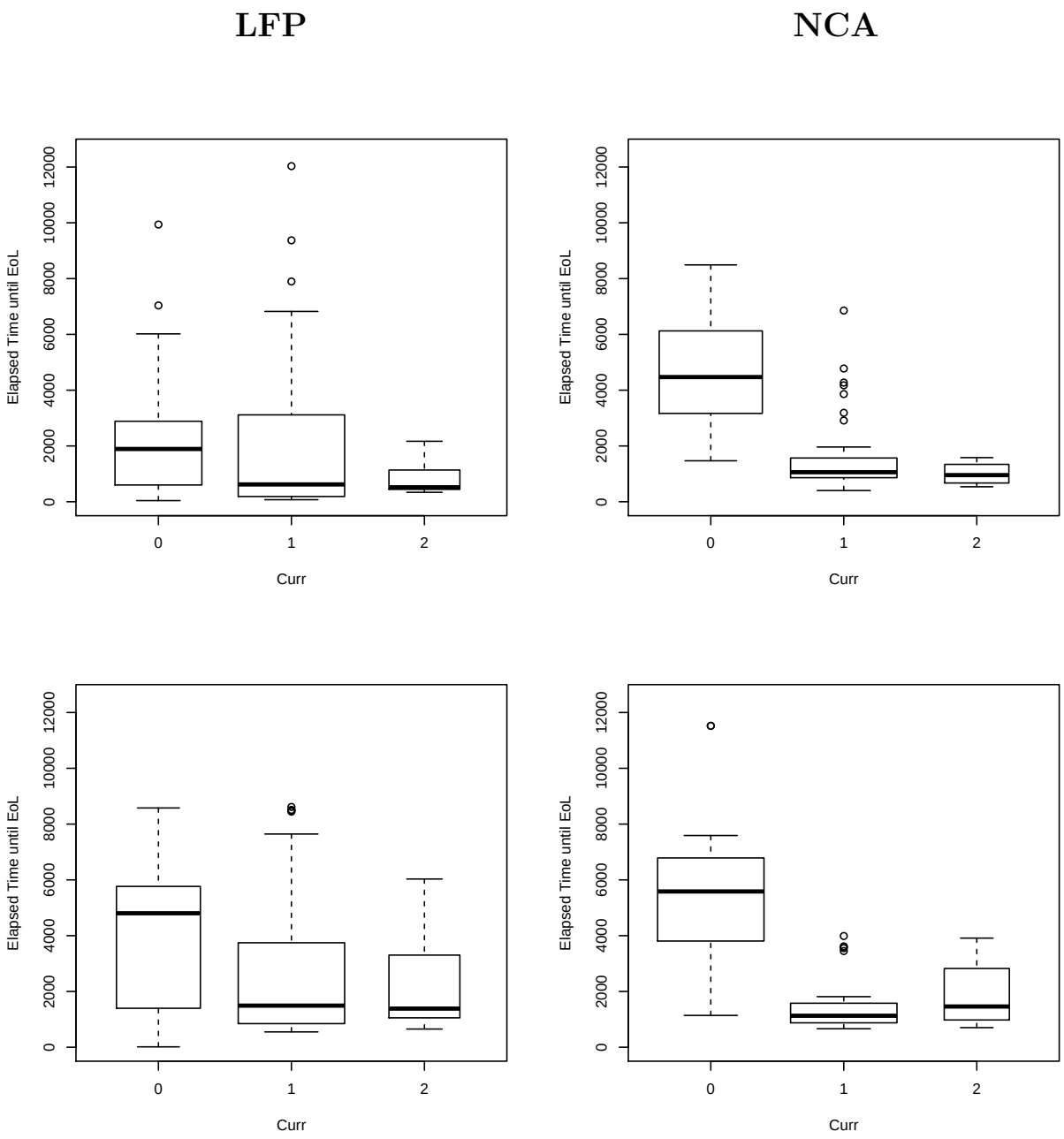

Figure 8.9: Box plots showing the influence of the **factor** `Curr`.
The top plots visualize the capacity criterion, whereas the bottom plots show the resistance criterion. Furthermore, the left plots show LFP and the right plots show NCA cells.

Figure 8.9 suggests that the current has a strong influence on NCA cells as `Curr`=0 yields the best results by far, whereas the high current of $2C$ fares particularly badly for the capacity criterion.

For LFP cells it is also clear that a high current, and keep in mind here that these cells were intended to reach $5C$, results in short cell lives. However, there seems to be hardly any distinction between cells aged at $1C$ and those aged at $0C$ except for the median value, which is highest for storage cells as had to be expected.

8.1.4 Scatter plots

The plots shown in figure 8.10 merely form a feeble attempt at capturing the whole nature of the data and show all influential factors combined in one graphic. The result is a rather unclear scatter plot that definitely needs some getting accustomed to.

Here, the temperature factor was chosen to be displayed on the x-axis, whereas the other three factors need to be highlighted by means of graphical measures. The different values of the `SoC` factor are indicated by different colors, those of the `dSoC` factor by the point character and the current factor, finally, is accentuated by the different point sizes in use. Note, however, that the latter is hard to distinguish, especially in connection with different point characters.

One thing these plots manage to show nonetheless is that there are only certain combinations of the factors present. For example, red LFP cells for the resistance criterion are rare which implies that there were only very few cells effectively aged at `SoC`=0.9. Furthermore, note that for the resistance criterion many cells aged at $10°C$ overlap. Nonetheless, it is clear that for this criterion temperatures $10°C$ and $30°C$ fare almost equally well, although the standard deviation should be slightly smaller at $10°C$.

We can also see that NCA cells aged at a high current (large symbols) died very early, making the plots more clearly laid out. Since storage cells are indicated by small crosses, note the large amount of those cells in the NCA plots that survived remarkably long.

Since there are hardly any new conclusions to be drawn from these plots, it being perfectly clear that NCA cells survived longest if stored without cycling, we shall now proceed to look at the matter from a slightly different angle, by means of a 3D scatter plot. Note that there are many different possibilities to produce such a plot but that the method used and depicted here is the only one suitable for our needs as it allows to incorporate the vertical lines indicating more clearly to which group, given by the coordinate on the x- and y-axes, the respective point belongs. Nonetheless, it seems to be impossible to change the colors of these lines independently of their end points, which means that the resulting plot is more colorful than intended, or to alter the angle of the y-axis label.

Here, we depict the temperature factor on the x- and the current factor on the y-axis, which means that we can drop the messy distinction by point size. Note that for a better view the plots are tilted slightly, which unfortunately ruins the scaling on the bottom axes. This does not bother us too much since we know that only few possible values have to be considered and in figure 8.11 the factors have actually been placed quite well. Note the slight ambiguity for the combinations `Temp`=10, `Curr`=0 and `Temp`=-15, `Curr`=2 in the

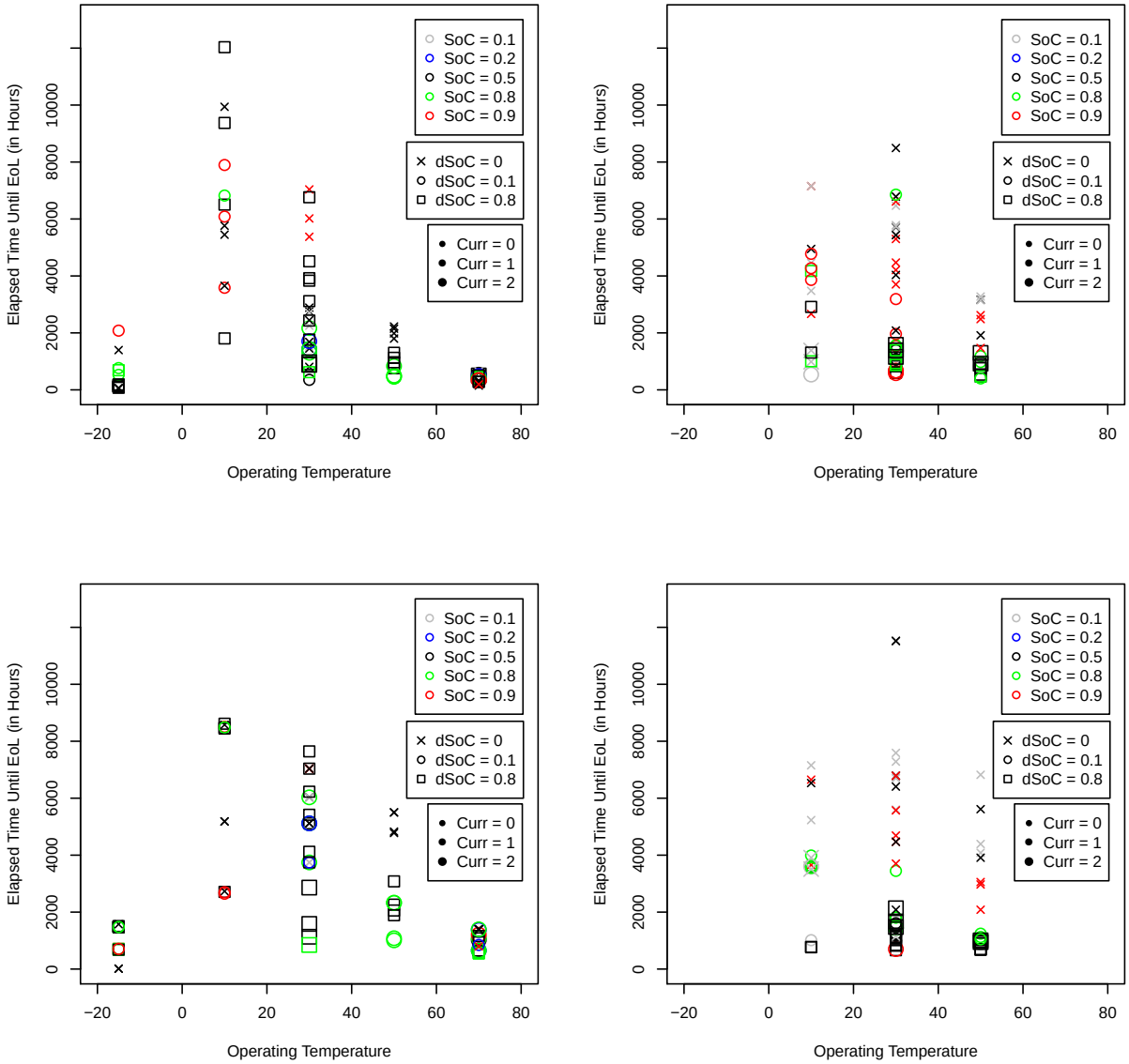

Figure 8.10: All-in-one scatter plots for the capacity criterion on top and the resistance criterion on the bottom, LFP cells on the left and NCA cells on the right.

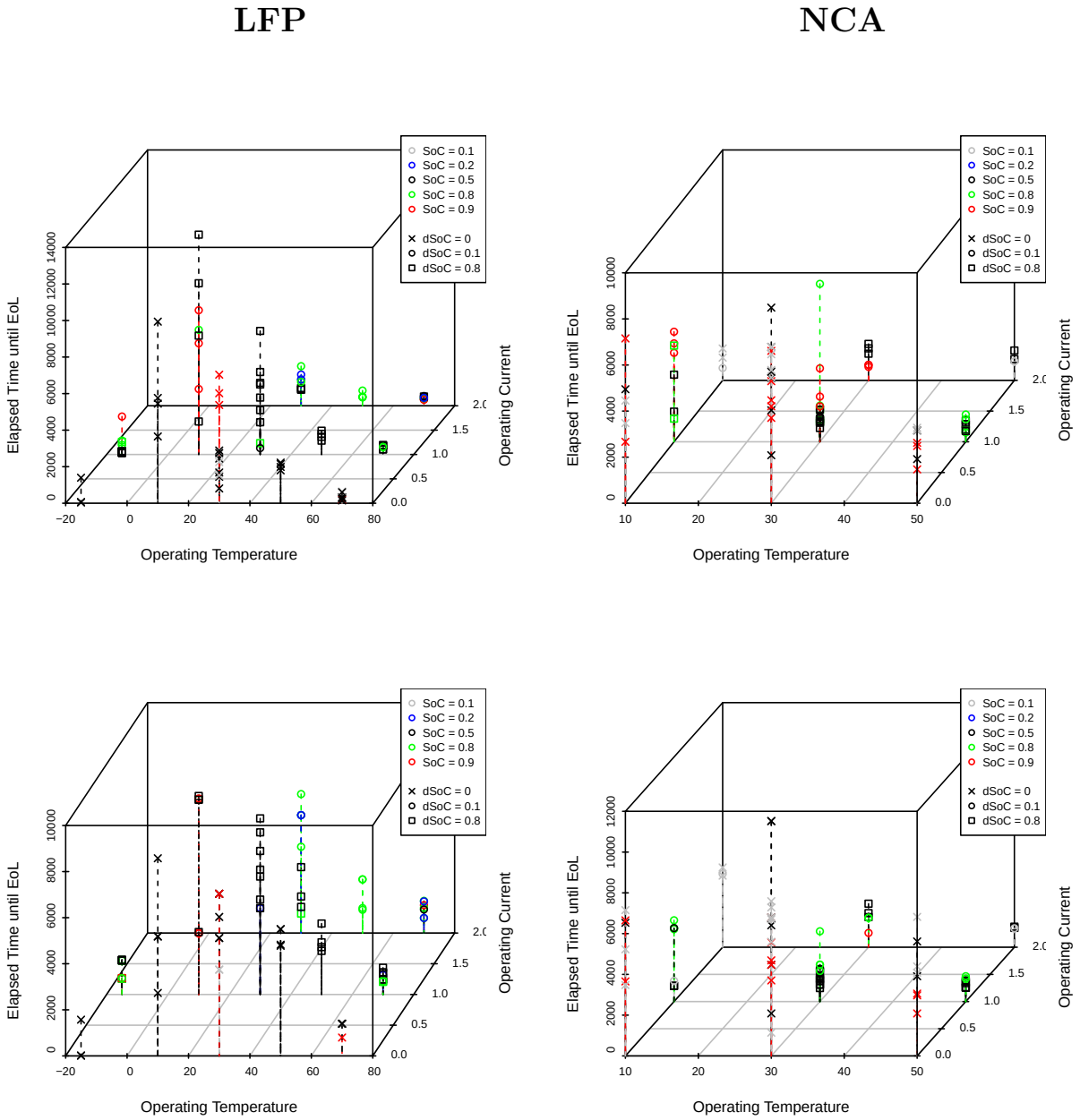

Figure 8.11: 3D-scatter plots showing the **factor interactions** for the capacity criterion on the top and the resistance criterion on the bottom, LFP cells on the left and NCA cells on the right.

LFP plots. This can easily be resolved when confirming that no cell aged at $-15°C$ lived longer than roughly 2000 hours and that therefore these points have to belong to the other setting.

Even though these plots look clearer than the ones before, there is little more information about the data to be inferred from them. That the capacity decline for LFP cells is slowest when aged at $10°C$ has already been apparent from other plots. The resistance increase, however, seems to be less prone to the temperature as most cells aged at $30°C$ have high values as well. Note how those cells aged at the high current of $2C$ lower the average which is why this has not been obvious from the respective boxplot. Also be aware of the fact that for $10°C$ no cells were aged at the high current.

8.2 Experimental Design Settings

We also should investigate how the different settings from the underlying experimental design influence the cells' life time. For this purpose we intend to plot so-called error bar plots depicting for each pair of settings (henceforth denoted as 'DoE-point') the mean of the cells run at this particular calibration as a dot and the respective standard deviation as a bar extending to both sides of the mean value. These error bar plots will be given for the two cell chemistries separately.

Since it is to be expected that there are several cells that can be considered an outlier, we include in each of the plots a second error bar in a different color depicting the means and standard deviations for each DoE-point once for each group the cell with the greatest deviation to the group mean is removed. Such cells can be found using the function `max_div` which is explained in the appendix dedicated to plot code.

Note, however, that for such few cells per DoE-point these red bars have to be taken with care. Removing one of two or three observations can yield arbitrary results. These plots were simply intended to show that one outlier can greatly influence the mean and standard deviation of the whole group.

For figure 8.12 we also create new 'Doe-ID' vectors, this time containing all possible combinations of the grouped actual settings for each of the responses separately. Table 8.1 states those IDs together with the counts of cells aged at those conditions.

Note that several of these aging conditions have only one cell to them. Of course, we could make an attempt at getting rid of that particular DoE-point by simply reallocating that single cell to a 'neighbor'. However, we find that the actual settings do not matter that much and that it is not worth the effort.

It is clear that the means (and mostly the standard deviations as well) are highest for DoE-points containing favorable temperature settings. This stands to reason as for many other DoE-point no long-living cells exist at all, which of course must result in smaller variation among the cells of the group.

	Levels				Counts			
#	**Temp**	**Curr**	**SoC**	**dSoC**	**LFP** (Cap)	**NCA** (Cap)	**LFP** (Res)	**NCA** (Res)
1	-15	0	0.5	0.0	3	—	3	—
2	-15	1	0.5	0.8	12	—	8	—
3	-15	1	0.8	0.1	2	—	3	—
4	-15	1	0.8	0.8	1	—	3	—
5	-15	1	0.9	0.1	1	—	1	—
6	-15	1	0.9	0.8	—	—	1	—
7	10	0	0.1	0.0	—	3	—	3
8	10	0	0.5	0.0	4	1	4	1
9	10	0	0.9	0.0	—	2	—	2
10	10	1	0.1	0.1	—	—	—	1
11	10	1	0.5	0.1	—	—	—	1
12	10	1	0.5	0.8	4	2	4	2
13	10	1	0.8	0.1	1	—	1	3
14	10	1	0.8	0.8	—	2	—	—
15	10	1	0.9	0.1	3	3	3	—
16	10	2	0.1	0.0	—	2	—	2
17	10	2	0.1	0.1	—	1	—	1
18	30	0	0.1	0.0	3	6	3	7
19	30	0	0.5	0.0	6	6	5	6
20	30	0	0.9	0.0	3	5	2	5
21	30	1	0.2	0.1	—	—	1	—
22	30	1	0.5	0.1	1	3	—	3
23	30	1	0.5	0.8	7	6	7	9
24	30	1	0.8	0.1	—	2	—	4
25	30	1	0.8	0.8	1	3	—	—
26	30	1	0.9	0.1	—	3	—	—
27	30	2	0.2	0.1	2	—	2	—
28	30	2	0.5	0.8	3	3	3	3
29	30	2	0.8	0.1	3	—	3	2
30	30	2	0.8	0.8	—	—	7	—
31	30	2	0.9	0.1	—	3	—	1
32	50	0	0.1	0.0	—	3	—	3
33	50	0	0.5	0.0	4	2	4	2
34	50	0	0.9	0.0	—	3	—	3
35	50	1	0.5	0.8	4	5	4	6
36	50	1	0.8	0.1	—	3	—	3
37	50	1	0.8	0.8	—	1	—	—
38	50	2	0.1	0.1	—	1	—	1
39	50	2	0.5	0.8	—	2	—	2
40	50	2	0.8	0.1	4	—	4	—
41	70	0	0.1	0.0	1	—	1	—
42	70	0	0.5	0.0	4	—	5	—
43	70	0	0.9	0.0	1	—	1	—
44	70	1	0.2	0.1	—	—	1	—
45	70	1	0.5	0.8	6	—	5	—
46	70	1	0.8	0.1	2	—	—	—
47	70	1	0.8	0.8	—	—	3	—
48	70	2	0.2	0.1	2	—	2	—
49	70	2	0.5	0.1	1	—	1	—
50	70	2	0.5	0.8	1	—	—	—
51	70	2	0.8	0.1	2	—	3	—
52	70	2	0.9	0.1	2	—	1	—

Table 8.1: An overview over the actually achieved factor settings.

