OECD-TG305 R-Package bcmfR User Guide (v0.3-2)

OECD-TG305 R-Package bcmfR User Guide (v0.3-2).docx

tom.aldenberg@rivm.nl

2016-04-12

CONTENTS

1 INTRODUCTION

This User Guide describes the installation and use of the R-package *bcmfR* (Version 0.3-2).

The purpose of the program is to implement some of the models put forward in the OECD Guidelines for the Testing of Chemicals, Section 3, Test No. 305: **Bioaccumulation in Fish: Aqueous and Dietary Exposure**, as publicized on the website:

To run *bcmfR*, one needs the statistical software system 'R'. R can be found at:

https://www.r-project.org/

Before you consider to install or run any software that is mentioned in this document, notice that both R, and the present R-package *bcmfR*, come with ABSOLUTELY NO WARRANTY.

Package *bcmfR* is not available on the R archive site 'CRAN', yet, and can be regarded as an *in development* version. It is targeted towards problems of bioconcentration, and biomagnification, of chemicals in living organisms, and implements some of the models and estimation procedures discussed in the upcoming *Guidance Document to OECD TG 305, drafted by the Lead Countries DE, UK, and NL*, and. At this moment, *bcmfR* has not been tested by third parties.

If you are willing to try the package for research or guidance purposes, and would like to communicate any findings, bugs, or suggestions for improvement, then please send an email to the maintainer of the package in the DESCRIPTION file, see Chapter 2 INSTALLATION.

2 INSTALLATION

2.1 INSTALLING R AND RSTUDIO

To use the *bcmfR* package, you need to have the statistical software package 'R' installed on your computer. Not necessarily needed, but very convenient to run *bcmfR*, is to have installed the 'RStudio' IDE (Integrated Development Environment) program as well.

If you are in an IT-managed organization, let them make these programs available to you. The current versions are: R version **3.2.4** (3.3.0 very soon), and RStudio version **0.99.893**.

If you have Administrator rights on your computer, but are unaware of R or RStudio, then there is no better guidance on how to install R and RStudio than this 15-minute video from Sagar Nikam:

https://www.youtube.com/watch?v=MFfRQuQKGYg

You get some relaxing sounds, as well as a well-paced overview of how to obtain the software, and where (in Windows) it lands. Then, we get instructed on how to type code like: `2+2`, or: `print("Hello")`, and where to expect the answer. When you are new to R/RStudio, this is a must-see.

We skip on installing R, as well as RStudio, on Mac-computers, or Linux-based computers. You may check YouTube for that as well. The strength of both R and RStudio is that –in principle at least– these programs run on all current platforms.

Although the R and RStudio versions demonstrated in the video are somewhat outdated, most of it is valid for the recent versions. Website names have been updated:

https://www.r-project.org/

and

https://www.rstudio.com/

From Windows 7 onwards, you may want to select the 64-bit version of R, in addition to, or instead of, the 32-bit version. RStudio will automatically run the 64-bit version, when installed.

In the Appendix, a typical configuration adaptation for R is shown.

2.2 R-PACKAGES NEEDED BY *bcmfR*

The strength of R derives partly from the availability of thousands of R-compatible packages: pre-programmed R code that can be used from another R-program, or in an interactive R-session.

Package *bcmfR* needs the following packages to be installed in R on your computer:

```
car, stringr, lucid, nlstools, ggplot2, and ggthemes
```

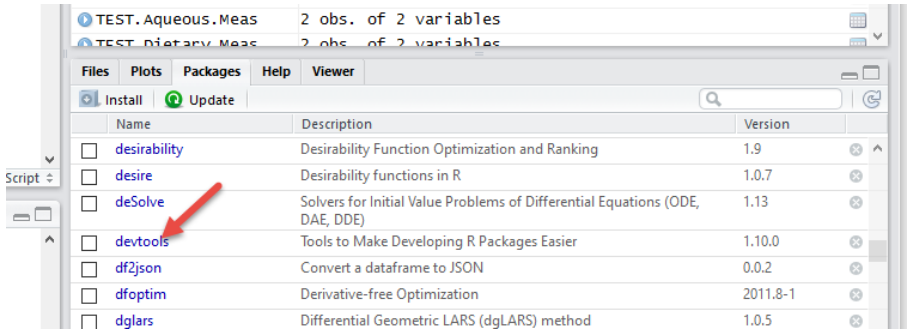
To learn how to install an R package, you may watch this video by thenewboston

https://www.youtube.com/watch?v=LUhU_GX6fys

As in the video, recent versions of the packages can be installed from the CRAN archive network. Installing packages needs only be done once, unless somebody de-installs them.

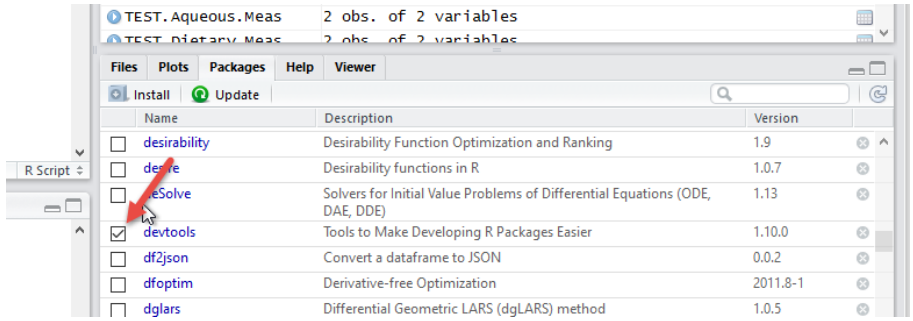
We will also need another R-package: `rio`. On CRAN, this package is currently in version 0.3.4. The development version by Thomas Leeper is on website https://github.com/leeper. This version is essential for reading input files into *bcmfR*.

To install the development version, you need the R-package `devtools`. Check in RStudio, whether `devtools` is installed under the **Packages** tab (Ctrl + 7) in the lower right main panel:



The arrow shows that it is installed on this system. If you don't have it installed, first install it (from CRAN).

Next, we need to attach `devtools` to activate it. Just check the box to the left of it:



Then, in the `Console` (Ctrl + 2), type

```
> install_github("leeper/rio")
```

This will install the latest development version (**0.3.17** or later) from https://github.com/leeper. This needs only be done once, unless it is de-installed in the mean time.

We plan to distribute *bcmfR* through CRAN and its mirror sites in the near future. For now, *bcmfR* is distributed as a so-called **local .zip archive package** file. This is an R-generated package that is not on CRAN, but can be distributed via email, on a data stick, or through a normal website.

The current DESCRIPTION file is the information that will be visible on CRAN, when the package will be released:

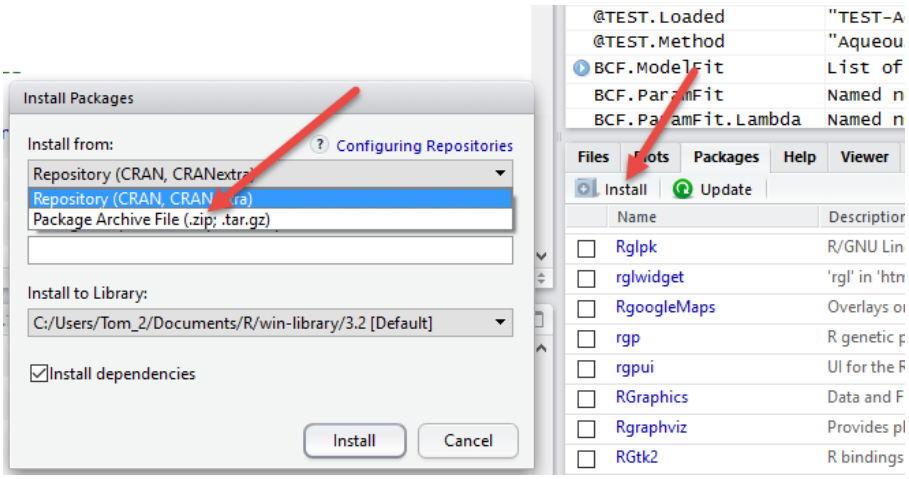
```
Package: bcmfR
Type: Package
Title: Tools for Modeling Bioaccumulation Potential in Fish
Version: 0.3-2
Author: Tom Aldenberg [aut, cre]
```

```
Maintainer: Tom Aldenberg <tom.aldenberg@rivm.nl>
Description: Implements models for both aqueous and dietary experimental exposure
    of chemicals to fish, as proposed in OECD Test Guideline 305. Estimation of
    uptake and depuration rate constants, growth correction, kinetic BCF and BMF,
    lipid correction, parameter and model confidence limits, and compilation of
    OECD TG 305-targeted summary tables with standard errors and uncertainty
    estimates.
License: GPL-3
LazyData: TRUE
Imports: car,
    stringr,
    lucid,
    nlstools,
    ggplot2,
    ggthemes,
    rio (>= 0.3.17)
```

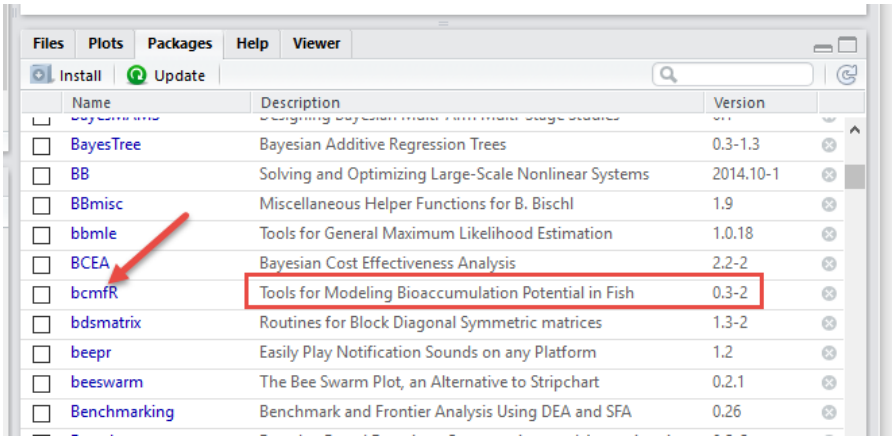
2.3 INSTALLING bcmfR VERSION 0.3-2

The current version of *bcmfR* is a local .zip archive package file: **`bcmfR_0.3-2.zip`**. This is a binary form of the *bcmfR* R-package, only for the Windows operating system. Other operating systems (Mac, Unix) have to wait until the package is on CRAN.

This file is to be saved somewhere on disk on your computer, e.g.: **My Documents**. To install the **`.zip`** binary package, click on **Install** of the Packages tab in RStudio (Ctrl + 7):



This will open the **Install Packages** dialog box. The default option (blue) is **Repository (CRAN, CRANextra)**. But for now: click **Package Archive File (.zip; .tar.gz)** instead. This will let you browse to the relevant directory, e.g. **My Documents**, where you have stored **`bcmfR_0.3-2.zip`**. Clicking the **Install** button will then install *bcmfR* as if it came from CRAN. This should be the result:


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
BayesTree
Bayesian Additive Regression Trees
0.3-1.3
BB
Solving and Optimizing Large-Scale Nonlinear Systems
2014.10-1
BBmisc
Miscellaneous Helper Functions for B. Bischl
1.9
bbmle
Tools for General Maximum Likelihood Estimation
1.0.18
BCEA
Bayesian Cost Effectiveness Analysis
2.2-2
bcmfR
Tools for Modeling Bioaccumulation Potential in Fish
0.3-2
bdsmatrix
Routines for Block Diagonal Symmetric matrices
1.3-2
beepr
Easily Play Notification Sounds on any Platform
1.2
beeswarm
The Bee Swarm Plot, an Alternative to Stripchart
0.2.1
Benchmarking
Benchmark and Frontier Analysis Using DEA and SFA
0.26

3 INPUT FILES FOR bcmfR

Input file format in bcmfR is a combination of the well-known **.csv** (comma-separated values) textual file format, preceded by a metadata header specified in **YAML**. This combination is indicated as a **.csvy** file. Two example files, one for aqueous and one for dietary exposure are included in bcmfR 0.3-2, and reproduced in the Appendix.

