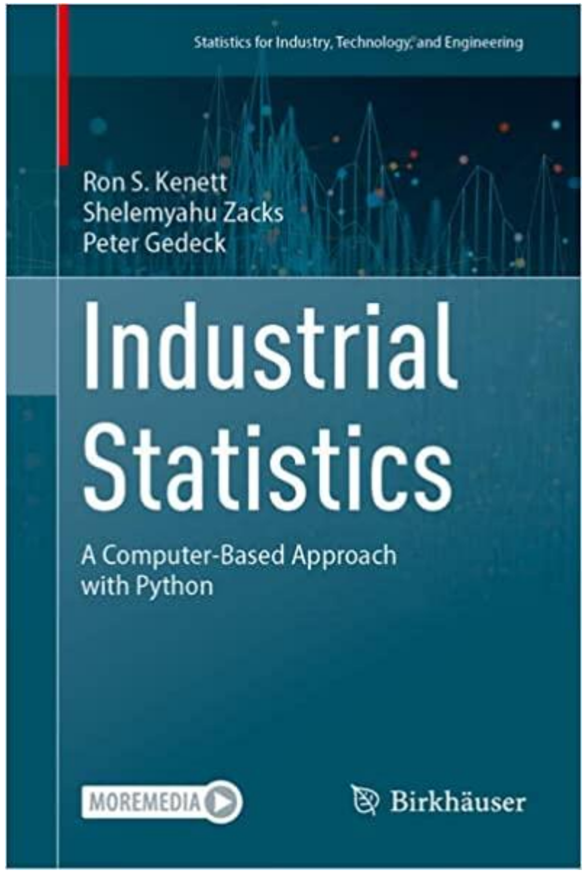


Industrial Statistics:

A Computer-Based Approach with Python

https://link.springer.com/book/9783031284816 https://www.springer.com/series/15677

Book series

Statistics for Industry, Technology, and Engineering

Editorial board

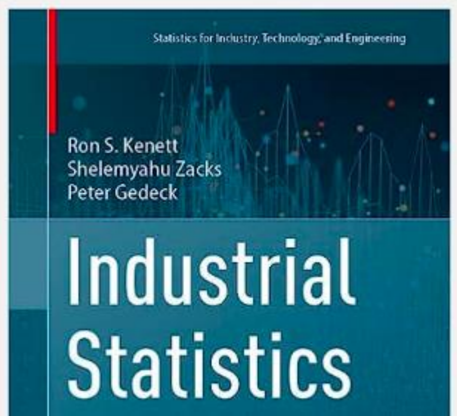
About this book series

The *Statistics for Industry, Technology, and Engineering* series will present up-to-date statistical ideas and methods that are relevant to researchers and accessible to an interdisciplinary audience: carefully organized authoritative presentations, numerous illustrative examples based on current practice, reliable methods, realistic data sets, and discussions of select new emerging methods and their application potential. Publications will appeal to a broad interdisciplinary readership including both researchers and practitioners in applied statistics, data science, industrial statistics, engineering statistics, quality control, manufacturing, applied reliability, and general quality improvement methods. — show all https://link.springer.com/book/9783031284816

This innovative textbook presents material for a course on industrial statistics that incorporates Python as a pedagogical and practical resource. Drawing on many years of teaching and conducting research in various applied and industrial settings, the authors have carefully tailored the text to provide an ideal balance of theory and practical applications. Numerous examples and case studies are incorporated throughout, and comprehensive Python applications are illustrated in detail. A custom Python package is available for download, allowing students to reproduce these examples and explore others.

https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/

Code repository



Industrial Statistics: A Computer Based Approach with Python
by Ron Kenett, Shelemyahu Zacks, Peter Gedeck

Publisher: Springer International Publishing; 1st edition (August 5, 2023)
ISBN-13: 978-3-031-28481-6 (hardcover)
ISBN-13: 978-3-031-28484-7 (softcover)
ISBN-13: 978-3-031-28482-3 (eBook).

Buy at Springer, Amazon,

The first chapters of the text focus on the basic tools and principles of process control, methods of statistical process control (SPC), and multivariate SPC. Next, the authors explore the design and analysis of experiments, quality control and the Quality by Design approach, computer experiments, and cybermanufacturing and digital twins. The text then goes on to cover reliability analysis, accelerated life testing, and Bayesian reliability estimation and prediction. A final chapter considers sampling techniques and measures of inspection effectiveness. Each chapter includes exercises, data sets, and applications to supplement learning.

Industrial Statistics: A Computer-Based Approach with Python is intended for a one- or two-semester advanced undergraduate or graduate course. In addition, it can be used in focused workshops combining theory, applications, and Python implementations. Researchers, practitioners, and data scientists will also find it to be a useful resource with the numerous applications and case studies that are included.

A second, closely related textbook is titled *Modern Statistics: A Computer-Based Approach with Python*. It covers topics such as probability models and distribution functions, statistical inference and bootstrapping, time series analysis and predictions, and supervised and unsupervised learning. These texts can be used independently or for consecutive courses.

This book is part of an impressive and extensive write up enterprise (roughly 1,000 pages!) which led to two books published by Birkhäuser. This book is on Industrial Statistics, an area in which the authors are recognized as major experts. The book combines classical methods (never to be forgotten!) and “hot topics” like cyber manufacturing, digital twins, A/B testing and Bayesian reliability. It is written in a very accessible style, focusing not only on HOW the methods are used, but also on WHY. In particular, the use of Python, throughout the book is highly appreciated. Python is probably the most important programming language used in modern analytics. The authors are warmly thanked for providing such a state-of-the-art book. It provides a comprehensive illustration of methods and examples based on the authors longstanding experience, and accessible code for learning and reusing in classrooms and on-site applications

Professor Fabrizio Ruggeri
Research Director at the National Research Council, Italy
President of the International Society for Business and Industrial Statistics (ISBIS)
Editor-in-Chief of Applied Stochastic Models in Business and Industry (ASMBI)

https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/

Industrial Statistics: A Computer Based Approach with Python is a companion volume to the book Modern Statistics: A Computer Based Approach with Python.

Table of contents (with sample excerpts from chapters)

https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/

Industrial Statistics: A Computer Based Approach with Python is a companion volume to the book Modern Statistics: A Computer Based Approach with Python.

This part of the repository contains:

- `notebooks` : Python code of individual chapters in Jupyter notebooks - download all as notebooks.zip
- `code` : Python code for solutions as plain Python files - download all as code.zip
- `solutions manual` : Solutions_IndustrialStatistics.pdf: solutions of exercises
- `solutions` : Python code for solutions in Jupyter notebooks - download all as solutions.zip
- `all` : zip file with all files combined - download all as all.zip
- `datafiles` : zip file with all data files - download all as data_files.zip - the `mistat` package gives you already access to all datafiles, you only need to download this file if you want to use it with different software

All the Python applications referred to in this book are contained in a package called `mistat` available for installation from the Python package index https://pypi.org/project/mistat/. The `mistat` packages is maintained in a GitHub repository at https://github.com/gedeck/mistat.

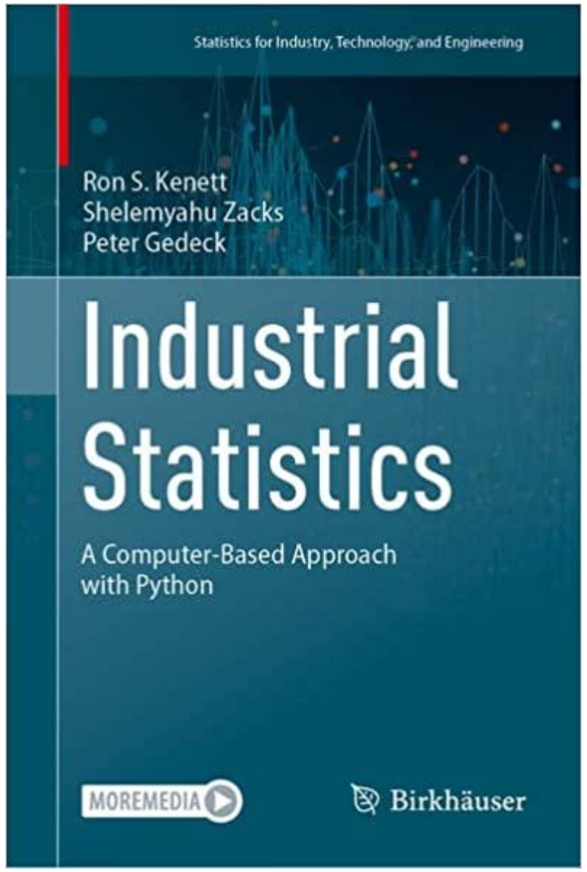
Code repository **Installation instructions** **Excerpts from chapters**

https://gedeck.github.io/mistat-code-solutions/ModernStatistics

- `notebooks` : Python code of individual chapters in Jupyter notebooks - download all as notebooks.zip
- `code` : Python code for solutions as plain Python files - download all as code.zip
- `solutions manual` : Solutions_Modernstatistics.pdf: solutions of exercises
- `solutions` : Python code for solutions in Jupyter notebooks - download all as solutions.zip
- `all` : zip file with all files combined - download all as all.zip

All the Python applications referred to in this book are contained in a package called `mistat` available for installation from the Python package index https://pypi.org/project/mistat/. The `mistat` packages is maintained in a GitHub repository at https://github.com/gedeck/mistat.

Companion book


Statistics for Industry, Technology, and Engineering
Ron S. Kenett
Shelemyahu Zacks
Peter Gedeck
Industrial Statistics
A Computer-Based Approach with Python
MOREMEDIA
Birkhäuser

Chapter 1
The Role of Statistical Methods in Modern Industry

Preview Industrial statistics is a discipline that needs to be adapted and provide enhanced capabilities to modern industrial systems. This chapter presents the evolution of industry and quality in the last 300 years. The transition between the four industrial revolutions is reviewed as well as the evolution of quality from product quality to process, service, management, design, and information quality. To handle the new opportunities and challenges of big data, a current perspective of information quality is presented, including a comprehensive InfoQ framework. The chapter concludes with a presentation of digital twins which are used in industry as a platform for monitoring, diagnostic, prognostic and prescriptive analytics. The Python code used in this and the following chapters is available from `https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics`.

3.4 Cumulative Sum Control Charts

3.4.1 Upper Page's Scheme

When the process level changes from a past or specified level, we expect that a control procedure will trigger an "alarm". Depending on the size of the change and the size of the sample, it may take several sampling periods before the alarm occurs. A method that has a smaller ARL than the standard Shewhart control charts, for detecting certain types of changes, is the **cumulative sum** (or CUSUM) control chart which was introduced by Barnard (1959) and Page (1954).

CUSUM charts differ from the common Shewhart control chart in several respects. The main difference is that instead of plotting the individual value of the statistic of interest, such as X, $\bar{X}$, S, R, p or c, a statistic based on the cumulative sums is computed and tracked. By summing deviations of the individual statistic from a target value, T, we get a consistent increase, or decrease, of the cumulative sum when the process is above, or below, the target. In Fig. 3.4 we show the behavior of the cumulative sums

$$S_t = \sum_{i=1}^{t} (X_i - 10) \tag{3.4.1}$$

of data simulated from a normal distribution with mean

$$\mu_t = \begin{cases} 10, & \text{if } t \leq 20 \\ 13, & \text{if } t > 20 \end{cases}$$

and $\sigma_t = 1$ for all t.