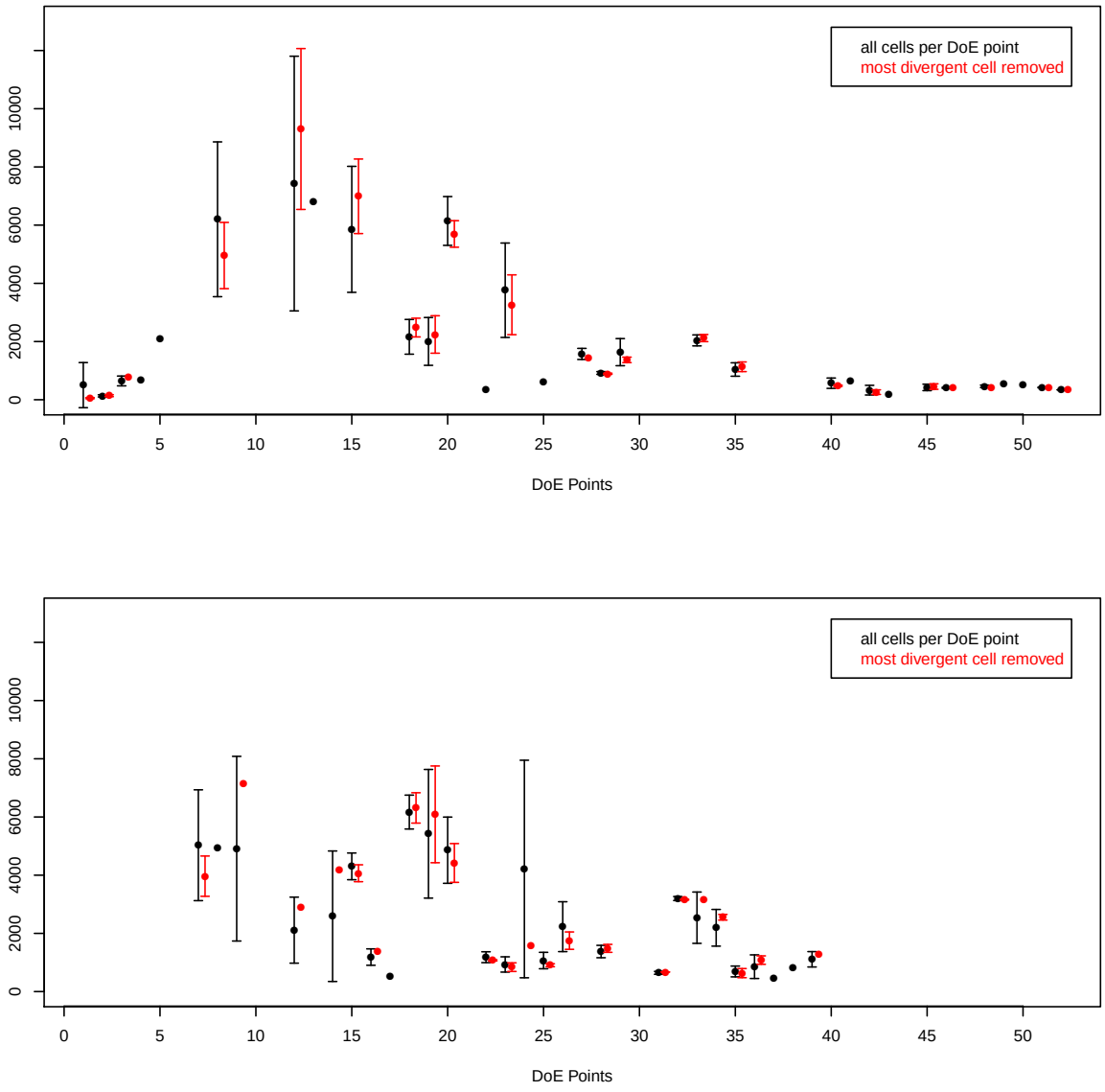

Figure 8.12: Error bar plots showing the **spread among cells** aged at roughly the same conditions for the **capacity** criterion. The top plot shows LFP cells and the bottom plot NCA cells. The DoE-points are in ascending order for the factor `Temp` but the precise factor settings corresponding to the numbers in the plot can be found in table 8.1.

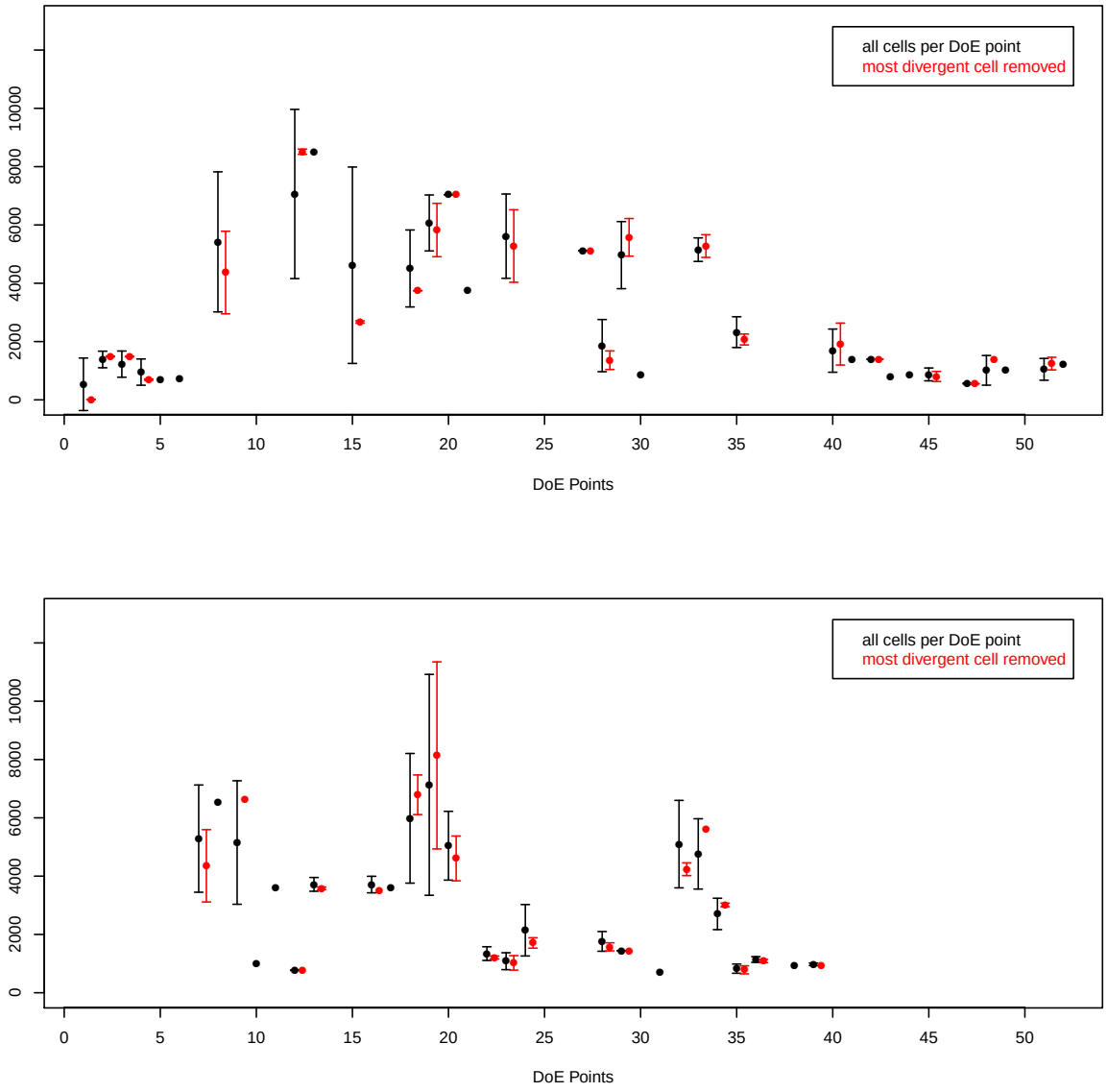

Figure 8.13: Error bar plots showing the **spread among cells** aged at roughly the same conditions for the **resistance** criterion. The top plot shows LFP cells and the bottom plot NCA cells. The DoE-points are in ascending order for the factor `Temp` but the precise factor settings corresponding to the numbers in the plot can be found in table 8.1.

The cells for which the EoL-determination does not work correctly have been removed from this plot.

8.3 Summary of Results

- The separate considerations of the two cell chemistries is necessary.
- The factor `Leak` has a an obvious influence on the life spans and should at least be considered in the modeling process.
- Extremal temperatures have a great negative influence on the life time of LFP cells.
- Most long-living NCA cells were storage cells.
- Higher temperatures (of around $30° - 50°$ C) seem to trigger capacity loss faster than resistance increase.

Chapter 9

Analyzing the Design of Experiment

As mentioned before, the underlying data was obtained using a D-optimal computer-generated design which then could not be adhered to exactly.

Generally, it is favorable to work with standard designs such as (fractional) factorial or central composite designs as for these classes the analysis is mathematically sound. The most noticeable advantages of such designs are orthogonality of the contrasts, meaning that no effects are aliased with each other except for fractional factorial designs in which case the alias structure is at least known from the beginning and in parts chosen by the experimenter though.

Another very important issue the experimenter should keep in mind is that of balancedness of the conducted experiment, meaning that each treatment or factor setting combination should be conducted an equal amount of times. It is clear that due to unexpected events losses can introduce some unbalancedness into the experiment, however, the initial experiment should at least be attempted to be balanced.

In our case several aspects need to be pointed out which are by no means desirable for the professional analysis of the resulting data:

- The probably not unavoidable use of a non-standard design has already been mentioned and is thus not commented on further here.
- After the construction of a D-optimal design for a full quadratic model in the four factors of interest the design was unevenly split into two designs, one for each of the two different chemistries in use, although each of those was expected to serve yet again the full quadratic model. It must be understood that the efficiency for each of the resulting sub-designs can be much worse than that of the original complete design.

  Of course, these issues are rendered mute by the fact that the experiment originally designed has never actually been conducted.
- The number of replications for each treatment combination is by no means balanced, not even when considering the two sub-designs separately. For both LFP and NCA cells the replications vary from 1 to 7. It needs to be pointed out at this point

that one replication does not even allow to test for errors. However, it also needs to be emphasized that some of the unreplicated runs resulted from the hard-to-control factor settings.

- The unbalancedness introduces one more grievous problem for the analysis, namely that the effects are no longer orthogonal even if they started out to be in the unreplicated design. Note that for unorthogonal designs the sums of squares of the terms do not sum up to the total sum of squares.

As pointed out in the theoretical part, it is customary to standardize the variables to lie in the interval $[-1, 1]$, as will also be done in the chapters on the construction of a design.

A separate coding for the two chemistries is not only possible but also necessary since the levels were very differently chosen for the chemistries as they are dealt with independently. We shall give a table for each one stating the respective natural and coded values.

LFP		
	natural	coded
`Temp`	$-15, 10, 30, 50, 70$	$-1.00, -0.41, 0.06, 0.53, 1.00$
`Curr`	$0, 1, 5$	$-1.0, -0.6, 1.0$
`SoC`	$0.1, 0.2, 0.5, 0.8, 0.9$	$-1.00, -0.75, 0.00, 0.75, 1.00$
`dSoC`	$0, 0.1, 0.8$	$-1.00, -0.75, 1.00$

NCA		
	natural	coded
`Temp`	$10, 30, 50$	$-1, 0, 1$
`Curr`	$0, 1, 2$	$-1, 0, 1$
`SoC`	$0.1, 0.2, 0.5, 0.8, 0.9$	$-1.00, -0.75, 0.00, 0.75, 1.00$
`dSoC`	$0, 0.1, 0.8$	$-1.00, -0.75, 1.00$

Note that for simplicity we use the coded values of the settings initially introduced. Alternatively, we could recode the temperature such that the values are 'nicer' by simply redefining $-15°C$ to be $-10°C$ instead, which would result in the coded values $-1, -0.5, 0, 0.5$ and 1.

Note that for this analysis the explanatory variables `Temp`, `Curr`, `SoC` and `dSoC` need to be specified as factors, whereas the response `Time` must remain in its original numerical form.

Throughout this section we are making an attempt at analyzing the design of experiment. To show the problems with unbalanced designs we will use the original design (intended experiment) rather than the actual experiment. In the following chapter on modeling we will then work with the settings actually achieved.

Since the results might not be very useful, we will only use one chemistry and one response variable (we choose LFP and capacity) throughout. Of course, given the according data,

an analogous investigation could be conducted for the second chemistry and response as well.

9.1 An Attempt to Analyze the Unbalanced Design

In the following section we shall outline how a design is usually analyzed by `R` and what changes are necessary in the underlying situation.

Since the effects of the factors are immediately related to the respective sums of squares, an analysis of variance is carried out to determine the significant and unimportant influential factors and interactions. In `R` this is usually done with the function `aov`, a wrapper function for `lm`, however, we need to be very careful in our case. By default, `aov` uses type I sums of squares which are explained to more detail in the theoretical part of this thesis. For unbalanced data, type I sums of squares cannot be used or rather they cannot be interpreted as the order of the factors in the model under consideration plays an important role, which is not usually an intended effect.

It is possible, with function `Anova` (mind the capital 'A') from package `car`, to analyze unbalanced designs by means of type II or III sums of squares, however, it must be kept in mind that the associated hypothesis tests are not the ones usually applied. Furthermore, type III sums of squares depend crucially on the contrasts in use, where polynomial contrasts are the most practical perhaps. Also, it has to be pointed out that the results for type III are no longer easily interpretable if the terms are aliased. Although it is possible to carry out such an analysis of variance for aliased coefficients by explicitly setting option `singular.ok=TRUE` in `Anova`, this is not recommended. By default, a model containing aliased coefficients will produce an error for the type III analysis as will be demonstrated shortly.

However, it has already been noted in the theoretical part that the use of type III sums of squares is not suggested for designs with missing cells, which our D-optimal design clearly does.

The call `replications` can give an overview over the distribution of the factors' level settings in either a data frame or a given model formula. However, if there are many terms included a very comprising list of all possible combinations and their counts is provided which might be rather confusing. In our case, the quickest way to get the numbers of replications for each of the treatment combinations is probably the `table`-command. Although zeros are included in the table it is still very clear.

Linear Model with Interactions

We shall now illustrate how type I analysis of variance is dependent on the order of the factors by example of the full linear model with 2-way interactions. The data frames used for this purpose contain the coded values of the four factors `Temp`, `Curr`, `SoC` and `dSoC` respectively, specified as factors. Also note that type III analysis includes an intercept term whereas type II analysis does not. According to John Fox, the maintainer of the `car` package, this is because *'The computational approach taken in Anova() makes it simpler*

to include the intercept in the "type-III" tests and not to include it in the "type-II" tests.'

The following is a part of the output of the call to `summary` for a model fitted with `aov` and is intended to point out the different results obtained for different ordering of the factors.

```
             Df    Sum Sq  Mean Sq F value  Pr(>F)
Temp          4 387246667 96811667  60.658 < 2e-16 ***
Curr          2  14147144  7073572   4.432 0.01525 *
SoC           4  28371677  7092919   4.444 0.00286 **
Temp:Curr     8  26405237  3300655   2.068 0.04999 *
Temp:SoC      6  12325516  2054253   1.287 0.27390
Residuals    73 116510645  1596036


             Df    Sum Sq  Mean Sq F value   Pr(>F)
Curr          2  43168599 21584299  13.524 1.01e-05 ***
Temp          4 358225213 89556303  56.112  < 2e-16 ***
SoC           4  28371677  7092919   4.444  0.00286 **
Curr:Temp     8  26405237  3300655   2.068  0.04999 *
Temp:SoC      6  12325516  2054253   1.287  0.27390
Residuals    73 116510645  1596036
```

Note how the sums of squares and the significance values for the factors 'Temp' and 'Curr' have changed. Furthermore, the model we put into `aov` consisted of four linear and six interaction terms but only five terms in total are shown in the output of the `summary`-command as all other ones are indicated to have zero degrees of freedom. We assume that this is due to the fact that the other terms are aliased. This can generally be observed by the alias structure given by `alias`, however, the output of this function is not nicely readable. Unfortunately, a better suited alias function is only available for 2-level fractional factorial designs.

We now proceed to demonstrate the use of type II and III analyses by means of `Anova` from package 'car'. Here, the contrasts should be reset using the following command

```
options(contrasts=c("contr.sum", "contr.poly")).
```

```
> Anova(int.modLFP3)

Anova Table (Type II tests)

Response: Time
             Sum Sq  Df F value    Pr(>F)
Temp      338626216   5 35.7810 < 2.2e-16 ***
Curr       40367977   3  7.1092 0.0001770 ***
SoC        52049476   4  6.8748 4.442e-05 ***
dSoC              0
Temp:Curr  76860521  11  3.6916 0.0001271 ***
Temp:SoC   63186716  12  2.7819 0.0019927 **
Temp:dSoC         0
Curr:SoC          0
Curr:dSoC         0
SoC:dSoC          0
Residuals 263094865 139
```

```
> Anova(int.modLFP3, type=c("III"), singular.ok=TRUE)

Anova Table (Type III tests)

Response: Time
              Sum Sq  Df F value    Pr(>F)
(Intercept)  52686957   1 27.8359 4.951e-07 ***
Temp         56432611   5  5.9630 4.992e-05 ***
Curr         10733980   3  1.8903 0.1340308
SoC           8062489   4  1.0649 0.3763262
dSoC                    0
Temp:Curr    76860521  11  3.6916 0.0001271 ***
Temp:SoC     63186716  12  2.7819 0.0019927 **
Temp:dSoC                0
Curr:SoC                 0
Curr:dSoC                0
SoC:dSoC                 0
Residuals   263094865 139
```

The demonstration of the different types of sums of squares was intended purely to show that the results for unbalanced data do vary greatly and that it is crucial to be aware of which hypotheses have to be tested.

We now show an alternative way of how to receive estimates of the effects for the factors considered in the design. This approach uses the `lm` function directly and calls it with a model formula including the coded numerical values of the explanatory variables. The coded numerical values could also be used to check the non-orthogonality of the factors by a simple correlation test via `cor` if the levels were all evenly spaced and centered around 0, which they are not.

Note that by using coded variables the coefficients would directly correspond to the effects of the factors if they were orthogonal and their magnitude could thus be read from the summary immediately.

The following is the output of a call to `summary` for a model fitted with `lm`.

```
lm(formula = Time ~ (Temp + Curr + SoC + dSoC)^2, data = cod.datLFP)

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1887.35     243.25   7.759 1.44e-11 ***
Temp         -623.33     338.57  -1.841  0.06898 .
Curr         -951.41     288.16  -3.302  0.00139 **
SoC          1346.36     513.97   2.620  0.01037 *
dSoC           38.68     266.54   0.145  0.88494
Temp:Curr     292.40     354.84   0.824  0.41215
Temp:SoC    -1060.72     689.02  -1.539  0.12728
Temp:dSoC      54.25     366.83   0.148  0.88278
Curr:SoC     -402.72     510.66  -0.789  0.43246
Curr:dSoC    -435.03     320.37  -1.358  0.17797
SoC:dSoC          NA         NA      NA       NA

Residual standard error: 2311 on 88 degrees of freedom
Multiple R-squared: 0.1965,     Adjusted R-squared: 0.1143
F-statistic: 2.391 on 9 and 88 DF,  p-value: 0.01796
```

Here, we can see that only three factors, namely `Curr`, `SoC` and `Temp`, are significant and none of the interactions seem to matter. This might strike us as rather odd since there were two interactions that were deemed significant by all calls to `aov`. Furthermore, the last interaction cannot be estimated.

As can already be seen from the poor adjusted R^2-value, no satisfying fit is to be expected. One is unavoidably stricken by the fact that the fit is especially poor for fitted values higher than 2000 hours. We shall try to enhance this shortly.

Note that we will not even take into account the possibility of a data transformation in this chapter as we find this approach of analysis hopeless. This will be done in the upcoming response modeling chapter.

Graphical Diagnostics

We shall now look at some standard analyzing plots including the well-known QQ- and residual plots as well as effect and interaction plots. The first plot in figure 9.1 (residuals versus fitted values) clearly shows a non-constant variance and the QQ-plot also shows that the tails do not match a normal distribution. The linear model with interaction is thus not adequate. The design plot in figure 9.2 shows the type/order and magnitude of the factors' effects. The horizontal line does hereby state the overall mean. It can be seen that a specific setting of `Temp` has the greatest effect on the mean of the response and that none of the factors have a linear effect. This can be seen by the fact that for none of the factors the levels appear in a purely descending or ascending order in the plot. It is therefore reasonable to investigate the higher order effects of the factors. Note that for `SoC` three of the levels seem to have no impact on the response at all, whereas the other two do. We can also see the strong negative effect of extreme temperatures on the cells' life time. This was apparent in the preceding graphical analysis of the data as well.

The next plot is a so-called factor plot and shows the means of the response variable for the given factor settings (levels) of the explanatory variables. It has to be pointed out that these plots might lead to faulty conclusions when viewed independently of the interaction plots. Even though the preceding analysis did not show any significant interactions between factors it is still prudent to take them into account, especially with such an unclear alias structure at hand.

Note that the scale of the y-axis has been chosen such that the magnitudes of the factors are also evident. Consequently, these plots are simply a different depiction of the design plot. To not draw wrong conclusions, it is also important to note that the values on the x-axis are automatically placed equidistantly. Even though it is clear that for each factor there is a distinctive favorable level setting, `Temp` and `SoC` seem to be the most crucial as small deviations from the respective optimum setting lead to horrendously shortened life times. The influence of the other two factors is only marginal.

Next, we have a look at the six interaction plots of the four factors. Here it is important to note that due to the optimal design not all factor combinations were in the experiment and thus some of the plots are incomplete.