3.1 YAML METADATA PART

YAML (*YAML Ain't Markup Language*) is a human-friendly markup language to specify metadata, initialization files, markup specifications:

https://en.wikipedia.org/wiki/YAML

It is used for simple initializations, e.g. in RStudio itself, and increasingly found everywhere. It is compatible with the web interchange format JSON, and as powerful as XML, which is truly everywhere, but unreadable for human beings.

Here is a part of the Dietary exposure metadata section that we use in bcmfR:

```
---
# Dietary Exposure Fish Test
filename: Test-Dietary-RHCB10-22.1ugXpergFood.csvy

# type of experiment
type: Dietary Exposure Fish Test
guideline: OECD 305

# where data came from
data source:
    name: DM
    origin: OECD Ring
    comments:

# test chemical and biological species
chemical:
    name: RHCB10
    chemical info:
species:
    name: Oncorhynchus?
    biological info:

# food concentration and ingestion rate
cfood:
    value: 22.10
    unit: ugX/gFood
ingestion:
    value: 0.03
    unit: gFood/gFish/day
```

(not completely listed)

You'll notice that we see plain ASCII text with colon prompts and indentation. Lines with # are comments that are not read by the program. The major advantage over JSON and XML is clarity: no brackets, or curly braces, no single or double quotes, and no abundance of XML-style tags: <…>, etc.

However, there are a few things to know about this format, as simplicity and readability come with a price.

The initial triple dash marks a YAML file. The indentation is very critical. Within the `data source:` field, we see sub-fields: `name:` , `origin:`, and `comments:` indented four characters. The indent number (four) is **not critical, but *must* match exactly**. **No TAB characters are allowed**.

After every colon (':') a space is obligatory in a name-value pair.

The gain is that we need almost no single and double quotes; we can use spaces in names, slashes in values, and so on. Note that we put bioaccumulation experiment conditions here, as well as the units involved.

Both fields and subfields may be empty. You may also add additional (sub) fields, say `CAS no.`, or `life stage`, anything, or additional comments. The comments are not read on input, but additional fields are read. However, *bcmfR* will not use them. For example, in the current version of *bcmfR*, the field and subfield:

```
species:
    name: Oncorhynchus?
```

are read by the program, but not used. (This is *not* due to the question mark.)

We currently supply an Aqueous and a Dietary exposure example file with real data (Appendix). Just use these as a template for your data. To adapt the files, you can use any plain text editor, like Notepad, or Notepad++

http://notepad-plus-plus.org/

In Notepad++, you can set specific language formatting, e.g. for YAML, or R, and, via preferences, one can enforce TABs to be expanded into 2 or more spaces. It has no specific language settings for `.csvy` files.

3.2 CSV DATA PART

You probably know already how to export data from MS-Excel to a .csv file. Otherwise, watch this video from gonggongcomm:

https://www.youtube.com/watch?v=aBijHzKyteU

The idea behind `.csvy` files is to combine the YAML metadata header with the `.csv` data, as explained at:

http://csvy.org/

Here we show a part of the same file, further down, revealing the transition between the YAML metadata part and the CSV data part:

(middle of the file)

```
lipidfish:
    value: 4.65
    unit: percent

# time span of depuration
tdepur:
```

```
    value: 0.00
    unit: day
tend:
    value: 28.00
    unit: day

# fish growth
kgrowth:
    value: 0.0373
    unit: 1/day

# measured timeseries
time:
    name: Time
    unit: day
cfish:
    name: CFish
    unit: ugX/gFish
---
Time,CFish
1.0,5.930
1.0,5.050
1.0,4.790
3.0,3.790
3.0,4.530
3.0,2.250
```

(incomplete)

There is a second triple dash, followed by the experimental measurements in CSV form. Thus, when we have a `.csv` file generated from a possibly larger spreadsheet, we can simple copy paste the `.csv` text after the YAML header.

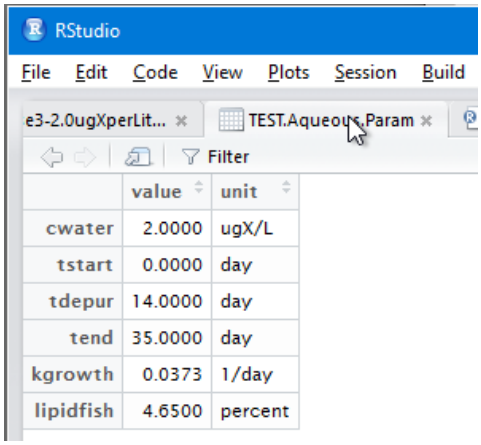
Note also that we announce the header names, as well as the units of the measured data. Now the rio package can read this file in one sweep and stores the experimental data as 'attributes' of the measurement dataframe in the R/ RStudio session. In this way, the data and its metadata always stay together.

The responsibility is with the user to combine the right experimental condition metadata with the right measurements, check values and units, double check them, again, and so on. After that, this information is never detached, anymore. So, in *bcmfR*, no tricky real-time peeking in complex spreadsheets, because they will always be organized differently.

The **Appendix** lists the two example input files for Aqueous and Dietary exposure test data and metadata.

3.3 Required input parameters Aqueous Exposure test

The next screen copy from *bcmfR* running in RStudio shows the essential input parameters derived from the YAML metadata header in the Aqueous Exposure `.csvy` input file, as listed in the Appendix:

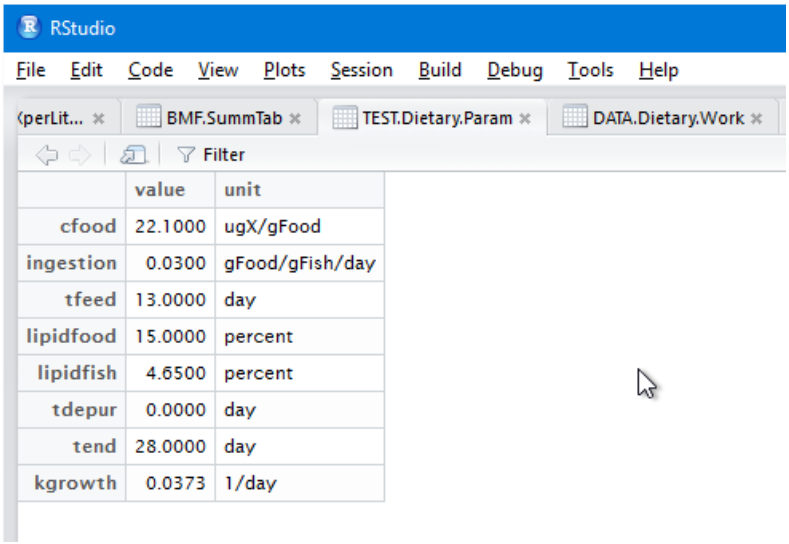


The test parameters (conditions) are in the same order as those in the input file, cf. Appendix. These are exactly the values (and units) to adapt for a different Aqueous Exposure test.

Although an R-savvy user could change the values within an R run, **we advise not to do so, as the actual run might be corrupted**. We recommend changing input values in a reproducible fashion, that is in the .csvy input file.

3.4 Required input parameters Dietary Exposure test

The following screen copy from *bcmfR* running in RStudio shows the experimental meta-information read from the Dietary Exposure .csvy input file:



Just as in the Aqueous case, these are the Dietary Test input parameters to adapt for another Dietary Test case. As stated in the previous paragraph, this should only be done in the plain text input file, as listed in the Appendix, **to avoid non-reproducible results,**.

4 RUNNING bcmfR REPRODUCIBLY

The bcmfR package has three calculation modes: Un-transformed, Ln-transformed, and Box-Cox transformed, as explained in the Guidance Document, Section 3.4. These transformations are applied to the major response data of interest: 'CFish', the concentration of chemical in the test organism (fish). Transformations influence the fit of the models, the uncertainty of the estimates and of further predictions. For the rationale behind transformations in relation to the supposed error distribution see the Guidance Document (Section 3.3).

4.1 AQUEOUS EXPOSURE TEST RUN NOTEBOOK

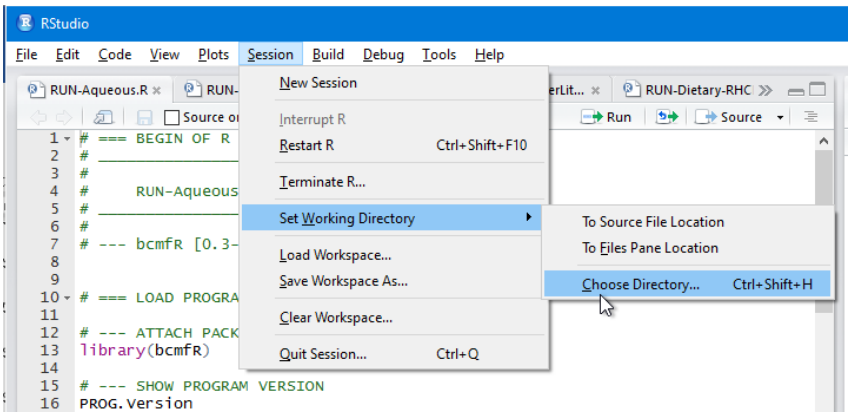
The Aqueous Exposure Test R-script is reproduced in the Appendix. You have to copy this text, as plain ASCII, to a new script file in RStudio. You do this by creating an R Script file under menu File/ New File/ R Script and copy the text in the new window. This file can be saved for later use in the working directory.

You can now evaluate the script file line by line by clicking the 'run' icon, or by selecting all or sections of the script with the mouse and then click the 'run icon'. As explained below, the whole script should be evaluated and all relevant output is generated using the **Compile Notebook button** (see below).

The test data input file is found around line 22:

```
# --- READ AQUEOUS TEST DATA
readTest_Aqueous("TEST-Aqueous-CRCase3-2.0ugXperLiter.csvy")
```

Instead of this filename, you insert your specific test file name. These are normally stored in your RStudio working directory: menu item **Session**, then **Set Working Directory**:



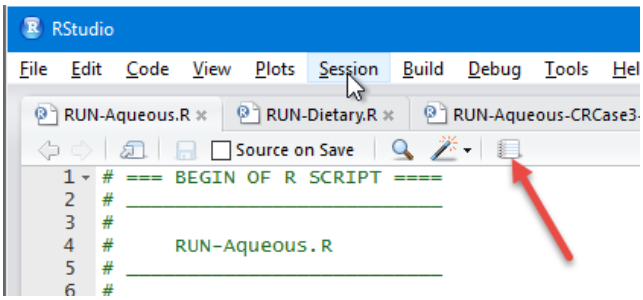
The two built-in example test data files are also available in this release of bcmfR, as built in 'external' data. You invoke in the Console (Ctrl + 2):

```
> aqua3 <- system.file("extdata", "TEST-Aqueous-CRCase3-2.0ugXperLiter.csvy", package = "bcmfR")
```

Then insert

```
# --- READ AQUEOUS TEST DATA
readTest_Aqueous(aqua3)
```

To obtain reproducible output, that is a computation that each time it is run on the same input file will generate the same output file, then, in RStudio, press the **Compile Notebook button**:



You have three output document options: as HTML, as PDF, or as MS-Word.

