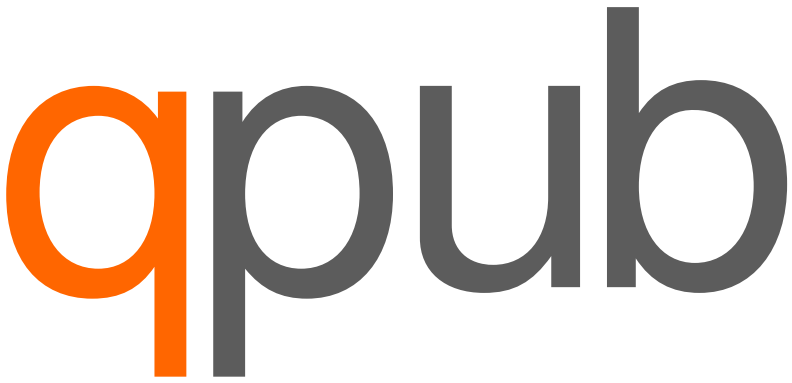

qpub

Version	0.1.0
Date	2019-06-01
Title	Quantification of Peptides using Bayesian inference
Authors	Sarah Henze, Debdas Paul, & Juliane Liepe
Maintainers	Sarah Henze, Debdas Paul, & Juliane Liepe
Depends	R (>= 3.5.0)
Description	Estimate conversion factors for the peptides from the mass spectrometry data
URL	Github link

Contents

Nomenclature

Sets of numbers

Symbol	Description
$\mathbb{N}$	natural numbers
$\mathbb{Z}$	integers
$\mathbb{Z}_{>0}$	positive integers
$\mathbb{R}$	real numbers
$\mathbb{R}_{\geq 0}$	non-negative real numbers
$\mathbb{R}_{>0}$	positive real numbers
$\mathbb{R}^d$	d-dimensional real field

Statistics & probability

Symbol	Description
$\mathrm{Var}[\delta]$	variance of δ
$\mathbb{E}[\delta]$	expectation of δ
$\mathbb{P}(x)$	probability of x

Acronyms

Immunology

APP	antigen presentation pathway
MHC	major histocomaptibility complex

Techniques

MS	mass spectrometry

Algorithms

QME	quantification with minimal effort (Mishto et al., 2012)
MCMC	Markov chain Monte Carlo
M-H	Metropolis-Hastings (Metropolis et al., 1953; Hastings, 1970)

1 Introduction

The central dogma in molecular biology describes how proteins are generated inside a cell. Protein dynamics are, however, tightly regulated not only by the rate of protein synthesis, but also by protein degradation through proteases. The resulting (poly)peptides from protease digestions can be further degraded into single amino acids and recycled for protein metabolism. Proteases can also produce functional peptides, which regulate signalling processes. One example is the antigen presesntation pathway of the immune response. Peptides produced through various proteases, such as proteasome, metalloproteases, and others, are subsequently transported to the cell surface where they are displayed on MHC-I/II molecules,to ensure an efficient imune response.
Understanding the dynamics of protein degradation and peptide production requires quantitative information. This can be obtained through mass spectrometry (MS) measurement of protein or polypeptide digestions in vitro with purified proteases. Mass spectrometry, however, is only semi-quantitative, because the MS signal, i.e. the extracted ion peak area, is strongly influenced by the characteristics, e.g. peptide sequence, of the measured peptides. The MS ion peak areas resulting from different peptides can therefore not be directly copared (Bassani-Sternberg et al., 2015; Cox et al., 2014; de Graaf et al., 2014; Liepe et al., 2016, 2019). To circumvent this, labelling techniques have been developed. While labelling techniques resulted in great advance in understanding protein metabolism, they still only provide relative measures of protein / peptide abundance. Absolute quantification of peptides using MS can be ashieved through titration of every peptide of interest using synthetic peptide equivalents. This is, however, very laborious and cost inefficient when quantifying many peptides.
To overcome this limitation, algorithms have been developed that aim to estimate for each peptide a conversion factor that relates the MS ion peak area to absolute peptide amounts. Those algorithms make use of the conservation of mass in closed in vitro systems by estimating conversion factors for each peptide that minimise the mass loss of amino acids between degraded substrate and generated peptide products. For less specific proteases, such as the proteasome, many peptide products can be observed and for all of them conversion factors need to be estimated. This results in a parameter estimation problem with a high dimension search space, that imposes a challenge for most optimisation routines.
We recently developed a tool to overcome those problems using Bayesian inference. The algorithm, named QPuB (Quantification of Peptides using Bayesian inference), employs a generalised adaptive Metropolis (AM) algorithm with global adaptive scaling to estimate the full posterior distribution of conversion factors. Here, we present an R package, which implements QPuB in a flexible and user friendly manner.
In this manual we provide installation guides, an overview about the package and step-by-step examples on how to use QPuB. We furthermore provide an appendix that describes the working principles of QPuB and gives an overview about the parameter estimation techniques used in the implementation.

2 Elements of QPuB

QPuB is organized into three modules namely the **INPUT**, **ALGORITHM**, and **OUTPUT** (see Figure 2.1). The **INPUT** module takes the MS signal intensities of the substrate and all major peptide products. Along with that, the algorithm also requires the specification of protease, for e.g. whether the protease is an endo or an exopeptidase. As optional inputs, users can provide the titration measurements as well as the time point sequence following which the respective intensities will be compared (see Section 2.3.1). In addition, the **INPUT** module must be provided with a .txt file called **input.txt** that contains values of all the parameters (numerics as well as logicals) required to execute the algorithm and generates the results afterwards.
Upon receiving all the input parameters, the **ALGORITHM** module of QPuB invokes an adaptive Metropolis (AM) sampling algorithm (see Appendix 6.2) to generate samples from the posterior densities of the conversion factors.
As the sampling proceeds, the **OUTPUT** module generates the histograms (excluding burn-In) and traces (including burn-In) of the Monte Carlo samples and as well as the residual plots for each conversion factors. Moreover, trace plots for the nuisance parameters such as the punishment parameter and the variance in the data, are also generated. At the end of simulation, the Monte Carlo samples are saved as **.RDate** format and a file containing the summary statistics of the chain is generated. In case the titration data is provided at the beginning, the **OUTPUT** module also calculate the concentrations of the products from their intensities and the estimated conversion factors using the Eq. (2.1.1) and generates the corresponding plot.

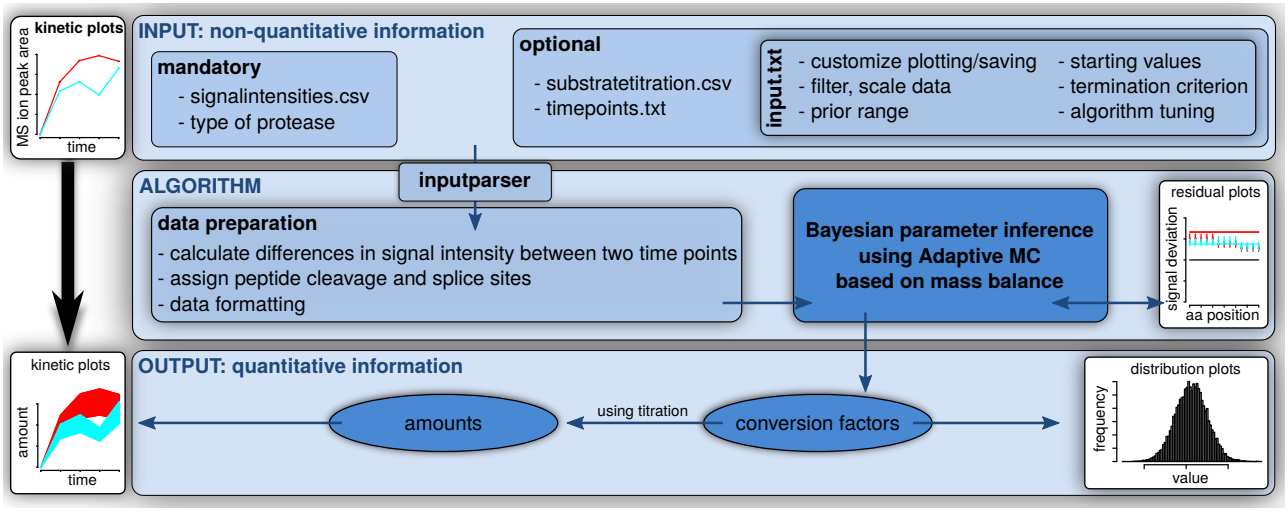


Figure 2.1: **Schematic of QPuB package**

2.1 Principles of QPuB

QPuB is based on the following two principles:

- Linear correlation between the signal intensities and the conversion factors.

- Mass conservation of peptide products over time assuming the *in vitro* system of digestion as a closed system.

Below, we the explain in detail the two principles mentioned above.

Linear correlation

This principle is based on (Peters et al., 2002, Figure 1), where the authors experimentally observed a linear relationship between signal intensities, denoted as $s \in \mathbb{R}$, and the concentrations of products, denoted as $c \in \mathbb{R}$. The concentrations are expressed in picomoles. Mathematically, the relationship translates to the following:

$$c_i = v_i s_i, \;\; i \in [1, P] \tag{2.1.1}$$

where P is the total number of products, $v_i \in \mathbb{R}^+$ is the unknown conversion factor for the i^{th} product. The conversion factor of the substrate is set to 1.

Mass conservation

The second principle that we make use of is the law of mass conservation. Since we consider a closed system of *in vitro* digestions, we know that each amino acid present in the solution at time $t = 0$ is still present at any time $t > 0$. However, depending upon the precision of the mass spectrometer, small peptide products may remain undetected. This systemic loss of mass has to be accounted for. On the other hand, a gain of mass is impossible, because no amino acids are created during the digestion. Formally, the sum of amino acids of the products should be equal to the amount of substrate degraded since the beginning, for every amino acid position and for every time points:

$$\sum_{i=1}^{P} c_i(t) b_{ia} = c_0(t) - c_0(0) = \Delta c_0(t) \quad \forall a, t, \tag{2.1.2}$$

where a denotes the amino acid position, t denotes the time points, and $c_i(t)$ denotes the concentration of product i at time t. The factor $b_{ia} \in [0, 1]$ is the probability that the product i contains amino acid a. The amount of substrate degraded since time $t = 0$ is $\Delta c_0(t)$. Now, substituting the expression for c_i according to equation (2.1.1), we have

$$\sum_{i=1}^{P} v_i s_i(t) b_{ia} - \Delta c_0(t) = 0 \quad \forall a, t. \tag{2.1.3}$$

Since this difference should be zero for every a and t, also the sum over all a and t must be zero:

$$\sum_{t=1}^{T} \sum_{a=1}^{A} \Big[\sum_{i=1}^{P} v_i s_i(t) b_{ia} - \Delta c_0(t) \Big] = 0. \tag{2.1.4}$$

The left-hand side then gives us our objective function, which should be as close to zero as possible. In total, our aim thus is to find values for the conversion factors that minimize the difference between our simulation results and the true amount of products:

$$\min_{v_i \in \mathbb{R}^+} \sum_{t=1}^{T} \sum_{a=1}^{A} \Big[\sum_{i=1}^{P} v_i s_i(t) b_{ia} - \Delta c_0(t) \Big]. \tag{2.1.5}$$

Instead of finding the sole minimizer of (2.1.5), QPuB uses a Bayesian approach to obtain a full distribution of possible values.

Calculation of the concentrations

Once a distribution for the conversion factor of every peptide product is obtained, the concentrations can be calculated by fitting the data using a linear model $y = mx+b$ according to Eq. (2.1.1). Because the conversion factor of the substrate is set to 1, the expression for the concentrations of the substrate takes the following form

$$c_i = \frac{v_i s_i - b}{m}, \tag{2.1.6}$$

where the slope m and the intercept b can be obtained from the kinectics of the substrate.

2.2 The software

As a package QPuB contains the following R scripts:

Scripts	Functionalities
runQPuB	• the main routine which is invoked using command line arguments (see Chapter 4 and 5)
inputparser	• creates output folder • parses input specifications from a text file or set defaults • reads the experimental data • reads the time points to compare, if provided • reads the titration data file, if provided
datapreperation	• filters data • plots input signal kinetics • assign cleavage and splice sites of peptide products • creates position probability matrix • calculates the change in MS signal intensities between time points • scales the data
MCMCroutine	• runs the random walk adaptive Metropolis algorithm with generalized adaptive scaling (Andrieu and Thoms, 2008) • definition of the prior distribution • definition of the likelihood function • definition of the proposal distribution

additionalfunctions	• contains all functions needed during and after the QPuB routine • during: – plot distributions and history of parameters – plot current mass deviation • after: – provide summary of the parameter chain – calculate absolute concentrations of all reagents based on the titration data if provided – plot kinetics of the absolute concentrations

Utilities of the scripts mentioned above are explained in the next section.

2.3 Dissecting the modules

In this section, we will go step-by-step description of the all the necessary inputs, algorithms and output routines in the QPuB.

2.3.1 Input

Mandatory input to the QPuB routine are the experimental data and some input parameters. Optional input include the substrate titration and a file of time points to be compared. For input folder structure see Section 2.3.2.