For those plots with missing combinations, which are all but two, the pointwise means were

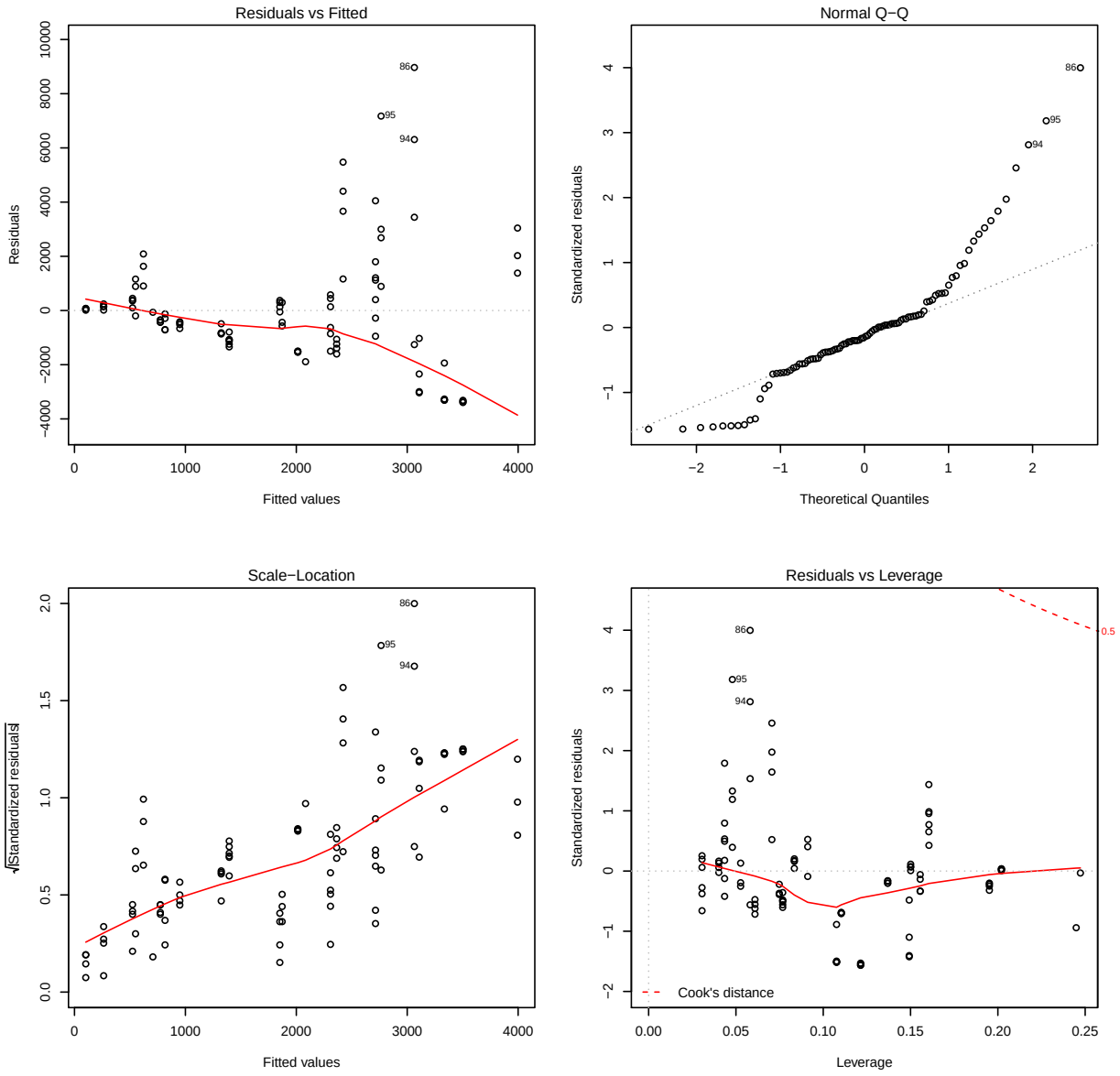

Figure 9.1: Diagnostic plots to check the model assumptions for the linear model with interactions.

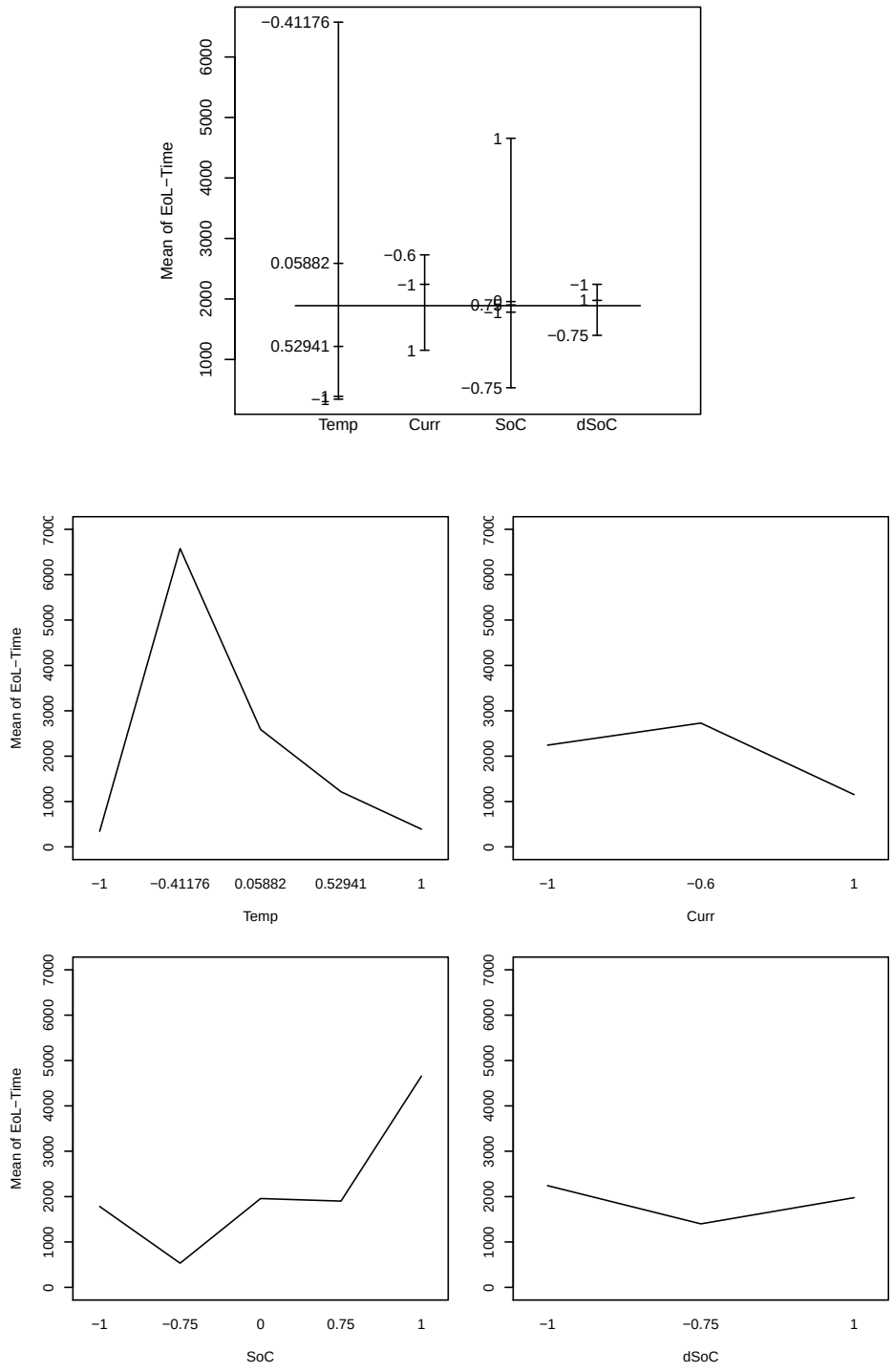

Figure 9.2: Design plot (top) and factor plots (bottom) for the capacity criterion and LFP cells.

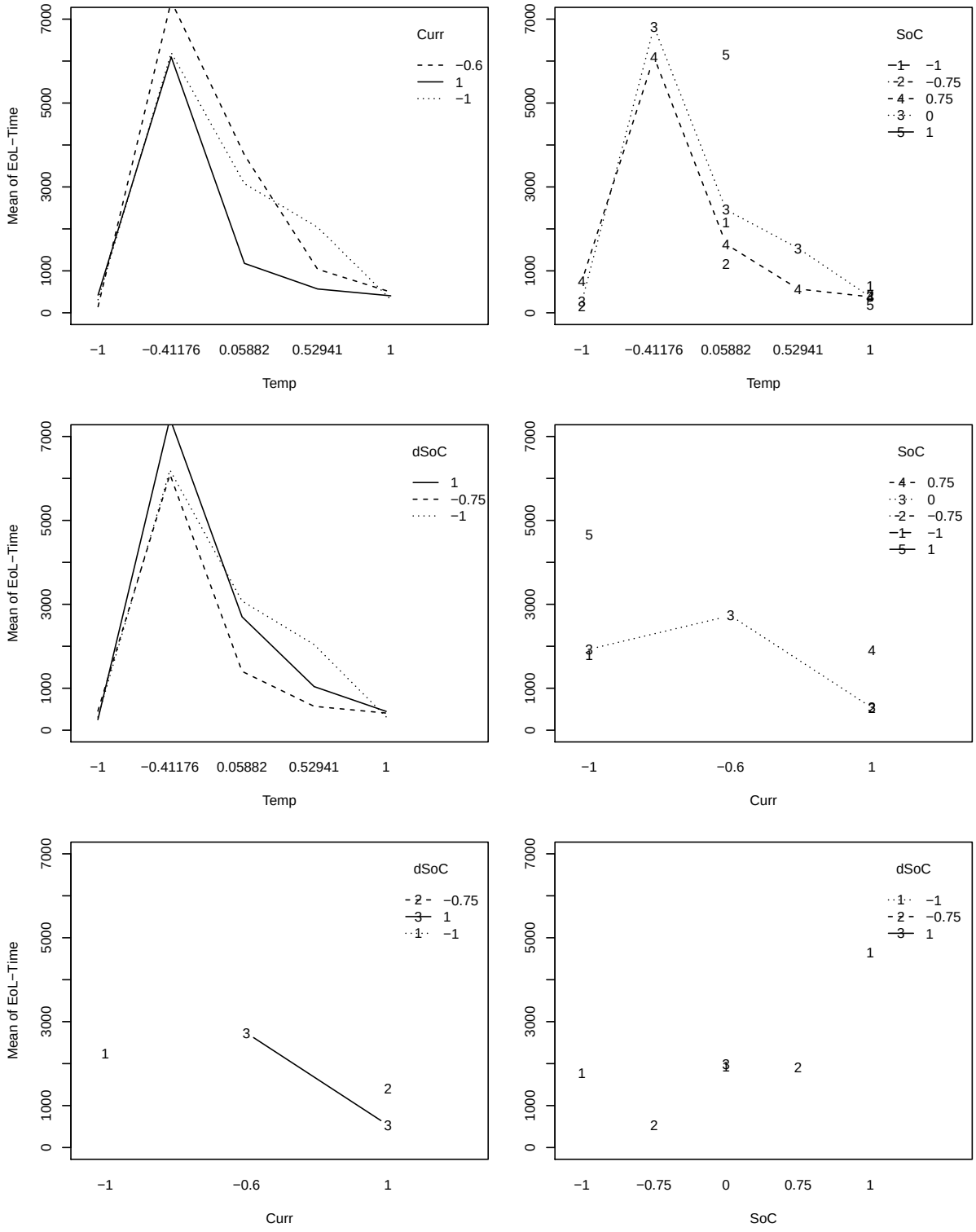

Figure 9.3: Interaction plots for the capacity criterion and LFP cells.

included, displayed by numbers. Whereas the interaction plot of `Temp` and `SoC` (top right) remains interpretable, the high number of missing combinations for all plots not containing `Temp` allows only vague guesses as to which combination of those factors should best be chosen.

`Temp` does not seem to interact with any other factor, which can be seen from the mostly parallel curves for each setting of the other factors in the respective interaction plots. When assuming that the setting for `dSoC` does not really matter, the following settings seem to yield the highest mean of the response variable.

- `Temp` $= -0.41176 \mathrel{\widehat{=}} 10°C$
- `Curr` $= -0.6 \mathrel{\widehat{=}} 1$ C
- `SoC` $= 1 \mathrel{\widehat{=}} 90\%$
  Note that here we are assuming parallel curves for the missing values in the interaction plot with `Temp`, which seems to be fairly reasonable.
- `dSoC` $= 1 \mathrel{\widehat{=}} 0.8$
  This is suggested by the interaction plot with `Temp` and essentially not contradicted by any other interaction.

However, since this exact combination was not run in the experiment the actual yield for this setting can only be estimated.

9.1.1 Polynomial Model with Interactions

For each of the factors several levels were chosen for the design allowing to test for the effects of higher order as well as their interactions. For this we now call `lm` with the coded variables in factor form and specify each of the factors as `contr.poly` indicating that polynomial contrasts are to be used. Note that this function automatically uses polynomials of degree $p-1$ for a factor with p levels. We would have only been interested in the quadratic polynomials but have so far not managed to override this behavior.

```
> poly.modLFP = lm(Time ~ (Temp+Curr+SoC+dSoC)^2, data=fact.datLFP,
                   contrasts=list(Temp='contr.poly', Curr='contr.poly',
                                  SoC='contr.poly', dSoC='contr.poly'))


lm(formula = Time ~ (Temp + Curr + SoC + dSoC)^2, data = fact.datLFP,
    contrasts = list(Temp = "contr.poly", Curr = "contr.poly",
        SoC = "contr.poly", dSoC = "contr.poly"))

Coefficients: (40 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    2624.69     830.25   3.161  0.00229 **
Temp.L        -1624.23    2618.77  -0.620  0.53704
Temp.Q        -3734.04    1645.29  -2.270  0.02619 *
Temp.C         3384.09     361.44   9.363 3.90e-14 ***
Temp^4        -1860.69     350.56  -5.308 1.15e-06 ***
Curr.L         -544.98     385.61  -1.413  0.16183
Curr.Q         -646.39     313.38  -2.063  0.04270 *
SoC.L          2513.29     956.24   2.628  0.01046 *
SoC.Q          1056.65    2345.85   0.450  0.65373
SoC.C           962.68     729.39   1.320  0.19101
SoC^4           -18.73     799.01  -0.023  0.98136
```

```
dSoC.L               NA          NA       NA        NA
dSoC.Q               NA          NA       NA        NA
Temp.L:Curr.L   -226.47      675.43   -0.335   0.73836
Temp.Q:Curr.L   1039.48      696.41    1.493   0.13984
Temp.C:Curr.L    533.63      623.19    0.856   0.39465
Temp^4:Curr.L    391.60      630.47    0.621   0.53645
Temp.L:Curr.Q    249.15      646.59    0.385   0.70111
Temp.Q:Curr.Q   1398.47      628.57    2.225   0.02918 *
Temp.C:Curr.Q   -940.26      628.86   -1.495   0.13918
Temp^4:Curr.Q   -719.65      582.99   -1.234   0.22100
Temp.L:SoC.L   -4207.03     3194.99   -1.317   0.19204
Temp.Q:SoC.L    -286.44     1788.96   -0.160   0.87323
Temp.C:SoC.L         NA          NA       NA        NA
Temp^4:SoC.L         NA          NA       NA        NA
Temp.L:SoC.Q      40.26     6989.63    0.006   0.99542
Temp.Q:SoC.Q   -1764.29     3436.91   -0.513   0.60927
Temp.C:SoC.Q         NA          NA       NA        NA
Temp^4:SoC.Q         NA          NA       NA        NA
Temp.L:SoC.C   -1691.64     1710.57   -0.989   0.32596
Temp.Q:SoC.C         NA          NA       NA        NA
Temp.C:SoC.C         NA          NA       NA        NA
Temp^4:SoC.C         NA          NA       NA        NA
Temp.L:SoC^4      68.86     1765.24    0.039   0.96899
Temp.Q:SoC^4         NA          NA       NA        NA
Temp.C:SoC^4         NA          NA       NA        NA
Temp^4:SoC^4         NA          NA       NA        NA
Temp.L:dSoC.L        NA          NA       NA        NA
Temp.Q:dSoC.L        NA          NA       NA        NA
Temp.C:dSoC.L        NA          NA       NA        NA
Temp^4:dSoC.L        NA          NA       NA        NA
Temp.L:dSoC.Q        NA          NA       NA        NA
Temp.Q:dSoC.Q        NA          NA       NA        NA
Temp.C:dSoC.Q        NA          NA       NA        NA
Temp^4:dSoC.Q        NA          NA       NA        NA
Curr.L:SoC.L         NA          NA       NA        NA
Curr.Q:SoC.L         NA          NA       NA        NA
Curr.L:SoC.Q         NA          NA       NA        NA
Curr.Q:SoC.Q         NA          NA       NA        NA
Curr.L:SoC.C         NA          NA       NA        NA
Curr.Q:SoC.C         NA          NA       NA        NA
Curr.L:SoC^4         NA          NA       NA        NA
Curr.Q:SoC^4         NA          NA       NA        NA
Curr.L:dSoC.L        NA          NA       NA        NA
Curr.Q:dSoC.L        NA          NA       NA        NA
Curr.L:dSoC.Q        NA          NA       NA        NA
Curr.Q:dSoC.Q        NA          NA       NA        NA
SoC.L:dSoC.L         NA          NA       NA        NA
SoC.Q:dSoC.L         NA          NA       NA        NA
SoC.C:dSoC.L         NA          NA       NA        NA
SoC^4:dSoC.L         NA          NA       NA        NA
SoC.L:dSoC.Q         NA          NA       NA        NA
SoC.Q:dSoC.Q         NA          NA       NA        NA
SoC.C:dSoC.Q         NA          NA       NA        NA
SoC^4:dSoC.Q         NA          NA       NA        NA

Residual standard error: 1263 on 73 degrees of freedom
Multiple R-squared: 0.8008,     Adjusted R-squared: 0.7354
F-statistic: 12.23 on 24 and 73 DF,  p-value: < 2.2e-16
```

At first sight the resulting summary table contains many NAs. However, it also suggests that the influence of the factor `Temp` is not linear but rather of a higher order, where all up to the fourth degree appear to be highly significant. For `Curr` a quadratic influence is suggested, whereas for `SoC` the linear effect is strongest. Note that factor `dSoC` is aliased and is thus not estimable. However, the design plot shows a rather small effect of this

factor and suggests it is of a higher order. Furthermore, note the significant interaction of the quadratic parts of `Temp` and `Curr`.

Since it seems odd that such high orders of the factors (especially for `Temp`) are significant and also because of the model initially put into the algorithm computing the D-optimal design (which was the full quadratic model including 2-way interactions in the four factors), we cannot completely trust this analysis. It could well be that the significant effects do in fact correspond to some aliased effect but since the alias structure is vast and incomprehensible we have no means of isolating them. We could try to include the suggested terms in a linear model when attempting to explain the response in a following chapter.

For now, from the (maybe only partly conclusive) analysis of the experimental design, we conclude that for the type of design at hand and given its unequal replications it is probably a good idea to consider the insight gained from such devices as factor and interaction plots but the other analysis is best taken with a pinch of salt.

9.2 An Attempt to Analyze the Balanced Design

As an initial remark, note that the factors in this design are also not orthogonal which implies that effects are still aliased with each other. One major advantage over the previous analysis should, however, be that the use of atypical types of sums of squares can be avoided.

First, let us decide how to balance the data, that is which observations to exclude. The command `table` states how many cells were run at a particular setting. These numbers range from 1 to 7, but for LFP cells luckily there were at least 3 cells run at all except two DoE-points. We shall therefore use this as the amount of replications in our experiment.

It seems reasonable to exclude all malfunctioning cells from the balanced design. Furthermore, we choose to discard all cells that deviate from the respective group median the most.

With the resulting adapted data we repeated the analysis from the last section.

The results are mainly in agreement with those found for the unbalanced design. For a quick graphical means of comparison one could consult the (omitted) design and factor plots. The basic form of both plots is the same as for the unbalanced data, however, the exact values have changed slightly due to less variation in the data which was achieved by removing those cells with greatest deviance to the median.

Since the results here are not very different from those of the unbalanced design, we conclude that the insight gained from this section is conform with that of the previous one.

As a final remark, contour and response surface plots are generally a very insightful method of data display for designed experiments. However, since they require the input of a specific model and we simply cannot show the validity of our results as of yet it is best we postpone the use of such devices until the next chapter. Looking at contour plots derived from highly aliased data and badly fitting models might just lead us to draw wrong conclusions.

Chapter 10

Modeling the Cells' Life Spans

In this chapter we shall use several techniques to find linear models for the two responses of interest. These approaches are the use of the function `step`, a closer investigation of factor and interaction plots as well as the experimenter's insight into the matter. As already mentioned in the `R` part, linear models and thus the use of the function `lm` are completely sufficient for our purpose.

In any case, the main statistic upon which the selection is based is the adjusted R^2-value which is part of the output from `R`'s `summary` command. Note that `step`, however, internally uses the AIC as a selection criterion.

Once a few (more or less) satisfactory models have been chosen, these are going to be analyzed by means of a graphical residual analysis.

10.1 Finding Models

Beforehand, the practitioner already had a hunch about the influence of each factor when viewed independently. This inkling stems from the screening phase done by a literature study.

We are of course interested in whether we can see the respective behavior in the resulting models later on.

10.1.1 Using `step()`

We used the function `step` to obtain a first selection of possibly significant terms mainly for a very comprehensive model, such as a full quadratic model in all the factors in the design and `Leak`, and then 'cleaned' the result by sequentially removing insignificant terms (according to the respective t-test) until the resulting model contained only significant terms. Note that the order in which terms are removed influences the results since the t-test is sensitive to other terms in the model. We therefore chose to, as a general rule of thumb but not necessarily always, remove interaction terms first.