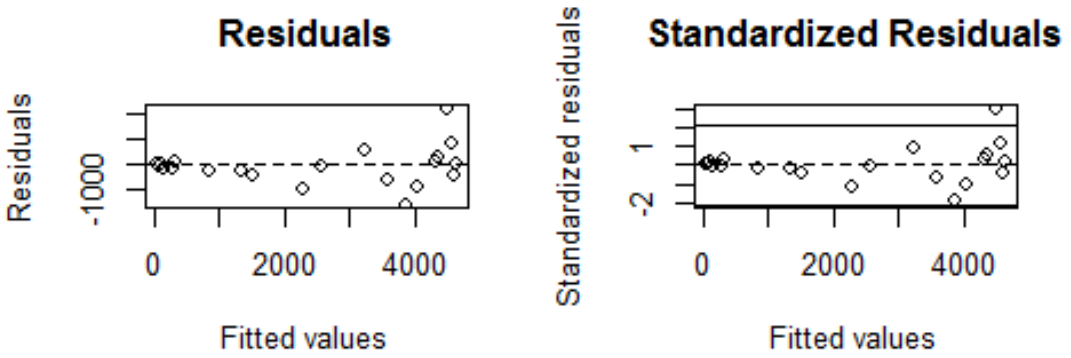
This is MS-Word Notebook document containing the input script and intermediate results:

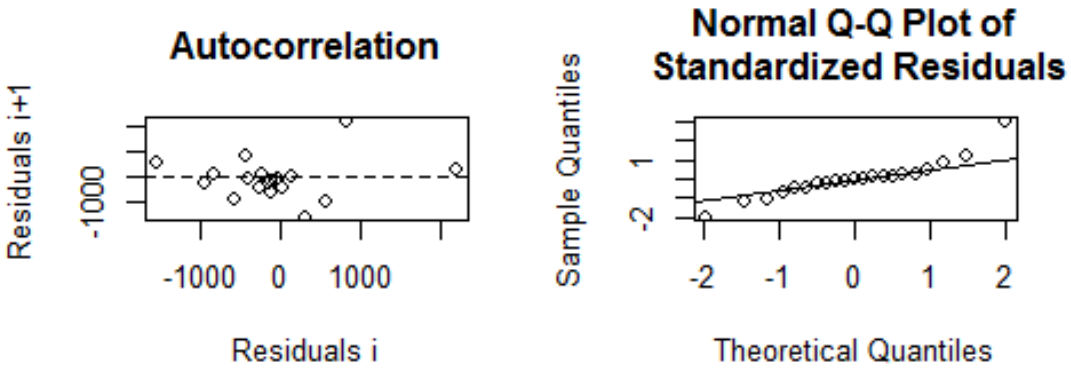
RUN-Aqueous.R

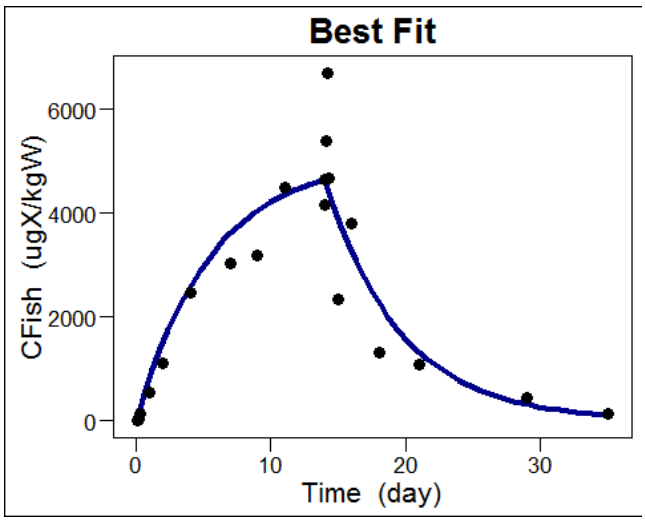
Mon Apr 11 12:59:20 2016

```
# === BEGIN OF R SCRIPT ====
# _____________________________
#
#     RUN-Aqueous.R
# _____________________________
#
# --- bcmfR [0.3-2] R Script


# === LOAD PROGRAM ====

# --- ATTACH PACKAGE
library(bcmfR)

# --- SHOW PROGRAM VERSION
PROG.Version

## [1] "0.3-2"

# === READ DATA ====

# --- READ AQUEOUS TEST DATA
readTest_Aqueous("TEST-Aqueous-CRCase3-2.0ugXperLiter.csvy")

##    time.data cfish.data
## 1       0.04      10.50
## 2       0.08       7.73
## 3       0.17      24.12
## 4       0.33     128.80
## 5       1.00     553.70
## 6       2.00    1105.47
## 7       4.00    2464.88
## 8       7.00    3025.53
## 9       9.00    3195.05
## 10     11.00    4485.04
## 11     14.00    4652.28
## 12     14.04    4167.07
## 13     14.08    5385.64
```

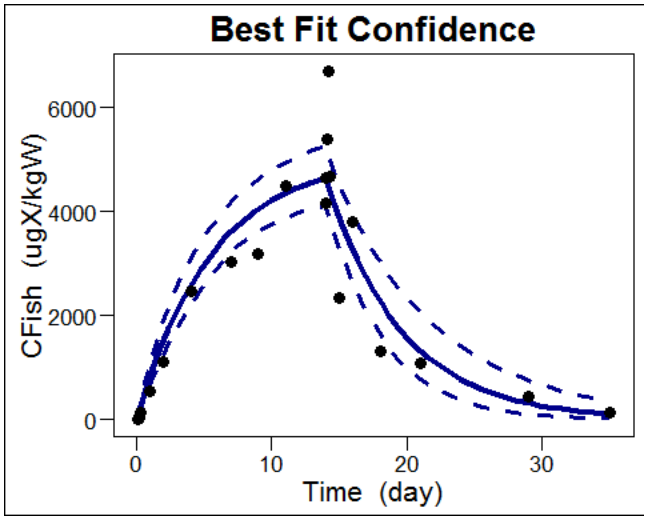
```
## 14     14.17    6692.33
## 15     14.33    4674.34
## 16     15.00    2329.99
## 17     16.00    3797.43
## 18     18.00    1328.29
## 19     21.00    1080.29
## 20     29.00     438.57
## 21     35.00     128.83

# --- SHOW TEST PARAMS
TEST.Aqueous.Param

##             value    unit
## cwater     2.0000   ugX/L
## tstart     0.0000     day
## tdepur    14.0000     day
## tend      35.0000     day
## kgrowth    0.0373   1/day
## lipidfish  4.6500 percent

# === UNTRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Aqueous()

## Nonlinear regression model
##   model: cfish.data ~ runModel_Aqueous(time.data, cwater, tdepur, fitk1,     fitk2)
##    data: datacfish
##    fitk1    fitk2
## 451.3105   0.1784
##  residual sum-of-squares: 10797483
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 2.88e-06

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous()

##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.89165, p-value = 0.02419
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
## Standard Normal = -1.1114, p-value = 0.2664
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Aqueous() -> rbf.a.un

# --- PLOT BEST FIT
plotBestFit_Aqueous()
```



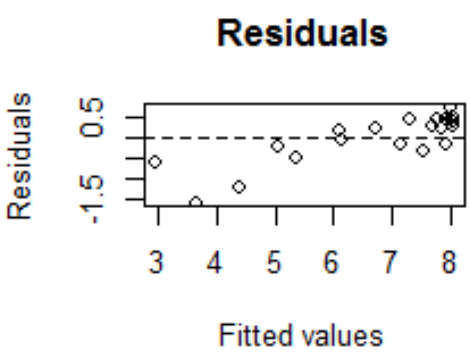


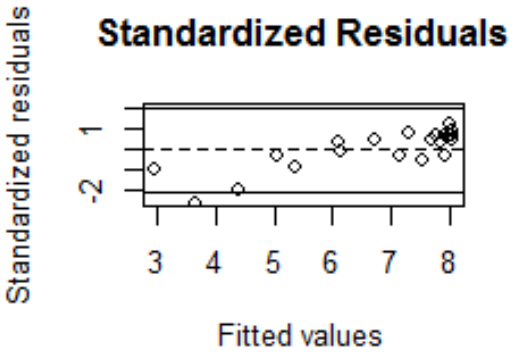


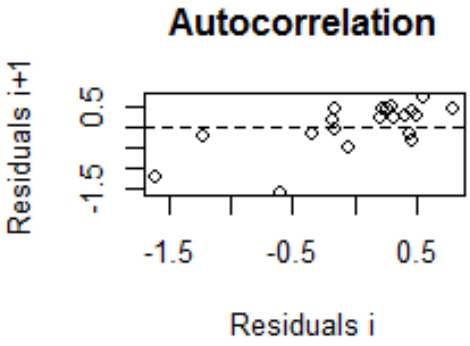

```
# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous() -> bbm.a.un

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous() -> rcf.a.un

# --- PLOT MODEL FIT AND CONF LIMITS
plotConfFit_Aqueous()
```

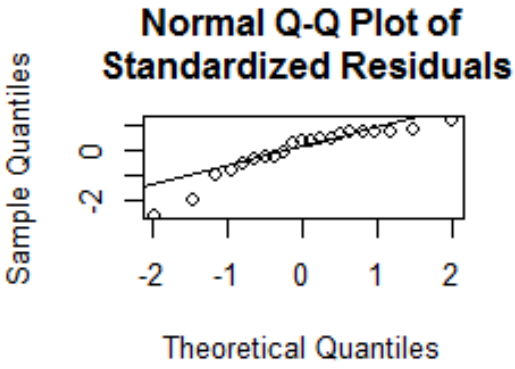



```
# --- BCF SUMMARY TABLE
summaryTable_Aqueous(digits = 5)

##         Estimate Std.Error      2.5%    97.5%
## k1       451.31    79.05    296.37    606.25
## k2         0.178    0.0379    0.1041    0.253
## k2g        0.141    0.0379    0.0668    0.215
## BCFK    2529.5    179.68   2177.3    2881.6
## BCFKg   3198      369.8    2473.2    3922.8
## thalfg     4.911    1.3198    2.3238    7.497
## BCFKgL  3438.7    397.64   2659.4    4218.1

# === LN-TRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Aqueous_Ln()

## Nonlinear regression model
##   model: ln.cfish.data ~ log(runModel_Aqueous(time.data, cwater, tdepur,     fitk1,
 fitk2) + lnstarter)
##    data: datacfish
##    fitk1    fitk2
## 240.3745   0.1277
##  residual sum-of-squares: 7.113
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 4.556e-07

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous_Ln()
```