Experimental Data (Mandatory)

The data provided must be a kinetic data from mass spectrometry that contains signal intensities of an *in vitro* digestion over a few hours time for all major products. We strongly advise to provide more replicates for more certainty. For each replicate, a separate data file of a **.csv** format that contains the amino acid sequences and the numeric values, must be provided. In that file the first column must contain the sequences and all other columns must contain the signal intensities over time. Each row of the file corresponds to one peptide product. The first row must always contain the substrate. The header must contain numeric type time points having the same unit. User has the flexibility to name the **.csv** files.

```
sequence,0,0.5,1,1.5,2,3,4
RTKAWNRQLYPEW,8.89E+11,5.55E+11,2.75E+11,1.38E+11,8.35E+10,2.01E+10,3.38E+09
RTKAWNR,6.02E+08,2.68E+10,3.67E+10,3.92E+10,4.70E+10,5.41E+10,4.14E+10
RTKAWNRQ,5.58E+07,6.40E+08,9.25E+08,8.88E+08,1.18E+09,1.23E+09,7.77E+08
QLYPEW ,1.14E+10,2.87E+11,4.30E+11,4.94E+11,5.55E+11,6.34E+11,5.14E+11
RTKAWNRQL,1.06E+07,2.43E+08,3.90E+08,4.23E+08,4.54E+08,3.99E+08,2.91E+08
...
```

Parameters (Mandatory)

The input parameters are specified in a text file named **input.txt**. This name is mandatory and **must not be changed**. There are numerical as well as logical parameters. Logical parameters can be set to `TRUE` or `FALSE`. Other possible ways of spelling: `True, true, T`. The feasible values for the numerical parameters are defined below. Numbers may be formatted as decimal (2000000) or as scientific (2e6, 20e5), but should not contain asterisk notation '`*`' (`2*10**6`) or a caret notation '`^`' (`2*10^6`).
In the following we discuss the contents of **input.txt** specifying low level and advanced level input parameters. For datasets having small number of parameters, we provide default values for most of the parameters such that there is only one or two mandatory inputs that the user must specify.

```
#################################################################
# INPUT FILE FOR QPUB PACKAGE
#################################################################

# WARNING: THIS FILE MUST NOT BE RENAMED!
# Mandatory name: input.txt

#####################################################
# LOW LEVEL INPUT
#####################################################

########## MANDATORY INPUT

# mechanism of the underlying enzyme: 'endopep', 'exopep'
enzyme: 'endopep'

# amount of substrate loaded for kinetics
loaded: 200
```

Description
Here, the user must choose between exopeptidase '`exopep`' and endopeptidase '`endopep`' to specify the nature of the protease. In case the substrate titration is provided, the second mandatory input is the amount of substrate loaded in the kinetics. The unit must be identical to the titration data file.

```
########## OPTIONAL INPUT WITH DEFAULTS

### DIAGNOSTIC PLOTS
# plot distribution and history of parameter chain
plot_chain: TRUE
# plot residuals: mass deviation in current iteration
plot_residuals: TRUE
# plot every x iterations
freq_plot: 10000
# plot detailed: every 100 iterations before iteration 1000, every 1000
# iterations under iteration 10000
plotdet: TRUE
# generate vector graphics instead of flat png (default: FALSE)
vecplot: FALSE

### BACKUP
# save chain every x iterations
```

```
freq_save: 10000
```

Description
As intermediate outputs, QPuB creates diagnostic plots of the Markov chain in the folder plots_diagnostics and saves the current chain of parameters to a RData-file.
`Plot_chain`: The chain is shown using a line plot of the parameter value over the course of the iterations. The resulting distribution is shown in form of a histogram with the frequency of parameter values.
`Plot_residuals`: The residual plot depicts the mass deviation of the QPuB outcome from the true solution. Following the rule of mass conservation, the amount of every amino acid should sum up to the amount of substrate degraded between two time points to be compared, indicated by a horizontal line for each replicate. The boxplots represent the distributions of the concentrations for every amino acid, arising from the outcome distributions of the conversion factors. The user can choose whether these diagnostic plots of the Markov chain should be produced.
`freq_plot`: Specifies the frequency at which the plots will be generated and saved. The default value is 10000.
To inspect the behaviour of the chain in the first few iterations, the logical parameter `plotdet` is set to `TRUE`. In this case, the diagnostic plots are produced in every 100 iterations up to 1000 iterations, and in every 1000 iterations until 10000 iterations. After that, the plots are produced according to the specified value for `freq_plot`. This detailed plotting can be turned off by setting `plotdet` to `FALSE`. The diagnostic plots are saved as compressed pdf-files. All other figures are exported as **.png** format. In order to obtain the plots as vector graphics, the user needs to set the logical parameter `vecplot` to `TRUE`.

```
#####################################################
# ADVANCED INPUT
#####################################################

### DATA MANIPULATION
# Scale products to match order of magnitude of substrate
scaleprod: TRUE
# Filter out products which have a signal intensity
# below the threshold.
# If TRUE then threshold must be set.
filterprod: FALSE
threshold: 1e6
```

Description
The signal intensities of the peptide products are scaled to bring the substrate and the product intensities to the same order of magnitude. One can set the parameter `scaleprod` to `FALSE` if the scaling mentioned above is not required.
For large datasets, the dimension may be reduced by excluding some of the peptide products that have low intensities (`filterprod: TRUE`) below a threshold (`threshold: threshold_value`). This filtering is pragmatic as peptides having such low intensities contribute less to the mass conversion. We propose a `threshold` value of of 10^6 for datasets larger than 100 products.

```
### UNIFORM PRIOR DISTRIBUTION
# lower and upper bound of uniform prior range
# of conversion factors for all products
conv_lower: 1e-4
conv_upper: 1e4
# lower and upper bound of uniform prior range of the standard deviation
```

```
sigma_lower: 1e-4
sigma_upper: 1e4

### STARTING VALUES OF THE PARAMETERS
# initial values of the conversion factors of all products
conv_start: 1
# initial value of the standard deviation in the data
sigma_start: 10
# initial value of the punishment parameter. Choose a value between 0 and 1
pun_start: 0.1
```

Description

For the prior range, choose any positive values satisfying `lower` $<$ `upper`. If the product signals are scaled to fit the substrate (`scaleprod: TRUE`, default), then the prior range of the conversion factors is narrowed down by 7 orders of magnitude centered around the value 1, because.... For both the conversion factors and the standard deviation, the default prior range is $[10^{-4}, 10^{4}]$.
For conversion factors, we choose 1 as the starting value. Along with that, an initial value of 10 for the standard deviation has turned out to be a reasonable choice for our examples. The user may choose any starting values in the prior range specified above. Moreover, the default starting value of the punishment parameter is set to 0.1. The punishment parameter takes a value between 0 and 1. A value of 0 signifies strictest avoidance, whereas a value of 1 impose a zero avoidance for the mass gain.

```
### ALGORITHM PARAMETERS
# optimal acceptance rate
OptAccRate: 0.234
# track performance of the run: print acceptance rate and effective sample size
Track: TRUE
# Exponent for the sequence to realize vanishing adaptation.
# Choose a value between 0 and 1.
GammaExponent: 0.5
# Fraction of the Markov chain to consider as burn-in, 0.25 means 25% burn-in
burnFrac: 0.5

### PROPOSAL FUNCTION
# burn-in of the internal Gibbs sampler
burnGibbs: 1000
# Thinning parameter of the internal Gibbs sampler
thinGibbs: 20
```

Description

A typical choice of the optimal acceptance rate for the Metropolis algorithm is 0.234 (Roberts et al., 1997). The user may choose any value between 0 and 1. With the setting `Track: TRUE` the current acceptance rate and the current effective sample size are printed at every 1000 iterations.
Vanishing adaptation is achived via the so called Gamma function with exponent `GammaExponent`. Any value between 0 and 1 may be chosen, the default is 0.5. For details on the Adaptive Metropolis algoritm see the Appendix 6.2.
When the Markov chain starts at a region of low likelihood, it will probably take a while to arrive at the region of the target distribution. Therefore, initial samples have low contribution towards the posterior distribution, and are typically discarded as *burn-in*. We use 50% of the chain as burn-in by default. The value of the paramter `burnFrac` decides the fraction of the samples that will be discarded as burn-in. The parameter takes a value between 0 and 1. For example, a value of 0.25 means that 25% of the samples will be discarded as burn-in.

The R function *rtmvnorm* generates proposal samples from a truncated multivariate normal distribution using a Gibbs sampler. The parameter `burnGibbs` specifies the amount of burn-in samples for the Gibbs sampler. The default value of `burnGibbs` is 1000. In addition, the degree of uncorrelated samples is taken care by the parameter `thinGibbs` which has a default value of 20.

```
### TERMINATION CRITERION
# TRUE: termination via effective sample size calculation.
# FALSE: enables the MaxIter variable, which is the maximum number of
# iterations the MCMC will run
TerminationCriterion: TRUE
MaxIter: 1e6
# iteration when to start checking the criterion
ESSiter: 10000
# Confidence level for calculating the effective sample size
# using the minESS routine
ConfLevel: 0.05
# Monte Carlo error in percent for the minESS routine
Tolerance: 0.05
```

Description
The algorithm QPuB terminates based on the effective sample size (ESS) of the Markov chain. The algorithm checks for convergence at every 1000 iterations after iteration `ESSiter: 10000` by default or accoring to the user input. It is recommended to choose a higher value of, for e.g.`ESSiter` > 1000, in order to constitute a positive definite covariance matrix of the samples (Vats et al., 2019). `ConfLevel` and `Tolerance` are the internal parameters of the *minESS* routine and signifying the cofidence level for the ESS, and the fraction of the accepted Monte carl error in parameter estimation, respectively. Both the paramters takes a value between 0 and 1. For example, a value of 0.05 for `ConfLevel` means that the calculated value of the ESS is within the 95% level of confidence. For large and complex datasets, reaching the minimum number of ESS can take a very long time. Upon visual inspection, it may appear that the chain has already been converged, though mathematically it might not be. In such cases, the user can opt for a termination criterion based on the maximum number of iterations by setting `TerminationCriterion` to `FALSE`, and assiging a desired number to the parameter `MaxIter`. The default value of `MaxIter` is set to 10^6 meaning that the algorith will terminate after 10^6 iterations. Additionally, for high dimensional problems, it may happen that the covariance matrix of the target distribution becomes singular or ill-conditioned. In such cases routines *tmvtnorm* and multiESS will not work. But, QPuB avoid such premature termination by first inspecting the covarince matrix, and then automatically switches to the maximum iteration mode if the matrix is found to be singular.

Timepoints (Optional)

In QPuB, not the signal intensities per se but the differences in intensities between a pair of time points are taken into account. Using the input file **timepoints.txt**, the user may specify, which time points should be compared. The name of the file is mandatory and must not be changed. If no **timepoints.txt** file is provided, two consecutive time points are considered by default:

```
0 1
1 2
2 3
3 4
```

Description
The file must contain numbers arranged in two columns. For each row, intensities of the first column are subtracted from the intensities at the second column. Regardless of the unit and numeric values of the time points in the data file, the timepoints file has to be provided with the number of time points.

number	0	1	2	3	4
hours	0	0.5	1	1.5	2
minutes	0	30	60	90	120

User is recommended to manipulate the sequence of time point indices for comparison if certain time points are supposed to be weighted more or less than others...

Titration (Optional)

To calculate absolute concentrations from the estimated conversion factors and the input signal intensities, QPuB requires the substrate titration data as input. Titration measurements are specified in a csv-file, which has the following structure:

```
"pmol","rep1","rep2"
0,8738783.03163,33850140.1962817
6.5,125855871751.115,126066103768.273
12.5,228222898282.954,252195890030.762
25,518494461424.112,513196824959.805
50,1029977038639.7,1004577689796.97
75,1323855190838.57,1263027816310.47
100,1654653991391.97,1751554906143.47
```

The first column is the amount of substrate loaded into the mass spectrometer and the rest contain the intensities in replicates. The column names are unimportant but a header must be provided, for example: `"amount","rep1","rep2",...`. The name of the file can be user defined and will be specified as one of the command line arguments, see Chapter 4 for further details. If no titration data is provided, then normalized intensities are returned.

2.3.2 The inputparser

QPuB requires a particular input folder structure. The input folder inputfoldername contains the input.txt file, the timepoints.txt file (if provided), the titration csv-file (if provided), and a data folder that contains all the data csv-files.

```
parentdirectory/
|-- inputfoldername/
|    |-- data/
|    |    |-- rep1.csv
|    |    '-- rep2.csv
|    |-- input.txt
|    |-- timepoints.txt
|    '-- titration.csv
'-- QPuB-Output/
     |-- OUTPUT_inputfoldername
     |-- OUTPUT_inputfoldername(1)
     '-- OUTPUT_inputfoldername(2)
```

QPuB creates a output sub-folder QPuB-Output on the fly inside the parent directory of the input folder inputfoldername. Every time the routine is executed, another seperate output sub-folder with the name OUTPUT_inputfoldername is created inside QPuB-Output . If this input-specific sub-folder already exists, it will not be overwritten, but a new sub-folder will be created with the number specifiying its sequence, e.g. OUTPUT_inputfoldername(1). If the user wants to individually name the output folder, it can be done via a flag in the command line (see Chapter 4). Outputs of the algorithm regarding the current run will be saved inside these input-specific sub-folderes. The standard output of the algorithm will be redirected to an ROUT-file, which can be used to monitor the progress of the algorithm.
The inputparser reads and parses the parameters and their values (separated by a colon (see Chapter 2.3.1)) in **input.txt**. If a mandatory parameter is not provided, default values and settings for that parameter will be assigned. Kinetic data specifiying mass spectrometry intensity measurements are stored in the internal variable `dat` upon reading from the data folder.
When the **timepoint.txt** file is porvided, the parser parses the timepoint sequence specified in that file for the calculation of the likelihood. Otherwise, successive timepoints are taken into account. Finally, the parser parses the titration data as well if the same is provided. For a detailed input specifications, see Section 2.3.1.

2.3.3 Data preparation

In this part of the QPuB, first, the dimensions of the datasets are read from the data, such as

- number of replicates
- number of time points
- number of time point comparisons
- number of products
- number of amino acids in the substrate sequence

Next, depending on the mechanism of the protease used, the algorithm determines the amino acid position from where the sequence will be fitted. For an endopeptidase, $a_0 = 0$ Eq. (2.1.2).
For an exopeptidase, the challenge lies in the systematic mass loss. Since the short peptide products are often been undetected by the MS device (the precision varies across the device specifications), the law of mass conservation is violated in those cases. QPuB takes this shortcoming into account by considering only the amino acid positions contained in the shortest major product observed for the calculation of the likelihood.
Upon choosing the filtering option, the products with signal intensities below a threshold are filtered out, because they don't contribute much to the mass balance. We then visualize the input data by plotting the kinetics of signal intensity over time for all peptide products remaining.
To infer whether the peptide products originate from in the substrate, we align the product sequence to the substrate sequence, taking nonuniqueness into consideration. We assign cleaved (PCP) and spliced peptides (PSP) following the convention mentioned in the literature such as in Mishto et al. (2012). For example, if a product can be created by the process of cleavage, then it is assumed to be a PCP.

Example 1. *Assume a substrate with sequence ABCDEFGHIABCD and the products with sequences DEFGH and FGHABC respectively. We number the amino acids in the substrate consecutively* $a = 1, \cdots, 13$. *We decide peptide 1 to be a PCP with the unique position code* `4_8`*, since its sequence is a direct subsequence of the substrate sequence. On the other hand, peptide 2 is a PSP with a nonunique position code of either* `6_8_1_3` *or* `6_8_10_12`*. The peptide number, its sequence and its possible positions are saved in a table* ***identifier.csv****:*

number	*sequence*	*position code*	*position code*	*...*
1	*DEFGH*	`4_8`	-	
2	*FGHABC*	`6_8_1_3`	`6_8_10_12`	
...				