Following the practitioners' request we analyzed many models, including and excluding the factor `Leak` and using several functions for each of the regression variables. Omitting the details here, it turned out that no transformation on the regressors is needed, except for `Temp`. We shall therefore consider this factor either linearly or centered around $15°C$ or $20°C$ in the following, that is to say we consider

$$(\texttt{Temp} - x)^2, \quad \text{where} \quad x \in \{15, 20\}.$$

Furthermore, let us remark that it was much easier to find models with a high adjusted R^2 value for the *capacity criterion* than for the *resistance* analogon. For the latter it was not clear whether or not a transformation might be helpful, as the results varied remarkably. However, the logarithmic transformation on the response is definitely a good idea when modeling the capacity EoL-criterion.

We shall now state a few models that were found by means of `step`. The resulting models for the capacity criterion will in the following be referred to as the '**complex**' models for the respective chemistry. Note that in the following the models are not nested.

For the resistance criterion we consider only one model per chemistry in total, as no other satisfying models have been found. One of these has been found by `step` and is thus stated here. None of these models, however, uses a transformation on the response.

```
# complex model for capacity criterion, LFP cells
  lm(formula = log(Cap80Time) ~ I((Temp - 20)^2) + Temp + Curr +
      I(((Temp - 20)^2)^2) + I(SoC^2) + I((Temp - 20)^2):Temp +
      I((Temp - 20)^2):Curr + Temp:Curr, data = Cap80.LFPclean)

                          Estimate Std. Error t value Pr(>|t|)
  (Intercept)             8.474e+00  3.181e-01  26.640  < 2e-16 ***
  I((Temp - 20)^2)       -2.159e-03  3.152e-04  -6.850 1.04e-09 ***
  Temp                   -2.151e-02  1.156e-02  -1.861 0.066103 .
  Curr                    9.098e-01  3.699e-01   2.459 0.015921 *
  I(((Temp - 20)^2)^2)   -7.195e-07  2.371e-07  -3.034 0.003191 **
  I(SoC^2)                6.427e-01  3.066e-01   2.096 0.039011 *
  I((Temp - 20)^2):Temp   4.965e-05  9.319e-06   5.328 7.87e-07 ***
  I((Temp - 20)^2):Curr   1.024e-03  2.356e-04   4.344 3.81e-05 ***
  Temp:Curr              -4.860e-02  1.229e-02  -3.956 0.000156 ***

  Residual standard error: 0.5829 on 86 degrees of freedom
  Multiple R-squared: 0.8254,   Adjusted R-squared: 0.8092
  F-statistic: 50.82 on 8 and 86 DF,  p-value: < 2.2e-16


# complex model for capacity criterion, NCA cells
  lm(formula = log(Cap80Time) ~ I((Temp)^2) + Curr + Leak + Temp:Curr +
      SoC:dSoC + Curr:Leak, data = Cap80.NCAclean)

                 Estimate Std. Error t value Pr(>|t|)
  (Intercept)   8.900e+00  1.127e-01  78.992  < 2e-16 ***
  I((Temp)^2)  -4.488e-04  6.854e-05  -6.548 9.69e-09 ***
  Curr         -1.290e+00  1.979e-01  -6.519 1.09e-08 ***
  Leak0.5      -2.959e-01  1.518e-01  -1.949 0.055467 .
  Leak1        -2.347e-01  1.969e-01  -1.192 0.237471
  Curr:Temp     1.994e-02  5.119e-03   3.896 0.000228 ***
  SoC:dSoC     -2.305e+00  3.044e-01  -7.572 1.42e-10 ***
  Curr:Leak0.5  3.843e-01  1.676e-01   2.293 0.024971 *
  Curr:Leak1   -5.468e-01  3.589e-01  -1.524 0.132317
```

```
  Residual standard error: 0.4303 on 67 degrees of freedom
  Multiple R-squared: 0.776,    Adjusted R-squared: 0.7493
  F-statistic: 29.02 on 8 and 67 DF,  p-value: < 2.2e-16


# model for resistance criterion, LFP cells
  lm(formula = Res300Time ~ I((Temp - 20)^2) + dSoC + Leak + I((Temp - 20)^4) +
      I(dSoC^2) + Curr:dSoC, data = Res300.LFPclean)

                     Estimate Std. Error t value Pr(>|t|)
  (Intercept)       7.084e+03  3.295e+02  21.498  < 2e-16 ***
  I((Temp - 20)^2) -5.193e+00  5.826e-01  -8.914 8.07e-14 ***
  dSoC             -4.325e+03  2.556e+03  -1.692  0.09431 .
  Leak0.5          -1.410e+03  3.349e+02  -4.209 6.33e-05 ***
  Leak1            -1.524e+03  4.021e+02  -3.791  0.00028 ***
  I((Temp - 20)^4)  1.361e-03  2.091e-04   6.511 4.97e-09 ***
  I(dSoC^2)         5.546e+03  2.842e+03   1.951  0.05431 .
  dSoC:Curr        -1.627e+03  5.701e+02  -2.855  0.00541 **

  Residual standard error: 1315 on 85 degrees of freedom
  Multiple R-squared: 0.7454,   Adjusted R-squared: 0.7244
  F-statistic: 35.55 on 7 and 85 DF,  p-value: < 2.2e-16
```

The resistance model includes the factor `Leak`, which is the only categorical factor under consideration. Other than for quantitative (or continuous) factors, this factor merely changes the intercept according to the respective group.

For this model the adjusted R^2 value is surprisingly high given the difficulties finding any sensible model for this response at all.

Note that the LFP capacity model has a higher complexity than the NCA model. This is due to the fact that the inevitable influence of negative and very high temperatures has to be accounted for with the first chemistry. Nonetheless, the adjusted R^2 value for both models is quite high. We will, of course, have to analyze these models later on since one numerical value is no apt decision criterion.

Be aware of that although the term for `Leak=1` in the NCA model seems insignificant, it is impossible to remove this term from the model by itself. A qualitative factor is either included for all of its levels or for none at all.

10.1.2 Using Factor & Interaction Plots

Factor and interaction plots can be a useful tool to find an adequate model for the underlying data. However, the plots produced in the last chapter do not correspond to the measured data and factor settings but rather to those intended by the design. We therefore consider another set of factor and interaction plots, depicting the actual data, for this section.

Note that the interactions of the factor `Leak` with the other factors have not been investigated by means of interaction plots. Such terms are therefore not likely to appear in the upcoming models.

The following two models, both for the capacity criterion, have been found by reading the importance and type of influence of each factor from the corresponding factor and

interaction plots. However, not all terms that seem significant in the plots really resulted in an increase of the adjusted R^2 value, so some playing around with the terms was necessary. These models will be referred to as '**medium**' model for the capacity criterion for the respective chemistry.

```
# medium model for capacity criterion, LFP cells
  lm(formula = log(Cap80Time) ~ Temp + I(Temp^2) + Curr + SoC + dSoC + I(dSoC^2),
      data = Cap80.LFPclean)

               Estimate Std. Error t value Pr(>|t|)
  (Intercept)  6.3601188  0.3154966  20.159  < 2e-16 ***
  Temp         0.0855478  0.0072483  11.802  < 2e-16 ***
  I(Temp^2)   -0.0013984  0.0001053 -13.275  < 2e-16 ***
  Curr        -0.7184024  0.1683978  -4.266 4.99e-05 ***
  SoC          0.8529571  0.4918666   1.734  0.08640 .
  dSoC         4.8233545  1.6258922   2.967  0.00388 **
  I(dSoC^2)   -5.4324079  1.8366721  -2.958  0.00398 **

  Residual standard error: 0.7729 on 88 degrees of freedom
  Multiple R-squared: 0.6859,   Adjusted R-squared: 0.6645
  F-statistic: 32.03 on 6 and 88 DF,  p-value: < 2.2e-16


# medium model for capacity criterion, NCA cells
  lm(formula = log(Cap80Time) ~ Curr + I(Curr^2) + I(Temp^2) + dSoC:Curr + dSoC,
      data = Cap80.NCAclean)

               Estimate Std. Error t value Pr(>|t|)
  (Intercept)  8.835e+00  9.469e-02  93.307  < 2e-16 ***
  Curr        -2.538e+00  3.693e-01  -6.874 2.12e-09 ***
  I(Curr^2)    8.030e-01  1.782e-01   4.506 2.58e-05 ***
  I(Temp^2)   -3.821e-04  5.476e-05  -6.978 1.37e-09 ***
  dSoC        -1.206e+00  2.621e-01  -4.599 1.83e-05 ***
  Curr:dSoC    1.075e+00  2.476e-01   4.340 4.70e-05 ***

  Residual standard error: 0.398 on 70 degrees of freedom
  Multiple R-squared: 0.7999,   Adjusted R-squared: 0.7856
  F-statistic: 55.96 on 5 and 70 DF,  p-value: < 2.2e-16
```

Note that the adjusted R^2-value of the medium LFP model is lower than that for the complex model. For NCA cells the medium model has a higher adjusted R^2-value than the complex model. However, the models will be compared more thoroughly by means of a graphical assessment.

10.1.3 Using the Experimenter's Insight

As mentioned before, the experimenter already had certain expectations for this analysis. Let it not be unmentioned that many models selected by the function `step` were refused because although featuring a high adjusted R^2 value they did not make sense physically. The experimenter thus expected heavy over-fitting for these models.

The following models have been created using a mixture of other models and the experimenter's ideas as well as a small search for additional terms that are significant given the

included ones. The models for the capacity criterion will be referred to as the '**simple**' models in the evaluation process.

```
# simple model for capacity criterion, LFP cells
  lm(formula = log(Cap80Time) ~ Temp + I((Temp - 15)^2) + SoC + SoC:Curr,
      data = Cap80.LFPclean)

                     Estimate Std. Error t value Pr(>|t|)
  (Intercept)       6.3359362  0.2967166  21.353  < 2e-16 ***
  Temp              0.0409748  0.0041417   9.893 4.76e-16 ***
  I((Temp - 15)^2) -0.0013356  0.0001003 -13.309  < 2e-16 ***
  SoC               1.7308016  0.5232159   3.308 0.001352 **
  SoC:Curr         -0.7617157  0.2101079  -3.625 0.000478 ***

  Residual standard error: 0.7885 on 90 degrees of freedom
  Multiple R-squared: 0.6657,   Adjusted R-squared: 0.6508
  F-statistic:  44.8 on 4 and 90 DF,  p-value: < 2.2e-16


# simple model for capacity criterion, NCA cells
  lm(formula = log(Cap80Time) ~ I(Curr^2) + dSoC + Curr + Temp:SoC,
      data = Cap80.NCAclean)

              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  8.677052   0.116155  74.703  < 2e-16 ***
  I(Curr^2)    0.576801   0.198701   2.903 0.004923 **
  dSoC        -0.800791   0.237139  -3.377 0.001193 **
  Curr        -1.771555   0.433561  -4.086 0.000114 ***
  Temp:SoC    -0.022402   0.005099  -4.394 3.82e-05 ***

  Residual standard error: 0.4946 on 71 degrees of freedom
  Multiple R-squared: 0.6864,   Adjusted R-squared: 0.6688
  F-statistic: 38.86 on 4 and 71 DF,  p-value: < 2.2e-16


# model for resistance criterion, NCA cells
  lm(formula = Res300Time ~ Temp + Curr + dSoC + Leak + I(Temp^2) + I(Curr^2) +
      Curr:SoC, data = Res300.NCAclean)

              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  3798.873    981.188   3.872 0.000248 ***
  Temp          207.565     68.188   3.044 0.003334 **
  Curr        -6206.404   1273.081  -4.875 7.00e-06 ***
  dSoC        -1402.106    801.514  -1.749 0.084815 .
  Leak0.5     -1232.496    423.502  -2.910 0.004898 **
  Leak1       -2562.185    575.956  -4.449 3.34e-05 ***
  I(Temp^2)      -3.857      1.034  -3.730 0.000396 ***
  I(Curr^2)    2815.811    587.650   4.792 9.55e-06 ***
  Curr:SoC    -2183.753    761.589  -2.867 0.005528 **

  Residual standard error: 1476 on 67 degrees of freedom
  Multiple R-squared: 0.7047,   Adjusted R-squared: 0.6695
  F-statistic: 19.99 on 8 and 67 DF,  p-value: 4.841e-15
```

Note that the resulting NCA capacity model contains comparatively many `Curr` terms. We have thus to be careful not to go into over-fitting here as well.

Moreover, the adjusted R^2 values of the simple capacity models (for both chemistries) are lower than the ones for the medium and complex models.

10.2 Evaluating the Models

For each of the models found in the previous section, eight in total, we give some statistics that are supposed to help assess their quality. These statistics are the adjusted R^2 value, which has already been stated in the respective summaries, the AIC and corrected AIC (AICc) values, the maximum absolute value of a residual, the distance between the 95% quantile and the 5% quantile of the residuals and the maximum value according to Cook's distance.

The `R` commands `quantile` (core) and `cooks.distance` ('car') can be used to obtain the last two of these values.

Model	adj. R^2	AIC	AICc	$\max \lvert r \rvert$	$q^r_{0.95} - q^r_{0.05}$	$CD_{\max}$
Complex Cap. LFP	0.81	177.6	180.2	1.65	1.8076	0.18
Medium Cap. LFP	0.66	230.6	231.9	2.17	2.5831	0.08
Simple Cap. LFP	0.65	231.3	232.3	2.06	2.5629	0.06
Complex Cap. NCA	0.75	97.9	101.3	1.12	1.2425	0.17
Medium Cap. NCA	0.79	83.4	85.0	1.19	1.2812	0.27
Simple Cap. NCA	0.67	115.5	116.7	1.24	1.4410	0.18
Res. LFP	0.72	1609.4	1611.5	3231.10	3743.36	0.39
Res. NCA	0.67	1335.2	1338.6	4948.25	4330.38	0.24

Note that the AIC and AICc values are only useful for the capacity models, where there are alternatives to choose from.

For LFP the model choice seems clear, as the adjusted R^2-value is highest and the AIC(c) is lowest for the complex model. Based on the adjusted R^2-value alone there seem to be two plausible choices for NCA. However, the AIC(c) criterion indicates that the medium model is to be preferred over the complex one.

Furthermore, the maximum values of Cook's distance show that no outliers need to be investigated.

We also chose to use the following plots as a diagnostic means for the regression of all models from the previous section:

- The first plot depicts the **fitted** values against the **actual** values to show the quality of the fit. The points in the plot should be close to the line $x = y$.
- The second plot is a standard **normal QQ-plot** which tests the residual values for their normality. The QQ-line is also included in the plot.
- The third plot is a **residual** plot giving each residual against its index. This plot should show residuals being normally distributed around the horizontal line at a

residual of zero. Any distinct trend in the distribution shows that the residuals are not distributed normally.

- The fourth plot is a so-called **leverage** plot. The leverage of each point is a measure of how much the model coefficients might change if this point is removed. Points with a high leverage are outlying values without close neighbors, thus forcing the regression line to pass close to them.

  This plot also contains the contour lines for Cook's distance, Any point lying outside the line at distance 1 should be giving rise to concern as it might indicate a bad model. However, it is generally important to double check such points since sometimes they are simply extraordinary outliers.

These plots confirm that the complex capacity model for LFP cells seems to be an adequate model. The experimenter, however, is somewhat sceptical about this result as he fears over-fitting. We therefore include a contour plot for the resulting models (for both the capacity and the resistance criterion) to show the regions of the highest predicted response with respect to two factors. Since we did not consider higher-order interactions in the first place, this should not lead us to drawing wrong conclusions.

The diagnostic plots for the NCA capacity models are in accordance with the numerical statistics and thus the medium model seems adequate for this chemistry.

Figures 10.4 and 10.6 show contour plots for the complex LFP and NCA capacity models and the only resistance models under consideration respectively. For the capacity models, the response has been transformed to hours until EoL again by an exponentiation. Each subplot shows the response variable with respect to the change in the two factors. Factors that are not considered in the plots were held fixed at the following levels. Note that the level for `Leak` has to be declared a factor in function `predict` explicitly.

$$\texttt{Temp} = 10,\ \texttt{Curr} = 1,\ \texttt{SoC} = \texttt{dSoC} = 0.5,\ \texttt{Leak} = 0.$$

Indeed, it can be seen that the complex LFP model, although seemingly adequate, is badly over-fitting. The model suggests an increasing response with an increased current, which is physically simply wrong and is against all insight gained from previous sections. Of course, this partly stems from the fact that we consider the models in the designed factor limits rather than the actual ones. However, even at a current of $2C$ the observed response is nowhere near as high as promised by the model.

For NCA cells the diagnostic plots confirm that both the complex and the medium model could be used. Therefore, we also include the contour plots for alternative models, namely the 'medium' capacity models, in figure 10.5.

The contour plots for the medium LFP capacity model definitely fulfil the expectations better than those for the complex model. We thus conclude that the medium model is a better fit than the complex one.

The medium NCA model seems to be adequate in the given margins. Note, however, that the quadratic influence of `Curr` would also promise an increase in the response for an increased current higher than $2C$. Therefore, the experimenter feels more comfortable with the complex NCA model.

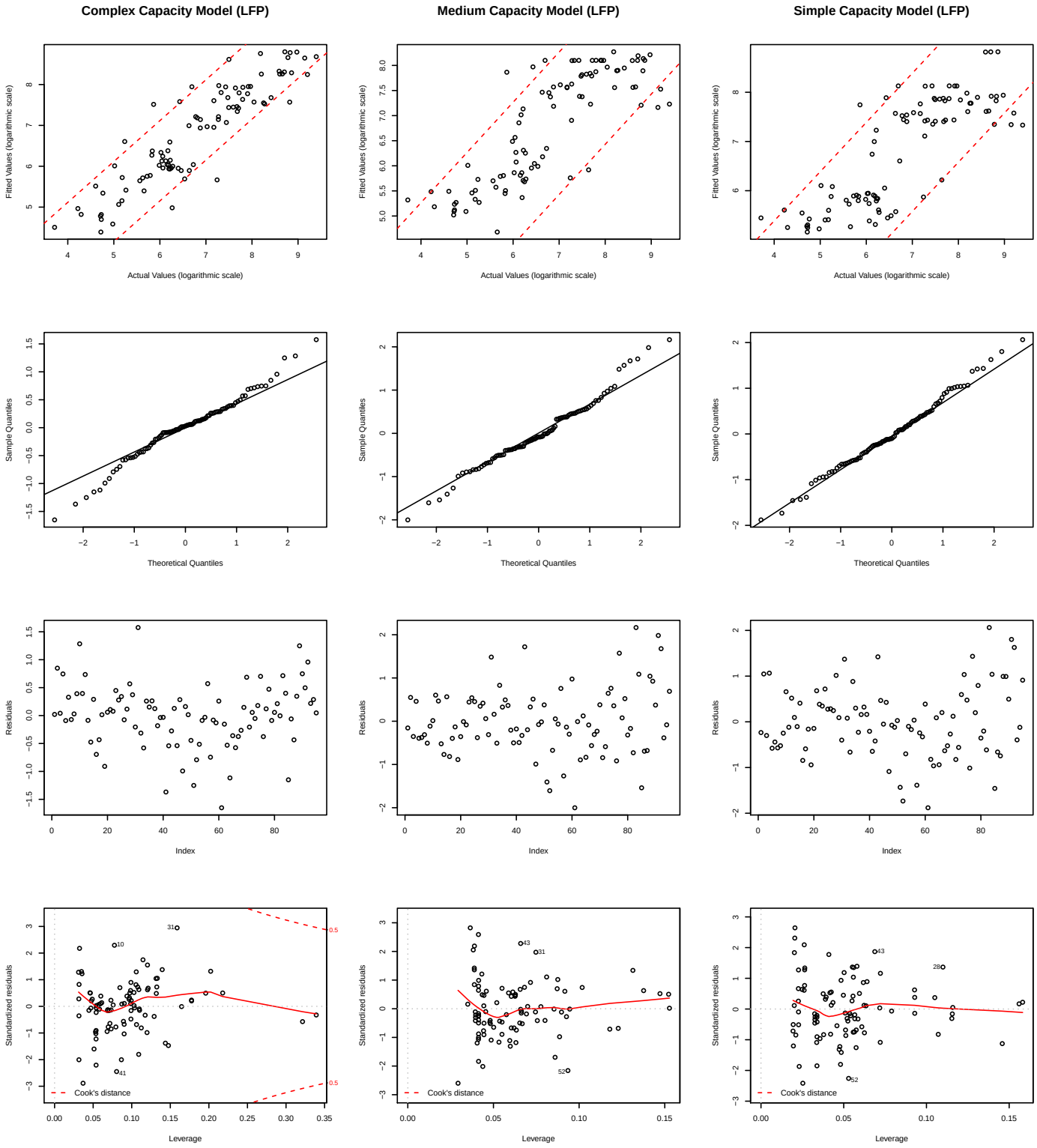

Figure 10.1: Graphical model diagnostics for the three capacity models for LFP cells.