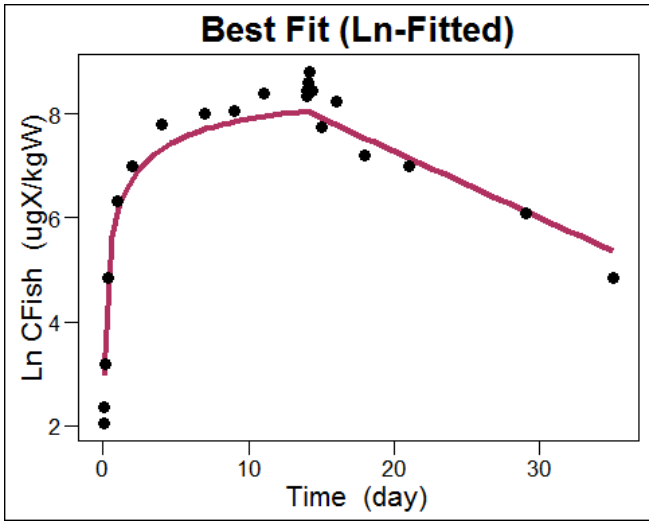




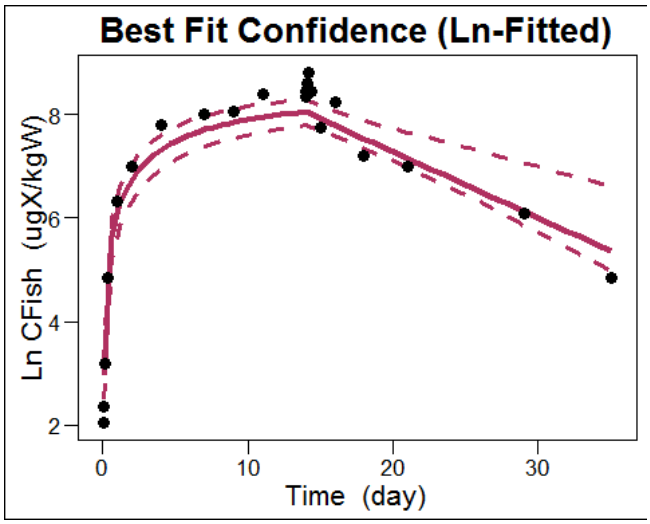




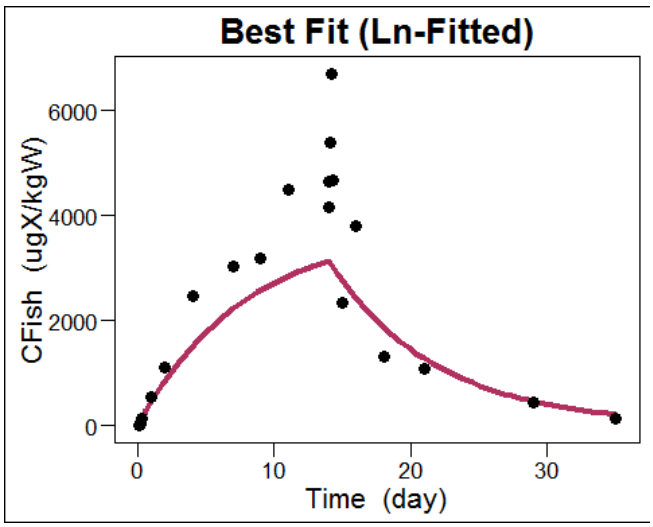

```
##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.8784, p-value = 0.01364
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
## Standard Normal = -2.8764, p-value = 0.004023
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Aqueous_Ln() -> rbf.a.ln

# --- PLOT BEST FIT
plotBestFit_Aqueous_Ln()
```



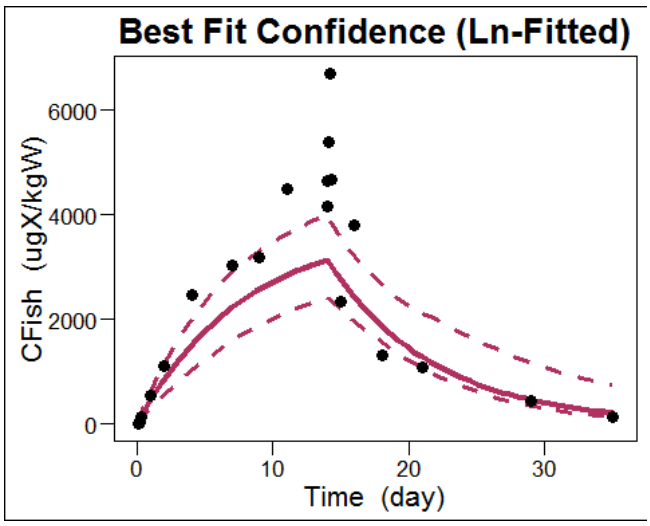

```
# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous_Ln() -> bbm.a.ln

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous_Ln() -> rcf.a.ln

# --- PLOT MODEL CONF LIMITS
plotConfFit_Aqueous_Ln()
```



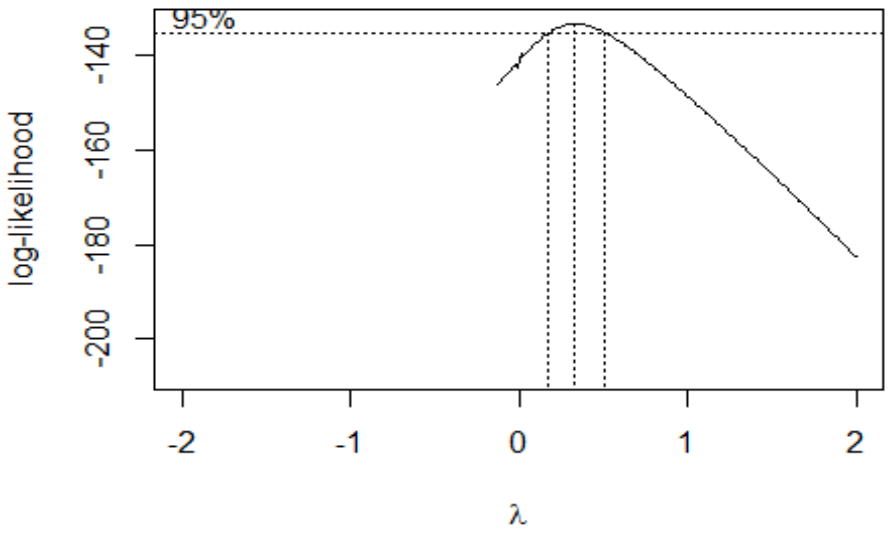

```
# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Aqueous_Ln()
```



```
# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Aqueous_Ln()
```

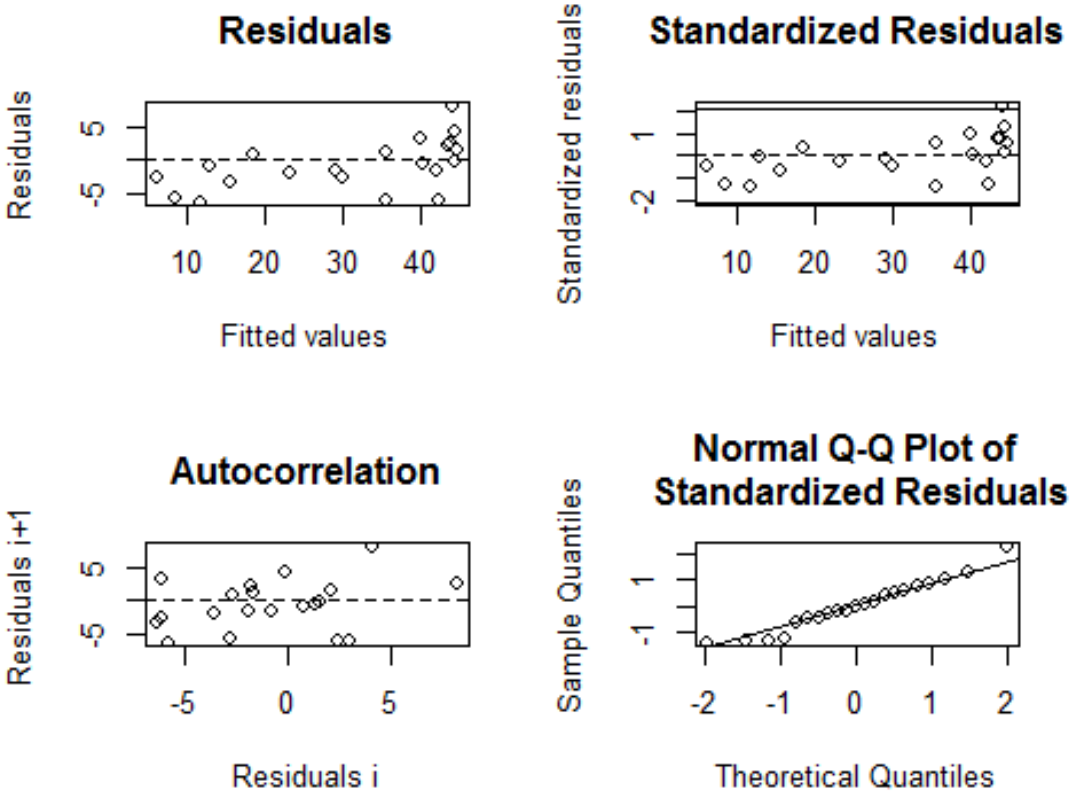



```
# --- BCF SUMMARY TABLE
summaryTable_Aqueous_Ln(digits = 5)

##          Estimate Std.Error      2.5%    97.5%
## k1       240.37     42.89    156.31    324.44
## k2         0.1277    0.0207    0.0872    0.168
## k2g        0.0904    0.0207    0.0499    0.131
## BCFK    1882.3     264.29   1364.3    2400.3
## BCFKg   2658.9     460.31   1756.7    3561.1
## thalfg     7.6655    1.752     4.2316   11.099
## BCFKgL  2859       494.96   1888.9    3829.1

# === BC-TRANSFORMED FIT ====

# --- BOX-COX PLOT AND LAMBDA ESTIMATE
modelTrans_BoxCox_Aqueous() -> lambda.est
```



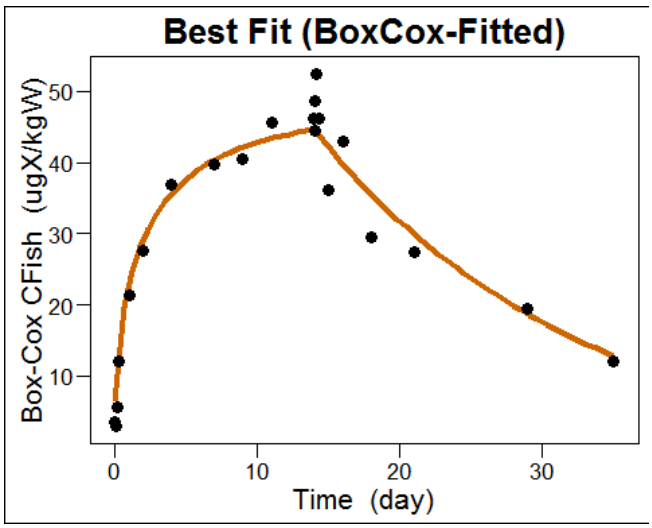

```
# --- SHOW LAMBDA FIT AND CONF LIMITS
lambda.est
```

```
##       fit    conflo    confup
## 0.3300000 0.1743490 0.5114643

# --- FIT BCF MODEL
fitModel_Aqueous_BoxCox(lambda = lambda.est["fit"])

## Nonlinear regression model
##   model: bc.cfish.data ~ car::bcPower(runModel_Aqueous(time.data, cwater,     tdepu
r, fitk1, fitk2) + lnstarter, lambda)
##    data: datacfish
##    fitk1    fitk2
## 375.5078   0.1585
##  residual sum-of-squares: 293.6
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 1.7e-07

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous_BoxCox()
```