Creation of the position probability matrix: Using the position information, the position probability matrix $b \in P \times A$ is defined. It contains the probabilities of the amino acid $a \in [1, A]$ of the substrate being used in the production of peptide $i \in [1, P]$. The entry b_{ia} takes the following form

$$\text{ppm}(i, a) = \frac{1}{N_a} \in [0, 1], \tag{2.3.7}$$

where N_a is the number of possible origins the amino acid at position a of product i has in the substrate. In Example 1, the frequencies of the single amino acids in the substrate sequence are [2,2,2,2,1,1,1,1,1,2,2,2,2]. Now consider the product $i = 1$, which is the cleaved 5mer DEFGH with positions `4_8`. Following the convention, the positions 4 to 8 get the matrix entries $b_{14} = b_{15} = b_{16} = b_{17} = b_{18} = \frac{1}{1} = 1$, all the other positions have probability 0:

$$\text{ppm}(1, :) = [0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]. \tag{2.3.8}$$

Product 2, on the other hand, does not have an unique position code assigned. The amino acid A, for example could stem either from position 1 or 10 in the substrate, making $N_1 = N_{10} = 2$. We consider both possibilities equally likely. Hence the probability of A in the peptide 2 stemming from position 1 in the substrate is $b_{21} = \frac{1}{2}$. In total, the second row of the position probability matrix reads

$$\text{ppm}(2, :) = [0.5, 0.5, 0.5, 0, 0, 1, 1, 1, 0, 0.5, 0.5, 0.5, 0]. \tag{2.3.9}$$

Calculation of the change in MS signal intensities between time points: For the calculation of the likelihood, the amount of substrate degraded between two time points is required. By default, consecutive time points are considered. But, the user can also provide his/her own time point sequence in **timepoints.txt** as an input (see Section 2.3.1).
We first subtract the background noise at time 0 from all data points. Then the difference between time points is stored in the variable `signalP` for the substrate degraded and in `signalF` for all the products respectively.

Scaling the data: Finally, we scale all signals by the value specified for the parameter `scale_mag` to avoid high orders of magnitude and the product signals by the value specified for the parameter `scale_prod` to match the order of magnitude of the substrate and the products.

2.3.4 The adaptive Metropolis-Hastings algorithm

The parameters to be inferred by the algorithm are the conversion factors v_i for all peptide products. Along with that, the standard deviation `sigma` of the likelihood and the penalty parameter `pun` for mass gain (description below) are also been estimated. Hence, the dimension of the parameter space is $P + 2$. Finally, the parameter set becomes $\Theta = \{v_1, ..., v_P, \sigma, \varphi\}$.

The prior probability distribution expresses the initial belief for the unknown parameters, before any data is observed. When information about the unknown parameters is available, we can assume certain mathematical form for the prior distribution. It is important to not take this information from the data used in the inference. Previously acquired data can be informative, or the posterior resulting from an inference on that data can be used as an informative prior for the new data. To be unbiased, the QPuB algorithm uses a relatively uninformative flat prior, namely a uniform one:

$$\mathbb{P}(\Theta) = \sum_{i=1}^{P+2} \theta_i, \tag{2.3.10}$$

where

$$\theta_i \sim \begin{cases} \mathcal{U}(10^{-4}, 10^4) & \text{for } i \in [1, P+1] \text{ (conversion factors, sigma)} \\ \mathcal{U}(0, 1) & \text{for } i = P+2 \text{ (punishment parameter)} \end{cases}.$$

The range for the conversion factors is chosen like this, because experimental observations suggest a range of the order of magnitude of the conversion factors of around 7 and the factor for the substrate

is set to 1 by default. The range for sigma is chosen like this, because... The range for the penalty parameter is set to $[0, 1]$ to achieve a punishment of mass gain.

The likelihood describes the plausibility of a parameter value given new data. Here we make use of the law of mass conservation and define the objective function from Section 2.1 as our likelihood. Moreover, the likelihood function is truncated punishing the mass gain more than mass loss, because amino acids can remain undetected but definitely not created:

$$\mathbb{P}(\Theta|D) = \sum_{t=1}^{T}\sum_{a=1}^{A}\Big[\sum_{i=1}^{P} v_i s_i(t) b_{ia} - \Delta c_0(t)\Big], \tag{2.3.11}$$

where

$$\sum_{i=1}^{P} v_i s_i(t) b_{ia} - \Delta c_0(t) \sim \begin{cases} \mathcal{N}(0,\sigma) & \text{on } [-\infty, 0] \text{ (mass loss)} \\ \mathcal{N}(0,\varphi\sigma) & \text{on } (0,\infty] \text{ (mass gain)} \end{cases}.$$

The proposal function: Like in most biological applications, we use a multivariate normal distribution with the current value as the mean. We truncate on the prior interval as follows:

$$g(\theta'|\theta) \sim \mathcal{N}_{tr}(\theta, \sigma_2, a, b) \tag{2.3.12}$$

where

$$\begin{cases} a = 10^{-4}, b = 10^4 & \text{for } \theta_i \in [0, P+1] \text{ (conversion factors, sigma)} \\ a = 0, b = 1 & \text{for } \theta_i = P+2 \text{ (punishment parameter)} \end{cases}.$$

Presetting a good scale σ_2 for the proposal function is difficult. To overcome this problem, we use adaptive scaling, which learns a good scaling on the fly. see Appendix 6.2 for more details. The starting value of the scale parameter is set to $2.38/\sqrt{P+2}$ for all parameters. The sequence implementing the *vanishing adaptation* strategy has the following form $\Gamma(j) = j^{-\gamma}$ (Haario et al., 2001), where j is the current iteration number and γ is specified in the input as `GammaExponent` with a default of 0.5. In QPuB, we use the Rao-Blackwellised adaptive Metropolis algorithm. See Appendix 6.3 for more details.

Convergence criterion: Convergence is based on either the ESS or reaching maximum number of iterations provided by the user as previously discussed. In case the user stops the QPuB routine before termination, the R script **postprocessing.R** provided should be executed to post process the current chain and return the final results.

2.3.5 Post processing

After the termination of the adaptive Metropolis algorithm, the chain needs to be back scaled by $\texttt{scale\_prod}^{-1}$, because we scaled the data by `scale_prod` during data preparation. The resulting chain is saved as `chain_backscaled` in an RData-file. For further analysis 50% of the chain is removed as burn-in, or according to user input. Summary statistics of the chain is returned in the form of minimum, lower quartile, median, upper quartile, maximum, mean, and standard deviations. Besides, the absolute concentrations of the peptide products are calculated, saved in csv-files and plotted. If the substrate titration is provided, a linear model is used to fit the mean over the replicates. See figure `substratetitration_before.png`. The signal intensity for the amount loaded (provided as input parameter `loaded`) in the kinetics is inferred from this first fit and then used to normalize the titration to correspond to the signal intensity at time zero in the kinetics for all replicates. This normalized titration data is again fitted using a linear model, for each replicate separately, see Figure `substratetitration_after.png`. The intercept and the slope are then used to determine the absolute amounts via the formula (2.1.6) described above. The substrate concentration is calculated, saved in csv-files and plotted as concentration over time, see figure `substrate_conc.png`. The substrate degraded is plotted, see figure `substrate_degraded.png`. For the products, we calculate the 0.05, 0.5 and 0.95 quantiles of the concentrations via formula (2.1.6) using the quantiles of the distributions of the conversion factors respectively. The numeric values are saved in csv-files in the folder

concentrations and the kinetics are plotted as concentration over time for every peptide product in the folder plots_concentrations. To know which plot belongs to which peptide, use identifier.csv.
If the titration data is not provided, we use only the correlation (Eq. 2.1.1) to obtain normalized signals. These can be used to qualitatively compare the peptide products but they are not...

2.3.6 Output

After termination of the algorithm, the sub-folder OUTPUT_inputfoldername contains all the generated output. It has the following structure:

```
OUTPUT_inputfoldername
|-- boxplot_chain.pdf %summary plot
|-- chain.RData
|-- chain_backscaled.RData
|-- identifier.csv
|-- concentrations
|    |-- conc_five_1.csv
|    |-- conc_five_2.csv
|          :
|          :
|    |-- conc_five_R.csv
|    |-- conc_median_1.csv
|    |-- conc_median_2.csv
|          :
|          :
|    |-- conc_median_R.csv
|    |-- conc_ninetyfive_1.csv
|    |-- conc_ninetyfive_2.csv
|          :
|          :
|    '-- conc_ninetyfive_R.csv
|-- plots_concentrations
|    |-- peptide1.png
|    |-- peptide2.png
|          :
|          :
|    |-- peptideP.png
|    |-- substrate.png
|    '-- substrate_degraded.png
|-- plots_diagnostics
|    |-- chain_100.pdf
|    |-- chain_200.pdf
|          :
|          :
|    |-- chain_Niter.pdf
|    |-- residuals_0.pdf
|    |-- residuals_100.pdf
|    |-- residuals_200.pdf
|          :
|          :
|    '-- residuals_Niter.pdf
|-- plots_inputsignals
|    |-- peptide1.png
```

```
|    |-- peptide2.png
|          :
|          :
|    |-- peptideP.png
|    |-- substrate.png
|    '-- substrate_degraded.png
|-- runQPuB_ROUT.txt
'-- statistics.csv
```

Description
The folder plots_inputsignals is created at the start of the algorithm and contains the kinetics of the signal intensities for every peptide product, the substrate and the substrate degraded. The diagnostic plots (histograms, traces, and residuals) during the run are saved in the folder plots_diagnostics. The number denotes the iteration number. The folder concentrations contains the numeric values of the 0.05, 0.5 and 0.95 quantiles of the absolute amounts in csv-files, one for every replicate respectively. The kinetic plots for peptide products are contained in the folder plots_concentrations. In the plot, the median is depicted by a solid line, the region between 0.05 and 0.95 quantile is the shaded area. Different replicates are denoted by different colors. The concentration of the substrate and the substrate degraded are a single line for every replicate respectively, since the conversion factor is fixed to 1.

boxplot_chain.pdf	boxplot corresponding to the distributions of conversion factors
chain.RData	Markov chain: time series of all parameters
chain_backscaled.RData	Markov chain after backscaling (see Sec. 2.3.5)
conc_five_rp.csv, conc_median_rp.csv, conc_ninetyfive_rp.csv	0.05, 0.5 and 0.95 quantiles of concentration kinetics of all products for the rp^{th} replicate
runQPuB_ROUT.txt	Progress report of the algorithm. The output of every print command directly goes into this file.
chain_j.pdf	Trace plots and distributions of the Markov chain at j^{th} iteration
residuals_j.txt	Residual plots at j^{th} iteration
statistics.csv	Summary statistics of the Markov chain

3 Installation

3.1 Installation of R

QPuB is a software package that runs on the programming language R. To install R on your computer, visit the R webpage www.r-project.org. Users are advised to install an R version $\geq 3.5.0$.

3.2 Installation of dependencies

QPuB requires the following R packages to be installed before executing the main script:

R.utils	processing of command line arguments
tictoc	timing function: CPU time
sys	timing function: system time
tmvtnorm	proposal: truncated multivariate normal distribution
matrixcalc	proposal: check whether a matrix is positive definite
corpcor	proposal: make a matrix postive definite
ggdmc	proposal: generate univariate truncated normal distribution
dqrng	acceptance-rejection: generate fast pseudo random numbers
coda	termination: produce an mcmc object for further analysis of the Markov chain
mcmcse	termination: computation of the effective sample size
matrixStats	postprocess: computation of standard deviation of columns of a matrix

To install these packages, type in the command line

```
$ R
> install.packages("packagename")
```

Alternatively, one can run the following command to install all the packages at once

```
$ R
> install.packages(c("R.utils", "tictoc", "mcmcse", "tmvtnorm",
"corpcor", "dqrng", "coda", "matrixcalc", "ggdmc", "sys", "matrixStats"))
```

3.3 Installation of the QPuB routine

QPuB is free available on GitHub at the following address https://github.com/QuantSysBio/QPuB. The user requires to download the archive QPuB-master.zip and unzip it. The unzipped folder **QPuB-master** has the following structure:

```
QPuB-master/
|-- examples/
|    |    |-- Prot_K386/
|    |    |-- data
|    |    |    |-- k386_1.csv
|    |    |    '-- k386_2.csv
|    |    |-- input.txt
|    |    |-- timepoints.txt
|    |    '-- titration.csv
|    |-- toy_endo3/
|    |    |-- data
|    |    |    |-- endo3_1.csv
|    |    |    '-- endo3_2.csv
|    |    '-- input.txt
|    '-- toy_endo3_nonoise/
|         |-- data
|         |   '-- endo3_nonoise.csv
|         '-- input.txt
|-- LICENSE.md
|-- Documentation.pdf
|-- QPuBGit.Rproj
|-- QPuB/
|    |-- additionalfunctions.r
|    |-- datapreparation.r
|    |-- inputparser.r
|    |-- MCMCroutine.r
|    '-- runQPuB.r
'-- README.md
```

After unzipping, QPuB is ready to be executed. The folder QPuB is the heart of the routine, the rest may be deleted by the user. The user may move the QPuB folder to a desired location. The main script **runQPuB.r** can be invoked either from the folder QPuB or by specifying the relative path to **runQPuB.r** from where the user wants the script to be executed. Details are provided in Chapter 4. The folder examples contains three different data sets of varying complexity. We go through them step by step in Chapter 5. Note that they already have the folder structure required as stated in Section 2.3.2.

4 | Usage

To use QPuB on your data, follow these steps:

1. Make sure everything is properly installed on your computer (see Chapter 3).

2. Make sure the data files and the inputfolder have the right structure (see Sections 2.3.1 and 2.3.2).

3. Open the terminal/command prompt (How to: Linux, Mac, Windows).

4. Go to your working directory:

```
(C4.1) $ cd hostname:~/workingdirectory
```

5. The QPuB main script is executed using flags:

```
(C4.2) $ Rscript -file -infol -outfol -titr
```

QPuB is equipped with facilities to accept and parse command line arguments, where

-file	name/path of the **runQPuB.r**. This input is mandatory.
-infol	name/path of the **input folder**. This input is mandatory.
-outfol	userdefined name of the **output folder**. This input is optional.
-titr	name of the **titration data** csv-file. This input is optional.

Flags can be specified as `-infol inputfolder` or `--infol=inputfolder`.

Depending on your working directory, you also have to provide the path to the runQPuB.r file:

a. working directory is QPuB directory: `-file runQPuB.r`

b. working directory is somewhere else: `-file path\QPuB\runQPuB.r`

Note, that the flag `-file` for the R script can also be omitted. The first argument will always be taken as the name of the R script: `Rscript runQPuB.r`.

Depending on the location of the input folder, you also have to provide the path to the folder:

a. inputfolder is in same parent directory as QPuB: `-infol inputfolder`

b. inputfolder is somewhere else: `-infol path\inputfolder`

In the following section, we will illustrate the usage of QPuB with some examples.

5 Examples

In the following sections, we show a step-by-step execution of the QPuB algorithm on two artificially created data sets with and without noise, and on one experimentally obtained data set.

5.1 Example 1: *in silico* endopeptidase data without noise

We first show a trivial example with an exact solution. A toy data set was generated that mimics the digestion by a real protease.

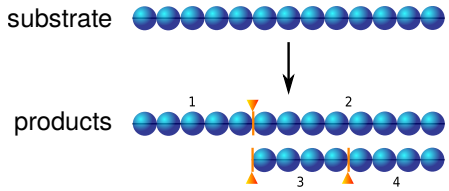


Figure 5.1: **Schematic of the toy protease.** Each blue circle symbolizes an amino acid. The orange triangles illustrate the cleavage sites of the protease. The resulting four peptide products are numbered accordingly.