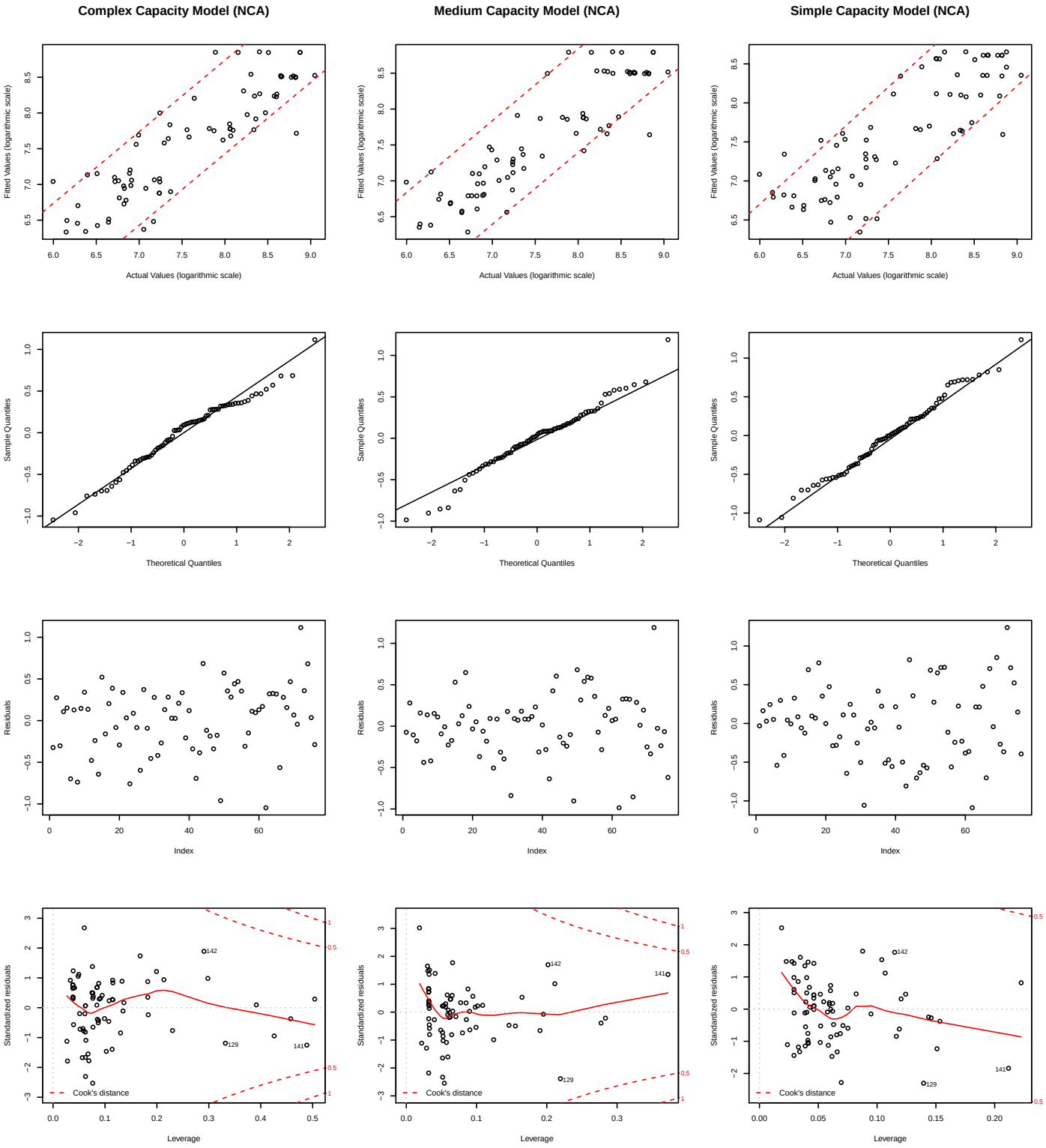

Figure 10.2: Graphical model diagnostics for the three capacity models for NCA cells.

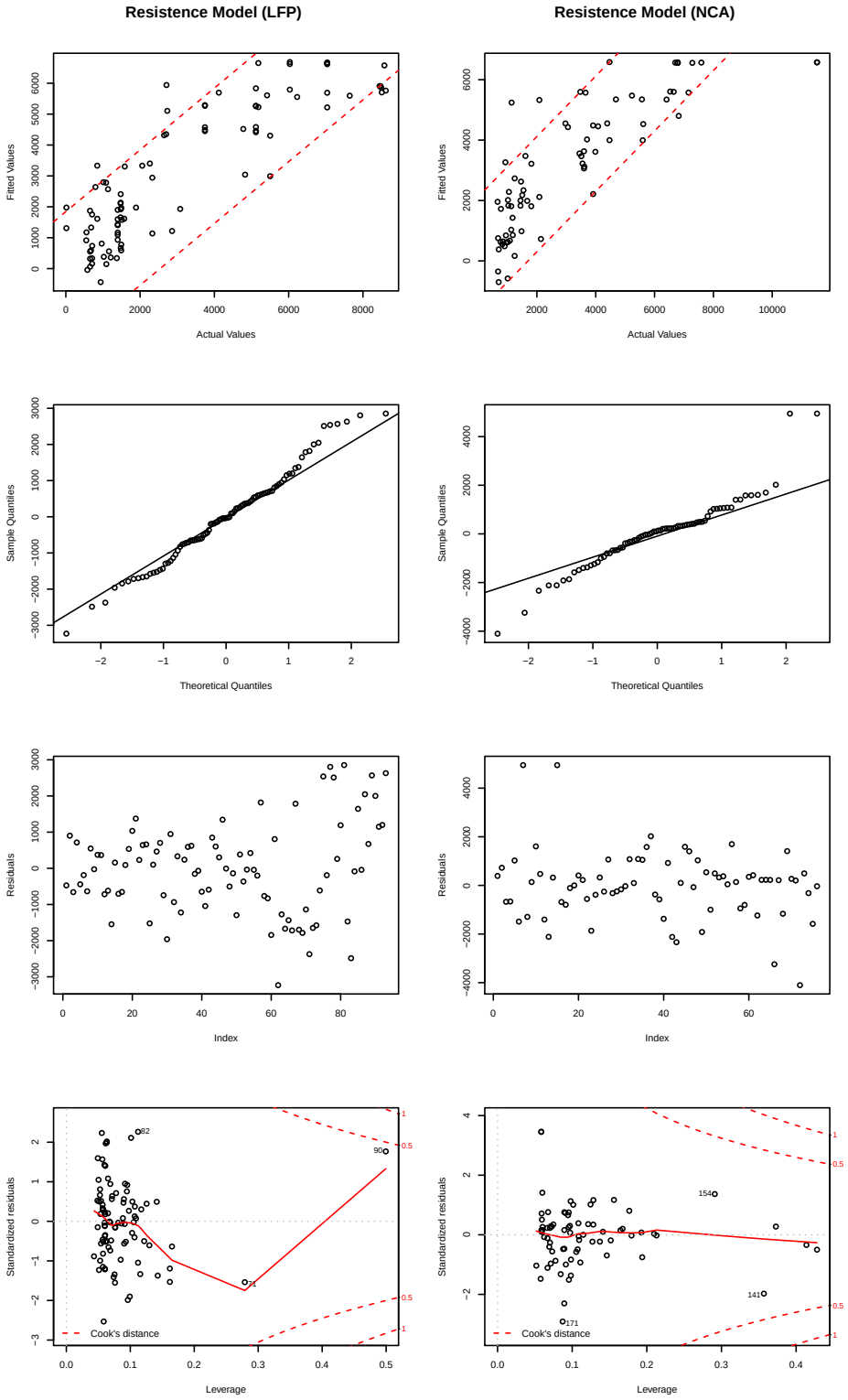

Figure 10.3: Graphical model diagnostics for the resistance models for LFP and NCA cells.

The contour plots for the resistance models can be found in figure 10.6. These plots look more interesting than the ones seen before. Note that in some extremal parts of the design region the predicted response actually becomes negative, note for example the combination `Curr`=5, `Temp`=70. This behavior can be accepted as it is indeed true that such conditions have terrible negative effects on the life expectancy of the cells.

10.2.1 Final Remark

Note that we started out modeling the responses using the designed factor settings instead of the actual ones. Of course, this mistake was soon realized and corrected, however, the interesting thing about these initial models is that they also fit the actual data quite well.

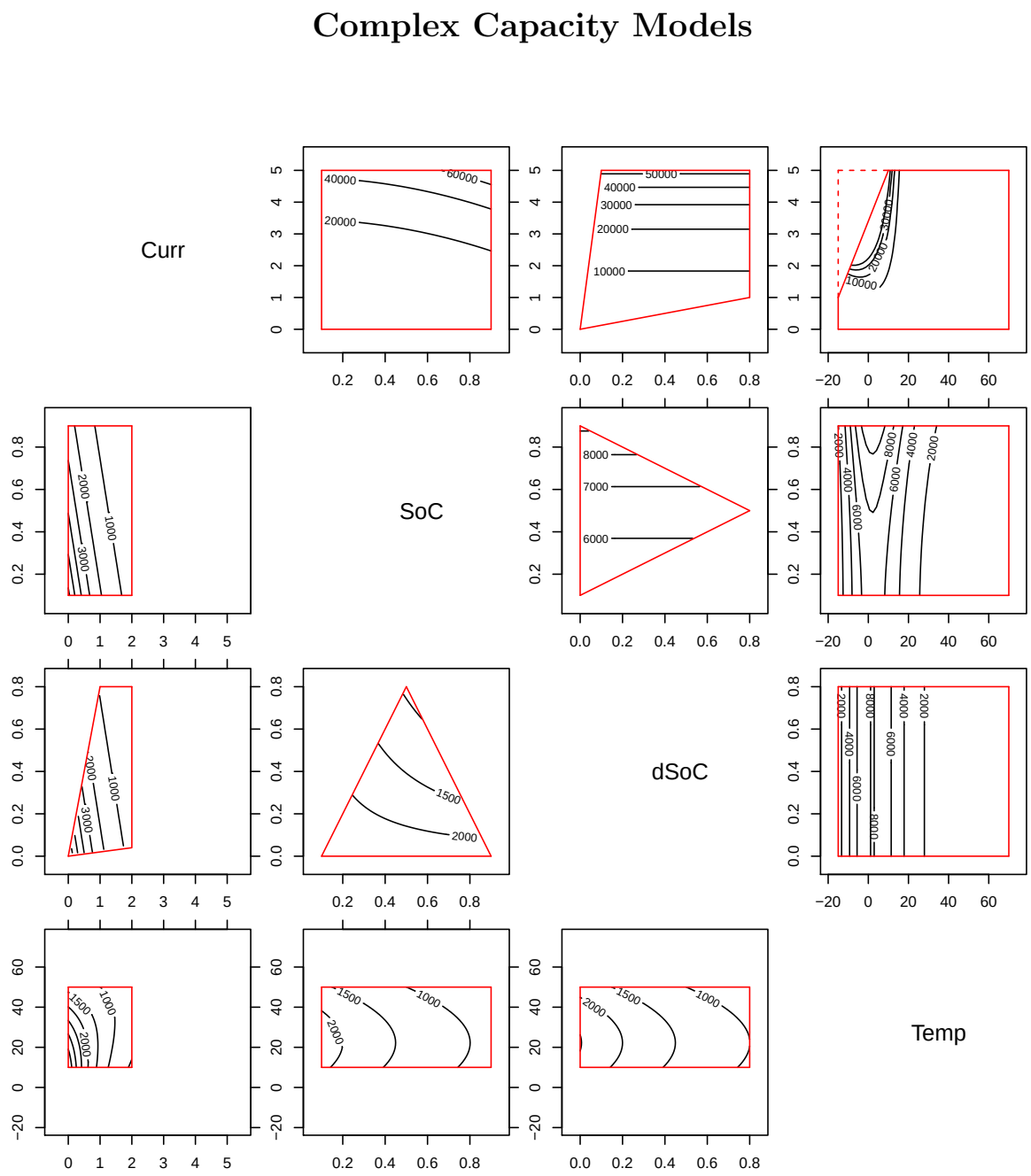

Figure 10.4: Contour plots depicting the fitted models graphically. The plots above the diagonal correspond to the complex LFP capacity model, the ones below the diagonal are for the complex NCA capacity model.

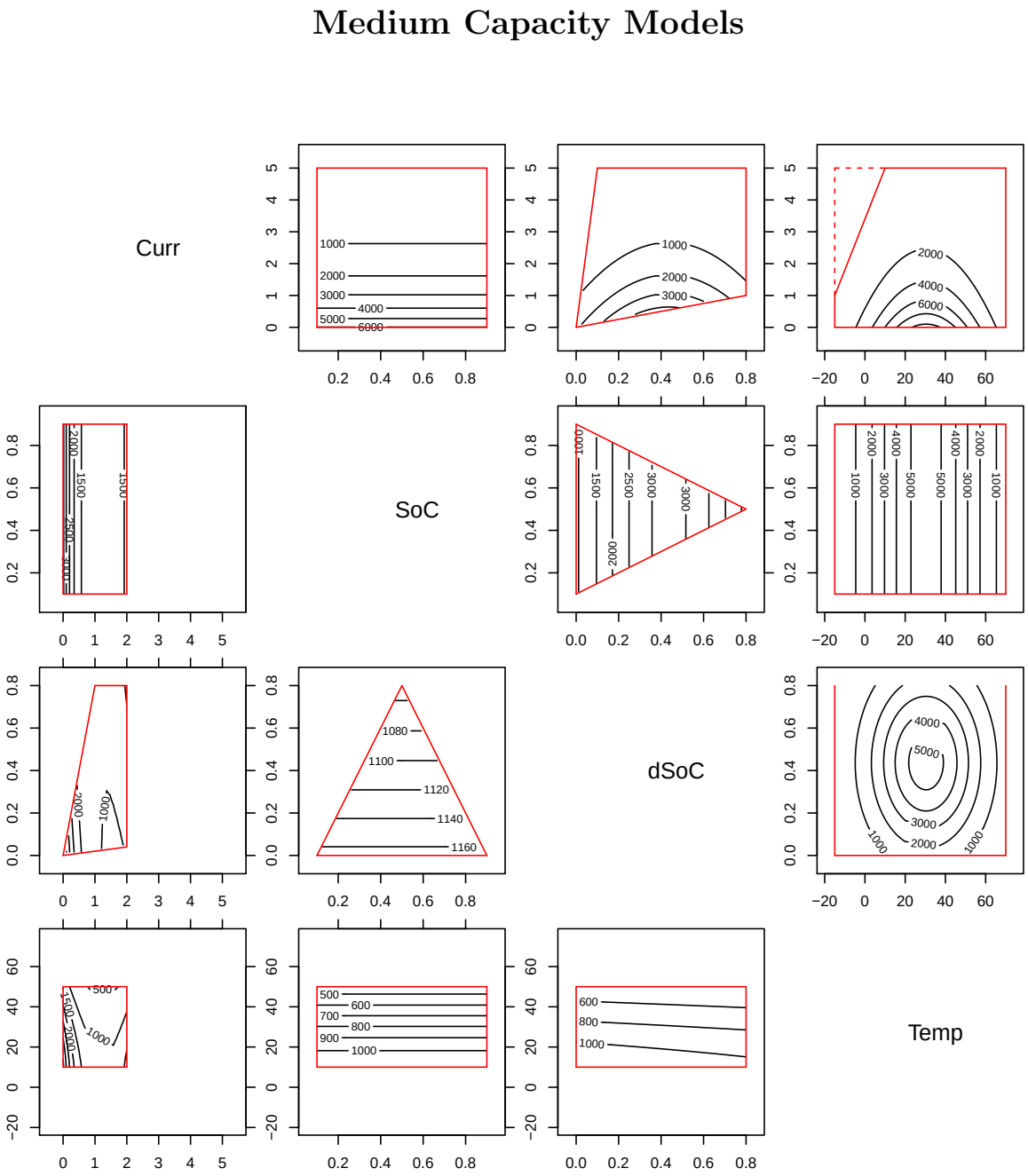

Figure 10.5: Contour plots depicting the fitted models graphically. The plots above the diagonal correspond to the medium LFP capacity model, the ones below the diagonal are for the medium NCA capacity model.

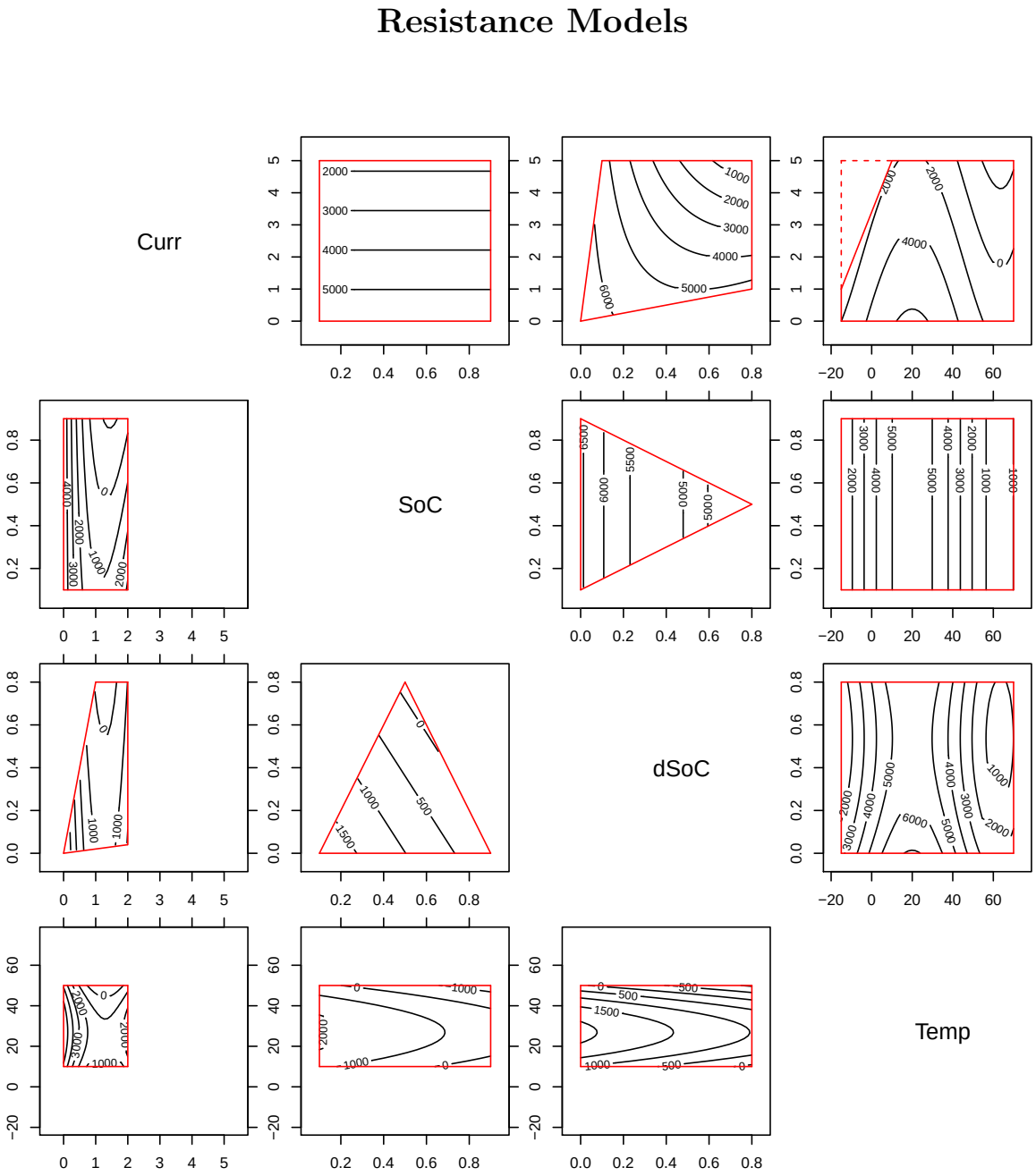

Figure 10.6: Contour plots depicting the fitted models graphically. The plots above the diagonal correspond to the LFP resistance model, the ones below the diagonal are for the NCA resistance model.

Chapter 11

Functionality Testing for D-Optimal Designs

The following section is dedicated to investigating the functionality concerning the construction of D-optimal designs for the different R-alternatives, pointing out their advantages and disadvantages as well as to illustrate their utilization.

11.1 Creating D-Optimal Designs

Since Haselgruber's DDoE algorithm was implemented for the developer only and there exists no detailed documentation, it seems impossible to use this algorithm unless we further consult the developer. We thus decided to look into other R-alternatives instead.

11.1.1 Using 'AlgDesign' and 'DoE.wrapper'

We investigate the functionality of `optFederov` and `optMonteCarlo` from package 'AlgDesign' as well as `Dopt.design` from 'DoE.wrapper'. All functions use Fedorov's exchange algorithm as a basis. Note that both 'AlgDesign' functions can be used to create or augment a design respectively and will thus be considered in a later section as well.

The following is a summary of the most striking differences between the procedures:

- **Constraints:** First of all, in 'AlgDesign' only `optMonteCarlo` can handle constraints directly, there they are to be input as a function. The way constraints are to be supplied to 'Dopt.design' differs significantly, as this function instead requires a single character string.

  Even though `optFederov` cannot seemingly handle constraints and we need them for our application, there is an easy way to overcome this shortcoming. By creating the candidate list in such a way that all combinations violating the given constraints are removed before feeding it to the algorithm, we keep this alternative open.

- **Option `data`:** Another significant difference between the procedures is the form of the requested `data`. `optFederov` takes a candidate list which is to be specified as a matrix or data frame. `optMonteCarlo` requests a very specific data frame specifying the factors' level settings in seven or eight columns (see the specifications for details).