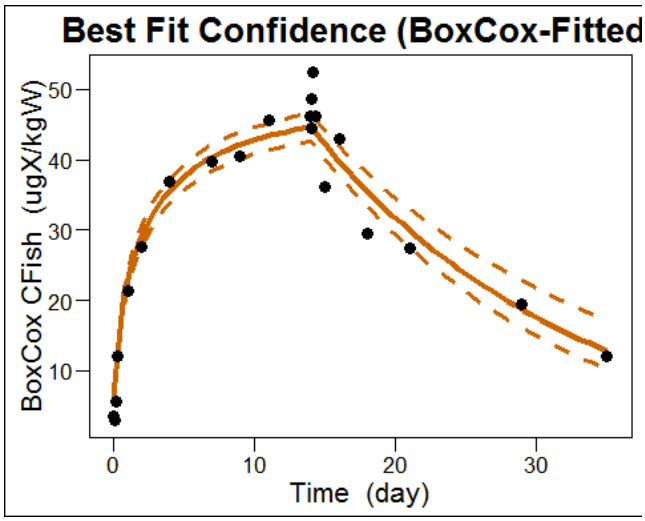



```
##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.96146, p-value = 0.546
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
```

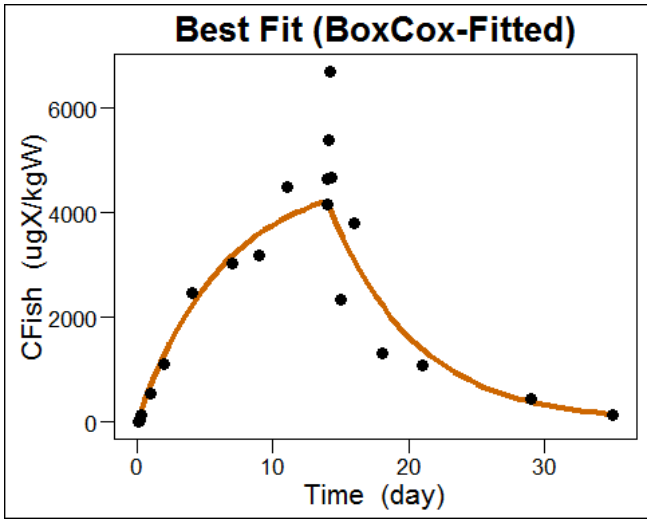
```
## Standard Normal = -1.1114, p-value = 0.2664
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Aqueous_BoxCox() -> rbf.a.bc

# --- PLOT BEST FIT
plotBestFit_Aqueous_BoxCox()
```



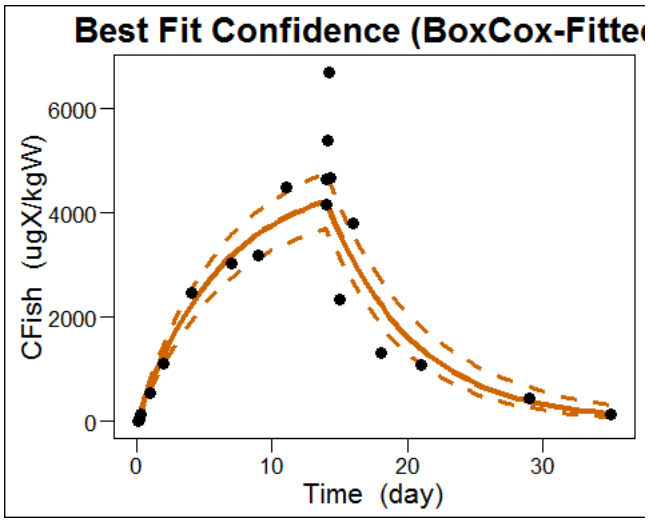

```
# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous_BoxCox() -> bbm.a.bc

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous_BoxCox() -> rcf.a.bc

# --- PLOT MODEL CONF LIMITS
plotConfFit_Aqueous_BoxCox()
```




```
# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Aqueous_BoxCox()
```




```
# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Aqueous_BoxCox()
```




```
# --- BCF SUMMARY TABLE
summaryTable_Aqueous_BoxCox(digits = 5)

##         Estimate Std.Error      2.5%    97.5%
## k1       375.51    45.75    285.84    465.18
## k2         0.158    0.0198    0.1197    0.197
## k2g        0.121    0.0198    0.0824    0.16
## BCFK    2369.2    177.91   2020.5    2717.9
## BCFKg   3098.4    295.31   2519.6    3677.2
## thalfg     5.718    0.934     3.8875    7.549
## BCFKgL 3331.6    317.54   2709.2    3954

# === END OF R SCRIPT ====
```

4.2 Dietary Exposure Test Run Notebook

The *bcmfR* package has three calculation modes as explained in the Guidance Document, Section 4.5.3: Untransformed, Ln-transformed, and Box-Cox transformed. These transformations are applied to the major response data of interest: 'CFish', the concentration of chemical in the test organism (fish) during the depuration phase. Transformations influence the fit of the models, the uncertainty of the estimates and of further predictions. For the

rationale behind transformations in relation to the supposed error distribution see the Guidance Document (Section 3.3 and 4.5.2).

The Dietary Exposure Test R-script is reproduced in the Appendix.

You have to copy this text, as plain ASCII, to a new script file in RStudio. You do this by creating an R script file under menu File/ New File/ R Script and copy the text in the new window. This file can be saved for later use in the working directory.

You can now evaluate the script file line by line by clicking the **Run** icon, or by selecting all or sections of the script with the mouse and then click the **Run** icon. As explained below, the whole script should be evaluated and all relevant output is generated using the **Notebook button**.

Similar to the previous Section on running the Aqueous Test case, we now show the reproducible Dietary Test case run, as output to MS-Word:

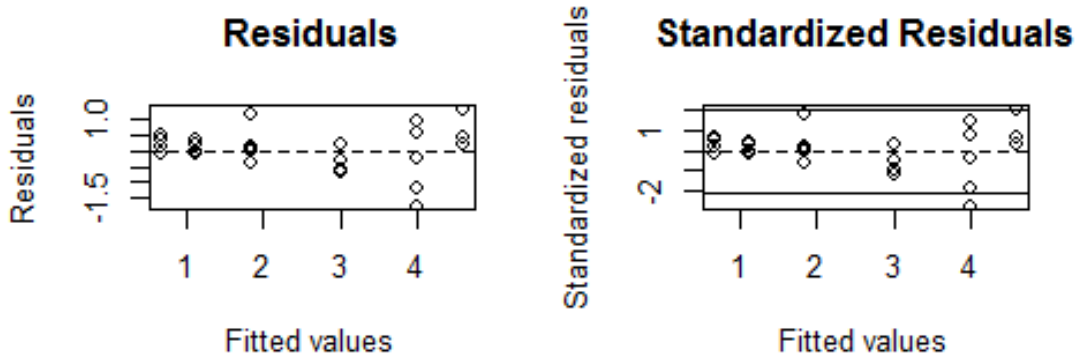
RUN-Dietary.R

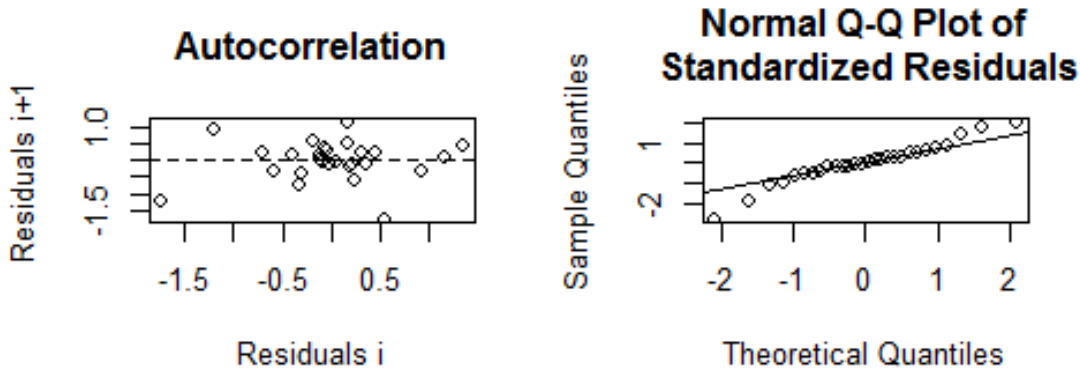
Mon Apr 11 13:23:13 2016

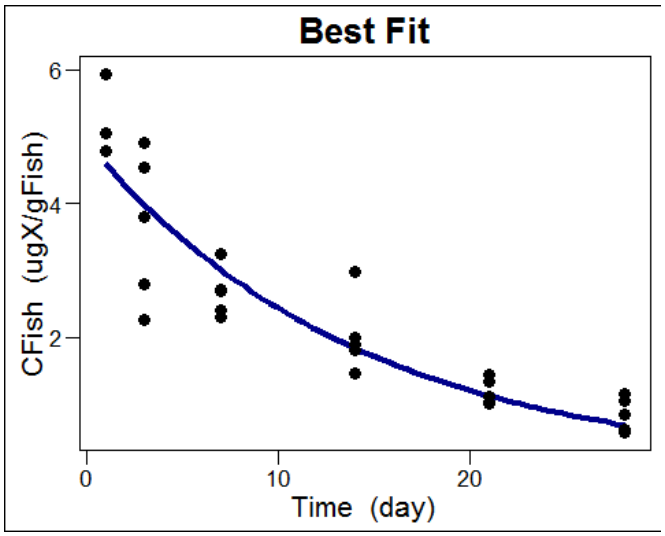
```
# === BEGIN OF R SCRIPT ====
# _____________________________
#
#     RUN-Dietary.R
# _____________________________
#
# --- bcmfR [0.3-2] R Script


# === LOAD PROGRAM ====

# --- ATTACH PACKAGE
library(bcmfR)

# --- SHOW PROGRAM VERSION
PROG.Version

## [1] "0.3-2"

# === READ DATA ====

# --- READ DIETARY TEST DATA
readTest_Dietary("TEST-Dietary-RHCB10-22.1ugXpergFood.csvy")

##    time.data cfish.data
## 1          1      5.930
## 2          1      5.050
## 3          1      4.790
## 4          3      3.790
## 5          3      4.530
## 6          3      2.250
## 7          3      2.800
## 8          3      4.900
## 9          7      2.690
## 10         7      2.300
## 11         7      3.250
```

```
## 12         7      2.410
## 13         7      2.700
## 14        14      1.450
## 15        14      2.000
## 16        14      2.980
## 17        14      1.890
## 18        14      1.800
## 19        21      1.440
## 20        21      1.330
## 21        21      1.010
## 22        21      1.110
## 23        21      1.020
## 24        28      0.611
## 25        28      1.040
## 26        28      0.573
## 27        28      0.852
## 28        28      1.150

# --- SHOW TEST PARAMS
TEST.Dietary.Param

##             value            unit
## cfood     22.1000       ugX/gFood
## ingestion  0.0300 gFood/gFish/day
## tfeed     13.0000             day
## lipidfood 15.0000         percent
## lipidfish  4.6500         percent
## tdepur     0.0000             day
## tend      28.0000             day
## kgrowth    0.0373           1/day

# === UNTRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Dietary()

## Nonlinear regression model
##   model: cfish.data ~ runModel_Dietary(time.data, fitc0d, fitk2)
##    data: datacfish
##  fitc0d   fitk2
## 4.92724 0.07047
##  residual sum-of-squares: 10.8
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 2.219e-06

# --- MODEL DIAGNOSTICS
modelDiag_Dietary()

##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.95334, p-value = 0.24
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
```