A substrate of length 13 amino acids is digested by an endopeptidase, that can further cleave generated products. It first cleaves the substrate into peptides 1 and 2, and then cleaves peptide 2 resulting in two more products 3 and 4.

The files needed for this example can be found in the folder examples/toy_endo3_nonoise.

5.1.1 Construction of the data

To create the numeric data, we assume that all products of the digestion are detected. We simulate the concentrations of the substrate and the four products over time according to the rules, i.e. the sum of the amounts of amino acids at a given time must sum up to the amount of substrate in the beginning. We divided the concentrations by preset conversion factors:

reagent	factor
substrate	1
peptide 1	0.01
peptide 2	0.1
peptide 3	0.5
peptide 4	0.99

The resulting signal intensities (see formula (2.1.1)) were multiplied by 10^7 to get to a more realistic order of magnitude. This data is saved in the file **endo3_nonoise.csv**:

```
"sequence",0,1,2,3,4
"ABCDEFGHIJKLM",1000000000,800000000,700000000,650000000,600000000
"ABCDE",0,20000000000,30000000000,35000000000,40000000000
"FGHIJKLM",0,400000000,2000000000,1500000000,1500000000
"FGHI",0,320000000,200000000,400000000,500000000
"JKLM",0,161616161.616162,101010101.010101,202020202.020202,252525252.525253
```

Note that it has the format required, see Section 2.3.1. The header specifies the time points in hours in an increasing order i.e. left to right is from time 0 to time $t > 0$. The first column contains the sequences of the substrate and all the peptide products. Subsequent columns show the mass spectrometry intensities for each sequence over time.

5.1.2 Input

For this simple example, we use all default settings in the input.txt (see Section 2.3.1). The only mandatory input without default is the mechanism of the protease, which in this case is an endopeptidase. Since no titration is provided, we dont need to specify how much substrate was loaded. This input is ignored by the parser.

```
########## MANDATORY INPUT

# mechanism of the underlying enzyme: 'endopep', 'exopep'
enzyme: endopep

# amount of substrate loaded for kinetics (if titration provided)
loaded: 100
```

We want to compare successive time points (default), so the file timepoints.txt does not need to be provided. For this artificial example we don't have any titration data.

5.1.3 Execution

Let's go through the steps of Chapter 4.

1. If you haven't done that already, install R and the required R packages now (see Chapter 3).

2. The examples in the QPuB-master folder are already in the right format.

3. Open the terminal.

4. In the following, it is assumed that the folder QPuB-master is allocated in your Downloads folder. If not, please change the paths in the commands accordingly. We choose the working directory to be the QPuB folder:

```
$ cd Downloads/QPuB-master/QPuB
```

5. Execute the runQPuB.r script with the desired input folder:

```
$ Rscript runQPuB.r -infol examples/toy_nonoise
```

If you want the output folder to have a specific name, add the flag `-outfol my_fancy_name`.

5.1.4 Output

Once executed, information on the QPuB directory and the input folder used is prompted in the command window:

```
Path of QPuB: /home/shenze/Downloads/QPuB-master/QPuB
Path of input folder: /home/shenze/Downloads/QPuB-master/toy_nonoise

CREATING OUTPUT DIRECTORY...
Path of output folder: /home/shenze/Downloads/QPuB-master/QPuB-Output/OUT_toy_nonoise
Do not change folder name during run!

Terminal output redirected to runQPuB_ROUT.txt.
```

A general output folder QPuB-Output will now be created in the examples folder. Here you can find all the outputs of all the runs we will execute in the following. By default, the output folder OUT_toy_nonoise is created (or the folder with the user defined name my_fancy_name). All output regarding this run can be found here.

The ROUT-file: The progress report can be inspected in runQPuB_ROUT.txt. First the input parameters, signal intensity data, timepoints and titration used are printed and information about the data manipulation is given:

```
READING INPUT PARAMETERS...
                     V1       V2
1                enzyme  endopep
2                loaded      100
3            plot_chain     TRUE
4        plot_residuals     TRUE
5             freq_plot    10000
6               plotdet     TRUE
7               vecplot    FALSE
8             freq_save    10000
9             scaleprod     TRUE
10           filterprod    FALSE
11            threshold      1e6
12           conv_lower     1e-4
13           conv_upper      1e4
14         sigma_lower     1e-4
15         sigma_upper      1e4
16          conv_start        1
17         sigma_start       10
18           pun_start      0.1
19          OptAccRate    0.234
20               Track     TRUE
21        GammaExponent      0.5
22             burnFrac      0.5
23            burnGibbs     1000
24            thinGibbs       20
25 TerminationCriterion     TRUE
26              MaxIter    10000
27              ESSiter    10000
28            ConfLevel     0.05
29            Tolerance     0.05

READING INPUT DATA...
[[1]]
[[1]][[1]]
       sequence          0           1           2           3           4
1 ABCDEFGHIJKLM 1000000000   800000000   700000000   650000000   600000000
2         ABCDE          0 20000000000 30000000000 35000000000 40000000000
3      FGHIJKLM          0   400000000  2000000000  1500000000  1500000000
4          FGHI          0   320000000   200000000   400000000   500000000
5          JKLM          0   161616162   101010101   202020202   252525253


SETTING TIMEPOINTS FOR COMPARISON...
Setting timepoints to default: comparing successive timepoints.
```

```
     [,1] [,2]
[1,]    0    1
[2,]    1    2
[3,]    2    3
[4,]    3    4

READING INPUT TITRATION...
No titration file given.

PREPARING DATA...
Data scaled by 1e-08 to avoid large orders of magnitude.
Products scaled by 0.0209415394228141 to match order of magnitude of the substrate.
```

Note that the data is rounded for the print command, but the exact values are used in the algorithm! As the simulation progresses the ROUT-file is constantly being updated:

```
Convergence achieved when effective sample size > 8708

RUNNING INFERENCE...
Starting time: 2019-07-25 17:01:00
Iteration: 1000
Acceptance rate: 0.235
Iteration: 2000
Acceptance rate: 0.244
Iteration: 3000
Acceptance rate: 0.237
Iteration: 4000
Acceptance rate: 0.23875
Iteration: 5000
Acceptance rate: 0.2338
Iteration: 6000
Acceptance rate: 0.231833333333333
Iteration: 7000
Acceptance rate: 0.231285714285714
Iteration: 8000
Acceptance rate: 0.230625
Iteration: 9000
Acceptance rate: 0.23
Iteration: 10000
Acceptance rate: 0.2305
Effective sample size:  603.08807349756
⋮
Iteration: 26000
Acceptance rate: 0.231192307692308
Effective sample size:  8334.19764449216
Iteration: 27000
Acceptance rate: 0.231592592592593
Effective sample size:  9827.97921636141

Stop run: Chain converged.

Finishing time: 2019-07-25 17:01:35
```

```
Total running time: 35.535 sec elapsed
```

After 10000 iterations, the algorithm starts checking for convergence every 1000 iterations. The simulation terminates once the effective sample size reaches the minimum value which here is 8708. In this run, it took 27000 iterations or 35 seconds to converge.

```
PRODUCING SUMMARY OF RESULTS...
        Statistical summary in statistics.csv
        Distributions of conversion factors in summary.pdf

CALCULATING ABSOLUTE CONCENTRATIONS...
WARNING: No substrate titration provided. Only calculating normalized signals!
        Numeric values in folder 'concentrations'.
        Kinetic plots in folder 'plots_concentrations'.

THE END.
```

After termination, the summary of parameters and the absolute concentrations are produced. Note that in this simple example, only normalized signals are returned, since we did not provide titration data.

The diagnostic plots: The performance of the algorithm can be checked in the diagnostic plots. The mass deviation of the QPuB solution to the real solution can be investigated in the residuals. Note the improvement of the residuals from before starting QPuB just using the startvalues 1:

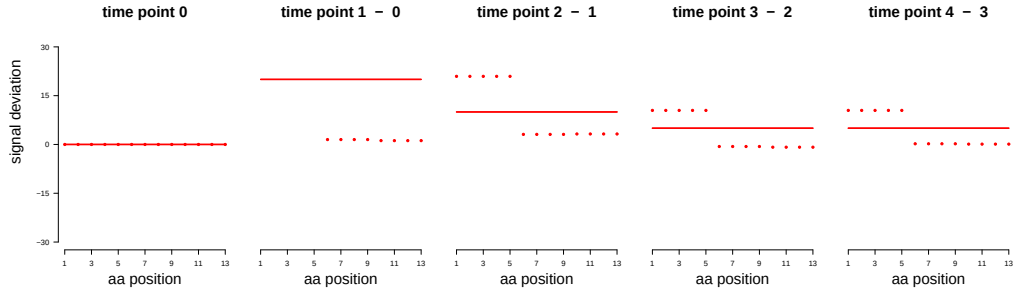


Figure 5.2: Residuals using startvalues (residuals_0.pdf)

to the residuals after termination of QPuB using the estimated values:

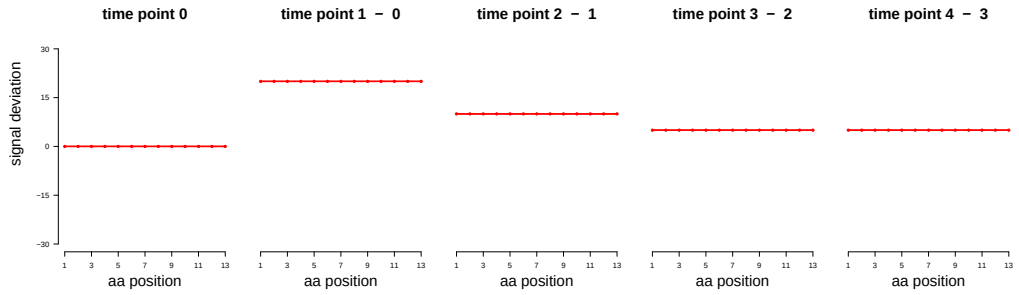
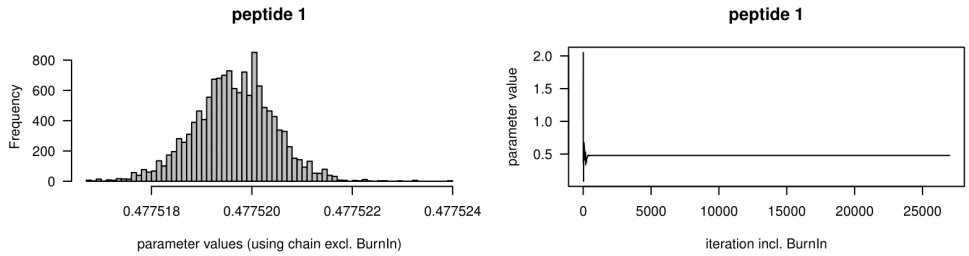


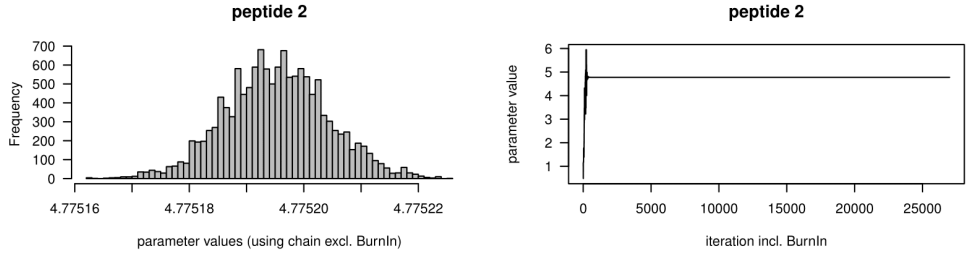
Figure 5.3: Residuals using estimated values (residuals_27000.pdf)

The horizontal line is the amount of substrate degraded between one time point and the preceeding and represents the real solution. This should coincide with the amount of each amino acid in the products, depicted as boxplots which are returned from QPuB. Here they are very narrow and coincide exactly with the real solution. We don't have any mass gain or mass loss.

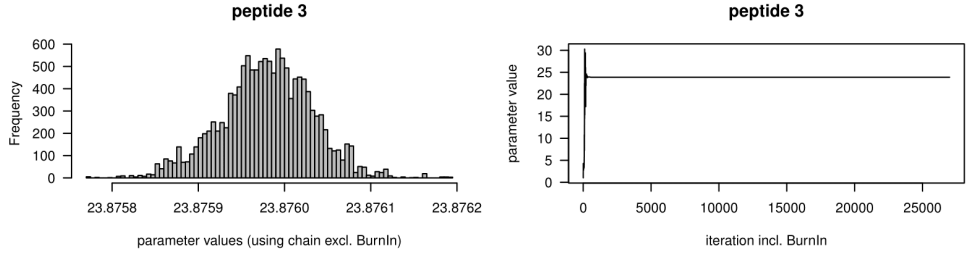
In the chain-plots we see the distribution of the parameter values in the first column and the trace of the Marcov chain in the second column:


peptide 1
Frequency
0.477518
0.477520
0.477522
0.477524
parameter values (using chain excl. BurnIn)

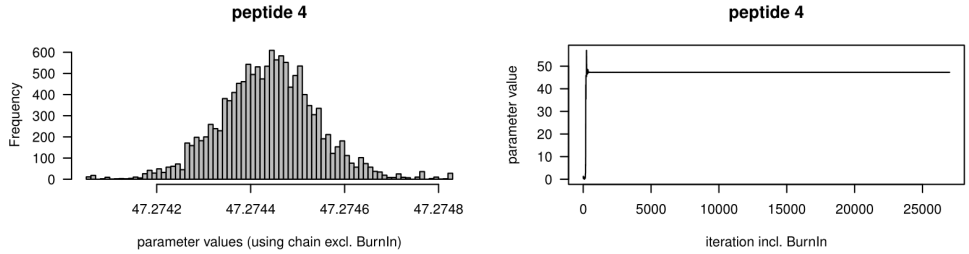
peptide 1
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn


peptide 2
Frequency
4.77516
4.77518
4.77520
4.77522
parameter values (using chain excl. BurnIn)

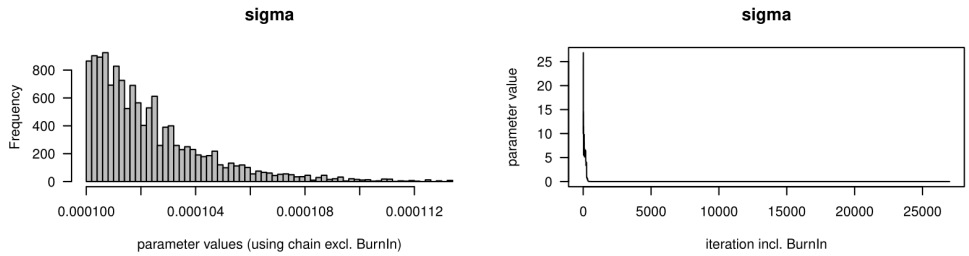
peptide 2
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn


peptide 3
Frequency
23.8758
23.8759
23.8760
23.8761
23.8762
parameter values (using chain excl. BurnIn)