  For `Dopt.design` there exist two ways to specify the candidate data, either as a list via a data frame or matrix defined in `data` or via the option `factor.names` which is to be given the factors' level settings. However, if constraints are to be used, their incorporation into `Dopt.design` seems to be easier if the data is specified via `factor.names` rather than via `data`.

- **Number of Runs:** Note that if no number of experiments to be conducted is specified in `optFederov` or `optMonteCarlo` via `nTrials`, the procedures use a default value of 'number of model terms + 5'. However, if one does not provide such a number of runs to `Dopt.design` via `nruns`, an error will occur.

`Dopt.design` is also capable of considering blocks, whereas `optFederov` and `optMonteCarlo` both have an option `approximate`, which when set to TRUE creates approximate instead of exact designs. However, we will not use either of these possibilities for this thesis.

According to local optimization theory, it is favorable to repeat the entire process several times in order to avoid local maxima. This can be achieved by the option `nRepeats` for all alternatives at hand. `optFederov` also has an option named `nullify`, which presumably results in better start designs by sequentially adding new design points according to some criterion instead of doing so randomly. In `optMonteCarlo` this option is changed a little and renamed to `RandomStart`, which must be set to FALSE. Note that `nullify=1` will render `nRepeats` useless for `optFederov` and `Dopt.design`.

As a short demonstration, we show the codes that will create a 20-run D-optimal design for the given constraints (in coded form) on the four factors that were also considered in the original design created by Haselgruber.

We will use `nRepeats=5` (denoted by a 1 after the design name), `nRepeats=200` (2) and `nullify=1` or `RandomStart=FALSE` (3) sequentially. Since the plotting of many 20-run designs would be too confusing, we plan to compare the resulting designs by means of their respective D-efficiency. However, whereas this measure is part of `optFederov`'s and `optMonteCarlo`'s output, for `Dopt.design` it needs to be accessed via the attributes.

In this first step, where only the basic functioning of the procedures is of interest, we need not be concerned with possible level settings, but note that this will definitely become an issue later on. As mentioned before, with the use of constraints, a candidate list has to be provided to `optFederov`. This is (for now) achieved by creating a full four-factor (5,5,3,3)-factorial design via `gen.factorial` from 'AlgDesign' and then selecting all valid runs via `subset`. Note that `gen.factorial` does not have an option to specify the levels' ranges and by default uses integers. Since we want all values within a [-1,1] interval, we can use a simple trick for the two factors with five levels each, we halve the values.

Also, it is worth mentioning that an almost full model (e.g. a quadratic model with several terms excluded, as in our case) need not be specified tediously term per term as there exist functions such as `quad` or `cubic`, which will produce a full quadratic or cubic model

respectively. How single terms could be excluded from this formula is shown below in the commented version of `model`. However, it was impossible to get it to work.

```
library('rsm')
library('AlgDesign')
library('DoE.wrapper')


# ------------------------------------------------------------------
#  optFederov
# ------------------------------------------------------------------

fullFact = gen.factorial(c(5,5,3,3), varNames=c("T","SoC","dSoC","Curr"))
T_coded = fullFact$T/2
SoC_coded = fullFact$SoC/2
fullCoded = data.frame("T"=T_coded, "SoC"=SoC_coded,
                       "dSoC"=fullFact$dSoC, "Curr"=fullFact$Curr)

constr = function(x){(x[1] >= -49/64+(25/64)*x[4]) &&
                     (x[3] <= 2*x[2]+1) &&
                     (x[3] <= -2*x[2]+1) &&
                     (x[3] <= 5*x[4]+4) &&
                     (x[3] >= 0.125*x[4]-0.875)}

dat = subset(fullCoded, constr(fullCoded)==TRUE)

model = "~T+SoC+dSoC+Curr+T*SoC+T*dSoC+T*I+SoC*Curr+dSoC*Curr+
        T^2+SoC^2"
# model = "~ quad(.) - I(SoC*dSoC) - I(dSoC^2) - I(I^2)"

desF1 = optFederov(frml=model, data=dat, nTrials=20, nRepeats=5)
desF2 = optFederov(frml=model, data=dat, nTrials=20, nRepeats=200)
desF3 = optFederov(frml=model, data=dat, nTrials=20, nullify=T)

desF1$Dea; desF2$Dea; desF3$Dea


# ------------------------------------------------------------------
#  optMonteCarlo
# ------------------------------------------------------------------

dat = data.frame(var=c("T","SoC","dSoC","Curr"), low=c(-1,-1,-1,-1),
         high=c(1,1,1,1), center=c(0,0,0,0), nLevels=c(5,5,3,3),
         round=1, factor=0)

constr = function(x){(x[1] >= -49/64+(25/64)*x[4]) &&
                     (x[3] <= 2*x[2]+1) &&
                     (x[3] <= -2*x[2]+1) &&
                     (x[3] <= 5*x[4]+4) &&
                     (x[3] >= 0.125*x[4]-0.875)}

desMC1 =  optMonteCarlo(nTrials=20, data=dat, frml=model,
                        constraints=constr, nRepeats=5)
desMC2 =  optMonteCarlo(nTrials=20, data=dat, frml=model,
                        constraints=constr, nRepeats=200)
desMC3 =  optMonteCarlo(nTrials=20, data=dat, frml=model,
                        constraints=constr, RandomStart=FALSE)

desMC1$Dea; desMC2$Dea; desMC3$Dea


# ------------------------------------------------------------------
#  Dopt.design
# ------------------------------------------------------------------
```

```
con = "T >= -49/64+(25/64)*Curr & dSoC <= 2*SoC+1 & dSoC <= -2*SoC+1
       & dSoC <= 5*Curr+4 &  dSoC >= 0.125*Curr-0.875"

fact = list(T=c(-1,1), SoC=c(-1,1), dSoC=c(-1,1), Curr=c(-1,1))

desD1 = Dopt.design(factor.names=fact, formula=model, nlevels=c(5,5,3,3),
                    constraint=con, nruns=20, nRepeats=5, seed=14028)

desD2 = Dopt.design(factor.names=fact, formula=model, nlevels=c(5,5,3,3),
                    constraint=con, nruns=20, nRepeats=200, seed=14028)

desD3 = Dopt.design(factor.names=fact, formula=model, nlevels=c(5,5,3,3),
                    constraint=con, nruns=20, nullify=1, seed=14028)

attr(desD1, "design")$optimality.criteria$Dea
attr(desD2, "design")$optimality.criteria$Dea
attr(desD3, "design")$optimality.criteria$Dea
```

The following table shows a summary of the results. All values given are the `Dea`-values, which are lower bounds of the true D-efficiencies of the designs, determined as

$$D_{eff} = \exp(1 - 1/G_{eff}).$$

	`nRepeats`=5	`nRepeats`=200	`nullify`=1
`optFederov`	singular	singular	singular
`optMonteCarlo`	0.641	0.694	0.709
`Dopt.design`	0.709	0.709	0.709

Note that `optFederov` fails to find a non-singular start design, both with random selection and with the nullification algorithm, for this given setting. Without a non-singular start design the optimization routine cannot be initialized and thus no useful result is returned.

It seems to be clear that `Dopt.design` delivers better results for few repetitions, but one has to be careful when jumping to hasty conclusions because the efficiency varies considerably with the random seed in use. It needs to be mentioned that it is only possible to change this seed for `Dopt.design`, not however for either algorithm taken from `AlgDesign`. Even though the table above contains only the results for one trial run, we have run all alternatives several times and the results were rather astounding: Whereas for `Dopt.design` there were no changes in the values at all, `optMonteCarlo` produced values with greater variation. This can probably be traced back to the fact that this procedure uses a random part of the candidate list. It is also noteworthy that the designs produced by `Dopt.design`, although resulting in the same efficiency values, were by no means all equal.

One conclusion that should be safe to make, however, is that it is sufficient to use several repetitions in `nRepeats`, 200 seems to be an overkill here.

11.2 Reconstructing Haselgruber's Design

Since the factors' levels are unevenly spaced, we need to be able to specify them directly, not only low and high levels with number of levels per factor respectively. While we have

not yet found a way to accomplish this with the `optFederov` or `optMonteCarlo` routine, in `Dopt.design` the factor levels can be set by hand in the specification of `factor.names`. It is important to know that all factors are of a quantitative nature.

The following `R`-code produces a D-optimal design, created with `Dopt.design`, with $n = 32$ runs while fulfilling the restrictions on the factors stated before. The constraints handled in `con` are the same as above, but `fact` now contains the factor names as well as the complete level specification. Note that, as a consequence, the parameter vector `nlevels` is no longer needed. Also, the full factorial created by `Dopt.design` at the beginning now contains more runs than before.

```
con = "T >= -49/64+(25/64)*Curr & dSoC <= 2*SoC+1 & dSoC <= -2*SoC+1
       & dSoC <= 5*I+4 &  dSoC >= 0.125*Curr-0.875"

fact = list(T=c(-1,-0.5,0,0.5,1), SoC=c(-1,-0.75,0,0.75,1),
            dSoC=c(-1,-0.75,0,0.75,1), Curr=c(-1,-0.6,1))

desD32 = Dopt.design(factor.names=fact, constraint=con, nruns=32,
                     nullify=1)
```

The resulting design has a D-efficiency of at least 0.786. After running the code for several other values of `nruns`, it turned out that the design for $n = 20$, denoted by `desD20`, has the highest D-efficiency of 0.838.

We also create Haselgruber's design by setting the amount of experiments to be conducted equal to the number of rows in the candidate set. Note that we had to use the 'long' version of the model specification above as the other one resulted in only singular start designs for `optFederov`. To this point we do not know the reason for this flaw, it might however be a bug in the function `quad`.

```
desHG = data.frame(T=c(0,0,1,-0.5,1,1,0,0,-1,-0.5,1,-1,1,0,1,1,-0.5,
                       0,0,0,-1, 1,1,1,1,0,-1,1,0,-1,1,1),
            SoC=c(-0.75,0.75,0,0,1,-1,0,0,1,0,0,-1,0,-0.75,0.75,-1,
                  0,1, 0,0.75,1,-0.75,0,1,-0.75,0,0,0.75,-1,-1,0,0),
            dSoC=c(-0.75, -0.75,1,1,-1,-1,-1,1,-1,1,1,-1,1,-0.75,
                   -0.75, -1,1,-1,-1,-0.75,-1,-0.75, 1,-1,-0.75,1,-1,
                   -0.75,-1,-1,1,-1),
            I=c(1,1,1,-0.6,-1,-1,-1,1,-1,-0.6,-0.6,-1,1,1,1,-1,-0.6,
                -1,-1,1,-1,1,-0.6,-1,1,1,-1,1,-1,-1,-0.6,-1))

desDHG = optFederov(frml=model, data=desHG, nTrials=32, nRepeats=20, rows=1:32)
desDHG$Dea

desDHG_alt = Dopt.design(data=desHG, formula=model, constraint=con,
                         nruns=32)
attr(desDHG_alt, "design.info")$optimality.criteria$Dea
```

This design, which has already been plotted and is thus omitted here, has a D-efficiency

of at least 0.811 according to `optFederov` and 0.827 according to `Dopt.design`, which is higher than that of the 32-run test design, yet lower than that of the 20-run design.

Note that the output design from `optFederov` cannot be used immediately, since the output contains all of the attributes as well. Thus, to access the actual design, a code line such as `design = output$design` is needed.

11.3 Augmenting an Existing D-Optimal Design

11.3.1 Using `DoE.wrapper`

For a quick check of the functionality of the routine `Dopt.augment`, we augment a 20-run design by $m = 12$ runs, keeping the same constraints as before.

As before, the D-efficiency of the 20-run design was 0.838 and that of the resulting augmented design is at 0.867, which yields a slight improvement. In order to make the designs reproducible, we choose a random seed this time (it does not need to be the same seed for both the creation and the augmentation though).

```
desD20 = Dopt.design(factor.names=fact, constraint=con, nruns=20,
                     seed=14)
desD20aug = Dopt.augment(desD20, m=12, constraint=con, seed=14)
```

Generally, we would wish to add further constraints for the augmentation process. However, it is not (yet) possible with this package to add constraints that have already been violated by the original design. A quick fix for this problem is to simply remove all experiments that violate the additional constraint and then 'feed' the resulting data frame or matrix into `Dopt.design` so as to obtain an object of class 'design' with the respective attributes.

```
desD20old = Dopt.design(factor.names=fact, constraint=con,
                        nruns=20, nullify=1, seed=100420)
desD20clean = subset(desD20old)[desD20old$dSoC <= 46/30 -
                                64/30*desD20old$I, ]
lendes = dim(desD20clean)[1]
desDoriginal = Dopt.design(data=desD20clean, nruns=lendes,
                           seed=100420, formula = model,
                           constraint=connew, nullify=1)
```

In our example, the cleaned up part of the original design contains 18 experiments and has a D-efficiency of at least 0.662. We want the resulting design to have $n = 32$ runs.

Now, we can augment `desDoriginal` almost the same way as shown above while incorporating the new constraint. The only difference to above is that the candidate list for the new design has to be given explicitly, since otherwise a full two-level factorial design

in the initial factors is chosen as a candidate list, resulting in an insufficient number of candidates. Therefore, another little hack is needed since `gen.factorial`, the `AlgDesign`-routine to create a full factorial design, only takes the number of levels for the factors, does however not allow any specifications thereof. We admit freely that the solution depicted below is not a nice one. We manually 'override' all level settings that do not comply with our expectations.

```
connew = "T >= -49/64+(25/64)*Curr & dSoC <= 2*SoC+1 &
          dSoC <= -2*SoC+1 & dSoC <= 5*I+4 &
          dSoC >= 0.125*Curr-0.875 & dSoC <= -64/30*Curr + 46/30"

cand = gen.factorial(c(5,5,5,3), varNames=c("T","SoC","dSoC","Curr"))
cand[cand$T==-1,1] = -0.5
cand[cand$T==-2,1] = -1
      .
      .
      .
cand[cand$Curr==0,4] = -0.6

desDnew = Dopt.augment(desDoriginal, m=14, candidates=cand,
                       seed=200, formula=model, nullify=1,
                       constraint=connew)
```

Unfortunately, there seems to be a bug and this is not yet working even though it should. The errors indicate that there is some internal problem, such as 'candidates must be a matrix or data frame', which comes up even after specifically defining `cand` as either matrix or data frame. Since it is mentioned in the documentation of this package that the functionality with optimal designs is yet in a beta state, we decide to accept this outcome as a bug.

11.3.2 Using `AlgDesign`

We try yet another method, using `optFederov` since it is the only algorithm in 'AlgDesign' that has the option to augment an existing design. Here, since this routine cannot handle constraints, we have to define the candidate set 'by hand'. For this, we use `cand` created above and clean it in such a way that the remaining rows fulfill the constraints. Note that the code fragment presented below will only work if an original design, which is to be augmented, is specified before. We call this design `desDold`. Also note that usually this design will have to be cleaned as well, since it is to be expected that one or more constraints are violated. However, if one persistently wishes to augment an existing design even though it did not fulfill the current constraints, other than `Dopt.augment` the method at hand permits this. In order to smoothly clean the candidate set, we have to define the new set of constraints yet again as a function.

```
connew = function(x) { x[1] >= -49/64+(25/64)*x[4] &
                       x[3] <= 2*x[2]+1 &
```

```
                        x[3] <= -2*x[2]+1 &
                        x[3] <= 5*x[4]+4 &
                        x[3] >= 0.125*x[4]-0.875 &
                        x[3] <= -64/30*x[4] + 46/30 }

cand_clean = subset(cand, connew(cand)==TRUE)


# use this line if the original design needs to be cleaned:
desDold_clean = subset(desDold, connew(desDold)==TRUE)

# use this line instead if the design does not need to be cleaned:
# desDold_clean = desDold

cands = rbind(desDold_clean, cand_clean)
lenold = dim(desDold_clean)[1]
desDaug = optFederov(frml=model, nTrials=32, data=cands, nullify=1,
                     augment=TRUE, rows=1:lenold)
```

We will show that this approach works, using Haselgruber's design as the original design to be augmented.

Before that, however, one advise should be given: even though it is quite common in R to abbreviate the values TRUE and FALSE to T and F respectively, this should better be avoided when either of these letters are already used for variables. For those readers interested in the effects, we recommend running the above `subset`-command with both T and TRUE and compare the results via the dimension command `dim`.

11.3.3 Augmenting Haselgruber's Design

We start by 'cleaning' up the design, since we have added a constraint that has been violated by the original design for certain. The resulting design `desHG_clean` contains 28 runs, thus only 4 experiments did not fulfill the new constraint. Note that the candidate list in use for the augmentation, that is the list taking into account all constraints, consists of 64 points only plus the original 28 when using the specified mesh density. If it were desired to have a design with significantly more runs than, say, 32, a higher mesh density should probably be considered in order to have a selection large enough to achieve a good D-efficiency.

The following R-code cleans out the original Haselgruber design to comply with the latest restriction and builds a candidate list for the augmentation from it.

```
desHG_clean = desHG[desHG$dSoC <= -64/30*desHG$Curr + 46/30, ]
cands = rbind(desHG_clean, cand_clean)
```

After having determined the functionality of the R-packages `AlgDesign` and `DoE.wrapper`, we are now ready to search for the optimal settings concerning the run size. We try experiment sizes ranging from the minimum possible to the full extent of the candidate list

in a function `best_runsize`. By minimum possible runs we mean the maximum of model terms plus one and the number of experiments in the old design (in our case the latter yields the maximum). However, sometimes the use of a small integer that is added to the aforementioned minimum number delivers better results and helps avoiding singular start designs. If we choose this offset to be around 5 there hardly ever seemed to be any problems. Alternatively, and not in all cases successfully, it is possible to repeat the function call until a non-singular design is found without the use of an offset. Unfortunately, `optFederov` does not take a specified random seed that would help avoid this problem.

The `best_runsize` function returns, among other things, a vector containing the D-efficiency bounds for the different run sizes investigated. The resulting values are plotted against their respective run sizes and shown in figure 11.1.

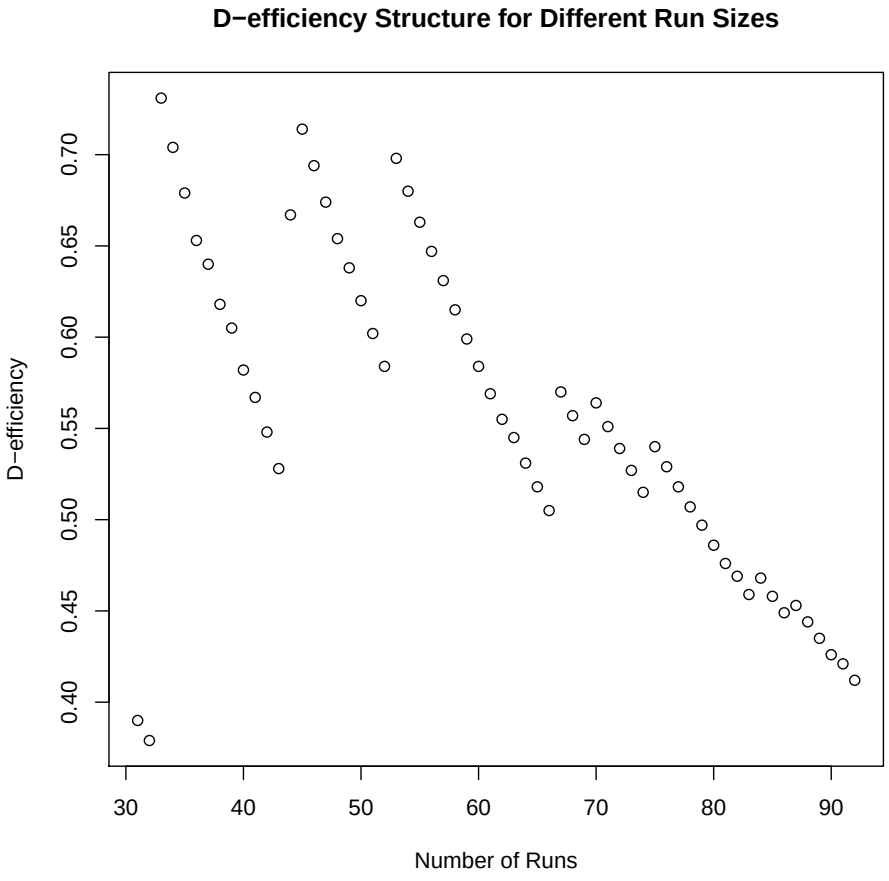

Figure 11.1: Plot showing the lower bounds for the D-efficiencies of the best design found for the respective run size.

Chapter 12

Constructing D-Optimal Designs

After the first experiment considering four factors was conducted, it was concluded that the true nature of the underlying problem is indeed much more complex than considered at the beginning. For this purpose, in a second stage, a more complex situation is to be taken into account. The general approach to use D-optimal computer-generated designs is to be kept. However, the practitioners intend on coming up with a better means of controlling the measurement instrumentation. It is of the essence to be able to better control the factor settings (since any design would be useless otherwise).