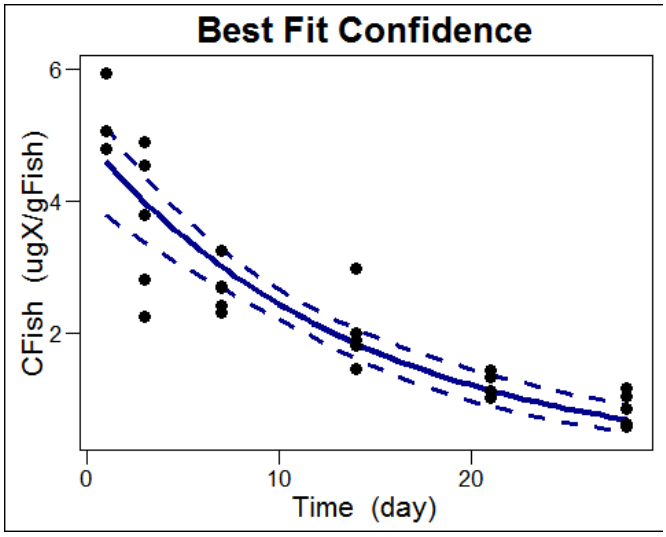
```
## Standard Normal = 0, p-value = 1
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Dietary() -> rbf.d.un

# --- PLOT BEST FIT
plotBestFit_Dietary()
```








```
# --- BAYESIAN BOOTSTRAP
bootModel_Dietary() -> bbm.d.un

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary() -> rcf.d.un

# --- PLOT MODEL FIT AND CONF LIMITS
plotConfFit_Dietary()
```



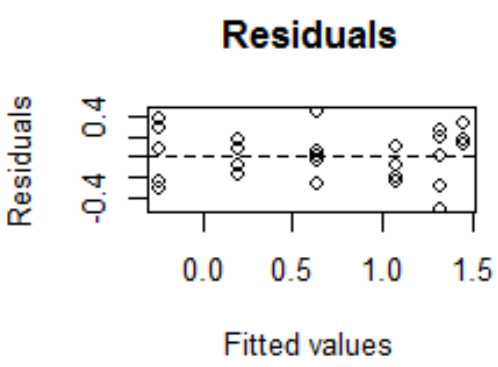

```
# --- BCF SUMMARY TABLE
summaryTable_Dietary(digits = 5)
```

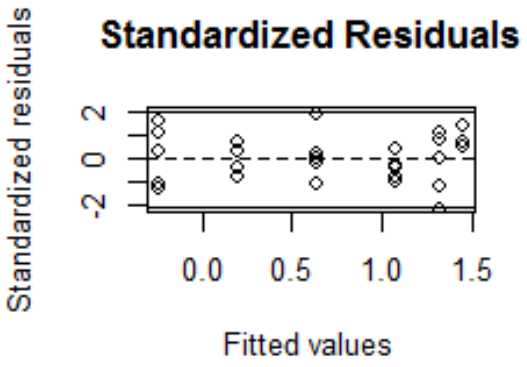
```
##         Estimate Std.Error      2.5%     97.5%
## C0d     4.9272    0.29993   4.3394    5.5151
## k2      0.070471  0.008513  0.053785  0.087157
## k2g     0.033171  0.008513  0.016485  0.049857
## kf      0.026189  0.0026    0.021092  0.031286
## alpha   0.87297   0.086682  0.70307   1.0429
## BMFK    0.37163   0.020211  0.33201   0.41124
## BMFKg   0.78951   0.13689   0.52121   1.0578
## thalfg 20.892     5.3619   10.382    31.401
## BMFKgL  2.5468    0.44158   1.6813    3.4123
```

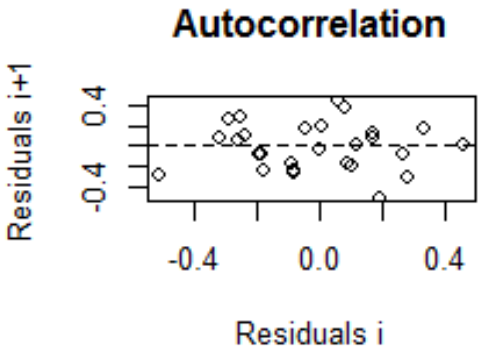
```
# === LN-TRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Dietary_Ln()
```

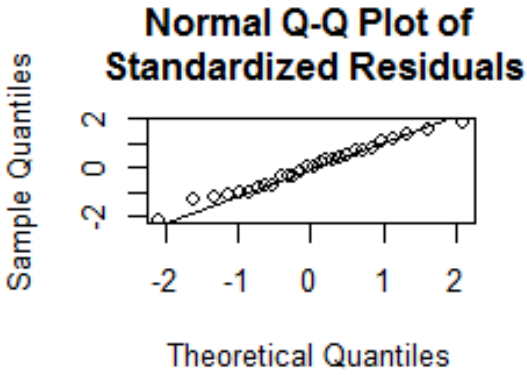
```
## Nonlinear regression model
##   model: ln.cfish.data ~ log(runModel_Dietary(time.data, fitc0d, fitk2) +     lnstar
ter)
##    data: datacfish
##  fitc0d   fitk2
## 4.52216 0.06251
##  residual sum-of-squares: 1.515
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 3.783e-09
```

```
# --- MODEL DIAGNOSTICS
modelDiag_Dietary_Ln()
```

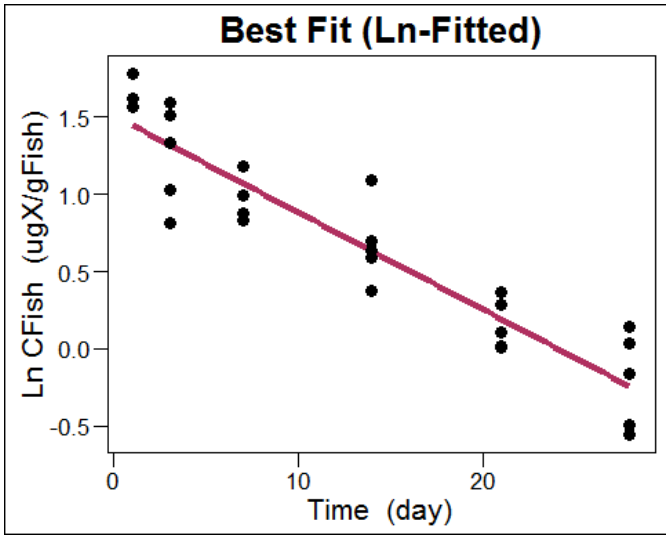




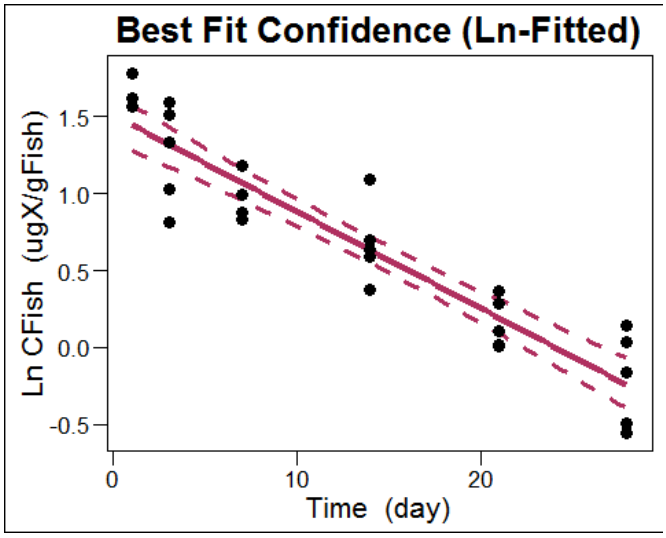




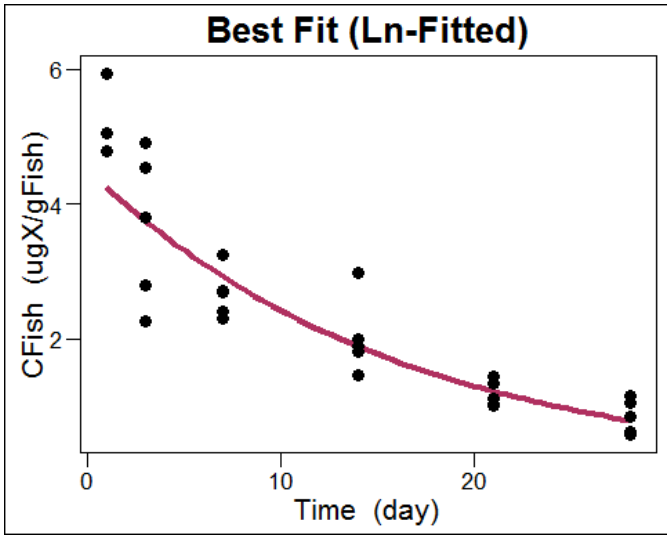

```
##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.98551, p-value = 0.9557
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
## Standard Normal = -0.74677, p-value = 0.4552
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Dietary_Ln() -> rbf.d.ln

# --- PLOT BEST FIT
plotBestFit_Dietary_Ln()
```



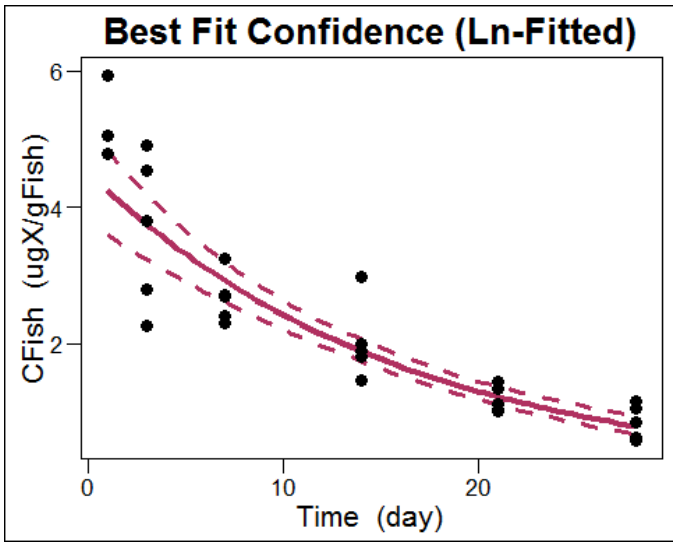

```
# --- BAYESIAN BOOTSTRAP
bootModel_Dietary_Ln() -> bbm.d.ln

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary_Ln() -> rcf.d.ln

# --- PLOT MODEL CONF LIMITS
plotConfFit_Dietary_Ln()
```



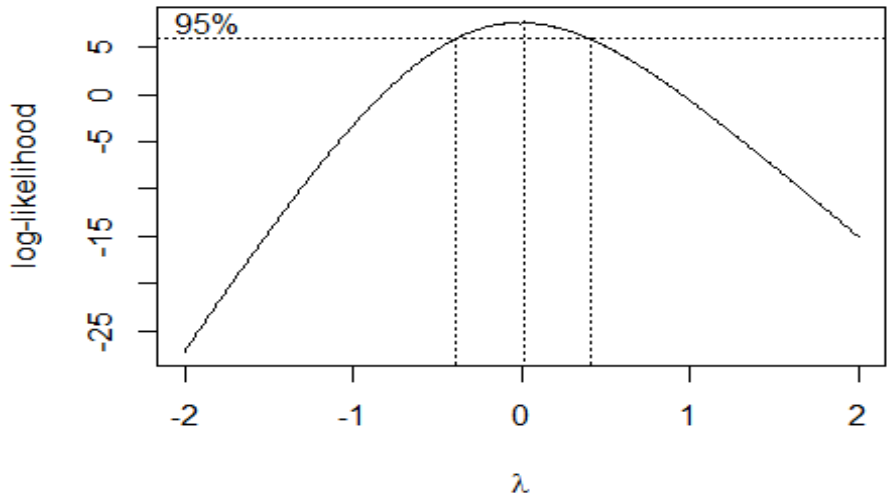

```
# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Dietary_Ln()
```