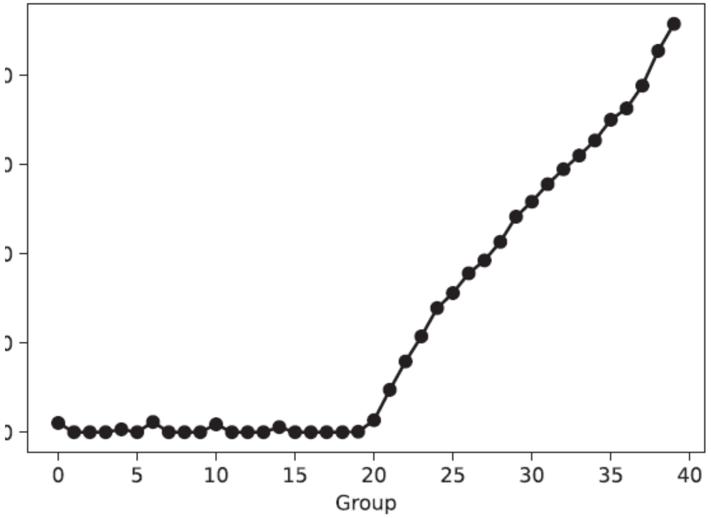
We see that as soon as the shift in the mean of the data occurred, a pronounced drift in S_t started. Page (1954) suggested to detect an upward shift in the mean by considering the sequence

$$S_t^+ = \max\{S_{t-1}^+ + (X_k - K^+), 0\}, \quad t = 1, 2, \cdots \tag{3.4.2}$$

where $S_0^+ \equiv 0$, and decide that a shift has occurred, as soon as $S_t^+ > h^+$. The statistics X_t, $t = 1, 2, \cdots$, upon which the (truncated) cumulative sums are constructed, could be means of samples of n observations, standard deviations, sample proportions, or individual observations. In the following section we will see how the parameters K^+ and h^+ are determined. We will see that if X_t are means of samples of size n, with process variance σ^2, and if the desired process mean is θ_0 while the maximal tolerated process mean is θ_1, $\theta_1 - \theta_0 > 0$, then

$$K^+ = \frac{\theta_0 + \theta_1}{2} \quad \text{and} \quad h^+ = -\frac{\sigma^2 \log \alpha}{n(\theta_1 - \theta_0)}. \tag{3.4.3}$$

$0 < \alpha < 1$.



g. 3.4: A Plot of Cumulative Sums With Drift After $t = 20$

A. Normal Distribution

We consider X_i to be normally distributed with **known** variance σ^2 and mean θ_0 or θ_1. In this case

$$\begin{aligned}\log \frac{f(X_i;\theta_1)}{f(X_i;\theta_0)} &= -\frac{1}{2\sigma^2}\{(X_i-\theta_1)^2-(X_i-\theta_0)^2\} \\ &= \frac{\theta_1-\theta_0}{\sigma^2}\left(X_i-\frac{\theta_0+\theta_1}{2}\right).\end{aligned} \tag{3.4.5}$$

Thus, the criterion

$$\sum_{i=1}^{t} \log \frac{f(X_i;\theta_1)}{f(X_i;\theta_0)} \geq -\log\alpha$$

is equivalent to

$$\sum_{i=1}^{t}\left(X_i-\frac{\theta_0+\theta_1}{2}\right) \geq -\frac{\sigma^2\log\alpha}{\theta_1-\theta_0}.$$

For this reason we use in the upper Page control scheme

$$K^+ = \frac{\theta_0+\theta_1}{2}, \quad \text{and} \quad h^+ = -\frac{\sigma^2\log\alpha}{\theta_1-\theta_0}.$$

If X_t is an average of n independent observations, then we replace σ^2 by σ^2/n.

3.4.3 Lower and Two-Sided Page's Scheme

In order to test whether a significant drop occurred in the process level (mean) we can use a lower page scheme. According to this scheme, we set $S_0^- \equiv 0$ and

$$S_t^- = \min\{S_{t-1}^- + (X_t - K^-), 0\}, \quad t = 1, 2, \cdots . \tag{3.4.12}$$

Here the CUSUM values S_t^- are either zero or negative. We decide that a shift down in the process level, from θ_0 to θ_1, $\theta_1 < \theta_0$, occurred as soon as $S_t^- < h^-$. The control parameters K^- and h^- are determined by the formula of the previous section by setting $\theta_1 < \theta_0$.

Example 3.4 In dataset **COAL.csv** one can find data on the number of coal mine disasters (explosions) in England, per year, for the period 1850 to 1961. These data are plotted in Fig. 3.6. It seems that the average number of disasters per year dropped after 40 years from 3 to 2 and later settled around an average of one per year. We apply here the lower Page's scheme to see when do we detect this change for the first time. It is plausible to assume that the number of disasters per year, X_t, is a random variable having a Poisson distribution. We therefore set $\lambda_0 = 3$ and $\lambda_1 = 1$. The formula of the previous section, with K^+ and h^+ replaced by K^- and h^- yield, for $\alpha = 0.01$,

$$K^- = \frac{\lambda_1 - \lambda_0}{\log(\lambda_1/\lambda_0)} = 1.82 \quad \text{and} \quad h^- = -\frac{\log(0.01)}{\log(1/3)} = -4.19.$$

In Table 3.8 we find the values of X_t $X_t - K^-$ and S_t^- for $t = 1, \cdots, 50$. We see that $S_t^- < h^-$ for the first time at $t = 47$. The graph of S_t^- versus t is plotted in Fig. 3.7. ■

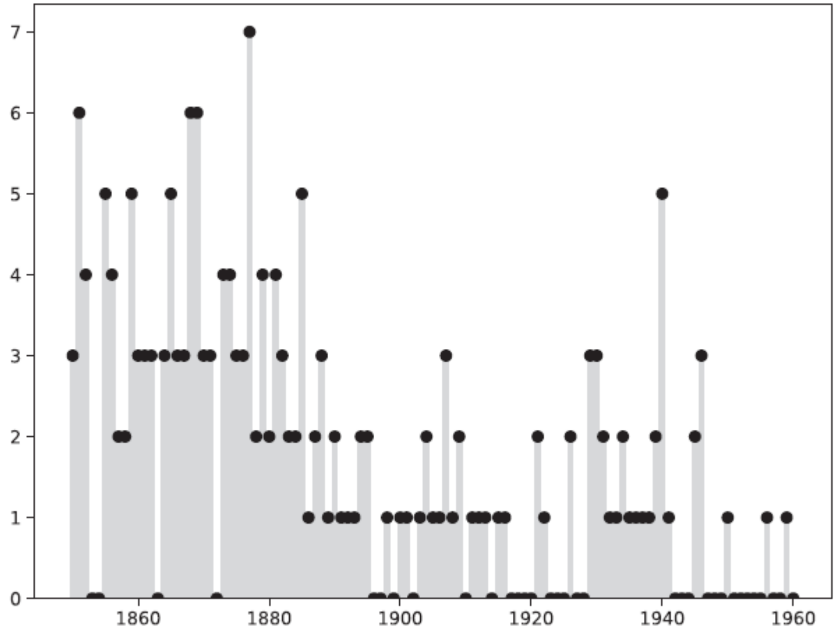


Fig. 3.6: Number of Yearly Coal Mine Disasters in England

If we wish to control simultaneously against changes in the process level in either upwards or downwards directions we use an upper and lower Page's schemes together, and trigger an alarm as soon as either $S_t^+ > h^+$ or $S_t^- < h^-$. Such a two sided scheme is denoted by the four control parameters (K^+, h^+, K^-, h^-).

Table 3.8: Page's Lower Control Scheme For The Coal Mine Disasters Data

t	X_t	$X_t - K^-$	S_t^-
1	3	1.179	0
2	6	4.179	0
3	4	2.179	0
4	0	−1.820	−1.820
5	0	−1.820	−3.640
6	5	3.179	−0.461
7	4	2.179	0
8	2	0.179	0
9	2	0.179	0
10	5	3.179	0
11	3	1.179	0
12	3	1.179	0
13	3	1.179	0
14	0	−1.820	−1.820
15	3	1.179	−0.640
16	5	3.179	0
17	3	1.179	0
18	3	1.179	0
19	6	4.179	0
20	6	4.179	0
21	3	1.179	0
22	3	1.179	0
23	0	−1.820	−1.820
24	4	2.179	0
25	4	2.179	0
26	3	1.179	0
27	3	1.179	0
28	7	5.179	0
29	2	0.179	0
30	4	2.179	0
31	2	0.179	0
32	4	2.179	0
33	3	1.179	0
34	2	0.179	0
35	2	0.179	0
36	5	3.179	0
37	1	−0.820	−0.820
38	2	0.179	−0.640
39	3	1.179	0
40	1	−0.820	−0.820
41	2	0.179	−0.640
42	1	−0.820	−1.461
43	1	−0.820	−2.281
44	1	−0.820	−3.102
45	2	0.179	−2.922
46	2	0.179	−2.743
47	0	−1.820	−4.563
48	0	−1.820	−6.384
49	1	−0.820	−7.204
50	0	−1.820	−9.025

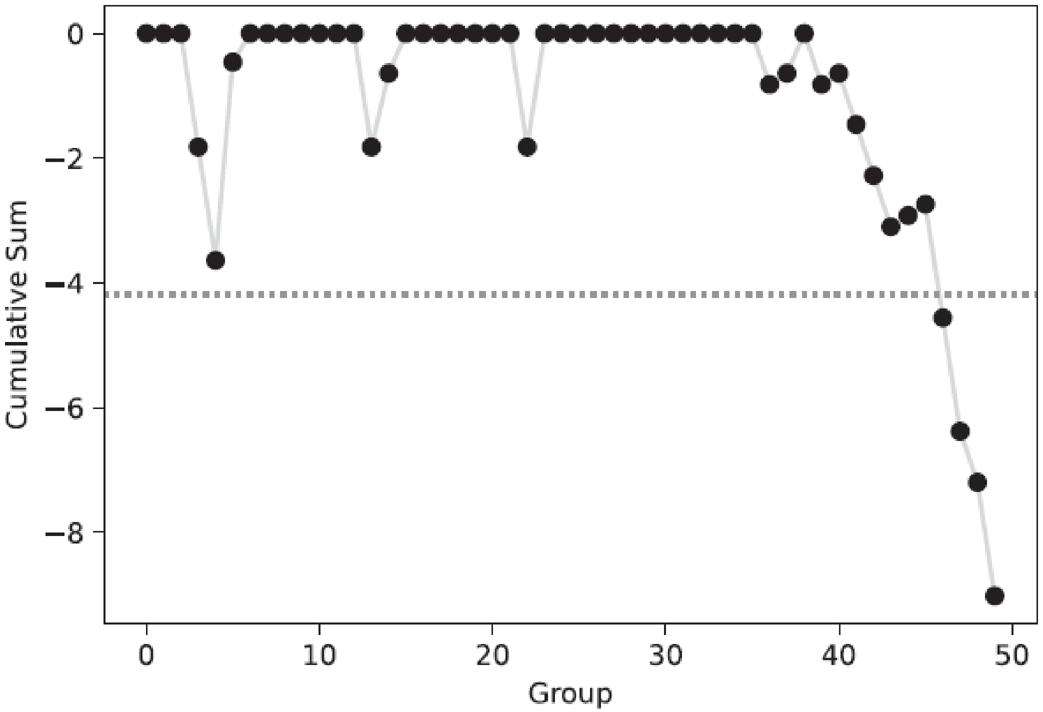


Fig. 3.7: Page's Lower CUSUM Control Chart

The two-sided Page's control scheme can be boosted by changing the values of S_0^+ and S_0^- to non-zero. These are called **headstart values**. The introduction of non-zero headstarts was suggested by Lucas and Crosier (1982) in order to bring the history of the process into consideration, and accelerate the initial response of the scheme. Lucas (1982) suggested also to combine the CUSUM scheme with the Shewhart Control Chart. If any X_t value exceeds an upper limit UCL, or falls below a lower limit LCL, an alarm should be triggered.