The practitioners came up with an extensive list of 12 factors that might influence the responses. Note that most of these have been identified before the initial experiment, but have been kept at a fixed level there. Table 12.1 contains these factors and their possible level settings, both in natural and coded values.

It soon turned out that already the creation of a complete candidate set for 12 factors is a computational challenge. Therefore, it was decided to break down the experiment and to create a six-factor design including all the factors printed in bold in table 12.1.

Since not all combinations of factor settings are technically achievable (as has already been the case with the original four factors), we also state a list of the extensive constraint structure.

I	$dSoC \leq 2 \cdot SoC + 1$
II	$dSoC \leq -2 \cdot SoC + 1$
III	$dSoC \leq 5 \cdot I + 4$
IV	$dSoC \geq (3/16) \cdot I - (13/16)$
V	$dSoC \leq -(5/16) \cdot I + (13/16)$
VI	$T \geq (25/64) \cdot I - (49/64)$
VII	$I \leq (9/5) \cdot AI + 1$
$VIII$	$I \geq 2 \cdot AI - 1$
IX	$I \geq -(1/5) \cdot AI - 1$

Factor	Description	Entity	Level Settings
T	**Temperature**	°C	natural: -10, 10, 30, 50, 70 coded: -1, -0.5, 0, 0.5, 1
TF	Temperature-Frequency	°C / sec	natural: 0, 0.01, 0.05, 0.1, 0.2 coded: -1, -0.9, -0.5, 0, 1
dT	Delta Temperature	°C	natural: 0, 5, 10, 35 coded: -1, -5/7, -3/7, 1
Soc	**State of Charge**	mAh	natural: 10, 20, 30, 40, 50, 60, 70, 80, 90 coded: -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1
dSoC	**Delta State of Charge**	%	natural: 0, 0.1, 1, 5, 10, 25, 50, 65, 80 coded: -1, -0.9975, -0.975, -0.875, -0.75, -0.375, 0.25, 0.625, 1
I	**Current**	rel. to Cap.	natural: 0, 0.5, 1, 5 coded: -1, -0.8, -0.6, 1
AI	**Current Asymmetry**	I_{cd} - I_{cc}	natural: -4.5, -1.5, -0.5 , 0, 0.5, 1.5, 4.5 coded: -1, -1/3, -1/9, 0, 1/9, 1/3, 1
P	Pressure	Bar	natural: 0, 1, 5, 10 coded: -1, -0.8, 0, 1
PF	Pressure-Frequency	Hz	natural: 0, 10, 50, 100 coded: -1, 0.82, -0.01, 1
dP	Delta Pressure	Bar	natural: 0, 1, 5 coded: -1, -0.6, 1
Hy	Humidity	relative %	natural: 0, 10, 50, 100 coded: -1, -0.8, 0, 1
Dy	**Dynamic of Current**	categorical	natural: 0, 1 coded: -1, 1

Table 12.1

X	$Dy \leq 4 \cdot T + 3$
XI	$Dy \leq -(8/3) \cdot T + (5/3)$
XII	$Dy \leq 4 \cdot SoC + 3$

Creating a Candidate List

The self-written function `cand_list` allows the incorporation of constraints directly at the point of creation and the consideration of the level settings for each of the factors. This yields an improvement over using the `AlgDesign`-routine `gen.factorial` since we are able to specify the factors' levels and there is no need to alter them afterward, which could prove tricky to do automatically sometimes. Furthermore, we apply a rather complex constraint structure and thus it is to be expected that the full factorial candidate list is reduced considerably by the constraints. Therefore, it is favorable to only generate those runs that fulfill the constraints.

The function's input consists of a list containing the factors' levels concatenated and a constraint structure defined as a function. The output is a data frame containing the candidate list of all possible settings that fulfill the constraints.

Note that the function currently has a progress bar incorporated as it is difficult to guess the run time for really large projects such as the full factor set from above.

12.1 Creating Six-Factor Designs

Using the factors printed in bold at the beginning of this section, we search for designs with a relatively high D-efficiency.

The constraints given for the six factors are to be specified as a function much the same as for 'AlgDesign'.

```
constr_six = function(x){x[3]<=2*x[2]+1 & x[3]<=-2*x[2]+1 &
                         x[3]<=5*x[4]+4 & x[3]>=3/16*x[4]-13/16 &
                         x[3]<=-5/13*x[4]+13/16 &
                         x[1]>=25/64*x[4]-49/64 & x[4]<=9/5*x[5]+1 &
                         x[4]>=2*x[5]-1 & x[4]>=-1/5*x[5]-1 &
                         x[6]<=4*x[1]+3 & x[6]<=-8/3*x[1]+5/3 &
                         x[6]<=4*x[2]+3}
```

With all this, the candidate list is created. This list contains only 1359 out of the 22680 possible level combinations.

The model to be used is to incorporate the results from previous sections and therefore we apply the complete quadratic model in the six factors as a basis but exclude all those

factor and interaction terms in the original four factors that did not seem significant in either of the four resulting models.

In order to quantify the importance of the foregoing results, we also compare the resulting design with that for the complete quadratic model, with which we start.

12.1.1 Complete Quadratic Model

For this part, the model at hand is the full quadratic model from which the term `Dy^2` has to be removed since this term is not estimable as there are only two levels for this factor. This model contains

$$6 + 6 + \binom{6}{2} - 1 = 26$$

terms, which means that the smallest possible design size for the model at hand is 27.

In Figure 12.1 we show the resulting (lower bounds for) D-efficiencies against the respective run sizes. Since clearly run sizes of above 100 are neither optimal nor practical, the plot contains only run sizes up to 80.

One can see that the highest determined D-efficiency value of 0.762 is reached for a run size of 57. Since 57 runs are usually too much to conduct in a technical application (keep in mind that replications would be needed as well), we recommend to instead settle for a design with only 46 runs, for which a D-efficiency of 0.731 has been reached. This design is a good compromise as it needs to be replicated only 1.04 times in order to achieve the same precision on the parameter estimation as the best design found. This would yield a run size of 48.

We show one possible design with 46 runs and the advertised D-efficiency value in table 12.2.

12.1.2 Incomplete Quadratic Model

In addition to `Dy^2`, the following terms are excluded from this model as they did not appear in any of the four final models found in the modeling chapter:

`SoC, SoC^2, T*SoC, T*dSoC.`

Here, the smallest possible design size is reduced to 23 since there are four additional terms that need not be estimable.

Figure 12.1 also contains the plot of D-efficiencies versus run sizes for this model. Here, the overall best efficiency of 0.848 is reached for 74 runs. However, the efficiency for the 46-run design is 0.75, so this design seems to be a good compromise. It is also depicted in table 12.2.

Note that the two resulting designs have a total of 27 runs (experiments) in common and many runs differ only slightly, that is to say that only the level setting of a single factor is different.

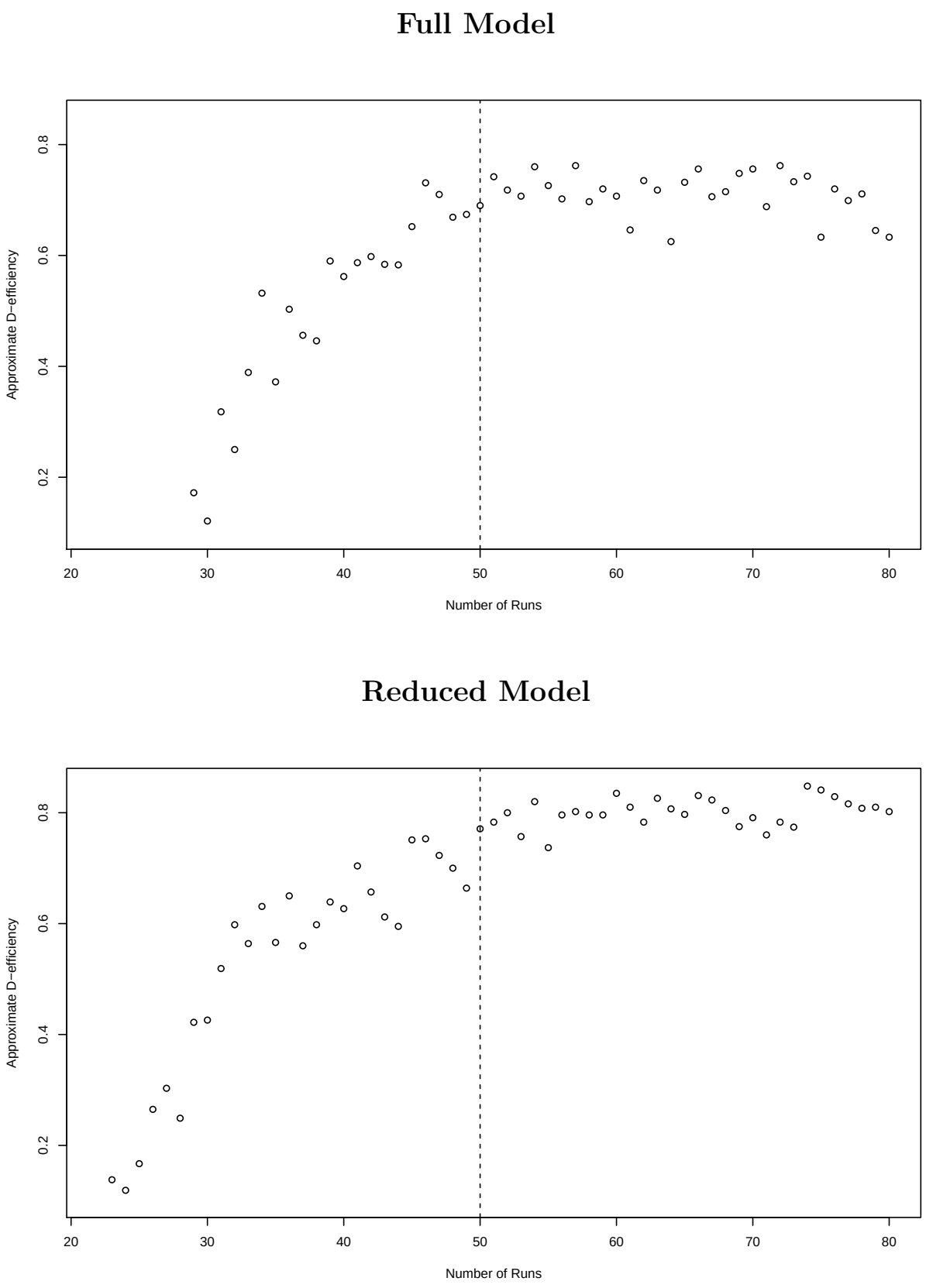

Figure 12.1: D-efficiency structures for the complete model (top) and the model with insignificant terms removed (bottom).
Note that these are the results of only one trial run.

Full Model

	T	SoC	dSoC	I	AI	Dy
1	0.0	-0.75	-0.875	-0.8	-1.00000	-1
2	1.0	-0.75	-0.875	-0.8	-1.00000	-1
3	-1.0	0.75	-0.875	-0.8	-1.00000	-1
4	1.0	0.75	-0.875	-0.8	-1.00000	-1
5	-1.0	-0.50	-0.375	-0.8	-1.00000	-1
6	1.0	-0.50	-0.375	-0.8	-1.00000	-1
7	0.0	0.50	-0.375	-0.8	-0.33333	-1
8	-1.0	-0.75	-0.875	-0.6	-0.33333	-1
9	-1.0	0.00	1.000	-0.6	-0.33333	-1
10	1.0	0.00	1.000	-0.6	-0.33333	-1
11	-1.0	-1.00	-1.000	-1.0	0.00000	-1
12	0.0	-1.00	-1.000	-1.0	0.00000	-1
13	1.0	-1.00	-1.000	-1.0	0.00000	-1
14	-1.0	0.00	-1.000	-1.0	0.00000	-1
15	1.0	0.00	-1.000	-1.0	0.00000	-1
16	-1.0	1.00	-1.000	-1.0	0.00000	-1
17	1.0	1.00	-1.000	-1.0	0.00000	-1
18	0.0	-0.50	-0.375	1.0	0.00000	-1
19	1.0	0.50	-0.375	1.0	0.00000	-1
20	0.0	0.25	0.250	1.0	0.00000	-1
21	0.0	0.00	-0.875	-0.6	0.11111	-1
22	-1.0	0.75	-0.750	-0.6	0.11111	-1
23	0.0	-0.25	0.250	-0.6	0.11111	-1
24	-1.0	0.00	1.000	-0.6	0.11111	-1
25	1.0	0.00	1.000	-0.6	0.11111	-1
26	1.0	-0.25	0.250	1.0	0.33333	-1
27	1.0	-0.50	-0.375	1.0	1.00000	-1
28	0.0	0.50	-0.375	1.0	1.00000	-1
29	0.0	-0.25	0.250	1.0	1.00000	-1
30	1.0	0.25	0.250	1.0	1.00000	-1
31	-0.5	-0.50	-0.875	-0.8	-1.00000	1
32	-0.5	0.75	-0.875	-0.8	-1.00000	1
33	0.0	0.50	-0.375	-0.8	-1.00000	1
34	0.0	-0.50	-0.375	-0.8	-0.33333	1
35	-0.5	0.00	1.000	-0.6	-0.33333	1
36	0.0	-0.50	-1.000	-1.0	0.00000	1
37	-0.5	1.00	-1.000	-1.0	0.00000	1
38	0.0	-0.50	-0.375	1.0	0.00000	1
39	0.0	-0.25	0.250	1.0	0.00000	1
40	-0.5	-0.50	-0.875	-0.6	0.11111	1
41	0.0	0.75	-0.875	-0.6	0.11111	1
42	-0.5	0.25	0.250	-0.6	0.11111	1
43	0.0	0.00	1.000	-0.6	0.11111	1
44	0.0	0.50	-0.375	1.0	0.33333	1
45	0.0	-0.50	-0.375	1.0	1.00000	1
46	0.0	0.25	0.250	1.0	1.00000	1

Reduced Model

	T	SoC	dSoC	I	AI	Dy
1	-1.0	-0.75	-0.875	-0.8	-1.00000	-1
2	1.0	-0.75	-0.875	-0.8	-1.00000	-1
3	-1.0	0.75	-0.875	-0.8	-1.00000	-1
4	1.0	0.75	-0.875	-0.8	-1.00000	-1
5	-1.0	-0.50	-0.375	-0.8	-1.00000	-1
6	0.0	0.50	-0.375	-0.8	-1.00000	-1
7	0.0	-0.50	-0.375	-0.8	-0.33333	-1
8	-1.0	0.00	1.000	-0.6	-0.33333	-1
9	1.0	0.00	1.000	-0.6	-0.33333	-1
10	-0.5	-1.00	-1.000	-1.0	0.00000	-1
11	1.0	-1.00	-1.000	-1.0	0.00000	-1
12	-1.0	1.00	-1.000	-1.0	0.00000	-1
13	1.0	1.00	-1.000	-1.0	0.00000	-1
14	0.0	-0.50	-0.375	1.0	0.00000	-1
15	1.0	0.50	-0.375	1.0	0.00000	-1
16	0.0	-0.25	0.250	1.0	0.00000	-1
17	1.0	0.25	0.250	1.0	0.00000	-1
18	-1.0	-0.75	-0.875	-0.6	0.11111	-1
19	0.0	0.75	-0.875	-0.6	0.11111	-1
20	1.0	-0.25	0.250	-0.6	0.11111	-1
21	-1.0	0.25	0.250	-0.6	0.11111	-1
22	0.0	0.00	1.000	-0.6	0.11111	-1
23	1.0	-0.50	-0.375	1.0	0.33333	-1
24	0.0	0.50	-0.375	1.0	0.33333	-1
25	1.0	-0.50	-0.375	1.0	1.00000	-1
26	0.0	0.50	-0.375	1.0	1.00000	-1
27	0.0	-0.25	0.250	1.0	1.00000	-1
28	1.0	0.25	0.250	1.0	1.00000	-1
29	0.0	-0.50	-0.875	-0.8	-1.00000	1
30	-0.5	0.75	-0.875	-0.8	-1.00000	1
31	-0.5	-0.50	-0.375	-0.8	-1.00000	1
32	0.0	0.50	-0.375	-0.8	-1.00000	1
33	-0.5	-0.50	-0.375	-0.8	-0.33333	1
34	0.0	0.75	-0.875	-0.6	-0.33333	1
35	0.0	0.00	1.000	-0.6	-0.33333	1
36	0.0	-0.50	-1.000	-1.0	0.00000	1
37	-0.5	1.00	-1.000	-1.0	0.00000	1
38	0.0	1.00	-1.000	-1.0	0.00000	1
39	0.0	-0.50	-0.375	1.0	0.00000	1
40	0.0	0.50	-0.375	1.0	0.00000	1
41	-0.5	-0.50	-0.875	-0.6	0.11111	1
42	-0.5	0.00	1.000	-0.6	0.11111	1
43	0.0	0.00	1.000	-0.6	0.11111	1
44	0.0	-0.25	0.250	1.0	0.33333	1
45	0.0	-0.50	-0.375	1.0	1.00000	1
46	0.0	0.50	-0.375	1.0	1.00000	1

Table 12.2: The resulting 46-run designs, for the complete model on the left and for the reduced model on the right.

Chapter 13

Summary

It was found that extreme operating temperatures should be avoided whenever possible since they vastly shorten a cell's life time. The same is true for a high operating current, although physically there seem to be problems reaching such anyway.

Furthermore, we would suggest the following models for the two responses and chemistries in consideration. Note, however, that especially for the resistance criterion it is possible and likely that other factors that have not been considered in the experiment could improve the fit.

Note that all models initially featuring a term containing factor `Leak` have been removed, so as to model the life span of sound cells.

$$
\begin{aligned}
\log(\texttt{Cap80.LFP}) &= \beta_0 + \beta_1 \cdot \texttt{Temp} + \beta_2 \cdot \texttt{Temp}^2 + \beta_3 \cdot \texttt{Curr} + \beta_4 \cdot \texttt{SoC} + \beta_5 \cdot \texttt{dSoC} + \beta_6 \cdot \texttt{dSoC}^2 \\
&= 6.36 + 0.09 \cdot \texttt{Temp} - 0.0014 \cdot \texttt{Temp}^2 - 0.72 \cdot \texttt{Curr} + 0.85 \cdot \texttt{SoC} + 4.82 \cdot \texttt{dSoC} - 5.43 \cdot \texttt{dSoC}^2 \\
\\
\log(\texttt{Cap80.NCA}) &= \beta_0 + \beta_1 \cdot \texttt{Curr} + \beta_2 \cdot \texttt{Temp}^2 + \beta_3 \cdot \texttt{Temp} \cdot \texttt{Curr} + \beta_4 \cdot \texttt{SoC} \cdot \texttt{dSoC} \\
&= 8.90 - 1.29 \cdot \texttt{Curr} - 0.0004 \cdot \texttt{Temp}^2 + 0.02 \cdot \texttt{Temp} \cdot \texttt{Curr} - 2.31 \cdot \texttt{SoC} \cdot \texttt{dSoC} \\
\\
\texttt{Res300.LFP} &= \beta_0 + \beta_1 \cdot (\texttt{Temp} - 20)^2 + \beta_2 \cdot \texttt{dSoC} + \beta_3 \cdot (\texttt{Temp} - 20)^4 + \beta_4 \cdot \texttt{dSoC}^2 + \beta_5 \cdot \texttt{dSoC} \cdot \texttt{Curr} \\
&= 7084.26 - 5.19 \cdot (\texttt{Temp} - 20)^2 - 4325.13 \cdot \texttt{dSoC} + 0.0014 \cdot (\texttt{Temp} - 20)^4 \\
&\quad + 5545.50 \cdot \texttt{dSoC}^2 - 1627.33 \cdot \texttt{dSoC} \cdot \texttt{Curr} \\
\\
\texttt{Res300.NCA} &= \beta_0 + \beta_1 \cdot \texttt{Temp} + \beta_2 \cdot \texttt{dSoC} + \beta_3 \cdot \texttt{Curr} + \beta_4 \cdot \texttt{Temp}^2 + \beta_5 \cdot \texttt{Curr}^2 + \beta_6 \cdot \texttt{Curr} \cdot \texttt{SoC} \\
&= 3798.87 + 207.57 \cdot \texttt{Temp} - 1402.11 \cdot \texttt{dSoC} - 6206.40 \cdot \texttt{Curr} - 3.86 \cdot \texttt{Temp}^2 \\
&\quad + 2815.81 \cdot \texttt{Curr}^2 - 2183.75 \cdot \texttt{Curr} \cdot \texttt{SoC}
\end{aligned}
$$

The practitioners are rather confident with these models with respect to the threat of over-fitting, which always is an issue when searching for relationships unreflectedly.

Outlook

For future experiments containing even more factors, which are planned, there are several issues that should be kept in mind:

- It has to be understood that any design is built for a certain model, therefore the adequacy of a model has to be verified somehow beforehand.
- Especially with computer-generated designs, it needs to be clear that terms that were not considered in the underlying model might not be estimable. That is to say that the blind inclusion of certain terms into the analysis will likely lead to faulty conclusions unless the alias structure can be interpreted.
- It is important to repeat a design evenly, meaning that each experiment is conducted the same amount of times. It is of course to be expected that several experiments fail and that unbalancedness is thus introduced, but for a reasonable amount of repetitions (r) it should be possible to discard failed experiments and still obtain data for a design with (say) $r - 1$ repetitions.
- If it becomes necessary during the experiment to introduce an additional categorical variable (such as the different cell chemistries in the current experiment), the design must be applied to each of the classes respectively. Splitting the design into parts usually results in great efficiency loss and one almost unavoidably introduces a new alias structure, which is generally not interpretable.
- If a certain factor setting should prove unrealistic at an early stage of the experiment, it might be best to restart with a more suitable choice (if financially possible) or to determine the actual setting immediately so as to be able to reset this setting for future experiments.

Part IV

Appendix

Appendix A

Function `EoL_crit`

This function is the core devise for the determination of the response variables analyzed throughout this thesis.

```
EoL_det(cellID, datum, criterion="capacity", crit, perc,
        time=TRUE, response=NULL, remTime=NULL, shortCirc=NULL)
```

The function's output consists of a list containing three vectors.

- `EoL`: A vector of length `unique(cellID)` giving a single time value (in hours) or absolute value (for the alternatives) per cell. This is the actual main objective of this function.
- `Case`: A vector of length `unique(cellID)` giving the mode of the EoL-determination for every cell. Note that this output is only included for the option `time==TRUE`, otherwise the vector contains only NAs.

  The possible modes are as follows:

  - 0: EoL-time = elapsed time until last measurement
  - 1: EoL-time = interpolated time
  - 2: EoL-time = extrapolated time (for short circuit cells only)
- `BadCells`: This vector of arbitrary length returns all cells that might want to be removed after this procedure. These are short circuit cells, for which the EoL-time by design is determined via an extrapolation, that have a wrong sign on the rise. Such cells have a negative life-time.

Note that the length of the output vector by design differs from that of the input vectors. Therefore, we will have to deal with the fact that all other vectors needed for the investigation are now of different length. For this purpose we have written an auxiliary function to shorten all vectors to one value per cell.

Out of the nine possible input parameters four have to be provided in all cases, the rest is optional and may be used to change the default settings.

A.1 Input Parameters

`cellID` A vector containing some means of identification that is unique for a cell.

Since the original data contains several measurements per cell and for the analysis to follow we require a single value per cell, `cellID` is used to indicate which measurements belong to which cell.

`datum` A vector containing information about the times of measurements.

To be safe, this should be an output from `data_prepare`. In any case, it needs to be a vector of string entries in the exact format such that `R` can convert them to date-time objects. This required format is: "yyyy-mm-dd hh:mm:ss", where it does not matter whether the year component is given in four or two digits.

`criterion` An end-of-life criterion, specified in quotation marks. The only possible options are `"capacity"` (default) and `"resistance"`.

Note that the criterion is not applied internally but that a sensible percentage has to be provided by means of `perc`.

`crit` A vector of actual values for the chosen criterion.

In the procedure, a cell's capacity or resistance value at its first measurement is considered the initial value and will be treated as 100%.

`perc` A relative number specifying the end-of-life. The most reasonable choices would be 0.8 for `"capacity"` and 3 for `"resistance"`.

`time` A logical value indicating whether the response is 'time' (TRUE, default) or user-specified. If `time=FALSE` an alternative response has to be provided to `response`.

`response` A vector containing values for the alternative response when `time=FALSE`.

`remTime` An optional integer declaring a 'minimum life span' when option `time=TRUE` is in use. The end-of-life time for cells that did not survive `remTime` days will be set to NA.

`shortCirc` An optional vector containing binary information indicating whether a cell experienced a short circuit while under surveillance (1) or not (0).

A special extrapolation of the end-of-life time will be conducted for certain cases. This is only considered if `time=TRUE`.

Each of the input vectors must have the same length. Ideally, they are all taken from a common data frame.

A.2 Function Code

```
# ------------------------------------------------------------------ #
#   Function EOL_CRIT                                                #
# ------------------------------------------------------------------ #


EoL_det = function(cellID, datum, criterion="capacity", crit, perc,
                   time=TRUE, response=NULL, remTime=NULL,
                   shortCirc=NULL)
{


# begin error management and case determination

if (time == TRUE & length(response) > 0) {
  cat("Error: no criterion needed if time=TRUE. \n")
  return()
}
if (time != TRUE & length(response) == 0) {
  cat("Error: please specify a valid response. \n")
  return()
}
if (time == TRUE & length(response) == 0) {
  resp = datum
  cat("The EoL-response in use is 'Time'. \n")
}
else {
  resp = response
  cat("The EoL-response in use is user-specified. \n")
}

cap = TRUE; res = FALSE
if (criterion=="capacity") {
  cap = TRUE; res = FALSE        # nothing changes (default setting)
}
else if (criterion=="resistance") {
  cap = FALSE; res = TRUE
}
else {
  cat("Error: the criterion must be 'capacity' or 'resistance'. \n")
  return()
}

# end error management and case determination


output = NULL
case = NULL
bad = NULL

cells = unique(cellID)

for (cell in cells) {
  shorted = FALSE
  if (length(shortCirc) > 0 & unique(shortCirc[cellID==cell]) == 1) {
    shorted = TRUE
  }
  firstTime = min(as.POSIXlt(datum[cellID==cell]))
  lastTime = max(as.POSIXlt(datum[cellID==cell]))

  crit100 = head(subset(crit, cellID==cell), n=1)
  critPerc = crit100 * perc

  # taking data subset for the respective cell & revert this subset
```

```
#   (last item first)
tempResp = subset(resp, cellID==cell)
tempCrit = subset(crit, cellID==cell)
temp = data.frame("Crit"=rev(tempCrit), "Resp"=rev(tempResp))


#  distinguish between capacity and resistance for index search!
#       cap: last time cap falls below critPerc
#       res: last time res rises above critPerc

if (cap == TRUE & res == FALSE) {
  # finding index where cap falls below critPerc for the last time
  #   (first time for reversed set)
  capTest = temp$Crit[1]
  ind = 1
  while (capTest > critPerc & ind <= length(temp$Crit)) {  # testing whether last value
      is below critPerc
    ind = ind + 1
    capTest = temp$Crit[ind]
  }
  while (capTest < critPerc & ind <= length(temp$Crit)) {
    ind = ind + 1
    capTest = temp$Crit[ind]
  }
}

else if (cap == FALSE & res == TRUE) {
  # finding index where res rises above critPerc for the last time
  #  (first time for reversed set)
  resTest = temp$Crit[1]
  ind = 1
  while (resTest < critPerc & ind <= length(temp$Crit)) {  # testing whether last value
      is below critPerc
    ind = ind + 1
    resTest = temp$Crit[ind]
  }
  while (resTest > critPerc & ind <= length(temp$Crit)) {
    ind = ind + 1
    resTest = temp$Crit[ind]
  }
}

if (ind > length(temp$Crit)) {  # cell never reaches EoL criterion
  if (time == TRUE & shorted == FALSE) {
    # EoLTime = total elapsed time
    EoL = as.numeric(round(difftime(lastTime, firstTime,
                                    units="hours"),0))
    case = c(case, 0)
  }
  else if (time == TRUE & shorted == TRUE) {    # extrapolate for short circuit cells
    initCrit = crit100
    endCrit = temp$Crit[1]  # temp is in reversed order
    initTime = 0
    endTime = as.numeric(difftime(lastTime, firstTime, units="hours"))
    rise = (endTime - initTime)/(endCrit - initCrit)
    offset = initTime - rise*initCrit
    EoL = round(rise*critPerc + offset, 0)
    case = c(case, 2)
    if ((cap==TRUE & rise > 0) | (res==TRUE & rise < 0)) {
      bad = c(bad, cell)    # extrapolation does not work for those cells
    }
  }
  else {
    # take last value of resp as EoL if EoL-criterion is never met
    EoL = round(temp$Resp[1], 2)
  }
  output = c(output, EoL)
}
else {
```

```
    # indices between which cell meets EoL-criterion for the last time
    # left and right refer to the position as in a graph
    indLeft = ind
    indRight = ind - 1

    critLeft = temp$Crit[indLeft]
    critRight = temp$Crit[indRight]
    respLeft = temp$Resp[indLeft]
    respRight = temp$Resp[indRight]

    if (time == TRUE) {
      # EoLTime = elapsed time until last point on right plus interpolation
      diff = as.numeric(difftime(as.POSIXlt(respRight),
                                 as.POSIXlt(respLeft), units="hours"))
      before = as.numeric(difftime(as.POSIXlt(respLeft),
                                   as.POSIXlt(firstTime), units="hours"))
      EoL = round(((critLeft - critPerc)/(critLeft - critRight)) * diff +
                  before, 0)
    }
    else {
      diff = respRight - respLeft
      before = respLeft
      EoL = round(((critLeft - critPerc)/(critLeft - critRight)) * diff +
                  before, 2)
    }
    output = c(output, EoL)
    case = c(case, 1)
  }
}

# removing all cells whose EoL-time was before remTime
if (time == TRUE & length(remTime)>0) {
  for (i in 1:length(output)) {
    if (output[i] < remTime*24) {
      output[i] = NA
    }
  }
}

return(list("EoL"=output, "Case"=case, "BadCells"=bad))
}
```

Bibliography

[1] Ampanthong, P., and Suwattee, P. A Comparative Study of Outlier Detection - Procedures in Multiple Linear Regression. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I* (Hong Kong, 2009), Newswood Limited.

[2] Atkinson, A. C., and Donev, A. N. The Construction of Exact D-Optimum Experimental Designs with Application to Blocking Response Surface Designs. *Biometrika 76* (1989), 515–526.

[3] Bradley, N. The Response Surface Methodology. Master's thesis, Indiana University South Bend, 2007.

[4] Böhning, D. A Vertex-Exchange-Method in D-optimal Design Theory. *Metrika 33* (1986), 337–347.

[5] Calcagno, V. *glmulti: Model selection and multimodel inference made easy*, 2012. R package version 1.0.4, `http://CRAN.R-project.org/package=glmulti`.

[6] Carey, G. The General Linear Model: Theory. Lecture Script, University of Colorado, Boulder, 1998.

[7] Clark, A. Types of Sums of Squares, 2011. `http://afni.nimh.nih.gov/sscc/gangc/SS.html`.

[8] Cook, R. D., and Nachtsheim, C. J. A Comparison of Algorithms for Constructing Exact D-Optimal Designs. *Technometrics 22* (1980), 315–324.

[9] de Aguiar, P., Bourguignon, B., Khots, M., Massart, D., and Phan-Than-Luu, R. Tutorial D-Optimal Designs. *Chemometrics and Intelligent Laboratory Systems 30* (1995), 199–210.

[10] Dykstra, O. The Argumentation of Experimental Data to Maximize $|X'X|$. *Biometrika 13* (1971), 682–688.

[11] Everitt, B. S., and Hothorn, T. *A Handbook of Statistical Analyses Using R.* CRC Press, 2006.

[12] Fedorov, V. *Theory of Optimal Experiments.* Probability and Mathematical Statistics. Academic Press, New York, NY, 1972. Translated and edited by W.J. Studden and E.M. Klimko.

[13] Fischnaller, M., Lohmann, N., Melbert, J., Lamp, P., and Scharner, S. Alterungsuntersuchung und Modellierung von Li-Ionen Zellen für Hybridfahrzeuge. *VDI-Berichte* (2011).

[14] Fox, J., and Weisberg, S. *An R Companion to Applied Regression*, 2nd ed. Sage, Thousand Oaks, CA, 2011. `http://socserv.socsci.mcmaster.ca/jfox/Books/Companion`.

[15] Grömping, U. CRAN Task View: Design of Experiments (DoE) & Analysis of Experimental Data, 2011. `http://cran.r-project.org/web/views/ExperimentalDesign.html`.

[16] Grömping, U. *DoE.wrapper: Wrapper package for design of experiments functionality*, 2011. R package version 0.8-6, `http://CRAN.R-project.org/package=DoE.wrapper`.

[17] Grömping, U. Industrial Design of Experiments in R. Proceedings of the International Conference in Honor of the late Jagdish Srivastava, Center of Experimental Design, pp. 42–58.

[18] Halekoh, U., and Højsgaard, S. Generalized Linear Models. Lecture Script, University of Aarhus, 2007.

[19] Hanneman, R. Linear Models - Analyzing unbalanced data, basic methods (Littell, chapter 5). `http://faculty.ucr.edu/~hanneman/linear\_models/c5.html`.

[20] Happacher, M. Exact and Approximate D-Optimal Designs in Polynomial Regression. *Metrika 42* (1995), 19–27.

[21] Haselgruber, N. *Sampling and Design of Large-Scale Life Time Experiments*. PhD thesis, Graz University of Technology, 2007.

[22] Haselgruber, N. A Modular Algorithm for Dynamic Design of Large-Scale Experiments. *Austrian Journal of Statistics 37*, 3 (2008), 229–244.

[23] John, R., and Draper, N. D-Optimality for Regression Designs: A Review. *Technometrics 17* (1975), 15–23.

[24] Johnson, M. E., and Nachtsheim, C. J. Some Guidelines for Constructing Exact D-Optimal Designs on Convex Design Spaces. *Technometrics 25* (1983), 271–277.

[25] Jones, B., and Nachtsheim, C. J. Efficient Designs with Minimal Aliasing. *Technometrics 53* (2011), 62–71.

[26] Kiefer, J. General Equivalence Theory for Optimum Designs (Approximate Theory). *The Annals of Statistics 2* (1974), 849–879.

[27] Kiefer, J., and Wolfowitz, J. The Equivalence of Two Extremum Problems. *Canadian Journal of Mathematics 12* (1960), 363–366.

[28] King, W. B. Factorial Between Subjects Anova. http://ww2.coastal.edu/kingw/statistics/R-tutorials/factorial.html.

[29] Kuhn, M. *The Desirability Package*, 2009. http://cran.r-project.org/web/packages/desirability/vignettes/desirability.pdf.

[30] Kuhn, M. *Desirability: Desirability Function Optimization and Ranking*, 2012. R package version 1.05, http://CRAN.R-project.org/package=desirability.

[31] Kuhn, M., Weston, S., Wing, J., and Forester, J. *The Contrast Package*, 2011. http://cran.r-project.org/web/packages/contrast/vignettes/contrast.pdf.

[32] Kuhnert, P., and Venables, B. An Introduction to R: Software for Statistical Modelling & Computing. Lecture Notes of the Authors, 2005.

[33] Lalanne, C. *R Companion to Montgomery's Design and Analysis of Experiments*, 2006. http://www.aliquote.org/articles/tech/dae/dae.pdf.

[34] Langsrud, Ø. Anova for Unbalanced Data: Use Type II Instead of Type III Sums of Squares. *Statistics and Computing 13* (2003), 163–167.

[35] Lawson, J. *Package 'Vdgraph'*, 2012. R package version 2.0-1, http://CRAN.R-project.org/package=Vdgraph.

[36] Lenth, R. V. Response-Surface Methods in R, Using rsm. *Journal of Statistical Software 32*, 7 (2009), 1–17.

[37] Lenth, R. V. *Surface Plots in the RSM Package*, 2010. Companion to RSM Package, http://cran.r-project.org/web/packages/rsm/vignettes/rsm-plots.pdf.

[38] Ligges, U., and Mächler, M. Scatterplot3d - an R Package for Visualizing Multivariate Data. *Journal of Statistical Software 8*, 11 (2003), 1–20.

[39] Mazerolle, M. J. *AICcmodavg: Model selection and multimodel inference based on (Q)AIC(c)*, 2012. R package version 1.24, http://CRAN.R-project.org/package=AICcmodavg.

[40] Meyer, R. M., and Nachtsheim, C. J. The Coordinate-Exchange Algorithm for Constructing Exact Optimal Experimental Designs. *Technometrics 37* (1995), 60–69.

[41] Minitab Documentation. Technical Support Document - Variance Dispersion Graphs macro. http://www.minitab.com/support/documentation/answers/Variance%20Dispersion%20Graphs.pdf.

[42] Mitchell, T. J. An Algorithm for the Construction of 'D-Optimal' Experimental Designs. *Technometrics 16* (1974), 203–210.

[43] Montgomery, D. *Design and Analysis of Experiments*, 6th ed. Wiley, Hoboken, NJ, 2005.

[44] Myers, R., and Montgomery, D. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed. Wiley series in probability and statistics. Wiley, New York, NY, 2002.

[45] Möller, C., and Winter, M. Primäre und Wiederaufladbare Lithium-Batterien. Lecture Script, Graz University of Technology, 2005.

[46] Nau, R. F. Testing the Assumptions of Linear Regression, 2012. `http://www.duke.edu/~rnau/testing.htm`.

[47] NIST/SEMATECH. How Can I Tell if a Model Fits my Data? e-Handbook of Statistical Methods. `http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd44.htm`.

[48] Noble, J. M. Multiple Linear Regression and Experimental Design. Lecture Script, Linköping University, 2011.

[49] Prochazka, W., Pregartner, G., and Cifrain, M. Design-Of-Experiment And Statistical Validation Of A Large Scale Ageing Experiment For Two Popular Lithium Ion Cell Chemistries. Technical Report, Vif, 2012.

[50] Pukelsheim, F., and Rieder, S. Efficient Rounding of Approximate Designs. *Biometrika 79* (1992), 763–770.

[51] R Development Core Group. *Graphics with R.* `http://csg.sph.umich.edu/docs/R/graphics-1.pdf`.

[52] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0, `http://www.R-project.org/`.

[53] Reeves, C. R., and Wright, C. Genetic Algorithms and The Design of Experiments. In *In D.Whitley (1997) Proc. IMA Fall Workshop on Evolutionary Algorithms* (1996).

[54] Safari, M., and Delacourt, C. Aging of a Commercial Graphite/$LiFePO_4$ Cell. *Journal of The Electrochemical Society* (2011).

[55] Sarkar, D. *Lattice: Multivariate Data Visualization with R.* Springer, New York, NY, 2008. ISBN 978-0-387-75968-5.

[56] Sarkar, D., and Andrews, F. *latticeExtra: Extra Graphical Utilities Based on Lattice*, 2011. R package version 0.6-19, `http://CRAN.R-project.org/package=latticeExtra`.

[57] Silvey, S., Titterington, D., and Torsney, B. An Algorithm for Optimal Designs on a Finite Design Space. *Communications in Statistics - Theory and Methods 7* (1978), 1379–1389.

[58] Stadlober, E. Versuchsplanung. Lecture Script, Institut of Statistics, Graz University of Technology, 2012.

[59] Stadlober, E., and Schauer, J. Statistik. Lecture Script, Institut of Statistics, Graz University of Technology, 2008.

[60] Tamura, R. *Rdonlp2 - an R interface to DONLP2*, 2007. http://svitsrv25.epfl.ch/R-doc/library/Rdonlp2/doc/tutorial.pdf.

[61] Triefenbach, F. Design of Experiments: The D-Optimal Approach and Its Implementation As a Computer Algorithm. Bachelor's thesis, Umeå University, 2008.

[62] University of Minnesota. Balanced vs. Unbalanced Data. http://www1.umn.edu/statsoft/doc/statnotes/stat06.txt.

[63] van de Geer, S. Least Squares Estimation. *Encyclopedia of Statistics in Behavioral Science 2* (2005), 1041–1045.

[64] van der Vaart, A. W. Contrasts in R. Lecture Script, Amsterdam University, 2011.

[65] Various. [R] Checking for orthogonal contrasts, December 2010. https://stat.ethz.ch/pipermail/r-help/2010-December/262178.html.

[66] Venables, W. N., and Ripley, B. D. *Modern Applied Statistics with S*, 4th ed. Springer, New York, NY, 2002. ISBN 0-387-95457-0.

[67] Vikneswaran. *An R companion to Experimental Design*, 2005. http://cran.r-project.org/doc/contrib/Vikneswaran-ED_companion.pdf.

[68] Warnes., G. R. *gregmisc: Greg's Miscellaneous Functions*, 2011. R package version 2.1.2, http://CRAN.R-project.org/package=gregmisc.

[69] Wheeler, B. *AlgDesign: Algorithmic Experimental Design*, 2011. R package version 1.1-7, http://CRAN.R-project.org/package=AlgDesign.

[70] Wheeler, R. E. *Comments on Algorithmic Design*, 2004. http://cran.r-project.org/web/packages/AlgDesign/vignettes/AlgDesign.pdf.

[71] Wollschläger, D. Sum of Squares Type I, II, III: the underlying hypotheses, model comparisons, and their calculation in R, March 2011. http://www.uni-kiel.de/psychologie/dwoll/r/ssTypes.php.

[72] Wuertz, D., et al. *fRegression: Regression Based Decision and Prediction*, 2009. R package version 2100.76, http://CRAN.R-project.org/package=fRegression.

[73] Wynn, H. P. The Sequential Generation of D-Optimum Experimental Designs. *Annals of Mathematical Statistics 41* (1970), 1655–1664.

[74] Wynn, H. P. Results in the Theory and Construction of D-optimum Experimentals Designs. *Journal of the Royal Statistical Society Vol. 34, No. 2* (1972), 133–147.

[75] Yu, Y. D-Optimal Designs Via a Cocktail Algorithm. *Statistics and Computing 21* (2011), 475–481.

[76] Zeileis, A., and Hothorn, T. Diagnostic Checking in Regression Relationships. *R News 2*, 3 (2002), 7–10.