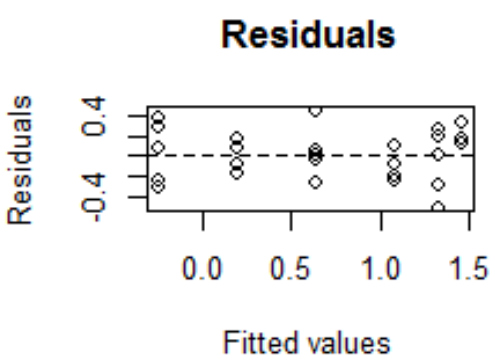

```
# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Dietary_Ln()
```

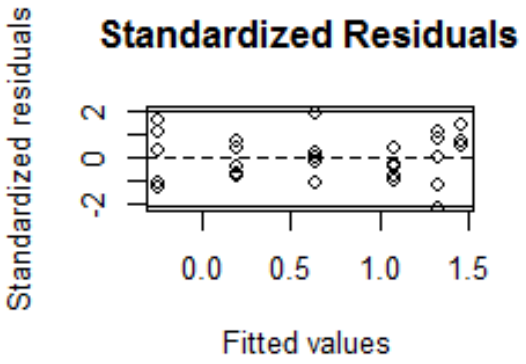


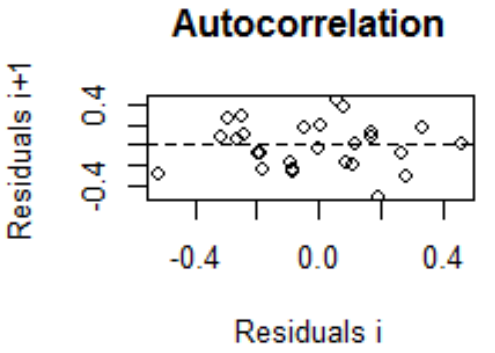

```
# --- BCF SUMMARY TABLE
summaryTable_Dietary_Ln(digits = 5)

##          Estimate Std.Error      2.5%    97.5%
## C0d      4.5222    0.3506    3.835     5.2093
## k2       0.062513  0.00477   0.053164  0.07186
## k2g      0.025213  0.00477   0.015864  0.03456
## kf       0.022993  0.002311  0.018464  0.02752
## alpha    0.76643   0.077017  0.61547   0.91738
## BMFK     0.36781   0.017471  0.33357   0.40205
## BMFKg    0.91195   0.099939  0.71607   1.1078
## thalfg 27.486      5.1998   17.295    37.678
## BMFKgL   2.9418    0.32238   2.3099    3.5737

# === BC-TRANSFORMED FIT ====

# --- BOX-COX PLOT AND LAMBDA ESTIMATE
modelTrans_BoxCox_Dietary() -> lambda.est
```

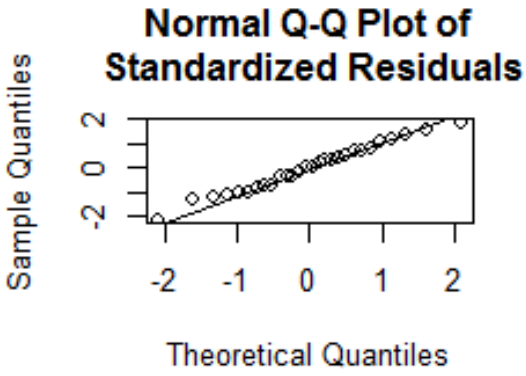



```
# --- SHOW LAMBDA FIT AND CONF LIMITS
lambda.est

##        fit     conflo     confup
##  0.0100000 -0.3951205  0.4054187

# --- FIT BCF MODEL
fitModel_Dietary_BoxCox(lambda = lambda.est["fit"])

## Nonlinear regression model
##   model: bc.cfish.data ~ car::bcPower(runModel_Dietary(time.data, fitc0d,     fitk2)
 + lnstarter, lambda)
##    data: datacfish
##  fitc0d   fitk2
## 4.52535 0.06255
##  residual sum-of-squares: 1.536
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 1.314e-06

# --- MODEL DIAGNOSTICS
modelDiag_Dietary_BoxCox()
```

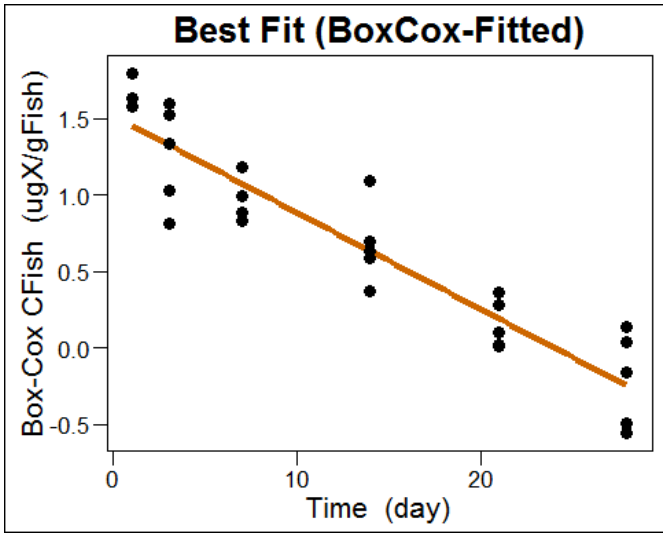




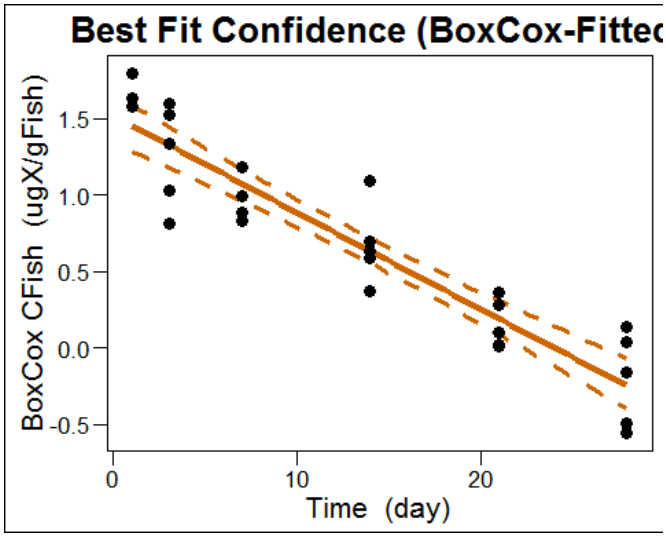




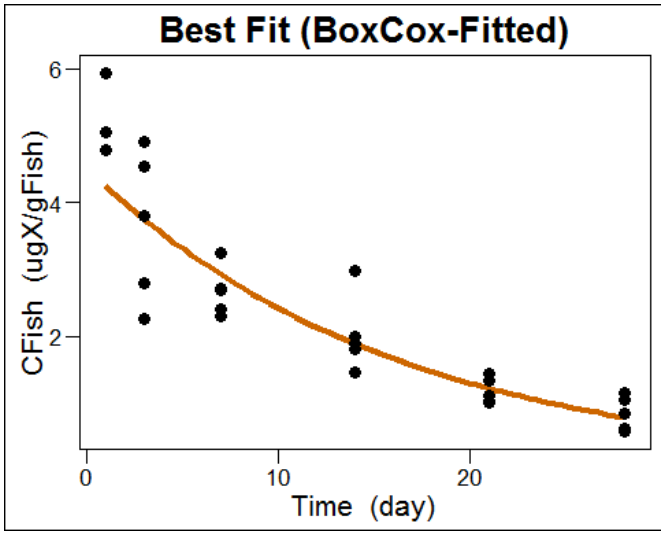

```
##
## ------
##  Shapiro-Wilk normality test
##
## data:  stdres
## W = 0.98541, p-value = 0.9543
##
##
## ------

##
##  Runs Test
##
## data:  as.factor(run)
## Standard Normal = -0.74677, p-value = 0.4552
## alternative hypothesis: two.sided

# --- CALC BEST FIT
runBestFit_Dietary_BoxCox() -> rbf.d.bc

# --- PLOT BEST FIT
plotBestFit_Dietary_BoxCox()
```



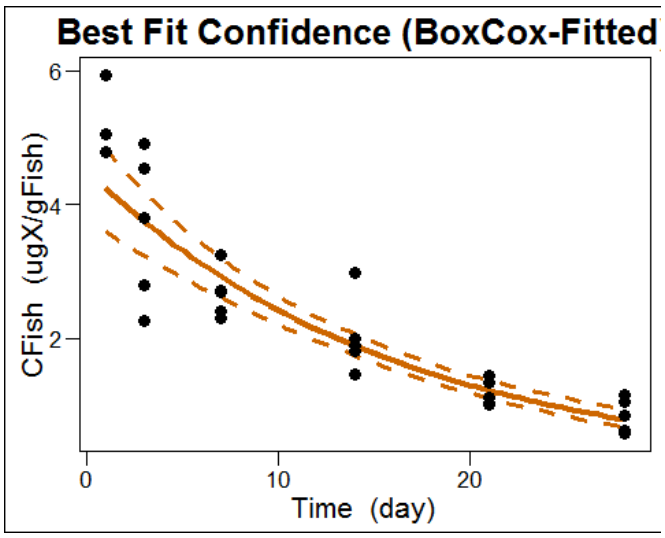

```
# --- BAYESIAN BOOTSTRAP
bootModel_Dietary_BoxCox() -> bbm.d.bc

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary_BoxCox() -> rcf.d.bc

# --- PLOT MODEL CONF LIMITS
plotConfFit_Dietary_BoxCox()
```




```
# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Dietary_BoxCox()
```




```
# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Dietary_BoxCox()
```




```
# --- BCF SUMMARY TABLE
summaryTable_Dietary_BoxCox(digits = 5)

##          Estimate Std.Error      2.5%    97.5%
## C0d      4.5253    0.34929   3.8407    5.21
## k2       0.062546  0.004779  0.05318   0.07191
## k2g      0.025246  0.004779  0.01588   0.03461
## kf       0.023013  0.002305  0.018496  0.02753
## alpha    0.76711   0.076827  0.61653   0.91769
## BMFK     0.36794   0.017431  0.33378   0.40211
## BMFKg    0.91157   0.1003    0.71498   1.1082
## thalfg 27.45       5.1958   17.266    37.634
## BMFKgL   2.9405    0.32355   2.3064    3.5747

# === END OF R SCRIPT ====
```