3.4.4 Average Run Length, Probability of False Alarm And Conditional Expected Delay

The **run length** (RL) is defined as the number of time units until either $S_t^+ > h_t^+$ or $S_t^- < h^-$, for the first time. We have seen already that the **average run length** (ARL) is an important characteristic of a control procedure, when there is either no change in the mean level (ARL(0)), or the mean level has shifted to $\mu_1 = \mu_0 + \delta\sigma$, before the control procedure started (ARL(δ)). When the shift from μ_0 to μ_1 occurs at some change point τ, $\tau > 0$, then we would like to know what is the probability of false alarm, i.e., that the run length is smaller than τ, and the conditional expected run length, given that RL $> \tau$. It is difficult to compute these characteristics of the Page control scheme analytically. The theory required for such an analysis is quite complicated (see Yashchin, 1985). We provide Python methods in the `mistat` package which approximate these characteristics numerically by simulation.

The method *cusumArl* computes the average run length, ARL, and *cusumPfaCed* returns the probability of false alarm, FPA, and conditional expected delay, CED, for a given distribution, e.g. normal, binomial, or Poisson.

In Table 3.10 we present estimates of the ARL(δ) for the normal distribution, with NR = 100 runs. S.E. = standard-deviation(RL)/$\sqrt{\text{NR}}$.

Program **cusumArl** can be used also to determine the values of the control parameters h^+ and h^- so that a certain ARL(0) is attained. For example, if we use the Shewhart 3-sigma control charts for the sample means in the normal case, the probability that, under no shift in the process level, a point will fall outside the control limits is 0.0026, and ARL(0) = 385. Suppose we wish to devise a two-sided CUSUM control scheme, when $\mu_0 = 10$, $\sigma = 5$, $\mu_1^+ = 14$, $\mu_1^- = 6$. We obtain $K^+ = 12$, $K^- = 8$. If we take $\alpha = 0.01$ we obtain $h^+ = \frac{-25 \times \log(0.01)}{4} = 28.78$. Program **cusumArl** yields, for the parameters $\mu = 10$, $\sigma = 5$, $K^+ = 12$, $h^+ = 29$, $K^- = 8$, $h^- = -29$ the estimate ARL(0) = 411 ± 33.6. If we use $\alpha = .05$ we obtain $h^+ = 18.72$. Under the control parameters (12, 18.7, 8, -18.7) we obtain ARL(0) = 70.7 ± 5.5. We can now run the program for several $h^+ = -h^-$ values to obtain an ARL(0) estimate close to 385. The value in Fig. 3.9 ARL is 411.4 with SE of 33.6.

```
for h in (18.7, 28, 28.5, 28.6, 28.7, 29, 30):
  arl = mistat.cusumArl(randFunc=stats.norm(loc=10, scale=5),
         N=300, limit=7000, seed=1, kp=12, km=8, hp=h, hm=-h,
         verbose=False)
  print(f"h {h:5.1f}: ARL(0) {arl['statistic']['ARL']:5.1f} ",
       f"+/- {arl['statistic']['Std. Error']:4.1f}")
```

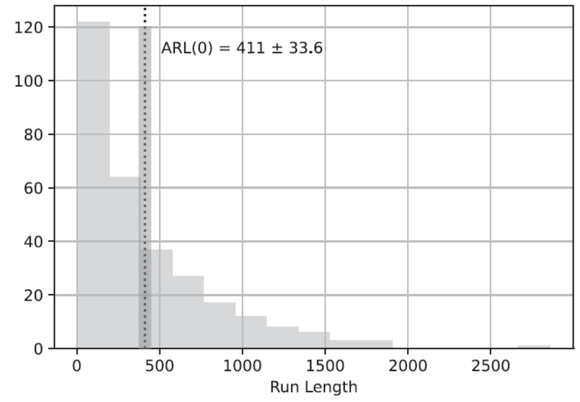


Fig. 3.9: Histogram of RL for $\mu = 10$, $\sigma = 5$, $K^+ = 12$, $h^+ = 29$, $K^- = 8$, $h^- = -29$.

Chap003.ipynb - Notepad

File Edit Format View Help


```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "6038d2b2",
   "metadata": {},
   "source": [
    "# Chapter 3 \n",
    " Industrial Statistics: A Computer Based Approach with Python<br>\n",
    "by Ron Kenett, Shelemyahu Zacks, Peter Gedeck\n",
    "\n",
    "Publisher: Springer International Publishing; 1st edition (2023) <br>\n",
    "<!-- ISBN-13: 978-3031075650 -->\n",
    "\n",
    "(c) 2022 Ron Kenett, Shelemyahu Zacks, Peter Gedeck\n",
    "\n",
    "The code needs to be executed in sequence."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "af3bfca3",
   "metadata": {
```


Chap003.ipynb

https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/

Ln 1, Col 1 | 100% | Unix (LF) | UTF-8

Chapter 3
Advanced Methods of Statistical Process Control

Industrial Statistics: A Computer Based Approach with Python

Solutions

For the most recent version of the solution manual, go to https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/.

Import required modules and define required functions

```
import numpy as np
import pandas as pd
from scipy import stats
from scipy.special import factorial
import statsmodels.formula.api as smf
import mistat
import matplotlib.pyplot as plt
```

Exercise 3.1 Generate the distribution of the number of runs in a sample of size $n = 25$, if the number of elements above the sample mean is $m_2 = 10$.

(i) What are Q_1, Me and Q_3 of this distribution?
(ii) Compute the expected value, μ_R, and the standard deviation σ_R.
(iii) What is $\Pr\{10 \le R \le 16\}$?
(iv) Determine the normal approximation to $\Pr\{10 \le R \le 16\}$.

Solution 3.1 First define a function to calculate the probability of runs:

```
from scipy.special import binom
def probRuns(m1, m2, R):
    n = m1 + m2
    k = R // 2
    if R % 2:
        denom = binom(m1-1,k-1) * binom(m2-1,k) + binom(m1-1,k) * binom(m2-1,k-1)
        return denom / binom(n, m2)
    else:
        return 2 * binom(m1-1, k-1) * binom(m2-1, k-1) / binom(n, m2)
```

For the case of $n = 25$ and $m_2 = 10$ calculate the distribution

```
n = 25
m2 = 10
m1 = n - m2
```

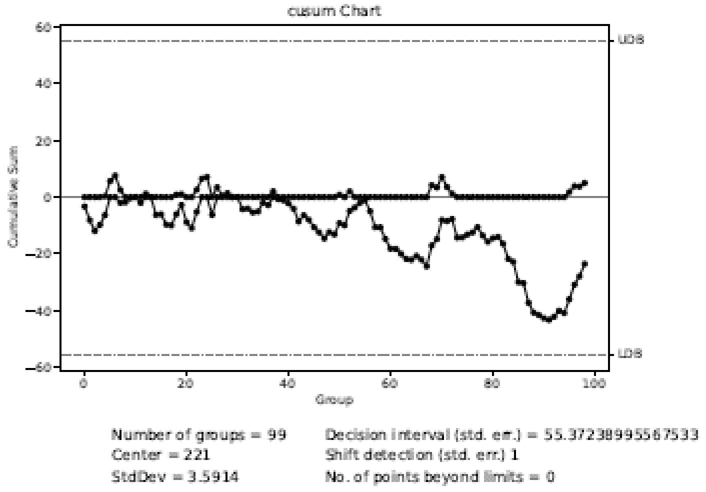



Fig. 3.7: CUSUM chart for dataset **OELECT.csv**

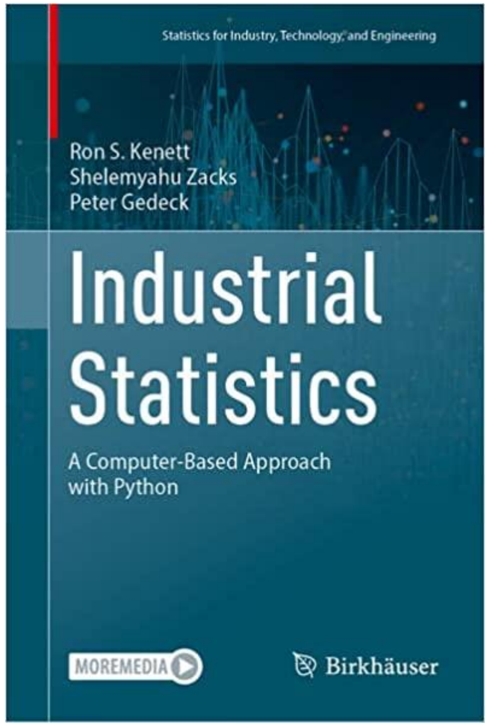
```
lambda0 = 15
lambda1_p = 25
lambda1_m = 7
alpha = 0.001
tau = 30

kp = (lambda1_p - lambda0) / np.log(lambda1_p/lambda0)
hp = - np.log(alpha) / np.log(lambda1_p/lambda0)
km = (lambda1_m - lambda0) / np.log(lambda1_m/lambda0)
hm = - np.log(alpha) / np.log(lambda1_m/lambda0)

print(f'Kp={kp:.3f}, hp={hp:.3f}')
print(f'Km={km:.3f}, hm={hm:.3f}')

arl = mistat.cusumPfaCed(randFunc1=stats.poisson(mu=15),
                         randFunc2=stats.poisson(mu=25),
                         tau=tau,
                         kp=kp, km=km,
                         hp=hp, hm=hm,
                         N=4000, limit=1000, seed=1)
result = arl['statistic']
```

```
Kp=19.576, hp=13.523
Km=10.497, hm=-9.064
PFA 0.01075  CED 2.1663  Std. Error 0.51072
```

Exercise 3.20 A CUSUM control scheme is based on sample means.

(i) Determine the control parameters K^+, h^+, K^-, h^-, when $\mu_0 = 100$, $\mu_1^+ = 110$, $\mu_1^- = 90$, $\sigma = 20$, $n = 5$, $\alpha = 0.001$.

Chapter 5
Classical Design and Analysis of Experiments



Preview Experiments are used in industry to improve productivity, reduce variability, enhance quality and obtain robust products and manufacturing processes. In this chapter we study how to design and analyze experiments which are aimed at testing scientific or technological hypotheses. These hypotheses are concerned with the effects of procedures or treatments on quality and productivity; or the general relationship between variables. Designed experiments help determine the conditions under which a production process yields maximum output or other optimum results, etc. The chapter presents the classical methods of design of experiments. It starts with an introductory section with examples and discusses guiding principles in designing experiments. The chapter covers the range of classical experimental designs including complete block designs, Latin squares, full and fractional factorial designs with factors at two and three levels. The basic approach to the analysis is through modeling the response variable and computing ANOVA tables. Particular attention is given to the generation of designs using Python.

5.4.1.2 Randomization Tests

A randomization test for paired comparison, constructs a **reference distribution** of all possible averages of the differences that can be obtained by randomly assigning the sign + or − to the value of D_i. It computes then an average difference $\bar{D}$ for each one of the 2^n sign assignments.

The P-value of the test, for the two sided alternative, is determined according to this reference distribution, by

$$P = \Pr\{\bar{Y} \geq \text{Observed } \bar{D}\}.$$

For example, suppose we have four differences, with values 1.1, 0.3, −0.7, −0.1. The mean is $\bar{D}_4 = 0.15$. There are $2^4 = 16$ possible ways of assigning a sign to $|D_i|$. Let $X_i = \pm 1$ and $\bar{Y} = \frac{1}{4}\sum_{i=1}^{4} X|D_i|$. The possible combinations are listed in Table 5.1

Table 5.1: Sign Assignments and Values of $\bar{Y}$

Signs				D
-1	-1	-1	-1	-0.55
1	-1	-1	-1	0
-1	1	-1	-1	-0.4
1	1	-1	-1	0.15
-1	-1	1	-1	-0.20
1	-1	1	-1	0.35
-1	1	1	-1	-0.05
1	1	1	-1	0.50
-1	-1	-1	1	-0.50
1	-1	-1	1	0.05
-1	1	-1	1	-0.35
1	1	-1	1	0.2
-1	-1	1	1	-0.15
1	-1	1	1	0.40
-1	1	1	1	0
1	1	1	1	0.55

Under the reference distribution, all these possible means are equally probable. The P-value associated with the observed $\bar{D} = 0.15$ is $P = \frac{7}{15} = 0.47$. If the number of pairs (blocks) n is large the procedure becomes cumbersome, since we have to determine all the 2^n sign assignments. If $n = 20$ there are $2^{20} = 1,048,576$ such assignments. We can however estimate the P-value by taking a RSWR from this reference distribution. In Python this is performed with the following commands:

```
random.seed(1)
X = [1.1, 0.3, -0.7, -0.1]
m = 20000

Di = pd.DataFrame([random.choices((-1, 1), k=len(X)) for _ in range(m)])
DiX = (Di * X)

np.mean(DiX.mean(axis=1) > np.mean(X))
```

```
0.31425
```

5.7 Full Factorial Experiments

5.7.1 The Structure of Factorial Experiments

Full factorial experiments are those in which complete trials are performed of all the combinations of the various factors at all their levels. For example, if there are five factors, each one tested at three levels, there are altogether $3^5 = 243$ treatment combinations. All these 243 treatment combinations are tested. The full factorial experiment may also be replicated several times. The order of performing the trials is random.

In full factorial experiments, the number of levels of different factors do not have to be the same. Some factors might be tested at two levels and others at three or four levels. Full factorial, or certain fractional factorials which will be discussed later, are necessary, if the statistical model is not additive. In order to estimate or test the **effects** of **interactions**, one needs to perform factorial experiments, full or fractional. In a full factorial experiment, all the main effects and interactions can be tested or estimated. Recall that if there are p factors $A, B, C, \cdots$ there are p types of main effects, $\binom{p}{2}$ types of pairwise interactions $AB, AC, BC, \cdots$, $\binom{p}{3}$ interactions between three factors, $ABC, ABD, \cdots$ and so on. On the whole there are, together with the grand mean μ, 2^p types of parameters.

In the following section we discuss the structure of the ANOVA for testing the significance of main effects and interaction. This is followed by a section on the estimation problem. In Sections 5.7.4 and 5.7.5 we discuss the structure of full factorial experiments with 2 and 3 levels per factor, respectively.

5.7.4 2^m Factorial Designs

2^m factorial designs are full factorials of m factors, each one at two levels. The levels of the factors are labelled as “Low” and “High” or 1 and 2. If the factors are categorical then the labelling of the levels is arbitrary and the values of the main effects and interaction parameters depend on this arbitrary labeling. We will discuss here experiments in which the levels of the factors are measured on a continuous scale, like in the case of the factors effecting the piston cycle time. The levels of the i-th factor ($i = 1, \cdots, m$) are fixed at x_{i1} and x_{i2}, where $x_{i1} < x_{i2}$.

By simple transformation all factor levels can be reduced to

$$c_i = \begin{cases} +1, & \text{if } x = x_{i2} \\ \\ -1, & \text{if } x = x_{i1} \end{cases} \qquad , i = 1, \cdots, m.$$

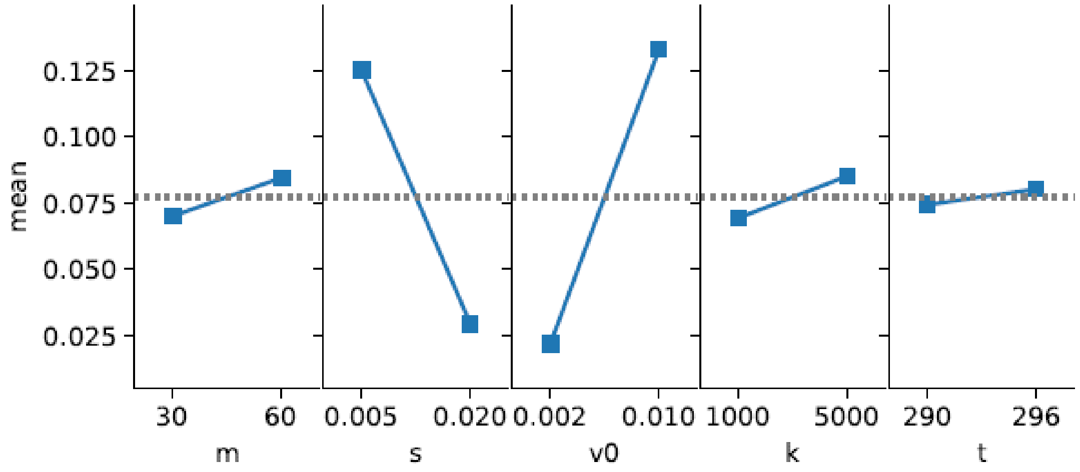
In such a factorial experiment there are 2^m possible treatment combinations. Let $(i_1, \cdots, i_m)$ denote a treatment combination, where $i_1, \cdots, i_m$ are indices, such that

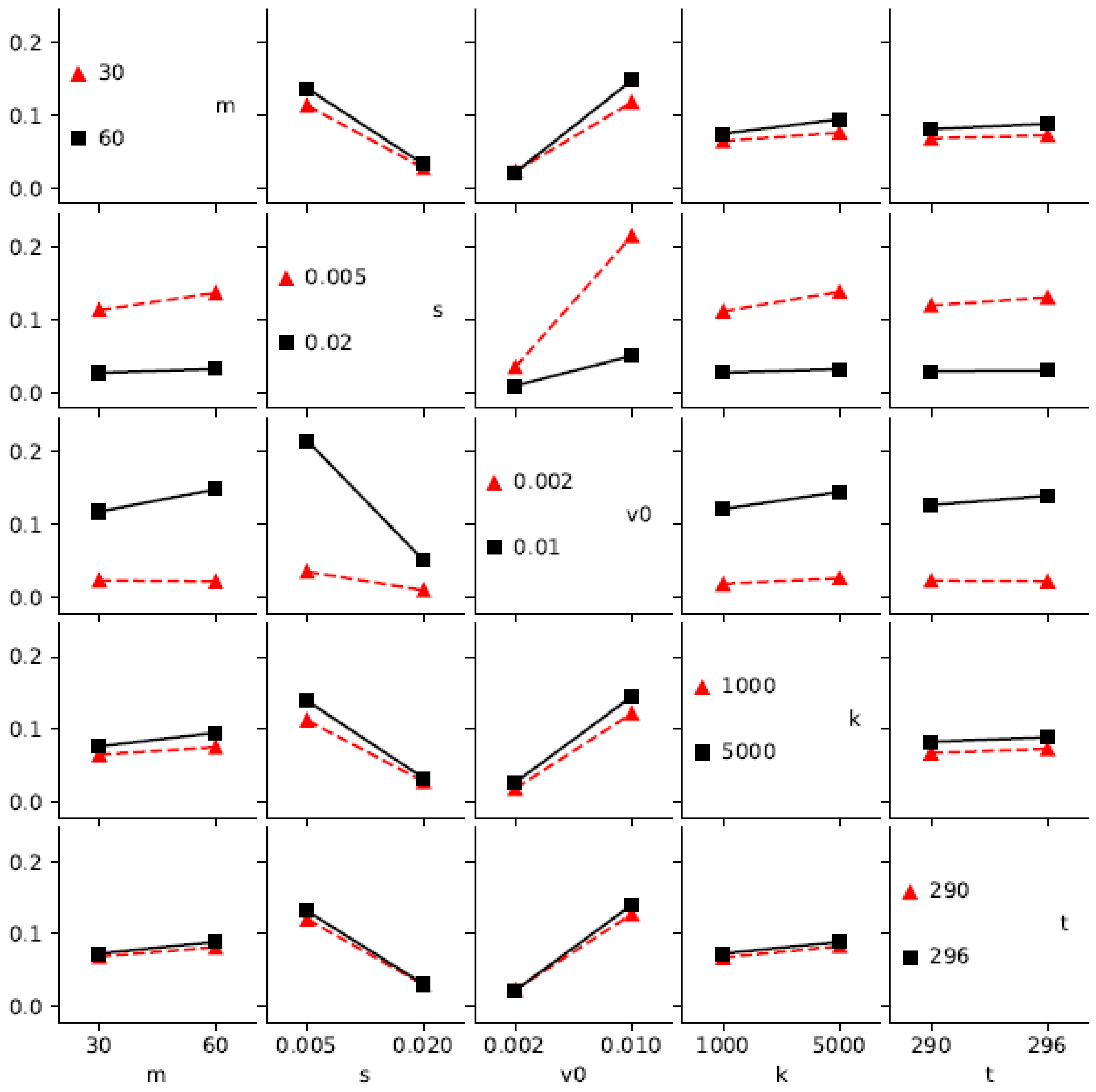
$$i_j = \begin{cases} 0, & \text{if } c_i = -1 \\ \\ 1, & \text{if } c_i = 1. \end{cases}$$

Thus, if there are $m = 3$ factors, the number of possible treatment combinations is $2^3 = 8$. These are given in Table 5.17

Table 5.21: LSE of Main Effects and Interactions

	LSE	S.E.	t	
m	-0.0054	0.00222	-2.44	
s	37.9277	0.00222	17097.52	**
v0	-61.9511	0.00222	-27927.08	**
k	-0.0000	0.00222	-0.00	
t	-0.0001	0.00222	-0.07	
m:s	-0.0413	0.00222	-18.62	**
m:v0	0.1332	0.00222	60.03	**
m:k	0.0000	0.00222	0.00	
m:t	0.0000	0.00222	0.01	
s:v0	-1165.8813	0.00222	-525569.93	**
s:k	-0.0004	0.00222	-0.17	
s:t	-0.1173	0.00222	-52.86	**
v0:k	0.0005	0.00222	0.21	
v0:t	0.2834	0.00222	127.76	**
k:t	0.0000	0.00222	0.00	





0.0
0.1
0.2
30
60
m
0.005
0.02
s
0.002
0.01
v0
1000
5000
k
290
296
t
0.005
0.020
0.002
0.010

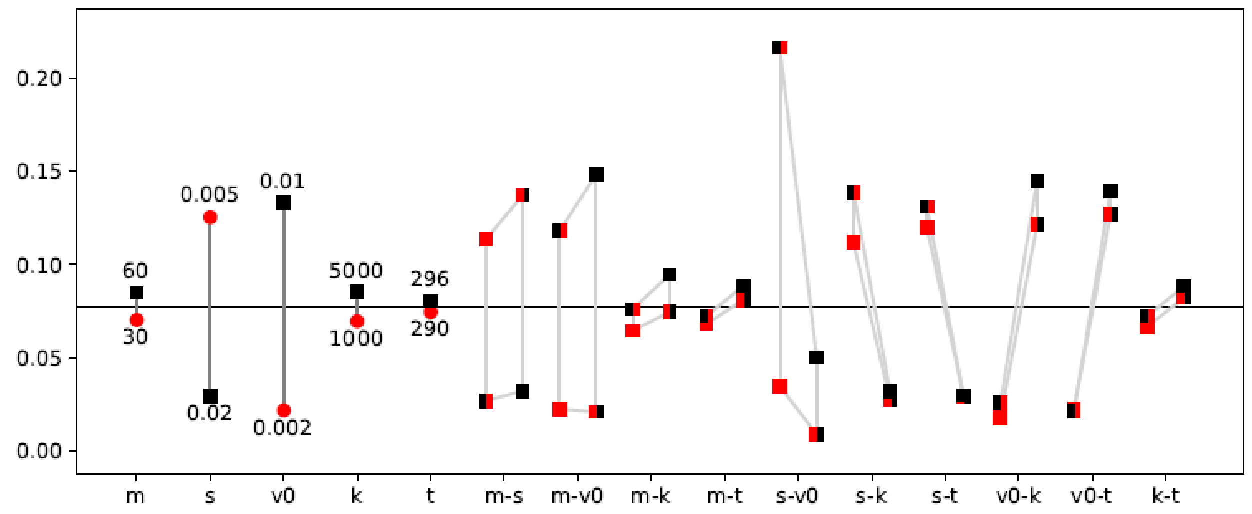
is in practice lost anyway.

A full factorial experiment is a combination of fractional factorial designs. In Python we obtain a fraction of a full factorial design with the `mistat` package.

```
d1 = {
    'A': [-1, 1],
    'B': [-1, 1],
    'C': [-1, 1],
    'D': [-1, 1],
    'E': [-1, 1],
}
mistat.addTreatments(doe.frac_fact_res(d1, 4), mainEffects=['A', 'B', 'C', 'D', 'E'])
```

```
   Treatments  A  B  C  D  E
0         (1) -1 -1 -1 -1 -1
1          AE  1 -1 -1 -1  1
2          BE -1  1 -1 -1  1
3          AB  1  1 -1 -1 -1
4          CE -1 -1  1 -1  1
5          AC  1 -1  1 -1 -1
6          BC -1  1  1 -1 -1
7        ABCE  1  1  1 -1  1
8           D -1 -1 -1  1 -1
9         ADE  1 -1 -1  1  1
10        BDE -1  1 -1  1  1
11        ABD  1  1 -1  1 -1
12        CDE -1 -1  1  1  1
13        ACD  1 -1  1  1 -1
14        BCD -1  1  1  1 -1
15      ABCDE  1  1  1  1  1
```

Fig. 5.6: Two-Way Interaction Plots



This is a half fractional replications of a 2^5 designs as will be explained in Section 5.8. In Table 5.18 we present the design of a 2^5 full factorial experiment derived using Python.

5.7.5 3^m Factorial Designs

We discuss here the estimation and testing of model parameters, when the design is full factorial, of m factors each one at $p = 3$ levels. We assume that the levels are measured on a continuous scale, and are labelled Low, Medium and High. We introduce the indices i_j ($j = 1, \cdots, m$), with values 0, 1, 2 for the Low, Medium and High levels, correspondingly, of each factor. Thus, we have 3^m treatment combinations, represented by vectors of indices $(i_1, i_2, \cdots, i_m)$. The index ν of the **standard order** of treatment combination is

$$\nu = \sum_{j=1}^{m} i_j 3^{j-1}. \tag{5.7.42}$$

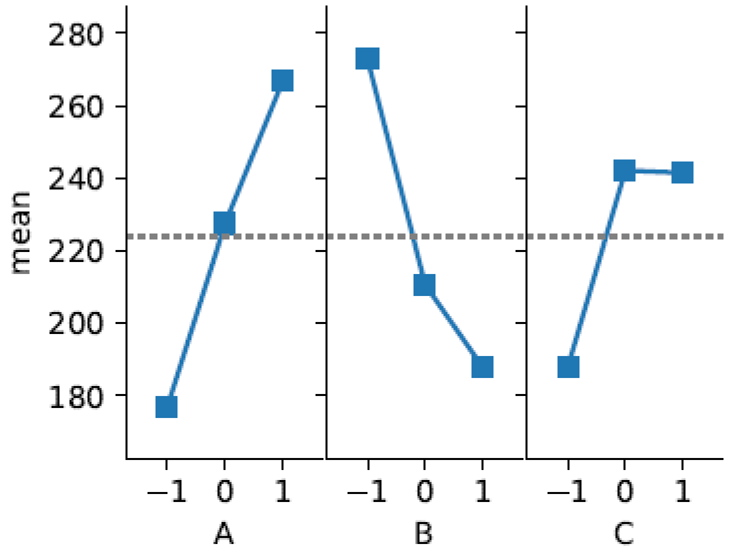
This index ranges from 0 to $3^m - 1$. Let $\bar{Y}_\nu$ denote the yield of n replicas of the ν-th treatment combination, $n \geq 1$.

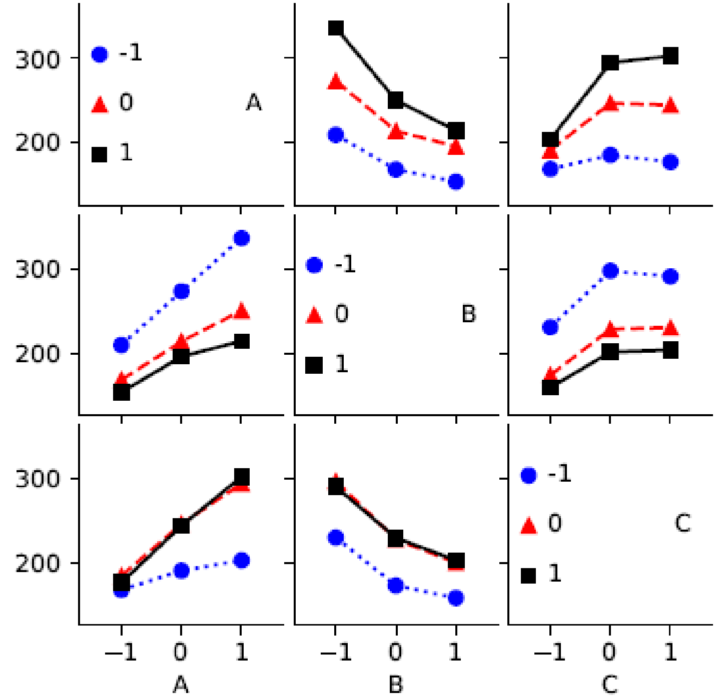
Since we obtain the yield at three levels of each factor we can, in addition to the linear effects estimate also the quadratic effects of each factor. For example, if we have $m = 2$ factors, we can use a multiple regression method to fit the model

$$\begin{aligned} Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_1 x_2 + \\ \beta_5 x_1^2 x_2 + \beta_6 x_2^2 + \beta_7 x_1 x_2^2 + \beta_8 x_1^2 x_2^2 + e. \end{aligned} \tag{5.7.43}$$

This is a quadratic model in two variables. β_1 and β_3 represent the linear effects of x_1 and x_2. β_2 and β_6 represent the quadratic effects of x_1 and x_2. The other coefficients represent interaction effects. β_4 represents the linear $\times$ linear interaction, β_5 represents the quadratic $\times$ linear interaction, etc. We have two main effects for each factor (linear and quadratic) and 4 interaction effects.

Generally, if there are m factors we have, in addition to β_0, $2m$ parameters for main effects (linear and quadratic) $2^2\binom{m}{2}$ parameters for interactions between 2 factors, $2^3\binom{m}{3}$ interactions between 3 factors, etc. Generally, we have 3^m parameters, where





5.9 Exploration of Response Surfaces

The functional relationship between the yield variable Y and the experimental variables $(x_1, \cdots, x_k)$ is modeled as

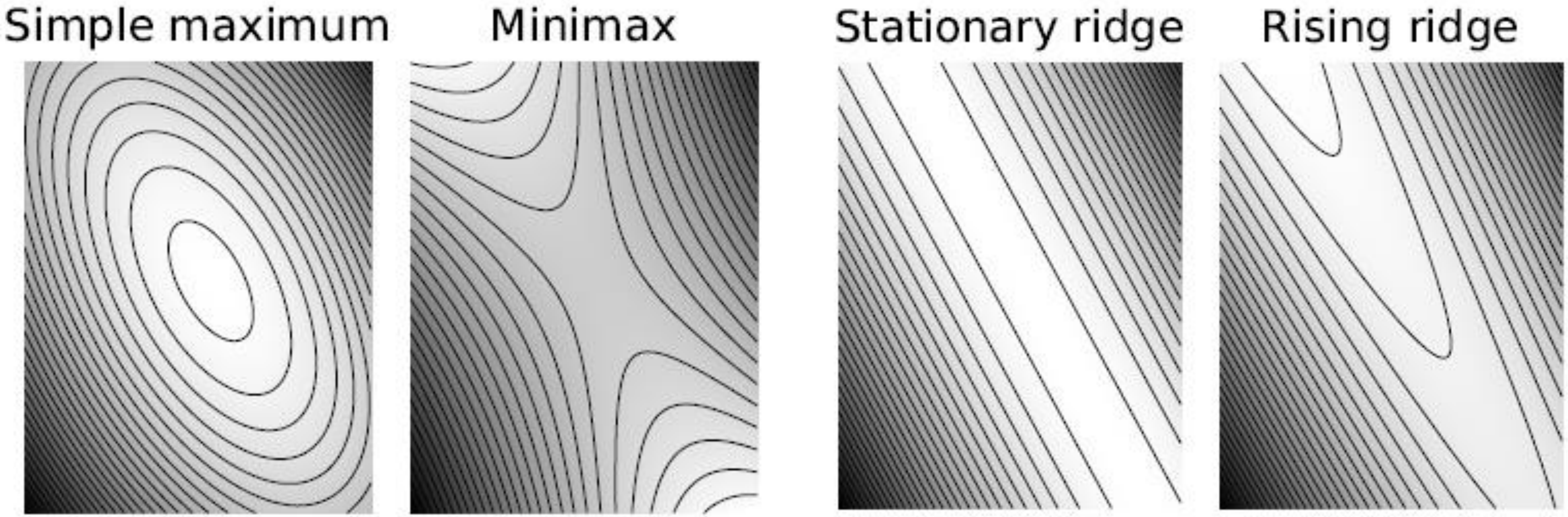
$$Y = f(x_1, \cdots, x_k) + e,$$

where e is a random variable with zero mean and a finite variance, σ^2. The set of points $\{f(x_1, \cdots, x_k), x_i \in D_i, i = 1, \cdots, k\}$, where $(D_1, \cdots, D_k)$ is the experimental domain of the x-variables, is called a **response surface**. Two types of response surfaces were discussed before, the **linear**

$$f(x_1, \cdots, x_k) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i \tag{5.9.1}$$

and the **quadratic**

$$f(x_1, \cdots, x_k) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i \neq j} \sum \beta_{ij} x_i x_j. \tag{5.9.2}$$


Simple maximum
Minimax
Stationary ridge
Rising ridge

5.9.2.2 Central Composite Designs

A Central Composite Design is one in which we start with $n_c = 2^k$ points of a factorial design, in which each factor is at levels -1 and $+1$. To these points we add $n_a = 2k$ axial points which are at a fixed distance α from the origin. These are the points

$$(\pm\alpha, 0, \cdots, 0), (0, \pm\alpha, 0, \cdots, 0), \cdots, (0, 0, \cdots, 0, \pm\alpha).$$

Finally, put n_0 points at the origin. These n_0 observations yield an estimate of the variance σ^2. Thus, the total number of points is $N = 2^k + 2k + n_0$. In such a design,

$$\begin{aligned} b &= 2^k + 2\alpha^2, \\ c &= 2^k, \\ c + d &= 2^k + 2\alpha^4, \quad \text{or} \\ d &= 2\alpha^4. \end{aligned} \tag{5.9.15}$$

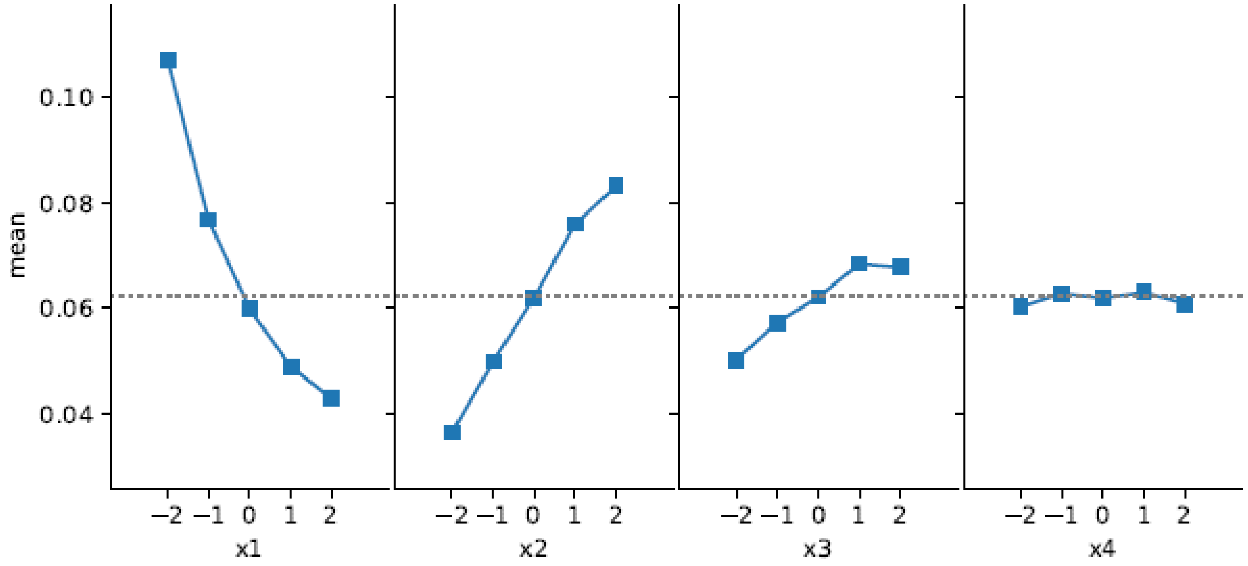
The rotatability condition is $d = 2c$. Thus, the design is rotatable if

$$\alpha^4 = 2^k \qquad \text{or} \qquad \alpha = 2^{k/4}. \tag{5.9.16}$$

For this reason, in central composite designs, with $k = 2$ factors we use $\alpha = \sqrt{2} = 1.414$. For $k = 3$ factors we use $\alpha = 2^{3/4} = 1.6818$. For rotatability and

Table 5.30: Factors and Level in Piston Simulator Experiment

Factor	Code	Levels				
Piston Surface Area s	x_1	.0075	.01	.0125	.015	.0175
Initial Gas Volume v_0	x_2	.0050	.00625	.0075	.00875	.0100
Spring Coefficient k	x_3	1000	2000	3000	4000	5000
Filling Gas Temperature t_0	x_4	340	345	350	355	360
	code	−2	−1	0	1	2



Optimal designs are constructed using several approaches. Their optimality is determined by different criteria that lead to different solutions. The **D-optimality** criterion minimizes the determinant of the covariance matrix of the model coefficient estimates, the information matrix $X'X$ of the design. D-optimality aims at deriving precise estimates of effects, i.e. main effects, quadratic effects and interactions. This assumes a known pre-specified model and is fully determined by the experimental design before the experiment is conducted. D-optimal designs are used in experiments conducted to test for significance of effects in order to best interpret the fitted model. Their main application is in designs aimed at distinguishing active from inert factors. A design is **A-optimal** if it minimizes the sum of the variances of the regression coefficients, the trace of the inverse of the information matrix. This is another approach to obtain precise estimates of the effects. **I-optimal** designs minimize the average variance of prediction over the design space. If the primary experimental goal is to predict a response or determine regions in the design space where the response falls within an acceptable range, the I-optimality criterion is more appropriate than the D-optimality criterion. In these cases, precise prediction of the response takes precedence over precise estimation of the parameters. A related approach is **G-optimal** designs, which minimize the maximum prediction variance over the design region. The maximum entry in the diagonal of the hat matrix $X^{-1}(X'X)^{-1}X$. These designs are calculated using Monte Carlo experiments of the design space. The minimum G aberration criterion is useful when selecting good, regular two-level fractional factorial designs and is used to discriminate among regular fractional factorial designs with the same resolution. Minimum aberration compares the frequency of aliases of regular designs at different levels. Regular designs with the smallest frequency of worst aliases are considered the best. The minimum G-aberration criterion can also handle irregular design spaces (Tang and Deng, 1999; Goos, 2011).

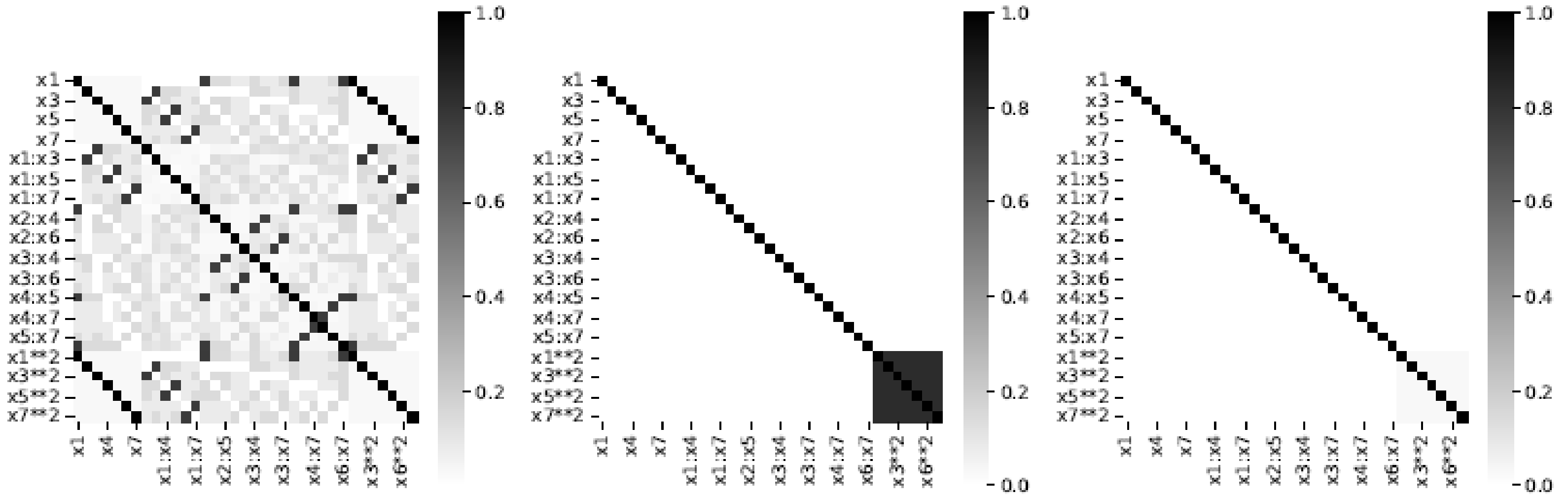


Fig. 5.14: Correlation plots of experimental arrays with 35, 80 and 169 experimental runs

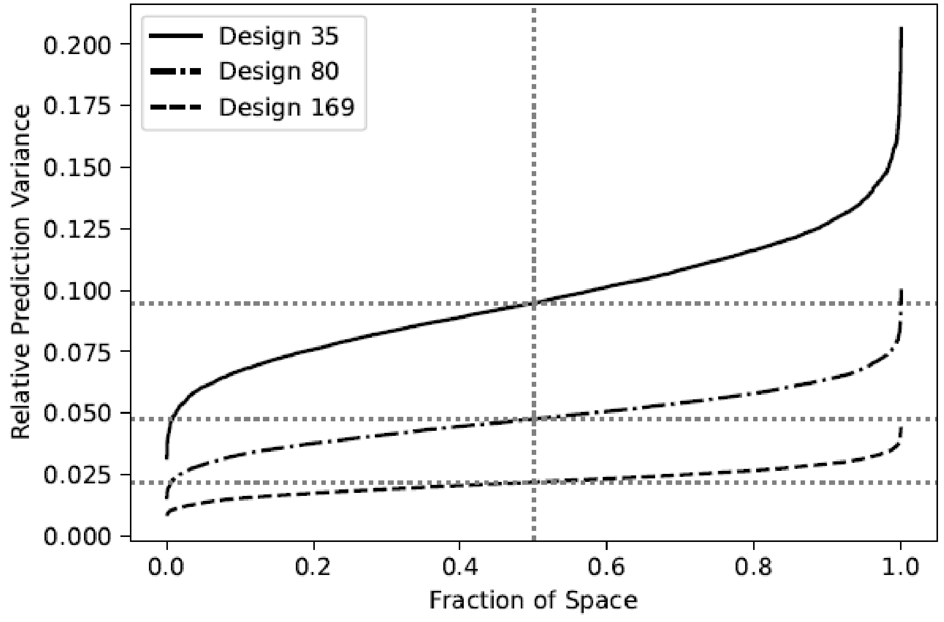


Fig. 5.15: Fraction of design space plot for the three designs

Chapter 6
Quality by Design

Preview Quality is largely determined by decisions made in the early planning phases of products and processes. A particularly powerful technique for making optimal design decisions is the statistically designed experiment introduced in Chapter 5. This chapter covers the basics of experimental designs in the context of engineering and economic optimization problems. Taguchi's loss function, signal to noise ratios, factorial models and orthogonal arrays are discussed using case studies and simple examples. A special section is dedicated to the application of Quality by Design (QbD) in the pharamceutical industry. QbD is supported internationally by the International Conference on Harmonization of Technical Requirements for Registration of Pharmaceuticals for Human Use (ICH) and by the Food and Drug Administration (FDA).

Chapter 7
Computer Experiments

Preview Computer experiments are integrated in modern product and service development activities, Technology is providing advanced digital platforms for studying various properties of suggested designs, without the need to physically concretize them. This chapter is about computer experiments and the special techniques required for designing such experiments and analyzing their outcomes. A specific example of such experiments is the piston simulator used throughout the book to demonstrate statistical concepts and tools. In this simulator random noise is induced on the control variables themselves, a non standard approach in modeling physical phenomena. The experiments covered include space filling designs and latin hypercubes. The analysis of the experimental outputs is based on Kriging or design and analysis of computer experiments (DACE) models. The chapter discusses the concept of a stochastic emulator where a model derived from the simulation outputs is used to optimize the design in a robust way. A special section is discussing several approaches to integrate the analysis of computer and physical experiments.

Chapter 8
Cybermanufacturing and Digital Twins

Preview Cybermanufacturing is a name given to smart manufacturing systems which are part of the fourth industrial revolution. These systems combine sensor technologies with flexible manufacturing and advanced analytics. In this chapter we introduce the main elements of cybermanufacturing as background and context to modern industrial analytics. Digital twins are related to cybermanufacturing but carry a wider scope. The idea is to provide a digital platform that parallels physical assets. This has implications, for example in healthcare, where MRI imaging is interpreted on the spot with artificial intelligence models trained on million of images (`https://www.aidoc.com/`). Other application domains of digital twins include agriculture, smart cities, transportation, autonomous vehicles, added manufacturing and 3D printing etc.. The chapter covers the topics of computer complexity, computational pipelines and reproducibility of analytic findings. It presents an integration of models for enhanced information quality, the Bayesian flow analysis and the Open ML community where datasets and data flows are uploaded in the spirit of open data. An additional topic covered in the chapter covers customer survey models used in modern companies to map their improvement journey.

Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-021-01817-9

Check for updates

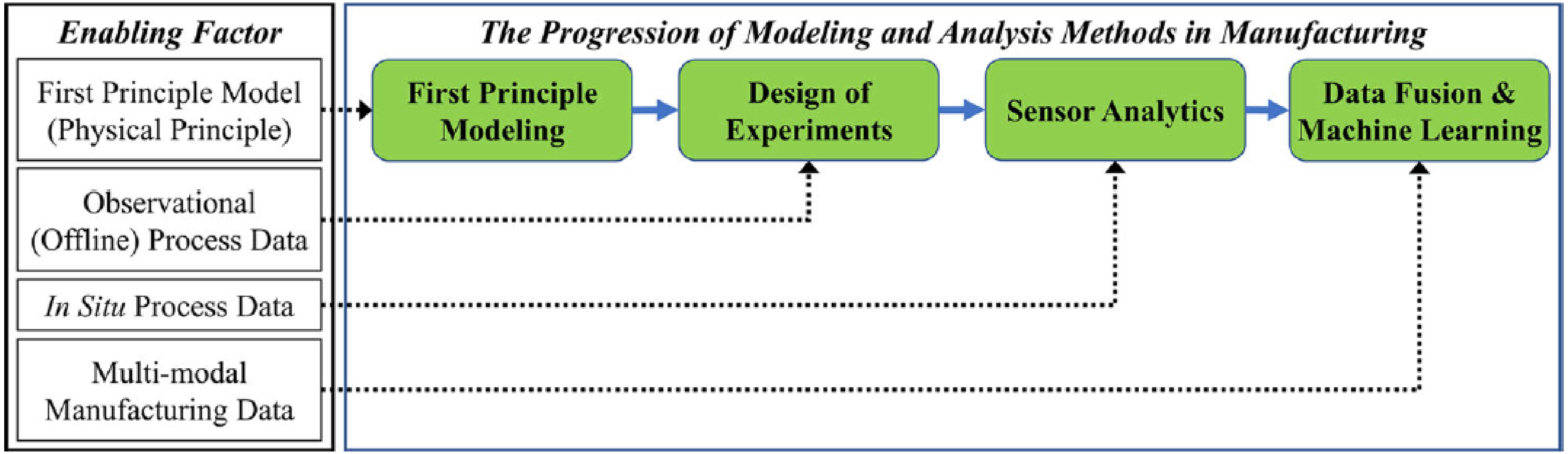
Challenges of modeling and analysis in cybermanufacturing: a review from a machine learning and computation perspective

SungKu Kang[1] · Ran Jin[1] · Xinwei Deng[1] · Ron S. Kenett[2]

Received: 31 December 2020 / Accepted: 19 July 2021

Abstract

In Industry 4.0, smart manufacturing is facing its next stage, cybermanufacturing, founded upon advanced communication,
computation, and control infrastructure. C
and provide a new perspective called comp
computation requests throughout manufac
However, the complexity of information t
under cybermanufacturing, ranging from i
less, existing reviews have focused on the
communication protocol), rather than the
In this paper, we review the fundamental
we introduce the existing efforts in comp
of method options for data analytics/mac
pipeline recommendation as a promising r
expect that computation pipeline recomm
the post-COVID-19 industry.



https://link.springer.com/content/pdf/10.1007/s10845-021-01817-9.pdf

Received: 17 October 2020 | Revised: 15 May 2021 | Accepted: 4 June 2021

DOI: 10.1002/qre.2948

SPECIAL ISSUE ARTICLE

WILEY

The digital twin in Industry 4.0: A wide-angle perspective

Ron S. Kenett[1] | Jacob Bortman[2]

[1] KPA Group, Raanana and Samuel Neaman Institute, Technion–Israel Institute of Technology, Haifa, Israel

[2] Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer Sheva, Israel

Correspondence
Ron S. Kenett, KPA Group, Raanana and Samuel Neaman Institute, Technion–Israel Institute of Technology, Haifa, Israel.
Email: ron@kpa-group.com

Abstract

The move towards advanced manufacturing and Industry 4.0 is fed by increased demand for speeding up innovation, increasing flexibility, improving maintenance, and becoming more customized while saving on the total cost of operations. This is accompanied by increased dependence on virtual product and process development, data-driven processes, and product knowledge. Other characteristics, affecting modern design, include big data intelligence in product, process, and maintenance. New technologies that empower data analytics include added manufacturing, flexible manufacturing, robotics, sensor technology, smart value/supply chains, and industrial information backbones. This paper is about surrogate models, also called *digital twins*, that provide an important complementary capacity to physical assets. Digital twins capture past, present, and predicted behavior of physical assets. Digital twin models are updated periodically to represent the current state of physical assets. This distinguishes digital twins from conventional simulations in that sensor data can continuously feed them. The type of curated information on the state of physical asset's history depends on how digital twins are used. For example, if a digital twin is used for fault classification, the history captured is operational data from equipment in healthy and faulty states. We provide here a review of digital twins, with an emphasis on Industry 4.0 applications.

https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2948

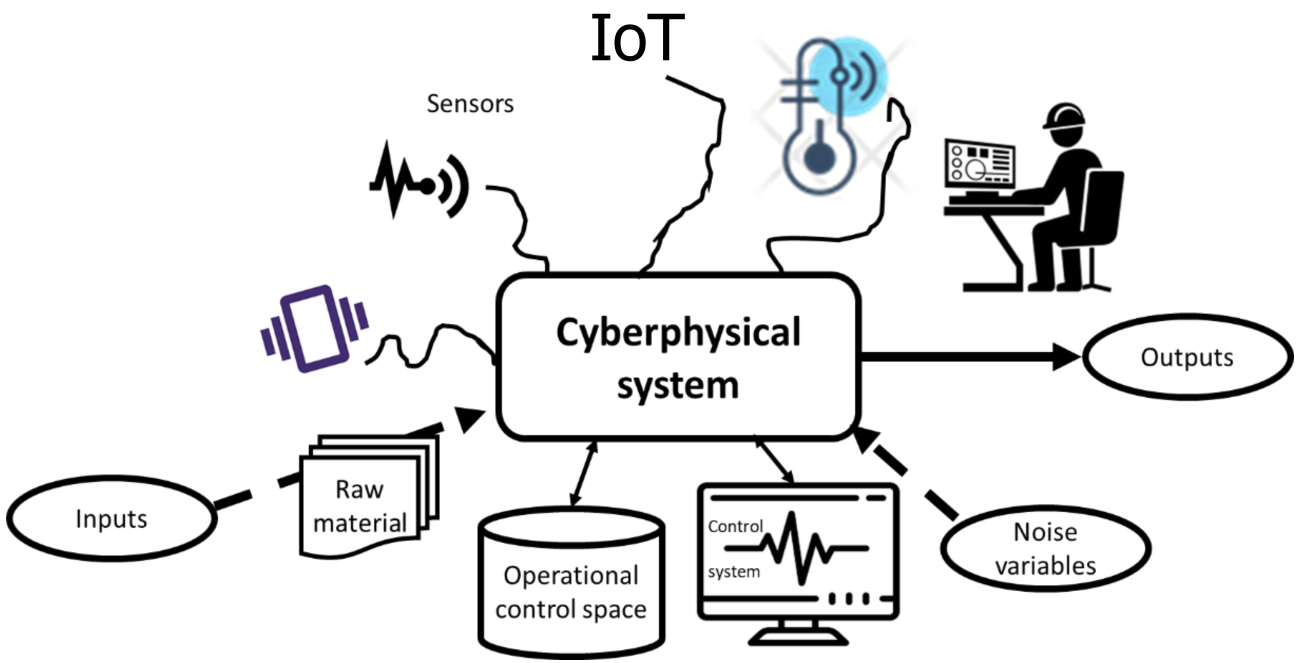
Received: 17 October 2020 | Revised: 15 May 2021 | Accepted: 4 June 2021
DOI: 10.1002/qre.2948

SPECIAL ISSUE ARTICLE

WILEY

The digital twin in Industry 4.0: A wide-angle perspective

Ron S. Kenett[1] | Jacob Bortman[2]



Physical assets

Received: 17 October 2020 | Revised: 15 May 2021 | Accepted: 4 June 2021

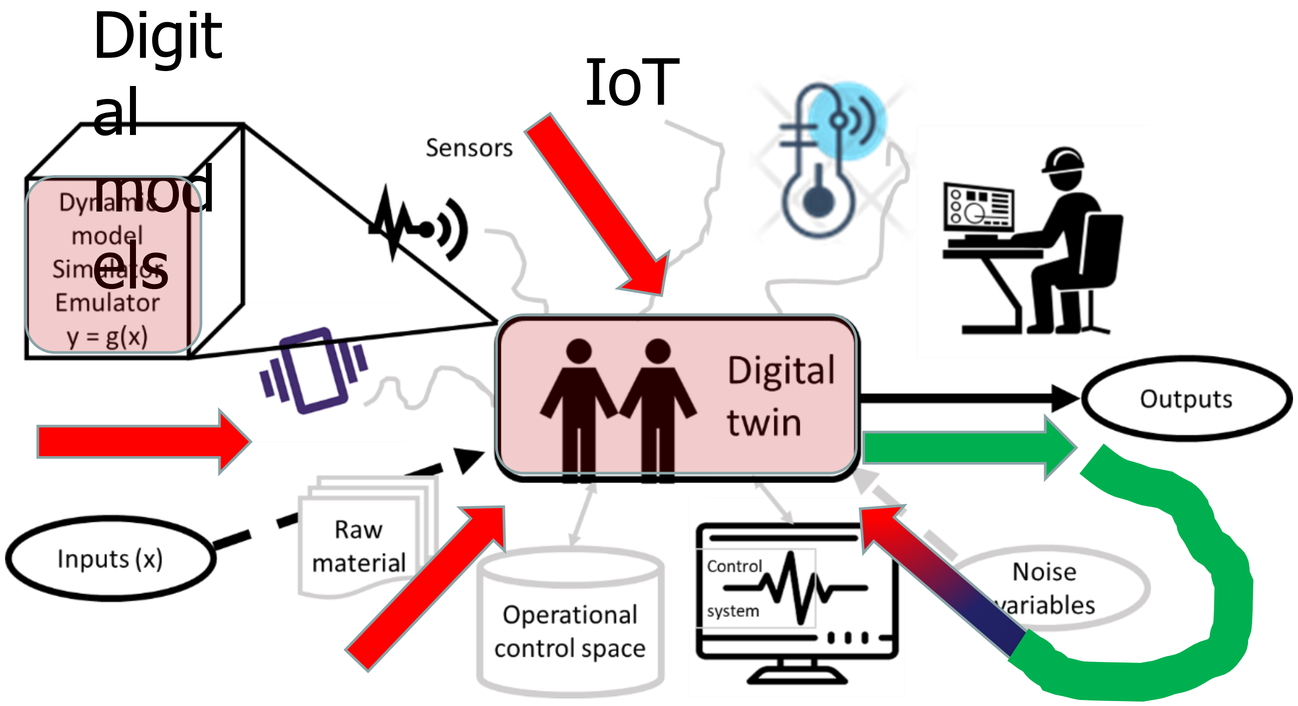
DOI: 10.1002/qre.2948

SPECIAL ISSUE ARTICLE

WILEY

The digital twin in Industry 4.0: A wide-angle perspective

Ron S. Kenett[1] | Jacob Bortman[2]



Digital assets

Chapter 9
Reliability Analysis

Preview The previous chapter dwelled on design decisions of product and process developers that are aimed at optimizing the quality and robustness of products and processes. This chapter is looking and performance over time and discusses basic notions of repairable and non repairable systems. Graphical and non parametric techniques are presented together with classical parametric techniques for estimating life distributions. Special sections cover reliability demonstration procedures, sequential reliability testing, burn-in procedures and accelerated life testing. Design and testing of reliability is a crucial activity of organizations adopting advanced quality and industrial standards discussed in Chapter 1.

Systems and products are considered to be of high quality, if they conform to their design specifications and appeal to the customer. However, products can fail, due to degradation over time or due to some instantaneous shock. A system or a component of a system is said to be **reliable** if it continues to function, according to specifications, for a long time. Reliability of a product is a dynamic notion, over time. We say that a product is highly reliable if the probability that it will function properly for a specified period, is close to 1. As will be defined later, the reliability function, $R(t)$, is the probability that a product will function in at least t units of time.

Chapter 10
Bayesian Reliability Estimation and Prediction

Preview It is often the case that information is available on the parameters of the life distributions from prior experiments or prior analysis of failure data. The Bayesian approach provides the methodology for formal incorporation of prior information with current data. This chapter presents reliability estimation and prediction from a Bayesian perspective. It introduces the reader to prior and posterior distributions used in Bayesian reliability inference, discusses loss functions and Bayesian estimators and nonparametric distribution free Bayes estimators of reliability. A section is dedicated to Bayesian credibility and prediction intervals. A final section covers empirical Bayes methods.

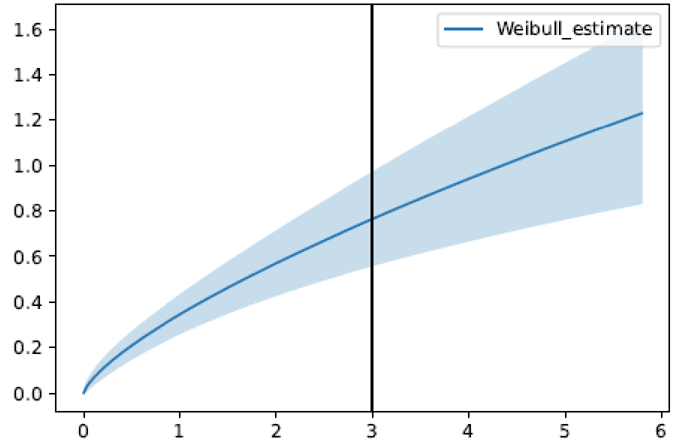


Fig. 10.1: Distribution profile of parametric Weibull model fitted to system failure data

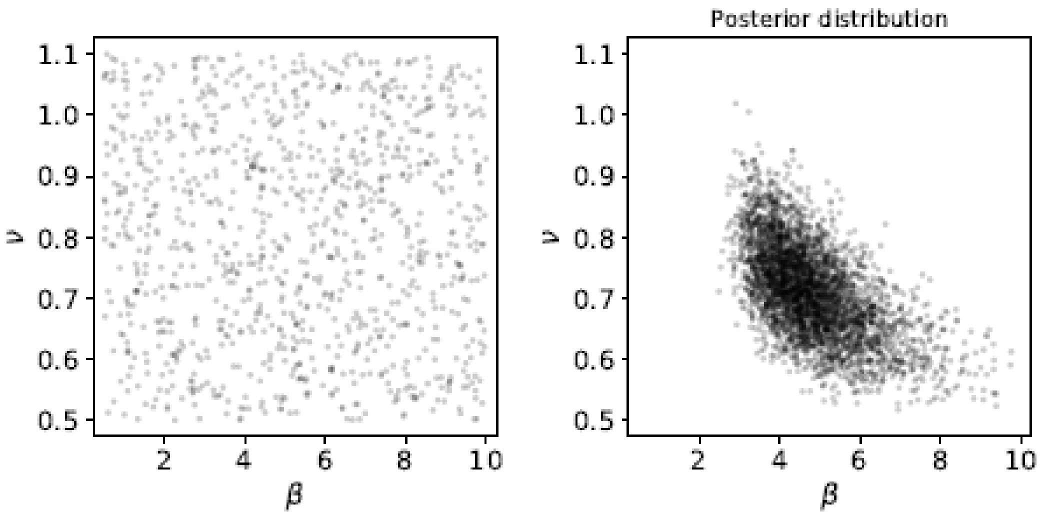


Fig. 10.3: Parameter values sampled from prior and posterior distribution of a Weibull model fitted to system failure data using uninformative priors

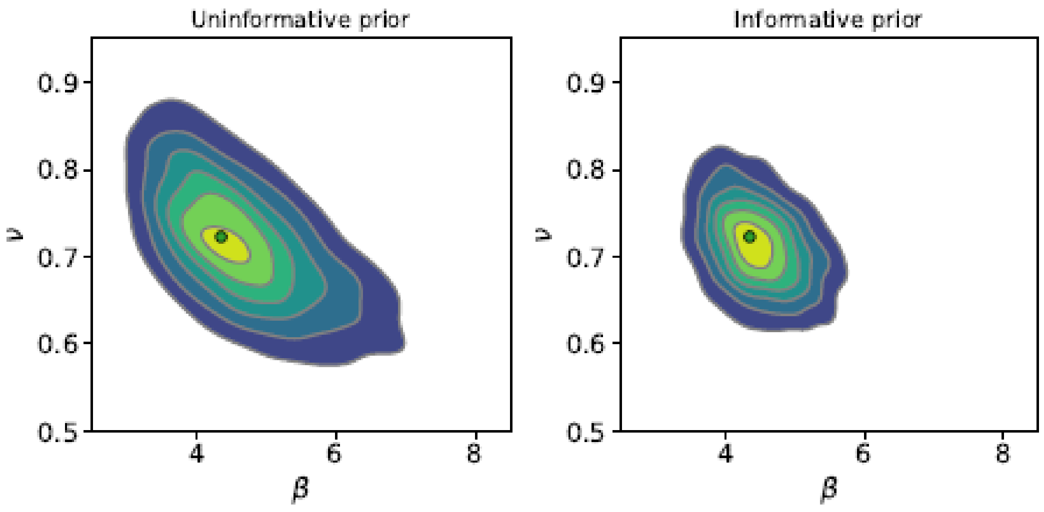


Fig. 10.4: Distribution of sampled parameter values using models defined with uninformative and informative priors

Industrial Statistics

A Computer Based Approach with Python

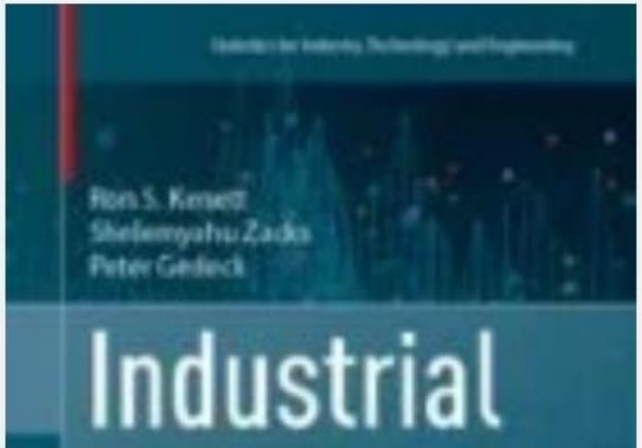
Home >**README.md**

Test passing

launch binder

https://gedeck.github.io/mistat-code-solutions/IndustrialStatistics/

Code repository



Industrial Statistics: A Computer Based Approach with Python
by Ron Kenett, Shelemyahu Zacks, Peter Gedeck

Publisher: Springer International Publishing; 1st edition (August 5, 2023)
ISBN-13: 978-3-031-28481-6 (hardcover)
ISBN-13: 978-3-031-28484-7 (softcover)
ISBN-13: 978-3-031-28482-3 (eBook).

Buy at Springer, Amazon,

Industrial Statistics: A Computer Based Approach with Python is a companion volume to the book Modern Statistics: A Computer Based Approach with Python.

This part of the repository contains:

- `notebooks` : Python code of individual chapters in Jupyter notebooks - download all as notebooks.zip
- `code` : Python code for solutions as plain Python files - download all as code.zip
- `solutions manual` : Solutions_IndustrialStatistics.pdf: solutions of exercises
- `solutions` : Python code for solutions in Jupyter notebooks - download all as solutions.zip
- `all` : zip file with all files combined - download all as all.zip
- `datafiles` : zip file with all data files - download all as data_files.zip - the `mistat` package gives you already access to all datafiles, you only need to download this file if you want to use it with different software

All the Python applications referred to in this book are contained in a package called `mistat` available for installation from the Python package index https://pypi.org/project/mistat/. The `mistat` packages is maintained in a GitHub repository at https://github.com/gedeck/mistat.

Table of contents (with sample excerpts from chapters)

Table of contents (with sample excerpts from chapters)

Installation instructions

Instructions on installing Python and required packages are here.

These Python packages are used in the code of *Industrial Statistics*:

- mistat (for access to data sets and additional functionality)
- numpy
- pandas
- scipy
- statsmodels
- matplotlib==3.6.0
- seaborn
- pingouin
- lifelines

Thank you for listening