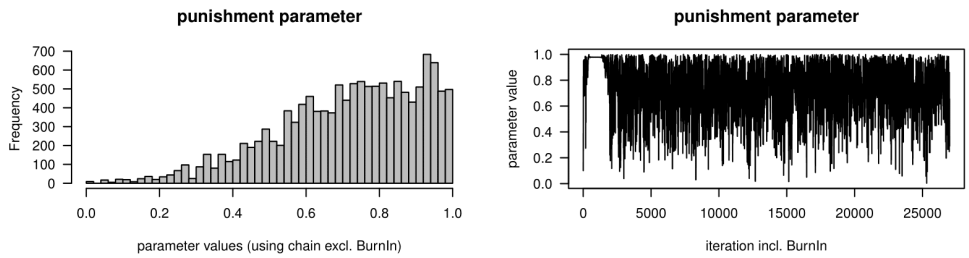
peptide 3
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn


peptide 4
Frequency
47.2742
47.2744
47.2746
47.2748
parameter values (using chain excl. BurnIn)

peptide 4
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn


sigma
Frequency
0.000100
0.000104
0.000108
0.000112
parameter values (using chain excl. BurnIn)

sigma
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn

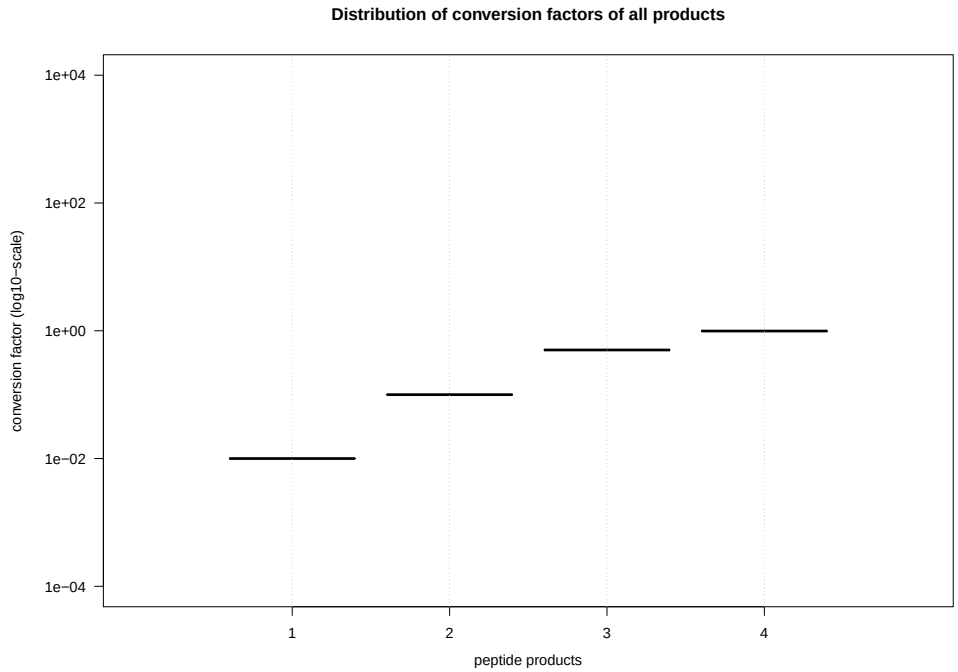

punishment parameter
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
parameter values (using chain excl. BurnIn)

punishment parameter
parameter value
0
5000
10000
15000
20000
25000
iteration incl. BurnIn

The conversion factors: By the way the data was constructed, it is easy to control, if the algorithm achieves to correctly infer the "unknown" conversion factors, which here are $(0.01, 0.1, 0.5, 0.99)$. Since there is no noise, with the help of QPuB, these conversion factors should be retained almost exactly from the intensity measurements. Note that the chain-plots are created using the scaled chain. Therefore here the values are not correct yet. For the file statistics.csv the chain has been backscaled. We see that the distributions have a very narrow spread and the medians equal the exact values almost perfectly:

parameter	min	0.05-quart.	median	0.95-quart.	max	mean	std
ABCDE	1.00E-02	1.00E-02	1.00E-02	1.00E-02	1.00E-02	1.00E-02	1.66E-08
FGHIJKLM	1.00E-01	1.00E-01	1.00E-01	1.00E-01	1.00E-01	1.00E-01	1.88E-07
FGHI	5.00E-01	5.00E-01	5.00E-01	5.00E-01	5.00E-01	5.00E-01	1.10E-06
JKLM	9.90E-01	9.90E-01	9.90E-01	9.90E-01	9.90E-01	9.90E-01	2.16E-06
sigma	1.00E-04	1.01E-04	1.02E-04	1.03E-04	1.07E-04	1.02E-04	2.11E-06
pun	1.70E-01	5.90E-01	7.43E-01	8.73E-01	1.00E+00	7.15E-01	1.96E-01

Note that the numeric values in the file are given with all digits and not rounded as shown here. The same is observed graphically in statistics.pdf, which contains the distributions of the conversion factor estimates (y-axis in log-scale):



Comparing the input signals and the output normalized signals: The folder plots_inputsignals contails the signal intensity kinetics and the folder plots_concentrations contains the resulting concentration kinetics, or normalized signal kinetics, when no titration is provided.

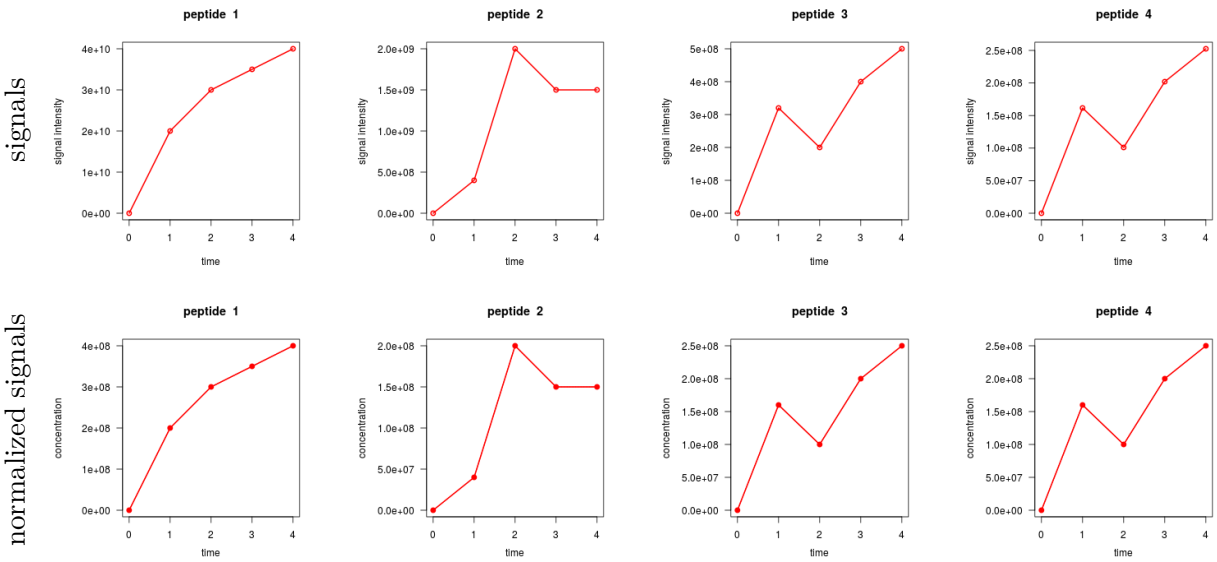


Figure 5.5: Kinetics of the input signal intensities and the output normalized signal intensities.

Because the conversion factor distributions have such a very narrow spread, the normalized signals seem to be single lines instead of shaded areas.
Note that when comparing the peptides based on their signal intensities, peptide 2 is more abundant than peptide 3 and 4. When comparing them based on their normalized signal intensities on the other hand, peptide 3 and 4 are more abundant.

5.2 Example 2: *in silico* endopeptidase data with noise

In reality, MS data is noisy. In the next example, we use the same toy endopeptidase as in Example 1 but a different set of conversion factors to create the data. In addition, we add multiplicative noise.

5.2.1 Description of the data

The set of conversion factors used has a bigger range [0,100], which corresponds better to the new Orbitrap data:

reagent	factor
substrate	1
peptide 1	0.01
peptide 2	1
peptide 3	10
peptide 4	100

To the resulting signal intensities, we added multiplicative noise with a standard deviation of 0.2:

$$s_i = \frac{c_i}{v_i} * 10^7 * \nu \quad \text{where } \nu \sim \mathcal{N}(1, 0.2) \tag{5.2.1}$$

MS data usually comes in at least duplicates of biological and/or technical replicates. In this example, we have two replicates saved as **toy_noise_1.csv** and **toy_noise_2.csv**:

```
"sequence",0,1,2,3,4
"ABCDEFGHIJKLM",1329645593.90514,568892651.666362,739675212.686849,540034688.5801,657770688.171482
"ABCDE",0,19550437927.5625,35093569014.2052,26785506340.1762,39645400838.1994
"FGHIJKLM",0,50803398.6346583,175111773.541086,129201866.259695,171923799.737804
"FGHI",0,19107329.1597266,11876738.232878,18509229.8771212,27264645.4799232
"JKLM",0,1516140.01373127,528357.666760176,1486536.72037556,2002493.27216031
```

```
"sequence",0,1,2,3,4
"ABCDEFGHIJKLM",819141294.60402,760543309.443102,768562309.0676,751308648.361924,565482583.256731
"ABCDE",0,24837882034.8424,18057674163.491,48865876487.9947,27284340396.4229
"FGHIJKLM",0,35746926.3663144,200506865.840624,156332268.517916,135109719.872524
"FGHI",0,15846649.3284839,9457129.03488233,21181638.8346666,31924244.7741753
"JKLM",0,1315410.61255498,713261.520152447,2069670.89032181,1297700.25667272
```

5.2.2 Input

The input specification is kept the same as that of the previous example without noise.

5.2.3 Execution

The steps are analogous to Example 1 following the general usage routine as described in Chapter 4. For the execution we use the input folder toy_noise and the output folder OUT_toy_noise inside the general output folder is created:

```
$ Rscript runQPuB.r -infol examples/toy_noise

Path of QPuB: /home/shenze/Downloads/QPuB-master/QPuB
Path of input folder: /home/shenze/Downloads/QPuB-master/toy_noise

CREATING OUTPUT DIRECTORY...
Path of output folder: /home/shenze/Downloads/QPuB-master/QPuB-Output/OUT_toy_noise
Do not change folder name during run!

Terminal output redirected to runQPuB_ROUT.txt.
```

5.2.4 Output

In this run, convergence was achieved after 284000 iterations, which took roughly 20 minutes. Again, QPuB achieved an improvement in mass conservation when comparing the residuals using the startvalues 1:

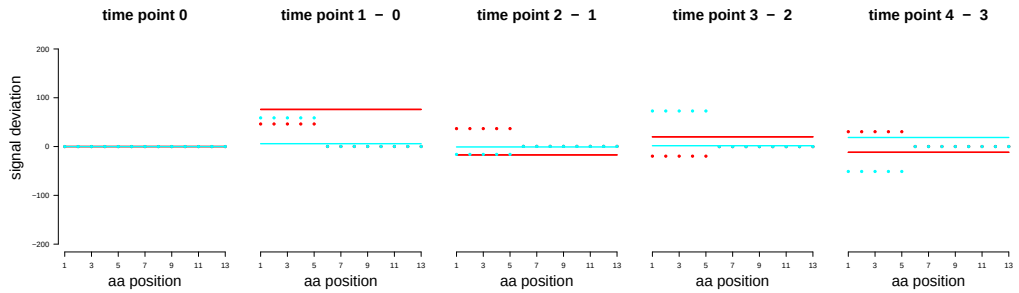


Figure 5.6: Residuals using startvalues (residuals_0.pdf)

to the residuals after termination of QPuB using the estimated values:

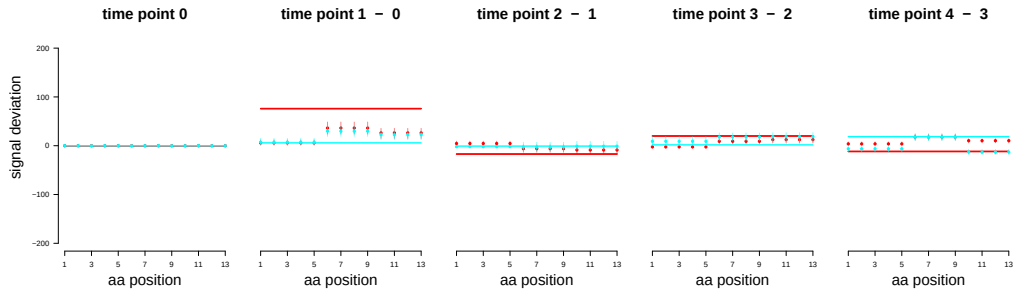


Figure 5.7: Residuals using estimated values (residuals_284000.pdf)

The two colors represent the two replicates. Again, the horizontal line is the signal of substrate degraded. The dot depicts the median of the distribution of the QPuB solution and the vertical line stretches from 5% to 95% quantile. During the QPuB run, the boxplots approach the vertical lines to minimize the mass deviation.
Note that for the noisy data, the distributions now are broader than in the case without noise:

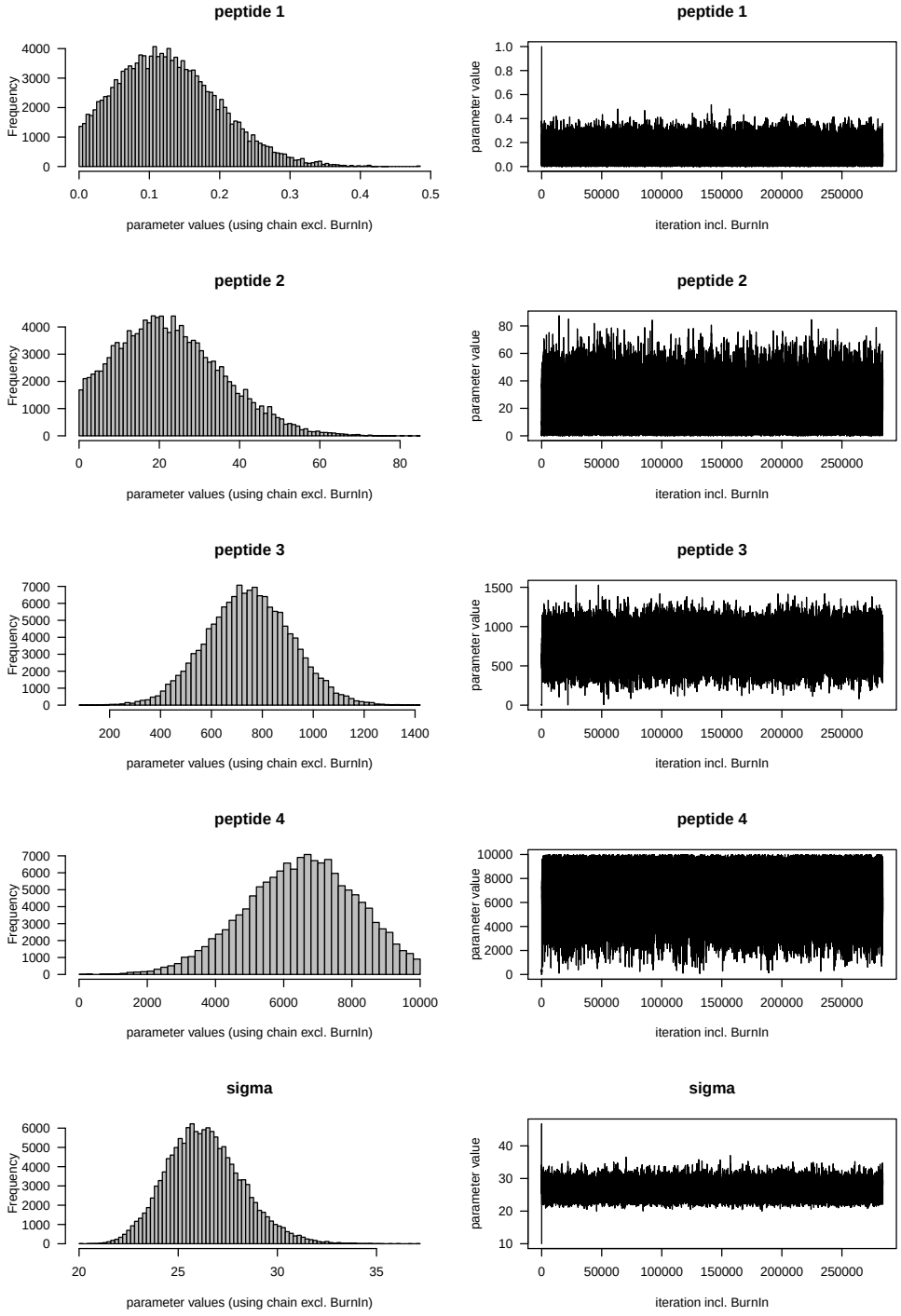
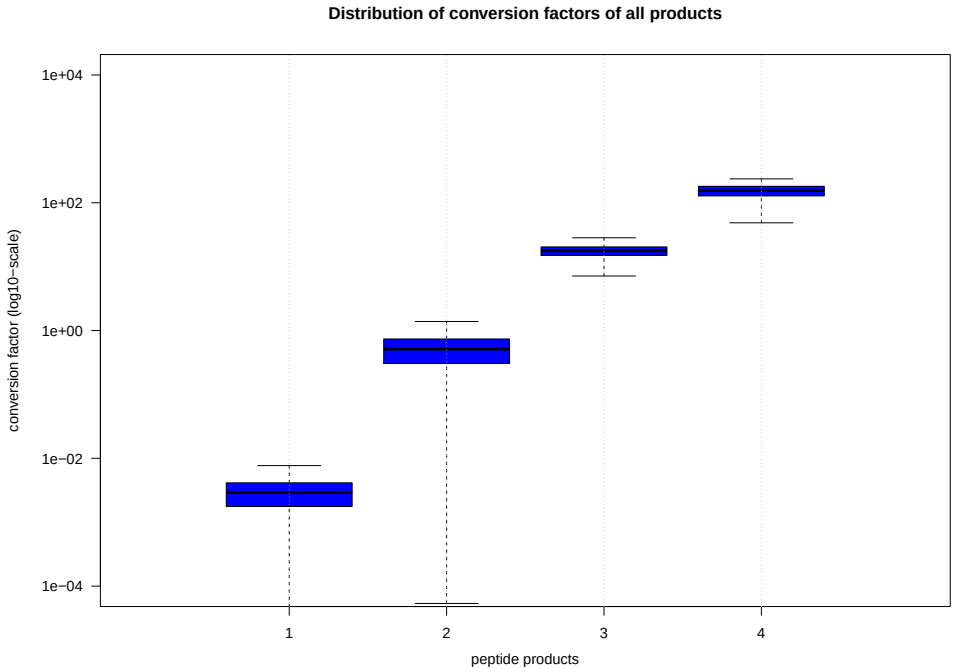


Figure 5.8: Distribution and trace of the Marcov chain after termination (chain_284000.pdf)

Again, the chain has to be backscaled before being able to validate the values with the preset values $(0.01, 1, 10, 100)$:



The noise in the data is now reflected in the kinetics of the amounts. We now clearly see the uncertainty in the form of shaded areas. We plot the region between 5% and 95% quantile surrounding the medians (solid line). Different colors represent the two replicates matching for the input and the output figures.

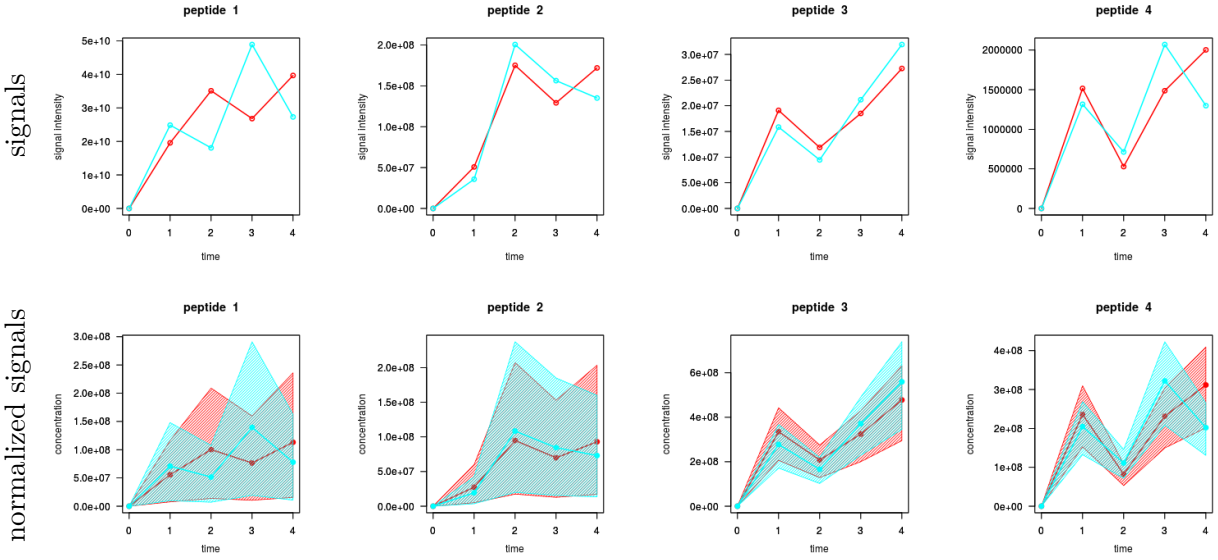


Figure 5.9: Kinetics of the input signal intensities and the output normalized signal intensities.

5.3 Example 3: Digestion of K386 by the proteasome

In this last example, we run QPuB on real data. We consider a proteasome digestion of the polypeptide K386. This data set is more complex than the toy data, with 23 products. Of course, the conversion factors are unknown. Due to data for more time points being available, we hope for good accuracy. Additionally, titration data is available and provided in the file **190423_K386_titration_substrate_charge_3.csv**. With this, QPuB will be able to return estimates of the absolute concentrations instead of only normalized signal intensities.

5.3.1 Description of the data

We performed measurements in two biological replicates with two technical replicates each. Here we show the data of replicate 1 (file **MB_rep1.csv**) representative.

```
"sequence",0,0.5,1,1.5,2,3,4
"RTKAWNRQLYPEW",889000000000,555000000000,275000000000,138000000000,83500000000,20100000000,33800000
"RTKAWNR",602000000,26800000000,36700000000,39200000000,47000000000,54100000000,41400000000
"RTKAWNRQ",55800000,640000000,925000000,888000000,1180000000,1230000000,777000000
"QLYPEW",11400000000,287000000000,430000000000,494000000000,555000000000,634000000000,514000000000
"RTKAWNRQL",10600000,243000000,390000000,423000000,454000000,399000000,291000000
"AWNRQLYPEW",70300000,559000000,465000000,326000000,239000000,71100000,19500000
"WNRQLYPEW",292000000,2030000000,1550000000,1060000000,827000000,342000000,79000000
"NRQLYPEW",36500000,290000000,313000000,232000000,264000000,207000000,139000000
"RQLYPEW",0,36800000,40000000,34700000,46000000,49200000,42500000
"YPEWRT",5090,0,587000,711000,490000,854000,567000
"ALYPEW",80000,3010000,4470000,4360000,5950000,6900000,5860000
"WNRQRTKAWNR",167,77900,157000,143000,143000,108000,55300
"WNRQQLYPEW",559000,0,2450000,1170000,3940000,2080000,1030000
"KAWNRQL",2640,75000,150000,188000,212000,254000,211000
"RTKAWNRQ",23100,411000,828000,820000,669000,820000,774000
"AQLYPEW",309000,3680000,6750000,9200000,9870000,13200000,15300000
"AWNYPEW",0,157000,300000,287000,294000,496000,450000
"QLYPEWNR",2960000000,26000000000,31100000000,22500000000,22900000000,19300000000,11700000000
"QLYPEWRTK",0,691000,629000,2100000,1700000,937000,1120000
"TKAWNR",7600,91700,184000,86400,180000,234000,191000
"AWNRQL",627000,10100000,31300000,31500000,82200000,56200000,62300000
"AWNRQLYQ",1.73,0,34500,139000,94800,268000,192000
"AWQLYPEW",58900,1010000,1100000,1220000,1430000,1690000,911000
"RQLYPYPE",4290000,15500000,17800000,21900000,26100000,24400000,18800000
```

5.3.2 Input

As mandatory input, we provide the mechanism of the protease: The proteasome is an endopeptidase. Since we provide a titration file, the amount of substrate loaded for the kinetics must be specified - In our case that was 200 pmol. The unit has to match the units in the titration file.

```
########## MANDATORY INPUT

# mechanism of the underlying enzyme: 'endopep', 'exopep'
enzyme: endopep

# amount of substrate loaded for kinetics (if titration provided)
loaded: 200
```

For a data set with this complexity, termination using the ESS criterion takes several days. We use a maximum iteration number of 10^5 instead, because the chains look already pretty converged:

```
### TERMINATION CRITERION
# TRUE: termination via effective sample size calculation.
# FALSE: enables the MaxIter variable, which is the maximum number of
# iterations the MCMC will run
TerminationCriterion: FALSE
MaxIter: 1e5
```

All other input stays as default for now. We will test different settings later (Section 5.3.5).

5.3.3 Execution

The steps are analogous to Example 1 and 2 following the general usage routine as described in Chapter 4. Since we want to use the titration data, we have to add another flag with the name of the file to the command line:

```
$ Rscript runQPuB.r -infol Prot_K386 -titr 190423_K386_titration_substrate_charge_3.csv

Path of QPuB: /Users/shenze/Downloads/QPuB-master/QPuB
Path of input folder: /Users/shenze/Downloads/QPuB-master/Prot_K386

CREATING OUTPUT DIRECTORY...
Path of output folder: /Users/shenze/Downloads/QPuB-master/QPuB-Output/OUT_Prot_K386
Do not change folder name during run!

Terminal output redirected to runQPuB_ROUT.txt.
```

5.3.4 Output

The ROUT-file: Although we have half time points (0.5h, 1.5h), the numbers in the time points used must be integers, numbering the six time points in the kinetics consecutively. For starters, we use the default of comparing successive time points. Later we will check, what changes when playing with the time points.

```
SETTING TIMEPOINTS FOR COMPARISON...
Setting timepoints to default: comparing successive timepoints.
     [,1] [,2]
[1,]    0    1
[2,]    1    2
[3,]    2    3
[4,]    3    4
[5,]    4    5
[6,]    5    6
```

The titration file provided using the flag -titr is read in:

```
READING INPUT TITRATION...
...from file 190423_K386_titration_substrate_charge_3.csv
   pmol         rep1         rep2
1   0.0 8.738783e+06 3.385014e+07
2   6.5 1.258559e+11 1.260661e+11
3  12.5 2.282229e+11 2.521959e+11
4  25.0 5.184945e+11 5.131968e+11
5  50.0 1.029977e+12 1.004578e+12
6  75.0 1.323855e+12 1.263028e+12
7 100.0 1.654654e+12 1.751555e+12

PREPARING DATA...
Data scaled by 1e-10 to avoid large orders of magnitude.
Products scaled by 1.67036496314502 to match order of magnitude of the substrate.
```

Termination: As preset, the run stops after 10^5 iterations, which took roughly 18 minutes:

```
Iteration: 1e+05
Acceptance rate: 0.23293

Stop run: Maximum number of iterations reached.

Finishing time: 2019-07-29 12:32:55
Total running time: 1114.477 sec elapsed
```

Diagnostic plots: Again, QPuB achieved an improvement in mass conservation when comparing the residuals using the startvalues 1:

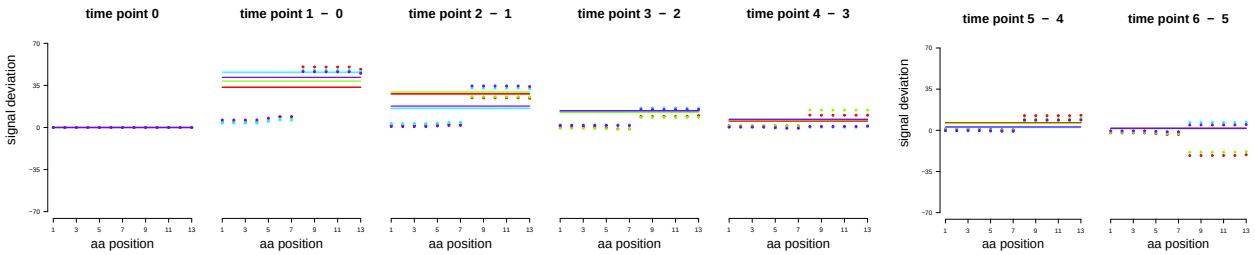


Figure 5.10: Residuals using startvalues (residuals_0.pdf)

to the residuals after termination of QPuB using the estimated values:

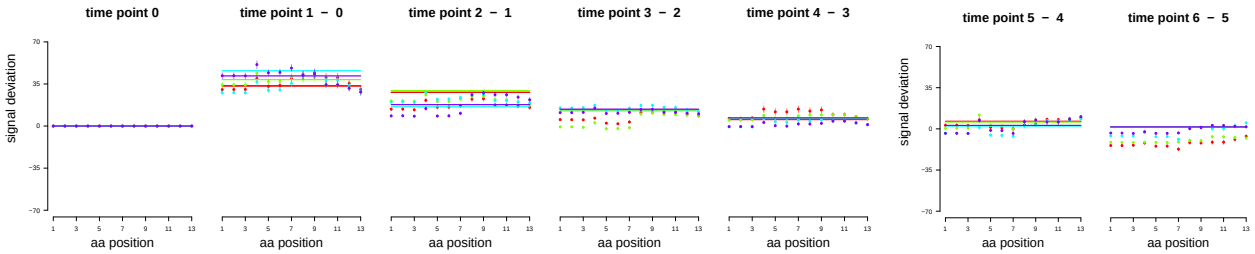


Figure 5.11: Residuals using estimated values (residuals_100000.pdf)

Here is a choice of parameter distributions:

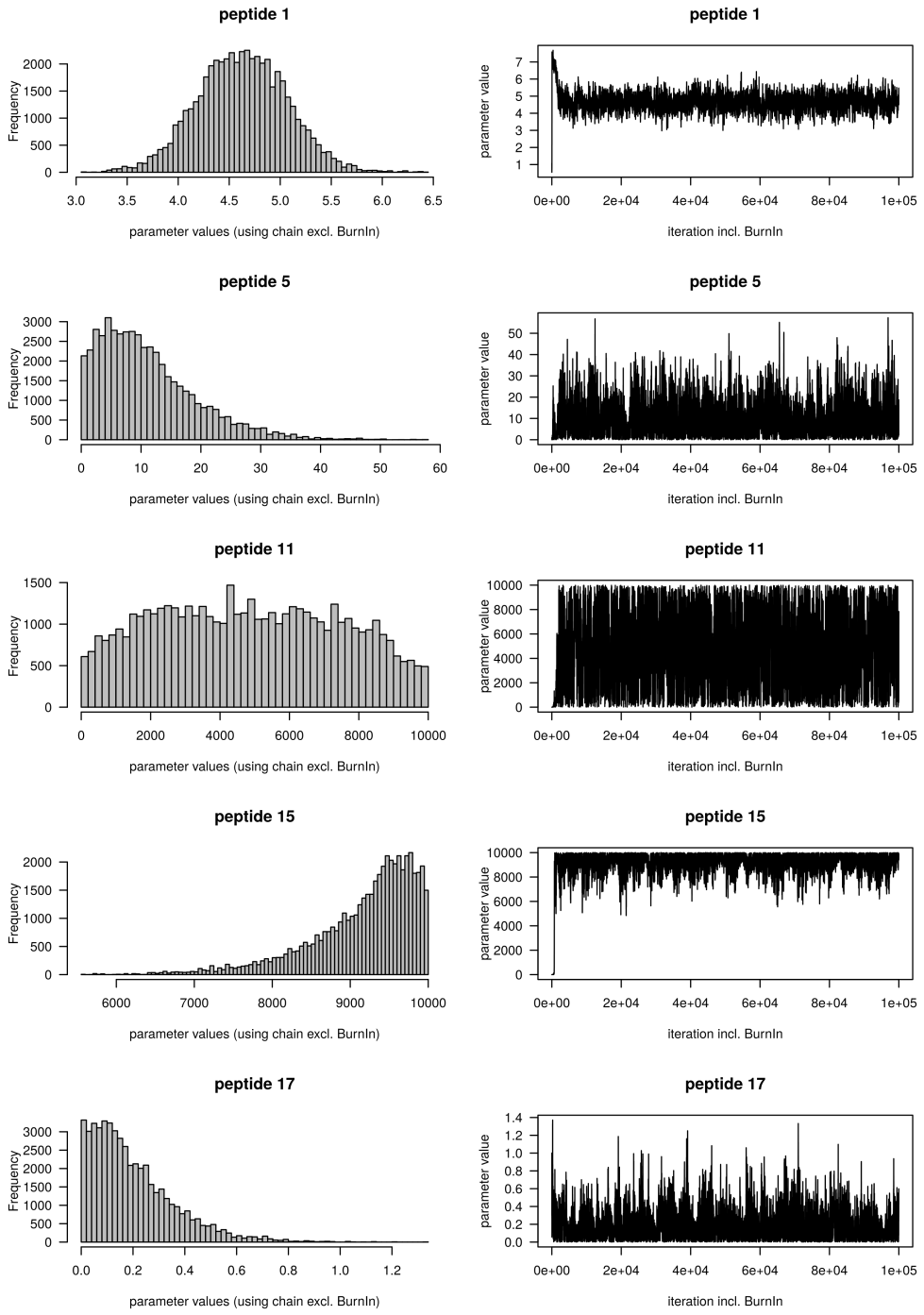
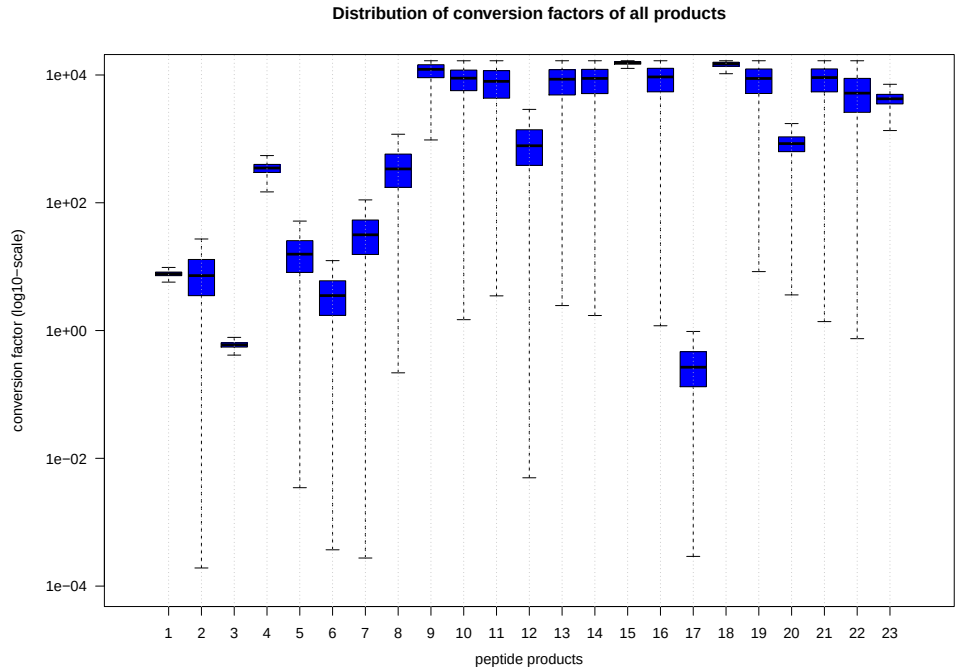


Figure 5.12: Distribution and trace of the Marcov chain of some peptides after termination (chain_100000.pdf)

Final results: Again, the chain has to be backscaled before being able to calculate the concentrations. The conversion factors that QPuB infers from the data are graphically shown in the file **statistics.pdf** with numeric values saved in **statistics.csv**:



Since we provided the titration data of the substrate, the absolute concentrations can be calculated from the input signals and not only the normalized signals are returned.

```
CALCULATING ABSOLUTE CONCENTRATIONS...
... using substrate titration from file 190423_K386_titration_substrate_charge_3_K386.csv
Linear fit of the substrate titration before normalization:
        intercept: 44479107527.5953, slope: 17042776518.3551
Using this fit, the signal for amount loaded 200 is 3453034411198.62
Linear fit of the substrate titration after normalization, mean over replicates:
        intercept: 11451356077.9449, slope: 4387743219.61028
Substrate and product amounts:
        Numeric values in folder 'concentrations'.
        Kinetic plots in folder 'plots_concentrations'.
```

Plots of the linear fits of the substrate titration before and after normalization are created. (Reminder: Normalization means modifying the titration data such that the signal of the amount of substrate loaded to the kinetics equals the signal at the beginning of the substrate signal kinetics.)

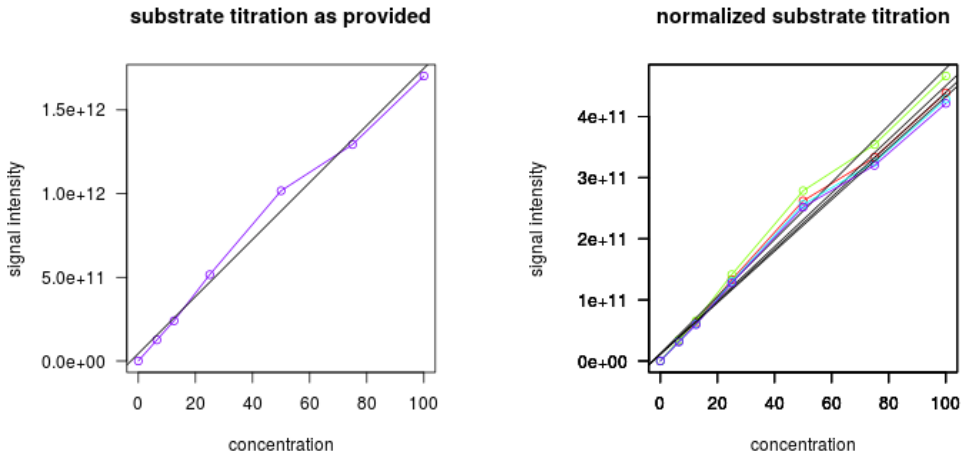


Figure 5.13: The files substratetitration_given.png and substratetitration_normalized.png.

For numeric values of the amounts of the substrate and the products, see folder concentrations. For graphical output see folder plots_concentrations. Here we only show 4 of the 23 products.

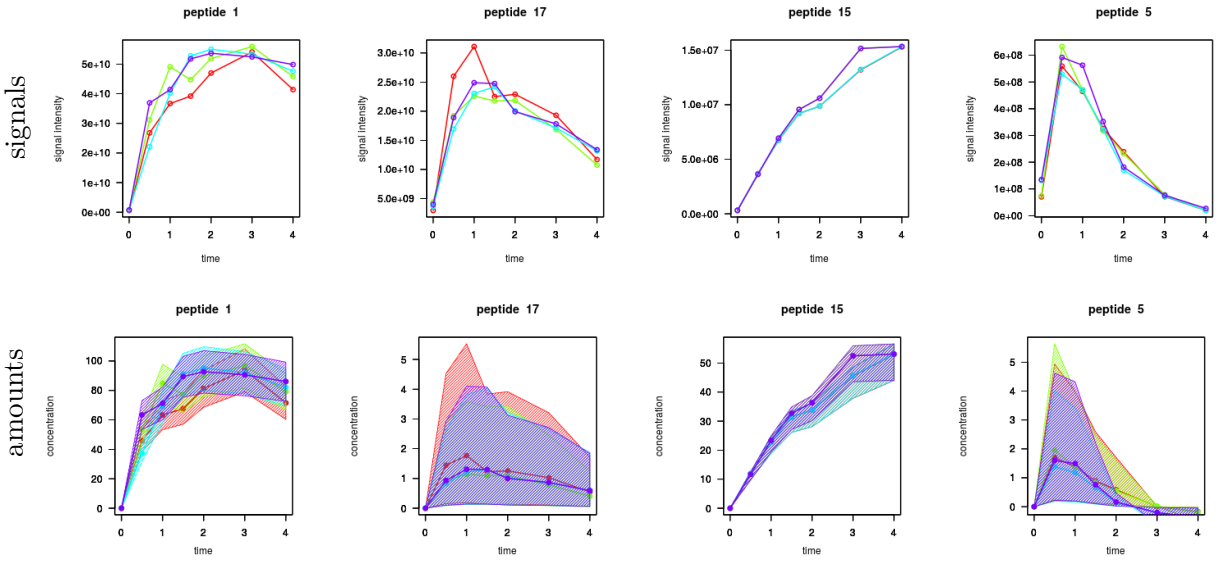


Figure 5.14: Kinetics of the input signal intensities and the output concentrations.

Peptide 1 is an example of a product with a high signal intensity and a resulting high concentration. Peptide 17 also has a high signal intensity but actually has quite a low abundance. Peptide 15 on the other hand has a very low signal intensity but a reasonably high amount. Note that if the signal at some time point $t > 0$ drops below the signal intensity of time $t = 0$, then the concentrations may become negative like for peptide 5.

5.3.5 Changing some of the Options

Some text...

Remove last time point

Since in the last time point (after 4h) the substrate is almost completely degraded and the difference between 3h and 4h is so small, that the effect size is too small compared to the level of noise in the data, we decide to remove the last time point from the calculation of the likelihood. This can be done by providing the file **timepoints.txt**:

```
0 1
1 2
2 3
4 5
```

Filter products

When inspecting the chain plots, note that some peptides can have a very broad distribution. As an example see peptide 11 in Fig. 5.12. Their signal intensity is so small that they don't contribute much to the mass balance and their conversion factors are hard to infer, since the level of noise in the data is bigger.

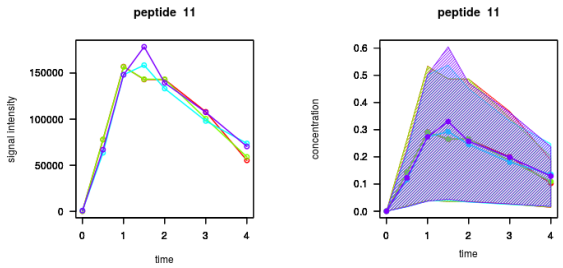


Figure 5.15: Signal intensities and absolute amount of peptide 11.

By activating the filtering option in the input.txt, these products are removed from the data set. We use the default threshold of 10^6 as the minimum signal intensity - all peptide products with a signal intensity below this threshold will be removed. This setting reduces the complexity of the data set by decreasing the number of products:

```
### DATA MANIPULATION
# Scale products to match order of magnitude of substrate
scaleprod: TRUE
# Filter out products which have a signal intensity below the threshold.
# If TRUE then threshold must be set.
filterprod: TRUE
threshold: 1e6
```

In this example, the dimension is reduced from 23 peptides to 16 peptides. Note that the number of a peptide in identifier.txt does not correspond to the number of the peptide in the unfiltered case!

Do not scale products to match order of magnitude of substrate

By default, QPuB scales the signal intensities of all peptide products to match the order of magnitude of the substrate. In this way the prior ranges are less dependent on the specific data set and therefore can be set to ... If you dont use this then set prior and starting values To illustrate the effect of this scaling option, we try a run without scaling:

```
### DATA MANIPULATION
# Scale products to match order of magnitude of substrate
scaleprod: FALSE
# Filter out products which have a signal intensity below the threshold.
# If TRUE then threshold must be set.
filterprod: FALSE
threshold: 1e6
```

Different Gamma exponent

For high dimensional data, we recommend adjusting the Gamma exponent used in the function to achieve vanishing adaptation. With a lower exponent, the adaptation vanishes later:

```
### ADVANCED ALGORITHM PARAMETERS
# optimal acceptance rate
OptAccRate: 0.234
# track performance of the run: print acceptance rate and effective sample size
Track: TRUE
# Exponent for the sequence to realize vanishing adaptation. Choose a value between 0 and 1.
GammaExponent: 0.01
# Fraction of the Markov chain to consider as burn-in, 0.25 means 25% burn-in
burnFrac: 0.5
```

6 Appendix

6.1 Bayesian statistical inference

Bayesian statistical inference deduce properties of an underlying probability distribution following the Bayes' rule, which tells us the formal way to update the probabilities based on new information. For a typical parameter estimation problem, assuming θ is the set of parameters, M denotes the models, and d denotes the data set, the Bayes' rule states that

$$\mathbb{P}(\theta|d, M) = \frac{\mathbb{P}(d|\theta, H)\mathbb{P}(\theta|M)}{\mathbb{P}(d|M)}, \tag{6.1.1}$$

where

- $\mathbb{P}(\theta) := \mathbb{P}(\theta|d, M)$ is called the **posterior probability**.
- $\mathcal{L}(\theta) := \mathbb{P}(d|\theta, M)$ is called the **likelihood**.
- $\pi(\theta) := \mathbb{P}(\theta|M)$ is called the **prior probability**.
- $z := \mathbb{P}(d|M)$ is called the **marginal likelihood probability** and expressed as

$$z = \int \mathcal{L}(\theta)\pi(\theta)d\theta. \tag{6.1.2}$$

The normalization constant z makes the $\mathbb{P}(\theta)$ a genuine probability density for θ. For model comparison, this constant is immensely important as for example, the relative posterior probability between two models M_1 and M_2 is expressed as

$$\frac{\mathbb{P}(M_1|d)}{\mathbb{P}(M_2|d)} = \frac{\pi(M_1)\mathbb{P}(d|M_1)}{\pi(M_2)\mathbb{P}(d|M_2)} = \frac{\pi(M_1)z_{M_1}}{\pi(M_2)z_{M_2}}. \tag{6.1.3}$$

Unfortunately, for a high-dimensional problem, the integral Eq. (6.1.2) is difficult to solve analytically and can even become computationally intractable. In such scenarios, we rely on Markov chain Monte Carlo (MCMC) methods.

6.2 Markov chain Monte Carlo

The basic idea of an MCMC algorithm is to learn the posterior distribution $\mathbb{P}(\theta)$ by first initiating a Markov chain of parameters whose equilibrium distribution corresponds to $\mathbb{P}(\theta)$, and then approximating the desired distribution by producing enough samples of θ using Monte Carlo sampling techniques.
Among various MCMC algorithms, two extremely popular ones include the Metropolis-Hastings (M-H) algorithm (Metropolis et al., 1953; Hastings, 1970) and the Gibbs sampler (Geman and Geman, 1987).

Metropolis-Hastings algorithm

Assuming $\mathbb{P}(\theta)$ is defined on a state space $\mathcal{X}_\theta$, the M-H algorithm proposes a geometric way to construct a Markov chain $\{X_n\}_{n\in\mathbb{N}}$ on $\mathcal{X}_\theta$ that is ergodic(statistical property can be deduced from a single and sufficiently long random sample) and stationary (irreducible) with respect to $\mathbb{P}(\theta)$, which means if $X_n \sim \mathbb{P}(\theta)$, then $X_{n+1} \sim \mathbb{P}(\theta)$. Algorithmically, the M-H sampler can be written in the following way (Robert and Casella, 1999):

Given X_0, $n = 0$;
while $X_n \nsim \mathbb{P}(\theta)$ **do**
  Propose a candidate $Y_n \sim q(y|X_n)$;

$$X_{n+1} = \begin{cases} Y_n & \text{with probability } \rho(X_n, Y_n) \\ X_n & \text{with probability } 1 - \rho(X_n, Y_n) \end{cases}$$

  where,

$$\rho(x, y) = \min\left\{\frac{\mathbb{P}(y)q(y|x)}{\mathbb{P}(x)q(x|y)}, 1\right\}$$

  $n = n + 1$
end

Algorithm 1: The Metropolis-Hastings algorithm

In QPuB, we assumed a symmetric proposal distribution i.e. $q(y|x) = q(x|y)$, and therefore used the Metropolis algorithm (Metropolis and Ulam, 1949; Metropolis et al., 1953). For example, one typical choice of a symmetric q is the normal distribution

$$q_\sigma(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - x)^2\right),$$

where σ^2 is the variance of the proposed increments and defines the Markov transition probability. It has to be noted that for an optimal performance of an MCMC algorithm involving such variance parameter σ^2 associated with the Markov transition probability has to be tuned which leads to a class of adaptive version of MCMC algorithms (Andrieu and Thoms, 2008).

6.3 Adaptive Markov chain Monte Carlo

The goal of adaptive Markov chain Monte carlo (MCMC) methods is to fine-tune the proposal distribution for a specific target distribution. The adaptation step proceeds following a Robbins-Monroe recursion (Robbins and Monro, 1951), which is a stochastic approximation algorithm to find the root of a function whose values are not available directly. To put mathematically, assume $N(\theta)$ is a random variable whose measurement is available, and $\mathbb{E}[N(\theta)] = M(\theta)$, whose measurement is not available, then for a constant α the recursion will find the root of the equation $M(\theta) = \alpha$ using the following recursion

$$\theta_{n+1} = \theta_n - a_n(N(\theta_n) - \alpha), \tag{6.3.4}$$

where $\{a_n\}_{n=1,2,\cdots}$ are the sequence of positive step sizes satisfying the following conditions

$$\sum_{n=0}^{\infty} a_n = \infty \text{ and } \sum_{n=0}^{\infty} a_n^2 < \infty,$$

A suitable choice for a_n satisfying the above two conditions is the quantity $\frac{a}{n}$, for $a > 0$ as suggested by Robbins-Monro. Now, consider the observed random variable as $N(\theta, X)$, where X is some random effect. Then if we consider $M(\theta^*) = \min_\theta \mathbb{E}[N(\theta, X)]$, this is equivalent to find the root of the gradient of the $M(\theta) = 0$. Denoting $\nabla M(\theta)$ as $Q(\theta, X)$, we have the redefine the recursion Eq. (6.3.4) as:

$$\theta_{n+1} = \theta_n - a_{n+1} Q(\theta_n, X_{n+1}) \tag{6.3.5}$$

This is equivalent to the attempt to find a root to the mean field equation

$$h(\theta) = \int_{\mathbb{R}^d} Q(\theta, X)\mathbb{P}(X)dX$$

In adaptive Metropolis (AM) setting, $\theta_n = (\mu_n, \sigma_n^2)$. and considering $\mathbb{P}(X)$ has second moment, $h(\theta)$ has the unique root in $\left(\mu_{\mathbb{P}(X)}, \sigma^2_{\mathbb{P}(X)}\right)$. For multivariate case, the variance will be translated to the covariance matrix Σ

By construction, adaptive version of MCMC is not Markovian unless the stepsize a_{n+1} is a monotonically decreasing sequence as shown in Andrieu and Thoms (2008). In addition, the authors provide an adaptive strategy for the scaling factor for Σ, based on the same Robbins-Monro recursion. Assuming AM in a multivariate setting, updating steps for μ, Σ, and the scaling factor r after acceptance/rejection of a sample are as follows

$$\begin{aligned}
\log(r_{n+1}) &= \log(r_n) + a_{n+1}\Big(\rho(X_{n+1}, X_n), \rho^*\Big) \\
\mu_{n+1} &= \mu_n + a_{n+1}\Big(X_{n+1} - \mu_n\Big) \\
\Sigma_{n+1} &= \Sigma_n + a_{n+1}\Big((X_{n+1} - \mu_n)(X_{n+1} - \mu_n)^T - \Sigma_n\Big)
\end{aligned}$$

Rao-Blackwellization of AM algorithm

Rao-Blackwellization of M-H algorithm facilitates the finding of the unique unbiased estimators for μ and Σ with minimum most variance (Casella and Robert, 1996). Before the Rao-Blackwellization of the M-H algorithm, let us briefly review the Rao-Blackwell theorem and its significance to further motivate this improvement.

Consider X is a random vector representing the data and θ is the set of parameters defines the distribution of X. A statistics $S(X)$ is said to be *sufficient* if the conditional distribution of X given S i.e, $\mathbb{P}(X|S)$ is independent of θ. Furthermore, a statistic $T(X)$ is an unbiased estimator of $g(\theta)$. a function of θ, if $\mathbb{E}_\theta[T(X)] = g(\theta)$, then the Rao-Blackwell theorem says the following:

Theorem 1. *the variance of the conditional expectation* $\mathbb{E}[T(X)|S(X)]$ *i.e.* $\mathrm{Var}[\mathbb{E}[T(X)|S(X)]]$ *is upper bounded by the variance of the unbiased estimator* $T(X)$ *i.e.* $\mathrm{Var}[\mathbb{E}[T(X)|S(X)]] \leq \mathrm{Var}[T(X)]$, *and* $\mathbb{E}[T(X)|S(X)]$ *is unbiased.*

In the M-H algorithm, the acceptance-rejection step of the proposal Y_n can be rewritten as

$$\mathrm{if}(\mathcal{U}_n[0,1] \leq \rho(X_n, Y_n))\{\text{then accept } Y_n\}.$$

Now, using the indicator variable $\mathbb{I}$, the above condition can be reformulated as the following:

$$X_{n+1} := \mathbb{I}\{\mathcal{U}_n \leq \rho(X_n, Y_n)\}Y_n + \mathbb{I}\{\mathcal{U}_n > \rho(X_n, Y_n)\}X_n.$$

The expectation of X_{n+1} with respect to $\mathcal{U}_n$ conditional upon X_n and the proposed transition Y_n leads to

$$\mathbb{E}_{\mathcal{U}_n}[X_{n+1}|X_n, Y_n] := \overline{X_{n+1}} = \rho(X_n, Y_n)Y_n + (1 - \rho(X_n, Y_n))X_n. \tag{6.3.6}$$

Furthermore, if we consider the statistic $S(X_n)$ as X_n, then $S(X_n)$ is *linearly sufficient* (Witting, 1987). Moreover, $T_n(X) = X_n$ is unbiased (for large n) since $\mathbb{E}_\theta[T_n(X)] = E(X)$ (ignores all the points except the last one for any n). Therefore, according to Theorem 1, we have

$$\text{Var}[\mathbb{E}_{\mathcal{U}_n}[X_{n+1}|X_n, Y_n]] \leq \text{Var}[\mathbb{E}_{\mathcal{U}_n}[X_{n+1}]].$$

Applying the Rao-Blackwellised updates for μ and Σ we have

$$\begin{aligned} \mu_{n+1} &= \mu_n + \gamma_{n+1}(\bar{X}_{n+1} - \mu_n) \\ \Sigma_{n+1} &= \Sigma_n + \gamma_{n+1}[\overline{(X_{n+1} - \mu_n)(X_{n+1} - \mu_n)^T} - \Sigma_n] \end{aligned} \tag{6.3.7}$$

6.4 Truncated multivariate normal distribution

In many real life scenarios, support for the proposed samples are constrained. In that case, instead of the full distribution, we have to consider the *truncated* version of the distribution and draw sample from the same. Mathematically, this means we have to sample from the following probability distribution function (Wilhelm and Manjunath, 2010)

$$f(X, \mu, \Sigma, A, B) = \begin{cases} \dfrac{\exp\left\{\frac{1}{2}(X-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right\}}{\int_A^B \exp\left\{\frac{1}{2}(x-\mu)^T\Sigma^{-1}(X-\mu)\right\}dX} & \text{for } A \leq X \leq B \\ 0 & \text{otherwise,} \end{cases} \tag{6.4.8}$$

where, μ is the mean vector, and Σ is covariance matrix. In QPuB, we used the R routine **rtmvnorm** from the package **tmvtnorm** (Wilhelm and Manjunath, 2010) to generate proposals from a truncated multivariate normal distribution.

Bibliography

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and computing*, 18(4):343–373.

Bassani-Sternberg, M., Pletscher-Frankild, S., Jensen, L. J., and Mann, M. (2015). Mass spectrometry of human leukocyte antigen class I peptidomes reveals strong effects of protein abundance and turnover on antigen presentation. *Molecular & Cellular Proteomics*, 14(3):658–673.

Casella, G. and Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.

Cox, J., Hein, M. Y., Luber, C. A., Paron, I., Nagaraj, N., and Mann, M. (2014). Accurate proteome-wide label-free quantification by delayed normalization and maximal peptide ratio extraction, termed MaxLFQ. *Molecular & cellular proteomics*, 13(9):2513–2526.

de Graaf, E. L., Giansanti, P., Altelaar, A. M., and Heck, A. J. (2014). Single-step enrichment by Ti4+-IMAC and label-free quantitation enables in-depth monitoring of phosphorylation dynamics with high reproducibility and temporal resolution. *Molecular & Cellular Proteomics*, 13(9):2426–2434.

Geman, S. and Geman, D. (1987). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier.

Haario, H., Saksman, E., Tamminen, J., et al. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications.

Liepe, J., Marino, F., Sidney, J., Jeko, A., Bunting, D. E., Sette, A., Kloetzel, P. M., Stumpf, M. P., Heck, A. J., and Mishto, M. (2016). A large fraction of HLA class I ligands are proteasome-generated spliced peptides. *Science*, 354(6310):354–358.

Liepe, J., Sidney, J., Lorenz, F. K., Sette, A., and Mishto, M. (2019). Mapping the MHC Class I–Spliced Immunopeptidome of Cancer Cells. *Cancer immunology research*, 7(1):62–76.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341.

Mishto, M., Goede, A., Taube, K. T., Keller, C., Janek, K., Henklein, P., Niewienda, A., Kloss, A., Gohlke, S., Dahlmann, B., et al. (2012). Driving forces of proteasome-catalyzed peptide splicing in yeast and humans. *Molecular & Cellular Proteomics*, 11(10):1008–1023.

Peters, B., Janek, K., Kuckelkorn, U., and Holzhütter, H.-G. (2002). Assessment of proteasomal cleavage probabilities from kinetic analysis of time-dependent product formation. *Journal of molecular biology*, 318(3):847–862.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Robert, C. P. and Casella, G. (1999). The Metropolis—Hastings Algorithm. In *Monte Carlo Statistical Methods*, pages 231–283. Springer.

Roberts, G. O., Gelman, A., Gilks, W. R., et al. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The annals of applied probability*, 7(1):110–120.

Vats, D., Flegal, J. M., and Jones, G. L. (2019). Multivariate output analysis for Markov chain Monte Carlo. *Biometrika*, 106(2):321–337.

Wilhelm, S. and Manjunath, B. (2010). tmvtnorm: A package for the truncated multivariate normal distribution. *sigma*, 2(2).

Witting, T. (1987). The linear markov property in credibility theory. *ASTIN Bulletin: The Journal of the IAA*, 17(1):71–84.