5 APPENDIX

5.1 AQUEOUS EXPOSURE TEST DATA FILE (.CSVY)

(begin of file)

```
---
# Aqueous Exposure Fish Test
filename: Test-Aqueous-CRCase3-2.0ugXperLiter.csvy

# test experiment
type: aqueous exposure fish test
guideline: OECD 305

# where data came from
data source:
    name: CR
    origin:
    comments:

# test chemical and biological species
chemical:
    name: CRCase3
    chemical info:
species:
    name: Danio
    biological info:

# aqueous exposure
cwater:
    value: 2.00
    unit:  ugX/L

# time events
tstart:
    value: 0.00
    unit:  day
tdepur:
    value: 14.00
    unit:  day
tend:
    value: 35.00
    unit:  day

# fish parameters (both assumed)
kgrowth:
    value: 0.0373
    unit:  1/day
lipidfish:
    value: 4.65
    unit:  percent

# measured timeseries
time:
```

```
    name: Time
    unit: day
cfish:
    name: CFish
    unit: ugX/kgW
---
Time,CFish
0.04,10.50
0.08,7.73
0.17,24.12
0.33,128.80
1.00,553.70
2.00,1105.47
4.00,2464.88
7.00,3025.53
9.00,3195.05
11.00,4485.04
14.00,4652.28
14.04,4167.07
14.08,5385.64
14.17,6692.33
14.33,4674.34
15.00,2329.99
16.00,3797.43
18.00,1328.29
21.00,1080.29
29.00,438.57
35.00,128.83
```

(end of file)

5.2 DIETARY EXPOSURE TEST DATA FILE (.CSVY)

(begin of file)

```
---
# Dietary Exposure Fish Test
filename: Test-Dietary-RHCB10-22.1ugXpergFood.csvy

# type of experiment
type: Dietary Exposure Fish Test
guideline: OECD 305

# where data came from
data source:
    name: DM
    origin: OECD Ring
    comments:

# test chemical and biological species
chemical:
    name: RHCB10
    chemical info:
species:
    name: Oncorhynchus?
    biological info:

# food concentration and ingestion rate
cfood:
    value: 22.10
    unit: ugX/gFood
```

```
ingestion:
    value: 0.03
    unit: gFood/gFish/day

# feeding period
tfeed:
    value: 13.00
    unit: day

# lipid fractions
lipidfood:
    value: 15.00
    unit: percent
lipidfish:
    value: 4.65
    unit: percent

# time span of depuration
tdepur:
    value: 0.00
    unit: day
tend:
    value: 28.00
    unit: day


# fish growth
kgrowth:
    value: 0.0373
    unit: 1/day


# measured timeseries
time:
    name: Time
    unit: day
cfish:
    name: CFish
    unit: ugX/gFish
---
Time,CFish
1.0,5.930
1.0,5.050
1.0,4.790
3.0,3.790
3.0,4.530
3.0,2.250
3.0,2.800
3.0,4.900
7.0,2.690
7.0,2.300
7.0,3.250
7.0,2.410
7.0,2.700
14.0,1.450
14.0,2.000
14.0,2.980
14.0,1.890
14.0,1.800
21.0,1.440
21.0,1.330
21.0,1.010
21.0,1.110
21.0,1.020
28.0,0.611
28.0,1.040
28.0,0.573
28.0,0.852
28.0,1.150
```

(end of file)

5.3 AQUEOUS EXPOSURE TEST R SCRIPT

```
# === BEGIN OF R SCRIPT ====
# _________________________
#
#     RUN-Aqueous.R
# _________________________
#
# --- bcmfR [0.3-2] R Script


# === LOAD PROGRAM ====

# --- ATTACH PACKAGE
library(bcmfR)

# --- SHOW PROGRAM VERSION
PROG.Version


# === READ DATA ====

# --- READ AQUEOUS TEST DATA
readTest_Aqueous("TEST-Aqueous-CRCase3-2.0ugXperLiter.csvy")

# --- SHOW TEST PARAMS
TEST.Aqueous.Param


# === UNTRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Aqueous()

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous()

# --- CALC BEST FIT
runBestFit_Aqueous() -> rbf.a.un

# --- PLOT BEST FIT
plotBestFit_Aqueous()

# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous() -> bbm.a.un

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous() -> rcf.a.un

# --- PLOT MODEL FIT AND CONF LIMITS
plotConfFit_Aqueous()

# --- BCF SUMMARY TABLE
summaryTable_Aqueous(digits = 5)


# === LN-TRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Aqueous_Ln()

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous_Ln()

# --- CALC BEST FIT
runBestFit_Aqueous_Ln() -> rbf.a.ln

# --- PLOT BEST FIT
plotBestFit_Aqueous_Ln()

# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous_Ln() -> bbm.a.ln

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous_Ln() -> rcf.a.ln

# --- PLOT MODEL CONF LIMITS
plotConfFit_Aqueous_Ln()

# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Aqueous_Ln()

# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Aqueous_Ln()
```

```
# --- BCF SUMMARY TABLE
summaryTable_Aqueous_Ln(digits = 5)


# === BC-TRANSFORMED FIT ====

# --- BOX-COX PLOT AND LAMBDA ESTIMATE
modelTrans_BoxCox_Aqueous() -> lambda.est

# --- SHOW LAMBDA FIT AND CONF LIMITS
lambda.est

# --- FIT BCF MODEL
fitModel_Aqueous_BoxCox(lambda = lambda.est["fit"])

# --- MODEL DIAGNOSTICS
modelDiag_Aqueous_BoxCox()

# --- CALC BEST FIT
runBestFit_Aqueous_BoxCox() -> rbf.a.bc

# --- PLOT BEST FIT
plotBestFit_Aqueous_BoxCox()

# --- BAYESIAN BOOTSTRAP
bootModel_Aqueous_BoxCox() -> bbm.a.bc

# --- CALC MODEL CONF LIMITS
runConfFit_Aqueous_BoxCox() -> rcf.a.bc

# --- PLOT MODEL CONF LIMITS
plotConfFit_Aqueous_BoxCox()

# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Aqueous_BoxCox()

# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Aqueous_BoxCox()

# --- BCF SUMMARY TABLE
summaryTable_Aqueous_BoxCox(digits = 5)

# === END OF R SCRIPT ====
```

5.4 DIETARY EXPOSURE TEST R SCRIPT

```
# === BEGIN OF R SCRIPT ====
# _____________________________
#
#     RUN-Dietary.R
# _____________________________
#
# --- bcmfR [0.3-2] R Script


# === LOAD PROGRAM ====

# --- ATTACH PACKAGE
library(bcmfR)

# --- SHOW PROGRAM VERSION
PROG.Version


# === READ DATA ====

# --- READ DIETARY TEST DATA
readTest_Dietary("TEST-Dietary-RHCB10-22.1ugXpergFood.csvy")

# --- SHOW TEST PARAMS
TEST.Dietary.Param


# === UNTRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Dietary()

# --- MODEL DIAGNOSTICS
modelDiag_Dietary()

# --- CALC BEST FIT
runBestFit_Dietary() -> rbf.d.un

# --- PLOT BEST FIT
plotBestFit_Dietary()

# --- BAYESIAN BOOTSTRAP
```

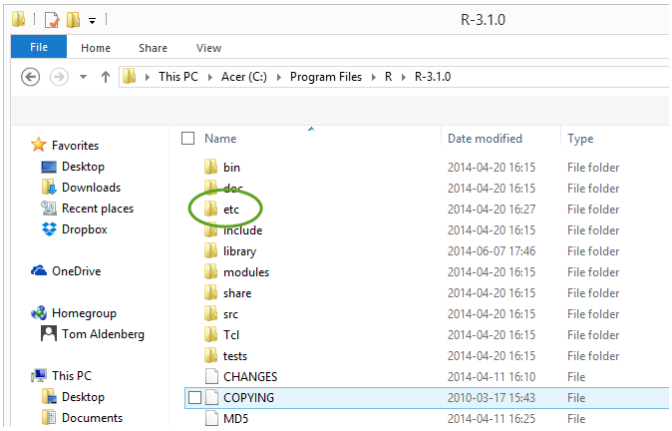
```
bootModel_Dietary() -> bbm.d.un

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary() -> rcf.d.un

# --- PLOT MODEL FIT AND CONF LIMITS
plotConfFit_Dietary()

# --- BCF SUMMARY TABLE
summaryTable_Dietary(digits = 5)


# === LN-TRANSFORMED FIT ====

# --- FIT BCF MODEL
fitModel_Dietary_Ln()

# --- MODEL DIAGNOSTICS
modelDiag_Dietary_Ln()

# --- CALC BEST FIT
runBestFit_Dietary_Ln() -> rbf.d.ln

# --- PLOT BEST FIT
plotBestFit_Dietary_Ln()

# --- BAYESIAN BOOTSTRAP
bootModel_Dietary_Ln() -> bbm.d.ln

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary_Ln() -> rcf.d.ln

# --- PLOT MODEL CONF LIMITS
plotConfFit_Dietary_Ln()

# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Dietary_Ln()

# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Dietary_Ln()

# --- BCF SUMMARY TABLE
summaryTable_Dietary_Ln(digits = 5)


# === BC-TRANSFORMED FIT ====

# --- BOX-COX PLOT AND LAMBDA ESTIMATE
modelTrans_BoxCox_Dietary() -> lambda.est

# --- SHOW LAMBDA FIT AND CONF LIMITS
lambda.est

# --- FIT BCF MODEL
fitModel_Dietary_BoxCox(lambda = lambda.est["fit"])

# --- MODEL DIAGNOSTICS
modelDiag_Dietary_BoxCox()

# --- CALC BEST FIT
runBestFit_Dietary_BoxCox() -> rbf.d.bc

# --- PLOT BEST FIT
plotBestFit_Dietary_BoxCox()

# --- BAYESIAN BOOTSTRAP
bootModel_Dietary_BoxCox() -> bbm.d.bc

# --- CALC MODEL CONF LIMITS
runConfFit_Dietary_BoxCox() -> rcf.d.bc

# --- PLOT MODEL CONF LIMITS
plotConfFit_Dietary_BoxCox()

# --- PLOT BEST FIT ON ORIGINAL SCALE
plotInvBestFit_Dietary_BoxCox()

# --- PLOT CONF FIT ON ORIGINAL SCALE
plotInvConfFit_Dietary_BoxCox()

# --- BCF SUMMARY TABLE
summaryTable_Dietary_BoxCox(digits = 5)

# === END OF R SCRIPT ====
```

5.5 R-INSTALLATION CONFIG

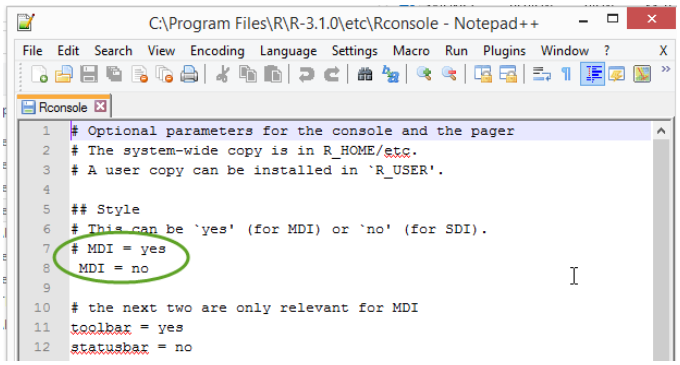
The default folder location of R for Windows 10 looks like this:

C:\Program Files\R\R-3.2.4

In this directory, there are several folders: bin, doc, etc, include, library, and so forth.



The etc folder contains a file: console. Edit this file, e.g. with Notepad++ (see below) and change MDI (Multiple Document Interface) to `MDI = no`, that is: SDI (Single Document Interface).



SDI is more suitable for operating GUI-dependent R-programs.

In the file Rprofile.site, you might want to change the default paper size (= letter) to A4. And you may set a local CRAN site to install packages: